Distributed Architectures for Mars Surface Exploration

by

Christopher E. Carr

B.S. Aeronautics and Astronautics
B.S. Electrical Science and Engineering
Massachusetts Institute of Technology, 1999

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September, 2001

Author .....
Department of Aeronautics and Astronautics
August 10, 2001

Certified by .....
Dava J. Newman
Associate Professor, MacVicar Faculty Fellow
Department of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

2nd copy

Distributed Architectures for Mars Surface Exploration

by

Christopher E. Carr

Submitted to the Department of Aeronautics and Astronautics on August 10, 2001, in partial fulfillment of the requirements for the degree of Master of Science in Aeronautics and Astronautics

Abstract

Surface-based distributed architectures may help future robotic and human Mars explorers cope with broad mission goals requiring significant mobility and long deployment periods. Surface-based distributed architectures can be characterized by connectivity and performance metrics such as the mean cost of message delivery. A trade study of connectivity for a lander and sensor pod network, based on a simple trade study process for distributed systems, demonstrates that the mean cost of message delivery is sensitive to small changes in surface topography statistics. To understand how human explorers can best utilize distributed systems, the use of information and communication was studied during a three-week geologic mapping project and through an analysis of nearly sixty thousand voice communications and anecdotal data from the Apollo lunar missions. Distributed systems can help human explorers share information in real-time and better coordinate their activities: A short-range wireless networking system would allow field geologists to pursue distributed geology activities, in which geologists might discuss a geologic feature from several different perspectives simultaneously. Subsequent modeling of line-of-sight connectivity during simulated geological traverses suggests that such a system might be feasible for team-member separation distances from tens to hundreds of meters. An approach to modeling the cost of node mobility is developed for human and robotic agents and used to develop the concept of the cost of system reconfiguration. Changes in the spatial configuration of distributed system elements are considered in terms of a collection of traverses from one point to another by different system elements. A process for traverse planning and execution, applicable to human and robotic explorers, is developed and applied to several situations in order to understand how distributed architectures can support traverse planning, and vice-versa.

Distributed systems fundamentally enhance the exploration process by changing the way in which information is collected and disseminated by exploration agents. This additional access to information, coupled with appropriate decision-making tools, enhances self-sufficiency and autonomy and allows individual explorers to enhance their probability of survival, coordinate their activities with other explorers, and increase the value they add to the overall mission. However, the design and optimization of distributed systems is a difficult process, and the benefits of distributed architectures come with the significant burden of added system complexity. Targeted deployment of distributed systems for planetary surface exploration can lead to increases in system performance, in terms of enhanced flexibility and robustness, when system complexity is adequately managed.

Thesis Supervisor: Professor Dava J. Newman
Title: Associate Professor of Aeronautics and Astronautics, MacVicar Faculty Fellow

Acknowledgments

This thesis is dedicated to two dear friends who taught me so much both in their lives, and in their deaths: James W. Wolkin (1977-1999) and Lowell H. Hovis (1929-2001).

It is only because of the many wonderful people in my life that I have completed this journey and emerged to behold the stars. I cannot possibly hope to thank all of the people who, directly or indirectly, have contributed to this project. To anyone whose name has not been mentioned below, I thank you.

To my advisor, Dava J. Newman: I thank you for the freedom you have given me to pursue my research, and for the guidance and support you have provided over the last two years. To my wonderful office-mate, Joe Saleh, thank you for your intellectual contributions on flexibility in system design, but more importantly, know that I am forever indebted to you for your friendship, encouragement, and all that you have taught me over the last year (I now keep a pencil on hand while reading). To my former office-mates Amir, Grant, Patricia, and Heiko, thank you for your patience, advice, and friendship. Thank you also to the other students, staff, and faculty of the Man-Vehicle Laboratory.

To my dear parents Dean and Kathy, and to my little brother Jonathan: How could anyone ask for a better family. I promise, someday all of the years of schooling will pay off.

Dingli: Thank you for your unconditional care, friendship, and support since we first met six years ago. Andrea: Thank you for your constant friendship, thesis advice, and poetic voice. Aditya: Thank you for your unwavering friendship these last six years, and many wonderful brainstorming sessions. Michal: Thank you for being a supportive friend and a wonderful person.

I am forever indebted to John Southard and Lindsay Schoenbohm for reintroducing me to geology, something I loved as a child but only recently began to explore again. Thanks also to Clark Burchfiel, Kip Hodges, and all of the geology students with whom I had the opportunity to spend a month in the Bird Spring Mountains of Nevada. I especially want to thank Lindsay and Ben for being patient teachers of geologic mapping in the field.

I must also gratefully acknowledge support for this work by the National Science Foundation Graduate Research Fellowship Program.

I have emulated the page layout of Edward R. Tufte, and encourage anyone not familiar with his work to read the three part series including *The Visual Display of Quantitative Information*. I make no claims to have faithfully executed visual displays in a manner consistent with his work, but I have tried to tell a story with text and images, and hope that the reader will take advantage of the margins to make use of whatever writing implements seem appropriate.

In considering the thesis as a whole, I am often reminded of the following quotation:

Errors using inadequate data are much less than those using no data at all.

Charles Babbage (1792-1871)

I have decided it was better to have written this thesis, than to have written nothing at all.

All of the aforementioned people, and many not mentioned, have collectively served as my guide on this adventure. Thank you again for a wonderful journey. For a moment, one must take in the view of a dark sky filled with glowing points of shining light, and then begin again.

E quindi uscimmo a riveder le stelle.

Dante Alighieri, *Inferno, Canto XXXIV, 140*, c. 1307.

Biographical Note

Christopher Carr was born in 1976 in Helsinki, Finland, as a citizen of the United States. He was raised in Seattle, Washington, and in 1995 was accepted to the Massachusetts Institute of Technology. He graduated in 1999 with degrees in Aeronautics and Astronautics, and Electrical Science and Engineering, and spent the following two summers working on a study for a Mars Sample Return mission at the Jet Propulsion Laboratory in Pasadena, California. He returned to MIT in the fall of 1999 to continue working with the MIT Man-Vehicle Laboratory as a graduate student in the Department of Aeronautics and Astronautics while pursuing his doctoral degree in the Medical Engineering and Medical Physics program in the Harvard-MIT Division of Health Sciences and Technology.

Table of Contents

List of Figures

List of Tables

Preface

The first human step on the Martian surface:
Inevitable Descent, by Pat Rawlings.

On a historic day, in the not-so-distant future, a human will set foot upon the surface of the planet Mars as part of a combined human and robotic effort to explore the Red Planet. We will go to Mars for many reasons. We will look for evidence of the possibility of present or past life on Mars, perform geological surveys, and look for subsurface water and other resources necessary for long-term human survival in the Martian environment. We will characterize the atmospheric, surface, and subsurface environments of the planet. We will develop hypotheses about the past history of Mars, and compare its history to that of the Earth. Ultimately, the exploration of Mars may result in the growth of a second branch of humanity, not (I am sure) as an expression of escapism, but as an expression of what it means to be human. We will go to Mars, and in doing so we will better understand what it means to be human and what it means to inhabit the beautiful planet called Earth. Many will argue as to whether humans should be sent to explore the harsh environments of space. It is clear that humans and robots have differing talents, but that humans can provide one thing that no current robot can approach: a human perspective and a human description of the experience of exploration. Humans on the Martian surface will observe, record, express, and communicate their experiences to the rest of humanity, not in competition with robotic explorers of the Red Planet, but as complementary elements of a cooperative venture. Observation, interpretation, and communication may be accomplished via autonomous robots or via remote operation of robots, but is also likely to involve a significant amount of direct contact with the Martian surface in the form of extravehicular activity.

This thesis attempts to apply the concept of distributed architectures to the challenges of planetary surface exploration, and to demonstrate some of the capabilities of distributed architectures for the surface exploration of Mars. It was my intent to demonstrate that similar modeling approaches can be used to model both human and machine components of a distributed system, and to emphasize the interactions between distributed system elements. If I have opened more doors than I have closed, but managed to close one or two, then I have succeeded in accomplishing what I set out to do: to tell a compelling story, supported in part by my research. It is the author's hope that future human planetary surface exploration will extend far beyond the Earth, Moon, and Mars. While the human and robotic exploration of Mars is the primary focus of this thesis, I hope that some of the material presented herein may find an application to future human planetary surface exploration well beyond the red dust of our nearest neighbor.

It is ironic that artist Pat Rawlings' depiction of the first human footstep on the Martian surface contains the word "inevitable." The human exploration of Mars, and any part of the universe, is anything but inevitable. The human and robotic exploration of space has been, and will continue to be, achieved only through the will and hard work of individuals with the curiosity and vision to persevere in times of light and dark to enrich and improve the human condition.

Ad astra.

Christopher E. Carr
May 11, 2001
Earth

We shall never cease from exploration
And the end of all our exploration
Will be to arrive where we started
And know the place for the first time.

T.S. Eliot, *The Four Quartets, Little Gidding, Verse V* (1943)

1 *Introduction*

In keeping with the immortal words of T.S. Eliot, this thesis begins with its destination:

Surface-based distributed architectures, and the elements of which they are composed, can be characterized by a quantifiable set of performance metrics including cost of transport of system elements and connectivity metrics. However, the design and optimization of distributed systems is a difficult process, and the benefits of distributed architectures come with the significant burden of added system complexity. Targeted deployment of distributed systems for planetary surface exploration can lead to increases in system performance, in terms of enhanced flexibility and robustness, when system complexity is adequately managed.

System architecture includes not only the structure of a system and its elements, but must also include the structure of the environment and the process required to build and operate the system [Rechtin, 1991]. Figure 1-1 illustrates the dual role of system architecture characterization in mission design and operations for planetary surface exploration missions. Understanding the advantages and disadvantages of distributed architectures for Mars surface exploration requires methods for characterization of distributed architectures: the rest of the thesis develops some methods of characterizing distributed architectures for surface exploration, and demonstrates how distributed architectures can be utilized to achieve self-sufficiency, autonomy, and group collaboration and coordination.

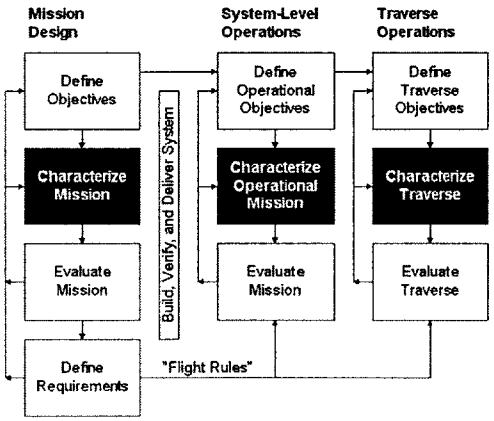


FIGURE 1-1. Mission architecture characterization is critical during space mission design and during mission operations at the system- and element-level. During surface exploration missions, day-to-day operations are typically broken down into "traverse" operations, in which one or more elements of a system cooperate to achieve a limited subset of overall mission goals while working within the constraints of a set of (typically heuristic) rules called "flight rules."

1.1 Motivation

A system for future human and robotic exploration of the Martian surface needs the capability to cope with challenges imposed by broad mission goals, a minimum surface stay in some mission architectures of more than 600 days [Hoffman et al., 1997], and the dynamic character and lack of knowledge of the Martian environment. Current planning for future exploration of Mars is hampered by:

- disproportionately little focus on the surface exploration phase in previous human Mars mission architectures,
- a lack of planetary extravehicular activity experience,

- under utilization of existing technologies and data to the art and science of exploration, and
- the lack of an evolutionary pathway that bridges the gap from robotic exploration to joint human-robotic exploration, and that incorporates cooperative strategies for future human and robotic explorers.

This thesis attempts to address these deficiencies by describing the exploration of the Martian surface as a distributed architecture of interacting elements both human and robotic. This description incorporates elements from theories of distributed systems, including multi-agent systems and distributed artificial intelligence. Analysis of previous extravehicular activity on the lunar surface and the author's personal experiences of field geology are also described, and a unified approach for traverse planning and execution for human and robotic explorers on the Martian surface is proposed. Mechanisms for how distributed architectures can be used to support extravehicular activity planning and execution on the Martian surface are proposed and explored.

1.2 Problem Statement

The primary goal of system engineering is to ensure that "the system is designed, built, and operated so that it accomplishes its purpose in the most cost-effective way possible, considering performance, cost, schedule, and risk [NASA SP-6105, 1995]." In the context of systems engineering, the crucial questions for this research effort are:

- Why utilize a distributed architecture for Mars surface exploration? What are the major trades for distributed architectures for Mars surface exploration?
- What element-level and system-level metrics are appropriate for characterizing the element-level and system-level performance of a distributed architecture for Mars surface exploration?
- What determines the behavior of individual elements of the distributed system, and how can the individual and collective behaviors of those elements be optimized to achieve mission goals?
- Specifically, how can a distributed architecture for Mars surface exploration support the planning and execution of extravehicular activity, and the collection, interpretation, and dissemination of information?

The primary driving question of this thesis can be stated as:

How can distributed architectures increase the likelihood of mission success by enhancing the flexibility and robustness of a human and/or robotic Mars surface exploration mission?

Figure 1-2 illustrates some of the major system architecting activities required to evaluate how distributed architectures can lead to flexible and robust systems. While cost is an important factor in the systems architecting and systems engineering processes, it will not be a major focus of this work.

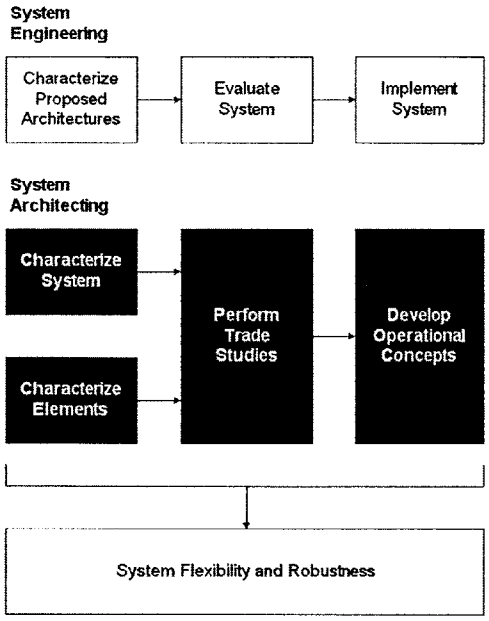


FIGURE 1-2. Architectures must be characterized at the system and element level, trades must be conducted, and operational concepts must be developed in order to demonstrate how distributed systems might achieve flexibility and robustness.

1.3 Research Objectives

The research objectives of this thesis are:

- to define major trades for distributed architectures for Mars surface exploration and demonstrate under what conditions these architectures might be desirable over alternatives,
- to develop a set of desirable features of distributed architectures for Mars surface exploration and to illustrate how some of these features might be achieved.
- to develop an basic operational concepts of how humans may explore the Martian surface by evaluating available data from the lunar surface operations portion of the Apollo missions and by developing an understanding of field geology through participation in a month-long field geologic mapping project,
- to demonstrate, through simple examples and models, how a distributed architecture for Mars surface exploration can enhance traverse planning and execution for elements of distributed systems, and
- to make recommendations for potential development of distributed architectures for Mars surface exploration and for further work.

The overall success criteria of these objectives is whether the collective results prove useful in evaluating system design alternatives for Martian surface exploration or demonstrating useful mechanisms for supporting future human and robotic explorers.

1.4 Overview of Methods

Several methods were utilized to examine the multiple facets of the research objectives:

- A review of the existing literature was conducted.

- An approach to characterizing surface-based distributed architectures was developed to demonstrate desirable characteristics of distributed architectures for planetary surface exploration.
- An observational study of field geology was undertaken during a month-long geologic mapping trip in the Bird Spring Mountains, Nevada.
- Apollo voice communication transcripts, together with other Apollo data, were analyzed in an attempt to develop an understand of the role of information and communication during planetary extravehicular activity. Metabolic cost data were studied to determine the relationships between topography and activity on the cost of transport during extravehicular activity.
- Simple examples were developed that demonstrate how a distributed architecture for Mars surface exploration can support extravehicular activity planning and execution.
- Recommendations for the development of future distributed systems for Mars surface exploration and for future work were developed, based on an integrated analysis of the other results.

1.5 Contributions

Contributions of this thesis include:

- Identification of major trades of distributed systems for planetary surface exploration,
- Development of a structured approach and a set of analysis tools for trade studies of distributed systems for planetary surface exploration,
- Development of a structured approach to traverse planning for human or robotic planetary surface explorers, and
- Recommendations for future distributed system development and operational concepts for future human planetary surface exploration.

1.6 Organization

Figure 1-3 provides a high-level thesis road-map. Individual chapters are described in more detail in the following paragraphs.

Chapter 1, *Introduction*, introduces and describes the motivation and objectives for this thesis.

Chapter 2, *Background*, provides background on distributed architectures, the evolution of extravehicular activity and exploration technologies, and discusses the exploration and environment of

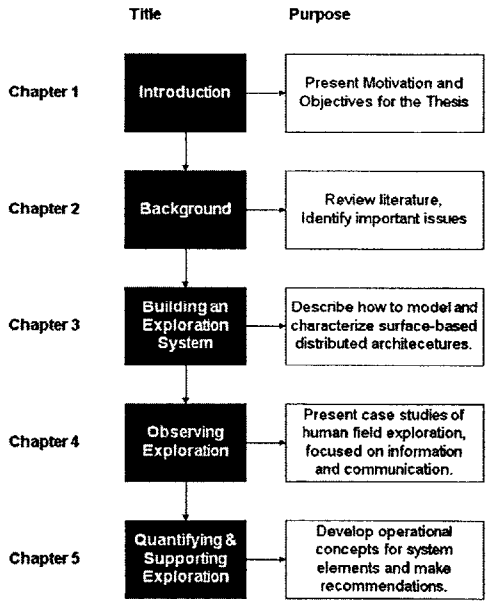


FIGURE 1-3. High Level Thesis Road-map: Readers short on time may want to quickly skim Chapter 2, and focus on the trade study process and sensor network example in the latter half of Chapter 3. Reviewing the discussion in the final section of Chapter 4 may also prove valuable before reading Chapter 5.

Mars. Chapter 2 also outlines an overall methodology for this thesis, and provides a brief introduction for the following three chapters.

Chapter 3, *Building an Exploration System*, describes the concept of an distributed architecture for Mars surface exploration in detail, including exploring the trade space of distributed architectures and illustrating desirable and undesirable features of a distributed architecture through simple examples.

Chapter 4, *Observing Exploration*, offers a case study of exploration as observed by the author during field geologic mapping and as recorded by voice communication transcripts and other data sources from the Apollo program. This chapter focuses on understanding how information and communication are used during field work and extravehicular activity, and on developing operational concepts that may be applicable to future human Mars surface exploration.

Chapters 5, *Quantifying and Supporting Exploration*, revisits the distributed architecture model developed in Chapter 3 with complexity added by incorporating the findings of Chapter 4. Modeling of metabolic cost and cost of transport is introduced. Operational concepts for supporting human and robotic system elements during traverse planning and execution are developed. Recommendations for future exploration system development and further work are also suggested.

1.7 References

Hoffman, S.J., and Kaplan, D.L., *Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team*, Lyndon B. Johnson Space Center, National Aeronautics and Space Administration, July 1997 (*http://spaceflight.nasa.gov/mars/reference/hem/hem1.html*).

NASA Systems Engineering Handbook, SP-6105, National Aeronautics and Space Administration, June, 1995.

Rechtin, Eberhardt, *Systems Architecting*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Wertz, J.R., and Larson, W.J., *Space Mission Analysis and Design, 2nd Edition*, Kluwer Academic Publishers, Norwell, Massachusetts, 1992.

It is the theory that decides what can be observed.

Everything should be made as simple as possible, but not simpler.

Albert Einstein

2 *Background*

This chapter provides the background necessary to describe a class of distributed architectures for Mars surface exploration. Distributed systems, human exploration, extravehicular activity, and past and future Mars exploration constitute the major themes covered.

The goals of this chapter are threefold:

1. to define a vocabulary for the thesis,
2. to review the existing literature and draw connections between the multiple themes of the thesis,
3. to discuss the methodology of the thesis.

Section 2.1 covers the definitions of basic concepts used throughout the thesis. Section 2.2 covers the theoretical development and terminology of multi-agent systems and distributed artificial intelligence, and serves as the base upon which the concept of an "exploration system" will be developed and applied to the exploration of the Martian surface. Existing analog distributed architectures are briefly described in Section 2.3; important similarities and differences are highlighted. Section 2.4 briefly reviews past human exploration and discusses how exploration agents (human and robotic) can be characterized. Section 2.5 briefly reviews some of the challenges of extravehicular activity for future planetary surface exploration with a focus on the use of information and communications. Section 2.6 reviews what is known about the Martian environment from past Mars exploration, possible architectures for human mars exploration missions, and past work on the integration of human and robotic exploration. Section 2.7 broadly discusses the methodology of this thesis.

2.1 Basic Definitions

System. The NASA Systems Engineering Handbook [NASA SP6105, 1995] defines a *system* as "a set of interrelated components which interact with one another in an organized fashion toward a common purpose." The International Council on Systems Engineering [INCOSE, 1998] defines a system as "an integrated set of elements [that operate] to accomplish a defined objective." Likewise, [INCOSE, 1998] defines a *system architecture* as "the arrangement of elements and subsystems and the allocation of functions to them to meet system requirements." A system must always

be considered in the context of its super-system, the system for which a system is but a sub-system or component [Saleh, 2001].

Systems engineering. Systems engineering can be defined as "an interdisciplinary approach and means to enable the realization of successful systems"[INCOSE, 1998] or "an interdisciplinary approach encompassing the entire technical effort to evolve and verify an integrated and life-cycle balanced set of system people, product, and process solutions that satisfy customer needs"[NASA SP6105, 1995]. Figure 2-1 illustrates part of the systems engineering process - the process of mapping a set of customer needs to a set of functional requirements and design parameters. In the context of distributed architectures, mapping a set of functional requirements into a set of design parameters is a very challenging task because of the potential complexity of distributed systems and the possibility of desirable or undesirable "collective behavior" of system elements.

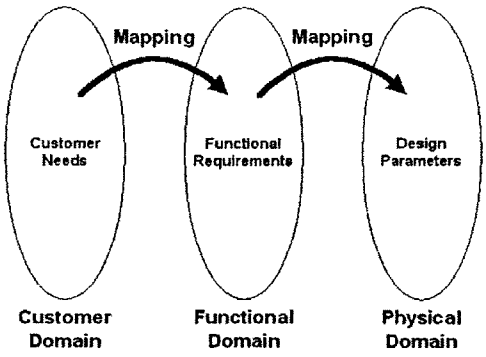


FIGURE 2-1. Systems engineering includes the process of mapping a set of customer needs to a set of functional requirements, and mapping that set of functional requirements to a set of design parameters. Adapted from [Saleh, 2000].

Flexibility and robustness. This thesis is concerned with how distributed architectures can respond to change. Flexibility and robustness are duals of each other and characterize the ability of a system to respond to change. Robustness is the ability of a system to satisfy the same set of functional requirements given changes in system design parameters (changes in the physical domain). Flexibility is the ability of a system to satisfy a different set of functional requirements given the same set of design parameters. Robustness therefore corresponds to the ability of a system to maintain performance in a given environment given changes in the system elements. Flexibility corresponds to the ability of the system to adapt to changes in the environment or mission requirements given the same system elements. While there is considerable ambiguity in the typical use of these words, the above definitions from [Saleh, 2001] unambiguously define flexibility and robustness, and differentiate one from the other. Saleh also distinguishes between flexibility in the systems engineering process, and flexibility of the designed system. Unless otherwise specified, this thesis will always be concerned with flexibility and robustness of the designed or envisioned system, and not of the system engineering process.

Distributed architecture. The word *distributed* means "dispersed...through a whole space or surface" or "separated and allocated to distinct places or compartments" according to the Oxford English Dictionary. Architecture is defined as "the construction or structure" or "the conceptual structure and overall logical organization of a... system from the point of view of its user or design; a particular realization of this." *Distributed architecture* therefore refers to a system with a *structure consisting of distributed elements.* Distributed systems may be more flexible or robust than other types of systems because the system performance is a function of both the elements that make up the system, and the *distributed structure* of those elements. Changing the structure of a distributed system may

enable that system to meet a new set of functional requirements with the same element-level design parameters (flexibility) or may enable that system to meet the same set of functional requirements after a change in the set of design parameters (robustness). Examples demonstrating this structure-function relationship will be covered in Chapter 3.

Exploration system. The term *exploration system* will be used to describe a system that collects, analyzes, and disseminates information and that (1) is situated in an environment, (2) has a distributed architecture and (3) is composed of a set of elements that may include humans and/or machines and may be fixed or mobile relative to some system-wide coordinate system.

Uncertainty. Uncertainty is fundamentally important in system design and operation because uncertainty drives a need for system flexibility and robustness. Sources of uncertainty include lack of information about the current state of a system or an environment, and a lack of ability to predict the future state of a system or environment. One illustration of a fundamental uncertainty relationship is the Heisenberg uncertainty principle, which bounds the accuracy to which position and momentum can be measured simultaneously [Heisenberg, 1927]. In the context of a system for Mars surface exploration, it is important to understand the uncertainty in how the performance of the system may change in the future (driving a need for system robustness), and how the requirements of the system may change in the future (driving a need for system flexibility).

Entropy. Entropy is often associated with uncertainty, and indeed one can use entropy as a measure of uncertainty. However, entropy can also be used as a measure of "equality, disorder, diversity, lack of concentration, similarity, objectivity, unbiasedness, randomness... and many other characteristics that do not even require probabilistic concepts for their description and that have no relationship with uncertainty [Kapur and Kesavan, 1992, p. 10]." Entropy methods may be used to characterize uncertainty or information content, or may be used in an attempt to make unbiased decisions based on limited data. The Shannon measure of entropy [Shannon and Weaver, 1963] will be used herein: a mathematical definition of entropy is given in Appendix A. Cross-entropy will be used as a measure of directed divergence in Chapter 3.

These definitions provide a foundation for the following review of distributed systems and the following sections of the thesis.

2.2 Distributed Systems

This section focuses on multi-agent systems, which are a particular type of distributed system in which the elements of the system are

called agents, and have the characteristics of agency, defined below.

A multiagent system is typically both modular, extensible, and flexible, and focuses attention on the interactions between elements of the system - the information transferred between elements, or between elements and the environment. Multiagent systems are one approach to achieving distributed artificial intelligence, and can be powerful for both qualitative and quantitative modeling of complex systems. In the context of Martian surface exploration, a multiagent system approach provides the benefits of:

- An incremental top-down and bottom-up modeling approach with well defined interfaces.
- A focus on information flow and communications between system elements which is critical to accomplishing the fundamental goals of exploration.
- A method to explore how synergies between system elements such as humans and robots can be achieved: this synergy is distributed artificial intelligence - it is how both groups working together can accomplish more than the sum of each group working independently.
- An ability to model a complicated exploration process, simulate a variety of conditions, and demonstrate limitations and benefits of such a system in both qualitative and quantitative ways.

2.2.1 Agency

Before specifically defining a multiagent system, it is necessary to define the basic component of such a system: the *agent*. An exact definition of *agent* may prove challenging, because multiagent systems researchers do not agree on the exact definition [Weiss, 2000].

Weiss (2000) defines an *agent* as "a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives." We will drop the word "computer" from this definition so that other systems may also be considered to be agents if they meet the definition given below. Figure 2-2 shows a generic agent interacting with an environment.

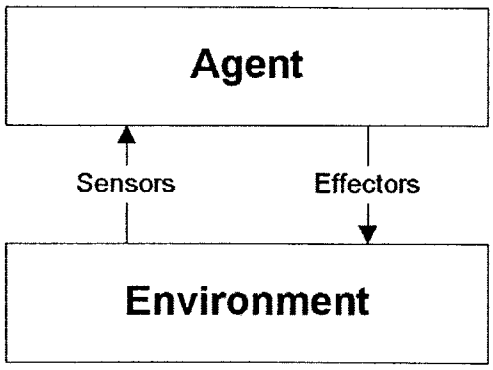


FIGURE 2-2. A generic agent is some entity that interacts with an environment using sensors and effectors.

In general, an agent is defined as a hardware or software system that exhibits the following traits [adapted from Woolridge et al, 1995]:

- Autonomy: Agents have some level of control over their internal state and interactions, even without external input.
- Reactivity: Agents have the ability to sense at least some characteristics of their environment,

- Social ability: Agents interact with other agents via some kind of agent-communication language.
- Pro-activeness: Agents do not simply react to their environment and are able to exhibit goal-directed behavior by taking the initiative.

Reactivity, pro-activeness, and social ability are characteristics of so-called *intelligent agents*, while the notion of autonomy is central to agency.

Figure 2-3 schematically shows the factors involved in determining agent behavior. Agent behavior is determined by several factors including the characteristics of the agent or "organism," the tasks that it will perform (i.e., related to its goals) and the environment in which it exists.

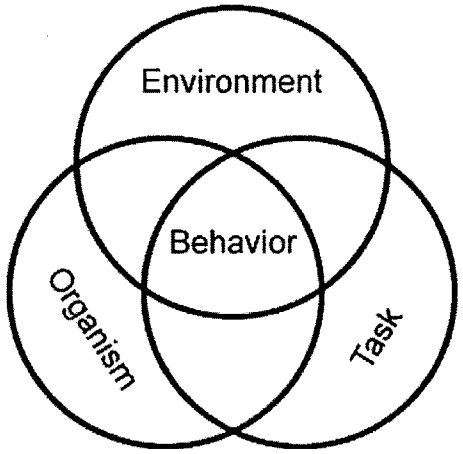


FIGURE 2-3. Schematically, behavior is determined by three factors, including the environment, tasks or goals, and factors inherent to the organism or agent. Redrawn from *Empirical Methods for Artificial Intelligence*, Paul R. Cohen, MIT Press, Cambridge, Massachusetts, 1995.

2.2.2 Multiagent Systems

Now that the notion of agency has been defined, it is possible to define the term *multiagent system* as a collection of agents in an environment. To understand a given multiagent system one must therefore understand (1) the environment in which the system exists, and (2) the agents that make up the system, including their capabilities and goals. These two aspects of multiagent systems will be discussed in turn.

Properties of the environment. In an abstract sense, one can characterize an environment by the appearance of the environment from the viewpoint of an agent in that environment. Table 2-1 lists many of the important properties of an environment of a multi-agent system. Note that in a broad sense, what a given agent sees as the environment may include all other agents in the environment, or any sub-part of an environment. In some multiagent systems, the environment itself is modeled as an agent.

Properties of agents. The agents of a multiagent system, as a group, may be characterized by many properties including the number of agents and the extent to which they are homogeneous or heterogeneous. Homogeneity or heterogeneity of a agent group may refer to similarities or differences in, for example, agent goals, internal architectures, or interaction capabilities.

Properties of interactions. Interactions between agents and other agents, or agents and the environment, are often characterized by the statistical nature of their interactions (for example, the frequency and persistence of a given interaction behavior), or the level of information processing involved in an interaction. The level of information processing may vary from simply acting as a router of information (signal-passing) to knowledge-intensive processing where the information content of a message is greatly altered from

receipt to transmission. In addition, the pattern (flow of data and control; decentralized vs. hierarchical), variability (fixed/changeable) and purpose (competitive/cooperative, social) of interactions are important. The information content of communications can also characterized by whether it is descriptive or prescriptive, a personal interpretation versus a conventional (system-level or group-level agreed upon) interpretation, or is subjective or objective.

2.2.3 Distributed Artificial Intelligence

Developing a multiagent system is one approach to achieving distributed artificial intelligence. Another way of thinking about distributed artificial intelligence is that it is an emergent property of multiagent systems. Distributed artificial intelligence is a label given to the value added to a distributed system by the use of distributed algorithms.

The mathematics of distributed artificial intelligence systems (i.e., systems that run distributed algorithms) can be described in terms of graph theory, which will be covered in the necessary detail in Chapter 3 - *Building an Exploration System*. Fundamentally, distributed algorithms require a system capable of message passing between connected nodes. Distributed algorithms can take advantage of the information processing and storage capabilities of many nodes.

Nodes in a general purpose distributed artificial intelligence system do not need to meet the definition of agency, and therefore multiagent systems could be considered a subset of distributed artificial intelligence systems. Nodes in a distributed artificial intelligence system perform computations (or perform actions) locally, and pass messages to other connected nodes. Fundamentally, information and computation are physical: no fundamental difference between "action" and "information processing" exists.

Routing and Flow Control. Nodes must serve as message routers. When messages cannot be sent directly from one node to another, a message must be passed along other nodes in order to reach the destination node. This means that in addition to receiving nodes for local processing, nodes must act as message routers for other nodes. At the scale of many nodes, the problem of routing and flow control becomes important in distributed artificial intelligence systems.

Synchronism and Asynchronism. Distributed computations may require synchronization between nodes. In terms of multiagent systems, agents may act in both a synchronous and/or an asynchronous manner. Both may be required to achieve system-level goals. In the physical world, exact synchrony is not physically realizable, but bounded asynchrony is possible.

TABLE 2-1. Environment Properties for Multiagent Systems

Property	Range
Accessibility	Extent to which characteristics of the environment are observable or unobservable
Determinism	Extent to which changes in the environment are deterministic or nondeterministic
Periodicity	Extent to which changes in the environment are periodic or aperiodic
Dynamicity	Extent to which the environment is static or dynamic
Continuity	Extent to which the environment is quantized (i.e. discrete vs. continuous)
Diversity	Level of heterogeneity of the environment
Predictability	Extent to which changes in the environment can be predicted.
Resource Availability	Extent to which the environment contains resources usable by agents.

Algorithmic stability and termination. In a distributed artificial intelligence system, it is important to know whether a given algorithm will produce a stable result, and whether a given algorithm will terminate in a reasonable amount of time. Algorithms that may take forever to terminate may not be desirable in many cases because local information processing, local interaction, and message passing between nodes all have a cost associated with them both in terms of time and in terms of energy.

Service discovery protocols. In systems where nodes of the exploration system are mobile, and where the graph of communication links between nodes changes as a function of time, nodes need a method of identifying what communication links are available to communicate with other nodes. A service discovery protocol is a distributed algorithm for determining the existence of a communication link between one or more nodes.

In future dialog, the phrases "multiagent system", "distributed system", and "distributed artificial intelligence system" will be used interchangeably to refer to a system that consists entirely of agents and an environment, and that can be abstracted as a graph in which nodes represent agents and edges represent opportunities for interaction (or communication) between agents. Figure 2-4 schematically illustrates this abstracted view of a multiagent system. The words "communications link", "communications channel" and "edge" are used as synonyms to represent a pathway through which two agents can communicate or interact.

2.2.4 Multiagent Systems for Exploration

In an abstract sense, explorers, whether human or robotic, are a group of agents interacting in and with an environment. Multiagent systems theory provides a framework and vocabulary for simulating groups of interacting agents, and have been applied to the modeling of various organizations. Multiagent systems theory is used here to model the exploration system and process; multiagent systems are especially applicable to the domain of exploration because they emphasize the role of information and interaction that are so fundamental to exploration. In the context of exploration, the results of distributed algorithms are the coordinated behaviors of members of the exploration system that result in the accomplishment of system-level goals.

The desirable properties of multiagent systems for exploration are similar to the desirable properties of many other distributed systems. Self-stabilization and bounded termination of distributed algorithms would, in most cases, be desirable properties of the system, because each activity in the system (local information processing or interaction, or message passing between agents) may have some cost associated with it.

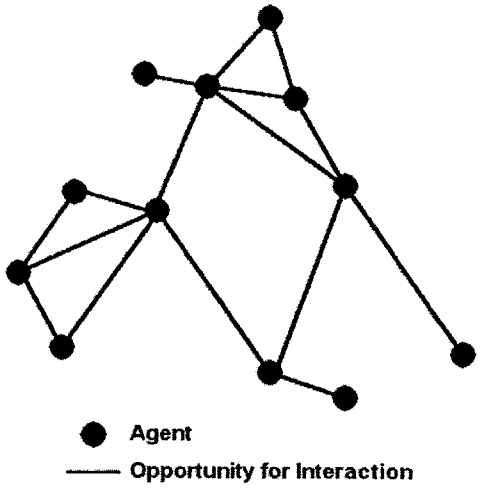


FIGURE 2-4. An abstracted view of a multiagent system where agents are represented as nodes or vertices in a graph, and interaction opportunities (e.g., communications channels) between agents are represented as edges in the graph.

Time and positioning services. Elements of a physical exploration system exist in some physical space and thus exist in some location. Navigating in this space may require an agent to have knowledge about its position in relationship to the environment and in relationship to other agents. The agent may need to:

- build a coordinate system of sorts and relate it to the coordinate systems of other agents or the environment,
- maintain a temporal reference for navigation, or for relating events in the environment to some internal sense of time or place, and
- construct an internal model, in time and space, of the environment, other agents, or itself.

It may therefore be desirable for the multiagent system to provide time and positioning services to agents in the system.

Routing and data storage services. Agents in an exploration system need to communicate and interact even when direct communication channels do not exist between two or more agents. Sharing of information during exploration is also critical. Routing and data storage services are necessary services for exploration systems. Data flow optimization for routing algorithms involves trade-offs between different resources such as power usage, delivery speed or probability of delivery, and algorithmic implementation complexity. Routing protocols also require that each node be differentiated from other nodes so that proper message addressing can be performed.

The human as an agent and multiagent system. Humans meet all the requirements for agency as defined above. Furthermore, humans qualify as nodes in a distributed system in that they are excellent information processors, pass messages to the environment and to other agents, and receive messages from the environment and from other agents. Humans may also be considered multiagent systems in their own right.

Machines and robots as agents. Machines and robots that meet all of the definitions of agency are relatively common, although there are few machines or robots that could be considered truly proactive. Consider, for example, a group of networked computers solving a finite element model problem. They exhibit autonomy because each computer has some level of control over its internal state, without receiving messages from other computers over the network. Each computer is reactive because it can sense characteristics of its environment, such as sensing user input from a keyboard. Each computer has some social ability because each computer sends messages to and receives messages from other computers. In this case such messages might include boundary conditions for its portion of the finite element model problem, or its completion status for a given calculation. A planning routine on one or more of the

The human heart: An example of algorithm termination difficulties in a distributed system. *A simple example of a multi-agent system within a human can illustrate the difficulties associated with algorithm termination in distributed systems: in this example, the multiagent system consists of the collection of cells that together form the human heart. In a vastly simplified view of this system, the agents are cardiac myocytes. Cardiac myocytes are autonomous in the sense that they regulate, for example, the concentrations of various ions in their intracellular matrix without changes in the external environment. Myocytes are reactive in that they respond to various signalling molecules such as extracellular calcium by depolarizing their membranes. They have social ability in that they communicate with other myocytes via gap junctions (where membrane depolarization can be propagated) and express a variety of signalling proteins on the surface of their cell membranes. One may also argue that one reason that damage to cardiac myocytes is so debilitating is that cardiac myocytes are not very proactive - they tend to be rather reactive and to undergo compensatory changes due to their environment (such as increasing myocyte diameter in response to pressure overload conditions in the heart) without necessarily anticipating the system-level consequences of those changes. An example of a distributed algorithm in the system of cardiac myocytes is the creation and propagation of an action potential and the resulting coordinated contraction of the chambers of the heart. When this algorithm does not terminate appropriately, seemingly random and destructively interfering depolarizations and contractions can result. This failure to terminate is due to reentrant loops of depolarization. This example demonstrates that multiagent systems theory can provide a modeling construct on a wide range of scales, and may be applicable to a wide variety of different types of systems, for modeling of both qualitative and quantitative phenomena.*

computers might query other computers about their progress in order to plan future distribution of processing tasks - in this sense the computers would also be proactive.

Properties of the physical world. Humans, machines, and robots operate in a physical world. Several properties of the physical world are summarized in Table 2-2. The physical world, from the point of view of any agent, has limited accessibility - that is to say, a given agent has limited information about its environment, and interacts with its environment in a limited number of ways. The physical world is also nondeterministic, although it is adequately deterministic on a macroscopic scale so that for many types of interactions predicting the future state of an agent's environment may be possible given some information about the current state. The physical world is not exactly periodic or episodic, but most physical environments have important periodicities (day/night cycles, tides, seasons). An agent in a coordinate frame fixed relative to a point on the Martian surface will experience many important periodicities related to the cycle of night and day: this cycle influences exposure to electromagnetic radiation, the thermal environment, gas pressure, wind, dust deposition, and many other important parameters of the environment. The physical world is also dynamic, and we perceive the state evolution of the physical world as a continuous time process. It is diverse in the sense that it can be highly heterogeneous, and evolution of elements of the physical world can range from predictable to non predictable, depending upon the dynamics of the element of interest and the time-scale over which the prediction is made.

TABLE 2-2. Properties of the real world

Property	Description
Accessibility	Often inaccessible
Determinism	Nondeterministic
Periodicity	Non-periodic but many semi-periodicities
Dynamicity	Dynamic
Continuity	Quantized but treatable as continuous
Diversity	Diverse interactions
Predictability	Bounded predictability

2.2.5 Summary

A multiagent system has been defined as a collection of agents, situated in some environment, where intelligent agents demonstrate autonomy, reactivity, social ability, and pro activeness. An environment may also be modeled as a set of agents such that a multiagent system is purely defined as a collection of agents, some of that may or may not demonstrate all of the qualities of agency. From the perspective of a given agent, the environment may refer to all other agents in a multiagent system. Interaction, or message passing, is the fundamental activity of agents. One emergent property of agents passing messages in a multiagent system is distributed artificial intelligence. Major challenges in systems for distributed artificial intelligence are routing and flow control, synchronization, algorithm stability, and algorithm termination. Multiagent systems modeling is applicable to exploration because of the relevance of information processing and interaction to exploration. Agents in an exploration system may "desire" services such as positioning and timing, message routing and information storage services. Humans and some machines can be considered agents because they meet the definitions of agency. Finally, the physical world is a highly com-

plex environment as compared to the typical environment as modeled for a multiagent system.

The following section briefly discusses a few existing distributed systems using the language of multi-agent systems, and illustrates that real systems can be described in the language of multi-agent systems.

2.3 Analogous Systems

Examination of existing analogous distributed systems can demonstrate some of the important trades in distributed system design. Several types of distributed systems, including cellular phone networks, ad-hoc wireless networks, and battlefield information systems are briefly described. Astronauts form a distributed system during extravehicular activity - this system is discussed briefly in Chapter 4. Existing systems and architectures can provide a point of reference for trade studies of distributed architectures for planetary surface exploration by demonstrating the relationships between the functions of the systems and their network architectures.

2.3.1 Cellular Networks

Cellular networks consist of a set of fixed nodes (cell towers) that serve as access points for mobile nodes (cellular phones, etc.). In most cellular networks, fixed nodes provide routing services, and direct node-to-node connections are not possible. This heterogeneous architecture provides a solution for point-to-point communication when the coverage area of the cellular network does not need to change quickly with time. In the language of distributed systems (as described in the previous section), a cellular system is a distributed system composed of two types of agents: cell towers are agents that consist primarily of a communications processor, and cellular phones are agents without communications processors (these mobile agents provide no forwarding services to other agents). Cell towers provide bridges to other communication systems or to other cell towers. This traditional single-hop cellular architecture can lead to (adapted from [Lin and Hsu, 2000]):

1. high cost due to extensive fixed infrastructure (i.e., number of cell towers)
2. throughput limited by the number of fixed infrastructure elements
3. high power consumption of mobile stations having the same transmission range as fixed infrastructure elements

Mobile agents must contain some sort of service discovery protocol to identify what service is available, but fixed infrastructure elements can be considered static, and thus routing between fixed

infrastructure elements does not need to be dynamic (dynamic routing might still be valuable to eliminate network congestion or cope with fixed infrastructure failures).

2.3.2 Ad-hoc wireless networks

A mobile ad-hoc wireless network consists of a group of mobile elements connected by wireless links. The major benefit of ad-hoc networks is that they allow the nodes of the network to be mobile. Ad-hoc wireless networks can also improve energy efficiency by utilizing multiple small hops between nodes instead of much longer single hops.

Because mobile elements are free to move from place to place, the network topology can change rapidly. In a homogenous ad-hoc wireless network, nodes (or agents) must serve as both hosts and communication processors (to provide forwarding services for other nodes). This introduces complexity to the problem of routing messages from one node to another: the routing problem must now cope with a dynamic network structure.

To implement dynamic routing, each node either must update every other node about its state (position, operational state, etc.) or routing must be accomplished without global network state information.

Many wireless networks have some form of fixed (or relatively fixed) core, and this fixed core can simplify the routing problem. The MIT Grid system provides geographic-based routing services [Jha et al., 2001]. Cellular networks are similar to ad-hoc wireless networks with fixed cores in that routing paths between the fixed elements are known: this helps to reduce the difficulty of the routing problem.

Ad-hoc network quality-of-service (probability of delivery, latency, etc.) problems can be very challenging. For example, ad-hoc networks suffer from path vulnerability (due to transit of signal through many stations, and potential failure of any station to retransmit the signal). Path vulnerability can be reduced if the number of hops is limited and station mobility is low [Lin and Hsu, 2000].

Lin and Hsu (2000) suggest that a multiple-hop cellular architecture, in which mobile agents serve as bridges (e.g., provide message forwarding services to other mobile agents), can reduce the number of required fixed infrastructure elements, provide connections without fixed infrastructure elements, and reduce path vulnerability. In some ways, this is similar to providing a fixed core as part of an ad-hoc wireless network.

One of the main roles of a system architect is to reduce system complexity [Rechtin, 1991]. In a grossly generalized manner, ad-hoc networks introduce complexity in exchange for flexibility as compared to cellular-type networks with fixed infrastructure compoents. It seems clear that both have appropriate uses, and that some blending of the two might best serve a given distributed system, depending on the function(s) of the distributed system (e.g. mobility of system elements, required operating margins, etc.).

2.3.3 Battlefield Information Systems

Battlefield information systems, also known as "communications, command, control, and information systems," have many of the relevant properties of distributed systems for exploration. Battlefield information systems support a group of mobile agents (human and robotic systems) via a highly mobile infrastructure (with potentially some fixed infrastructure elements). Both systems collect information about an environment, and relay information to and from mobile and fixed agents.

The primary differences between battlefield information systems and distributed systems for exploration are that battlefield information systems have a significantly greater need for secure transmissions between agents, and resistance to intentional jamming of communications links. The potential cost of system failure for battlefield information systems may include destruction of the system or loss of life, both internal and external to the system.

An analogy can be drawn between a group of exploration agents coordinating to accomplish a task and a squad of soldiers. Schoening and Christian (1992) describe network topologies that might support soldiers, and partition the network architecture problem into squad-internal and squad-external domains. Heterogeneity of network architectures at different scales might allow different routing strategies to be utilized at different scales, and could enhance power efficiency.

Distinguishing features of battlefield information systems may include (adapted from Graff and Liebman, 1994):

1. planned mobility of nodes,
2. high survivability of connectivity (for example, by requiring multiple paths between local networks),
3. priority-based utilization of communications resources, and
4. high frequency of planning activities.

2.4 Exploration

To better understand how distributed architectures can support exploration, the discussion must move from the soldier to the explorer.

To explore is to investigate, to search out, to closely examine, touch or probe, and to go into and to range over an area for the purpose of discovery. Humans have a proud history of exploration, and throughout that history, humans have made use of tools to extend their senses and to remake their environment or a part of their environment. Communication of information within a group of explorers and between explorers and the "world-at-large" has played an ever increasing role. Communications systems now allow a human to stay in touch with other humans from nearly any point on the globe (given the right hardware and software). Exploration has been pursued in order to yield new information and knowledge about the universe, and as an experience in and of itself. Robotic systems have extended the reach of explorers to include areas of extreme pressure, temperature, and radiation too extreme for any human to survive. Humans have ventured into space, but only encased inside our machines - already our survival during human exploration is highly dependent upon machines, and it is likely to become more so when we venture on to Mars and other worlds.

This thesis is concerned with exploration of an area for which the size of the explorer is small relative to the size of the area under study. This type of exploration requires significant mobility.

2.4.1 What is exploration?

Exploration is a process by which some group of entities (agents) extract information from some environment. The act of exploring ultimately changes the environment, and changes the explorer. The goal of an exploration system is therefore to extract (presumably useful) information from an environment, given some set of constraints for system operation (e.g., minimal cost, maximal value of information, or some combination of these and other constraints).

2.4.2 Impact of Communications on Exploration

Communications technologies have drastically enhanced exploration by:

1. reducing the time between discovery and sharing of the results of discovery,
2. enhancing the ability of explorers to adapt to obstacles by improving decision-making abilities through better sharing of information, and

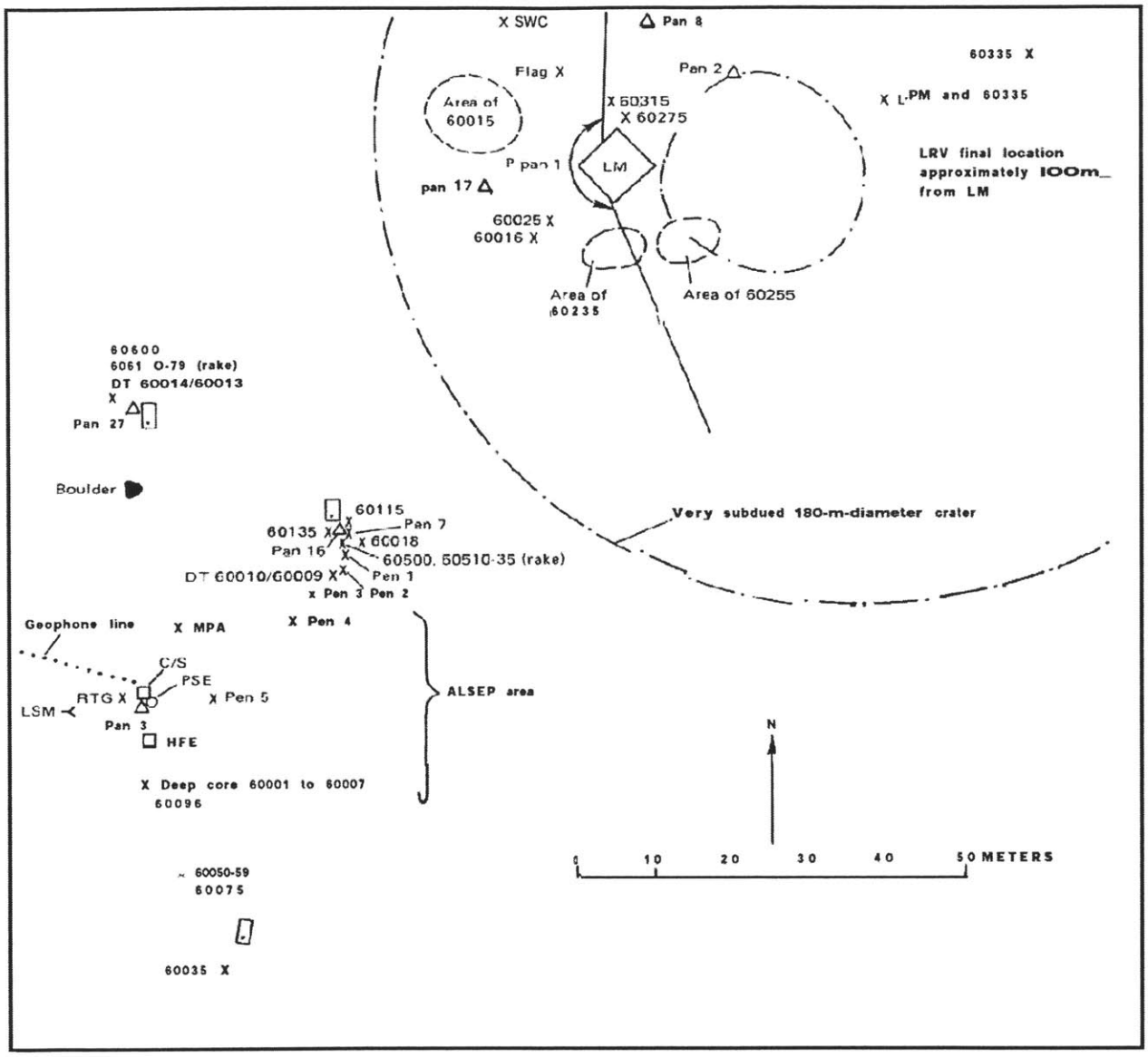
3. permitting "outsiders" to provide different perspectives and to participate in exploration from a distance.

Stuster (1996) discusses the role of "outside" communications and suggests that while beneficial, "outside" communications can also be disruptive. Stuster also discusses many of the psychological issues involved in exploration and reviews many of the elements of success and failure for many expeditions from the voyage of Columbus in 1493, through Antarctic exploration, to recent long-duration space flights and polar expeditions.

The value of exploration has been enhanced by reducing the period between exploration discoveries and the sharing of those discoveries with the "outside" world, whether that world consists of a select group within the community or the public at large. This delay reduces the value of information because information gained during exploration may be distorted or lost before it is communicated. The Lewis and Clark expedition of 1804-1806 was a tremendous success, but sharing that success with the public was less than successful:

> But as Lewis, appointed by Jefferson to serve as governor of Louisiana Territory, and Clark, designated as the territory's superintendent of Indian affairs, grew more and more involved in official duties and responsibilities in St. Louis, the work on their journals was pushed aside. Lewis's untimely and tragic death in October, 1809, while en route to Washington on government business, further delayed the completion and release of the official account...The first authentic "History of the Expedition," published in Philadelphia in 1814 after an almost unbelievable three years of confusion and delay, consisted of two volumes devoted almost entirely to the narrative itself, with a meager amount of scientific data appended to the second volume. It was certainly not the work that Lewis had projected, but it offered to the American public their first opportunity to assess the New West. [Allen, 1975, p. 374-375]

By the time of the Apollo missions, communications technologies allowed about six hundred million people to watch a human step onto the surface of the moon for the first time, and data collected on the lunar surface by Apollo astronauts was immediately transmitted back to Earth for interpretation. This not only enhanced the probability of survival for the lunar surface astronauts, but also allowed greater community participation in the exploration of the lunar surface and enhanced the value of future data collection during the missions by guiding what new data was collected. Yet even with "instant" communication of data, understanding the data from the Apollo missions required significant reconstructions of the missions including what data was collected, when, and where. These painstaking reconstructions (Figure 2-5, for example) were only possible because of extensive photographic documentation and other measurements combined with a constant running commentary provided by the lunar surface astronauts.



EXPLANATION

○	Crater rim	PSE	Passive seismic experiment
X 60335	Sample locality and number	LSM	Lunar surface magnetometer
DT	Drive tube	HFE	Heat-flow experiment
P pan	Partial panorama	RTG	Radioisotopic thermoelectric generator
△ Pan 2	Panorama location and number	MPA	Mortar package assembly
▯	LRV, dot on front	LPM	Lunar portable magnetometer
X Pen 5	Penetrometer reading, location and number	SWC	Solar wind composition
c/s	Central station		

Some instruments, such as the Lunar Portable Magnetometer, were wholly dependent upon the transmission of data via astronaut voice communications [Ulrich et al., 1981].

Even today, map compilations based on geologic field data often take years to compile or are never compiled, probably because of the time and attention to detail required, coupled with fading mem-

FIGURE 2-5. A reconstruction of astronaut activities near the Apollo 16 Lunar Module and the Apollo Lunar Surface Experiments Package deployment area. From [Ulrich et al., 1981], section D1, *Field Geology of Apollo 16 Central Region*, by G. G. Schaber, p. 22.

ories of field observations or unanswered questions [Burchfiel, 2001].

2.4.3 Exploration as world building

Human explorers, in addition to recording data in written, photographic, or electronic forms, build *mental models* of the world that they have explored. Human explorers also use physical models and three-dimensional reconstructions to record and synthesize data. Often, notes, sketches, journals, etc. serve as incomplete records of mental models, or help in the development or testing of mental models. Exploration provides the information that is used to construct a model of the environment being explored, whether or not this model is in the minds of the explorers or some other group.

Improving the context of collected data and enhancing the ability of explorers to build, test, and communicate mental models would profoundly enhance the value of exploration. The value of data collected during exploration has been greatly enhanced by:

1. accurate clocks and improved positioning technologies, such as the Global Positioning System,
2. software and hardware systems capable of capturing data and the context of that data
3. software and hardware systems capable of communicating relationships in data to decision-making agents (such as geographical information systems software communicating spatial relationships to the people using the software to analyze spatial data sets).

Measurements and Observations. Measurements and observations during exploration have a cost in terms of time or resources (during exploration) and in terms of planning or development. In addition, the cost of delivery of information from its source to the decision-making agents or the ultimate benefactors of the information must be considered. The meaning of the information must also be interpreted in some manner consistent with what "signal" is of interest and how much "noise" is present in the signal (what is signal and what is noise depends upon the goals of the analysis).

Clearly some approach must be taken to decide what measurements or observations should be made during exploration. Specific strategies are highly dependent on specific environments and specific objectives, but it will suffice to say that the most common problem solving strategy used by humans is heuristic search [Rechtin, 1991]. [Baecher, 1972] treats geological exploration as a statistical problem covering several classes of exploration in a probabilistic manner including reconnaissance, pattern recognition and reconstruction (mapping), and search. For robotic exploration, it may be fruitful to build a "mental model" of the environment using similar approaches to the techniques used by humans.

Virtual Exploration. A virtual model of the environment being explored could provide important information to both human and robotic agents of exploration, and would enable other humans to remotely participate in the exploration process. Not only would this allow remote participants to build their own mental models, but it would allow remote participants to positively influence what exploration might be carried out by exploration agents in the future.

2.4.4 Characterizing Exploration Agents

Exploration agents, both human and robotic, can be characterized by their spatial and temporal statistical behavior, their consumption or use of resources or production of waste products, their production of valuable information, or achievement of mission goals. For human agents, resource usage might include caloric requirements or oxygen consumption, while for a robotic agent, power consumption might be the primary metric of consumption. Exploration agents can also be tracked by their internal states, although modeling internal states of agents is beyond the scope of this thesis, and many processes in agents (especially human agents) may be unobservable.

Similar approaches can be used to characterize both human and robotic agents, but while understanding the behavior of individual agents is a necessary condition for understanding the performance of a distributed collection of agents, it is not a sufficient condition: understanding the performance of a distributed collection of agents, given a (accurate or incomplete) model of the individual agents, remains a challenging and complex task.

2.5 Extravehicular Activity

Extravehicular activity is, by definition, activity occurring external to a (space) vehicle that requires a life support suit. This section focuses on the role of information and communications during extravehicular activity.

For short, well-defined tasks, an extravehicular activity system need not provide more than the life support systems to sustain life, and the necessary capabilities to interact with the desired environment to accomplish a task. For longer duration extravehicular activities, uncertainty about tasks or task sequence necessitates additional flexibility and robustness both in terms of information about the environment and in terms of the coordination of a dynamic sequence of tasks between multiple astronauts.

Extravehicular activity has been critical to the space program because it adds robustness and flexibility to space missions: humans excel at working in unstructured or semi structured envi-

ronments. Extravehicular activity has given humans the opportunity to carry out fixed mission goals even with unexpected failures or problems (such as redeployment of a satellite) or to adapt to changing conditions (such as the development and deployment of a makeshift sunshade for the Skylab observatory). Nevertheless, extravehicular activity is very structured compared to many "everyday" forms of human activity.

Extravehicular activity requirements for future human planetary surface exploration missions dwarf all current and planned microgravity and partial gravity extravehicular experience (Figure 2-6). The construction and habitation of the International Space Station will require a large increase in extravehicular activity, and may provide the opportunity to develop and test technologies for planetary exploration. Nevertheless, extravehicular activity for human planetary surface missions will require additional long-duration experience, where issues such as wear and tear and maintenance of extravehicular activity equipment will play a much larger role. Long duration missions will also increase uncertainties associated with extravehicular activity training, planning, and execution, and will require increased flexibility in future extravehicular activity systems.

The lack of immediate ground support during future human planetary surface extravehicular activities will require crew members to play a more active role in planning, executing, and supporting extravehicular activity; increased autonomy and improvements in self-reliance will be required.

During on-orbit extravehicular activity, astronauts are far from the typical vision of flexible, autonomous, and self-sufficient teams that many envision will be required for long-duration planetary surface exploration. During current (2001) Space Shuttle extravehicular activities, more than a hundred people (experts in space systems from space suit components to extravehicular activity tools to space payloads) monitor extravehicular activity voice communications in real-time and are ready to support the astronauts at any moment. For future planetary exploration, similar support mechanisms will be required, but the support delivery system (whether for human or robotic explorers) must be significantly different. Real-time support will need to be either accessible to an astronaut via some local information repository, or could potentially be provided by astronauts back in a base camp facility.

In addition, the goals of future human planetary surface exploration are likely to be significantly different than the goals of extravehicular activity during on-orbit activities or even during the Apollo program. The focus is likely to be on exploration requiring significant mobility with frequent changes in the "activity manifest" - not the execution of a specific "grand plan." The moment to moment details are likely to be considerably less well defined. Training for

FIGURE 2-6. The number of extravehicular activities in the US and Russian space programs from 1965-2000 are compared to the number of extravehicular activities for a single human mission to Mars. Planned extravehicular activities due to construction of the International Space Station are shown for the time period 2001-2006. Number of extravehicular activities for a human Mars mission are based on a 600 day surface stay time of four to six crew members, each conducting approximately two extravehicular activities per week, a conservatively low estimate.

extravehicular activity will likely need to be based more on general skills.

Before attempting to design the "exploration system" and "extravehicular activity system" for Mars, we will need to better understand the process of exploration. This system should take advantage of what humans do best (and should certainly take advantage of what robotic agents do best), and should augment human capabilities where desirable and necessary. The system should match human capabilities to the requirements levied upon the human element of the system, and should monitor and compensate for physiological deconditioning of the crew. Workload considerations will need to change with time to compensate for changes in crew capabilities related to adaptation to weightlessness or partial-gravity environments.

Future Mars mission architectures and planning activities need to focus more on surface exploration activities, and the processes required for planning and execution of exploration activities and the coordination of those activities between human and robotic agents. While task-based training may still be useful, skills based training may be beneficial for many surface exploration activities.

2.6 Exploring Mars

This section provides a brief overview of the Mars environment, past Mars exploration activities, and some of the challenges of future human and robotic mars surface exploration, as these items relate to the operation of surface-based distributed systems.

2.6.1 The Mars Environment

Bulk parameters. Several of the bulk parameters of Mars are compared to the Earth in Table 2-3. The nearly 700-day Mars-year and the nearly 24 hour day drive major cyclical variations in global weather, including temperature profiles and dust activity. The increased distance of Mars from the Sun as compared to the Earth reduces solar irradiance, making solar-based power generation strategies less viable for surface operations (to say nothing of terrain shadowing, dust and degradation issues associated with solar-based power generation). The significantly lower surface gravity on Mars has profound implications for the cost of transport of surface-based mobile agents including their energy expenditures and dynamic stability while driving, walking, rolling, or locomoting.

Topography. Martian topography is some of the most varied and extreme in the solar system. With an equatorial radius of about half that of Earth, Mars has about 30 kilometers of vertical relief (the Earth has about 20 kilometers of vertical relief). Mars has the larg-

TABLE 2-3. Comparison of Bulk Parameters for Earth and Mars

Parameter	Mars	Mars/ Earth Ratio
Orbit semimajor axis	227.9×10^6 km	1.524
Orbital period	687 days	1.88
Obliquity relative to orbit	25.2 degrees	1.07
Black-body temperature	210 K	0.827
Irradiance (Mars orbit)	589 W/m^2	0.431
Gravitational acceleration	3.69 m/s^2	0.377

est canyons and some of the flattest terrain of any planet in the solar system. The smaller size of Mars as compared to Earth (and lack of an ionosphere) accentuates the over-the-horizon communication problem, and depending on the local terrain, terrain-masking may pose serious communication problems without significant satellite support or surface communications infrastructure. Topography is also a significant driver of the cost of transport. Regional topography considerations (surface roughness and slopes, for example) also play a major role in landing site selection and therefore affect both the initial deployment and evolution of distributed systems for surface exploration. Topography may also affect the number density of agents (number of agents per unit surface area) required to deploy or maintain a connected distributed system while meeting coverage (region of the surface "covered" by the distributed system in terms of communication coverage, visibility, or some other metric) requirements. Recent advances in global-scale topographic analysis of Mars (Figure 2-7) have been made by the Mars Global Surveyor Mars Orbiting Laser Altimeter team [Smith et al, 2000].

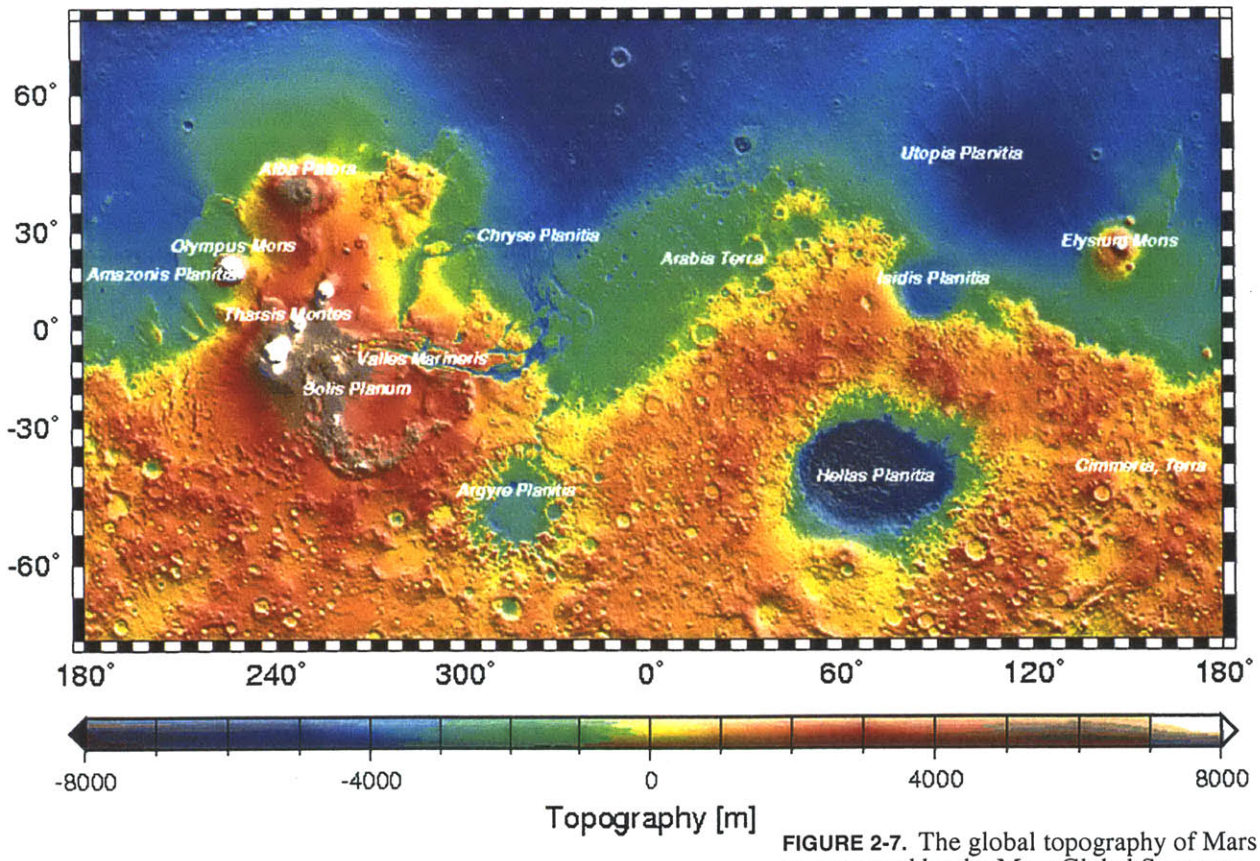


FIGURE 2-7. The global topography of Mars as measured by the Mars Global Surveyor Mars Orbiting Laser Altimeter (MOLA). This topographic map was generated by the MOLA Science Team [Mars Global Surveyor Website, 2001].

Atmosphere. Details of the composition of the Martian atmosphere are not relevant here, but of primary importance is the low pressure of the Martian atmosphere (0.6-1.0 kPa or 6-10 millibar, compared to 100 kPa or 1 bar on Earth).

The low pressure environment makes prospects of atmospheric vehicles with significant payload fractions a remote possibility, although the use of balloons, aeroplanes, and dirigibles in the Martian atmosphere has been discussed, and may be a viable option [Clapp, 1985].

The low pressure also reduces heat transfer by conduction and convection (as compared to higher pressure environments such as the Earth at sea level), increasing temperature differentials caused by differences in illumination or other heat sources or sinks. Morning and evening temperatures at a given location can still change very quickly due to radiative heat transfer or because of low thermal inertia. Temperature gradients and daily cycling can lead to small scale and large scale atmospheric instability, producing dust devils and dust storms. Viking experience suggests that dust storms can "go global" within a few weeks and last several months. Recent events (Figure 2-8) have confirmed this. Eolian (wind driven) processes are thought to be the major geomorphological factor in the current Martian environment and in the recent geological history of the planet.

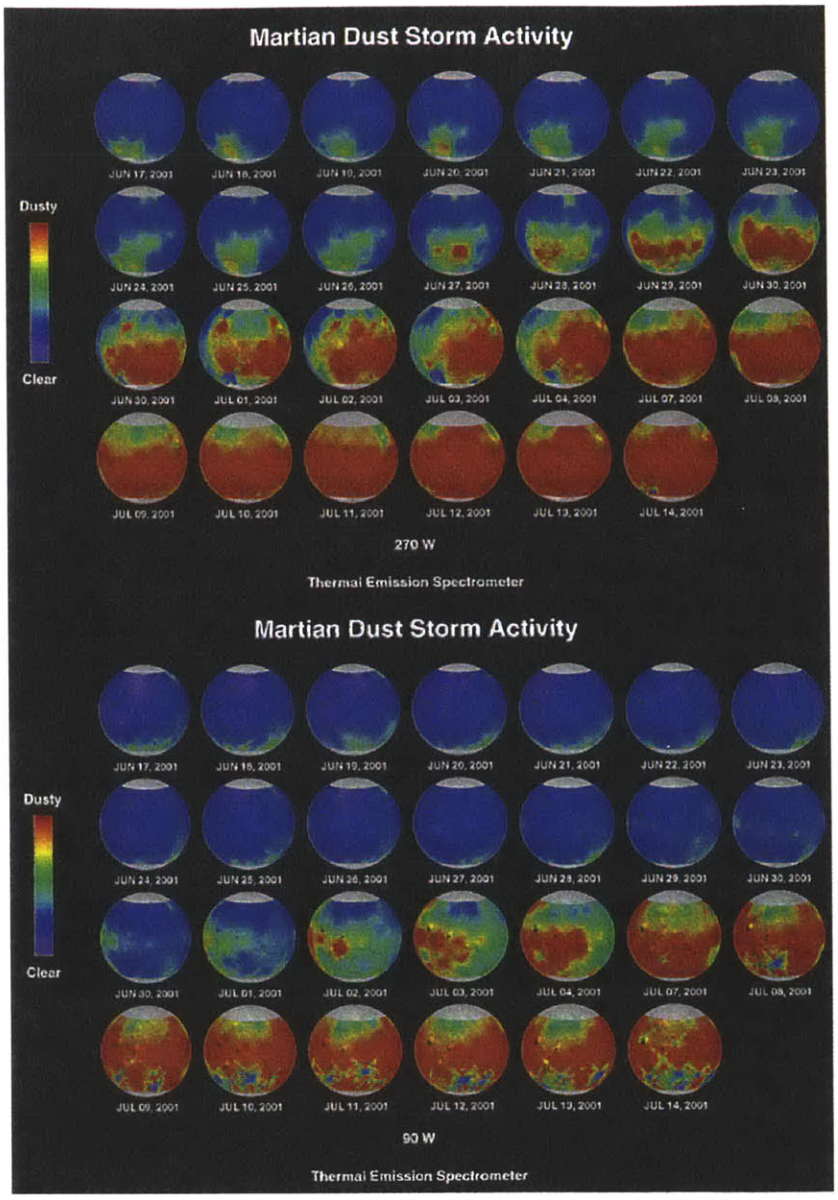


FIGURE 2-8. A time lapse sequence from June 17, 2001 to July 14, 2001 shows the rapid development of a global-scale dust storm for opposite hemispheres of Mars [MGS TES Website, 2001].

A primary effect of the thin Martian atmosphere is to provide a convective and conductive interface for heat transfer between an explorer and the Martian environment, something that must be taken into account when computing a heat budget for future Martian explorers. The thin atmosphere permits a high ultraviolet light flux at the surface that can be a direct hazard to biological material (e.g. human explorers) and can charge metal surfaces via the photoelectric effect, potentially creating grounding problems, arcing dangers, and exacerbating dust adhesion problems. Cosmic ray flux is also significant at the surface and may also cause significant damage to biological material and to radiation sensitive electronics.

Dark shadows may also cause visibility problems for astronauts during extravehicular activity.

Surface properties. Surface properties such as soil mechanical properties directly drive the cost of transport of mobile agents. The surface properties (dust grain sizes, packing, density, etc.) and interactions with the atmosphere may strongly impact dust adhesion and mechanical abrasion. The surface of Mars is also thought to be strongly oxidized, and chemical reactivity may also be a significant problem. Pathfinder [Rover Team, 1997] measured angles of repose of between 30 and 38 degrees, and both unconsolidated drifts and consolidated deposits were encountered by the rover Sojourner. Wheels on the rover would charge to several hundred volts in the Martian environment, and dust was also seen to collect on the wheels, especially in areas of small dust particle size [Rover Team, 1997]. Methods of evaluating surface properties include [Engineering Constraints, 2001]:

- Radar: can be used to estimate root-mean-square slopes (based on scatter), and to measure the bulk density of the surface (based on reflectivity; reflective surfaces are more dense).
- Infrared Thermal Inertia: Low infrared thermal inertia indicates extensive dust, while high infrared thermal inertia may indicate large rocks or extensive rock cover. Infrared thermal inertia provides a measure of surface cohesiveness.
- Visual images: High resolution images can be used to identify large hazards and local/regional slopes on meters to tens of meters scales. Local images from past surface missions can provide rock distribution estimates.
- Topography data (from the Mars Global Surveyor, for example) can be used to estimate slopes on large baselines (hundreds of meters to thousands of kilometers).

Past mission planners did not have a wealth of information with which they could plan Mars surface exploration missions: Mission designers should take advantage of the wealth of data on the Martian surface both in the planning and execution phases of future missions.

2.6.2 Integration of Human and Robotic Exploration

Many human Mars mission architectures have been proposed, including the NASA Design Reference Mission [Hoffman et al., 1997], and the Mars Direct mission [Zubrin, 1991]. [Singer, 1984] and others [Bishop et al., 2000] have proposed the use of the Martian moons, Phobos and Deimos, as a staging area for Mars surface exploration.

Few mission architectures have examined day-to-day surface operations in depth, and even fewer architectures have provided a natu-

ral bridge between current and future robotic exploration and future human exploration of Mars. Recent papers and workshops [Goddard Workshop, 2001] have begun to bring robotic and human expertise together, and have begun to consider task allocation between robotic and human explorers.

Exploration agents, human and robotic, can be characterized by their sensing, information processing, and control capabilities. [Sheridan, 1974] examines many information, control, and decision models of human performance from information theoretic and manual control perspectives. In terms of dynamic sensory capabilities, human senses have dynamic ranges that meet or exceed the dynamic range of most machines, but machines are obviously not bound to this limited set of senses, and generally have greater absolute sensing capabilities. Humans are exceptional information processors and excel in activities requiring [the following adapted from Goddard Workshop, 2001]:

- Synoptic 3-D view, near-field and far field
- In-situ judgement with rapid integration time
- Ability to accept complex or poorly defined input
- Pattern recognition
- Learning, serendipity, experiential leaps
- Hypothesis generation and testing
- High mobility and dexterity
- Ability to redesign experiments and build tools
- Generic flexibility and robustness
- Ability to sense danger and "say no"
- Flexible communication skills

Robotic agents typically excel in activities requiring [adapted from Goddard Workshop, 2001]:

- Extreme (large and small) applications of force or movement
- Very fast (or slow) constrained movement in a known environment
- Repetitive activity requiring predictable or tightly bounded performance
- Precision data collection and storage
- Extensive, well defined calculations
- Expendability

Few actual field trials of robotic and human cooperative activity have been conducted, but combined astronaut and rover field exploration activities have been carried out by [Kosmo et al., 1999, 2000]. Recommendations on robotic and human cooperative activity from the 1999 study in the Mojave desert suggest that effective

cooperation between astronauts and rovers may require or benefit from:

- control over the rover by the suited astronaut,
- a rover with a high traverse rate, in order to keep up with the astronaut,
- a rover-carried tool caddy and sample bag retention capability.

One of the recommendations by [Kosmo et al., 2000] recognized the need for an improved wireless communication system network for space suit and robotic vehicle support.

Much work remains to be done in order to understand how humans and robots can work most efficiently together in the field and during planetary surface operations.

2.7 Thesis Methodology

This thesis does not attempt to solve a particular narrowly posed problem, but instead tells a story about distributed architectures for planetary surface exploration with the support of several different research methods. [Cohen, 1995] discusses the wide range of types of experimentation, each appropriate at different points in the process of generation of new knowledge:

> Exploratory studies... yield causal hypotheses that are tested in observation or manipulation in experiments. To this end, exploratory studies usually collect lots of data, analyzing it in many ways to find regularities. Assessment studies...establish baselines and ranges, and other assessments of the behaviors of a system or its environment. Manipulation experiments...test hypotheses about causal influences of factors by manipulating them and noting effects, if any, on one or more measured variables. Observation experiments...disclose effects of factors on measured variables by observing associations between levels of the factors and values of the variables. These are also called natural and quasi-experimental experiments [Cohen, 1995, p. 7].

The purpose of this chapter was to serve as an assessment study and to examine the literature and to establish the bounds of the other work presented in this thesis.

The following chapter, *Building an Exploration System*, is, in essence, also an assessment study that utilizes simple exploratory examples to illustrate the structure and possibilities for distributed architectures for planetary exploration. Chapter 3 builds on some of the concepts presented in this chapter in a more mathematical fashion, and seeks to tell the story of distributed architectures for Mars surface exploration from a system-level perspective. A process for

enabling distributed architecture trade studies is proposed and its use demonstrated.

Chapter 4, *Observing exploration*, is an observational study of past exploration from two perspectives: the personal experience of the author during a month-long field geologic mapping project, and the experience of the apollo lunar-surface astronauts. The role of information and communication during exploration serves as a focus for both perspectives. Operational concepts for martian extravehicular activity are developed based on these two perspectives, and are explored further in Chapter 5.

Chapter 5, *Supporting and quantifying exploration*, focuses on how the distributed architecture structure described in Chapter 3 can support individual agents or a subset of agents in a system. Methods are explored for modeling the capabilities of human and robotic agents, and simple simulated manipulation experiments are developed to demonstrate potential capabilities of distributed architectures for supporting exploration. Chapter 5 also develops a set of recommendations for future extravehicular systems and for further work.

This thesis seeks to define how distributed architectures can be utilized for Mars surface exploration in a way that allows the methods used here to be incorporated into the systems engineering process to assist in the design and operation of distributed architectures. The focus here is not on the development of detailed or accurate models but to investigate the role of distributed system structure without necessitating detailed definitions of individual elements in the system architecture. Significant efforts were made to provide a framework for further work, while clearly communicating the thesis in a mathematical and visual manner.

2.8 References

A few topical references are briefly summarized to facilitate additional reviews of background literature, and all references are listed in alphabetical order below.

Distributed Systems, Multiagent Systems, and Agents. Russell and [Norvig, 1995] is an excellent general reference on artificial intelligence and on intelligent agents. *Introduction to Distributed Algorithms* [Barbosa, 1996] provides an excellent overview of distributed algorithms for distributed artificial intelligence systems from a computer network perspective. [Weiss, 2000] provides a broad overview on multi-agent systems theory in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Finally, a well-written work on agents and the practical implementation issues of multi-agent systems can be found in [Woolridge and Jennings, 1995].

Analog Systems. [Chakrabarti and Mishra, 2001] provides an excellent overview of quality-of-service issues for ad hoc wireless networking. For detailed technical considerations involved in quality-of-service, see [Chen, 1999].

Exploration. [Haase, 1990] explores the roles of exploration and invention in the discovery process through the use of computer programs. [Baecher, 1972] provides an in-depth discussion of geological site investigation using a probabilistic approach, and discusses biases of human explorers in decision making. [Stuster, 1996] reviews the factors of success (and failure) in many expeditions, from the expedition of Columbus in 1493 to recent long-duration space and Antarctic expeditions.

Extravehicular Activity. [Horrigan et al., 1996] provides a brief but relevant overview of extravehicular activity from the perspective of workload and discusses training and simulation of extravehicular activity. [Connors et al., 1994] interviews the Apollo lunar surface astronauts and identifies considerations for future extravehicular activity systems.

Exploring Mars. A wealth of information has been gathered by past robotic Mars exploration missions [Mars Chronology, 2001]. [Kieffer et al., 1992] provides the most comprehensive treatment of the science results of past Mars exploration, but does not incorporate the wealth of evidence from more recent Mars exploration including results from Mars Global Surveyor [Mars Global Surveyor Website, Smith et al., 2000].

Advanced Technology for Human Support in Space, National Academy Press, Washington D.C., 1997 (*http://books.nap.edu/books/0309057442/html/index.html*).

Allen, John L., *Passage Through the Garden*, University of Illinois Press, Urbana, Illinois, 1975.

Appelbaum, J., and Steiner, A., *Spectral Content of Solar Radiation on Martian Surface Based on Mars Pathfinder*, J. Propulsion and Power, Vol. 17, No. 3, May-June 2001.

Baecher, Gregory B., *Site Exploration: A Probabilistic Approach*, PhD Thesis, Massachusetts Institute of Technology, 1972.

Barbosa, Valmir C., *An Introduction to Distributed Algorithms*, MIT Press, Cambridge, Massachusetts, 1996.

Burchfiel, B. Clark, Professor of Geology in the Department of Earth, Atmospheric, and Planetary Sciences at the Massachusetts Institute of Technology, *Personal Communication*, 2001.

Chakrabarti, S., Mishra, A., *QoS Issues in Ad Hoc Wireless Networks*, IEEE Communication Magazine, February 2001.

Chen, S., *Routing Support for Providing Guaranteed End-To-End Quality of Service*, PhD Thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1999.

Chronology of Mars Exploration, National Space Science Data Center, Goddard Space Flight Center, National Aeronautics and Space Administration, Greenbelt, Maryland (*http://nssdc.gsfc.nasa.gov/planetary/chronology_mars.html*).

Clapp, W. Mitchell, *Dirigible Airships for Martian Surface Exploration (AAS 84-176)*, The Case for Mars II, p. 489-496, Science and Technology Series, Volume 62, American Astronautical Society, Univelt, San Diego, California, 1985.

Cohen, Paul R., *Empirical Methods for Artificial Intelligence*, MIT Press, Cambridge, Massachusetts, 1995.

Connors, Mary M., Eppler, Dean B., and Morrow, Daniel G., *Interviews with the Apollo Lunar Surface Astronauts in Support of Planning for EVA Systems Design*, Ames Research Center, National Aeronautics and Space Administration, 1994.

Engineering Constraints for Mars Surveyor 2003 Landing Site, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 2001 (*http://mars.jpl.nasa.gov/2001/landingsite/EngConstr.html*).

EX13-98-036, Reference Mission Version 3.0, Addendum to the Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team, Exploration Office, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, June, 1998.

Goddard Workshop, *Science and the Human Exploration of Mars Workshop*, National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland, January 11-12, 2001 (*http://www.lpi.usra.edu/publications/reports/CB-1089/CB-1089.intro.html*).

Graff, C., and Leibman, E., *Concepts for an Adaptive Network Planner for the Localized Access Area (LAA) for Battlefield Information Systems (BIS) 2015*, IEEE 1994.

Hass, Kenneth W., Jr., *Invention and Exploration in Discovery*, PhD Thesis, Massachusetts Institute of Technology, 1992.

Heisenberg, Werner, *Zeitschrift für Physik*, 43 (1927), 172-198, received 23 March 1927.

Hoffman, S.J., and Kaplan, D.L., *Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team*, Lyndon B. Johnson Space Center, National Aeronautics and Space Administration, July 1997 (*http://spaceflight.nasa.gov/mars/reference/hem/hem1.html*).

Horrigan, David J., Waligora, James M., Beck, Bradley, and Trevino, Robert C., *Chapter 24 - Extravehicular Activities*, Space Biology and Medicine, Volume III, Book 2, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1996.

IEEE 1220 Standard: Application and Management of the Systems Engineering Process, IEEE Standards Dept., NY, December 1998.

INCOSE System Engineering Handbook, Release 1.0, Internaional Council on Systems Engineering, San Francisco By Area Chapter, January, 1998.

Jha, S., Chalmers, M., Lau, W., Hassan, J., Yap, S., Hassan, M., *Universal Network of Small Wireless Operators (UNSWo)*, IEEE 2001.

Kapur, J.N., and Kesavan, H.K., *Entropy Optimization Principles with Applications*, Academic Press, San Diego, California, 1992.

Kieffer, H.H., Jakosky, B.M., Snyder, C.W., Matthews, M.S. (editors), *Mars*, University of Arizona Press, Tucson, Arizona, 1992.

Kosmo, Joseph J., Trevino, Robert, and Ross, Amy, *Results and Findings of the Astronaut-Rover (ASRO) Remote Field Site Test, Silver Lake, California* (CTSD-ADV-360), National Aeronautics and Space Administration, Crew and Thermal Systems Division, Lyndon B. Johnson Space Center, Houston, Texas, 1999.

Kosmo, Joseph, and Ross, Amy, *Results and Findings of the Representative Planetary Surface EVA Deployment Task Activities, Flagstaff, Arizona* (CTSD-ADV-470), National Aeronautics and Space Administration, Crew and Thermal Systems Division, Lyndon B. Johnson Space Center, Houston, Texas, 2000.

Lin,Y., Hsu, Y., *Multihop Cellular: A New Architecture for Wireless Communications*, IEEE InfoComm 2000.

Lu, J., Chuang, J., and Letaief, K.B., *Architecture and Signaling for Rapid Infrastructure Deployment Using Wireless Communication Networks,* IEEE, 1997.

Mars Global Surveyor Website, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California (http://mars.jpl.nasa.gov/mgs/).

MGS TES Website, Mars Global Surveyor Thermal Emission Spectrometer Website, 2001 (*http://tes.la.asu.edu/*).

NASA Systems Engineering Handbook, SP-6105, National Aeronautics and Space Administration, June, 1995.

Nicogossian, Arnauld E., Huntoon, Carolyn L., and Pool, Sam L., *Space Physiology and Medicine, 3rd Edition*, Lea & Febiger, Malvern, Pennsylvania, 1993.

Oxford English Dictionary, Oxford University Press, 2001 (http://dictionary.oed.com).

Rechtin, Eberhardt, *Systems Architecting*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Rover Team (Moore, H.J., point of contact), *Characterization of the Martian Surface Deposits by the Mars Pathfinder Rover, Sojourner*, Science, Vol. 278, 5 December 1997.

Russell, S. and Norvig, P., *Artificial Intelligence, A Modern Approach*, Prentice-Hall, 1995.

Saleh, Joseph H., *Unpublished work*, Cambridge, Massachusetts, 2001.

Schoening, J.R.; Christian, J.R., *Soldier's Radio,* Military Communications Conference, 1992. MILCOM '92, Conference Record. Communications - Fusing Command, Control and Intelligence., IEEE, Page(s): 497 -499 vol. 2., 1992.

Shannon, Claude E. and Weaver, Warren, *Mathematical Theory of Communication*, University of Illinois Press, 1963.

Sheridan, T.B., and Ferrell, W.R., *Man-Machine Systems*, The MIT Press, Cambridge, Massachusetts, 1974.

Smith, Zuber, Frey, Garvin, Head, Mhleman, Pettengill, Phillips, Solomon, Zwally, Banerdt, Duxbury, Golombek, Lemoine, Neumann, Rowlands, Aharonson, Ford, Ivanov, McGovern, Abshire, Afzal, and Sun, *Mars Orbiter Laser Altimeter (MOLA): Experiment Summary after the First Year of Global Mapping of Mars*, Journal of Geophysical Research, accepted September 16, 2000.

Stuster, Jack, *Bold Endeavors*, Naval Institute Press, Annapolis, Maryland, 1996.

Thwaites, R.G., editor, *Original Journals of the Lewis and Clark Expedition 1804-1806*, Volumes 1-7 and Atlas, Antiquarian Press Ltd, New York, 1959.

Tutschku, K., Leibnitz, K., and Tran-Gia, P., *ICEPT - An Integrated Cellular Network Planning Tool*, IEEE, 1997.

Ulrich, G.E., Hodges C.A., and Muehlberger, W.R., *Geology of the Apollo 16 Area, Central Lunar Highlands*, Geological Survey Professional Paper 1048, United States Government Printing Office, Washington D.C., 1981.

Webbon, Bruce, *Human Requirements for Life Support and Extravehicular Activity in Space*, Presentation at the MIT/NASA/Industry EVA and Life Support Workshop, Cambridge, Massachusetts, 1994.

Weiss, Gerhard, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 2000.

Wilde, Richard C. *100 Years of EVA - Progress and Challenges*, Presentation at the MIT/NASA/Industry EVA And Life Support Workshop, Cambridge, Massachusetts, 1994.

Woolridge, Michael, and Jennings, Nick, *Intelligent Agents: Theory and Practice*, Knowledge Engineering Review, Volume 10, No 2, Cambridge University Press, June 1995.

Zubrin, R.M., Baker, D.A., Gwynne, O., *Mars Direct: A Simple, Robust, and Cost Effective Architecture for the Space Exploration Initiative*, AIAA-91-0328, American Institute of Aeronautics and Astronautics, 1991.

A model is not reality.

The only thing added to the parts to make the whole greater than its parts is the interrelationships among them. Thus, relationships among the elements are what give systems their added value. From this, it follows that the greatest leverage in system architecting is at the interfaces.

The greatest dangers are also at the interfaces.

Eberhardt Rechtin, *Systems Architecting*, 1991

The whole is more than the sum of its parts.

Aristotle, *Metaphysica 10f-1045a, c. 330 B.C.*

3 *Building an Exploration System*

Exploration can be characterized as a process by which new information is collected, analyzed, and disseminated by some collection of elements, in some environment. The goals of this chapter are to describe a distributed system whose purpose is the collection, analysis, and dissemination of information during planetary surface exploration, and that consists of fixed and mobile elements near or on a planetary surface. Orbital (or high-altitude) elements may change the connectivity between surface elements significantly, but considering surface elements (the "surface segment") in isolation has significant value because of the cost of communication with orbital elements (in terms of energy and hardware) and because of the possibility that orbital elements may not be available during all times or at all locations on the surface. This chapter:

1. develops a mathematical basis for abstract representation of an exploration system and planetary surfaces,
2. demonstrates the structural relationships between elements of the exploration system and surface topography in terms of connectivity and coverage metrics,
3. reviews potential features of distributed architectures for Mars surface exploration,
4. defines major trades for distributed architectures for Mars surface exploration and discusses under what conditions distributed architectures might be desirable over other alternative architectures,
5. develops a process for exploration system trade studies for specific mission scenarios, and
6. applies that process to a specific task of surface exploration on the Martian surface.

Figure 3-1 highlights a key goal of this chapter: to demonstrate how the distributed architecture of exploration systems can be characterized given a set of requirements, a model of the environment, and an operational model for elements in that environment. Reconfiguration of the elements in a distributed architecture during operations may enhance system flexibility and robustness, but may also increase system complexity.

Section 3.1 develops an abstract description of the structure of the exploration system based on graph theory. Section 3.2 develops a three-dimensional surface model representation based on height fields and compares real and generated surfaces in terms of several

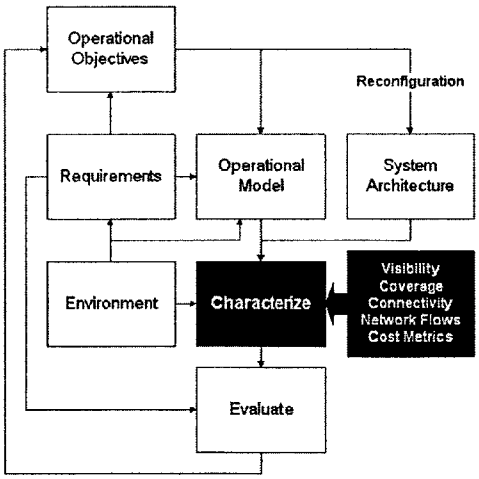


FIGURE 3-1. The process developed and applied in this chapter assists in the characterization of distributed architecture operations given a set of requirements, a model of an environment, and an operational model. Evaluation may modify operational objectives, which in turn may drive reconfiguration of the system architecture during operations to achieve a different subset of requirements.

surface statistics. Section 3.3 demonstrates how to create an exploration system composed of nodes on a particular surface using visibility graphs, and provides results for graph connectivity as a function of surface statistics. Section 3.4 reviews potential features of the exploration system and defines major trades for distributed architectures for planetary surface exploration. Section 3.5 develops a process for facilitating exploration system trade studies for exploration system design. Section 3.6 applies that process to an extremely simple mission architecture for Mars surface exploration that includes a lander and a surface-based sensor network. Section 3.7 discusses limitations of the methods and analysis, and suggests possibilities for enhancement and improvement of the process.

3.1 Exploration System Structure

The structure of the exploration system is described in terms of graph theory, and relies primarily on introductory graph theory as developed in [Diestel, 1997]. License has been taken to use the most informal language possible in order to promote understanding of the material presented herein for readers without a background in graph theory.

3.1.1 Graphs

The exploration system structure consists entirely of elements that will be called nodes. In its most basic form, the structure of the exploration system (at a particular time, in a particular environment) can be described by a graph $G = \{N,E\}$ where N is a set of unique values each representing a node (or point, or vertex), and E is a set of two-element subsets of N where each two-element subset represents an edge in the graph. Each edge represents an opportunity for interaction between elements, and will most commonly be used to represent line-of-sight visibility between two nodes. A graph can be used to represent interaction opportunities for agents in a multi-agent system: agents are represented as nodes, and opportunities for interactions between agents are represented as edges.

Figure 3-2 illustrates a simple example of a graph. The node set of a graph is referred to as $N(G)$ [Traditionally the node or vertex set is referred to as $V(G)$, but here we break with tradition to achieve a consistent vocabulary], and the edge set is referred to as $E(G)$. A graph with node set N is a graph on N, and a edge e is incident to a node n if e contains n. Two nodes n_i and n_j are adjacent if there exists an edge $e=(n_i,n_j)$ (often written as $e = n_i n_j$). Edges are adjacent if both edges contain a common node. A pair of non-adjacent vertices or edges is called independent.

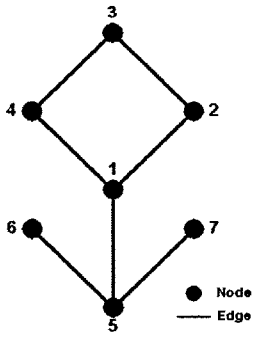


FIGURE 3-2. Example of a simple graph $G=\{N,E\}$. Here $N=\{1,2,3,4,5,6,7\}$ and $E=\{(1,2),(2,3),(3,4),(4,1),(1,5),(5,6),(5,7)\}$.

This graph can be augmented by writing $G_P = \{N,E,P\}$, where P is a set of node positions in some coordinate system. In practice, the elements of P will always be represented by a three-element cartesian vector $p = (x, y, z)$, with all $p \in P$ referenced to some common origin. The curvature of planetary surfaces limits the size of an exploration system that can be accurately represented in this fashion, but a cartesian coordinate system is the most simple and natural choice of coordinate system for the scale and low-curvature surfaces that will be discussed.

This graph can be further augmented by allowing each edge to have some associated "characteristics vector" by writing $G_C=\{N,E,C\}$ or $G_{PC}=\{N,E,P,C\}$ where the elements of C are in one-to-one correspondence with the elements of E, and each element of C consists of a vector $(c_1,c_2,...c_m)$ of some number of sub-elements m. Each sub-element of such a vector may represent different attributes of an edge, such as an edge length, change in coordinate variables, cost, or some alternative metric. Figure 3-3 illustrates the relationship between the sets N, E, P, and C and Figure 3-4 provides a simple example of an augmented graph G_{PC}. In this case, each element of C is a scalar, and might, for example, represent a conductance or resistance between nodes in a resistive network.

In some cases, it may be desirable to define a directed graph where the edge (i,j) is different than the edge (j,i). Then it would be possible to define different values of the cost vector for each direction of traversal of an edge while maintaining the one-to-one correspondence between elements of E and C. Examples where directed graphs might be necessary include modeling non-symmetric communication channels, in which the transmit and receive capabilities of one node are different than the transmit and receive capabilities of another node, or in which the channel characteristics are anisotropic. If nodes represent agents, then directed graphs might be a useful model for agent interaction opportunities when agent heterogeneity is present.

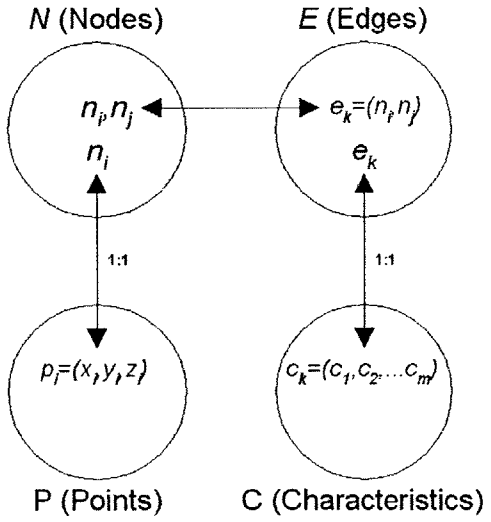


FIGURE 3-3. Structure of an augmented graph $G_{PC}=\{N,E,P,C\}$.

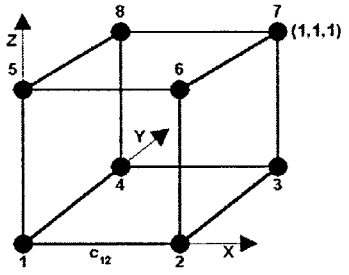


FIGURE 3-4. Example of an augmented graph $G=\{N,E,P,C\}$. Here $N=\{1,2,...8\}$, $E=\{(1,2), (2,3), (3,4), (4,1), (1,5), (2,6), (3,7), (4,8), (5,6), (6,7), (7,8), (8,5)\}$, $P=\{(0,0,0), (1,0,0), (1,1,0), (0,1,0), (0,0,1), (1,0,1), (1,1,1), (0,1,1)\}$, and $C=\{c_{12}, c_{23}, c_{34}, c_{41}, c_{15}, c_{26}, c_{37}, c_{48}, c_{56}, c_{67}, c_{78}, c_{85}\}$.

3.1.2 Basic Graph Properties

The number of nodes in a graph G is called its order, and can be denoted by $|G|$. The number of edges in a graph is denoted by $\|G\|$. Only graphs with finite orders (finite graphs) will be considered here. A graph is called complete if all nodes are pairwise adjacent, and is called independent if no two nodes are adjacent.

The average degree or valency of G is given by:

$$d(G) = \frac{1}{|N|} \sum_{n \in N} d(n) \qquad \text{(EQ 3-1)}$$

where $d(n) = |E(n)|$, the number of edges at node n. The minimum and maximum degrees of G are respectively given by:

$$\delta(G) = min\{d(n)|(n \in N)\}, \quad \textbf{(EQ 3-2)}$$

$$\Delta(G) = max\{d(n)|(n \in N)\}. \quad \textbf{(EQ 3-3)}$$

The ratio $|E|/|N|$, the number of edges of G per node, is often denoted $\varepsilon(G)$ and can be related to $d(G)$ by noting that in computing the average degree each edge is counted twice:

$$\varepsilon(G) = \frac{|E|}{|N|} = \frac{1}{|N|}\frac{1}{2}\sum_{n \in N} d(V) = \frac{1}{|N|}\frac{1}{2}d(G)|N| = \frac{1}{2}d(G) \quad \textbf{(EQ 3-4)}$$

It can be shown that every graph G with at least one edge has a subgraph H with $\delta(H) > \varepsilon(H) \geq \varepsilon(G)$. In words this means that "every graph G has a subgraph whose average degree is no less than the average degree of G, and whose minimum degree is more than half its average degree." [Diestel, 1997, p. 5]. Practically, this provides a simple approach to establish a minimum level of connectedness for some subset of G (for example, to evaluate functional redundancy).

3.1.3 Special Graphs

Path and cycle. A path is a non-empty graph for which $N = \{n_0, n_1, \ldots, n_k\}$ and $E = \{(n_0, n_1), (n_1, n_2), \ldots (n_{k-1}, n_k)\}$ where the n_i are distinct. The length of a basic path is the number of edges (k) and is denoted by P^k. A cycle is a path for which $k \geq 2$ and for which $n_k = n_0$. The length of a cycle is the number of edges in the cycle (k) and is denoted by C^k. Figure 3-5 provides a simple example of a path and a cycle. A *walk* is a sequence of nodes similar to a path but without the requirement that each node is distinct.

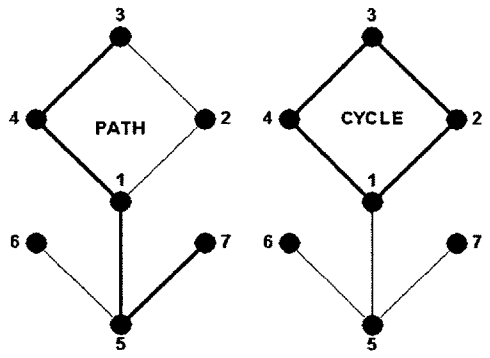


FIGURE 3-5. A P^4 path and a C^4 cycle on a simple graph $G=\{N,E\}$. The end-nodes of the path are n_3 and n_7, and the inner nodes of the path are n_4, n_1, and n_5.

Forest and tree. A graph G without cycles is a *forest*. If G is connected then G is also a *tree* (Figure 3-6). *Leaves* are nodes in a tree that have a degree of one. Removing any edge e from G results in a disconnected graph $G - e$. Several basic equivalent assertions about a graph T can now be stated [adapted from Diestel, 1997, p. 12-13]:

- T is a tree.
- Any two vertices x and y of T are linked by a unique path (written as xTy) in T.
- T is minimally connected, i.e. T is connected but $T - e$ is disconnected for all edges $e \in T$.
- T is maximally acyclic, i.e. T contains no cycle but $T + xy$ does, for any two non-adjacent vertices $x, y \in T$.

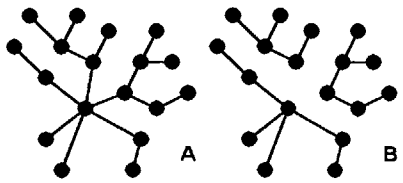


FIGURE 3-6. (A) depicts a small tree. (B) depicts a forest of three trees.

An important result of these equivalences is that every connected graph G contains at least one spanning tree. Any minimally connected subgraphs $G_i \subseteq G$ will be a tree. Trees are relevant to information flow in a distributed system in that they are, in many cases, an effective structure for routing information over a network.

3.1.4 Graph Connectivity

Several graph connectivity parameters have been defined, including the degree of a graph, the edge to node ratio, and the minimum and maximum degrees of a graph. One might also define metrics based on the ratio of nodes with $|G(n)| \geq k$ to the total number of nodes. Technically, a graph is called connected if any two nodes in the graph G are linked by a path in G. A graph is k-connected if there exist k node-disjoint paths between any pair of nodes (the paths share no nodes except for the end-nodes). The subset of a graph that is k-connected as a function of k is an important connectivity metric. The greatest integer k for which G is k-connected is called the connectivity of G, and is denoted $\kappa(G)$. Alternatively the connectivity can be thought of as the minimum number of vertices which can be removed to create a disconnected graph. A graph is l-edge-connected if one can create a disconnected graph by removing, at a minimum, l edges. L-edge-connectivity is denoted by $\lambda(G)$. A summary of these few basic graph connectivity metrics is given in Table 3-1. Any statistic based on the degree of a node would provide an additional connectivity metric: for example, variance or kurtosis (2nd or 3rd moments) of the degree distribution might be useful if the parameter of interest was not the absolute connectivity but was a characteristic of the distribution of connectivity.

In real exploration system scenarios it would also be useful to evaluate what functional subsets of a graph are k-connected because different parts of a distributed system may have different functions: this would allow analysis of functional redundancy. For large or dense graphs in a physical context (i.e., augmented graphs with $G_P=\{N,E,P\}$), analysis of connectivity as a function of spatial location may be valuable.

TABLE 3-1. Graph connectivity metrics

Description	Metric
Mean Degree	$d(G) = \frac{1}{\lvert G\rvert}\sum_{n \in G} d(G(n))$
Edges per node	$\varepsilon(G) = \frac{1}{2}d(G)$
Minimum Degree	$\delta(G) = min\{d(n) \mid (n \in G)\}$
Maximum Degree	$\Delta(G) = max\{d(n) \mid (n \in G)\}$
Proportion of nodes with at least k-neighbors	$\rho_k = \frac{1}{\lvert G\rvert}\sum_{n \in G} p(n)$ with $p(n) = \begin{cases} 1, d(G(n)) \geq k \\ 0, d(G(n)) < k \end{cases}$
k-connected	$\kappa(G) = k_{max}$ with $\lvert G\rvert > k$ and $\lvert X\rvert < k$ for all $X \subseteq N(G)$
l-edge-connected	$\lambda(G) = l_{max}$ with $\lvert G\rvert > 1$ and $G - F$ connected for all $F \subseteq E(G)$ with $\lVert F\rVert < l$

3.1.5 Flows and Networks

Flows. A flow f describes the movement of some quantity (information, for example) from one point to another. The expression $f(x, y) = k$ denotes the transport of k units (of something) from node x to node y. The transport of k units (of something) from node y to node x is denoted by $f(y, x) = -k$ so that $f(x, y) = -f(y, x)$ for adjacent nodes x and y.

Circulations. To describe flows and circulations, one must differentiate between an edge xy and an edge yx, and $G=\{N,E\}$ is now technically a multigraph (multigraphs also allow loops, represented by writing an edge as xx or yy). Often this is done by writing (e,x,y) and (e,y,x) to indicate edges from x to y and y to x, respectively: we will write $e_{xy} = (e, x, y)$ and $e_{yx} = (e, y, x)$ to simplify the standard notation. f is a circulation on G if [adapted from Diestel, 1997, p. 124]:

- $f(e_{xy}) = -f(e_{yx})$ for all $e_{xy} \in E(G))$ with $x \neq y$
- $f(n, N(G)) = 0$ for all $n \in N(G)$.

These results require that:

$$f(X, X) = 0 \text{ for all } X \subseteq N(G), \qquad \textbf{(EQ 3-5)}$$

$$f(X, N(G)) = \sum_{x \in X} f(x, N(G)) = 0 \qquad \textbf{(EQ 3-6)}$$

which together imply that for a circulation the flow across any cut is zero. This can be stated as:

$$f(X, \bar{X}) = 0 \text{ for all } X \subseteq N(G). \qquad \textbf{(EQ 3-7)}$$

where $\bar{X} = V \backslash X$ denotes the complement of a node set $X \subseteq N(G)$.

Networks. A network that models information flow from one source node s to a target node t is defined as the tuple (an ordered set of values) $\Gamma = (G, s, t, c)$ where $G=\{N,E\}$ is a multigraph, $s, t \in N(G)$, and c is a capacity function on G, defined independently for the two directions of an edge. A flow f on the network Γ must satisfy [Diestel, 1997, p. 126]:

- $f(e_{xy}) = -f(e_{yx})$ for all $e_{xy} \in E$ with $x \neq y$
- $f(n, N(G)) = 0$ for all $n \in V\backslash\{s,t\}$
- $f((e_{xy}) \leq c(e_{xy}))$ for all $e_{xy} \in E(G)$

With $S \subseteq N(G)$ such that $s \in S$ and $t \in \bar{S}$, the capacity of the cut $(S, \bar{S})$ is denoted by $c(S, \bar{S})$. With a single source s, every such cut satisfies:

$$|f| = f(S, \bar{S}) = f(s, N(G)) \qquad \textbf{(EQ 3-8)}$$

Therefore for every network the maximal value of a flow is equal to the minimum capacity of a cut. In the case of distributed elements with communications channels of different capacities that are functioning as an information pipeline from one location to another, a

rearrangement of elements may increase the minimum capacity of the cut and enable a higher maximal rate of information transfer.

3.1.6 Graph Theory and Multi-Agent Systems

A graph can be used to represent opportunities for interaction between agents in a multi-agent system. Although not implemented herein, a multi-agent simulation of a distributed system, where the interactions between surface topography and the system elements (agents) determine the graph structure, could be utilized to add realism and accuracy to the modeling of distributed architectures for Mars surface exploration. A multi-agent system environment would enhance a distributed architecture simulation by allowing a designer to define node characteristics and behavior, such as movement of a node based on local conditions or internal agent goals, and would allow investigation of the effects of node (agent) heterogeneity on the structure of the distributed system. Node behaviors are modeled, in a very limited extent, in Section 3.6 without utilizing a multi-agent system architecture. Chapter 5 returns to the issue of node behaviors and explores opportunities for how a distributed system can support traverse planning and execution by particular system elements (robotic and human agents).

3.2 Surface Modeling and Analysis

Modeling or analyzing a surface environment is an extremely difficult proposition: important factors relating to the functioning of and interfaces between system elements may include gravity, topography, surface characteristics (soil dynamics, for example), temperature, pressure, and the local radiation environment. Accurately modeling communication channel characteristics between physically separated elements might also require additional model parameters to characterize phenomenon such as multipath (both beneficial and detrimental), channel fading, and noise.

Surface modeling and analysis in this thesis focuses on topography. Gravity and some details of surface characteristics will be addressed in Chapter 5, but other important elements of environment modeling will not be discussed. Topographic modeling does permit several important questions to be addressed: topography can help answer questions about how the structure of the landscape affects the interfaces between elements of the surface segment of an exploration mission. While topography in and of itself cannot be used to determine what portion of a surface is navigable to different types of system elements, it can bound the navigability problem by permitting analysis of terrain features (such as slopes), identifying potential navigable routes, and permitting an analysis of where a

system element may go while still maintaining connectivity with other surface system elements.

The basic representation of planetary surface topography used is shown in Figure 3-7. A matrix Z represents a height field, where each matrix element z_{ij} represents a sample of the altitude at a point. Matrix indices i and j represent the position of an altitude sample on the surface of some geoid. A standard right-handed cartesian coordinate system with axes x, y, and z will be used to represent position in the East, North, and Zenith directions, respectively. Using a constant spacing between sample altitudes in both the x and y directions, the position of a point $z_{ij} = Z(i, j)$ can be written in terms of the spacing between samples w (constant in both the x and y directions) as:

$$\underline{p} = (p_x, p_y, p_z) = (wj, wi, z_{ij}) \qquad \textbf{(EQ 3-9)}$$

for zero-based indices i and j. This representation limits the scale and type of surfaces that can be represented (no vertical overhangs can be represented, for example). However, this representation is extremely simple and is sufficient for describing a subset of the surface of any approximately spheroidal planet with a radius much greater than the scale of the surface that is being represented: this ensures that over the area of the surface of interest, the reference geoid is approximately flat. Figure 3-8 illustrates the relationship between the curvature of a surface and the height above a surface required to maintain line-of-sight visibility.

Other coordinate systems could certainly be used, but a consistent cartesian coordinate system is used throughout this thesis for simplicity. Integration of an existing geographical information systems software package such as ESRI's ArcView GIS™ with a multi-agent simulation package and additional software tools would provide a more robust distributed architecture modeling platform, and would vastly increase the possible projections and models available for surface representations (most of the software developed for this project has been written in MATLAB™ because of its ease of use and rich library of supporting functions).

3.2.1 Scale of Interest

This thesis is concerned with representing surfaces at the length scales at which elements of distributed architectures for Mars surface exploration might operate on a day-to-day basis. As illustrated in Figure 3-9, typical traverse planning and execution activities might tend to occur on a scale from meters to tens of kilometers.

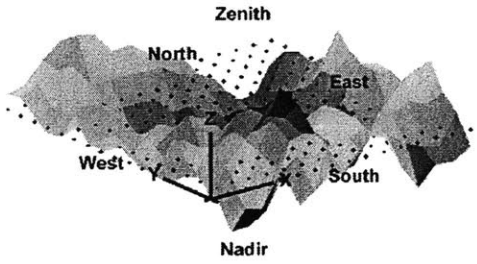


FIGURE 3-7. Cartesian surface model.

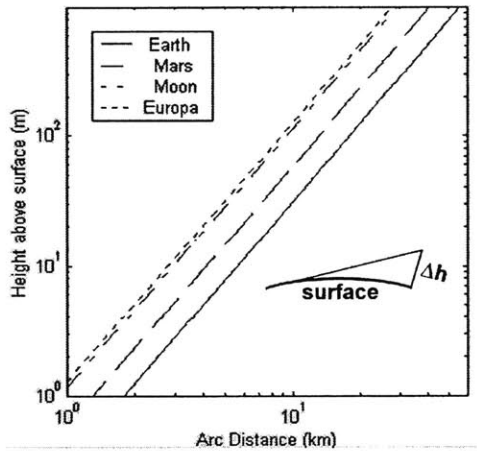


FIGURE 3-8. Relationship between surface arc distance and the height above the surface required for line of sight visibility to the opposite side of the surface, plotted for Earth, Mars, the Moon, and Europa. A spherical smooth surface is assumed. Line-of-sight visibility calculations based on a cartesian surface model will be based on Δh values of tens of meters for surfaces with dimensions larger than 10 km.

TABLE 3-2. Surface curvature for Earth, Mars, the Moon, and Europa

Planet	Radius	Curvature*
Earth	6378 km	0.008983 deg/km
Mars	3397 km	0.01687 deg/km
Moon	1737 km	0.03299 deg/km
Europa	1565 km	0.03661 deg/km

*based on equatorial radius

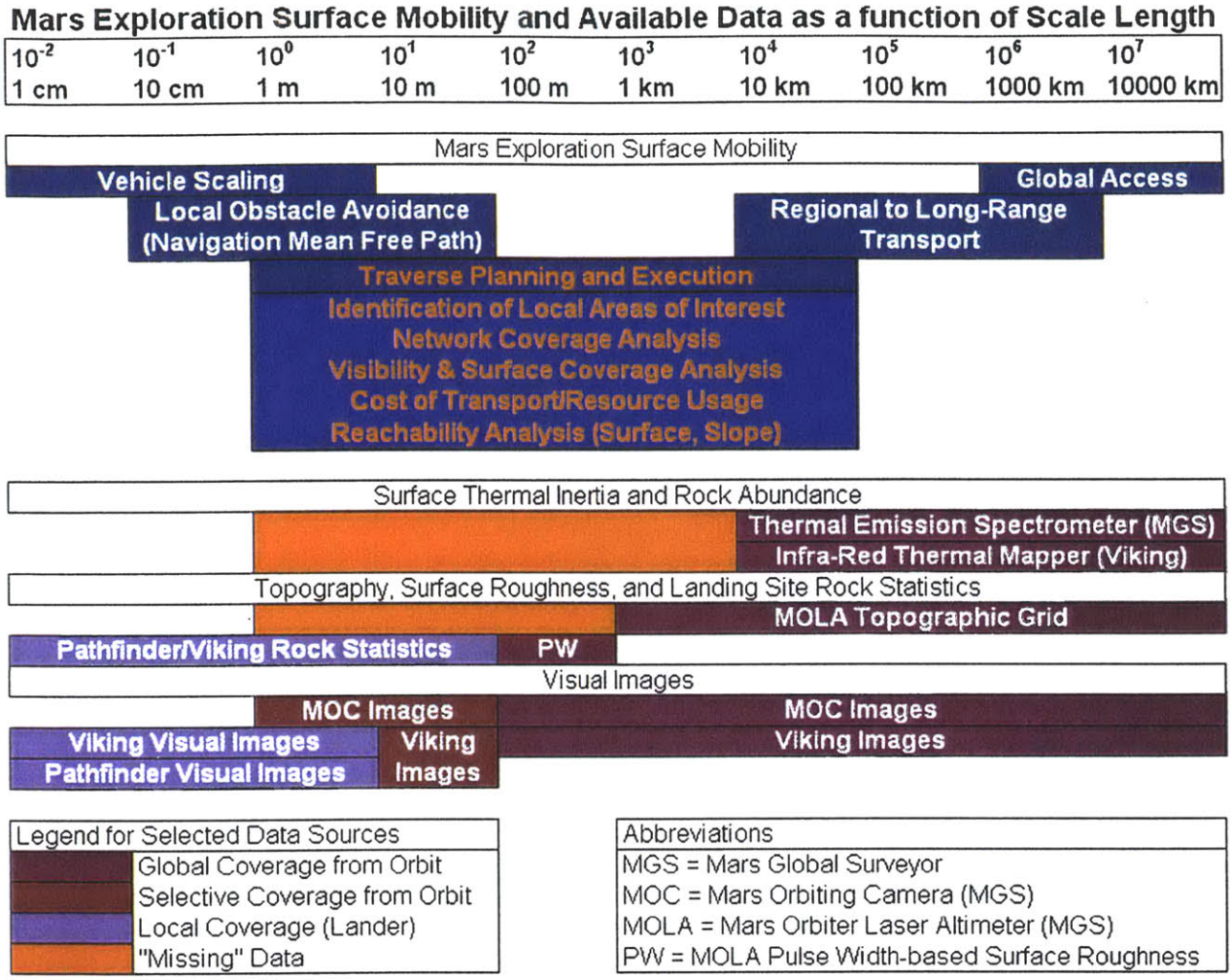


FIGURE 3-9. Mars exploration surface mobility activities and supporting data as a function of scale length: The range from 1 m to 10 km is critical for traverse planning and execution for mobile elements of a surface exploration system. Selected data sources from Mars Global Surveyor and Viking are shown where scale length indicates approximate horizontal spatial resolution of the data. Characterization of the Martian surface at the scale lengths critical for traverse planning requires additional topographic and thermal inertia data at higher spatial resolutions. While high resolution images from the Mars Orbiting Camera have identified many potential future local areas of interest, it is challenging to use these images in a quantitative way to support traverse planning.

The same techniques developed here might be used to model connectivity between a group of surface elements collectively spread over a greater scale than tens of kilometers given either a non-cartesian surface representation or a method of computing line-of-sight visibility between elements that compensates for surface curvature.

Unfortunately, a gridded topographic data set with spatial resolution better than 1 km is not yet available: the current highest resolution topographic data set has a resolution of 1/32 by 1/64 of a degree (corresponding to about 1 by 2 km at the equator). Individual altitude samples from the Mars Global Surveyor Mars Orbiting Laser Altimeter have a maximum spatial resolution (in the plane of the geoid surface) of about 168m, and global root-mean-square surface roughness data (from optical pulse-width measurements) of this spatial resolution are also available [Smith et al., 2000]. To better characterize dust deposits and areas of high rock abundance that may be hazardous or challenging for robotic agents to navigate,

additional thermal inertia data at higher spatial resolutions will also be required.

Until higher spatial resolution topographic data is available, surface environment models may need to rely on one or more of the following options:

- Model terrain only at low spatial resolutions (unacceptable).
- Painstakingly reconstruct local terrain from high resolution photographs (possible, but a significant challenge).
- Utilize analog high resolution topography data from Earth (for example) as representative terrain after comparing Earth and Mars data at similar scales, and understanding the limitations of the comparison.
- Generate representative terrain by matching topographical statistical measures at scales for which those statistics are known for Mars and apply geomorphological scaling laws to extend the representative terrain to higher spatial resolutions.

The last two approaches are adopted here. It must be stated up front that the results should not be interpreted as actually representing Martian terrain except in the sense that the surfaces are statistically consistent (to the extent possible) with existing topography data and with present understanding of geomorphological scaling laws.

3.2.2 Power Spectral Density-based Terrain Generation

The statistical nature of topography has important geomorphological origins that relate to the depositional or erosional environment of the landscape [Dodds and Rothman, 2000]. Many existing software packages such as the ArcView GIS Geostatistical Analyst provide terrain generation and statistics packages. However, for the purposes of this thesis, simple power spectral density-based terrain generation was adopted. Exponential scaling of the power spectral density of terrain can be observed over several orders of magnitude of scale length. This scaling relationship can be expressed as:

$$PSD(f) \propto af^{-\beta} \qquad \textbf{(EQ 3-10)}$$

where *PSD* is the power spectral density, a is some constant of proportionality, f is the spatial frequency, and β is the power law scaling exponent. Figure 3-10 illustrates this scaling relationship for the northern lowlands and the heavily cratered southern highlands of Mars at scales from 1 km to 1000 km. Many transforms can be used to estimate the power spectral density (or to transform the power spectral density from the frequency to the time domain): the two-dimensional discrete fourier transform (denoted *dfi2*) and its corresponding inverse transform (denoted *idfi2*) will be used here (see Appendix A for detailed definitions of these transforms). The rela-

tionship between a spatial domain field $x(n_1, n_2)$ and its two-dimensional discrete fourier transform $X(k_1, k_2)$ is written:

$$x(n_1, n_2) \leftrightarrow X(k_1, k_2) \qquad \textbf{(EQ 3-11)}$$

where

$$x(n_1, n_2) = idft2(X(k_1, k_2)) \qquad \textbf{(EQ 3-12)}$$

and

$$X(k_1, k_2) = dft2(x(n_1, n_2)) . \qquad \textbf{(EQ 3-13)}$$

A height field can then be generated by the following procedure:

- Assume a power spectral density exponential scaling relationship (the spectrum scales as the square root of the power spectral density).
- Construct a two-dimensional symmetric spectrum X using the scaling law as the radial profile (so that the radial profile of the spectrum scales as $f^{-\beta/2}$). Radial symmetry of the power spectral density is a sufficient but not necessary condition for a real height field.
- Compute the height field $x = idft2(X)$ using the inverse transform of the generated spectrum.
- Scale the height field or power spectrum as necessary to achieve the desired range of altitudes in the topography.

Figure 3-11 shows several generated height fields, for different values of the power law scaling exponent β.

To estimate the power law scaling exponent for real terrain, it is necessary to first compute the spectrum of the terrain and derive a representative radial profile for the power spectral density. Because the spectrum of a real field is unlikely to be radially symmetric, no true radial power spectral density profile exists. Appendix A discusses how a representative radial power spectral density profile can be estimated for both radially symmetric and non-radially symmetric power spectral densities. In general, sampling the power spectral density along its diagonals provides a reasonable estimate of a representative radial power spectral density profile for non-radially symmetric power spectral densities.

Figure 3-12 shows a real height field, its power spectral density, and its representative radial power spectral density. This height field is from a region of the Cottonwood, Nevada quadrangle 30 meter resolution digital elevation model, and was the primary field area for the geologic mapping project that is described in Chapter 4.

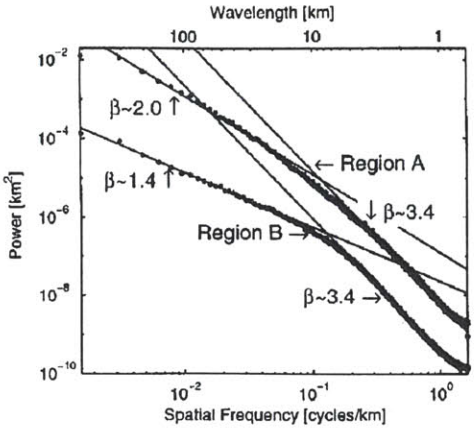


FIGURE 3-10. Martian average power spectrum of the topography of an area in the heavily cratered terrain (Region A) and of an area in the northern lowlands (Region B). Slopes indicate power law exponents. For short wavelengths the effective exponents are similar, while in the long wavelengths the exponent of Region B is lower than that of Region A. From [Aharonson et al, 2000].

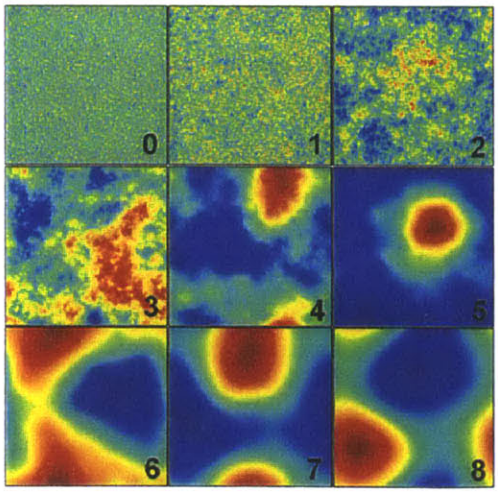


FIGURE 3-11. These generated height fields are colored by altitude, and the power law scaling exponent is shown in the lower right corner of each height field. The actual altitudes and dimensions of the height fields are unimportant here. Notice that for high values of the power law scaling exponent, low spatial frequencies dominate (as is expected). Height fields can be generated using the *surface_create.m* MATLAB function, described in Appendix B.

3.2.3 Altitude Distribution of Topography

In addition to matching a power spectral density law of terrain scaling, one might seek to match the altitude distribution of topography. As one might expect, these two topographic characteristics are related. A power spectral density scaling law (stochastically, when random phases are used for the time domain reconstruction) determines the altitude distribution of a height field. For real terrain, the best-fit power law scaling exponent based on the representative radial power spectral density or based on the resultant altitude distribution may be different.

To demonstrate the relationship between the power spectral density scaling law and the altitude distribution, a simple experiment can be performed:

- Compute the altitude distribution for a real height field and estimate β from the power spectral density.
- Choose a series of values $\{\beta_1, \beta_2, \ldots, \beta_n\}$ near β and for each β_i generate m height fields.
- Compute the altitude distribution for each height field, and compute the cross-entropy (see Appendix A) between the altitude distributions of the height field and the real height field.
- For each β_i, compute the mean cross-entropy.

The value of β_i for which the mean cross-entropy is a minimum defines a best-fit metric for the altitude distribution, according to the principle of minimum cross-entropy (see Appendix A). Cross-entropy is a measure of directed divergence: minimization of cross-entropy provides a criterion for minimizing the divergence between two distributions. This β_i should be approximately equal to β.

Performing this experiment for the real height field encountered in Figure 3-12 yields a best-fit value of $\beta \approx 4.3$. Figure 3-13 compares the altitude histogram of the real field to the altitude histogram of generated fields as a function of β. Figure 3-14 illustrates the results of computing the mean cross-entropy for each β_i.

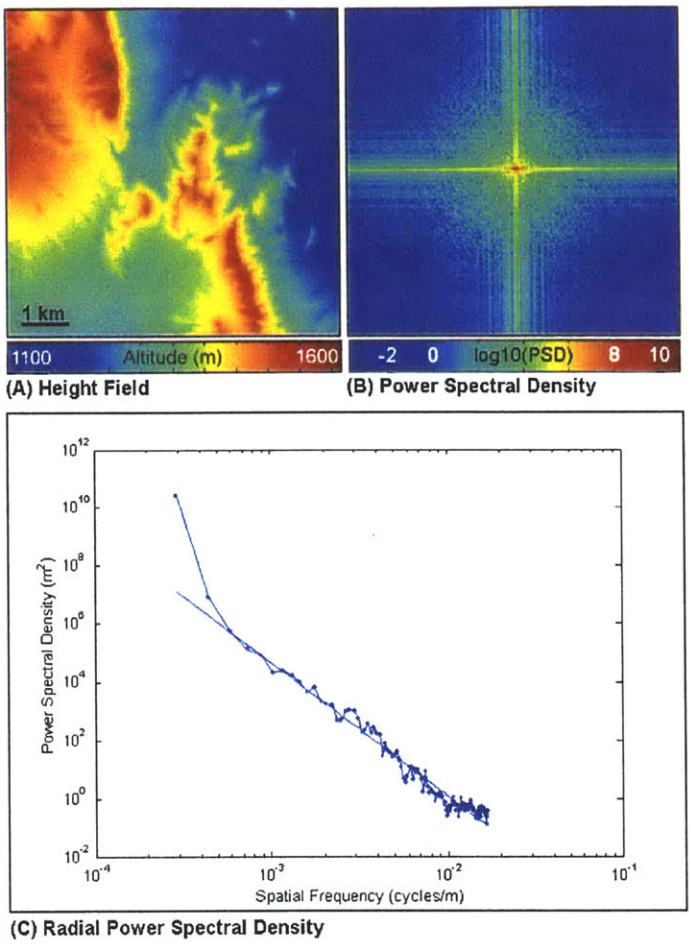


FIGURE 3-12. Height field representation and unnormalized power spectral density of a real surface. In (A), a region of the Cottonwood, Nevada quadrangle digital elevation model is colored by altitude. From the diagonals of the two-dimensional power spectral density (B), a representative power spectral density (C) has been derived. A least squares curve fit results in a power law scaling exponent of approximately 4.59 and a gain factor a of approximately 7.34×10^{-10}.

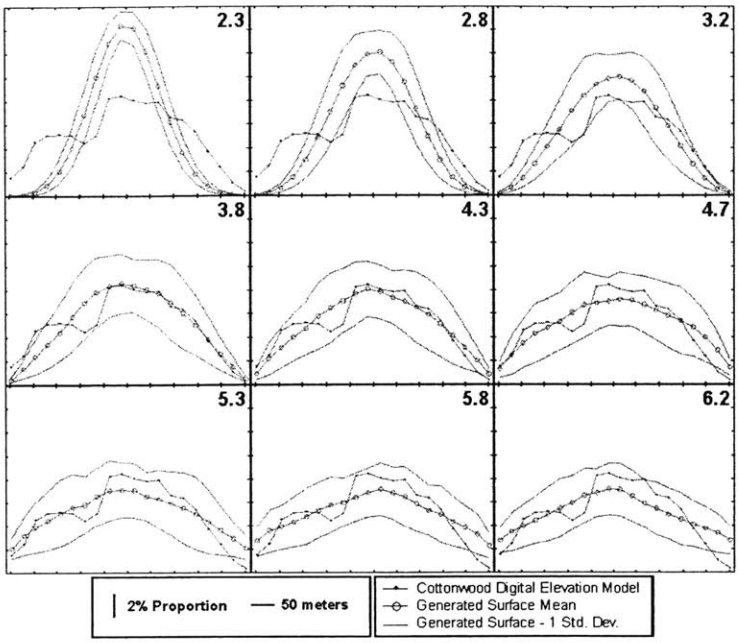


FIGURE 3-13. Altitude histogram for real and generated terrain as a function of the power law scaling exponent (shown in the upper right corner of each plot). The horizontal axis corresponds to altitude (shown in 50 m bins), and the vertical axis indicates the proportion of points on the surface in a given altitude bin. The real height field is a region of the Cottonwood, Nevada quadrangle digital elevation model. For each value of the power law scaling exponent, 100 surfaces were generated. A 67% confidence interval for the altitude distribution of the generated terrain is also shown.

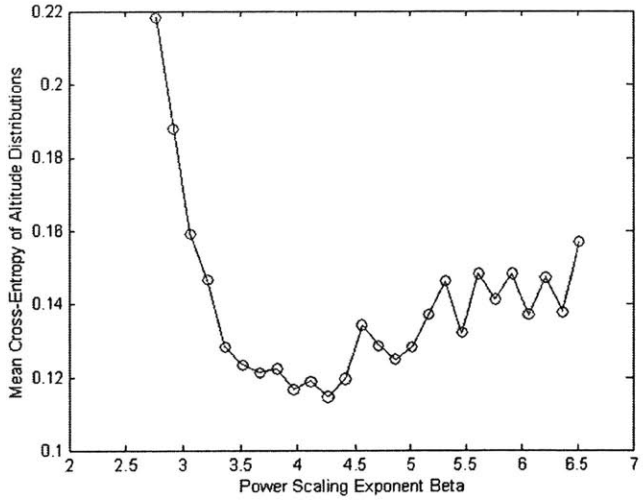


FIGURE 3-14. A plot of mean cross-entropy between the altitude distributions of the real height field and generated height fields as a function of the power law scaling exponent: According to the principle of minimum cross-entropy, the best-fit altitude distributions occur for a power law scaling exponent of approximately 4.3. To determine a more precise value of the best-fit power law, more trials would need to be run with power scaling exponents near 4.3. In this case, for each value of the power law scaling exponent, 100 different surfaces were generated and analyzed.

3.2.4 Correlative Statistics

[Dodds and Rothman, 2000] introduce two important correlative statistics for height fields called the height-height correlation function and the height autocorrelation function. These two statistics are functions of scale length, but are also dependent upon direction. Therefore they can provide a measure of isotropy or anisotropy of height field statistics. The height-height correlation function is a measure of the root-mean-square height fluctuation over some scale

length and in some direction. For a height field $h(\underline{x})$, and can be written:

$$C(\underline{r}) = \langle |h(\underline{x}+\underline{r}) - h(\underline{x})|^2 \rangle_{\underline{x}}^{1/2} \quad \textbf{(EQ 3-14)}$$

where $\hat{r}$ gives the direction of interest and r is the scale length of interest. The height autocorrelation function is similarly defined as:

$$C_a(\underline{r}) = \langle |h(\underline{x}+\underline{r}) \cdot h(\underline{x})|^2 \rangle_{\underline{x}}^{1/2}, \quad \textbf{(EQ 3-15)}$$

and can be used to measure the scale of correlations in a height field, and their directional dependence. [Dodds and Rothman, 2000] demonstrate that the height autocorrelation function scales as:

$$C_a(r) \propto e^{-r/\alpha} r^{-\beta} \quad \textbf{(EQ 3-16)}$$

where r represents the scale length, α is a measure of the extent of correlations in the surface (called the correlation length) and β is a power law exponent.

Although only height fields with isotropic statistics will be generated herein, characterizing the extent to which a real height field is anisotropic is an important step in creating representative terrain. In addition, understanding the directional dependence of surface statistics may be important for navigation: this use of these correlative statistics will be implemented in Chapter 5.

3.2.5 Roughness and Slope

While the height-height correlation function does provide a measure of surface roughness, there are other methods that can be used to analyze surface roughness and surface slopes. [Aharonson et al., 2000] covers several methods for estimating the surface slope along a one-dimensional track, including root-mean-square slope, median slope, and inter-quartile scale slope. Median slope and inter-quartile scale slope estimates are less sensitive to outlier slopes, and do not require the assumption of a gaussian slope distribution as does the root-mean-square slope statistic (Mars Orbiting Laser Altimeter pulse-width data provides an estimate of root-mean-square surface roughness). [Kreslavsky and Head, 1999] also favor median slope over root-mean-square slope because it is not influenced by small numbers of outliers at the upper end of the slope-frequency distribution tail; they also suggest that median slope is more sensitive than inter-quartile scale slope at smaller length scales. [Kreslavsky and Head, 2000] introduce median differential slope as a technique to exclude slopes due to larger features. They also examined the

relationship between kilometer scale surface roughness and different geologic units.

[Burton and Yoha, 1996] briefly summarize several slope algorithms for grids with evening spaced points [Burton and Yoha, 1996, section 2.2.3.1.2]. Two methods of note include:

- Estimation of a surface normal vector from each set of two-by-two points.
- Computation of the maximum slope between a center point and its eight neighbors using a weighted or unweighted centered-difference.

A weighted centered-difference approach will be used here. Figure 3-15 illustrates one weighted centered difference approach, from [Lewicki and Zong, 1999, p. 29], where the slope is estimated from the average elevation changes per unit distance in the x and y directions:

$$\Delta x = \frac{[(a+2d+g)-(c+2f+i)]}{8w} \qquad \textbf{(EQ 3-17)}$$

$$\Delta y = \frac{[(a+2b+c)-(g+2h+i)]}{8w}. \qquad \textbf{(EQ 3-18)}$$

The slope s can then be computed from:

$$s = \operatorname{atan}(\sqrt{(\Delta x)^2 + (\Delta y)^2}) \qquad \textbf{(EQ 3-19)}$$

The frequency and spatial distribution of surface slopes is intimately related to the reachability of the terrain by surface elements and to the cost of transport for surface elements.This use of statistics for slope and surface roughness characterization of surfaces will be discussed further in Chapter 5.

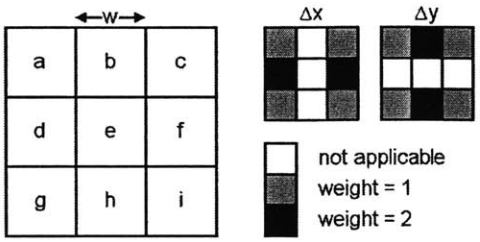


FIGURE 3-15. Visual illustration of weighted centered-difference slope algorithm for surfaces represented by grids of evenly spaced points. Here, w represents the spacing between grid points, and the other letters represent the mean altitude for a given cell (the altitude of a grid point). The altitudes of the surrounding cells are differenced with the weighing factors given by the colored squares.

3.2.6 Comparison of Two Digital Elevation Models

During the design of surface planetary exploration missions, multiple sites are generally considered for system deployment, and a range of surface types are considered to evaluate the potential system performance in different environments. Now that several approaches have been developed to characterize surfaces it is possible to compare two different digital elevation models in a number of ways. One is a region of the Cottonwood Quadrangle, Nevada digital elevation model, which was introduced earlier. The second surface is a high-resolution digital elevation model of Crater Lake, California. This second surface covers an area of roughly four times the first, and has a 10 meter horizontal spatial resolution (versus the 30 meter horizontal spatial resolution of the Cottonwood region).

There are several reasons to utilize the Crater Lake digital elevation model here: It is a relatively high spatial resolution as compared to other publicly available digital elevation models, and the coverage area is large enough to model traverses at a scale reasonable for day-to-day operations of planetary explorers exploring a local area. In addition, the crater and fluvial features made the Crater Lake digital elevation model similar in some regards to areas that might be explored on the Martian surface (admittedly, many of the features that will likely be explored on the Martian surface are significantly larger in scale).

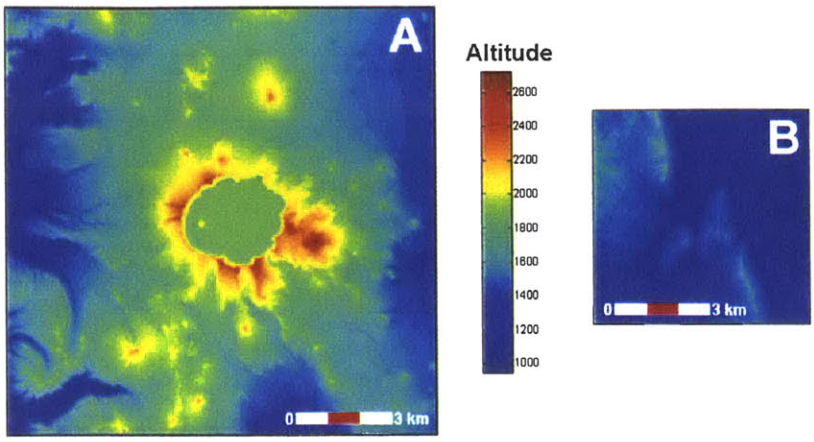


FIGURE 3-16. Altitude characteristics of two digital elevation models: In (A) Crater Lake and (B) a region of the Cottonwood, Nevada quadrangle, the digital elevation models are colored by altitude in meters.

Figure 3-16 plots the two digital elevation models colored by altitude. Although the Crater Lake area covers a much wider altitude range than the Cottonwood area, both have very similar representative radial power spectral densities, as shown in Figure 3-17.

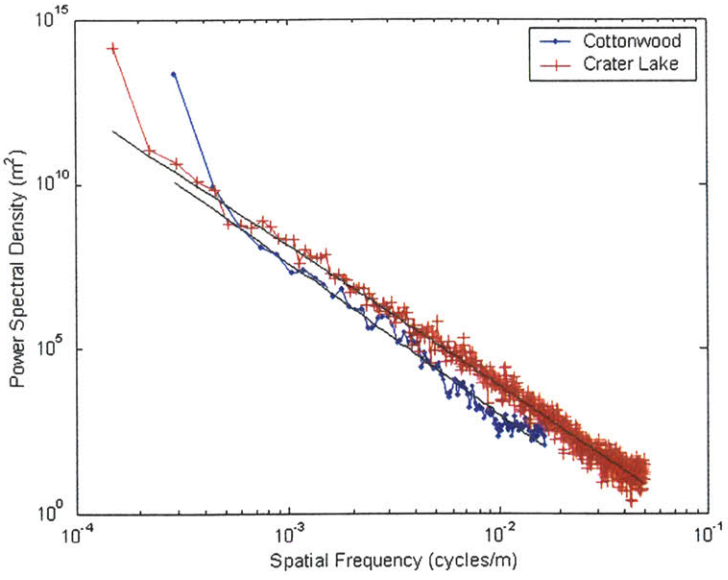


FIGURE 3-17. Representative normalized radial power spectral densities for the Crater Lake and Cottonwood digital elevation models: A least-squares fit for the Crater Lake power spectral density relationship yielded an approximate gain factor a of 2.4493×10^{-5} and a power law scaling exponent of 4.24. The Cottonwood power spectral density parameters were estimated to be 6.67×10^{-7} for a and 4.59 for the power law scaling exponent.

One might expect that with such similar power spectral densities, the altitude histograms of the digital elevation models

might have similar profiles, but Figure 3-18 illustı is not the case.

When creating representative terrain, it may be desirable to match the altitude distribution more closely (using the cross-entropy approach) or to match the power spectral density scaling law relationship. Which method is more appropriate depends upon the purpose of the distributed system and the type of environment upon which it will be deployed. Matching the power spectral density scaling relationship gives a more accurate character to the representative terrain in terms of the self-similarity of topography. However, one must keep in mind that real topography is not often self-similar except over some range of length scales.

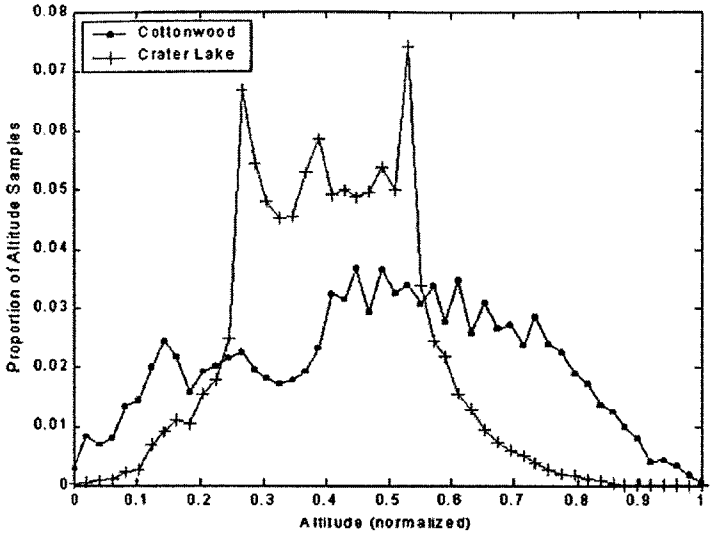


FIGURE 3-18. Altitude histograms of the Crater Lake and Cottonwood digital elevation models are shown with the respective altitudes normalized to [0,1] for ease of comparison.

In addition to limited self-similarity, real topography has anisotropy. Figure 3-19 illustrates anisotropy in the two regions by plotting the height-height correlation function as a function of scale length.

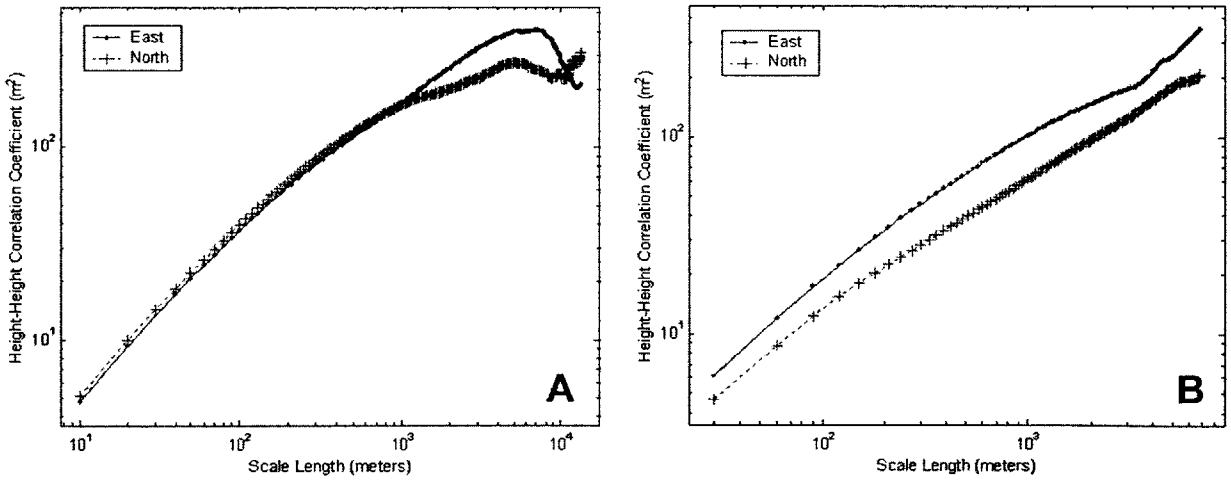


FIGURE 3-19. Height-height correlation functions in the E-W and N-S directions for the Crater Lake (A) and Cottonwood (B) digital elevation models.

Because the height-height correlation function is a measure of root-mean-square surface roughness one can conclude that the surface roughness below a scale length of about a kilometer is similar in the East-West and North-South directions for the Crater Lake region. Following the reasoning of [Dodds and Rothman, 2000, p. 599-600], the roughly parallel trends of the East-West and North-South height-height correlation function profiles in Figure 3-19 (B) imply that the Cottonwood topography is "quantitatively rougher, at all scales, and by the same factor, in the perpendicular direction [East-

West] than in the parallel direction [North-South]." This type of information might be utilized as a descriptive heuristic for a preferred general direction of travel in a given area for mobile system elements (e.g., to yield a reduction in energy expenditures in climbing and descending terrain).

The height autocorrelation functions for the Crater Lake and Cottonwood regions, plotted in Figure 3-20, illustrate that the topography is correlated over most of the scale lengths representable by the digital elevation models.

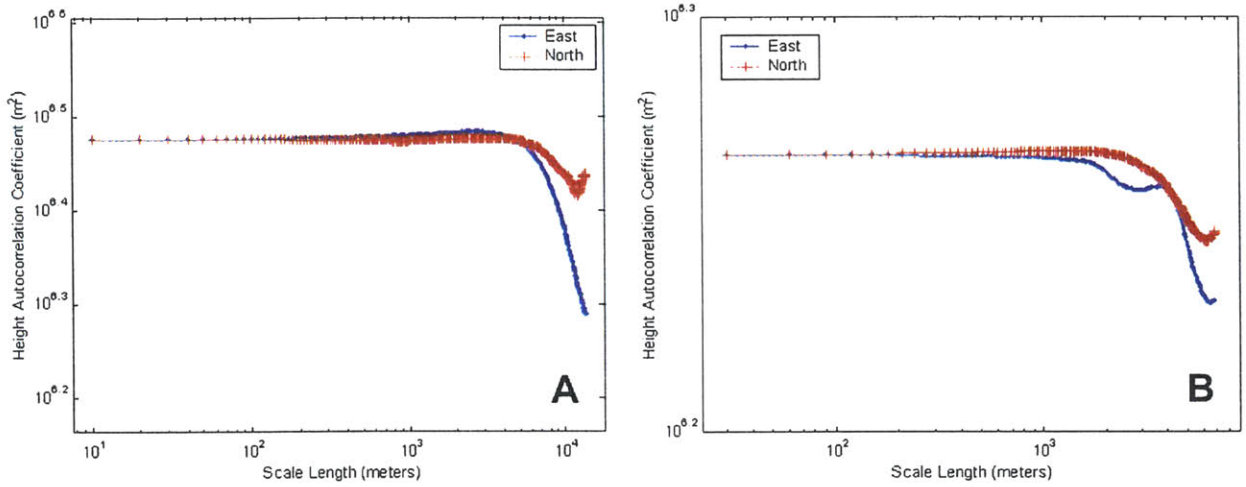


FIGURE 3-20. Height autocorrelation functions in the E-W and N-S directions for the Crater Lake (A) and Cottonwood (B) digital elevation models.

The slopes of the two regions are compared in Figure 3-21, using the previously described centered-difference slope algorithm. Most of the slopes are below 30 degrees, which is consistent with the typical range of angles of repose for the Earth of 30-40 degrees.

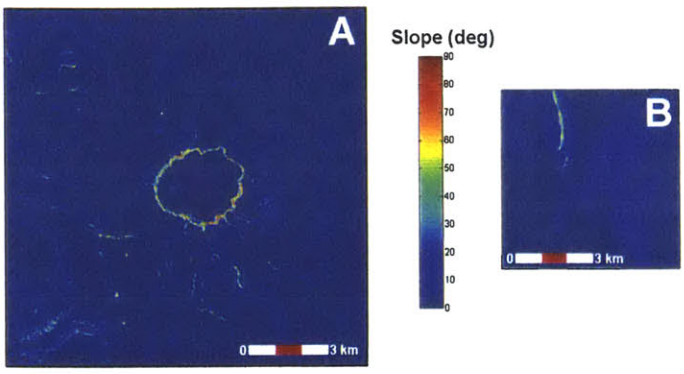


FIGURE 3-21. Slope characteristics of the Crater Lake and Cottonwood digital elevation models: In (A) and (B), the digital elevation models are colored by slope in degrees. Most of the slopes for both regions are below 30 degrees.

Slope histograms for the Crater Lake and Cottonwood regions are shown in 3-22. Both regions have similar exponential falloff in the proportion of altitude samples (or cells) with a given slope as the slope increases, but both regions are primarily composed of slopes

below 15 degrees, *as estimated by the limited resolution digital elevation models of the regions.*

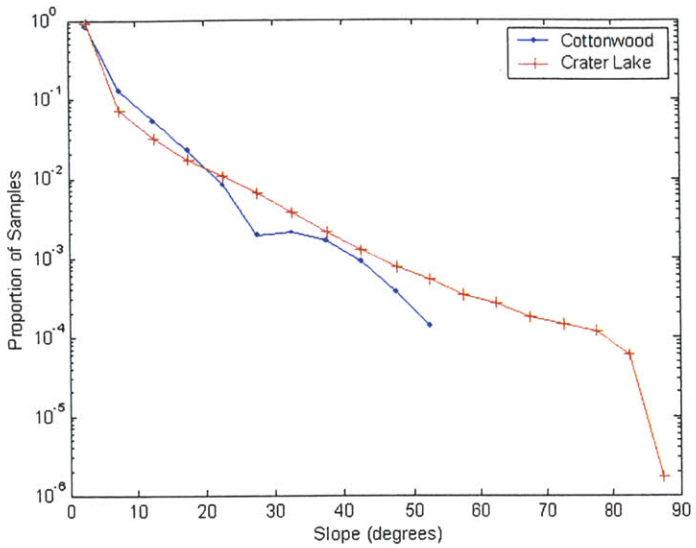


FIGURE 3-22. Slope histograms of the Crater Lake and Cottonwood digital elevation models demonstrate that the proportion of cells with a given slope could potentially be approximated by a linear function in log-log space. Ground truth in the Cottonwood region suggests that local areas of steep slopes (> 55 degrees) in that region are not captured by the 30 meter spatial resolution of the Cottonwood digital elevation model. The 10 meter resolution Crater Lake digital elevation captures slopes as high as 86 degrees.

Figure 3-23 illustrates the distribution of slopes for the Crater Lake and Cottonwood regions in a more visual fashion.

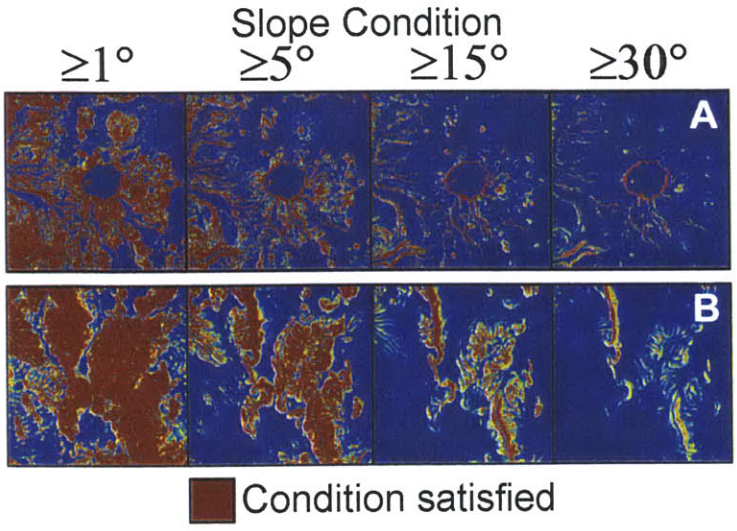


FIGURE 3-23. Slopes above 1, 5, 15, and 30 degrees are visually depicted for the Cottonwood (A) and Crater Lake (B) digital elevation models.

Slope fields can be used to generate slope-based accessibility maps that show what part of a surface is accessible to a system element given (1) some initial position of a system element and (2) slope constraints for movement of that system element. Combining slope-based accessibility maps for all of the elements of a distributed system can provide a system-level slope-based accessibility map. To

create a more general accessibility map for a distributed system, other constraints (such as surface characteristics, or network coverage) will clearly need to be taken into account.

This comparison between the Crater Lake and Cottonwood regions has facilitated this brief discussion about how real terrain may differ from generated terrain, and how surface characteristics might relate to the operation of distributed systems.

3.2.7 Limitations of Surface Modeling and Analysis

A power spectral density scaling law approach to creating artificial terrain was described, and approaches for characterizing topographic surface models were explored.

The major limitations of the power spectral density scaling law approach to terrain generation described herein include a lack of anisotropy and no model of correlations between surface features (random phases were used for reconstruction of terrain from a radially symmetric power spectral density). While these are serious limitations, generating terrain in this fashion does allow systematic variation of a height field that may be valuable in evaluating performance of distributed surface systems across a variety of environments.

The use of existing high resolution digital elevation models can provide a more realistic surface for developing operational scenarios. Systematic variation in the topographic statistics of existing digital elevation models can be accomplished, but is outside the scope of this thesis. Choosing a particular (existing or generated) digital elevation model (or a particular set of digital elevation models) with which to evaluate the performance of a distributed system is inherently a *problem of representative samples*, as described by [Cohen, 1995, p. 360] for the domain of artificial intelligence algorithms. In the domain of simulating distributed architectures, the same problem arises: what height field is truly representative? What are the characteristics that really matter?

Error sources in digital elevation models have not been discussed here, but are very important in understanding the relationship between any calculations based on surface models and any attempt to generalize those calculations to the real world. [Wood, 1996, *Chapter 3*] discusses uncertainty in digital elevation models and suggests several approaches for both assessing and visualizing uncertainty in digital elevation models.

In addition, the use of a cartesian surface model to approximate a curved surface limits the range of scales for which visibility computations (described in the following section) are valid, but is adequate for the purposes of the thesis.

As higher-resolution topography and surface characteristics data becomes available for the surface of Mars, the tools described above will become more powerful in terms of their ability to generate meaningful and representative results. Given the limited spatial resolution of the currently available data for the Martian surface, the most important aspect of this work is not the particular results from any one analysis, but the process by which the results were achieved.

3.3 Analyzing Visibility and Coverage

The idea of visibility and coverage analysis is simple. From the perspective of an agent in an exploration system, visibility analysis answers the question "what other agents can I see, and how well can I see them?" and coverage analysis answers the question "what region of the surface can I see, and how well can I see it?" At a system level, visibility analysis answers the question "what agents can see what other agents and how well?" and coverage analysis answers the question "what region of the surface can the agents collectively see, and how well can they see it?"

Visibility analysis is used here as an indicator of the ability of one agent to interact or communicate with another agent. Other metrics could certainly be used, but line of sight is a reasonable first order metric for interaction opportunities. The lack of an ionosphere on Mars supports the use of line of sight visibility as a measure of communication opportunities between agents, but this approach is extremely limited because it includes no model of interference (such as multipath) or reflection, and basically consists of an on-off channel model (Figure 3-24). An omnidirectional field of view is also assumed, neglecting a more realistic assumption of limited field of views with specific orientations and varying gain as a function of relative viewing angles. Communication links in a real exploration system would be significantly more complicated.

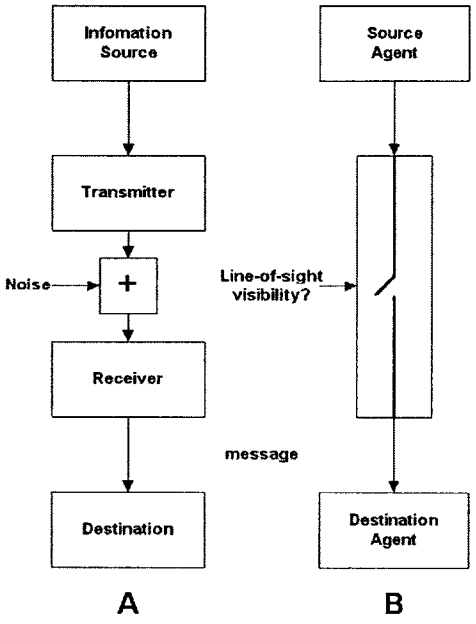


FIGURE 3-24. Line-of-sight visibility serves as a highly simplified channel model for representing opportunities for communication: a typical channel model structure (A), as compared to the line-of-sight visibility channel model structure (B).

Visibility analysis here is used primarily to assess the structure of a distributed system based on some particular configuration of the system. It may be used in the design process to play a "what if" game with respect to alternative potential configurations of a distributed system.

The focus here is on the surface segment of a distributed system: while the existing methodology could be used to simulate orbital links, analysis of the (isolated) surface segment of a distributed system is justified because orbital links may impose significant power costs or hardware requirements on surface agents, may not be available at all times, or may be succeptible to failure. Complete coverage of a surface with orbital assets may require many satellites, or fewer satellites with increased power requirements. Orbital assets

may also be succeptible to failure due to geomagnetic storms, micrometeoroids, or exposure to other potentially disruptive events. One highly efficient use of orbital assets in conjunction with distributed surface systems might be for the transmission of broadcast messages (in the tradition of the Global Positioning System Network, for example). Lower latency and lower power per bit transferred may be achievable with a surface system in many cases. Even with a global communications, navigation, and sensing satellite network for Mars, such as the one described by [Talbot-Stern, 2000], local networks are likely to be required at many scales on the Martian surface for truly global exploration. Local surface networks will be utilized, commensurate with the benefits they bring in terms of efficiency of operation and complementary capabilities.

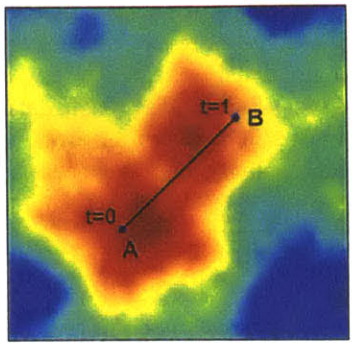


FIGURE 3-25. To compute line-of-sight visibility between two nodes on or near the surface of a height field, one can define a parameterized curve between two nodes A and B, and then check that the altitude at all points along the curve is greater than or equal to the altitude of the height field.

3.3.1 Line-of-Sight Visibility Algorithm

Figure 3-25 illustrates one possible algorithm for computing line-of-sight visibility between nodes on or near a surface represented by some height field h. Between any two nodes A and B at positions $\underline{p}_A$ and $\underline{p}_B$, one can define a parametrized curve with parameter t such that:

$$\underline{p}(t) = \underline{p}_A + t \cdot (\underline{p}_B - \underline{p}_A) \text{ for } 0 \le t \le 1 . \qquad \textbf{(EQ 3-20)}$$

Points along the parameterized curve can then be chosen at an interval of:

$$t_s = \frac{w}{|\underline{p}_B - \underline{p}_A|} \qquad \textbf{(EQ 3-21)}$$

where w is the spacing between altitude samples in the height field, and $|\underline{p}_B - \underline{p}_A|$ is the distance between nodes A and B. This ensures that each cell in a height field along the parameterized curve is sampled at least once. For each sampled value of $\underline{p}(t)$, a corresponding height field altitude $h(t) \equiv h(p_x(t), p_y(t))$ can easily be determined. Two points have line-of-sight visibility if:

$$p_z(t) \ge h(t) \text{ for } 0 \le t \le 1 . \qquad \textbf{(EQ 3-22)}$$

Figure 3-26 illustrates the application of this process for two points. To evaluate the effect of topography on the line-of-sight visibility for a distributed system, one need only check for line-of-sight visibility between each possible pair of system elements. *Line-of-sight visibility represents a criterion with which a graph structure of a distributed system can be built. That structure can then be analyzed with the tools of graph theory.*

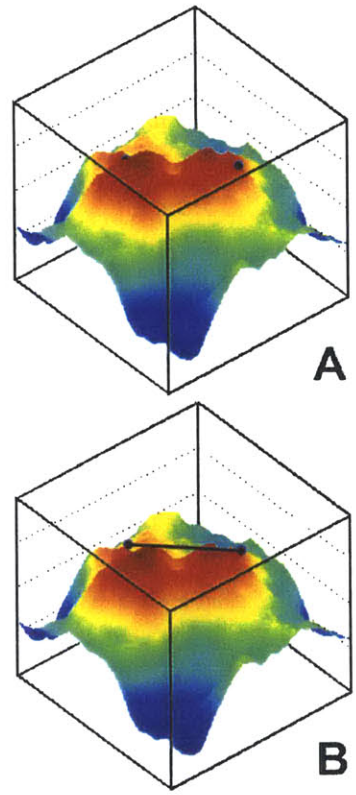


FIGURE 3-26. In (A), line-of-sight visibility does not exist between the two nodes (one node is partially hidden). In (B), the altitude of each node has been increased, and line of sight visibility exists between the two nodes.

Line-of-sight visibility is a starting point for many more interesting and useful line-of-sight metrics. If line-of-sight visibility does exist between two nodes, one may want to also specify the distance between the two nodes, or associate some cost with the "link" between the two nodes that depends upon other characteristics of the "link." For example, the distance between nodes is important for power budget computations. For two agents A and B, the distance between A and B, and the transmit and receive gain factors of A and B will affect the communication link budget. The case where these gain factors are directionally dependent would require specification of local coordinate frames for each agent so that the coordinate frame of one agent relative to the coordinate frame of the other agent could be computed. One may also want to limit valid line-of-sight visibility computations to within a certain distance, perhaps to simulate an "over-the-horizon" condition.

3.3.2 Surface Coverage Algorithm

The coverage for a node A on a surface defined by a height field h can be characterized as the collection of points $h_c \subseteq h$ for which line-of-sight visibility exists between $\underline{p}_A$ and $\underline{p}_c$ for every $\underline{p}_c \in h_c$. This computation can be made by computing line-of-sight visibility between a node A and every altitude sample in a height field h to create a coverage matrix C with the same dimensions as h. Entries in C are coded according to whether line-of-sight visibility exists between $\underline{p}_A$ and the corresponding $\underline{p}_c \in h$. In the simplest case, these value may be binary (1=line-of-sight, 0=no line-of-sight), or may represent some other metric, such as existence of line-of-sight divided by the distance between $\underline{p}_A$ and the corresponding $\underline{p}_c \in h$ (high values mean visible and near, low values mean visible and far, and near or far non-visible points are assigned zero values).

Figure 3-27 illustrates surface coverage results for a node at various heights above a surface using a binary coverage algorithm and an inverse distance coverage algorithm.

Figure 3-28 illustrates surface coverage for a collection of nodes that form a path. Surface coverage for a collection of nodes is a straightforward extension of surface coverage for a single node, except that some method for combining coverage matrices must be utilized. A simple approach to combining coverage matrices is to simply add coverage matrices, although in some cases it may be desirable to use some other function to combing multiple coverage matrices. In some cases it may be desirable to analyze surface coverage of a path as a function of time, or with some element of temporal weighing. This can easily be accomplished by sampling points along the path in even time intervals, and then applying the desired surface coverage algorithm to the resulting set of points.

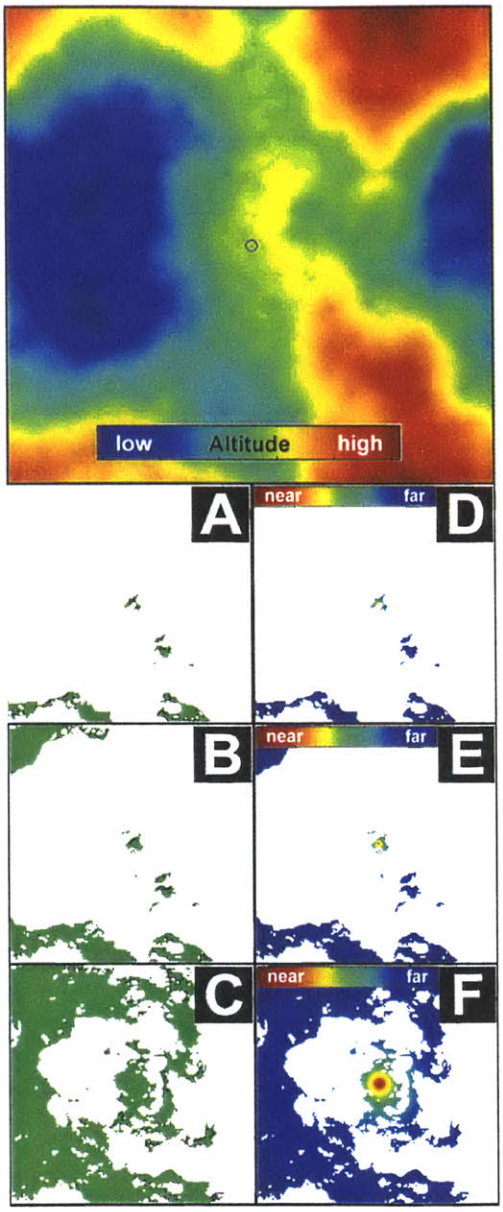


FIGURE 3-27. Surface coverage using a binary coverage algorithm is shown for a node at a height above a surface of (A) 1 meter, (B) 5 meters, and (C) 50 meters. The corresponding results using an inverse distance coverage algorithm are shown in (D), (E), and (F). Colored areas in (A)-(F) are visible from the node; white areas are not visible from the node.

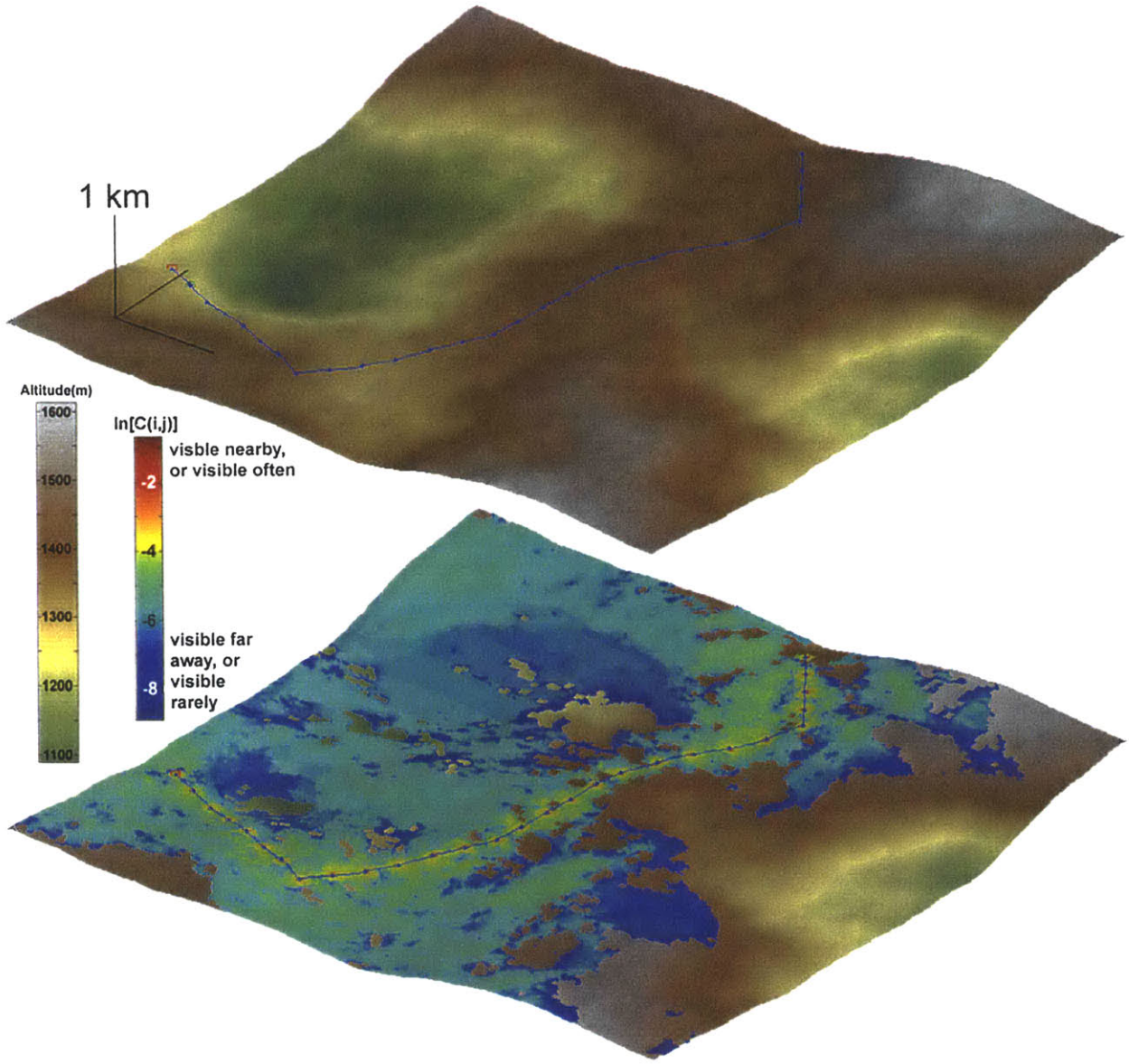


FIGURE 3-28. Surface coverage is computed here for a collection of nodes that together form a path on a surface. An inverse distance coverage algorithm is used to compute a coverage matrix, and the results are shown overlaid on the surface. Each coverage matrix entry $C(i,j)$ represents the sum of the inverse distance computations at that point for each of the nodes in the path. Non-visible points, corresponding to points where $C(i,j)=0$, are not drawn.

3.3.3 Accessibility Algorithm

The area of a surface accessible to a given explorer given one or more constraints is called an *accessibility map* or a *reachability map*. For example, given a robotic explorer with a constraint "must operate on slopes less than five degrees" one could approximate the accessible area to that explorer on a surface by computing an accessibility map given (a) a slope map of the surface, and (b) some initial location of the robotic explorer.

An algorithm capable of generating accessibility maps for simple inequality relationship relating to surface properties (i.e., *0 < slope (degrees) < 5*) is a simple *region-fill* algorithm (see *surface_compute_reachability.m* implementation in Appendix D). The basic idea, given an initial location, is to mark all nearby locations as reachable if they satisfy the constraint, and for each of the new locations, mark any new locations as accessible if and only if they satisfy the constraint.

This concept is easily extendable to multiple explorers by merging multiple accessibility maps: the merged map would then represent an accessibility map for the entire group of explorers. In addition to generating slope-based accessibility maps, one might generate an accessibility map based on other surface properties such as the weight-bearing nature of soil (e.g., wheeled vehicles shouldn't go places where they will get stuck in loose soil) or surface roughness.

3.3.4 Node Placement

Nodes representing agents in a distributed system must have some initial positions in relationship to the surface upon which they operate. In simulating the performance of distributed systems, one must choose how to assign initial positions to each node.

The appropriate node placement method is related to the process by which the distributed system might be built and operated. For example, a group of small payloads for Mars surface exploration might be delivered by an orbiter or entry vehicle as a single payload to a specific surface location, and the system might evolve to a much more sparsely distributed system as individual payloads explore different local areas (Figure 3-29, part A). Another possible architecture might utilize an orbiter to deliver multiple small payloads, each with its own entry and landing system (Figure 3-29, part B). The resulting collection of payloads, in its initial state of surface deployment, would be spread over some region of the surface determined by the differences in the entry corridors of each small payload (timing, release accuracy, atmospheric profile differences, entry and landing system differences, terrain avoidance procedures during landing, etc.). Any landing dispersion factor (except biases that equally effect all payloads) would contribute to a spreading of delivered surface payloads. For the purposes of this thesis, this spreading might be modeled as a two dimensional gaussian distribution. It might also be desirable to uniformly scatter elements of a distributed system over a surface. A uniform distribution could clearly be used to generate a random set of (x,y) positions for elements of a distributed system.

Collections of agents might operate on various scales: large agents with longer-range mobility might carry smaller, potentially more expendable, agents. These larger agents might in turn distribute

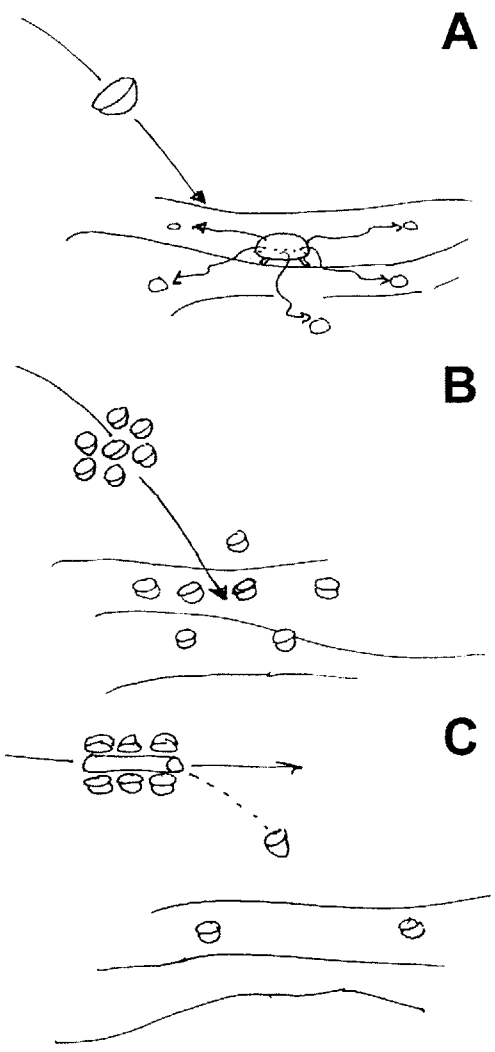


FIGURE 3-29. A few possible architectures for delivery of a surface-based distributed system are sketched. In the "careful deployment" delivery architecture (A), a single payload is delivered to the surface, and a distributed system evolves as sub-payloads are deployed on the surface. In the "rapid, stochastic deployment" delivery architecture (B), multiple small payloads are delivered to the surface. In the "targeted deployment" delivery architecture (C), individual payload are delivered to the surface one at a time by an orbiting satellite (a real implementation of "targeted deployment" might involve significant time between deployments, to wait for the satellite to reach an appropriate delivery position). Many other delivery architectures are possible.

(drop off) the smaller agents in various locations as the larger agents explore. This dropping off of smaller agents might be modeled as a Poisson process or a Markov process.

Other strategies for node placement might also be developed: if a mission goal consists of providing a certain level of coverage at minimum cost, or maximizing the coverage of a system given fixed resources, heuristics or other approaches might be developed to optimize the placement of individual nodes. For example, given a surface, what would be the minimum number of communication relays to provide some minimal level of communications coverage over 90% of the surface? Again, the node placement method is related to the process for building and operating the system: it doesn't matter if a few relays in select locations can provide service to a large area effectively if the relays cannot be delivered to the select locations in the first place. Node placement must therefore take into account the cost of transporting agents from some initial location to another location.

It is not the purpose of this thesis to define in detail any strategies for optimal node placement, but only to illustrate that a variety of node placement strategies exist, and they must be related to the specific delivery mechanism for a given distributed system, including an evaluation of the possibility and/or cost of delivering system elements to their "desired" positions (in the case of relatively fixed infrastructure elements of distributed systems).

Node placement, as outlined here, is a strategy for establishing the initial conditions for a distributed system. In general, nodes of a distributed system need not be fixed, so that the arrangement of nodes, and the network structure of a distributed system is dynamic. One might characterize a dynamic network as *dynamically stable* if the process of communicating changes in network structure to the rest of the distributed system (as required by many approaches to routing) is fast as compared to the rate of change of the network topology. If a distributed system is dynamically stable then routing algorithms relying on global state might be applicable. If a network is not dynamically stable then routing algorithms dependent upon global network state will not be useful. If a functional routing system cannot be maintained in the distributed system, then distributed algorithms cannot be reliable executed. Likewise, the assumption of node movement does not invalidate simulation of distributed algorithms on a dynamic graph (the network structure of a distributed system) if the execution time of the distributed algorithm is short compared to the time between changes in the graph.

Additional considerations that should be made when considering an approach to node placement should be value of coverage provided by each node and by the system as a whole. [Tutschku et al., 1997] describe a system that can be used to optimize placement of cellular-network towers using a digital terrain model and expected

demand statistics. In a similar fashion, one can treat the node positioning problem as a *maximal covering location problem* [Drezner, 1995].

System-level coverage maps provide a measure of where an additional agent might go and still have viable communication links to the rest of the distributed system: if constant communication capability is a requirement for elements of the distributed system (for example, an astronaut during an extravehicular activity, who must stay in contact with a base of operations at all time), then the coverage map defines the boundaries of the areas that agents of the distributed system are allowed to explore.

The designer of a surface-based distributed system must consider many questions about the distribution of nodes on a surface: What should the spatial node distribution be as a function of the number of nodes? Should nodes be heterogeneous or homogeneous (e.g. some nodes perform routing, some don't)? What is the relationship between topography and the number of nodes required to achieve certain levels of connectivity within the distributed system? Examples later in the chapter will illustrate how some of these choices might be made in specific circumstances.

3.3.5 A Visibility and Coverage Example

This example is designed to demonstrate how node density and topography affect coverage and connectivity of a distributed system, as measured by line-of-sight connectivity and coverage metrics. Node distribution will not be evaluated in this simple example.

Hypothesis. The degree distribution of the graph that characterizes the connectivity of a distributed system is likely to be shifted towards higher degrees as node density increases and as the power law scaling exponent increases. Coverage, measured as a percentage of the surface covered, is likely to improve as the power law scaling exponent increases and as the number of nodes increases.

Methods. Surfaces were created using the power spectral density-based terrain generation approach (described in Section 3.2.2). The Cottonwood region (used as an example in Section 3.2.6) provided horizontal and vertical dimensions for the baseline surface (vertical scaling of a surface will not change either the power law scaling exponent of a surface or a line of sight condition). The nominal surface was created to have a power law scaling exponent of 4, a value intermediate to the power law scaling exponent computed for the Cottonwood and Crater Lake digital elevation models and the values for Mars computed by [Aharonson et al., 2000].

To analyze the change in the degree distribution as a function of the topography power spectral density, 100 trials were performed for power law scaling exponents of 3, 4, and 5. For line-of-sight visibil-

ity computations, nodes are assumed to have a maximum height of 2 meters above the surface. For each trial:

- A random surface with the appropriate power law scaling exponent was created,
- 20 nodes were distributed (using a two-dimensional uniform distribution) on the random surface,
- Edges were constructed between nodes using a line-of-sight visibility criterion, and
- The degree distribution of the nodes was computed.

To analyze the change in the degree distribution as a function of the node density, the number of nodes was set to 10, 20, 40, or 80, and the power law scaling exponent was held constant at 4. As before, 100 trials were performed for each experimental condition.

Results. The degree distribution shifts towards higher degrees for increasing values of the power law scaling exponent (Figure 3-30).

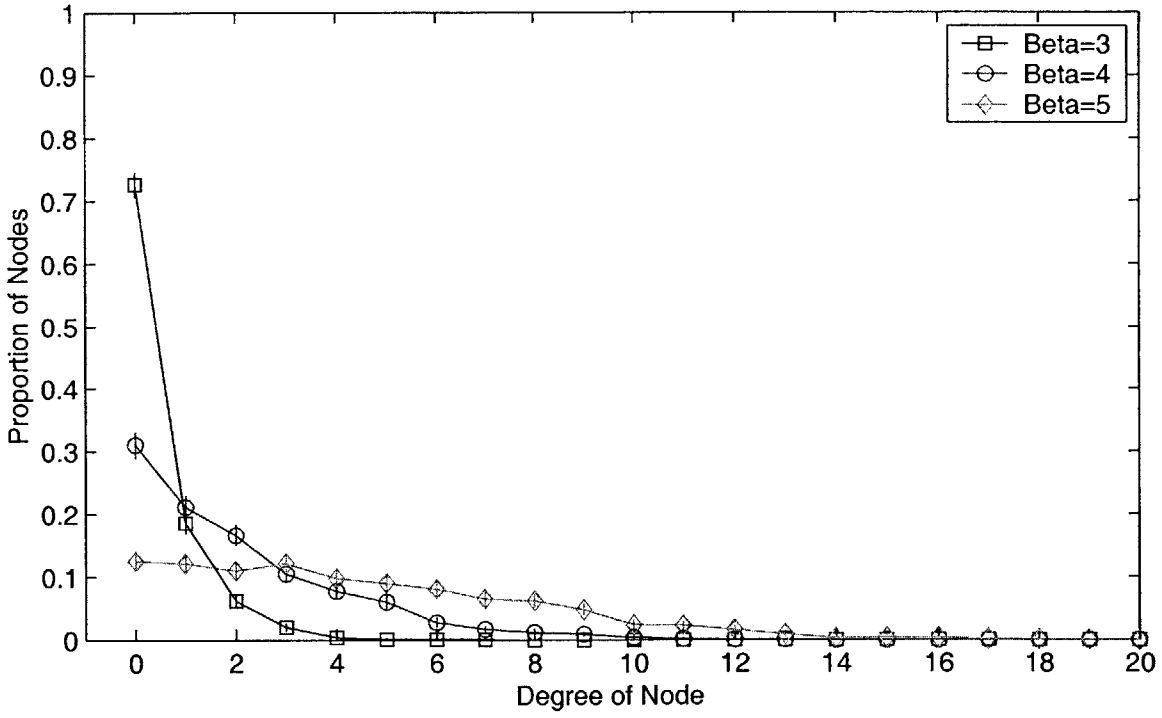


FIGURE 3-30. Power law scaling exponent (beta) significantly affects the degree distribution of a surface-based distributed system. Height of vertical lines at each data point indicate 95% confidence intervals for each computed proportion.

As the power law scaling exponent is increased, bigger valleys and mountains tend to dominate over small topographic structures, and this tends to enhance line-of-sight visibility between nodes. The degree distribution tends to flatten and extend to higher degrees.

From Figure 3-31 it is clear that (over the experimental conditions studied) increasing the node density has a similar effect on the degree distribution as increasing the power law scaling exponent: for higher node densities the degree distribution tends to flatten and extend to higher degrees.

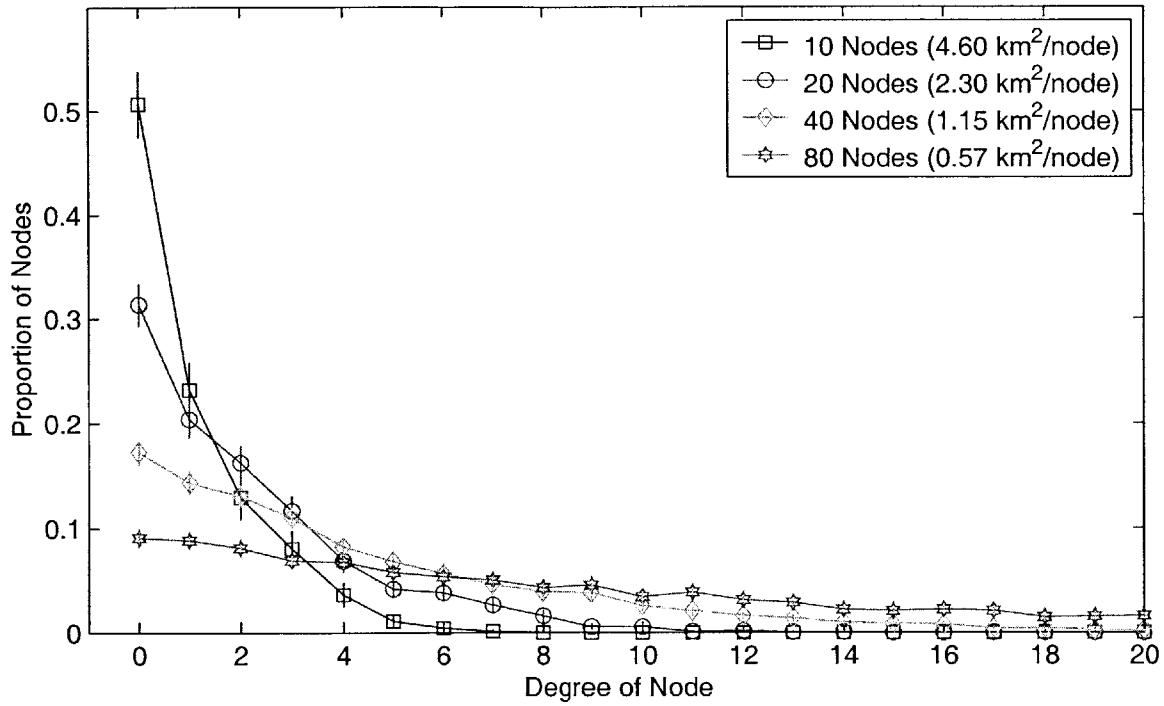
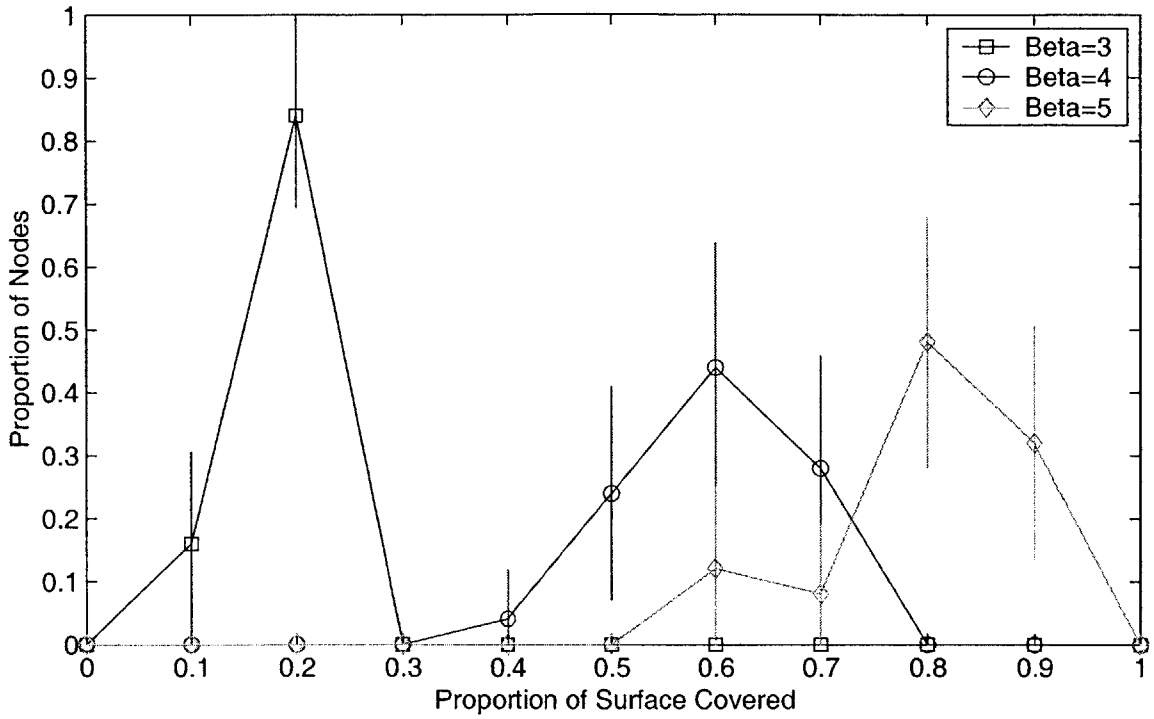


FIGURE 3-31. Node density significantly affects the degree distribution of a surface-based distributed system. The power law scaling exponent is 4 for each of the 100 trials per experimental condition. Height of vertical lines at each data point indicate 95% confidence intervals for each computed proportion.

Coverage, measured as a percentage of the surface covered, increases as the power law scaling exponent increases (Figure 3-32). Similarly, coverage increases as the node density increases.

By computing the mean degree one can establish a minimum degree for some subset of the system graph. For power law scaling exponent $\beta = \{3, 4, 5\}$ the mean degrees of the system connectivity graph G are 0.3, 2.1, and 4.7, respectively. Recall that for any graph G, a subgraph H exists with a minimum degree greater than or equal to one-half the mean degree of G (see Section 3.1.2). For $\beta = 5$ one can conclude that a subset of the graph exists that has a minimum degree of greater than or equal to 3 (minimum degree must be an integer greater than or equal to 4.7/2=2.35).

When the number of nodes is set to $N = \{10, 20, 40, 80\}$ (with $\beta = 4$), the mean degrees of the system connectivity graph are 0.9, 1.9, 3.8, and 7.8, respectively. Therefore in the experimental condition with $N = 80$, a subset of the graph has a minimum degree of 4.



Discussion. These results are based on a finite-size, limited spatial resolution (30 meter) digital elevation model, with nodes at a height of 2 meters above the surface. Although not dealt with here, finite size effects could be studied by creating a larger surface than necessary, and using a subset of the surface for connectivity studies). Increased spatial resolution would tend to decrease estimated connectivity (decreased line-of-sight visibility) because of increases in surface roughness at smaller scale lengths; reducing the altitude of nodes above the surface would also have a similar effect. However, connectivity in a real distributed system would likely be higher than estimated here since line-of-sight visibility is a somewhat overly restrictive criterion for connectivity between system elements.

Degree distribution has been used here to provide a measure of connectivity between system elements, but other graph characteristics could also be studied (such as determining the size of the largest k-connected subset of the graph, in order to study functional redundancy within the distributed system).

Although not simulated here, the effects of adding a high altitude node (perhaps simulating an orbiting satellite) on the connectivity topology could easily be performed. Adding a high altitude node would tend to shift the degree distribution to the right.

FIGURE 3-32. Power law scaling exponent (beta) significantly affects surface coverage of a surface-based distributed system. Height of vertical lines at each data point indicate 95% confidence intervals for each computed proportion (proportions are based on 25 trials per experimental condition, each with 20 nodes, a node density of 2.3 km^2/node).

This example is a simplistic and introductory approach to investigating the connectivity properties of surface-based distributed systems. This example demonstrates how graph properties can be studied in a probabilistic fashion: given a surface, a set of nodes, and a connectivity metric (line-of-sight visibility, for example) what is the probability that the graph has a certain property? The interested reader can explore the literature of random graphs (see [Diestel, 1997], Chapter 11).

3.3.6 Summary

Visibility and coverage analysis can be used to determine what members of a distributed system can see other members of the system, and what part of a surface environment is visible to members of a distributed system. Line-of-sight visibility represents a criterion with which a graph structure of a distributed system can be built: edges represent opportunities for interaction (communication) between members (nodes) of the distributed system. That graph structure can then be analyzed with the tools of graph theory. Coverage analysis can be used to assess both what part of a surface can be seen, and where additional system elements could be placed and have network coverage (connectivity with at least one or more other system elements). While it is clear that line-of-sight visibility is an extremely imperfect criterion for determining interaction opportunities between elements of a distributed system, it can nevertheless provide a first order criterion for interaction (communication) opportunities.

A simple example demonstrated that, for a given set of nodes, connectivity levels (as measured by the degree distribution, one of many possible ways to characterize connectivity) are positively related to the power law scaling exponent. Similarly, connectivity levels are positively related to higher node densities. The example also demonstrated that coverage increases as the power law scaling exponent or the node density increases,

Visibility and coverage analysis is used in Chapter 4 to assess visibility of human explorers during field geology and to understand navigation difficulties during the Apollo missions. Visibility and coverage analysis will also be used in Chapter 5, which will incorporate visibility and coverage analysis into the traverse planning process for individual members of distributed systems.

3.4 Trades

Several tools have now been described that can be used to build and analyze the structure of a distributed system in some environment. Surface modeling and analysis can be used to simulate the topography of an environment, visibility and coverage analysis can be used

to build a graph structure representing connectivity between system elements, and graph theory can be used to analyze the connectivity topology of a distributed system. While these tools can provide powerful insight into some system level trades, they are simplistic in many ways, and many additional trades must be considered. This section highlights major system-level and agent-level trades for surface-based distributed systems for exploration.

3.4.1 Distributed vs. Non Distributed

There is little question that future systems for planetary surface exploration are likely to be composed of distributed elements including surface and orbital assets. The extent to which a system is distributed is driven by the overall goals of the system and the benefits conferred by distribution relating to system performance, robustness/fault tolerance, lifetime cost, and flexibility [Saleh, 2000]. Distributed systems are typically used to handle large problems where the cost of a centralized failure needs to be avoided. Distributed systems are compatible solutions for missions that require the distribution and synthesis of information from multiple sources, or that require graceful degradation in the event of a failure. While distributed systems may help reduce cost by eliminating physical interfaces, highly connected distributed elements introduce significant system complexity: the problem of routing information from one element to another becomes challenging as the number of paths between two elements quickly becomes extremely large.

3.4.2 Multi-hop versus single-hop routing

Power efficiency is a critical issue during planetary surface exploration. Consider the power required to transmit a message from one node (A) to another node (B). By the Friis transmission equation (see [Wertz and Larson, 1992], Chapter 13, or [Delin, 2001]):

$$P_t \propto r^m P_r \qquad \text{(EQ 3-23)}$$

where P_t and P_r are the power transmitted and received, respectively, r is the distance between node, and $2 \le m \le 4$ ($m = 2$ in free space). Now assume nodes A and B are n nodes apart (Figure 3-33).

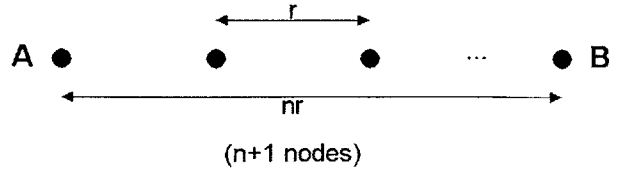


FIGURE 3-33. Multi-hop and single-hop transmission power can be compared by computing the power required to transmit a message from node A to node B using a single hop or via n sequential hops.

It is easy to see that in the single-hop scheme, the transmitted power must be:

$$P_{t,s} \propto (nr)^m P_r \quad \textbf{(EQ 3-24)}$$

whereas for the multi-hop scheme, the transmitted power must be:

$$P_{t,m} \propto nr^m P_r \quad \textbf{(EQ 3-25)}$$

so that the ratio of required multi-hop power to single-hop power is:

$$\frac{P_{t,m}}{P_{t,s}} = \frac{1}{n^{m-1}}. \quad \textbf{(EQ 3-26)}$$

On the surface, multi-hop is therefore very power efficient compared to single-hop, but other factors such as message routing and reliability of message delivery must also be considered. Because many nodes participate in routing in a multi-hop system, the routing burden is increased. To examine reliability issues of multi-hop systems, define p_s as the probability that a message is sent given that it has been received, and p_r as the probability that a message is received given that it has been sent (differentiation of these two probabilities is important for analyzing different node geometries than the simple linear array of nodes described in this example). Then the probability of delivery for the single-hop from A to B is:

$$p_{A \to B,s} = p_s p_r \quad \textbf{(EQ 3-27)}$$

while the probability of delivery for the multi-hop from A to B is:

$$p_{A \to B,m} = (p_s p_r)^n. \quad \textbf{(EQ 3-28)}$$

The expected power required to transmit a message from A to B is therefore given by the power of a transmission from A to B times the expected number of transmissions required to successfully send the message from A to B (here it is assumed that node A has perfect information about the message delivery, and node A must reinstate the transmission if it fails anywhere along the way). For the single-hope case, the expected power is given by:

$$E(P_{t,s}) \propto \frac{1}{p_s p_r}(nr)^m P_r \quad \textbf{(EQ 3-29)}$$

and for the multi-hop case, the expected power is given by:

$$E(P_{t,m}) \propto \frac{1}{(p_s p_r)^n} nr^m P_r. \quad \textbf{(EQ 3-30)}$$

The ratio of single-hop to multi-hop expected power is therefore:

$$\frac{E(P_{t,s})}{E(P_{t,m})} = (p_s p_r)^{n-1} n^{m-1} = R \qquad \text{(EQ 3-31)}$$

Single-hop is more power efficient for $R < 1$, while multi-hop is more power efficient for $R > 1$. Figure 3-34 plots the decision ratio R for various values of p with $m = 2$ and $p_s = p_r = p$ (for simplicity).

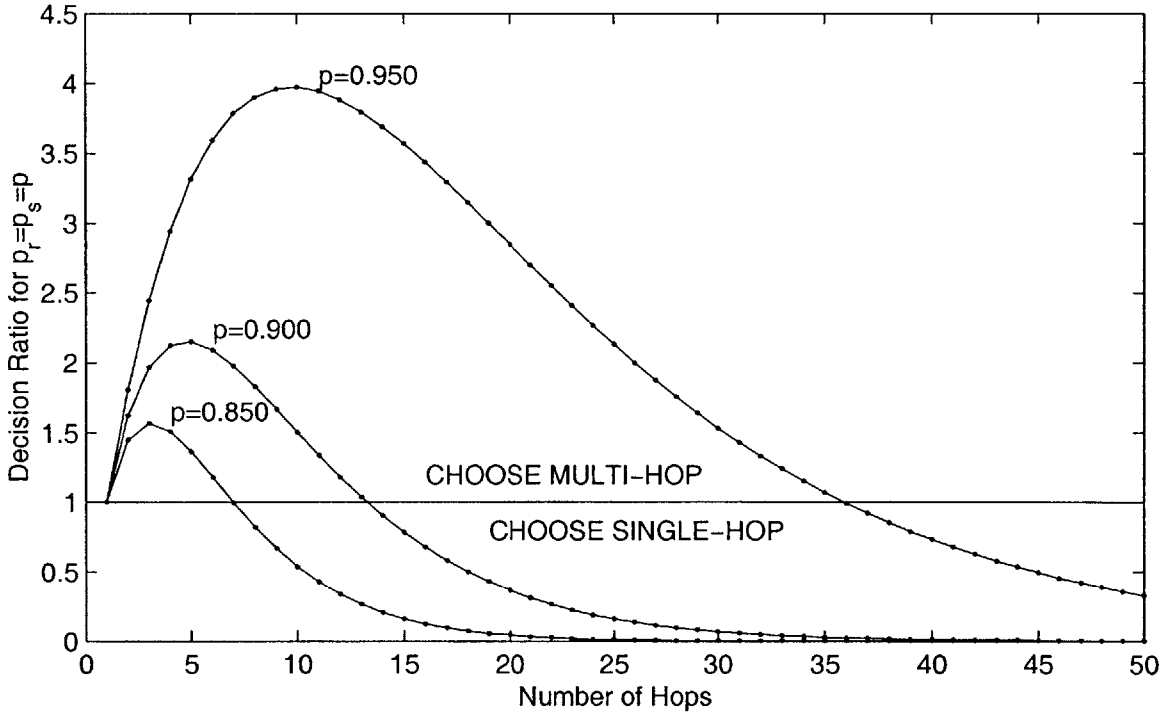


FIGURE 3-34. Decision ratio for multi-hop or single-hop transmission is plotted as a function of the number of hops of the multi-hop transmission path and the transmission and reception probabilities (Transmission and reception probabilities are assumed to be equal).

When transmission and reception by nodes is not assumed to be 100% reliable, multi-hop transmission should be chosen only when the number of hops is small (given the assumptions of this analysis). Transmission and reception probabilities might be less than one due to several factors: nodes might only function as routers in a part-time fashion, or may act selfish by sending their own data instead of data received from others. Nodes might also fail outright or fail in a probabilistic fashion due to communication channel conditions or other environmental factors.

Clearly, one need not limit multi-hop systems to linear arrays of nodes: other geometries can provide many redundant paths between two nodes, but at the expense of routing complexity.

One might be able to use a hierarchy of multi-hop systems to obtain the power-efficiency benefits of multi-hop with a limited number of total hops. In a hierarchy of multi-hop systems, messages sent locally would be sent very efficiently in a single transmission path sense, while messages sent between distant nodes would incur a more substantial transmission path power cost, but would minimize the expected power cost for delivery of the message. This hierarchy of multi-hop systems would require heterogeneous nodes. Delin and his colleagues at the Jet Propulsion Laboratory [Delin, 2001] have done precisely that: in their SensorWebs system, small "pods" forward data to "prime nodes" with longer range data transmission capabilities. A similar situation might be appropriate in an ecology of robotic explorers over large distances: pods might communicate to small nearby rovers when they happen to pass by, small rovers might communicate with large rovers, and large rovers might communicate to landers or to an orbital asset. In this way, the multi-hop approach is preserved, but the hops are of the appropriate scale such that the number of total hops from the source node to target node remains small.

This brief analysis has only considered the transmission of a message from one node to another, but often it may be desirable to transmit a message from one node to many other nodes. In this case, single-hop transmission will tend to be comparatively more power efficient than it is in this analysis. Orbital assets are particularly effective in the delivery of broadcast messages because of their large footprint (ability to deliver a message to a large number of nodes or large area of the planetary surface). In general, a surface system for delivery of information would be favored over a satellite system when (1) broadcast messages are infrequent, (2) satellite resources are temporally or spatially limited, or (3) surface nodes are sparse (dense surface node networks would typically involve challenging routing problems, especially if nodes are mobile).

3.4.3 Layers of Abstraction

Communication networks are commonly abstracted into a series of network layers. Figure 3-35 illustrates a five layer model applicable to communications between elements of a distributed system, and illustrates adjustable parameters that affect performance at each level. Trades within these set of parameters may be isolated to a single layer but often will have an effect across all layers in some form or another. Major trades within each network layer are briefly discussed below. To achieve an energy efficient or cost efficient network, "adaptability of the protocols is a key issue" [Havinga et al., 2000].

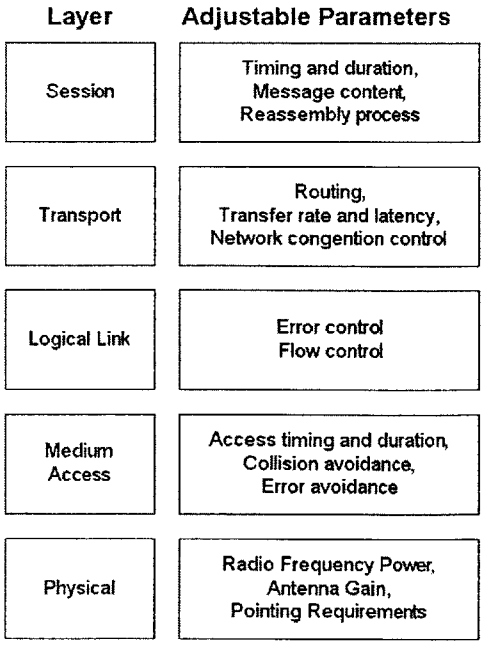


FIGURE 3-35. Network layers and adjustable parameters that drive network efficiency.

Physical layer. At the physical layer, trades may include the required transmit and receive powers required to achieve a certain bit error rate; pointing requirements and the antenna pattern (and

thus the antenna gain) must also be considered. Modulation techniques, frequency allocation, and signal bandwidth are also important parameters. Variable radio-frequency power could be used to maintain constant performance while minimizing power usage (e.g., only use as much energy as required to achieve the desired performance, based on the distance of the target).

Medium access layer. Power efficiency can be achieved by minimizing the time the radio needs to be powered up and minimizing the number of times it transitions from powered to unpowered (bulk data transfers will tend to reduce transitions and overhead in many cases). Collision avoidance and error correction techniques can enhance the probability that a message is delivered by reducing the number of bits that must be resent.

Link layer. Trades in the link layer involve channel characteristics and flow control. Channel stability and issues such as multipath should be considered. Adaptive error control can be used to match error correction protocols to channel conditions, increasing effective bandwidth and energy efficiency. Flow control should be used to prevent buffer overflow issues, and to eliminate stale data.

Routing or Transport layer. A primary trade in the routing and transport layer is to determine the path for a message from source to target. The transport layer should react appropriately to packet losses or network congestion. The extent to which the network has a dynamic structure will affect the routing algorithms: algorithms requiring perfect or near-perfect network state information will perform poorly or not at all in a dynamic network. Networks with dynamic topology may require routing algorithms that rely only on local network state information. A heterogeneous network, in which some nodes serve primarily as routers, may simplify the routing problem.

Session layer. Trades here relate to the timing and duration of point-to-point communication sessions. Minimizing session time and duration is one approach to minimizing power usage - approaches to minimizing session time and duration may include use of a point-to-point connection only when needed. Future access needs or access types can also be anticipated so that other network usage might be modified to anticipate network loads. If the cost (power, or some other metric) of transferring data changes with time (potentially due to a dynamic network structure or availability of resources), data can be transferred when the cost is cheap.

3.4.4 Node Density, Homogeneity, and Distribution

As observed in Section 3.3.5, node density and topography place an important role in determining connectivity between nodes and surface coverage of a distributed system. An adequate node density

should be used to achieve both the desired coverage and connectivity levels. For high node densities, effective bit rates between nodes may be reduced due to collisions or due to the burden placed on each node to forward data from other nodes, potentially reducing effective power efficiency. Node heterogeneity can improve this situation: for higher node densities, having some nodes dedicated to routing may increase effective bit rates by reducing collisions (nodes can also be optimized for their specific tasks instead of having to balance their performance on several tasks such as data collection and routing). Structural differences in nodes can potentially be important: nodes that are higher off the surface (a blimp or small tower, for example) would have better line-of-sight visibility to other nodes and to the surface.

Spatial allocation of heterogeneous nodes may depend upon routing function, but it may also depend upon reachability (the ability of a node to reach a point on a surface, given some initial point). Heterogeneous reachability provides a strategy for the spatial allocation of nodes (agents) in a distributed system: expendable or disposable assets can be send to dangerous locations, agents capable of traversing steep slopes can be sent to steep sloped areas.

Obviously, node distribution depends a great deal upon what activities are to be carried out at different locations on the surface (for example, different scientific studies), but routing function and reachability can provide additional guidance for the desired node heterogeneity and spatial distribution.

3.4.5 Network Services

Agents in a distributed system may "desire" access to several network services including:

- Timing and positioning services,
- Data storage and access services, and
- Coordinated sensing and control services.

Designers of distributed systems should consider the spatial and temporal requirements as well as the required accuracy or allowable latency for these services or other services of interest.

Timing and Positioning. Timing services are critical for coordinating activities between multiple agents such as for distributed sensing and control. In some cases, distributed collection of near-synchronous data might require fairly precise time estimates: for example, a distributed array of "geophone agents" might need to simultaneously record subsurface vibrations caused by a detonation charge, or a group of "surface weather agents" might need to simultaneously measure temperature, pressure, and wind speed over a wide area.

Future human and robotic Mars missions may not have the support of a global positioning system-like constellation to provide a complete positioning solution during Mars exploration; instead, a ground based system (using radio navigation beacons, for example) that supplements position estimates from occasional satellite overflights may be required. Accuracy requirements for some science activities may also require local high spatial resolution positioning systems. Rovers or other exploration agents may also use inertial navigation systems to provide accurate position estimates for short time scales (periodic use of position services could provide corrections to the inertial navigation systems). Considerations might include the frequency of use of position services, and the position accuracy obtainable with different types of positioning systems. Stationary nodes might provide excellent position estimates to other agents by estimating their position with high accuracy: mobile agents could then compute their position based on their relative position from several fixed nodes.

Data storage and access. If the purpose of an exploration system is to collect, synthesize, analyze, and distribute information, then information must be stored and accessed during all of these phases of information processing. Individual agents may not have the capability to store all of the information they need to access in order to carry out their mission goals or to maximize their performance.

The distributed system should provide mechanisms for agents to share specialized information, and to store and retrieve information. Data collected during exploration might be recorded via a store-and-forward mechanism or streamed in real-time over the network. Centralized storage might be an efficient solution for storing and accessing large quantities of scientific and other information, but distributed storage might be less prone to loss of data due to a single point failures.

3.4.6 Quality of Service

The main goal of quality of service protocols is accurate and timely delivery of information. The most common quality of service parameters are bandwidth and delay [Chakrabarti et al., 2001]. Routing algorithms must select a network path that has sufficient resources to meet quality of service requirements for specific applications, while attempting to achieve overall efficient use of network resources. Routing fundamentally involves two tasks:

1. Collection of network state information.
2. Searching of state information for a feasible path between network nodes that meets applicable constraints.

While many quality of service algorithms assume perfect network state information, maintaining perfect network state information is

not feasible for large-scale networks or for ad-hoc networks. Quality of service algorithms must balance performance and the overhead of maintaining local or global network state information. Typical quality of service requirements include bandwidth guarantees, delay or delay jitter constraints, and cost or path length constraints.

Best effort routing may be appropriate for non-time critical data: performance in best effort routing is based on availability of shared network resources, whereas quality of service routing guarantees that a point-to-point connection (or point-to-many-points connection) will meet a certain set of constraints. While unicast (point-to-point) routing usually involves finding a minimal cost path that meets quality-of-service constraints, multi-cast routing (point-to-many-points) routing involves finding a minimal spanning tree that meets quality-of-service constraints.

Routing algorithms are typically classified into three categories: source routing, destination routing, and hierarchical routing [Chakrabarti et al., 2001]. In source routing, the source needs global state information, and has the high computation burden of choosing a path from source to target. This avoids deadlock and distributed algorithm termination problems, but at a high cost both in terms of network state information updates, and in terms of local computation and storage requirements. The high overhead requirements for source routing create a scalability problem. In distributed routing, the path between source and target is computed by intermediate nodes from source to target. Routing can be based on local or global states, and loops can occur. Scalability also tends to be a problem for this class of algorithms. Hierarchical routing requires only partial global state information: global state information is aggregated, and this tends to make hierarchical routing algorithms scalable while negatively affecting quality of service. Hierarchical routing can help to reduce routing updates, which consume network bandwidth and router processing resources and may also increase delay jitter [Chakrabarti et al., 2001].

Quality of service routing incorporating multiple path constraints can be accomplished using heuristic-based distributed algorithms that use only partial network information [Chen, 1999]. These algorithms can be used for ad-hoc mobile networks where unplanned or unpredictable changes in network topology may occur: path redundancy and path repair techniques can be used to maintain or reroute a connection between two nodes. It should be noted that distributed algorithms are powerful for many computational problems including routing because they can provide access to global network state information while only accessing local network state information during execution at any given node.

3.4.7 Network Stability

Network stability is an issue even in networks with static topologies, but can be an even bigger issue in networks with dynamic topologies. A desirable property of a distributed algorithm executed on a distributed system is self-stabilization, which can be defined as the condition that execution of the distributed algorithm leads to a global state in which some property holds regardless of the initial network state [Barbosa, 1996]. Such a property is called a stable property. Algorithm termination is an example of a stable property, and is therefore (in most cases) a desirable characteristic of a distributed system.

Network stability is not always desirable: in a deadlock, nodes may wait for a condition that can only be satisfied by other nodes also waiting for a similar condition. Approaches to dealing with deadlock range from deadlock prevention to deadlock detection and resolution [Barbosa, 1996].

Routing algorithms for best-effort traffic, discussed in the previous section, are an example of a class of algorithms that do not always terminate in a specified amount of time: the path from a source to a target may contain loops, depending upon the routing implementation. Preventing loops is desirable both to increase computational efficiency and power usage in a distributed system. Routing attempts should terminate if a message is simply undeliverable (e.g., the target node is not connected to the network).

Other distributed algorithms (such as some possible implementations of positioning services) should terminate in a reasonable amount of time without large oscillations or lack of convergence. [Barbosa, 1996] discusses approaches for termination detection for both synchronous and asynchronous algorithms.

3.4.8 Flexibility & Robustness

In many cases it is not adequate nor desirable to make static trades between parameters that guide the design of a distributed system: the distributed system may need to have the flexibility to adapt to a new set of requirements or a changing environment, or the robustness to accomplish mission goals in the face of system failures. Hence, trades should take into account future uncertainty about the environment, system failures, and system requirements.

The desired system flexibility should be evaluated in terms of a time reference associated with the occurrence of change, and the element that is changing (such as the system itself, the environment, or the customer needs) [Saleh, 2001]. For distributed systems, flexibility can be a product of different arrangements of system elements, or can be internal to individual system elements.

For example, a physical reconfiguration of nodes in a distributed system might allow matching of routing capabilities to local data transfer needs. System design trades where flexibility might need to be considered include routing strategies, power usage, and quality of service parameters such as delay and bandwidth.

Robustness, like flexibility, can be achieved by reconfiguration of system elements (e.g., changes in functional capability distribution) but also by changes in system utilization (changes in functional capability utilization). Having multiple approaches to the acquisition and delivery of information is another example of robustness that can be provided by a distributed system.

3.5 Trade Study Process

The process developed here is not meant to be an end-to-end approach for simulating potential distributed architectures for planetary surface exploration - it should be viewed more as a process for exploratory design that may assist the designer of a distributed system to answer some of the big "what if" questions.

The process is simply a structured application of the tools developed so far in the thesis for the analysis of distributed systems, and does not explicitly attempt to optimize a design. Two design analysis methodologies that deal with design optimization include the Methodology for the Evaluation of Design Alternatives [Thurston, 1991] and Multiobjective Optimization [Mohandas et al., 1989]. Two additional design analysis methodologies that deal with design optimization for distributed (satellite) systems include [Jilla et al., 1999] and [Shaw, 1999]. These methodologies are complementary to this process in that the outputs of this process might be used as inputs into an optimization process.

To apply the trade study process:

1. Develop initial distributed system functional requirements. These requirements may need to be specified in a probabilistic form, and should be written in terms of some measurable quantity (required for verification during development and operations).
2. Identify a representative surface of the distributed system environment, or define the statistical properties of a representative surface of the environment (such as the power law scaling exponent). The process may need to be followed multiple times to test different representative surfaces in order to generate valid statistics relating to system properties or performance.
3. Identify the delivery mechanism for the distributed system elements (nodes) and compute initial node positions and internal states. Consider whether the system elements will be delivered

in a distributed fashion or whether the system must evolve into a distributed system from a common location (see Section 3.3.4).

4. Determine the topology of the distributed system using some connectivity algorithm (e.g., line-of-sight visibility or some other more appropriate metric).
5. Apply visibility, coverage, and reachability analysis tools to analyze the relationship between the system and the surface and to relate the topology of the distributed system structure to the system functional requirements. Further analysis of the network topology of the distributed system might include evaluation of node heterogeneity, computation of communication link budgets, evaluation of maximal data flow, or implementation and testing of routing algorithms (if the network is not dynamically stable, this may need to be done over many iterations of the process).
6. Evaluate other trades for the current configuration of the distributed system.
7. Evolve node internal states (internal resources, goals, models) and external states (e.g., position, orientation) based on some operational process model. For initial investigations, very simple stochastic models might be used. Failures or other uncertain events can be incorporated into the operational process model.
8. Evolve the surface model (if necessary) based on a some environment/surface evolution model. Other environment characteristics might also be modified (for example, sun angle relative to the surface, or changes in a communications channel model).
9. Repeat the process starting with #4 as needed.

During the primary evaluation portion of the process (4,5), factors to evaluate may include system performance, flexibility, robustness, and cost. Performance analysis may answer questions such as the following: Is the desired level of surface coverage obtainable with the desired number of nodes? Is node mobility adequate to reach sites of interest? Are the information transfer capabilities of the network adequate? Analysis of flexibility may include answering questions such as: To what extent are nodes internally or externally reconfigurable? Does the routing strategy support changes in network topology? How constrained are the actions of individual agents in the system? How constrained are the system level goals? Analysis of robustness may include such items as evaluation of path diversity between nodes, or determination of the extent of any k-connected subsets of the connectivity graph that represent functionally redundant communication paths between nodes. Spare strategies (e.g., in case of failures) may also be evaluated.

Overall cost considerations might include the extent to which nodes are homogeneous or heterogeneous, and the extent of commonality in communication interfaces. Minimizing path diversity while achieving the desired levels of redundancy may help reduce costs

by minimizing the number of interfaces between elements of the distributed system. Elements of distributed systems should be partitioned to minimize information flow between elements to the extent possible [Rechtin, 1991].

This process is extremely basic and can undoubtedly be improved and modified, but can serve as a starting point for analyzing properties and performance of a distributed system for surface exploration.

3.6 A Trade Study Example: Mars Lander and a Sensor Network

This example explores a distributed system trade for a hypothetical mission to Mars to deliver a lander and a distributed network of sensors "pods" to an ancient lakebed, where pods might be used as geophones, to record temperature, pressure and humidity data, or to perform limited in-situ analysis.

3.6.1 Applying the Trade Study Process

Requirements Definition. Many requirements might be written for the mission scenario described above. This analysis will focus on one possible requirement: 90% of nodes should be network-reachable by the lander with an 90% probability.

Representative Surface. Because of a lack of high-resolution digital elevation models of the Martian surface, a digital elevation model of the Earth's surface will be used for the purposes of this example. Two digital elevation models of adjacent regions of the Mojave National Preserve were created for this example (Figure 3-36). These regions were chosen because they are geomorphologically similar to the Martian surface in several ways. The dry lakebed of Soda Lake may be similar in form to some of the potential paleolake sites that have been identified on the Martian surface, including Gusev crater, a landing site candidate for the planned 2003 Mars Exploration Rover mission [CMEX, 2001]. The Mojave desert area also has many eolian (produced by wind-driven processes) landforms: for example, bivariate particle distributions (fine-grained soil and dust, mixed with larger particles of volcanic material) are present in nearby areas, and significant quantities of sand and dust have accumulated to form the Kelso dunes. Carbonates and volcanics (cinder cones) are also present to the east. Each digital elevation model is about 23 km (E-W) by 27 km (N-S).

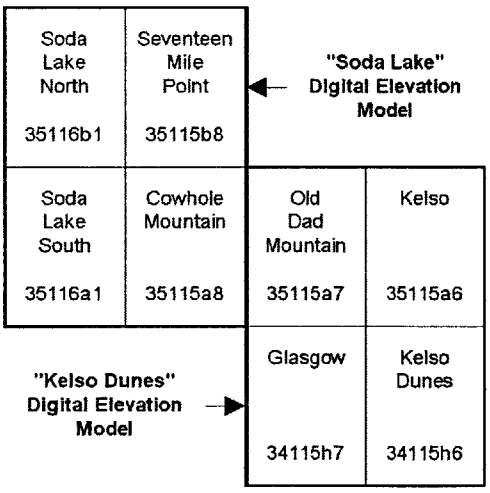


FIGURE 3-36. Two digital elevation models were created of the eastern region of the Mojave National Preserve, California. These digital elevation models, entitled "Soda Lake" and "Kelso Dunes," were each created from four United States Geological Survey 7.5 minute digital elevation model quadrangles (named above, with reference numbers for each quadrangle).

Delivery mechanism. For the purposes of this example, it is assumed that the lander will eject the sensor pods during the descent and landing mission phase, so that the distributed system

will be largely deployed by the time the lander has touched down on the surface. For the purposes of this example it will be assumed that expected lander landing position dispersions can be given by:

$$\{\sigma_x, \sigma_y, \rho\} = \{2.4 \text{ km}, 1.6 \text{ km}, -0.7\} \quad \textbf{(EQ 3-32)}$$

where σ_x and σ_y are the standard deviations of the position dispersions (assumed to be gaussian) in the East-West (x) and North-South (y) directions respectively, and ρ is the correlation coefficient between the positions in the x and y directions. Figure 3-37 illustrates lander landing-sites generated from these dispersion statistics.

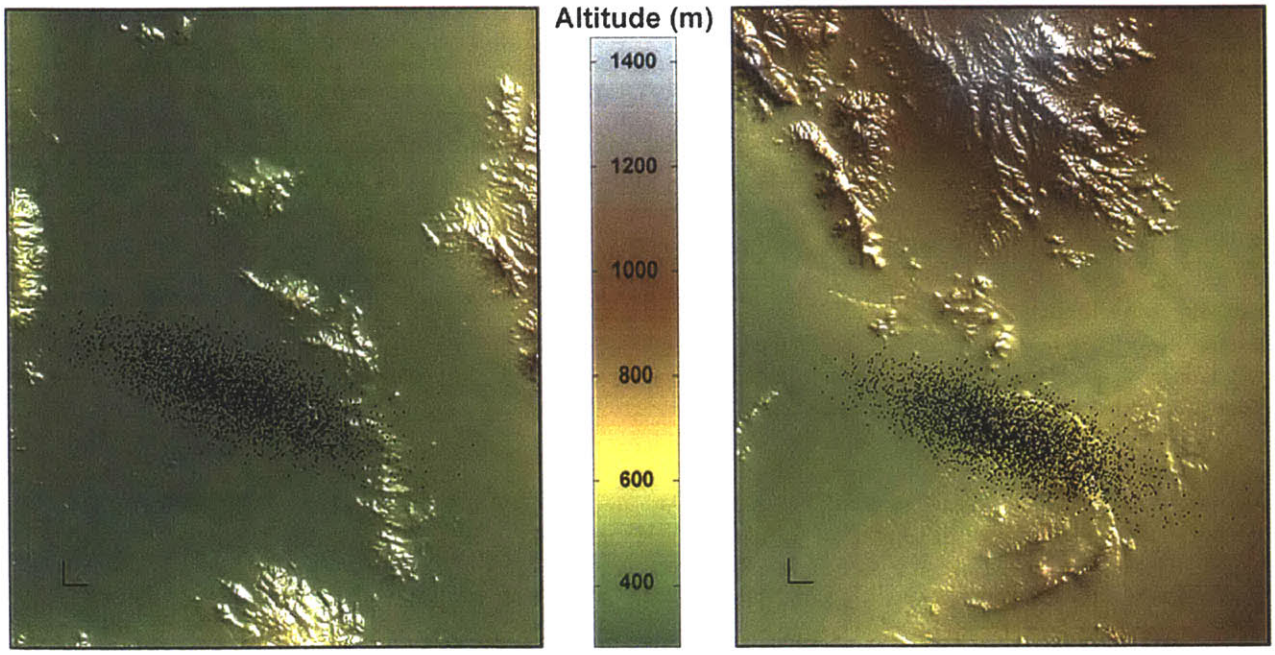


FIGURE 3-37. Assumed lander landing-site dispersions for the Soda Lake and Kelso Dunes digital elevation models. Five thousand points sampled from a bivariate gaussian distribution are plotted on each digital elevation model. The scale bar at lower left is 1 km in length, and digital elevation models are plotted North-up.

It will be assumed that deployment of the sensor pods results in a similar dispersion of the sensor pods, except that the dispersion of the sensor pods is likely to be smaller in magnitude than the dispersion of the lander (the sensor pods will experience many of the same dispersion biases as the lander; some biases tend to affect both lander and sensor pods and so will not contribute greatly to the dispersion of the sensor pods relative to the lander). It will be assumed that the dispersions of the sensor pods can be characterized by:

$$\{\sigma_x, \sigma_y, \rho\} = \{0.8 \text{ km}, 0.5 \text{ km}, -0.7\}. \quad \textbf{(EQ 3-33)}$$

It is further assumed that the number of sensor pods is 100 and that each pod has an effective antenna height of 30 cm, a nominal communication range (at nominal bandwidth) of 300 meters, and a maximum communications range of 1000 meters.

Network topology. The distributed system topology is analyzed using line-of-sight visibility as the connectivity metric. In order to evaluate the stated requirement, the network topology is evaluated for many (300) different possible lander positions.

Analysis of network topology. The stated requirement can be analyzed by determining the subset $G_{Lander} \subseteq G$ of the network topology graph G that is connected and includes the lander as one of its nodes. In this case, Dijkstra's shortest (lowest cost) path algorithm is used to compute G_{Lander} and determine the lowest cost path to each accessible sensor pod. For this computation, the cost (weight) of edge $e_{ij} = (n_i, n_j)$ is given by:

Dijkstra's shortest path algorithm is implemented by the graph_compute_dsp.m function (see Appendix D).

$$C(e_{ij}) = \left(\frac{|\underline{p}_i - \underline{p}_j|}{d}\right)^m \qquad \text{(EQ 3-34)}$$

where d is the nominal distance between sensor pods, $\underline{p}_i$ and $\underline{p}_j$ are the positions of nodes n_i and n_j respectively, and m is the communications link space-loss exponent (equal to 2 for free space). This metric measures the ratio of the estimate received power at a node relative to the nominal received power at a node due to a transmission from another node. Therefore, computing a lowest cost path from the lander to each node is equivalent to determining the most power efficient path to each node, given the available connectivity links (energy constraints of individual nodes are not taken into account here!).

The globally optimum (with respect to some edge cost function) network topology of the distributed system might also be determined by computation of the minimum cost spanning tree, which is implemented by the graph_compute_mcst.m function (see Appendix D).

Analysis of other trades. Other system trades will not be considered in this brief example.

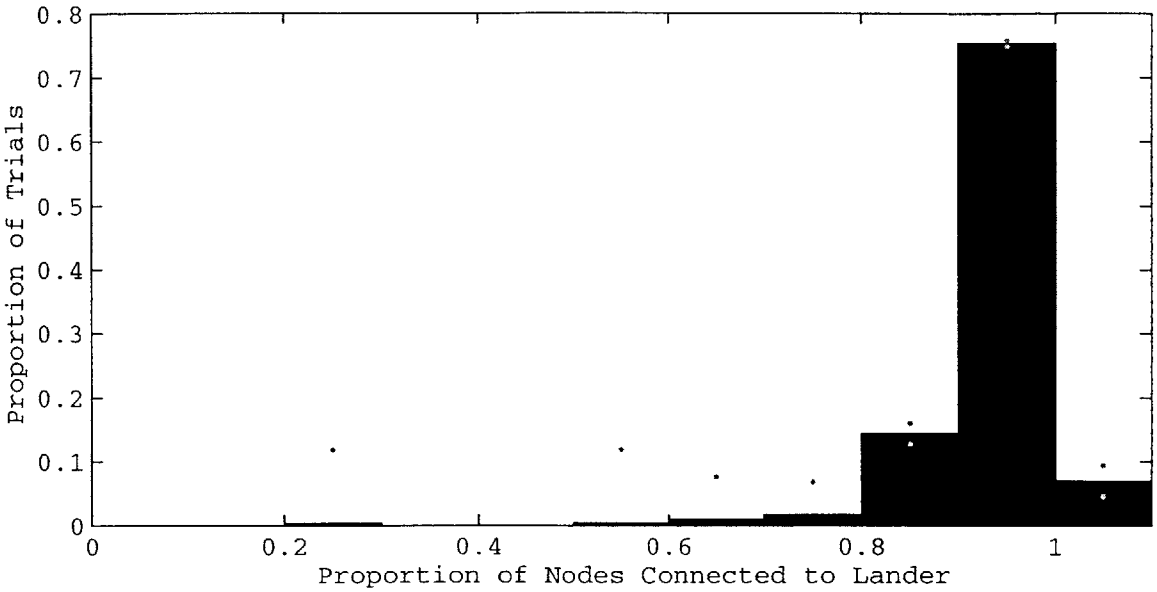
Node evolution. Node evolution might include modeling of power usage of individual nodes using a lowest cost path algorithm or other designated network topology. Node failures might be simulated or other temporal operational models developed.

Surface evolution. Surface evolution will not be considered for this brief example.

Repeating the process. This process is repeated many times for different lander positions in order to generate results that can demonstrate an ability or inability of the nominal architecture to meet the specified requirement, and is performed for two possible "landing sites" - the two digital elevation models previously mentioned.

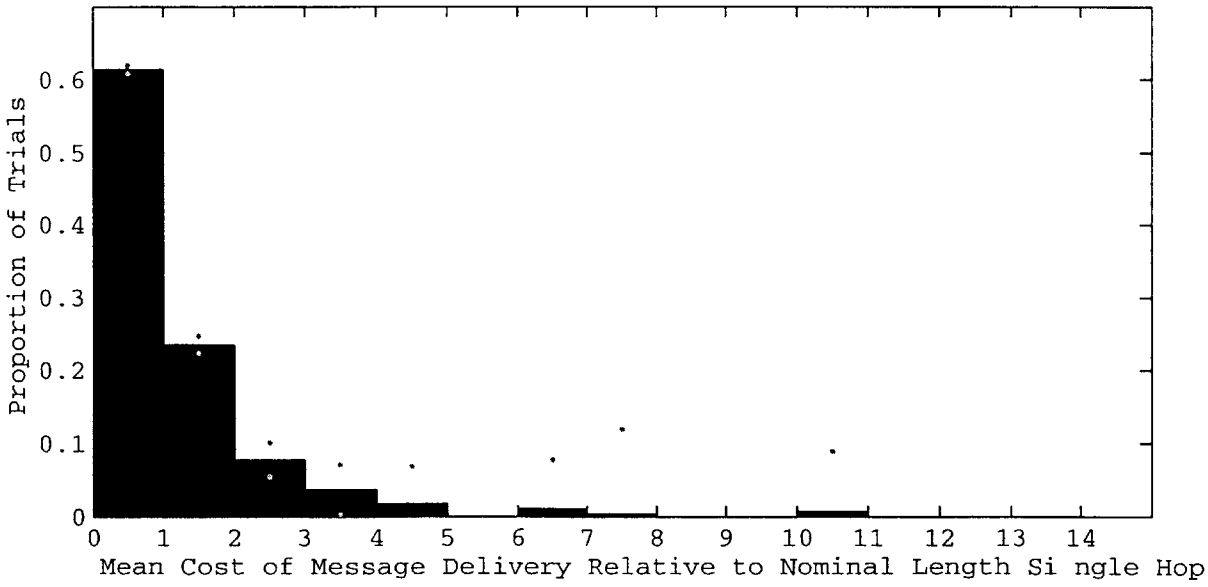
3.6.2 Results

Figure 3-38 and Figure 3-39 show results for the Soda Lake "landing site" for node connectivity and cost of message delivery.



Green dots in Figure 3-38 and Figure 3-39 represent 95% confidence boundaries for the proportion estimates.

FIGURE 3-38. Node connectivity for the sensor pods network for the Soda Lake "landing site." The mean number of nodes connected to the lander was 91.4 (91.4%).



While the mean number of nodes connected to the lander was 91.4, 90% of nodes were connected for only 217 of 300 trials, or with a probability of 72%. Therefore, the requirement is not satisfied. Results for the Kelso Dunes "landing site" (Figure 3-40 and Figure 3-41) are similar, but are further from satisfying the requirement.

FIGURE 3-39. Mean power cost of message delivery from nodes to the lander for the Soda Lake "landing site." Power cost is relative to the nominal distance single-hop power cost, and is based on a minimum cost path analysis of the distributed system network topology.

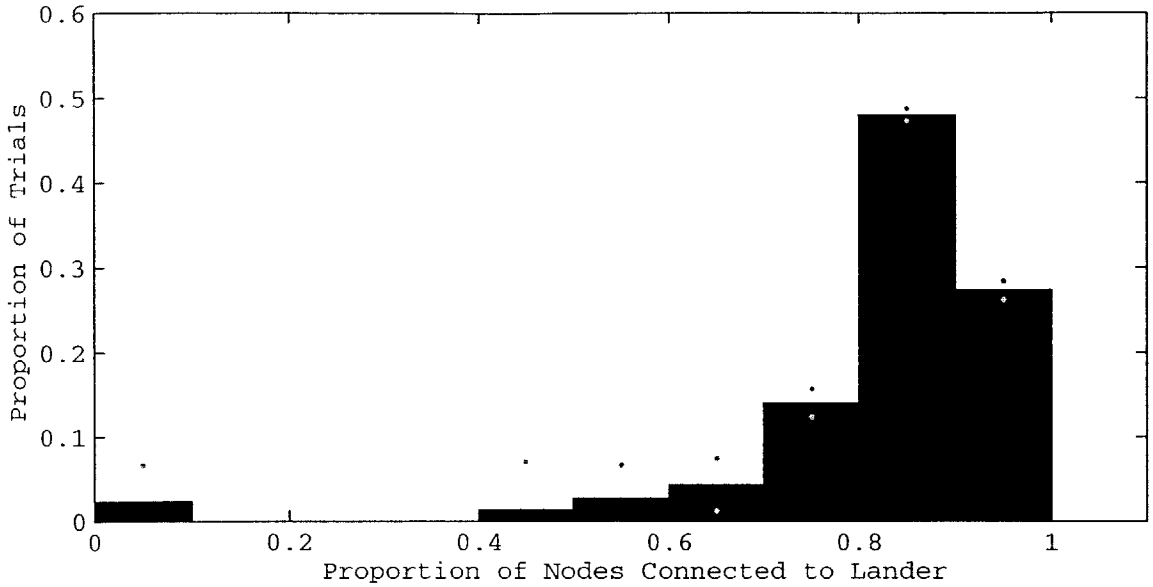


FIGURE 3-40. Node connectivity for the sensor pods network for the Kelso Dunes "landing site." The mean number of nodes connected to the lander was 78.9 (78.9%).

Green dots in Figure 3-40 and Figure 3-41 represent 95% confidence boundaries for the proportion estimates.

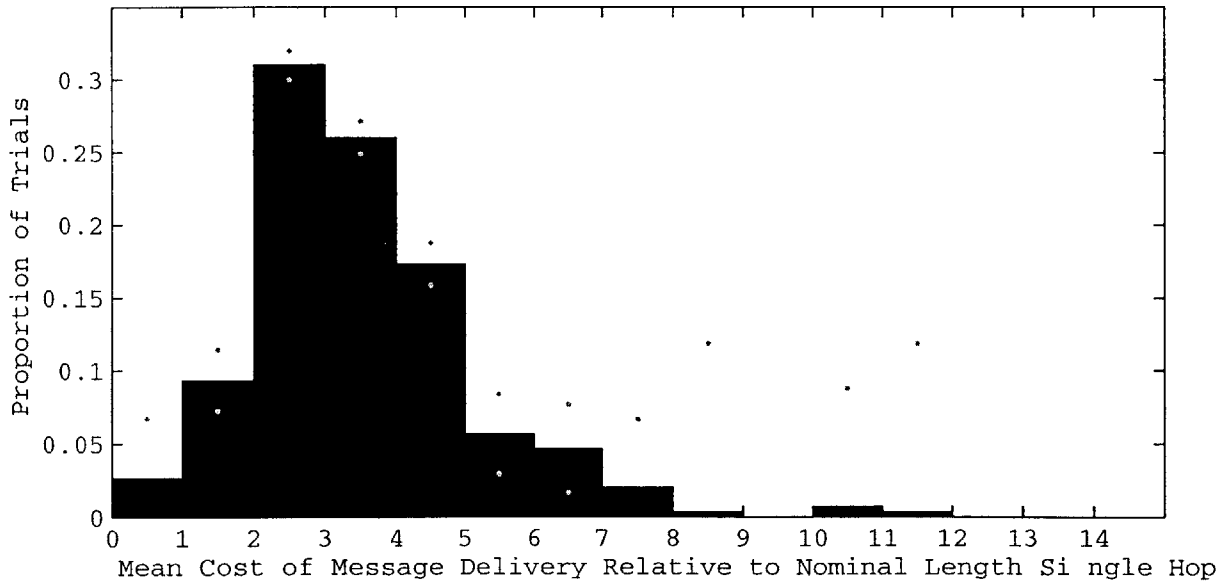


FIGURE 3-41. Mean power cost of message delivery from nodes to the lander for the Kelso Dunes "landing site." Power cost is relative to the nominal distance single-hop power cost, and is based on a minimum cost path analysis of the distributed system network topology.

For the Kelso Dunes "landing site" 90% of the nodes are connected to the lander only 18% of the time, a big change from the Soda Lake landing site. The normalized mean cost of message delivery curve has also changed significantly, and the expected normalized mean cost of message delivery has jumped from 1.7 for the Soda Lake site to 4.6 for the Kelso Dunes site.

3.6.3 Assessing the Results

The node connectivity results for Soda Lake (Figure 3-38) are consistent with the mostly very flat terrain of that area. The few outliers illustrate that some possible lander positions may place the distributed sensor pods in rough terrain (e.g., resulting in very low connectivity between pods). The expected mean cost of message delivery of 1.7 (as compared to a single-hop 300 meter transmission cost of 1) suggests that on average, few hops are required to transmit messages from sensor pods to the lander (this is partially a product of there being more nodes closer to the lander). The failure of the mission architecture to meet the proposed requirement can be dealt with in several ways, such as increasing the density of nods, developing nodes with higher effective heights, decreasing the node dispersions, or decreasing the lander dispersions (e.g., creating a higher likelihood of landing on the flat lakebed). Node heterogeneity might also be explored: a subset of nodes might be designed or developed that have better connectivity capabilities such as better line-of-sight capability or higher transmit power capabilities. In addition, the limitations of the line-of-sight visibility channel model should be considered: evaluating the node performance with a more accurate channel model might improve the predicted connectivity. A sensitivity analysis might be performed to evaluate the most cost efficient way to meet the requirements.

The node connectivity results for Kelso Dunes (Figure 3-40) are consistent with the relative roughness of the Kelso Dunes area as compared to the flat Soda Lake lakebed. This roughness is also reflected in the normalized mean cost of message delivery. The sand dunes in the Kelso Dunes area significantly reduce node-to-node line-of-sight and therefore cause the minimum cost paths between nodes and the lander to be, on average, longer than in the Soda Lake area.

Differences in performance characteristics between landing sites (for example, the extent to which each landing site meets the proposed requirement, all other factors being equal) could be compared to the differences in surface topographical statistical properties in an attempt to quantify the effects of topography on mission architecture performance. For reasons of brevity, this analysis will not be performed here, but it is illustrative to examine the qualitative differences in the graph topologies for each landing site. Figure 3-42 shows the distributed system network topology during one randomly selected trial (one lander position) for each of the two landing sites. Nearly all of the nodes are connected to the lander in Figure 3-42 (A), while many are disconnected from the lander in Figure 3-42 (B). Typical network topologies for the Soda Lake site (A) may consist of up to several thousand edges, while for the Kelso Dunes site (B), network topologies typically consist of several hundred edges.

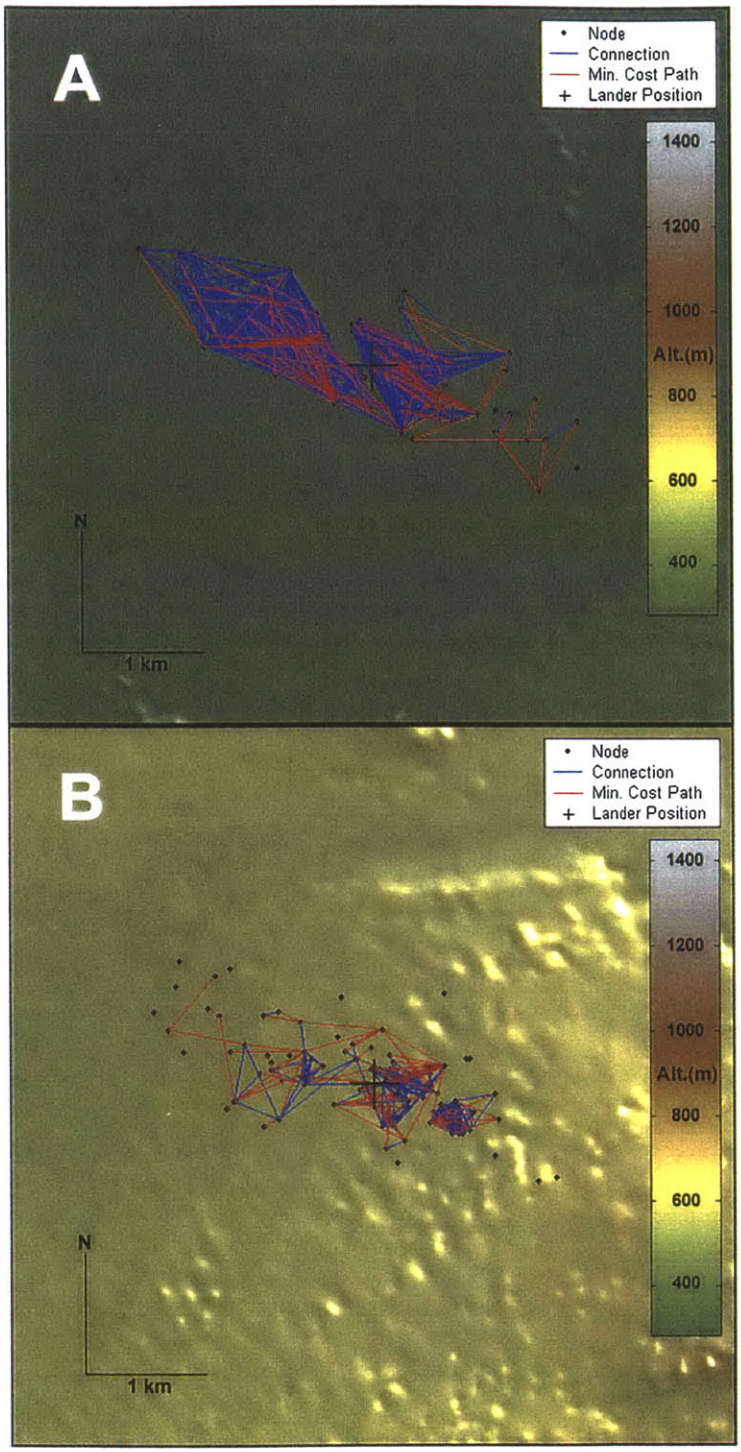


FIGURE 3-42. Sample network topologies for the Soda Lake (A) and Kelso Dunes (B) "landing sites." One lander position was randomly selected for each site using the previously described lander dispersion statistics, and 100 nodes were distributed randomly using the previously described sensor pods deployment dispersion statistics. Network connectivity is based on a line-of-sight visibility algorithm, and each node has an effective height of 30 cm above the surface, and a maximum communication range of 1 km. In A, the lander and distributed system positions are near the eastern margin of the Soda Lake dry lakebed. In B, the lander and distributed system positions are on the northern margin of the Kelso Dunes dune field.

It is clear that much additional analysis is possible just for this simple example of analyzing distributed system performance with respect to a single requirement. However, the example clearly demonstrates the power of the trade study process and some of the analysis techniques developed in this chapter.

3.7 Discussion

This chapter developed several conceptually simple but powerful tools for the design and analysis of distributed systems for surface exploration, and has reviewed many of the important trades that exist in these systems.

It is important to consider the limitations of both the modeling and analysis methods developed here, and to consider enhancements to these methods and to the trade study process. From the Mars lander and sensor network example it is clear that design optimization for distributed systems is a challenging task, and that distributed systems can quickly become very complicated. Hence reducing complexity in distributed system designs is an important consideration. These topics will briefly be considered in the following sections.

3.7.1 Modeling Methods and Limitations

Graph model of network topology. The network topology of distributed systems can be described by a graph, but the graph model has several limitations including its inability to represent the "fuzziness" of the real world, and its lack of dynamic character. Because the graph model is a snapshot of the distributed system structure at a specific time or over a limited period of time, the graph model requires *bounded asynchrony* between nodes in the graph. when bounds on asynchrony become weak, graph structure becomes meaningless; this limits the spatial size of system that graph structure can represent due to an upper bound of the speed of information propagation. The graph representation of the network topology of distributed systems is extremely versatile, and enhancements such as the use of directional edges or hypergraphs (in which more than two nodes can share an edge) can enhance the ability of a graph to appropriately model arbitrary network topologies involving spatially separated nodes.

Surface Modeling. Modeling a surface as a height field can allow easy and rapid characterization of a surface as a function of different scale lengths, at the expense of accurate characterization of small scale lengths. Effects of small surface features on distributed system elements may need to be treated in a statistical fashion (for example, simulating rocks in the path of a navigating rover) or characterized in some other fashion. The reliance upon a digital surface model means that the analysis techniques described here can only be performed for environments for which significant topographical information is available. Fortunately, for many proposed planetary surface exploration sites, this topographical information either is available or will be available (for example, by reconstruction of altitude information from high-resolution photographs). Sources of error must be considered in surface models, because small altitude errors can adversely affect line-of-sight visibility (and potentially could affect other connectivity metrics as well).

Likewise, representative terrain should be carefully selected. Non-cartesian projections of surface data could also be used to model surface curvature and line-of-sight visibility computations should account for this curvature, instead of computing line-of-sight visibility with a maximum range criterion. In some analysis, other surface characteristics (besides topography) may need to be modeled, including such characteristics as the spatial distributions of thermal inertia, dust, or terrain slopes. These characteristics may be especially important when modeling mobile nodes.

Connectivity and Communication Channel Modeling. While line-of-sight visibility served as a channel model for the purpose of this chapter, better channel models are available and may need to be used for accurate prediction of distributed system performance. A simple enhancement to the line-of-sight visibility channel model might be a metric of how close two nodes are to line-of-sight (e.g., what fraction of the path between two nodes is not line-of-sight). Antenna pointing issues and antenna patterns should be taken into account.

3.7.2 Analysis Methods and Limitations

Network topology analysis. Connectivity of distributed system nodes can be measured in many ways, and it is important to be sure that the right connectivity metric is being used to analyze a given network topology. For example, the shortest (minimum cost) path analysis used in the trade study example computes the minimal cost paths between each node and a central node (the lander), but does not provide a global minimum cost network topology (the choice of connectivity metric must take into account the purpose of the analysis). Shortest path analysis and other approaches are used in Chapter 5. Even so, this thesis only scratches the surface of the existing literature on the applications of graph theory for characterization or optimization of networks.

Topography characterization. The best fit power law scaling exponent can be used to characterize many surfaces over scales of several orders of magnitude, but there may be spatial variation in power spectral density magnitude or slope. Analysis of surface roughness and slope (along with other surface properties) becomes more important when mobility of system elements is considered. Topographic statistics can be used to evaluate directional dependence of topographic features, but one must always consider that any analysis is based upon a limited and potentially erroneous set of topographic data. Understanding the error sources in the model is critical to assessing the appropriate level of confidence in the analysis results.

Analyzing Quality of Service. Additional analysis tools are required to conduct distributed system trade studies for quality of

services parameters. The models and analysis methods described in this chapter treat individual agents as nodes without defining any internal node structure. In order to facilitate many of the trades described in this chapter, including quality of services trades, internal models of nodes are required. Simple internal node models might allow for simulation of routing algorithms. The simulation of routing with incomplete information is especially important for distributed systems in which (1) network topology changes fast enough that router updates would require a significant proportion of the network bandwidth, or (2) network size imposes a burden upon the storage capabilities required to maintain a local store of global network state (in this case, routing based on local information is required for scalability).

3.7.3 Trade Study Process

The trade study process laid out here is simple in form but can be computationally intensive. Incorporation of the trade study process into another design evaluation approach may reduce the computational burden by helping the designer to more quickly optimize a design. The trade study example illustrated how one type of trade (level of connectivity between a sensor network and a lander versus landing site) can be assessed and how trade studies might be related to system requirements. Many trades can easily be framed in terms that are soluble with the analysis tools described in the chapter, but for many other trades, new analysis tools will need to be developed.

3.7.4 Reducing system complexity

The goal of system architecting is to reduce complexity [Rechtin, 1991]. Several approaches can be used to mitigate some of the complexity imposed by distributed systems.

Complexity arises from the sheer number of interfaces between elements of distributed systems: complexity is related to connectivity of system elements. Consequently, reducing the number of interconnections between system elements while maintaining the number of interconnections required for path diversity (redundancy) requirements, can help reduce system complexity.

For large systems, the used of heterogeneous nodes for different functions (e.g., node functional specialization) can be used to develop a hierarchical structure that may promote efficiency and reduce complexity by aggregation. Partitioning of the distributed system into spatial structures with minimal external communication and maximum internal communication is another approach to minimizing system complexity [Rechtin, 1991].

3.7.5 Conclusions

This chapter has focused on the description of a class of distributed systems for surface exploration from the system perspective. A host of modeling and analysis tools have demonstrated how the network topology and performance of distributed systems can be characterized and related to the surface characteristics of the environment. Major trades of distributed systems have been defined, and a simple process for pursuing trade studies has been proposed and demonstrated.

While the modeling and analysis tools here have expressed some of the power of distributed systems, it is important to move beyond the view of distributed systems as static networks of nodes and to consider distributed systems as networks of mobile agents with dynamic behavior. Chapter 5 will work closer toward that ideal by demonstrating how individual agents can utilize a distributed system to support planning and execution of traverses, and other exploration activities.

While robotic agents can be designed with certain agent behaviors and capabilities in mind, humans have a long history of exploration strategies and techniques. To understand how human and robotic agents will fit into future exploration systems, Chapter 4 assesses how humans utilize information and communication during field exploration and extravehicular activity, and develops operational concepts for planning and support of extravehicular activity for planetary surface exploration. In Chapter 5 we will return to the modeling and analysis methods developed in this chapter and apply them in a structured way to agent mobility and traverse planning for individual agents, borrowing ideas from both human and robotic exploration.

3.8 References

Aharonson, O., Zuber, M.T., and Rothman, D.H., *Statistics of Mars' Topography from the Mars Orbiter Laser Altimeter: Slopes, Correlations, and Physical Models*, submitted to J. Geophysical Research, 2000.

Burton, John H., Project Manager, and Yoha, Robert, Principal, *GeoSAR: A Radar Based Terrain Mapping Project, Year 2 Research and Development Status Report*, California Department of Conservation, 1996 (*http://www.consrv.ca.gov/radar/geosar/year2rpt/toc.html*).

Chakrabarti, S. and Mishra, A., *QoS Issues in Ad Hoc Wireless Networks*, IEEE Communications Magazine, February 2001.

Chen, S., *Routing Support for Providing Guaranteed End-To-End Quality of Service*, PhD Thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1999.

CMEX: Center for Mars Exploration, Ames Research Center, National Aeronautics and Space Administration, Moffet Field, California, 2001 (*http://cmex.arc.nasa.gov*).

Cohen, R., *Empirical Methods for Artificial Intelligence*, MIT Press, Cambridge, Massachusetts, 1995.

Delin, K.A., *JPL Sensor Webs Project*, Jet Propulsion Laboratory, Pasadena, California, 2001 (*http://sensorwebs.jpl.nasa.gov*).

Diestel, R., *Graph Theory*, Springer-Verlag, New York, New York, 1997.

Dodds, P.S., and Rothman, D.H., *Scaling, Universality, and Geomorphology*, Annu. Rev. Earth Planet. Sci. 2000. 28:571-610.

Drezner, Z., *Facility Location - A Survey of Applications and Methods*, Springer, New York, Berlin, 1995.

Gershenfeld, N., *The Nature of Mathematical Modeling*, Cambridge University Press, Cambridge, United Kingdom, 1999.

Gershenfeld, N., *The Physics of Information Technology*, Cambridge University Press, Cambridge, United Kingdom, 2000.

Gross, J., and Yellen, J., *Graph Theory and Its Applications*, CRC Press, Boca Raton, Florida, 1999.

Havinga, P., Smit, G, and Bos, M., *Energy-Efficient Adaptive Wireless Network Design*, IEEE 2000.

Jilla, C.D., Sedwick, R.J., and Miller, D.W., *Application of Multidisciplinary Design Optimization Techniques to Distributed Satellite Systems (AIAA-99-4632)*, AIAA Space Technology Conference & Exposition, Albuquerque, New Mexico, September 28-30, 1999.

Kreslavsky, M.A. and Head, J.W. III, *Kilometer-scale slopes on Mars and their correlation with geologic units: Initial results from Mars Orbiter Laser Altimeter (MOLA) data*, J. Geophysical Research, Vol. 104, No. E9, p. 21911-21924, September 25, 1999.

Kreslavsky, M.A. and Head, J.W. III, *Kilometer-scale roughness of Mars: Results from MOLA data analysis*, J. Geophysical Research, Vol. 105, No. E11, p. 26695-26711, November 25, 2000.

Lewicki, S. A., and Zong, J., *Multi-angle imaging spectro-radiometer, Level 1 Ancillary Geographical Product Algorithm Theoretical*

Basis (JPL D-13400, Rev. A), Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 1999 (*http://eospso.gsfc.nasa.gov/ftp_ATBD/REVIEW/MISR/ATBD-MISR-05/atbd-misr-05.pdf*).

Mohandas, S. U. and Sandgren, E., 1989, *Multiobjective Optimization Dealing with Uncertainty*, Advances in Design Automation - Design Optimization (Ravani, B., ed.), ASME, Vol. 19-2, pp. 241-248, Montreal, Canada, September 17-21, 1989.

Rechtin, Eberhardt, *Systems Architecting*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Shaw, G.B., *The Generalized Information Network Analysis Methodology for Distributed Satellite Systems*, PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1999.

Smith, Zuber, Frey, Garvin, Head, Mhleman, Pettengill, Phillips, Solomon, Zwally, Banerdt, Duxbury, Golombek, Lemoine, Neumann, Rowlands, Aharonson, Ford, Ivanov, McGovern, Abshire, Afzal, and Sun, *Mars Orbiter Laser Altimeter (MOLA): Experiment Summary after the First Year of Global Mapping of Mars*, Journal of Geophysical Research, accepted September 16, 2000.

Talbot-Stern, J., *Design of an Integrated Mars Communication, Navigation, and Sensing System*, AIAA Paper 2000-0011, 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 10-13, 2000.

Tutschku, K., Leibnitz, K., and Tran-Gia, P., *ICEPT - An Integrated Cellular Network Planning Tool*, IEEE, 1997.

Wertz, J.R., and Larson, W.J., *Space Mission Analysis and Design, 2nd Edition*, Kluwer Academic Publishers, Norwell, Massachusetts, 1992.

Wood, J., *The Geomorphological Characterisation of Digital Elevation Models*, Ph.D. Thesis, Department of Geography, University of Leicester, United Kingdom, 1996 (*http://www.geog.le.ac.uk/jwo/research/dem_char/thesis/index.html*).

The principal difference between an adventurer and a suicide is that the adventurer leaves himself a margin of escape (the narrower the margin, the greater the adventure). A margin whose width and breadth may be determined by unknown factors, but whose successful navigation is determined by the measure of the adventurer's nerve and wits. It is always exhilarating to live by one's nerves or towards the summit of one's wits.

Tom Robbins, *Even Cowgirls Get the Blues* (1976)

4 *Observing Exploration*

In *Even Cowgirls Get the Blues* (1976), Tim Robbins states that adventurers, if they survive, do so "by the measure of the adventurer's nerve and wits." However, surviving and prospering during exploration requires something else: it requires the collection and analysis of information, both for the purpose of disseminating that information to others, and for purposes of decision making that ultimately determine survival and mission success. Historic journeys such as the Lewis and Clark expedition of 1804-1806 (Figure 4-1) would not have been possible without such carefully collection and analysis of information.

Human explorers have adopted techniques and technologies for the collection, analysis, and dissemination of information that enhance the value of exploration and help them cope with the reduced margins between death and survival during exploration in extreme environments.

In the context of planetary surface exploration, this chapter attempts to answer the following questions:

1. How do human explorers collect, analyze, and disseminate information?
2. How do astronauts use information in the planning and execution phases of extravehicular activity?
3. How can explorers utilize distributed systems to enhance the value of exploration, and their margins for survival? How can these systems enhance self-sufficiency, autonomy, and group collaboration?

To attempt to answer these questions, the author first attempted to develop a user-perspective of field exploration by participating in a month-long field geologic mapping project, as described in Section 4.1. Section 4.2 presents a case study of the use of information and the role of communication during extravehicular activity, focusing on arguably the nearest analog to surface exploration on the Martian surface - lunar surface operations during the Apollo missions. These two case studies, while certainly not generalizable to all planetary surface exploration, provide insights that will be useful for the development of systems to support human exploration of the Martian surface. Section 4.3 incorporates the results of the two case studies and discusses opportunities for enhancing the process of exploration through the use of distributed systems. Strategies for coping with the challenges of extravehicular activity and traverse planning on the Martian surface are discussed.

FIGURE 4-1. A map from Clark's journal showing the course and camping locations of Lewis and Clark from September 18-20, 1805 during their 1804-1806 expedition across the Louisiana Purchase. Clark often took multiple measurements with his sextant and compass to make position estimates, and often made dead-reckoning distance estimates. The numbers on the left side of the map are likely to be a series of distance estimates that Clark wanted to sum. From Thwaites (1957), *Original Journals of the Lewis and Clark Expedition*, Volume 3, p. 72.

4.1 Geologic Mapping: The User Perspective

A geologic mapping project of the Bird Spring Mountains, Nevada was undertaken in January, 2001 by graduate students and faculty in the MIT Department of Earth, Atmospheric, and Planetary Sciences, and by the author as part of a two-semester graduate course in field geology. Figure 4-2 illustrates the location and scale of the mapping area.

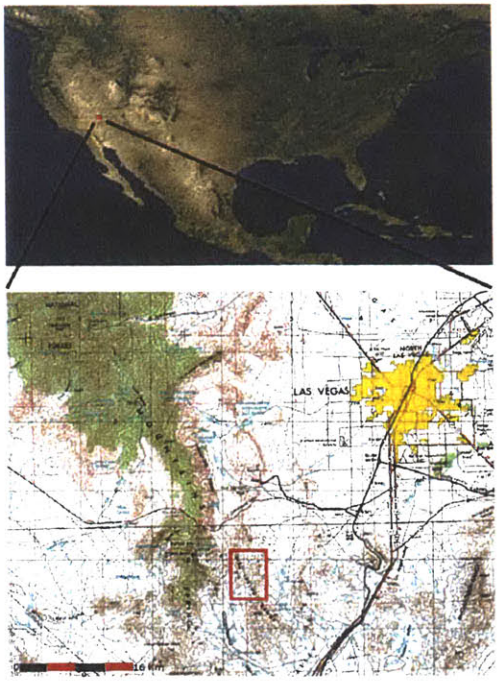


FIGURE 4-2. The geologic mapping project focused on mapping the north-west portion of the Bird Spring Mountains and the Bird Spring Thrust. The field area was approximately 40 km to the south-west of the city of Las Vegas, Nevada (Topographic base image courtesy of United States Geological Survey).

4.1.1 Overview

The primary purpose of the geologic mapping project was to develop and practice the art and science of geologic mapping by mapping the bird spring formation and the bird spring thrust fault, a small thrust fault that is part of a series of thrust faults in the southwestern United States that developed during the evolution of the Cordilleran orogenic belt.

A secondary purpose for the author was to draw analogies between the experience of geologic mapping in the Bird Spring Mountains and the process of planetary surface exploration by future astronauts by collecting quantitative and qualitative data about the geologic mapping process, during geologic mapping.

The author sought to make qualitative observations about how future extravehicular activity systems can support the field geology process and how distributed architectures can support field exploration. The goal of taking quantitative measurements during geologic mapping was to capture the spatial and temporal characteristics of the process, and to understand the constraints imposed on the process of geologic mapping by surface topography.

This often proved to be challenging, and the conflicts of being both a subject and an observer and the responsibilities as a participant in the geologic mapping process often limited the quantity and quality of the data collected. Nevertheless, both the qualitative and quantitative data collected during this experience can provide insights which will be useful for the development of distributed systems to support the human and robotic exploration of the Martian surface.

4.1.2 Background

Field exploration, including field geology, will be a major part of future human and robotic exploration of the Martian surface. Field geology depends on iterative observation, interaction, and hypothesis generation and evaluation, and is often highly exploratory and unstructured. Field geology is an appropriate activity for future human explorers working with robotic explorers, and will drive

future extravehicular activity system design. Furthermore, this process can be improved through analysis and appropriate application of technology. Human explorers excel at field geology and field exploration as compared to robotic explorers because of their comparatively greater abilities to accept complex input, rapidly integrate data from multiple sources and at multiple scales, generate and test hypotheses, learn from past experiences, cope with unexpected events, navigate efficiently through complicated or varied terrain, and communicate in a powerful and flexible manner.

This study is focused on four basic questions of field geology:

- Where to go?
- How to get there?
- What data to collect?
- How to share the data?

Another obviously important question the meaning of collected data. Interpretations of data are important in that they affect the value of data and therefore affect the answers to the four questions posed above. While this is an important and interesting issue, it is outside the scope of this thesis.

Where to go? A field exploration site is typically chosen with specific problems in mind. Surface exploration of Mars is no different, except that because of the extensive previous and ongoing robotic exploration, a wealth of data exists both on local and global scales. This provides an unprecedented opportunity to identify local and regional geological sites of interest where the tasks and processes required to answer particular questions are not well defined or are extremely complex (where humans can add a great amount of value to the exploration process). Examples of field exploration tasks that may require humans include sample collection, geologic mapping, drilling, and setup, repair, or breakdown of experiments and equipment. If the problem requires access to rock outcrops, the explorers must get to the outcrop. In geologic mapping, one generally maps for the problem, not the area [Burchfiel, 2001]. Oftentimes (as will be the case with future Mars surface exploration) many "points of interest" are known before a specific field exploration activity begins. Many additional points of interest are identified as field exploration proceeds, and the general goals and even the problem to be solved may change as exploration evolves and data is collected and analyzed.

How to get there? Typically, multiple methods of transportation may be available for traveling from some established base of operations to a work site or field area, but trade-offs exist between different methods of transportation. Large vehicles may provide mechanisms to carry large cargoes, but may offer significantly less mobility than smaller vehicles or foot travel depending on the ter-

rain. The general approach is one of *energy conservation*, from the perspective of the explorers. Considerations may include the time spent utilizing different types of transportation, and the areas visible along the route (for example, driving along a route might provide a quick opportunity for out-of-the-window geology while covering a lot of area). The likelihood of revisiting a work site might also be considered when planning what form of transportation to use when traveling to and from the work site. Communications coverage and the capability to communicate between members of the "field party" (a group of explorers) might also be considerations.

What data to collect? The data to be collected at a work site or in a field area is clearly highly dependent upon the specific problem to be solved, but in general may include spatial and temporal information (where the object of study is located, and when is it being observed), a description of the object of study, the relationship of the object of study to other nearby objects, the orientation of the object of study (if relevant), sketches, photographs, and other more specialized data.

How to share the data? Clearly, explorers can share the raw data products generated by exploration, such as written records, sketches, photos, or other specialized media or data formats. Discussion can be held in the field, or upon returning to some base of operations. For human explorers, at the heart of sharing the results of exploration is communicating a mental model to other explorers. Group discussions can lead to additional hypotheses, eliminate existing hypotheses, and can provide new questions that require additional data collection in the field. Different types of field exploration may have different types of models to communicate: enhancing the ability to create and evaluate these models will add significant value to the exploration process.

4.1.3 Methods

A mapping team of approximately twelve people was divided up into three groups, and each group was assigned a group mapping area. Figure 4-3 illustrates the location of each group mapping area in relation to the camp site. Mapping activities began on January 9, 2001 and were completed on January 31, 2001.

Table 4-1 provides the activity allocation by days for this 23-day period. Out of the 23 days, several days were devoted to group tours of regional and local geology, and a total of two weeks time was devoted to geologic mapping proper (the analysis in this section will focus solely on these 14 days). About once a week, a trip to Las Vegas was made to allow participants to satisfy their hygiene needs such as showering and laundering of clothing. Several days of persistent snow cover late in January allowed the group to take a

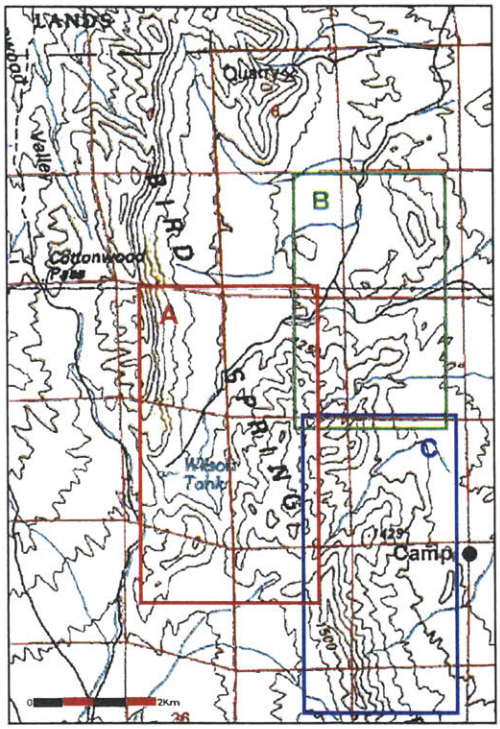


FIGURE 4-3. Approximate field areas in the Bird Spring Mountains for the three mapping teams are shown in relation to the group campsite. The author was assigned to mapping area A. Consequently, mapping traverses typically started and ended with a bumpy drive between camp and the north end of the field area via a route that wrapped around the north Bird Spring Mountains. The extent of the map shown above is 115W 26'55" to 115W 22'40" and 35N 54'16" to 35N 59'27" with North up and 50 meter contour intervals (Topographic base image courtesy of United States Geological Survey).

TABLE 4-1. Activity Allocation by Days

Activity	Days
Regional Geology Tour	1
Local Geology Tours	3
Geologic Mapping	13.5
Weekly trips to Las Vegas	3
Death Valley Field Trip*	2.5
Total	**23**

*due to snow days

field trip to Death Valley that focused on metamorphic core complexes and the Death Valley basin-and-range system.

In a typical day of mapping, the author's group would depart camp at about 8 am, drive a vehicle to the field area, and begin mapping by about 9 am. The group would return to the vehicle by 4 or 4:30 pm in order to return to camp by close to 5 pm. Some evenings were spent working in a group tent on the assembly of a stratigraphic column for the Bird Spring Formation or on the assembly of a communal map of the Bird Spring Mountains.

Equipment for Geologic Mapping. A clipboard with a plastic cover provided a mapping surface for a 1:20000 topographic base map. Mapping was performing with pencils and a small protractor for measuring angles and drawing straight lines. A geology hammer and pocket knife were also carried. A Brunton^{TM} compass was carried for sighting and for measuring the orientation of bedding planes or fault planes. A hand lens was carried for inspection of rock grains, and a small plastic bottle with dilute hydrochloric acid was carried for testing the calcareous content of outcrops.

Personal items. Personal items carried during geologic mapping typically consisted of a water bottle, lunch, extra food, and extra clothing. A group first aid kit was also sometimes carried by the author. A reasonable estimate of the total weight of all items carried (personal and otherwise) is about 9-11 kg (20-24 lbs).

Equipment for Data Collection. The equipment used for data collection is summarized in Table 4-2. While positioning in the field was accomplished by observation and direct reference to the topographic base map, a global positioning system unit was used to record *waypoints* (points along a track; position fixes). The waterproof stuff sack was used to reduce the exposure of the electronic equipment to moisture.

Qualitative Observations. Qualitative observations were made in the waterproof field notebook during geologic mapping, and in a personal journal when the author was in his tent during late evening or early morning hours. These observations were later compiled separately from notes pertaining to geologic mapping.

Waypoint Data Collection. Waypoints were collected at "important" points in a traverse using the MARK WAYPOINT feature of the Global Positioning System unit. Waypoints were always taken at any geological station where data (such as notes relating to specific outcrops, orientation measurements of bedding or fault planes, or digital photographs) were collected. During long traverse segments, waypoints were often taken when the character of the traverse (slope or direction) changed greatly or where the mapping group stopped to rest or converse. Waypoints were also taken at the

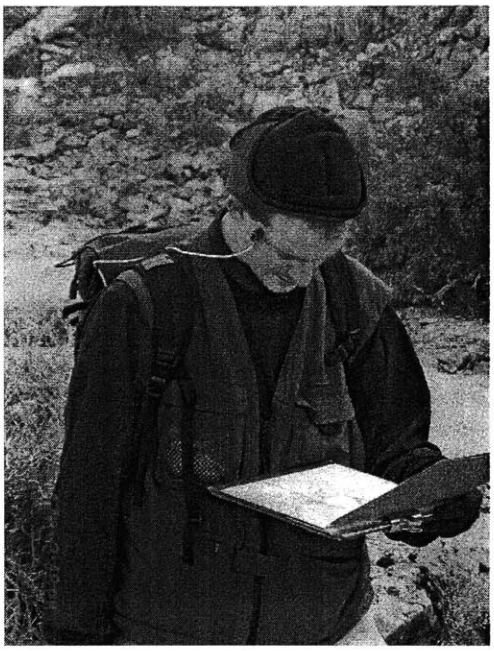


FIGURE 4-4. The author is shown here in the field while conducting a preliminary recording with the pulse-oximeter sensor. The serial cable from the global positioning system unit can also be seen near the top pocket of the author's backpack.

TABLE 4-2. Data Collection Equipment

Waterproof Notebook and Pencils
Hand-held Global Positioning System unit (Garmin eTrex^{TM})
Digital Camera (Sony DSC-P1)
Sub-notebook Computer (Sony Picturebook with Extended-Life Battery Pack)
Pulse Oximeter Sensor (Nonin Medical Reflectance Sensor and XPod Pulse Oximeter Module)
Waterproof Stuff Sack

starting and ending locations of a traverse. The sub-notebook computer was also used several times to record waypoints directly from the global positioning system unit. During these periods, the global positioning system unit was placed in the top pocket of the author's backpack (for proper satellite reception) and the computer, housed in the waterproof stuff sack in the main backpack compartment, recorded waypoints via a serial cable attached to the global positioning system unit. The MARK WAYPOINT feature was not used during these periods due to lack of access to the global positioning system unit.

Digital Images. The Sony DSC-P1 digital camera was used to collect digital images of important features, and images were frequently downloaded to the sub-notebook computer for storage. Recharging of the digital camera batteries and the laptop was accomplished during evenings in camp or during the drive to and from the field area using a standard direct-current to 120 volt alternating-current inverter powered by a vehicular 12 volt system.

Pulse-oximeter Data Collection. A pulse-oximeter sensor recorded blood oxygen saturation and heart rate during geologic mapping using a serial-line connection to the computer. Incoming samples of the heart rate were time stamped and stored on the computer hard drive. The reflectance-based pulse-oximeter sensor was affixed to the author's forehead near the temporal artery, and proper operation and data collection was confirmed by a quick test recording of data on the computer before the computer was stowed in the author's backpack for a more extended period of recording.

4.1.4 Results and Analysis

Data Collected. Written observations were recorded during geologic mapping and during evenings in camp. Waypoints were collected at 442 points of interest. 206 digital photographs of geologic data and geologic mapping activities were made. Physiological data was collected for two periods of durations of about 32 minutes and of about 3 hours. Testing of the pulse-oximeter sensor prior to the third planned pulse oximeter data collection session yielded an erratic incorrect pulse-pressure signal and no further physiological data was collected during geologic mapping.

Waypoint Analysis. A region of the Cottonwood quadrangle 7.5 minute digital elevation model was extracted to yield a digital elevation model for the field area that encompasses nearly all of the waypoints of interest. Table 4-3 gives the bounding box of the selected region. Altitude residuals between the waypoints and the Cottonwood digital elevation model (after conversion of the waypoints to the North American Datum 1927) are shown in Figure 4-5, and provide an estimate of waypoint position errors. Figure 4-6 shows the locations of all waypoints collected.

TABLE 4-3. Digital Elevation Model Coordinates for the Region of Study

Direction	Minimum	Maximum
East-West	639675 m	646425 m
North-South	3974055 m	3980805 m

*Coordinates are Universal Transverse Mercator, Zone 11S

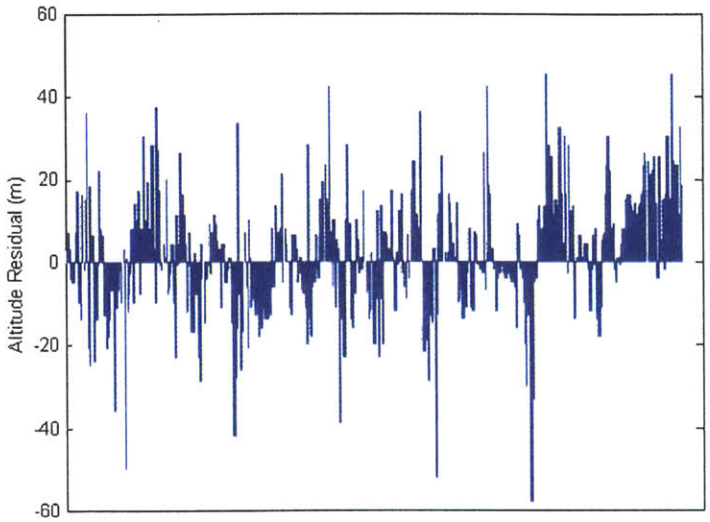


FIGURE 4-5. Altitude residuals are plotted for 435 waypoints, computed from the difference between waypoint estimated altitude and altitude according to the Cottonwood digital elevation model. These residuals provide an estimate of the error in the waypoint position estimate from the global positioning system unit assuming that all positioning error is in the vertical direction. It is important to understand that this estimate assumes that the digital elevation model is error free, which is certainly not the case. The mean and standard deviation of the altitude residuals are 1.63 m and 14.7 m, respectively (curiously, 1.6 m is approximately the height that the author holds the global positioning system unit while recording a waypoint while standing).

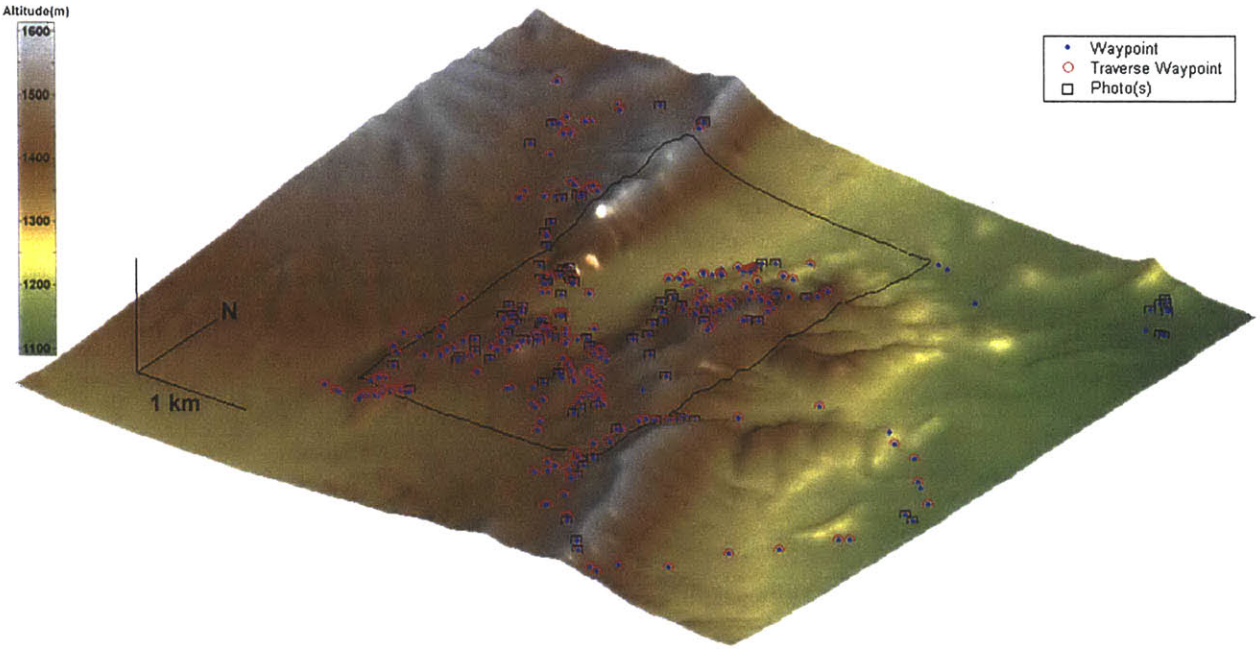


FIGURE 4-6. Waypoints collected during geologic mapping activities are plotted on a region of the Cottonwood, Nevada, digital elevation model. All waypoints are marked with blue dots. Waypoints used to define daily traverses are indicated by red circles. Waypoints where photographs were taken are indicated with black squares. The approximate boundary of mapping area A is outlined in black on the surface of the digital elevation model.

Traverse Analysis. A *traverse* (a collection of waypoints that form a path) was defined for 11 of the 13 full days of geologic mapping. Waypoints were not individually recorded during two days of geologic mapping. Traverses were defined for days 2, 4, 5, 6, 9, 11, 12, 13, 18, 22, and 23 of the geologic mapping project, where day 1 corresponds to January 10, 2001. Traverses were defined as an ordered collection of waypoints:

$$T = \begin{bmatrix} T_1 \\ T_2 \\ \cdots \\ T_n \end{bmatrix} \quad \text{(EQ 4-1)}$$

where each waypoint was defined as:

$$T_i = \begin{bmatrix} I & x & y & z & D & i & t_a & t_s \end{bmatrix} \quad \text{(EQ 4-2)}$$

where I is a global waypoint identifier, (x,y,z) defines the position of the waypoint in Universal Traverse Mercator coordinates (meters), D is the field day, i is a day-based waypoint identifier, t_a is the time of arrival at the waypoint in seconds since midnight, and t_s is the duration of time spent at the waypoint ("stay time") is seconds. The time of arrival and departure at waypoints were often recorded, and the times recorded when photographs were taken (often taken just before the author departed a geologic point of interest) were used to further constrain the period of time spent at each waypoint.

Traverses were analyzed in MATLAB using several subroutines and approaches. Traverses were visualized by projecting the traverses onto the digital elevation model. Metrics such as daily distance, elevation gain, and elevation loss were computed. The digital elevation model was used to estimate surface slopes, and a slope analysis of the traverses was performed. A load-carrying model, described in Chapter 5, was used to estimate the human metabolic cost of the traverses, based solely on the waypoint data and some reasonable assumptions.

To perform these analyses, the traverses were interpolated using either a temporal or spatial sampling interval. Interpolation was first performed in a straight-line manner between waypoint positions (in temporal sampling, a constant velocity between waypoints was assumed). Next, the interpolated traverses were projected onto the surface of the digital elevation model. Further analyses of the interpolated traverses were therefore based on the altitudes of the digital elevation model. Given the altitude residuals in Figure 4-5, there is little harm in basing traverse altitudes on the digital elevation model, rather than solely on the altitudes recorded by the global positioning system unit (global positioning system altitude errors are generally larger in magnitude than horizontal errors). This projection approach yields a better approximation of the actual path taken between waypoints than a constant climb rate or decent rate approximation (a straight-line approximation).

Figure 4-7 and Figure 4-8 show eleven traverses plotted on the surface of the Cottonwood, Nevada, digital elevation model.

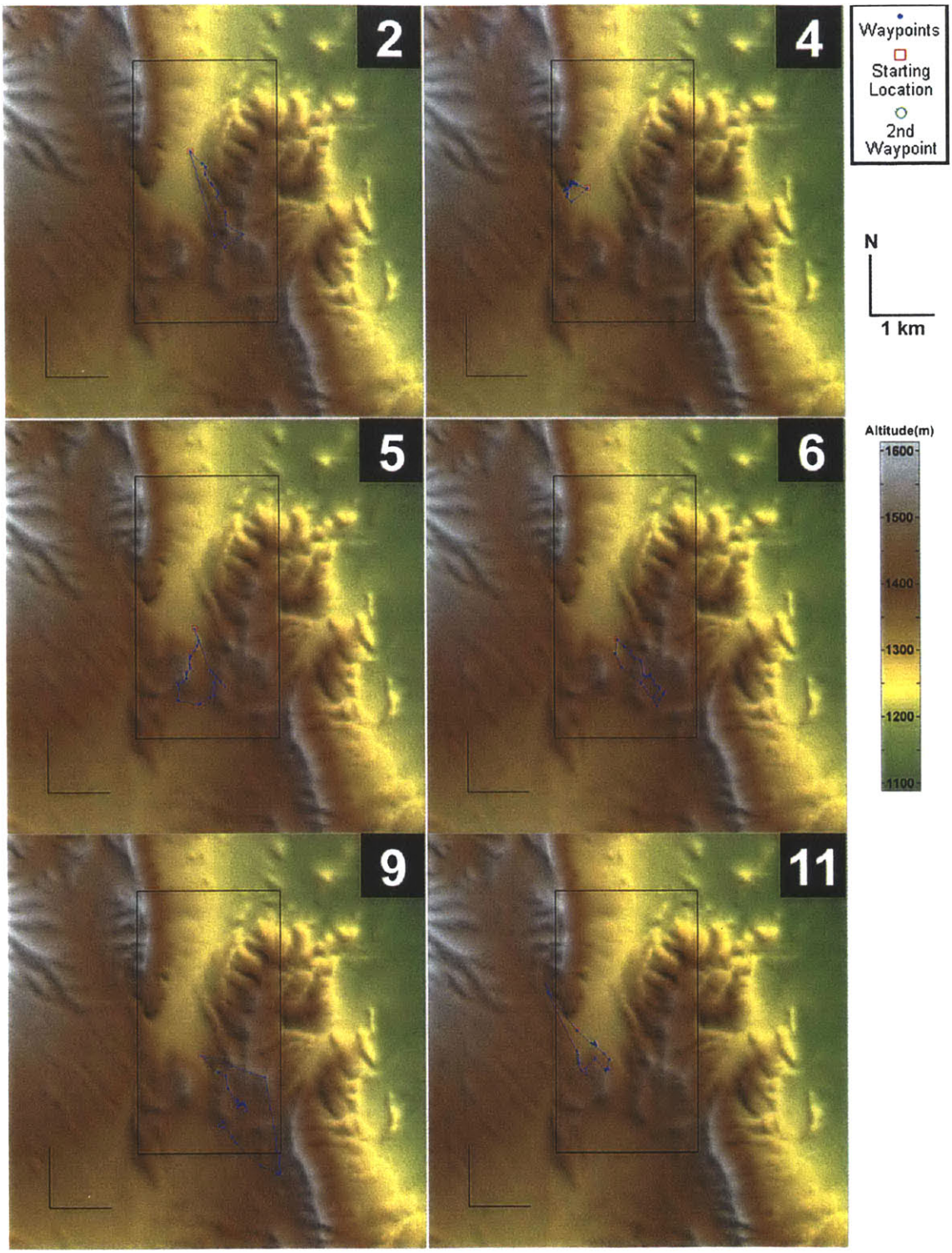
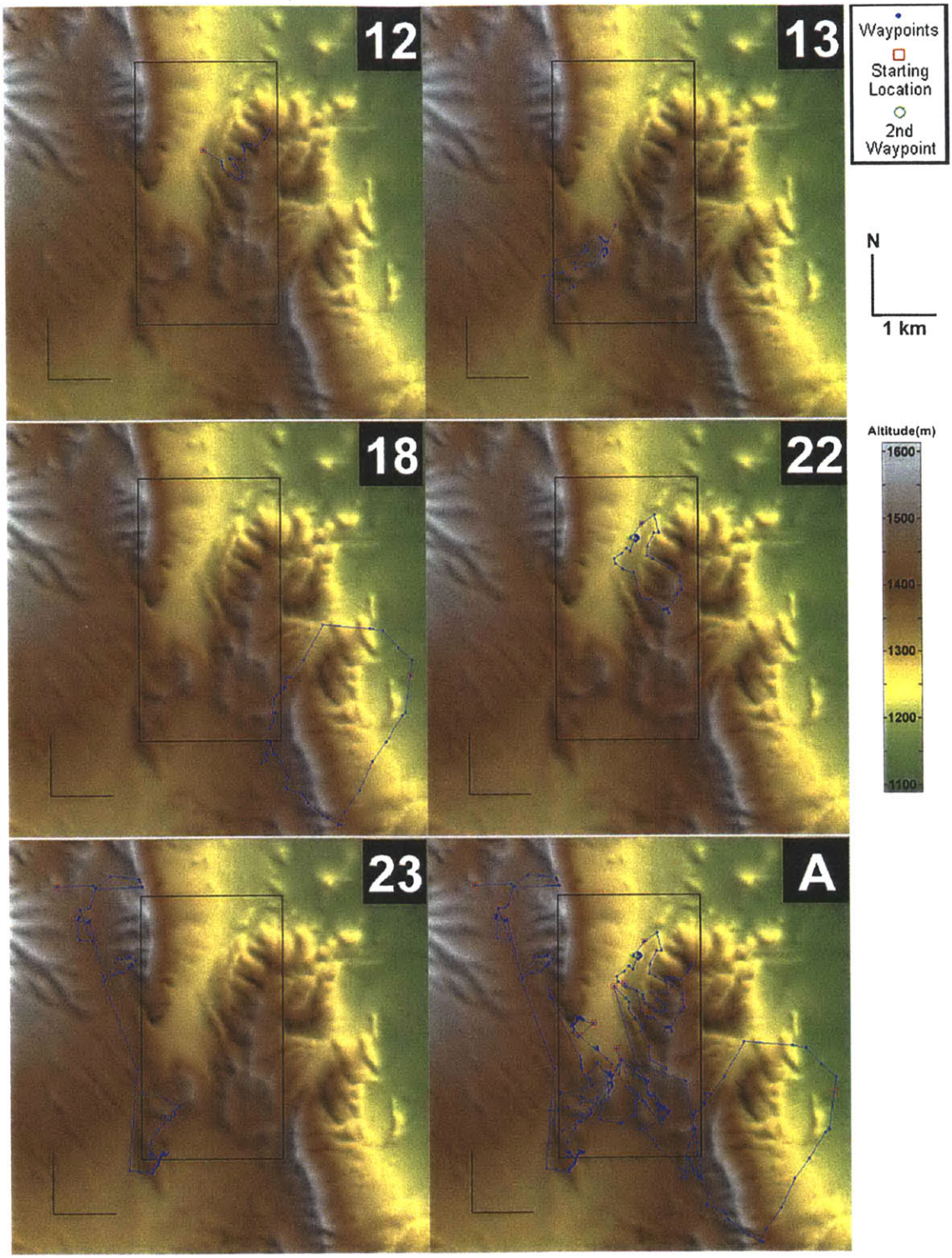


FIGURE 4-7. Traverses for mapping days 2, 4, 5, 6, 9, and 11. Scale bars (lower left) are 1 km. Mapping area A is outlined in black.

Notice that nearly all of the traverses in both Figure 4-7 and Figure 4-8 originate (traverse starting points are denoted by red squares) from the central valley, which was accessible to vehicles.



Daily distance and elevation gain and loss metrics were computed by sampling along each traverse (at a spatial sampling interval of 30 meters, the same as the resolution of the digital elevation

FIGURE 4-8. Traverses for mapping days 12, 13, 18, 22, 23. A final summary plot (A) shows all of the traverses.

model), projecting the resultant traverses onto the surface of the digital elevation model, and then computing the distance, elevation gain, and elevation loss along each interpolated traverse.

Figure 4-9 shows traverse distance (computed along the traverse, not in the horizontal plane) for each of the days in which a traverse was defined. Figure 4-10 shows the distance ascended as a function of the day.

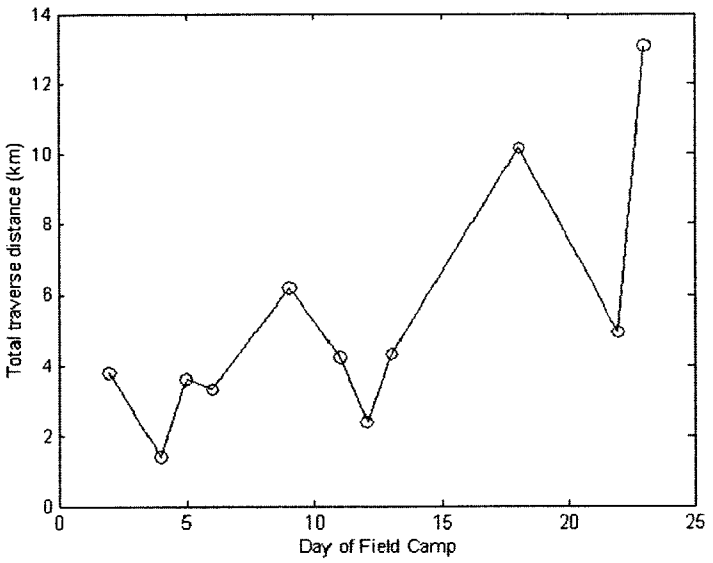


FIGURE 4-9. Traverse distance is plotted as a function of the day of the geologic mapping project. A general trend toward longer and longer traverses is noticeable, but there are significant variations in the total traverse distance. Some of the variations in the total traverse distance are likely due to differences in types of field geology activities: for example, most of day 4 was spent doing more-or-less detailed stratigraphy work, focused work in a (generally) very localized area. Less terrain may also be covered in structurally complicated areas, where mapping may proceed more slowly.

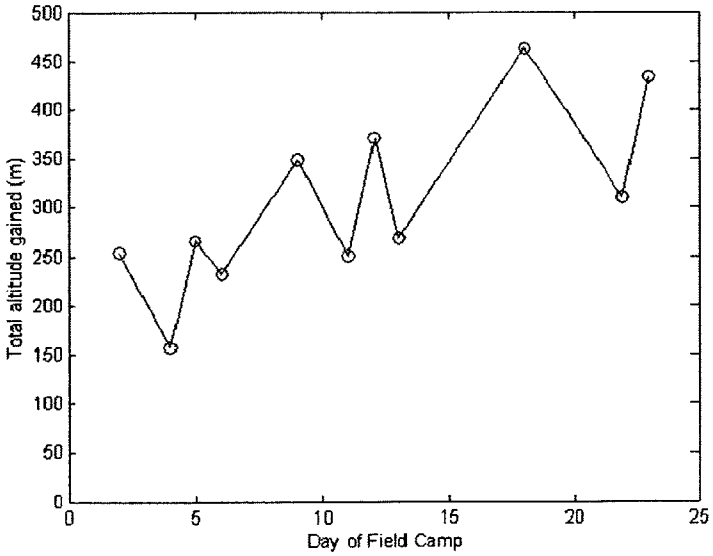


FIGURE 4-10. Altitude gained (total distance ascended) is plotted as a function of the day of the geologic mapping project. A general trend toward more and more altitude gain is noticeable, but there are significant variations in the total altitude gained. These variations are related to the variations in traverse distance seen in the previous figure. The total distance ascended was equal to the total distance descended during most days, except when a traverse did not return to the initial starting location (for example, the vehicle typically located at the starting location was moved, and met the geologic mapping group at another location).

The variations in the total distance ascended are related to the variations in the total traverse distance. Dividing the total distance

ascended by the total traverse distance for each day gives a metric of distance ascended per distance traversed. Figure 4-11 shows the distance ascended per distance traversed as a function of the day.

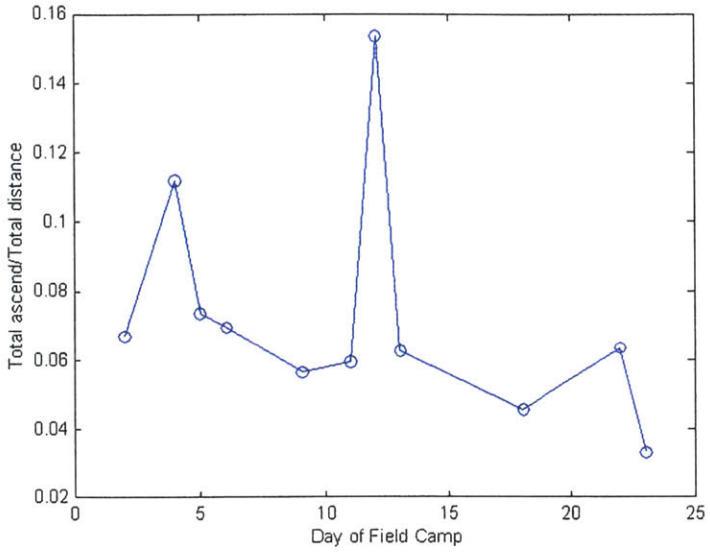


FIGURE 4-11. Distance ascended per distance traversed is plotted as a function of the day of geologic mapping activities for which traverses were defined. Distance ascended per distance traversed is more-or-less uniform (or perhaps trending slightly lower) except for days 4 and 12. Total traverse distance during days 4 and 12 is also low. One might hypothesize that work was done on steep slopes during these two days.

A slope algorithm (presented in Chapter 3) was used to compute a slope map of the field area; all traverses are plotted on a slope map in Figure 4-12.

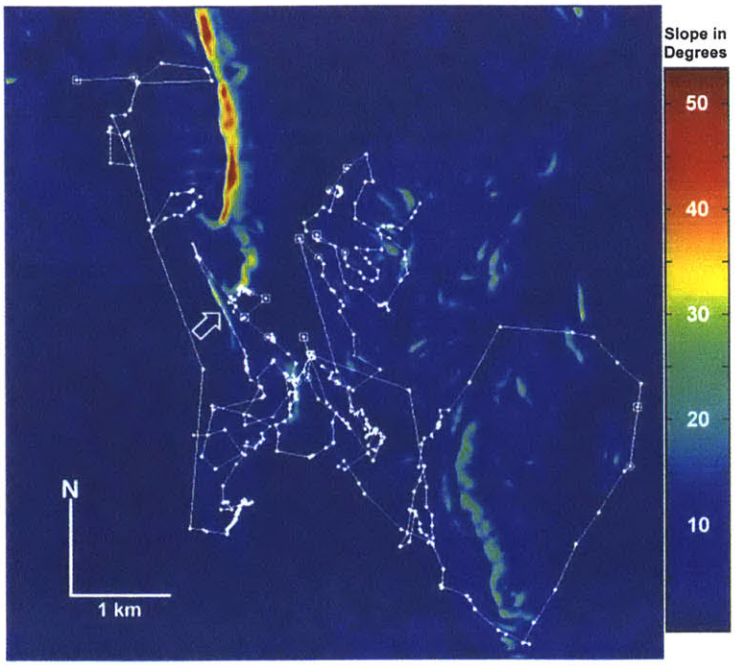


FIGURE 4-12. Traverses are plotted on a slope map of the field area. Many traverses seem to follow low-slope corridors. Some limitations of the data set are also visible: the white arrow points to where a traverse line between two waypoints crosses a high-slope area (a canyon wall). In this case, the route of travel was up the canyon (to the north-east of the canyon wall, heading north-west). Not enough waypoints were collected to accurately represent the traverse. Frequent and automated collection of waypoints could easily eliminate problems like this in future studies.

Further slope analysis was performed by projecting the interpolated traverses onto the slope map (the altitude z for each waypoint is replaced by the slope in degrees at that waypoint) and performing statistical analyses on the resulting traverses. Figure 4-13 shows a histogram of the slopes encountered on the eleven traverses.

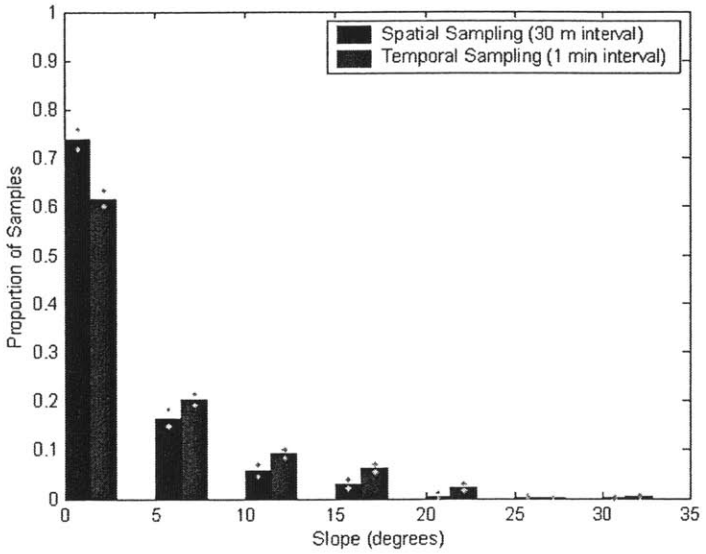


FIGURE 4-13. Surface slopes encountered during the eleven traverses were computed by spatially sampling or temporally sampling interpolated traverses and projecting them onto a slope map of the field area. Histograms for the collections of slope measurements were then computed. Green dots indicate 95% confidence boundaries for each proportion (assuming accurate slopes and no position errors). The spatial and temporal sampling proportions are based on 1744 and 4271 samples, respectively.

The analysis of Figure 4-11 suggests that traverses during days 4 and 12 might have been spent on generally steeper slopes than the other days. This possibility is explored in Figure 4-14 and Figure 4-15.

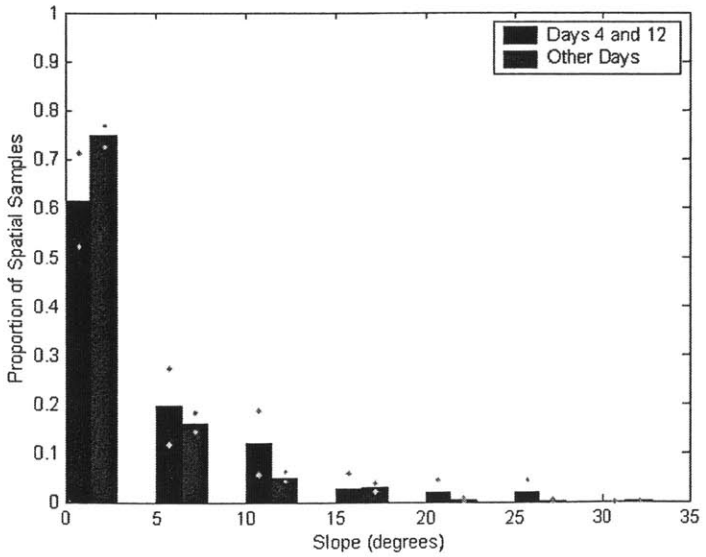


FIGURE 4-14. Surface slopes encountered during days 4 and 12 are compared to surface slopes encountered during the other traverses. This comparison is based on spatial sampling of the traverses, and thus answers the question: "Were the traverses of days 4 and 12 on steeper slopes than other traverses?" The proportions for days 4 and 12 are based on 107 samples, and the proportions for the other days are based on 1637 samples.

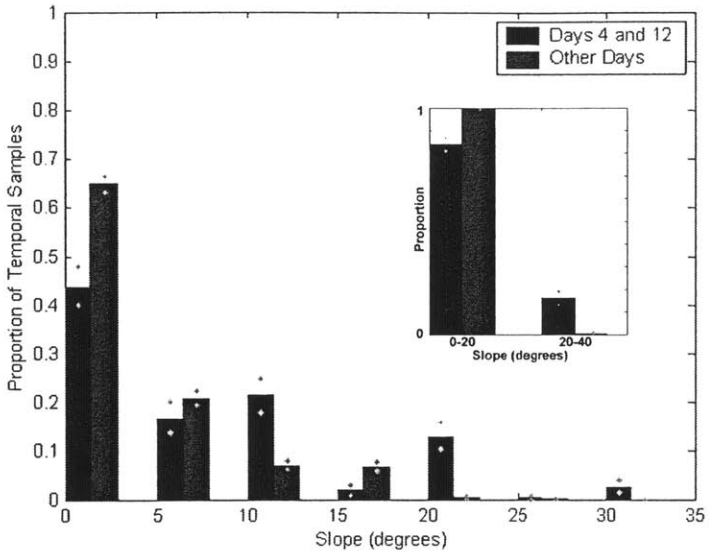


FIGURE 4-15. Surface slopes encountered during days 4 and 12 are compared to surface slopes encountered during the other traverses. This comparison is based on temporal sampling of the traverses, and thus answers the question: "Was more time spent on steeper slopes during days 4 and 12 than on other days?" The proportions for days 4 and 12 are based on 609 samples, and the proportions for the other days are based on 3662 samples. For the slope intervals of 10-15 degrees, 20-25 degrees, and 30-35 degrees, the separation of the confidence intervals indicates that more time was spent on steeper slopes (for those slope intervals). Re-partitioning the data into 0-20 degrees and 20-40 degrees (see inset) shows clearly that more time was spent on steeper slopes.

Conclusions based on this slope analysis should be limited, as the slope analysis is based upon assumptions of an accurate slope map, no position errors, and enough waypoints to accurately represent each traverse. It is already clear that in some areas, more waypoints would have more accurately represented the traverses, and that position errors certainly exist. In addition, computation of a slope map from a 30 m resolution digital elevation model biases the slope statistics. This low spatial resolution acts as a low-pass filter, and smooths the computed slopes as compared to the actual slopes.

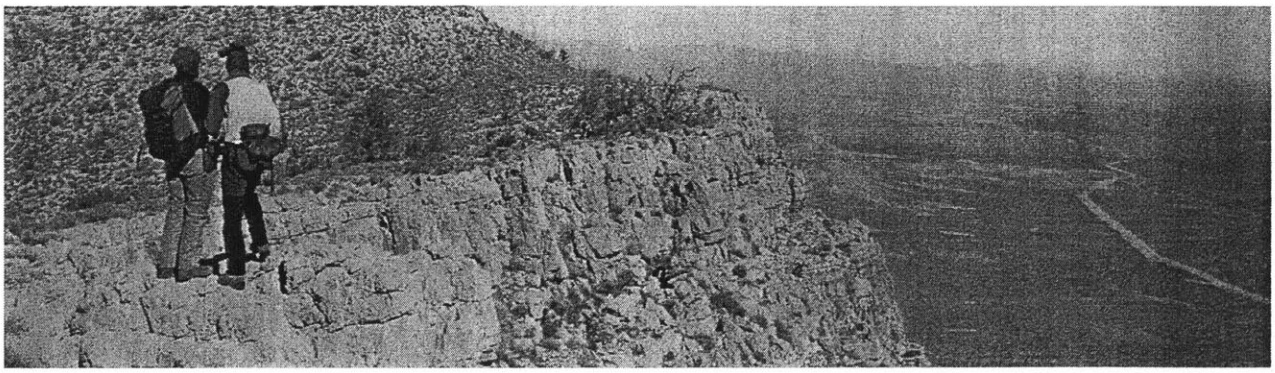
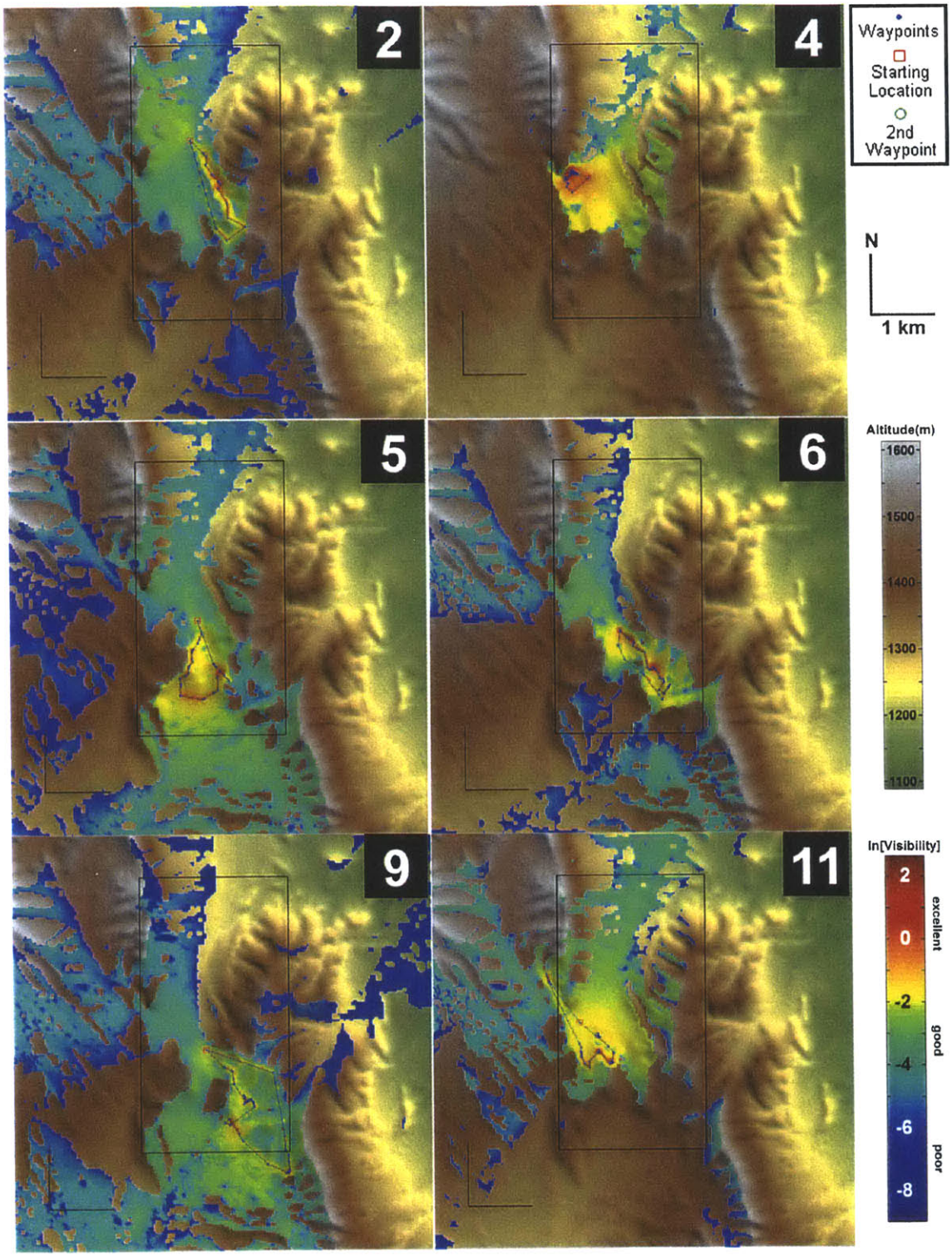


FIGURE 4-16. These cliffs in the North-West corner of mapping area A represent some of the steepest slopes in the Bird Spring Mountains. The slope of the cliffs shown here is approximately 75 degrees. The highest slope as computed from the field area digital elevation model was 54 degrees.

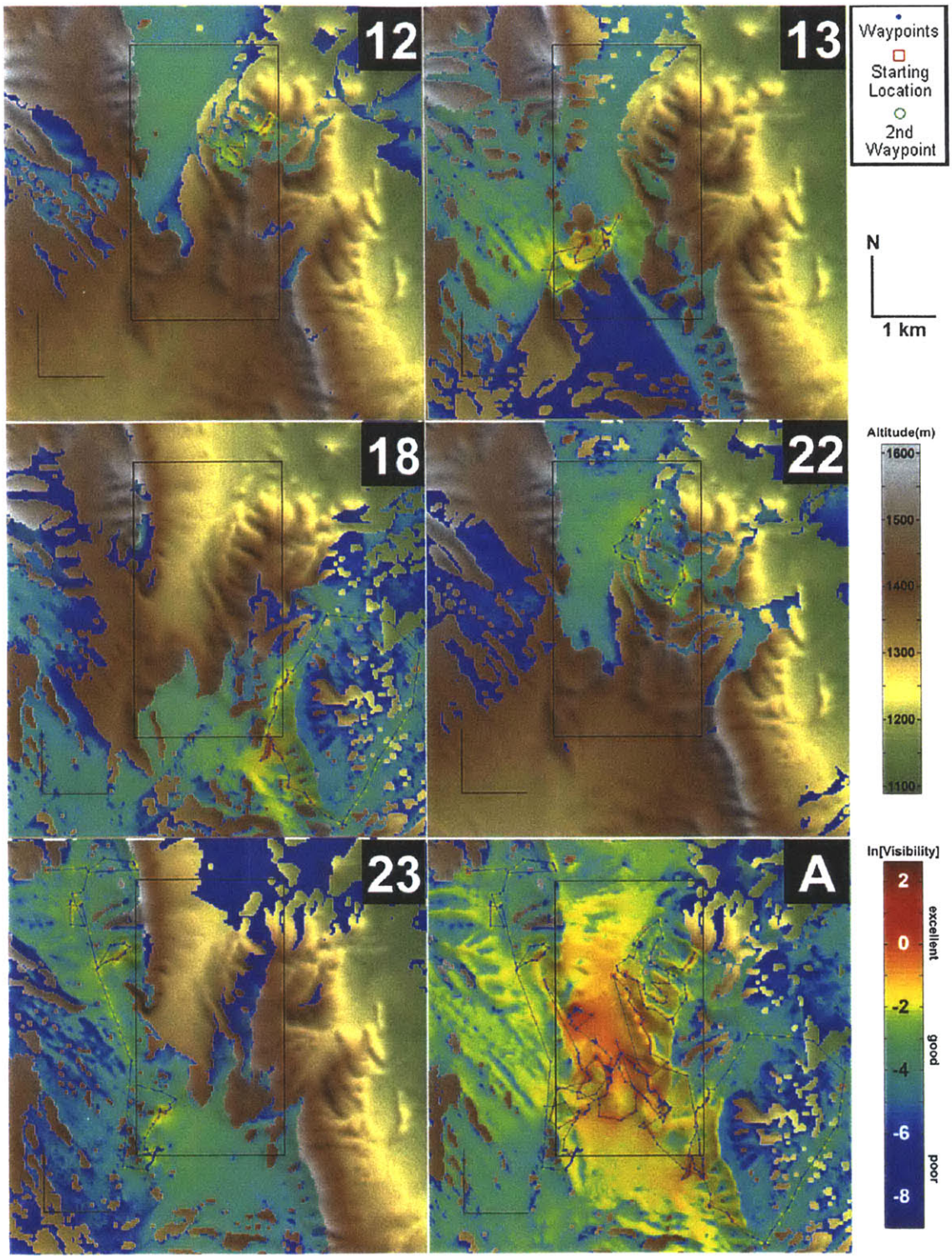
One should keep in mind that it may be possible to find low slope paths in areas where the algorithm computes high slopes, because the slope algorithm estimates a maximum slope for each cell.

Next, a visibility analysis of the traverses was performed to assess what parts of the terrain were visible and how well they could be seen. Figure 4-17 and Figure 4-18 show the result of this analysis.



Visibility of the surface of the field area is computed using the coverage algorithm described in Chapter 3, Section 3.3. Nearby (or frequently visible) surfaces are assigned higher visibility scores.

FIGURE 4-17. Inverse-distance based visibility analysis for traverses 2, 4, 5, 6, 9, and 11. Areas not visible were not colored.



The combined visibility plot (Figure 4-18, A) demonstrates that good visibility over most of the mapping area (Figure 4-18, rectangle) was obtained. The few areas of the mapping area for which the

FIGURE 4-18. Visibility analysis for traverses 12, 13, 18, 22, and 23. (A) shows a combined analysis of all the traverses.

traverses did not provide good visibility coverage (primarily in the north-east corner and east boundary of mapping area A) were covered extensively by the other mapping groups, and by other members of the author's mapping team. It is important to recall that the traverses represents only the author's traverses; occasionally, the mapping group did split up to cover larger areas more efficiently.

While visibility modeling provides valuable information about what areas were visible for each traverse, another important consideration must be what it costs to execute a given traverse, in terms of the energy costs. Clearly, in the case of field geology, the energy costs related to traveling between different locations is only a fraction of the energy used: many activities at a given waypoint or between waypoints may require significant expenditures of energy (hammering rocks, digging or other sampling activities, setting up or breaking down equipment). However, to first order, the mean velocity of travel between waypoints is still an important metric for future extravehicular system design, and is strongly related to energy consumption. Mean velocities between waypoints for the eleven traverses are shown in Figure 4-19. It is important to consider that the period between waypoints was not spent simply moving from one point to the other at constant velocity, but often included discontinuous activities such as looking at a local outcrops, changing an item of clothing or drinking some water, or briefly conversing with another member of the mapping team.

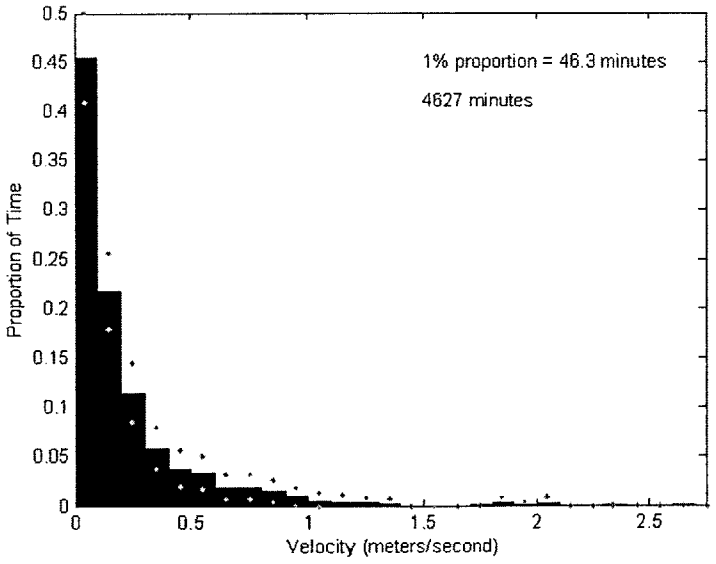


FIGURE 4-19. A histogram of the mean velocities between traverse waypoints shows that 88% of travel between waypoints was at a mean velocity of 0.5 m/s or slower (95% confidence boundaries for each proportion, assuming constant velocities between traverse waypoints, are indicated as green dots). These velocities are biases toward low values: the path between waypoints was approximated by a straight line, and in reality, some deviation from a straight line path is virtually guaranteed. In addition, velocities between waypoints were not constant, and brief additional activities (such as inspecting nearby outcrops, or briefly conversing with other mapping team members) were not uncommon between waypoints. Frequent and automatic collection of position data would eliminate these biases in the data.

The mean velocity between waypoints, and the mean slope between waypoints, can be used to compute a mean metabolic cost between waypoints using the load-carrying model developed by Santee et al. (2001). This load-carrying model (described in detail in Chapter 5) allows one to estimate the power and energy costs associated with

carrying a load (such as a geologist's tools and backpack) over mountainous terrain. The power and energy estimates are broken down into estimates of the power required to walk on level terrain (*internal power*) and the power required to climb or descend (*vertical power*). The total power estimate is a function of intrinsic factors such as muscle efficiency and body weight, and such extrinsic factors as the additional load carried (clothing and equipment), the velocity of travel, the slope of the terrain, and the rate of ascent or descent. Figure 4-20 shows mean metabolic power between adjacent waypoints for the eleven traverses.

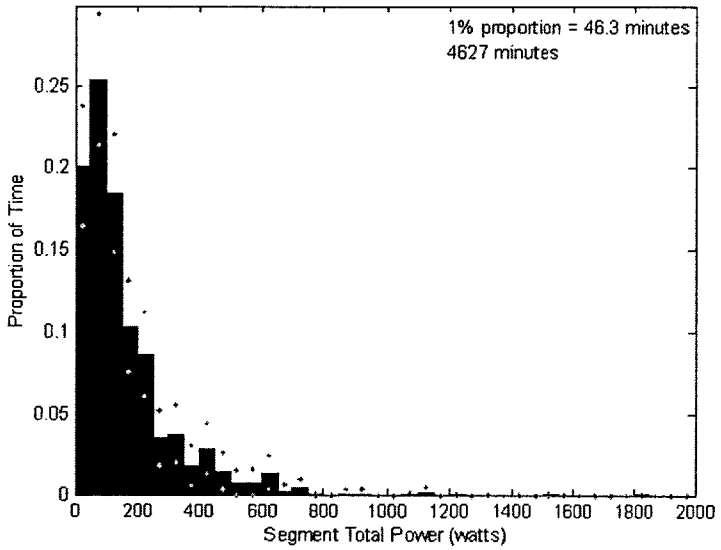


FIGURE 4-20. A histogram of mean metabolic power between traverse waypoints was computed, based on the author's mass of approximately 80 kg, and an assumed additional 13 kg for clothes, backpack, and equipment (daily weight of clothes and equipment was not measured). Green dots represent the 95% confidence boundaries for each proportion estimate. The mean metabolic power during the traverses was approximately 210 Watts, based on the load-carrying model of Santee et al. (2001). The few segments with extremely high mean metabolic power are likely due to extremely closely spaced waypoints and are potentially due to position errors or timing errors (waypoint times were only recorded to within one minute, so large errors might be expected for very short segments).

Figure 4-21 shows the total energy requirements for each traverse.

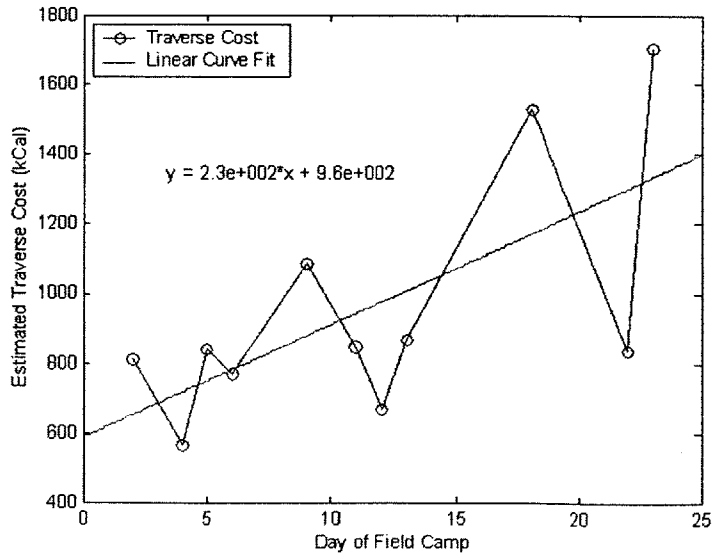


FIGURE 4-21. Estimated total energy requirements for each traverse can be computed by integrating the estimated metabolic power for each traverse. The data suggests a general trend toward increasing energy expenditures for traverses (as is expected, based on the generally increasing traverse distances).

Total energy per traverse as computed and plotted in Figure 4-20 and Figure 4-21 is clearly only one segment of energy consumption during geologic mapping: the above estimates only relate to moving between geologic sites of interest, and do not take into account the energy expenditure of any of the activities at waypoints or other activities between waypoints. Direct monitoring of metabolic power and continuous monitoring of position during specific types of field exploration would enhance the accuracy and meaningfulness of metabolic estimates and verify the model.

These simplistic metabolic power estimates are not directly applicable to planetary surface exploration in any case because of the significant changes in metabolic cost during locomotion in altered gravitational environments [Newman, 1992]. Conceptually, however, these estimates can be useful. Figure 4-22 plots the mean metabolic power for each segment (period between two adjacent waypoints in a traverse) versus the length of the segment, while Figure 4-23 illustrates how these results might suggest an approach to bound the metabolic power design envelope for future space suits for planetary surface exploration.

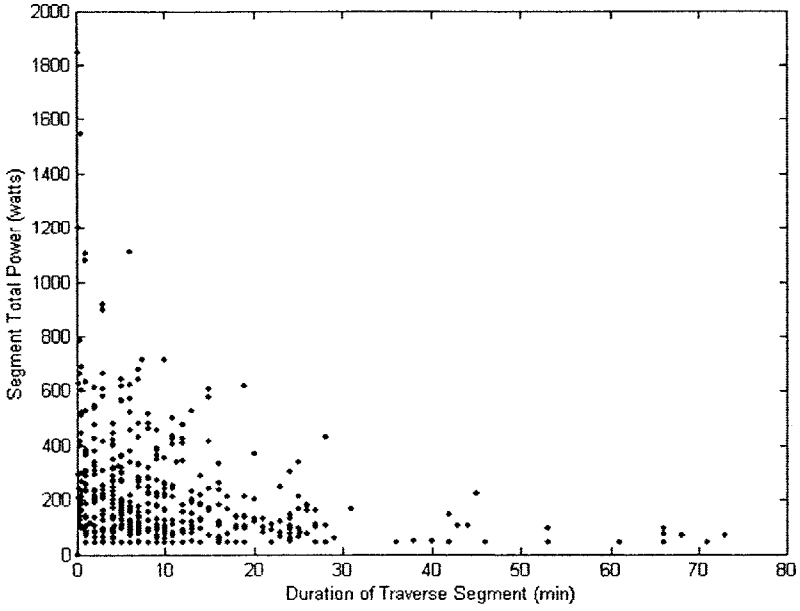


FIGURE 4-22. Estimated mean metabolic power between adjacent waypoints for all eleven traverses.

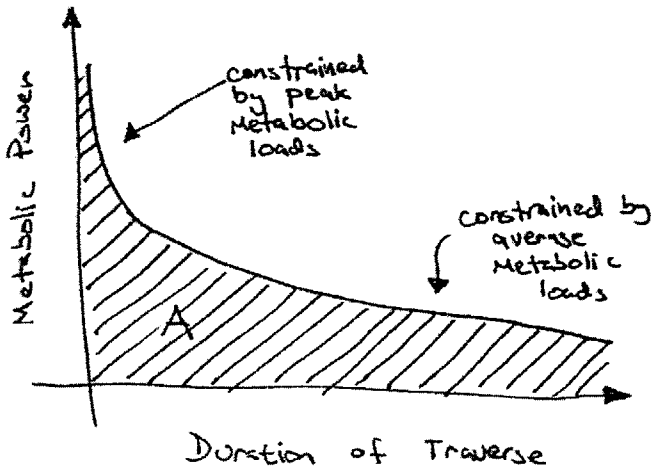


FIGURE 4-23. Conceptual illustration of a metabolic power design envelope for future space suit design. Future suits should be capable of operating over most (ideally all) of region A. During short periods of time, metabolic power limits (e.g., metabolic rates that can be safely imposed on the suit) can be constrained by peak metabolic power outputs expected from human explorers performing various activities. For longer periods of time, metabolic power limits can be constrained by maximum expected average metabolic loads from human explorers performing appropriate activities (typically limited by total consumable resources such as power, CO_2 scrubbing capabilities, or O_2 supply).

In addition to metabolic cost, an important consideration for space suit operations (and robotic operations as well) will be the surface slope. Specifically, given a digital elevation model and some initial starting point, one can compute the accessible area of the terrain given some criterion. This accessible area is called an *accessibility map* or a *reachability map* and was previously described in detail in Chapter 3. Figure 4-24 illustrates a reachability map computed for the field area using the (rather restrictive) criterion of "operation allowed or possible on slopes between 0 and 5 degrees." While clearly this is an unrealistic constraint for field geology here on Earth, it is not an unrealistic criteria for some robotic vehicles or for a suited astronaut in an Apollo-style suit (future planetary exploration will require significantly less stringent constraints).

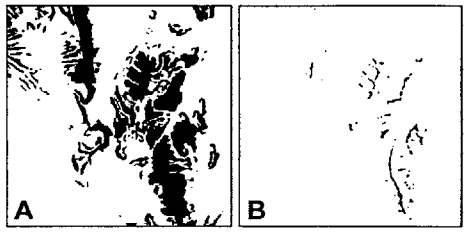
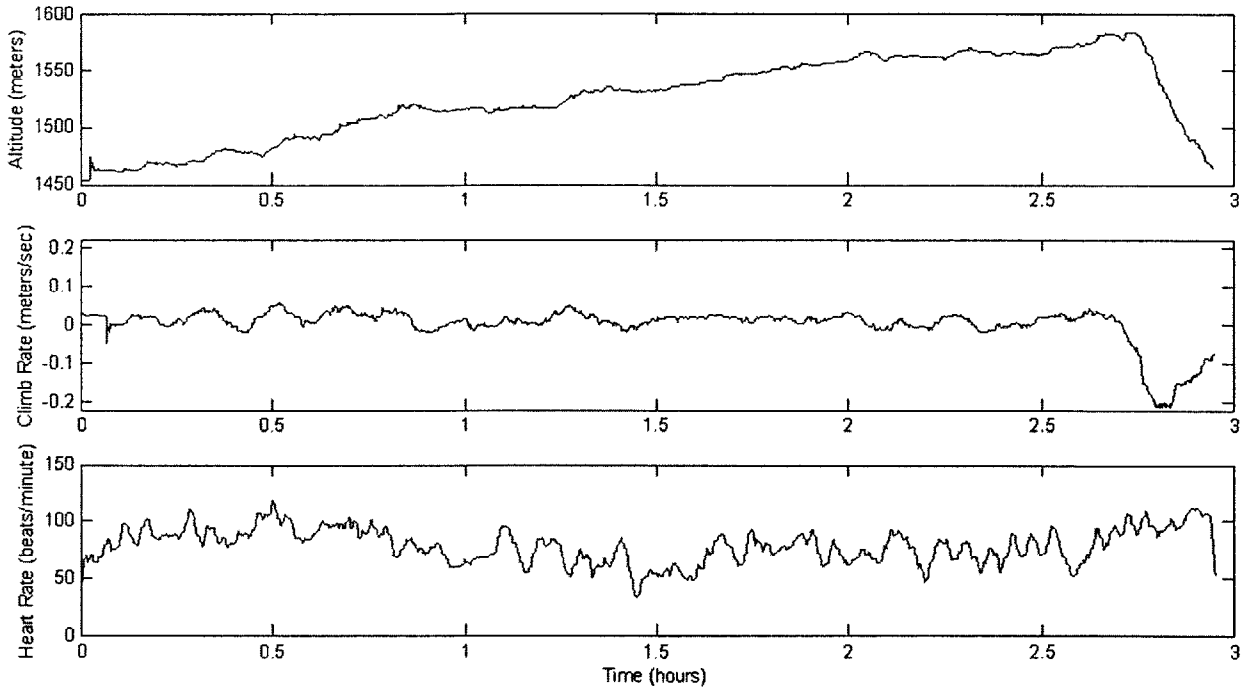


FIGURE 4-24. A reachability map (A) for the field area has been computed using a slope restriction of slopes less than or equal to five degrees, with an initial location given by a point in the central valley of the field area (the first point of the day 2 traverse was used). Accessible areas are shown in white, and inaccessible areas are shown in black. In (B), areas are colored black if the local slopes were less than five degrees, but the areas were inaccessible. All slopes were computed at a spatial resolution of 30 meters.

Combined Heart Rate and Position Analysis. It is unfortunate that, due to the sensor failure, only one combined position and pulse-oximeter data collection session was recorded (in addition to the initial half-hour system test). However, something can be learned from the minimal data that was collected. Figure 4-25 shows the combined data set obtained during the day 10 traverse.



While heart rate can be used as a metabolic index over long time scales, heart rate is a poor indicator of metabolic power on short time scales, and no strong correlation between climb rate and heart rate seems to be present here (some correlation, at least superficial in nature, is evident between the climb rate and heart rate curves here). While one would expect a significant correlation between

FIGURE 4-25. Altitude (top), climb rate (middle), and heart rate (bottom) are plotted as a function of time during a simultaneous recording of position and heart rate during the day 10 traverse. Climb rate and heart rate have been filtered using a five-minute boxcar filter.

heart rate and climb rate (for long time scales) in tightly controlled laboratory conditions such as those described in [Santee et al., 2001]. Less significant correlations might be expected during actual field geology because of significant "noise" sources such as the many field geology activities that the group pursued during the day 10 traverse. Accurate assessments of metabolic cost during field geology activities will require both an accounting of traverse geometry and analysis of the metabolic cost of other planned activities.

Digital Photography. Captions were written for each digital photograph, and a searchable index of photographs was created. The time intervals between photographs were analyzed. The time each image was taken was obtained either from a written record of the photograph time from the author's field notebook, or from the time recorded by the digital camera. Figure 4-26 demonstrates one of the characteristics of photographic activity in the field: given that one or more photographs had recently been taken, it was likely that the author would either take additional photographs, or wait a long time to take additional photographs.

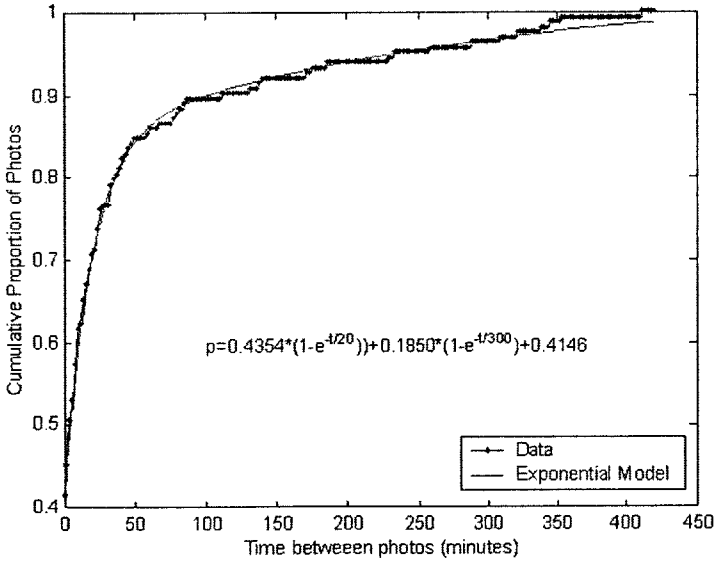


FIGURE 4-26. The time intervals between photographs were analyzed, and the cumulative proportion function was estimated as a sum of two exponential functions with time constants of 20 minutes and 300 minutes. The cumulative proportion represents the proportion of photographs for which the time interval between photographs was shorter than or equal to a specified time interval (plus or minus up to two minutes, since photograph time was only recorded to a resolution of minutes).

Qualitative Observations on the Senses. Sight was clearly the most important sense used during field geology. The use of sunglasses was found to affect the relative contrast of rock units and change the observed colors of the rock units. The time of observation was also important: low sun angles made observations challenging due to shadows or red-shift. Obtaining visual observations of some geologic feature from many different perspectives was an important activity. Small valleys were often crossed by one or more members of the mapping team to obtain a quick change in visual perspective. Similar activities during future planetary surface exploration may require extravehicular activity systems with signif-

icant mobility. In addition to sight, touch was often used during geologic mapping to examine outcrop surface characteristics such as graininess and texture. Density might also be evaluated by handling of outcrop materials. Clearly, using a geology hammer is a strongly visual and tactile activity. During the future, some of these activities might be supplanted or supplemented by devices capable of higher accuracy quantitative measurements. However, this may affect the ability of future explorers to assimilate information and to generalize about their environment.

Qualitative Observations on Group Communication. The mapping team frequently practiced what might be called *distributed geology*: Multiple team members communicated across small distances (tens of feet to hundreds of feet) about geologic features from concurrent but different viewpoints. In addition to voice communications, gestures or props, such as a map-board, were used to describe features and their orientations. Quantitative measurements, such as bedding-plane or fault-plane orientations, were often taken in a distributed fashion. Frequently, several members of the mapping team would take an orientation measurement from different points along an outcrop or on nearby outcrops, and the results were shared among the group in order to select a representative measurement for the local area. Distributed geology was complicated by difficulties in communication when team members were either far apart, or unable to see one another. Enhanced short-range communications would have further enabled distributed geology practices. Figure 4-27 conceptually illustrates group communication during geologic mapping, including distributed communications.

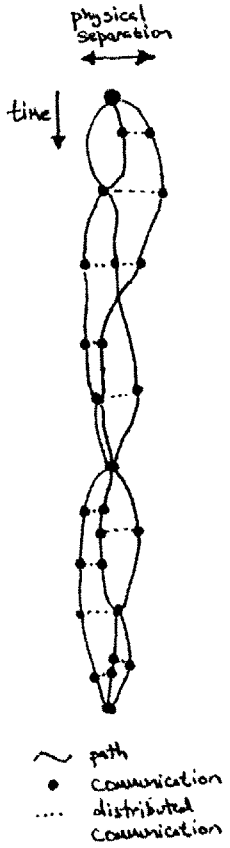


FIGURE 4-27. Group communication during geologic mapping is illustrated here in a conceptual, subjective sketch. As geologic mapping proceeded, different group members follow different physical paths, and occasionally group members physically met in the same location (*communication*) or communicated over distances of tens to hundreds of feet or more (*distributed communication*) via gestures, voice signals, or yelled instructions. Periodic physical meetings of the mapping group were essential for team coordination and group discussion. The above sketch might represent communications during a couple of hours of geologic mapping.

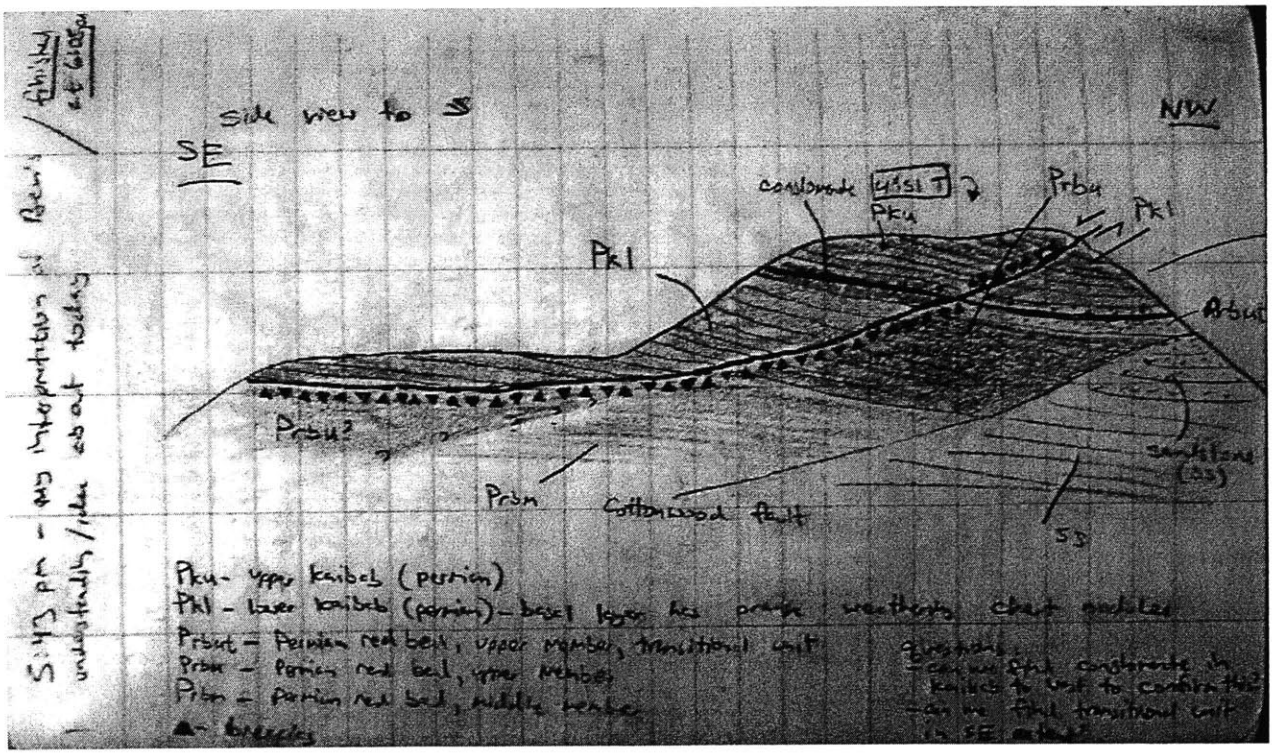
A frequent activity during group communications was the attempt by a group member to communicate a mental model to the rest of the group. This tended to be most successful in the field and in a location where group members could directly observe the area pertaining to the mental model. Drawing or annotating (on the topographic base map, for example) served as a primary means of communication of a mental model, along with gestures and an aural description of the model. In many cases, it took several minutes of discussion to ensure that all group members understood exactly what structure was under discussion. Because it was often difficult to see exactly where someone else was looking or pointing, a group member explaining a proposed structural feature might have to describe how to find the area relating to the proposed feature based on a series of unambiguous visual landmarks. What might have taken several minutes of explaining, could just as easily have been accomplished by taking an instant or digital photograph, and then annotating that photograph to quickly communicate the area of interest and the hypothesis or structural model being proposed.

In a couple cases, a digital photograph was taken and a group member pointed out a proposed geological structure on the digital cam-

era liquid crystal display. It was clear that this process suffered because of:

- inadequate visibility of the camera graphic display, especially in sunlight,
- inadequate size of the camera graphic display, and
- inability to annotate the digital photograph.

Some of the most productive group discussions took place back in camp in the evening after a day in the field. Figure 4-28 illustrates the results of one evening discussion: a hypothesis for potential geologic structures that would explain observations made earlier that day in the field.



Communication of data from geologic mapping to the other mapping groups and to others not in the field posed additional challenges. The high latency between the collection of data in the field and the interpretation of that data during the creation of a geologic map can reduce the value of collected data by decreasing the ability of the geologist to judge the quality of data and to put that data into a coherent framework (one might characterize this situation as coping with a degraded mental model). An ability to go back to the field, either physically or virtually (to replay the field experience) might enhance a degraded mental model, or allow others to build a coherent mental model. Standardization of data collection proce-

FIGURE 4-28. A sketch was made by the author one evening in camp during a group discussion of potential geologic structures observed in the field. In this case, the author's goal was to ensure that a model proposed by a team member had been accurately communicated by *sketching* the proposed structures and then *verifying* the sketch with the team members. The act of sketching the proposed geologic structures and discussing other possible explanations raised new questions (lower right) requiring additional investigation in the field.

dures in the field may improve the overall quality of a data set (through better internal consistency), but even without standardization of data collection, simple subjective assessments of the quality of data have the potential to improve the value of data collected in the field.

Qualitative Observations on Transportation. The drive between camp and the mapping area was used effectively as a daily planning and daily debrief session, including identification of daily goals, strategic approaches, hypotheses, and conclusions. The author's group frequently drove to similar starting locations for different traverses, and often pursued geologic observations from the car. Practicing *window geology* gave the group an opportunity to see other groups' mapping areas and to observe geologic features from different perspectives. The motion of the vehicle helped to provide a sense of scale of geologic features (probably due to optical flow).

4.1.5 Summary

The process of geologic mapping of a ~7 km by 7 km field area was analyzed from several perspectives including the temporal and spatial behavior of field geology activities by the author. Daily traverses evolved from simple to complex and trended towards longer traverses. Daily routines and common routes were utilized, and repeat visits were made to several sites of interest. Identification of geological sites of interest was an iterative process that continued throughout geologic mapping. Earlier traverses were more exploratory in nature, and later traverses were more focused on solving particular problems or "filling in holes."

A digital elevation model of the field area was used to compute approximate paths between waypoints, and slope analysis demonstrates that some field geology activities occurred on slopes as steep as 30 degrees, while most of the time field geology activities occurred on slopes of less than five degrees. Nearly all of the field area was accessible to locomotion except local steep areas and some larger cliffs. Visibility analysis of the traverses demonstrated what areas of the field area were easily visible or frequently visible, and demonstrated that good visibility of an area does not necessarily require physically visiting an area (the north-west corner of the mapping area, for example). Mean velocities between waypoints were found to be below 0.5 m/s nearly 90% of the time. A load-carrying model was used to estimate metabolic power during each traverse using only position and slope data along with an assumed weight for the author and his carried equipment. The estimated caloric cost per traverse ranged from less than 600 kcal to over 1600 kcal. Combined heart rate and position analysis failed to demonstrate a strong correlation between climb rate and heart rate, suggesting that field geology activities (as compared to purely traverse-related locomotion) are a significant factor in metabolic

power requirements, and that the metabolic power estimates using the load-carrying model represent only a fraction of the actual metabolic power requirements during geologic mapping.

Qualitative observations highlighted the importance of vision in the geologic mapping process, and the critical role played by group communication including distributed communication (communication while mapping team members are physically separated). Difficulties in communicating mental models were encountered, and sketching and group discussion were identified as the primary approaches to the communication of mental models. Vehicular transportation periods were also used effectively for group communication and coordination.

Distributed systems could improve the process of field geology by supporting access to a diversity of visual perspectives and enhancing communication between geologists. Specifically, tools that support distributed communications between group members in a given mapping group would substantially improve the ability of the group to coordinate individual activities and to pursue distributed geology.

4.2 Analyzing Apollo

Lunar surface operations during the Apollo missions are arguably the closest analogous experience to the extravehicular activity segment of future human Mars exploration, and can provide valuable insights that may benefit future exploration of the Martian surface.

Like the previous case study, results from this case study are not directly applicable to Mars surface exploration. Indeed, part of the goal of this case study is to highlight important differences in the experiences of the Apollo missions and the realities of future human exploration of the Martian surface.

4.2.1 Overview

This case study assesses the use of information and communication by the Apollo lunar surface astronauts, with an emphasis on voice communications. Metabolic data from lunar surface operations are analyzed to assess the relationship between the topography of the lunar surface and the metabolic power requirements of exploration.

Why study voice communications? Recall that message passing is the fundamental method of communication between agents in a multi-agent system (see Chapter 2). The study of voice communications during the Apollo missions is an approach to characterizing the message passing behavior of human explorers during planetary surface exploration.

Twelve astronauts walked on the surface of the moon during the Apollo lunar surface missions (Apollo 11, 12, 14, 15, 16, and 17). Accounts of lunar surface operations can be found in Chaikin (1994), and (from a geological and scientific perspective) in Wilhelms (1993) and Heiken et al. (1991). One of the best sources of

first-hand details of lunar surface exploration is clearly the *Apollo Lunar Surface Journal* [Jones, 2000], which contains not only raw transcripts of the voice communications during the Apollo missions, but extensive commentary from the lunar surface astronauts, mission checklists, maps, images, audio, and video from the Apollo missions. This case study relies heavily on the *Apollo Lunar Surface Journal.*

4.2.2 Methods

In order to evaluate the role of voice communications during the Apollo missions, voice communication transcripts were downloaded from the Apollo Lunar Surface Journal website, and a database of voice communications during the Apollo missions was created. Figure 4-29 illustrates the process used to create a database of voice communications for the lunar surface exploration portions of the Apollo missions. Transcripts from the website were parsed by Visual Basic code into a Microsoft Access database. During parsing, the context of each transcript entry was determined, including the mission elapsed time of the transcript entry, the speaker, and the speaker's mission role (such as Commander or Lunar Module Pilot). Transcript entries included voice communication records and post-mission comments from the Apollo astronauts and other editors of the transcripts.

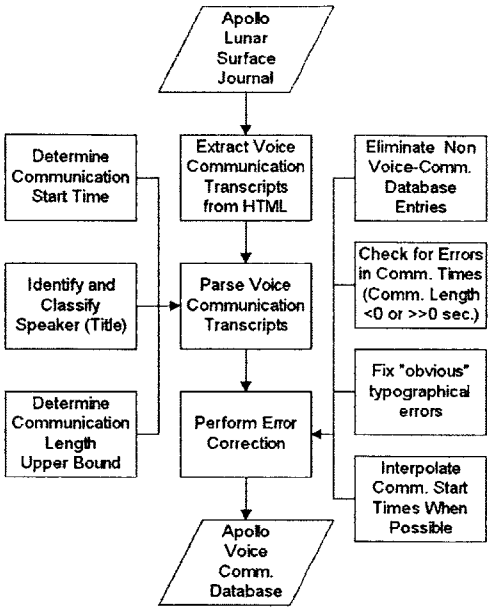


FIGURE 4-29. A database of voice communications during the lunar surface portions of the Apollo missions was created from [Jones, 2000].

Next, the database of voice communications was queried, and query results were imported into MATLAB for analysis (Figure 4-30). Analysis of the voice communications included:

- Analysis of cumulative voice communications initiations as a function of mission elapsed time and mission role,
- Analysis of the time interval between voice communications as a function of mission role (or equivalently, frequency of voice communications of an individual),
- Computation of an upper bound for voice communication durations, and subsequent analysis of communication durations,
- Comparison of voice communications during extravehicular activity versus non-extravehicular activity,
- Computation of an overall communications rate for each mission phase and entire mission, and
- Statistical summaries of voice communications for each mission, and comparisons between missions.

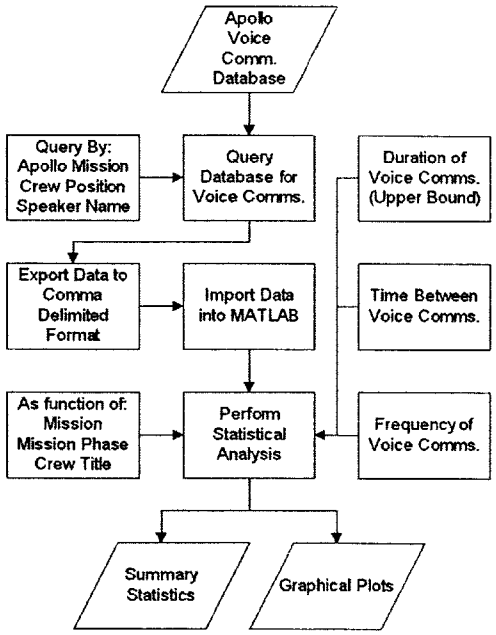


FIGURE 4-30. The voice communications database was queried, data was exported, and data analyses was performed using the software tool MATLAB.

The raw voice communication transcripts (and supporting literature) were examined to find cogent examples of the use of information and communication during lunar surface operations, and traverse distances and metabolic cost data from Johnston et al. (1975) was used to examine the cost of transport as a function of slope for the two Apollo 14 lunar surface astronauts (Apollo 14

astronauts conducted one of the most extensive walking traverses in all of Program Apollo; after Apollo 14, the Lunar Roving Vehicle provided an alternative means of transport). Difficulties in navigation during the second extravehicular activity of Apollo 14 are briefly considered, and a visibility analysis is used to illustrate how a different planning process might have anticipated some of the difficulties with the planned traverse.

4.2.3 Results and Analysis

Detailed results from the voice communications study are available in Appendix B. Key results are summarized here.

Voice Communication Database. Of the 58340 voice communications records processed from the lunar surface portion of the Apollo missions, 58096 were identified as valid, and 58174 records were analyzed after making corrections to 78 voice communication records. Corrections included identifying and fixing "obvious" typographical errors and interpolating approximate voice communication times where voice communication initiation times were well constrained. Table 4-4 summarizes the contents of the voice communication database.

TABLE 4-4. Voice Communications Database Records

Mission Role	Analyzed	Not Analyzed	Corrected	Total	% of Total
Commander	22510	32	36	22542	38.64%
Lunar Module Pilot	21869	46	25	21915	37.56%
CapCom*	10185	0	11	10185	17.46%
Command Module Pilot	545	0	2	545	0.93%
CapCom1	1648	31	0	1679	2.88%
CapCom2	911	40	0	951	1.63%
Others	504	17	3	521	0.89%
Total	58174	166	78	58340	100.00%
% of Total	99.72%	0.28%	0.13%	100.00%	

***CapCom = Mission Control radio communications operator**

Voice Communication Initiations. From Table 4-4 it is clear that the proportion of voice communication initiations (defined by the mission elapsed time at the start of a voice communication, as recorded by Mission Control) for the Commander and Lunar Module Pilot mission roles dwarf the proportion of voice communication initiations for other mission roles. Figure 4-31 shows voice communication initiations for the Apollo 17 lunar surface mission for different mission roles as a function of mission elapsed time.

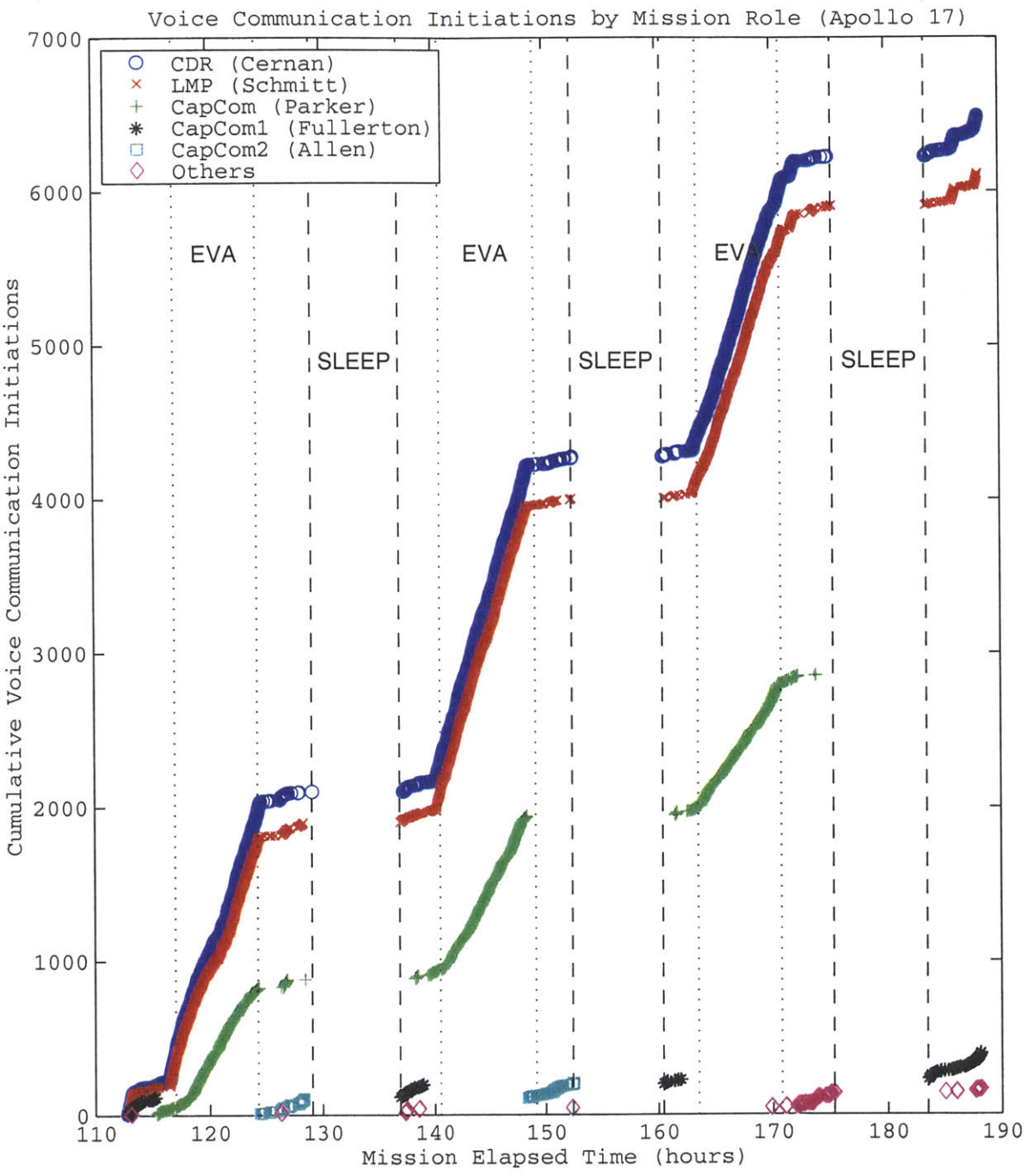
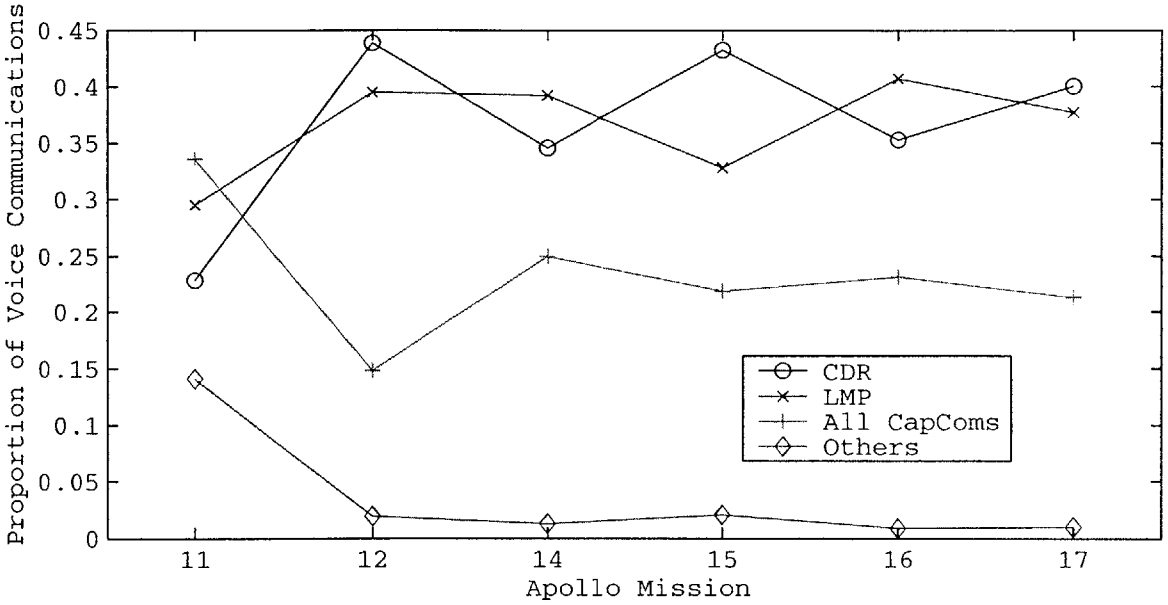


Figure 4-31 demonstrates a pattern followed during all of the Apollo lunar surface missions: Both the rate and number of voice communication initiations are higher for the Commander and Lunar Module Pilot than for other mission roles, and sharp increases in voice communication initiations clearly correspond to the start of periods of extravehicular activity.

FIGURE 4-31. Voice communication initiations for Apollo 17 lunar surface exploration by mission role as a function of mission elapsed time. Sleep and extravehicular activity (EVA) periods are denoted by vertical lines (Commander = CDR, Lunar Module Pilot = LMP).

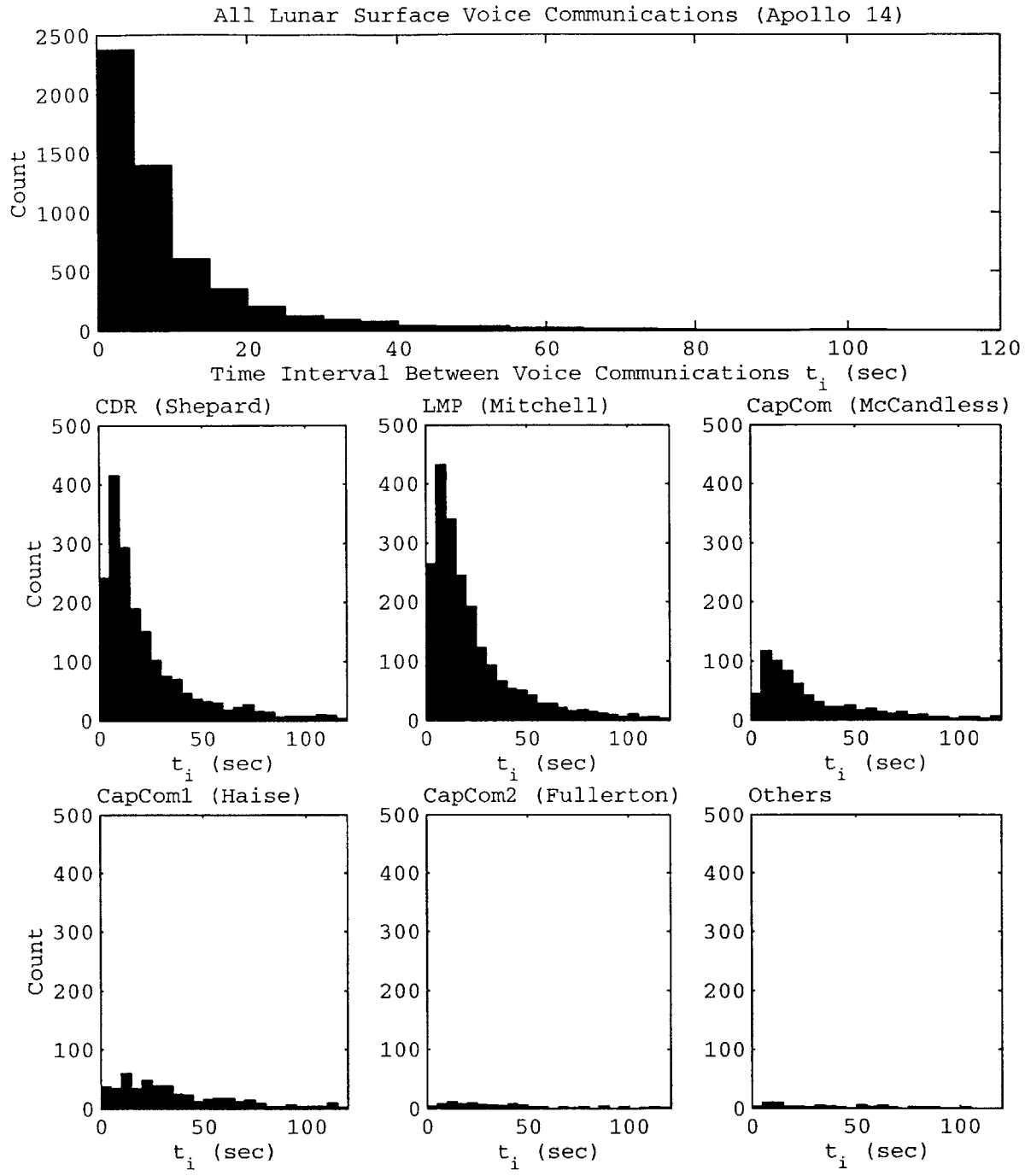
Proportion of Voice Communications. Proportion of voice communications by individuals not in the mission roles of Commander, Lunar Module Pilot, or primary CapCom decreased dramatically after Apollo 11 and continued to decrease as a function of mission number (Figure 4-32).



During Apollo 11, the lunar surface crew and Command Module Pilot (in the orbiting command module) often shared the same voice loop. This was largely eliminated in later missions, as evidenced by the significant reduction in the proportion of Command Module Pilot communications in the analyzed set of voice communications. In addition, CapCom workload may have been reduced (and lunar surface crew workload increased) as the goals of later missions focused more on exploring the lunar surface without the publicity pressures of the first human lunar landing. These proportions probably also reflect an adaptation process related to maximizing the value of lunar exploration: voice communications on this particular communications loop were focused on supporting the lunar surface astronauts and communicating information about the lunar surface.

FIGURE 4-32. Proportion of voice communications as a function of mission role for all of the Apollo lunar surface missions. By the later missions, the Commander (CDR) and Lunar Module Pilot (LMP) were responsible for nearly 80% of voice communication initiations.

Time Intervals Between Voice Communications. Histogram analysis of time intervals between voice communications for a given mission role demonstrate that individuals have many more short time intervals between voice communications than long time intervals between communications (a high proportion of voice communications for a given mission role have short time intervals between communications). This effect is especially pronounced for the Command and Lunar Module Pilot, as shown in Figure 4-33.



Time intervals between voice communications tended to be shorter during later Apollo missions as compared to earlier missions (Figure 4-34). This might be caused by a need for the Lunar Module crew to fit in more shorter communications to cope with higher workloads imposed by additional science and exploration goals.

FIGURE 4-33. Time intervals between voice communications for Apollo 14 lunar surface exploration by mission role. Bin size is 5 seconds. The few time intervals larger than two minutes are excluded from the plots, but not in other analyses (Commander = CDR, Lunar Module Pilot = LMP).

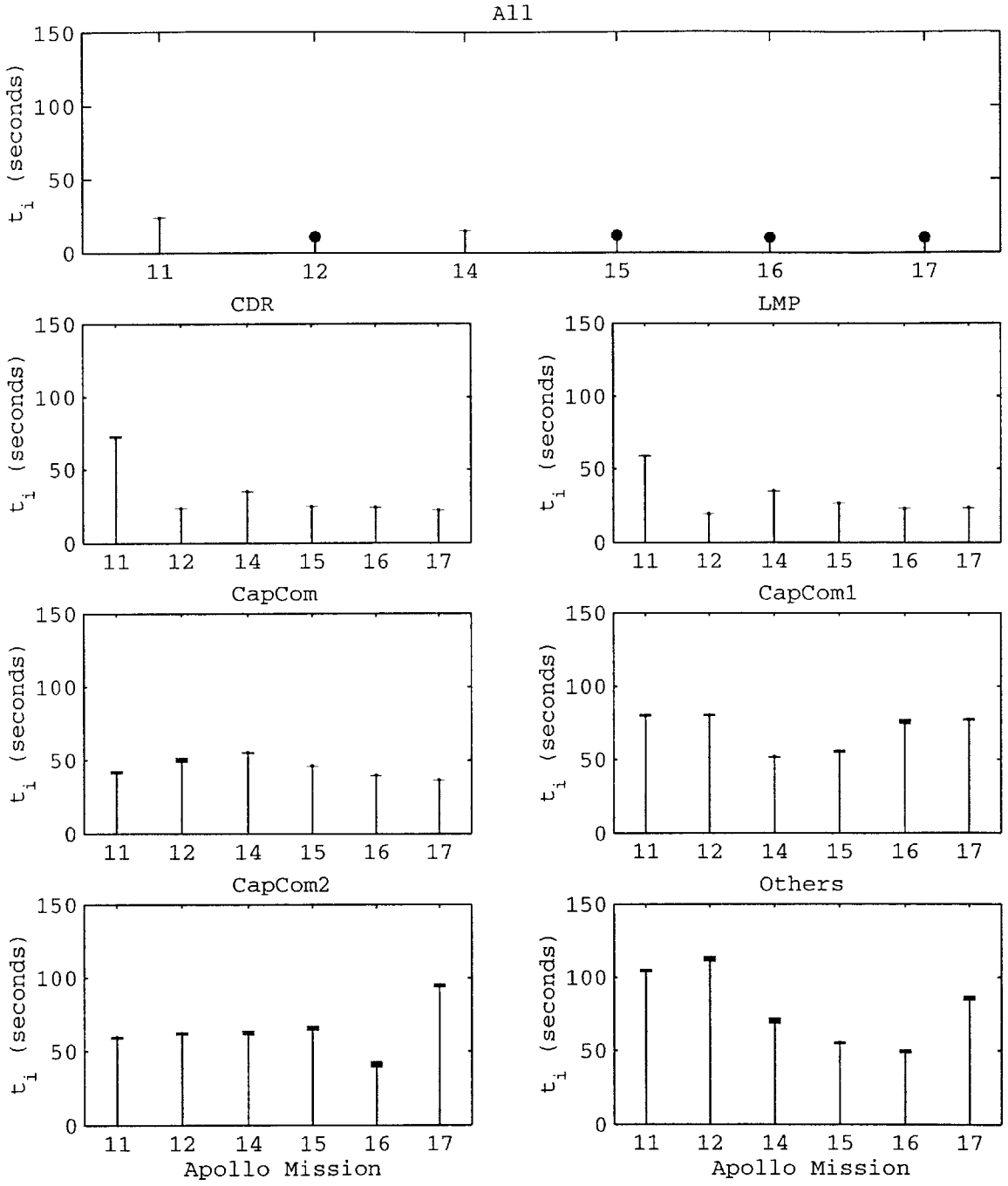
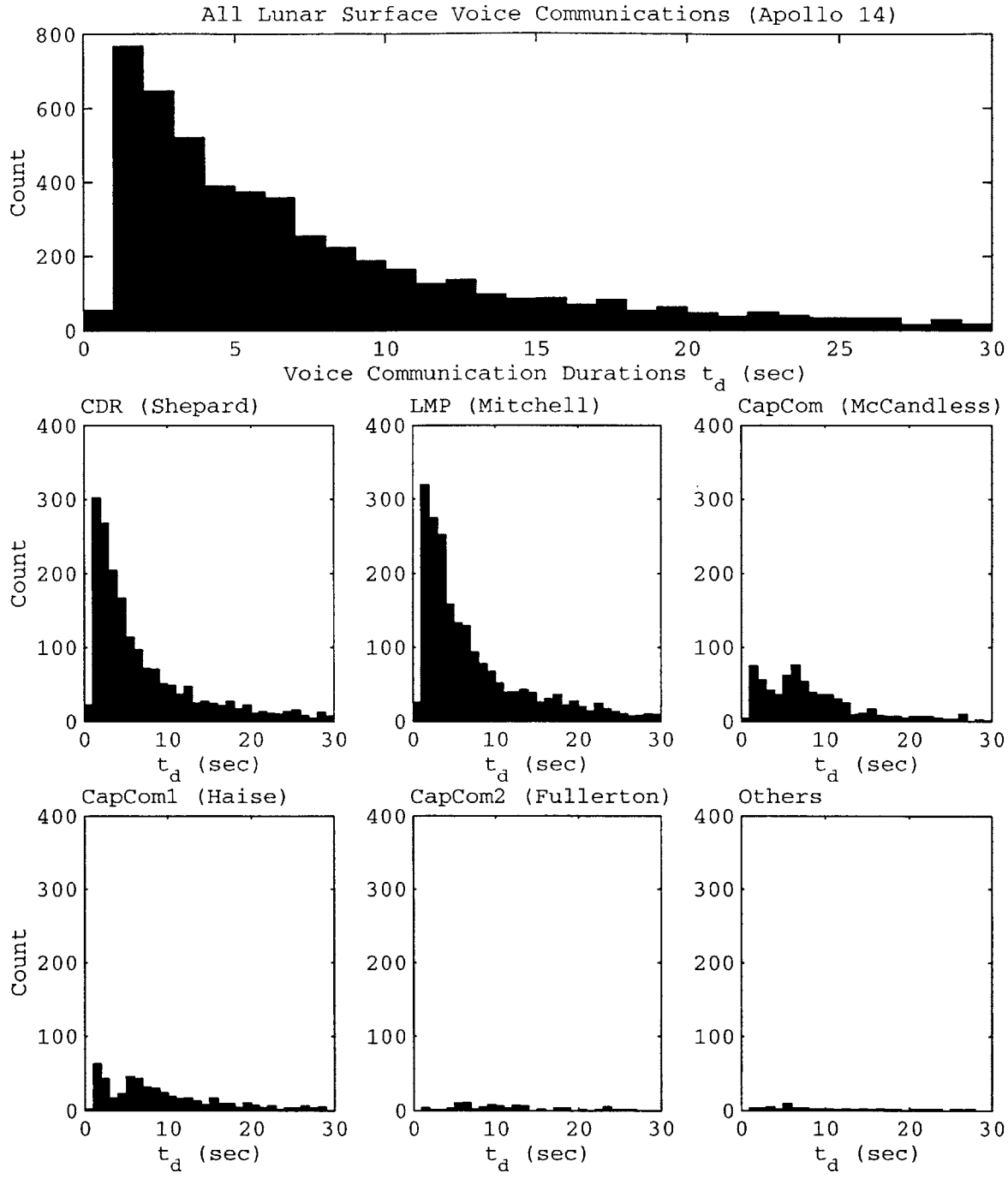


FIGURE 4-34. Mean time intervals between voice communications for different mission roles. Bar heights correspond to a 95% confidence interval as estimated by treating voice communications "arrivals" for each individual as a Poisson process (Commander = CDR, Lunar Module Pilot = LMP).

Because of the skewed nature of the time interval distributions in Figure 4-33, mean time intervals are significantly longer than median time intervals. Appendix B provides additional statistical characterization of time intervals between voice communications including 5th, 25th, 75th, and 95th percentile time intervals.



Duration of Voice Communications. As shown in Figure 4-35, voice communication duration distributions are similar in form to the time intervals between voice communications, and the same skewing effect is present. Voice communication durations also tend to be shorter during later Apollo missions as compared to earlier missions (Figure 4-36).

FIGURE 4-35. Voice communications durations (upper bounds) for Apollo 14 lunar surface exploration by mission role. Bin sizes are 1 second (accuracy limit from voice transcripts). Voice communication durations greater than 30 seconds are excluded in the plot, but not in other analysis (CDR = Commander, LMP = Lunar Module Pilot).

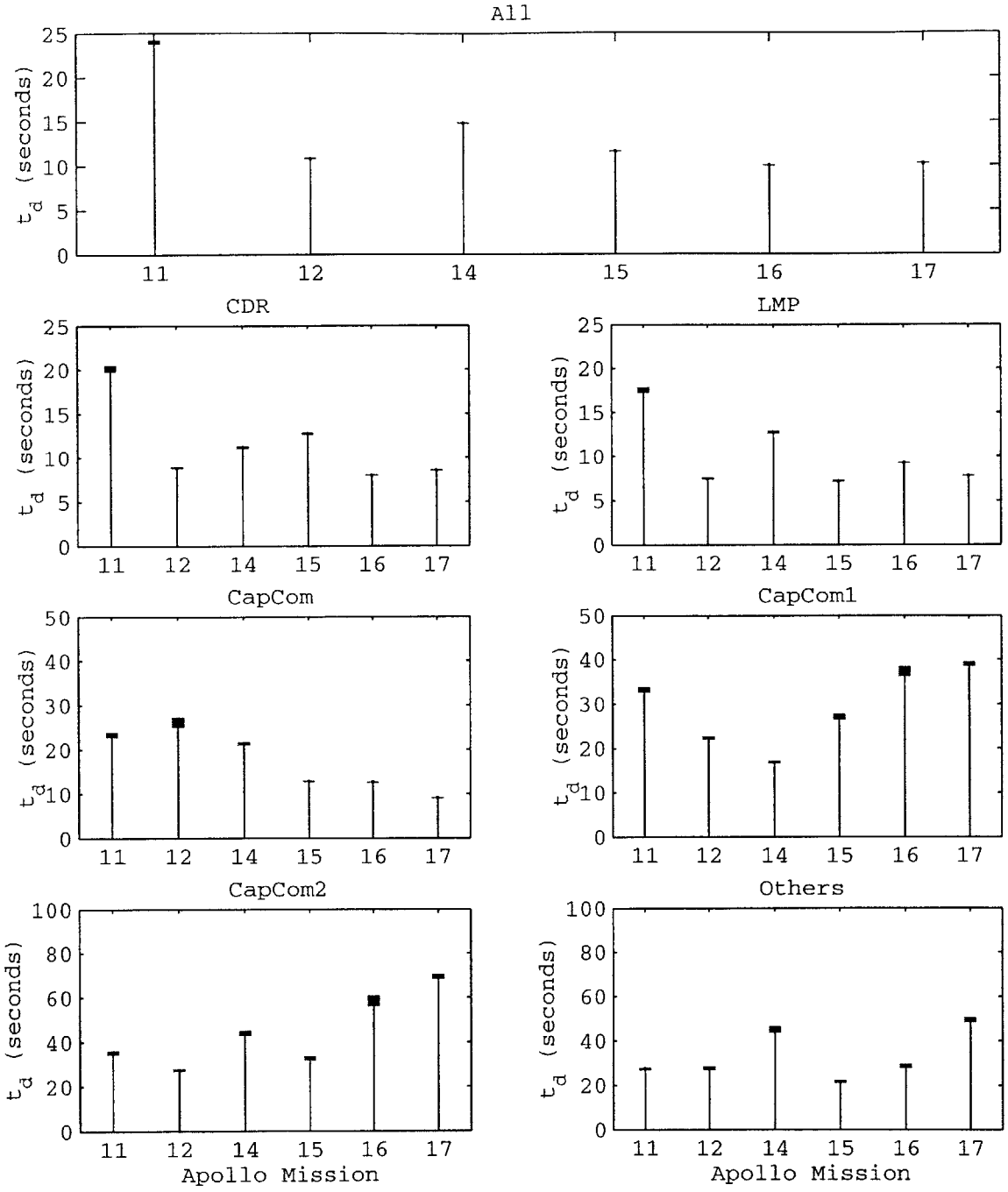


FIGURE 4-36. Mean voice communication durations for different mission roles for all Apollo lunar surface missions. Bar heights correspond to a 95% confidence interval as estimated by treating voice communications "arrivals" as a Poisson process (CDR = Commander, LMP = Lunar Module Pilot).

Shorter voice communication durations for the predominant communicators (Commander, Lunar Module Pilot, and CapCom) during later Apollo missions may reflect a need for and an approach to achieving a higher efficiency of information transfer during voice communications due to higher workloads imposed by additional science and exploration goals.

Mean Voice Communications Rates During Different Mission Phases and Activities. Mean voice communication rates for each Apollo mission are shown in Figure 4-37. As previously indicated (Figure 4-31) the mean rate of voice communication initiations during extravehicular activity is significantly higher than during non-extravehicular activity.

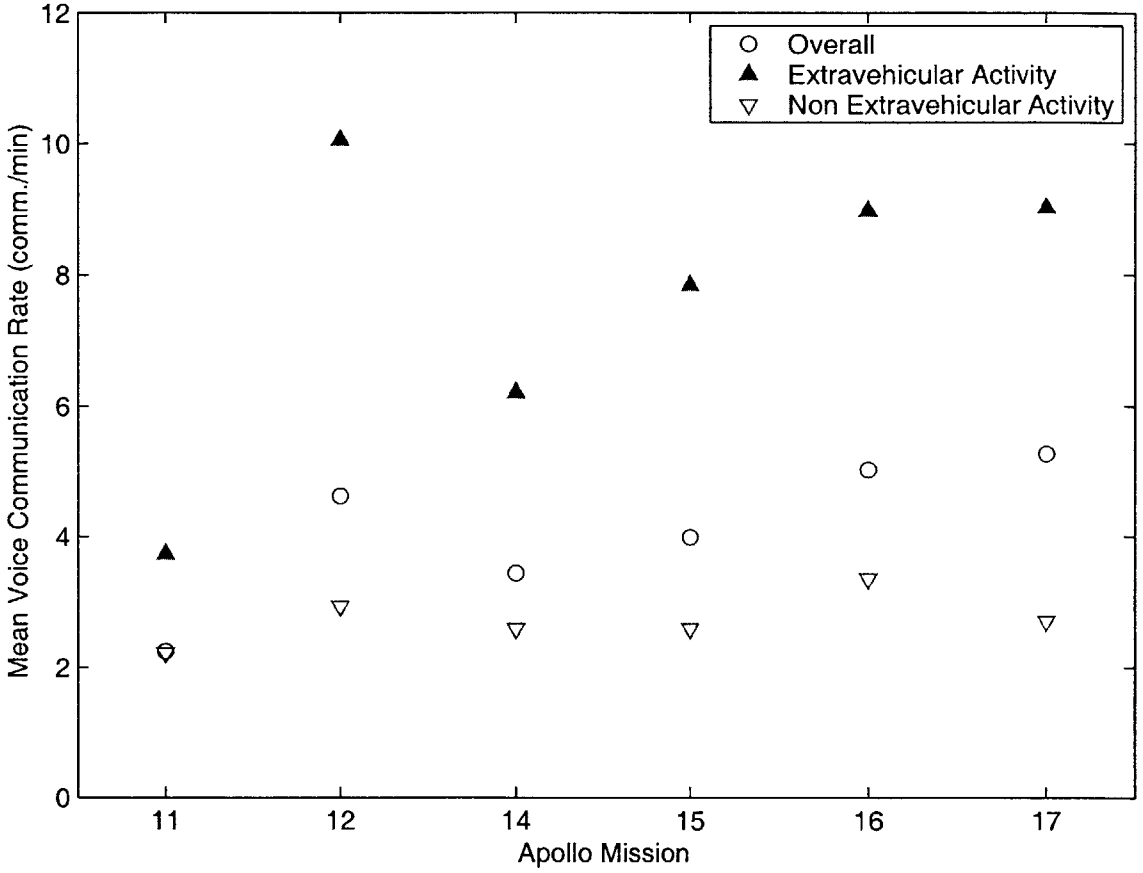
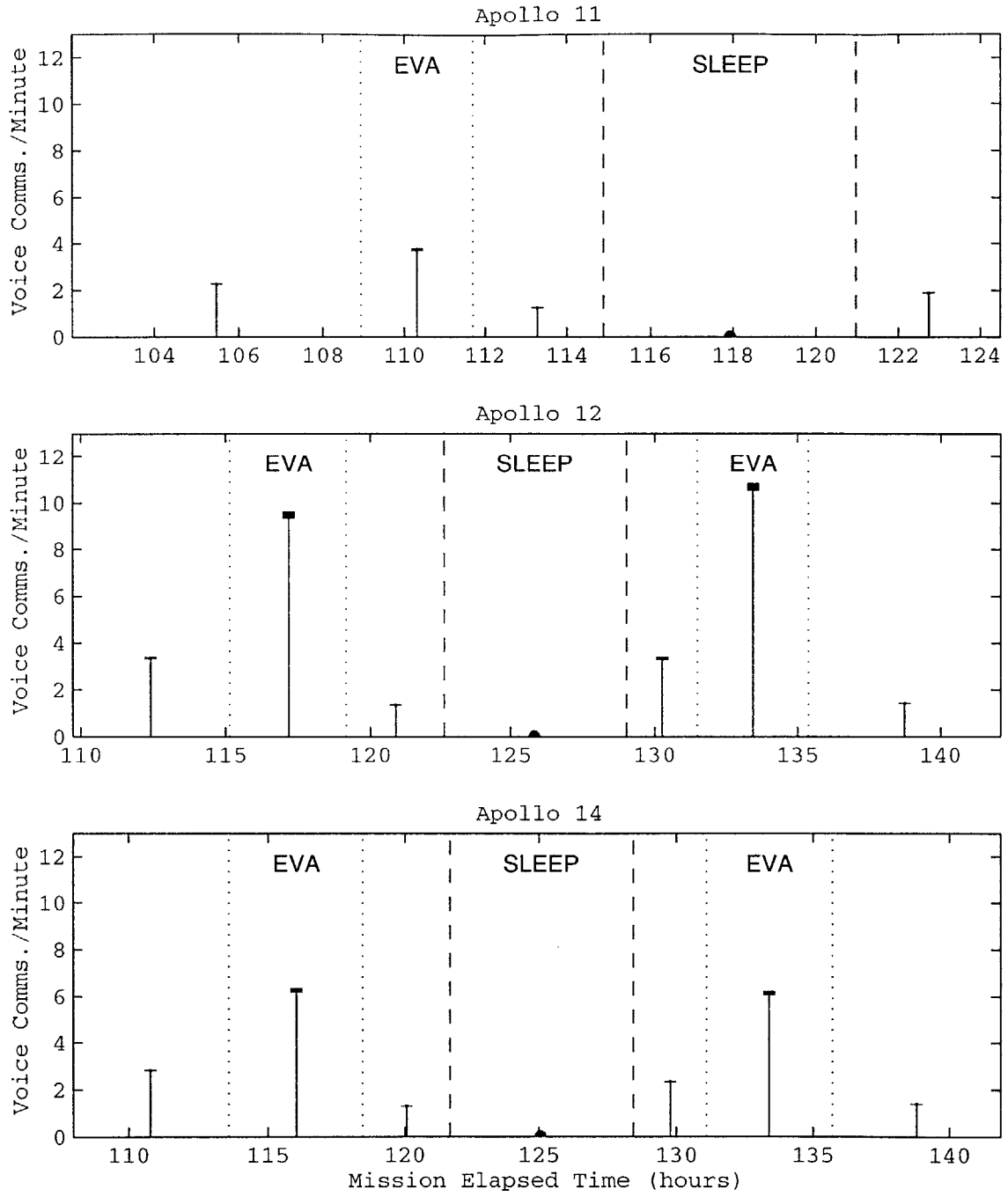


FIGURE 4-37. Mean voice communication rates during the Apollo lunar surface missions, including overall rate and rates during extravehicular activity and non-extravehicular activity. The extravehicular activity mean voice communication rate during Apollo 12 is a bit of an anomaly, and may be explained by the short duration of voice communications by Commander Pete Conrad and Lunar Module Pilot Alan Bean (see Appendix B).

The increase in the overall mean rate of voice communication initiations (in communications/minute) as a function of the Apollo mission number is largely due to increases in the voice communication rate during extravehicular activity. The non-extravehicular activity mean communication rate varies, but fails to indicate a clear trend.

Also illustrative is the mean communication rate during missions phase as a function of mission elapsed time. Figure 4-38 and Figure 4-39 segment the results of Figure 4-37 into individual periods of extravehicular activity and non-extravehicular activity and demonstrate the consistency of mean voice communication rates for different periods of the same type within the same mission.



During Apollo 12 and Apollo 14, mean voice communication rates during non-extravehicular activity periods fluctuated more than during extravehicular activity periods. This same pattern seems to continue for Apollo 15, 16, and 17 (Figure 4-39). Non-extravehicular activity periods had larger mean voice communication rates than other non-extravehicular activity periods if they were post-landing periods or proceeded a period of extravehicular activity.

FIGURE 4-38. Mean communication rates during extravehicular activity and non-extravehicular activity during Apollo 11, 12, and 14 as a function of Mission Elapsed Time. Bar heights indicate 95% confidence intervals, computed by treating voice communication "arrivals" as a Poisson process (EVA = extravehicular activity).

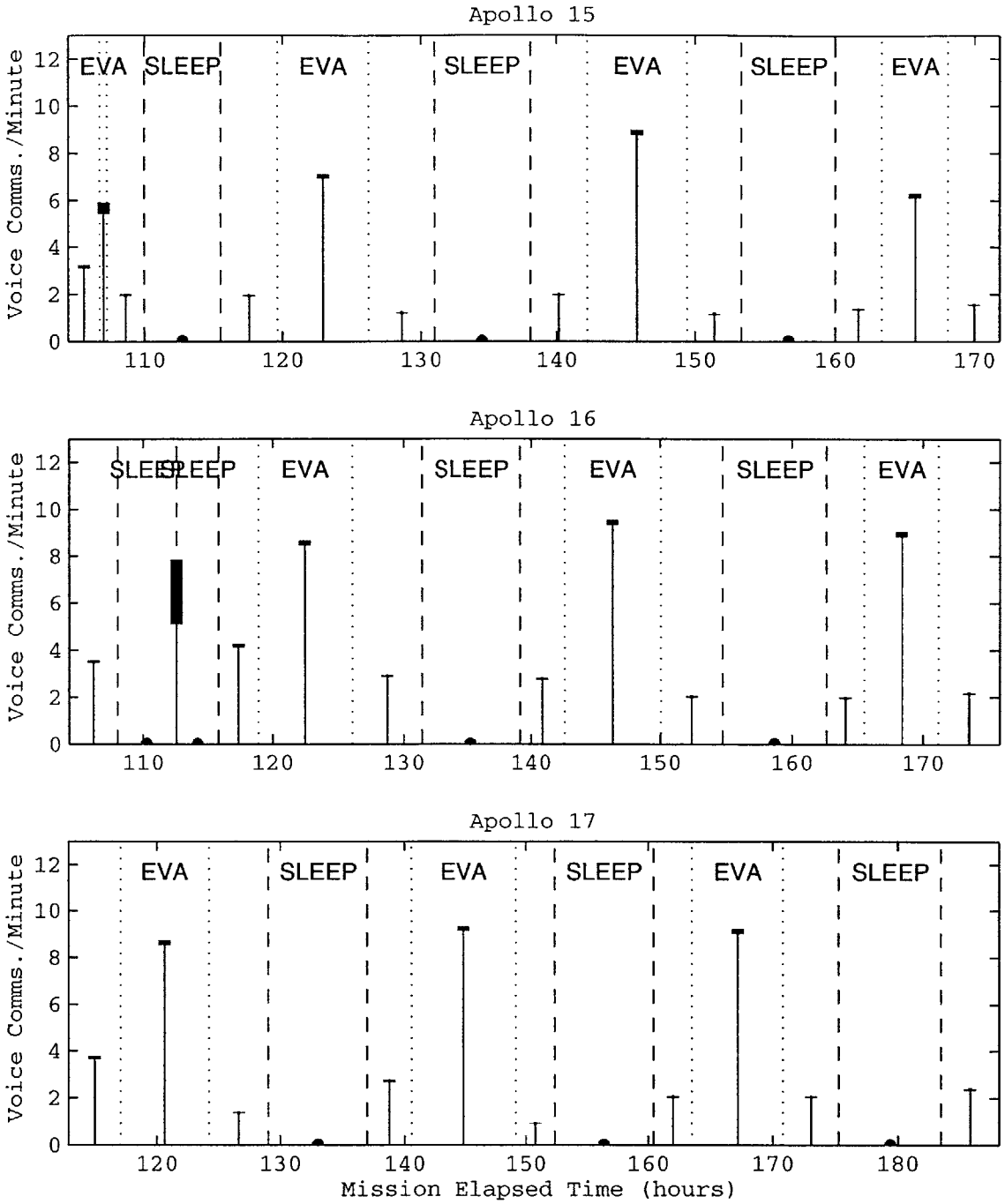


FIGURE 4-39. Mean communication rates during extravehicular activity and non-extravehicular activity during Apollo 15, 16, and 17 as a function of Mission Elapsed Time. Bar heights indicate 95% confidence intervals, computed by treating voice communication "arrivals" as a Poisson process (EVA = extravehicular activity).

It is clear that voice communications are of major importance during extravehicular activity. The continual flow of information from lunar surface astronauts to the CapCom greatly enhanced the value of data collected by astronaut-deployed experiments, rolls upon rolls of film, and lunar rocks. Many of the Apollo voice communications dealt with the lunar topography: continual knowledge of the position of the lunar surface astronauts on the lunar surface was

critical to achieving extravehicular activity mission goals while ensuring adequate crew consumables and safety. While crews after Apollo 14 used the Lunar Roving Vehicle for much of their regional traverses, the crew of Apollo 14 made one of the most extensive lunar walking traverses during their second extravehicular activity. During this traverse, the crew demonstrated both the critical role played by voice communication, and the metabolic cost of walking on lunar terrain.

Metabolic Cost Analysis. Metabolic cost analysis is to human explorer mobility what voice communication is to information mobility between human explorers. Real-time metabolic cost estimates were used to guide adjustments to the nominal extravehicular activity plan such as when to allow extensions to the extravehicular activity based on remaining consumables, when to discontinue certain activities, reduce or pick up the pace of a traverse, or adjust the workload of an activity [Apollo Experience Report, 1975]. Mean metabolic cost was computed between different geological points of interest for the second traverse of Apollo 14, and is presented along with segment traverse distances, changes in altitude, and mean traverse velocities in Johnston et al. (1975). Metabolic power (kJ/hr) was divided by the mean traverse velocity (km/hr) to give a simple metric of metabolic cost (kJ/km) (Figure 4-40).

For a more detailed discussion of metabolic cost, and how a more powerful non-dimensional cost of transport can be computed and applied to traverses, see Chapter 5.

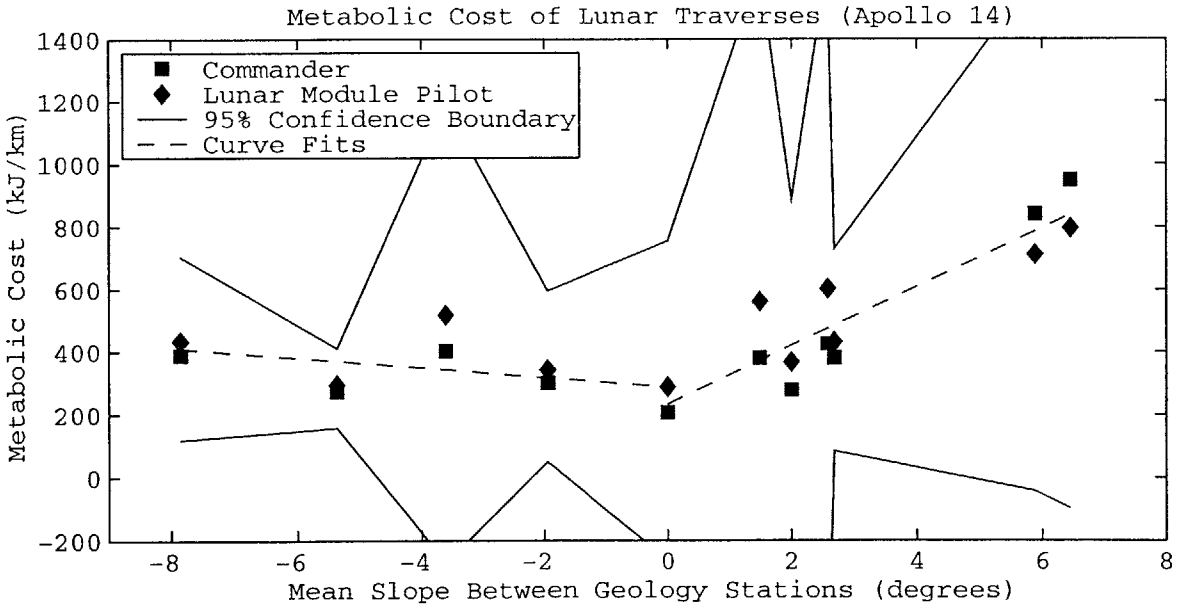


FIGURE 4-40. Metabolic cost of lunar surface astronauts during the second extravehicular activity of Apollo 14. Slopes and metabolic cost in kJ/km are computed from Johnston et al. (1975). Correlation coefficients are 0.83 and 0.23 for the positive slope curve fit and negative slope curve fit, respectively.

The form of the fitted curves in Figure 4-40 agrees with the metabolic cost predicted by the load carrying model described in Chapter 5: metabolic cost is a strong function of slope for positive slopes, and is close to linear. For negative slopes, metabolic cost tends to follow a more non-linear function, but in general metabolic

cost tends to be a minimum for small negative slopes, and to increase as the slope becomes larger in magnitude. Factors that contribute to variability in the above data may include differences in metabolism between the two Apollo 14 lunar surface astronauts, the presence of a space suit (and resultant mobility restrictions), and the "noise" associated with non-locomotion related activities such as traverse navigation, rock sampling, photography, or other lunar surface activities. The important point is that metabolic cost is highly sensitive to the surface slope, and that a slope of five or six degrees can more than double the metabolic cost. Therefore, when resource constrained exploration tasks require significant mobility on a surface with significant topographic variability, modeling metabolic cost becomes critically important to maximizing the probability of mission success.

Positioning and Navigation Challenges. The lunar surface astronauts used voice communications to provide scientific descriptions of the lunar surface to mission control in conjunction with other scientific activities, but many of the voice communications relate to what might be called "exploration management" activities relating to positioning, navigation, or timing. Lunar explorers judged distances and estimated their position using several techniques including:

- Dead reckoning navigation between geologic stations based on previous position estimates and traverse maps,
- Considering the size of the lunar module, if visible, and thereby estimating relative distance from the lunar module (a procedure described in lunar surface operations checklists), and
- Evaluating any large visible changes in topography or nearby craters and attempting to match them to the traverse maps.

During the Apollo 14 mission, Commander Shepard and Lunar Module Pilot Mitchell encountered significant challenges finding geologic station A and B during their second extravehicular activity. In the subsequent 1971 debriefing, both identified lack of prominent landmarks as the largest contributor to their difficulties [Jones, 2000]. The following exchange with CapCom Haise illustrates the difficulty (numbers on left are mission elapsed time):

> 131:52:37 Shepard: Probably [station] A right here, is it not?
>
> 131:52:38 Mitchell: It's right over here to our left a little bit, Al. I believe. (Pause) Well, let me see.

A few moments later Shepard again responds:

> 131:55:36 Shepard: The point where we're sampling is (pause) just about in the center of three craters of almost equal size. I would say, perhaps, 20 meters in diameter... I'm pretty sure we're just about where point A is on the map; as I recall, it fits the description of it. (Long pause)

An editor of the Apollo Lunar Surface Journal quickly points out that obvious differences in the crater patterns between the actual location of Shepard and Mitchell at that time and the planned location of station A on the traverse map, but further states "from Al's and Ed's point-of-view, the craters are not easy to sort out and the potential for confusion is easy to understand."

In the 1971 debrief of Apollo 14, Mitchell describes optical distortion in the helmet as a potential contributor to distance estimation problems, but agrees with Shepard that the primary problem in navigation was the lack of prominent landmarks caused by the "roughness and undulating character of the lunar terrain." Mitchell continued:

> You'd say, 'Well, this next big crater ought to be a couple of hundred meters away, or 100, or 150.' It just wasn't anywhere in sight...You could not get enough perspective from any one spot to pin down precisely where you were. The undulations over the neighborhood were probably 10 to 15 feet (high)...It looked like we were in a large group of sand dunes (from [Jones, 2000], Apollo 14 transcript, 132:32:57 Mission Elapsed Time).

Additional difficulties in identifying features were caused by the direction of sight relative to the sun. Apollo astronauts reported that identifying features was easier while traveling in cross-sun directions as compared up-sun (toward the sun) directions or down-sun directions.

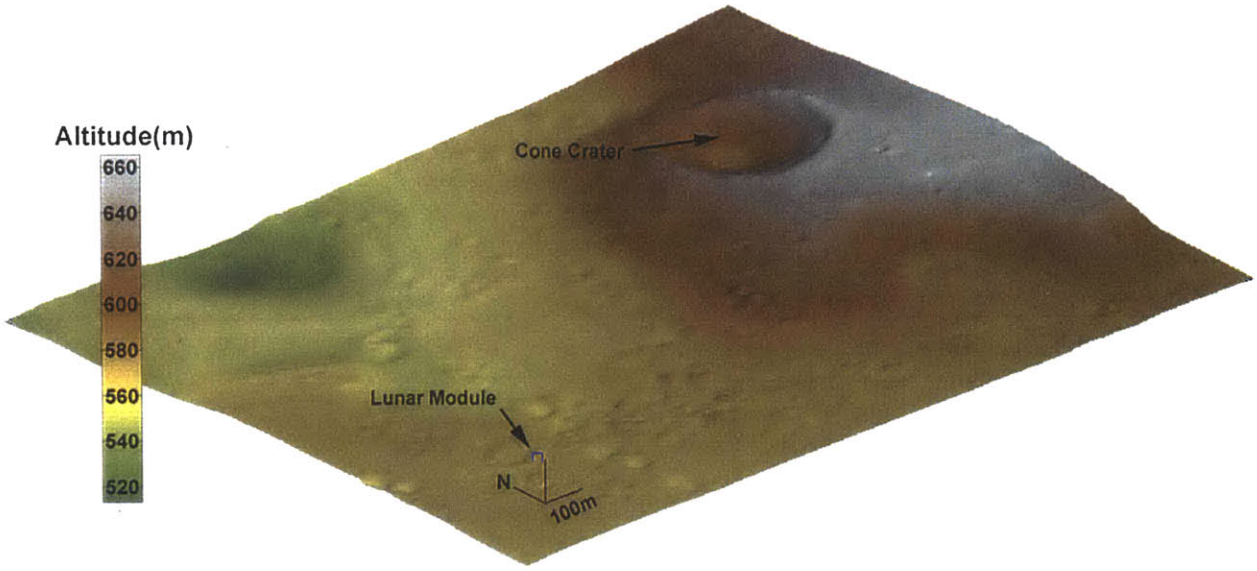
Based on the previous discussion, position estimation involves (a) recognizing surface features that stand out as compared to other surface features, and (b) estimating the distance of those features from the current position. These activities are intimately related both to the visibility of the surface of interest, and the statistical nature of the surface topography (see Chapter 3, Section 3.2). When terrain is self-similar (as might be expected on the lunar surface due to the cratering process), terrain will look the same at all scales. Without atmospheric degradation, and without non-self-similar features (such as a forest of trees on Earth, or the Lunar Module on the moon) distance parts of a self-similar lunar surface would look the same as nearby parts, creating a situation where distance estimation is extremely difficult.

Fortunately, the Lunar Roving Vehicle (with its inertial guidance system) allowed the crews of Apollo 15, 16, and 17 missions to estimate their position on the lunar surface much more easily once an initial position fix had been determined.

Visibility Analysis: Finding Cone Crater. During the second extravehicular activity of Apollo 14, Shepard and Mitchell traversed up a ridge to gain access to the rim of the 370 meter diameter Cone Crater to obtain samples of the underlying bedrock (ejecta near the rim was likely to be derived from a greater depth due to the

dynamics of the impact event that formed the crater). During traverses from one geologic station to another, they looked back at the Lunar Module to gauge their relative distance from the Lunar Module. Navigation challenges and time constraints prevented the two astronauts from reaching the actual crater rim, even though they came within about 40 meters of the crater rim. Because of the challenges in estimating their position relative to the crater rim, valuable time was spent trying to obtain better position estimates or find the crater rim by dead reckoning.

To perform a visibility analysis of the Apollo 14 traverse, a digital elevation model of the Apollo 14 traverse area (Figure 4-41) was created based on a topographic and aerial map of the Fra Mauro Highlands [Swann et al., Plate 2].



This digital elevation model is based on widely spaced contours and additional contours and craters were interpolated by hand. Error of the digital elevation model has not been assessed, but the digital elevation model does (at least qualitatively) demonstrate the problems faced by Shepard and Mitchell in their attempts to reach the rim of Cone Crater. Computation of a radial power spectral density for the digital elevation model demonstrates that the digital elevation model is self-similar, as might be expected due to the nature of the cratering process. The power law scaling exponent is approximately 4.9. For small scale lengths, the power law scaling curve shallows; this is indicative of a lack of small craters in the digital elevation model (only large craters and smaller craters in the immediate traverse area were included in the digital elevation model because of time considerations).

FIGURE 4-41. Digital elevation model of the Apollo 14 traverse area. The second extravehicular activity included one of the most extensive foot-based traverses of the Apollo program: an attempt to reach the upper rim of Cone Crater from the Lunar Module.

Figure 4-42 illustrates surface visibility from the Lunar Module, based on this digital elevation model.

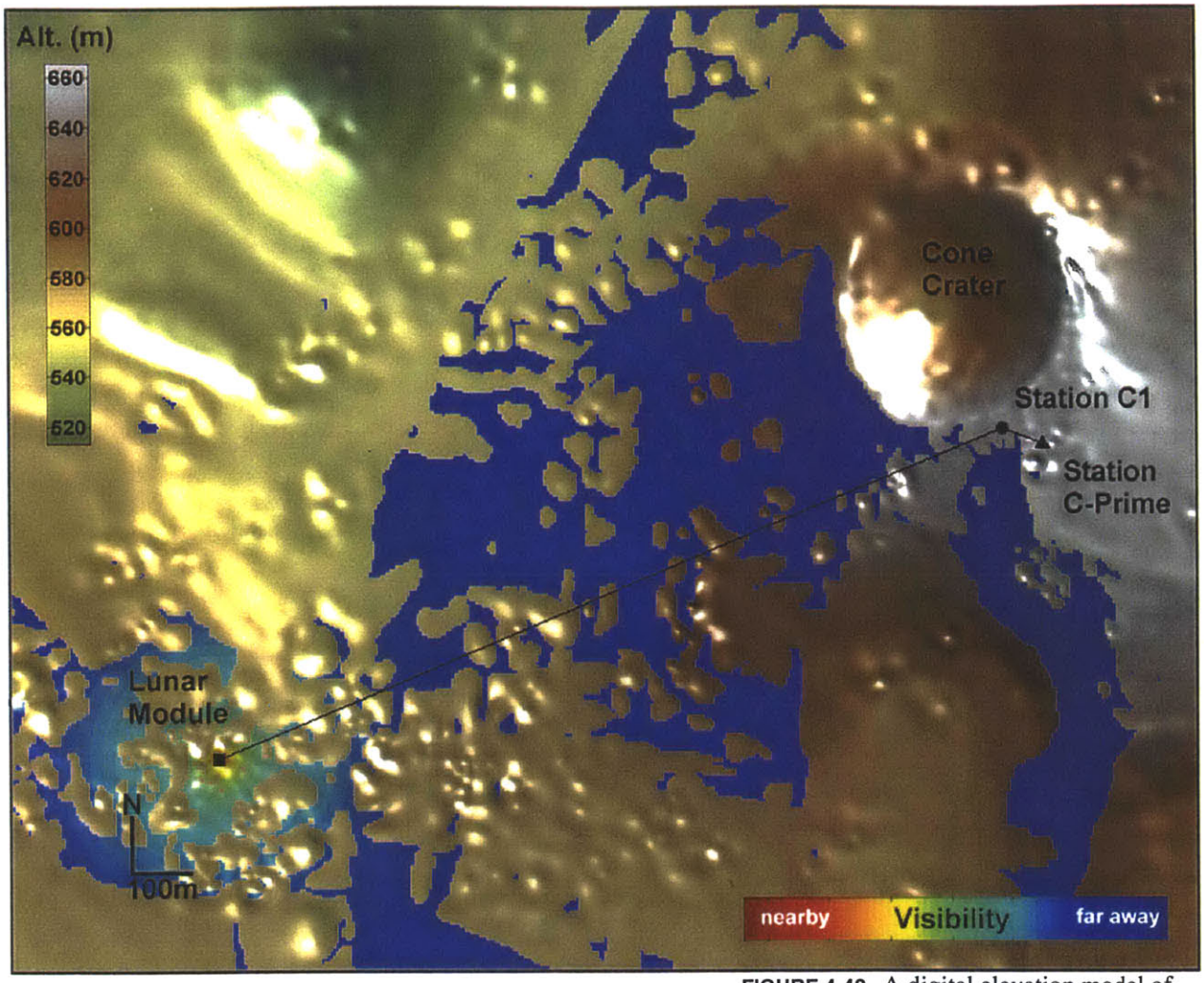


FIGURE 4-42. A digital elevation model of the Apollo 14 traverse area demonstrates the position of geologic stations C-prime and C1 relative to the Lunar Module. Visibility of the lunar surface from the Lunar Module is shown. Line-of sight analysis between a seven meter tall Lunar Module and a 1.8 meter tall astronaut at stations C-prime and C1 indicate that an astronaut at station C-prime might be able to see station C1, and an astronaut at C1 should be able to see the Lunar Module (indicated by dark lines).

Visibility analysis from the position of the lunar module demonstrates the extensive view of two flanks of Cone Crater. It should be noted that the visibility analysis here is one-way and represents visibility of the lunar surface: an astronaut standing on the surface in an area not visible from the Lunar Module might still be able to see the Lunar Module because of his height. In addition, crater geometries in the digital elevation model are inexact at best.

Figure 4-43 demonstrates the visibility of the lunar surface from geological stations C-prime and C1. In both cases, the predominant view is of a very localized area, and no real definitive features of Cone Crater are visible (no view into the Crater is possible, consistent with actual Apollo 14 experiences).

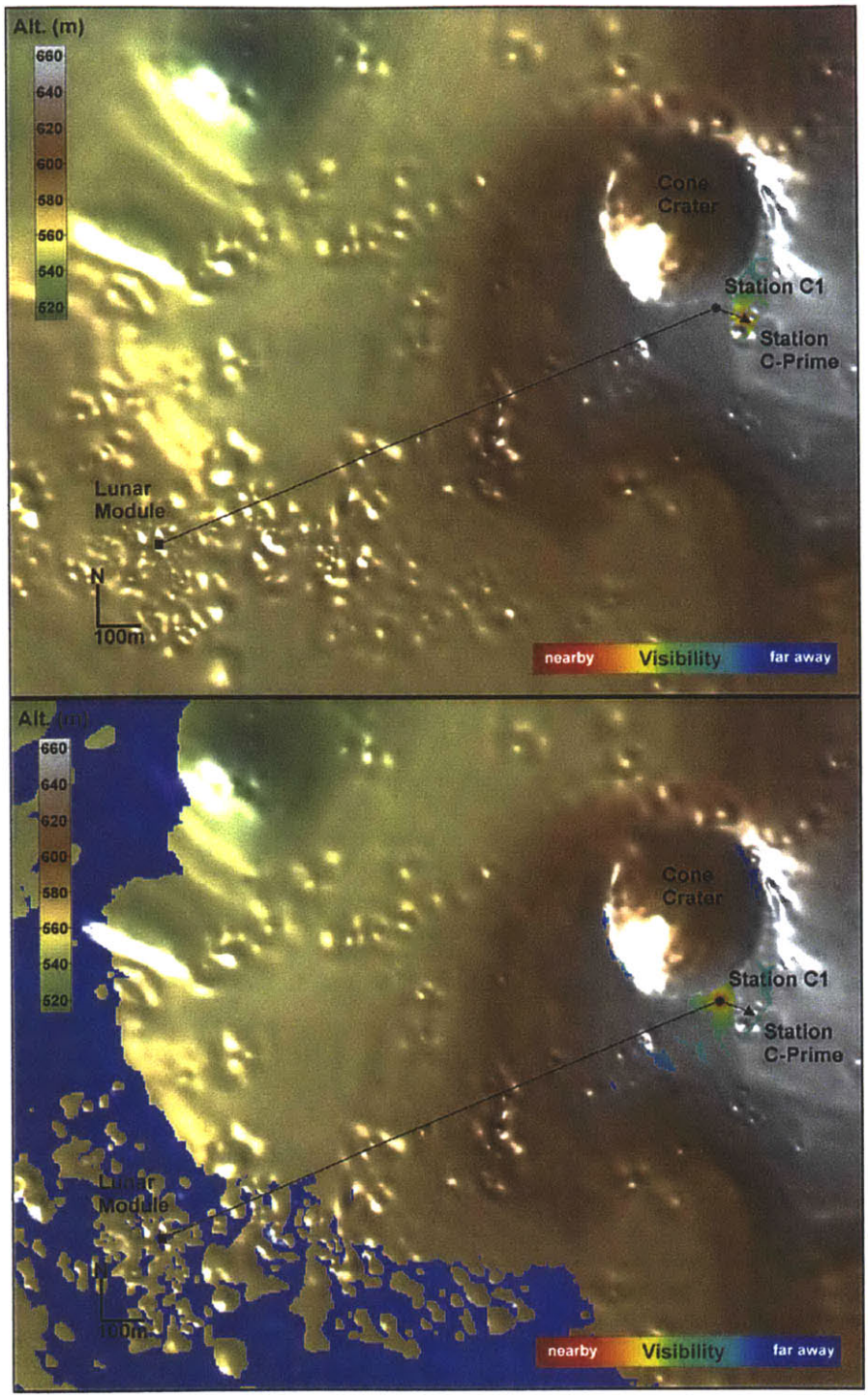


FIGURE 4-43. Visibility of the lunar surface from geological station C-prime (top) and station C1 (bottom). Only nearby features are visible at C-prime, and at station C1it would still be extremely difficult to positively identify the rim of Cone Crater.

The small bits of the rim of Cone Crater that might have been visible from station C1 would likely have been challenging to recognize as the actual rim of Cone Crater. Indeed, the comments from Shepard and Mitchell while at station C-prime illustrate how difficult it was just to identify station C1:

133:22:58 Shepard: Oh, the rim. That is negative. We haven't found that yet. (Pause)

133:23:10 Mitchell: This big boulder right here (on the traverse map), Al, which stands out bigger that anything else (undoubtedly Saddle Rock [at station C1]) ought... We ought to be able to see it.

[Because he has no references to help him judge size and distance, Ed does not recognize that the large boulder on the map is in sight. Al will first call attention to the "white boulder" at

133:25:40. Later, they will go over to Saddle Rock [station C1] and collect samples.]

133:23:17 Shepard: Well, I don't know what the rim is still way up here from the looks of things.

133:23:23 Haise [CapCom]: And, Ed and Al, we've already eaten in our 30-minute extension and we're past that now. I think we'd better proceed with the sampling and continue with the EVA [extravehicular activity].

An analysis of visibility along the nominal traverse plan might have provided clues that navigation near stations C-prime and C1 would be challenging, and might have led to changes in the planned geological traverse.

4.2.4 Summary

The voice communications study demonstrated that the bulk of voice communications for the lunar surface portion of the Apollo missions were made by the Commander, Lunar Module Pilot, and primary CapCom. The Commander and Lunar Module Pilot spoke most frequently, and spoke for comparatively shorter durations. The duration of voice communications by the Commander and Lunar Module Pilot followed a decreasing trend during later missions.

Voice communication rates were much higher during extravehicular activity periods than during non-extravehicular activity periods, and extravehicular activity period voice communication rates increased during later missions as compared to earlier missions. Non-extravehicular activity voice communication rates failed to show a similar trend and stayed significantly lower than the extravehicular activity voice communication rates.

Voice communication was heavily and increasingly used during the Apollo missions, especially during extravehicular activity. Communication happened on shorter and shorter time scales and more and more frequently during later missions, likely because of increased workload on the lunar surface astronauts and perceived efficiency gains by the use of more frequent, shorter voice communications. Although no inclusive analysis was done of the intended recipients of voice communications during Apollo, readings of some of the

voice communication transcripts from the later Apollo missions suggest that a great deal of the shorter frequent communication occurred between the two lunar surface astronauts: the shorter time intervals and more frequent communication probably are effects of the synergy developed between the lunar surface astronauts.

Frequent communication with Mission Control worked well during the Apollo missions, but will not be viable for future planetary missions that may include significantly longer communication delays. Real-time support during future planetary missions therefore must have a very different structure. Voice communication must play a very different role during Mars surface exploration, although similar operations may take place between an astronaut exploring the Martian surface and an astronaut in a Mars base (analogous to the lunar surface astronaut talking to mission control).

A metabolic cost analysis of Apollo 14 traverses demonstrates the strong dependence of metabolic cost on surface slope. Consequently, metabolic cost modeling and estimation becomes critical for exploration activities involving significant mobility on a surface with topographic variability. Errors in navigation can further reduce the resources that can be devoted to science or other exploration activities not directly related to a traverse.

Navigation challenges can be compounded by topographic variability, especially for self-similar surfaces and very rough surfaces that lack identifiable landmarks. Identification of surface features and estimation of distance to features were especially challenging on the lunar surface.

Finally, a visibility analysis of Apollo 14 navigation challenges during the search for the rim of Cone Crater illustrates the importance of visibility analysis in the traverse planning process: When adequate surface topography data is available, visibility analysis can be incorporated into the traverse planning process to ensure that checkpoints provide adequate visibility of prominent features that would enhance surface navigation.

4.3 Discussion

The two case studies presented in this chapter have demonstrated some of the analytical tools necessary to plan for traverses, and have demonstrated the need to carefully incorporate surface characteristics and visibility considerations into the traverse planning process. The Apollo case study also demonstrated the importance of replanning a traverse in real time, and demonstrated past reliance on voice communication as a mechanism for delivery of information to or from human explorers. Current networking technologies offer many other approaches to delivering information that need to

considered and evaluated. These technologies may also support some of the distributed interaction and communication activities found to be important during field geology, and also provide different approaches to providing real-time support to explorers without the potential disruption of constant voice communications.

4.3.1 Traverse Planning and Analysis

While traverse planning activities were not likely to be useful during the early field geology activities of the author in the Bird Spring Mountains, traverse planning might have contributed to field geology activities late in the geologic mapping process (especially in more extreme environments where resource constraints are more severe). Traverse planning is more useful when a good set of "points of interest" are known, such as was the case during the Apollo program. During the Apollo program, extensive planning was conducted for each traverse, yet many difficulties were still encountered. Because of the large and diverse set of data relating to the Martian surface, many "points of interest" already exist for the Martian surface, and the list is only growing longer. This suggests that traverse planning will be invaluable for Mars surface exploration.

The traverse planning process should incorporate a variety of analyses, including an assessment of visibility and timing issues, surface characteristics including surface slopes, metabolic cost estimates, and consumables usage predictions. The significant effect of surface slope on metabolic cost was demonstrated by data from Apollo 14. The Apollo lunar surface astronauts also recommend that traverse planning should incorporate well defined checkpoints, visible from previous checkpoints. Statistical analysis of surface topography may also provide an indication of where distance estimation may prove to be difficult. Permitting rapid traverse replanning during exploration would enhance the flexibility and robustness of the exploration process by permitting changes in the execution of the traverse to achieve a different set of traverse goals, or account for new constraints.

During the Apollo missions, the traverse replanning process was largely carried out in Mission Control. Traverse replanning for Mars surface exploration must account for two key challenges:

- A light travel-time delay much larger (in many cases) than the time constant over which traverse replanning decisions need to be made, and,
- A relative dearth of expert planners to assist in the replanning process.

The first challenge requires that traverse replanning be Mars-centric, while the second challenge suggest that the traverse replanning

process needs to be low workload from the perspective of Martian explorers. Significant automation of the traverse replanning process could make it possible for the process to be explorer-centric - carried out by the astronaut or robotic explorer while exploring. Automating the process would make it possible for robotic and human explorers to use similar replanning processes (although with different models of performance and capabilities, and different criteria relating to resource margins and other issues of safety). The traverse replanning activity would also need to include an assessment of whether the proposed traverse complied with the applicable set of "flight rules": for human explorers this might include a walk-back allowance in case of a vehicle failure, or a required margin of consumables at the end of the planned traverse. Results from previous traverses could be used to adjust the margins required by flight rules to appropriate levels.

Perhaps the process of traverse planning and authorization could be similar to filing a flight plan: an astronaut's "traverse plan" would need to meet some minimal flight rules and could be (electronically) filed or amended during extravehicular activity. Under certain conditions, no approval might be required from any Earth-based Mission Control, and under other circumstances, an astronaut might need to wait for approval (clearance) from Mission Control. This situation is roughly analogous to filing a flight plan under Visual Flight Rules, for which no clearance is required, or Instrument Flight Rules, for which a time-specific clearance is required. In this type of system, the responsibility of conforming to specific flight rules and filing a "traverse plan" is traded for the flexibility to make changes to a traverse during extravehicular activity without specific approval from Mission Control. Like the Pilot-In-Command of an aircraft, the astronaut should have the ultimate responsibility and decision making power in the event of an emergency.

4.3.2 Sensing and Perception Challenges

The two case studies illustrated the extent to which field geologists and lunar explorers relied upon sight as their primary method of observation during exploration. Visual sensing challenges described in both case studies can suggest what might be some of the primary visual challenges for Mars surface exploration.

The light intensity at Mars is approximately 38% of the light intensity at Earth due it its greater mean distance from the Sun (the light intensity on the Martian surface might be much lower during an intense dust storm). The thin Martian atmosphere tends to produce less scattering than the comparatively thick atmosphere of Earth, and shadows will tend to be darker. While contrast between lighted and shadowed areas can be useful, it can also be a challenge: Apollo astronauts walking in the direction of the Sun encountered a surface of dark shadows, and the problem of resolving objects was

further complicated by pupillary contraction due to the bright Sun. Walking away from the Sun produced a strongly lit surface for which the astronauts also had difficulties in identifying features.

Spectral differences may also contribute to visual interpretation challenges for human explorers. While the direct spectrum of the Sun on the Martian surface is similar to the direct spectrum encountered on the surface of Earth or the Moon, the reflected spectrum is rather different (due to differences in surface composition). The field geology case study suggested that simple modifications of the spectral content (such as an optical coating on a pair of sunglasses) can affect the ability of human explorers to detect differences in bedrock composition or coloration. Optical coatings for space suit visors, or optical distortion cause by visor geometry, should be carefully considered. Artificial lighting might be one solution to enhancing visual interpretation for human explorers on the Martian surface.

Traverse planning should therefore take into account the direction of the Sun relative to the direction of planned observations and the expected surface lighting conditions, including the spectral content of the reflected light. The reduced intensity of light, coupled with the thin atmosphere, make the timing of observations comparatively more important than on Earth.

4.3.3 Data Collection during Exploration

The field geology case study illustrated the importance of note taking and sketching for illustrating visual features and mental models of potential geologic structures. While sketching and photographic documentation might be completed fairly rapidly, the slow nature of note taking forces one to think carefully about the words being recorded. While photographic documentation of sample sites during Apollo was fairly workload intensive, automation of this process could reduce the workload. One approach to reducing the workload of photo-documentation during Apollo is shown in Figure 4-44.

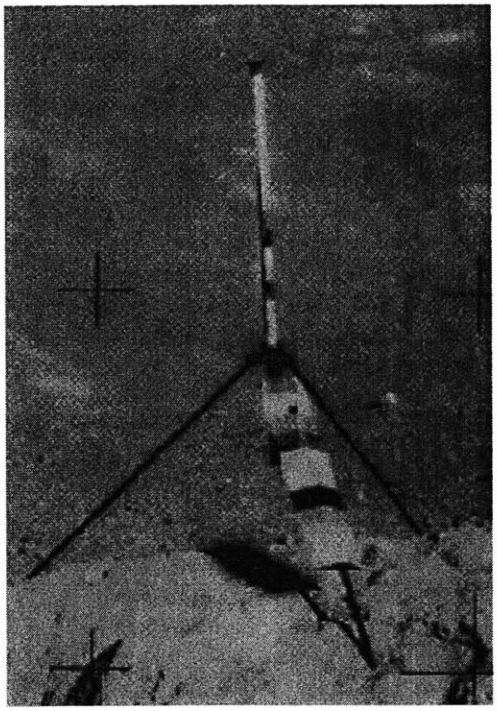


FIGURE 4-44. Use of this device, called a gnomon, reduced the photo-documentation workload of Apollo lunar surface astronauts. A photograph of the gnomon and a rock sample would provide contextual data for the sample including a scale, a local vertical, an approximate time and illumination (from sun angle), and grey scale and color references for color calibration (the objects attached to the nearest leg of the gnomon). This photo is a subset of Apollo photograph AS16-109 17802, from [Ulrich et al., 1981], Figure 18.

During the Apollo missions, extensive visual documentation of geologic features combined with a running commentary served a similar purpose as the notes and sketches in the field geology case study. Because of the short time spent on the lunar surface, exploration goals during the Apollo missions were primarily related to collecting information about the lunar surface and less focused on interpreting that data to yield a coherent picture of the lunar surface. For example, no real attempts were made by the lunar surface astronauts to develop a detailed stratigraphy of the lunar surface.

One of the ways in which humans will contribute to the exploration of the Martian surface is by their ability to generalize: this will require both collection and interpretation, and sketching will likely

be an important activity. While voice recordings are likely to be utilized, sketches provide a powerful approach to quickly record and communicate mental models that written language cannot match. Voice recording is a natural way to record observations or textual data, accuracy may be a problem depending on the noise environment of the space suit, and voice recording of data may interfere with other voice communications.

Providing a minimal ability for astronauts to sketch or annotate images during extravehicular activity would greatly enhance the value of human extravehicular activity operations. Other options might include extensive verbal- and photo-documentation (performed by an astronaut or perhaps a robotic agent) followed by annotation of images after completion of the extravehicular activity.

4.3.4 Information Delivery During Exploration

In both the field geology case study and the Apollo case study, the primary method of information delivery to explorers (geologists in the field, or astronauts on the lunar surface) was via paper or voice (spoken or transmitted). The sketches and notes from field geology and the verbal descriptions from the Apollo astronauts are not so different from the maps and journal entries of the Lewis and Clark Expedition of 1804-1806. Even current Shuttle and International Space Station operations utilize similar methods to deliver information to astronauts during extravehicular activity: written checklists and two-way voice communications.

The technologies supporting delivery of explorers to extreme environments and providing life support functions to explorers in extreme environments have advanced greatly since the Lewis and Clark expedition, but until recently, the technologies supporting *information delivery* to explorers (or from explorers to others) have remained largely unchanged. The capabilities now exist to deliver information to explorers in many ways, some potentially more efficient than voice communications. While voice communications can be disruptive, they can be extremely effective in the right circumstances and will continue to be effective as an option for recording observations during planetary exploration for an astronaut in the confines of a space suit. Wearable computing technologies may provide an alternative means of information delivery to a space-suited astronaut [Carr et al., 2000, 2001], and could support traverse planning and analysis.

4.3.5 Distributed Systems and Human Exploration

Distributed systems may enable or enhance human exploration by:

- enabling access to a diversity of perspectives,

- providing communications and networking coverage,
- matching computational and physical resources to local needs,
- enhancing intra-team (short-range) and inter-team (long range) communication and coordination,
- enabling distributed, coordinated, measurements, and
- automating information recording and distribution processes.

In summary, distributed systems can support human exploration by changing how explorers access and distribute information, and how explorers coordinate their activities.

4.3.6 Recommendations

Additional traverse planning tools should be developed, in addition to those described in this chapter. Flexibility should be incorporated into the traverse planning process: one approach to achieve flexibility in the traverse planning process would be to enabled replanning of a traverse in real-time. In addition, an approach to extravehicular activity during surface exploration should be developed that utilizes the Earth-Mars light-travel delay time constructively, and does not try to tightly "close the control loop" between Mission Control and explorers on the Martian Surface ("Mission Support" may better describe the role of such an organization, rather than Mission Control).

Support systems that can improve visual context by providing perspective at multiple scales, alternative viewpoints (perhaps from robotic agents), and a virtual equivalent of the Martian surface populated with data in real-time or near-real time. This virtual equivalent could provide mechanisms for robots and humans to assist in the exploration process. References of scale might need to be provided in areas without good landmarks - these references could be physical (such as wands used by climbers to mark out routes on a glacier), or virtual in nature.

Analog simulations can be used to study how robots and humans interact during exploration. Recording time-position-activity data in a transparent manner (such as continuous logging of global positioning system location and automatic logging of tool usage) would allow future extravehicular activity systems to be designed around the explorers that they serve, rather than having future Martian explorers adapt to an extravehicular activity system. Capturing this additional data during the exploration process should improve the ability to reconstruct the exploration process, and add value to the other data collected in the field.

4.3.7 Conclusion

This chapter begins to answer the questions posed at its outset. It has examined how explorers collect, analyze, and disseminate information, and how this information may be used in the planning and execution of extravehicular activity on a planetary surface. Distributed systems fundamentally enhance the exploration process by changing the way in which information is obtained and disseminated by exploration agents. This additional access to information, coupled with appropriate decision-making tools, enhances self-sufficiency and autonomy and allows individual explorers to enhance their probability of survival, coordinate their activities with other explorers, and increase the value they add to their overall mission.

This chapter has set the stage for the more structured discussion of traverse planning in Chapter 5, in which the material on distributed systems from Chapter 3 will be applied to the support and quantification of performance of individual explorers in distributed systems. Of primary importance will be a characteristically human approach to problem solving in the traverse planning process:

> Heuristic search... is in fact the principal engine for human problem solving [Simon, 1981, p. 56].

4.4 References

Allton, J.H., *Catalog of Apollo Lunar Surface Geological Sampling Tools and Containers* (JSC-23454), National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1989.

Apollo Experience Report: Assessment of Metabolic Expenditures (NASA TN D-7883/N75-18271), National Aeronautics and Space Administration, 1975.

Appelbaum, J., and Steiner, A., *Spectral Content of Solar Radiation on Martian Surface Based on Mars Pathfinder*, J. Propulsion and Power, Vol. 17, No. 3, May-June 2001.

Baldwin, J., Fisher, P., Wood, J., and Langford, M., *Modelling environmental cognition of the view with GIS*, Proceedings, Third International Conference on Integrated GIS and Environmental Modelling, 1996 (*http://www.geog.le.ac.uk/jwo/research/dem_char/SantaFe/index.html*).

Burchfiel, Clark B., *Personal communications*, January-March, 2001.

Carr, C.E., and Newman, D.J., *Applications of Wearable Computing to Exploration in Extreme Environments*, 3rd Annual Mars Society Convention, Toronto, Canada, 2000 (submitted).

Carr, C.E., and Schwartz, S.J., and Newman, D.J., *Preliminary Considerations for Wearable Computing in Support of Astronaut Extravehicular Activity*, Massachusetts Institute of Technology, 2001(unpublished).

Chaikin, Andrew, *A Man on the Moon*, Penguin Books, New York, 1994.

Compton, Robert R., *Geology in the Field*, John Wiley & Sons, New York, 1985.

Heiken, G., Vaniman, D., and French, B.M., *Lunar Sourcebook: A Users's Guide to the Moon*, Cambridge University Press, Cambridge, England, 1991.

Hodges, Kip V., *Personal communications*, January, 2001.

Hutchison, William E., *Personal communications*, January, 2001.

Johnston, R.S., Dietlein, L.F., and Berry, C.A., *Biomedical Results of Apollo (SP-368)*, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1975.

Jones, Eric M. (ed.), *Apollo Lunar Surface Journal*, July 2000 (*http://www.hq.nasa.gov/office/pao/History/alsj/*).

Newman, Dava J., *Human locomotion and energetics in simulated partial gravity*, Ph.D. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

Portee, D.S.F, and Trevino, R.C., *Walking to Olympus: An EVA Chronology*, NASA History Office, NASA Headquarters, Washington DC, 1997.

Rechtin, Eberhardt, *Systems Architecting*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Roberto, V., and Chiaruttini, C., *Modeling and Reasoning Techniques in Geologic Interpretation*, IEEE Transactions of Systems, Man, and Cybernetics, Vol. 29, No. 5, September 1999.

Robbins, Tim, *Even Cowgirls Get the Blues*, Bantam Books, New York, 1976.

Santee, W.R., Allison W.F., Blanchard, L.A, and Small, M.G, "A Proposed Model for Load Carriage on Sloped Terrain", *Aviation, Space, and Environmental Medicine*, Vol. 72, No. 6, June 2001.

Simon, H.A., *Sciences of the Artificial*, MIT Press, Cambridge, Massachusetts, 1981.

Swann, G.A., Bailey,N.G, Batson, R.M., Eggleton, R.E., Hait, M.H., Holt, H.E., Larson K.B., Reed, V.S., Schaber, G.G, Sutton, R.L., Trask, N.J., Ulrich, G.E., and Wilshire, G.E., *Geology of the Apollo 14 Landing Site in the Fra Mauro Highlands*, USGS Professional Paper 880, United States Geological Survey.

Thwaites, R.G, editor, *Original Journals of the Lewis and Clark Expedition 1804-1806*, Volumes 1-7 and Atlas, Antiquarian Press Ltd., New York, 1959.

Ulrich, G.E., Hodges C.A., and Muehlberger, William R., *Geology of the Apollo 16 Area, Central Lunar Highlands*, USGS Professional Paper 1048, United States Geological Survey, United States Government Printing Office, Washington, D.C., 1981.

United States Geological Survey Digital Raster Graphics (*http:// terraserver.microsoft.com*).

Waligora, J.M., *The Use of a Model of Human Thermoregulation During the Apollo and Skylab Programs (N76-11172)*, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1976.

Wilhelms, D.E., *To a Rocky Moon: A geologist's history of lunar exploration*, University of Arizona Press, Tucson, Arizona, 1993.

Sailors on a becalmed sea, I sense the stirring of the breeze.

Carl Sagan, *Pale Blue Dot* (1994)

5 *Supporting and Quantifying Exploration*

To understand the power of distributed systems for planetary surface exploration, elements of distributed systems need to be treated not as static nodes, but as a collection of individual mobile exploration agents with dynamic behavior. Allocation of tasks and objectives to individual explorers within a distributed system may also need to take into account constraints and capabilities of individual explorers. Quantifying aspects of the process of exploration facilitates comparison between system elements and helps to provide a basis for task allocation.

A structured application of the tools developed in this and previous chapters will demonstrate how distributed systems can support self-sufficiency, autonomy, and group collaboration and coordination of human and machine explorers during planetary surface exploration by supporting collection, analysis, interpretation, and dissemination of information by individual agents. To bound this task, the exploration process is conceptually segmented into manageable chunks here called *traverses* (Figure 5-1). This chapter:

1. Illustrates how the value and cost of some traverse operations can be quantified, and discusses the role of heuristics in traverse planning and execution.
2. Develops a traverse planning process for elements of distributed systems.
3. Demonstrates how distributed architectures can support individual human and robotic explorers during traverse planning and execution while applying the traverse planning process.
4. Develops recommendations for future extravehicular activity system development and for further work.
5. Summarizes the contributions of the thesis.

Section 5.1 describes the concept of the traverse and demonstrates how the cost of traverse operations, including mobility, can be quantified for human and robotic explorers. Section 5.2 describes the use of heuristics in traverse planning and execution. Section 5.3 develops a traverse planning process. Section 5.4 demonstrates how distributed architectures can support individual human and robotic explorers while applying the traverse planning process. Section 5.5 discusses limitations of the distributed system examples, and sets the stage for the recommendations developed in Section 5.6. Finally, concluding remarks and a summary of contributions of the work presented in this and previous chapters are presented in Section 5.7.

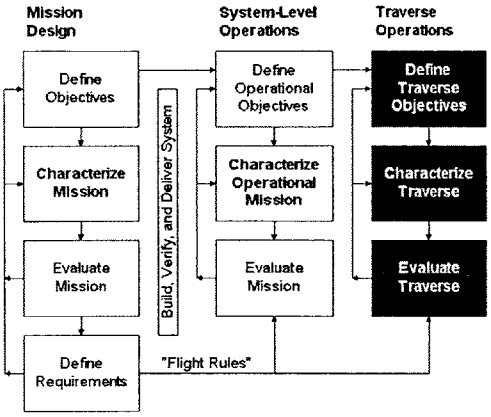


FIGURE 5-1. During surface exploration missions, day-to-day operations are typically broken down into "traverse" operations, in which one or more elements of a system cooperate to achieve a limited subset of overall mission goals while working within the constraints of a set of (typically heuristic) rules called "flight rules." This chapter focuses on how distributed systems, and the analysis tools developed in previous chapters, can be used for planning and execution of traverses.

5.1 The Traverse

5.1.1 The Traverse Concept

Throughout this chapter, the *traverse* will be used as a convenient encapsulation of a set of activities by one or more agents in a distributed system over some defined time period. During a traverse, one or more elements of a system cooperate to achieve a limited subset of overall mission goals while working within some set of constraints that are termed "flight rules" (Figure 5-2) While the word traverse typically implies movement, here a traverse is considered in a broader context to be a path in activity-position space with some defined start and end points (specified in time, position, or activity).

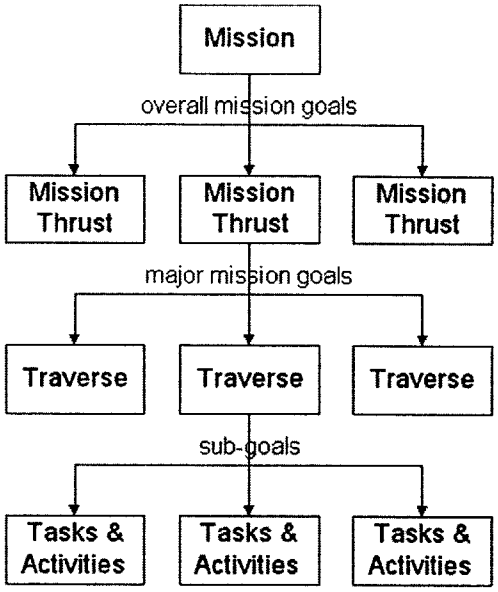


FIGURE 5-2. The traverse can be viewed as an encapsulation of tasks and activities in a larger hierarchy of mission operations.

5.1.2 Quantifying Traverse Operations

Quantification of benefits and costs of traverse operations can form a basis for comparing different traverse options for a given explorer or group of explorers, and for verifying that the current or planned traverse is in compliance with applicable constraints and is consistent with available capabilities. This quantification might also support task allocation between explorers.

Quantification of traverse operations is not a sufficient basis for all traverse operation decisions - in many cases important factors must be considered in traverse planning and execution that are not easily quantifiable. Ultimately, many decisions of traverse planning and execution rely on underlying value judgements about benefits and costs of different outcomes and the relative uncertainty of those outcomes. However, quantifying benefits and costs of different actions, where possible, can enhance the decision-making process and increase the value of exploration. While decision-making depends upon the balance of costs and benefits, this section will focus mostly on the cost side of the equation. While the traverse planning examples later in the chapter clearly consider the value side of the equation as well, these value judgements are meant primarily to illustrate how decisions can be made based on quantitative traverse analysis, not to indicate the correctness of any particular value judgement.

The power of distributed systems comes primarily from information sharing and the ability to change the spatial relationship of the system elements. Cost of information delivery was briefly discussed in Chapter 3 (see Section 3.4.2 and Section 3.6). Additional approaches to characterizing the cost of information delivery can be found in [Shaw, 1999]. Changing the spatial relationship of system elements requires mobility: quantifying the cost of mobility for ele-

ments of distributed systems is therefore the primary task of this section.

Mobility deals with time, position, and orientation: the cost of mobility can be estimated by considering the energy cost of changes in position and orientation with time. Different explorers (humans and robots, for example) may have very different energy costs associated with mobility, but also tend to have different ranges of capabilities. Robotic explorers must manage some internal energy source such as a battery, fuel cell, or radioisotope thermal generator, while energy resources for humans are significantly more complicated (oxygen, food, electrical energy). Metabolic cost will be used as an index of the energy cost associated with human activities. Thermoregulation is a critical issue for both human and robotic explorers that is closely tied to energy costs, but will not be dealt with in this thesis (for an overview of the extensive use of a model of human thermoregulation in the Apollo and Skylab programs, see [Waligora, 1976]).

5.1.3 Metabolic Cost Modeling

Metabolic cost estimates were used during the Apollo missions to guide real-time extravehicular activity planning. Metabolic cost was predicted using three methods including (1) a heart-rate based index of metabolic cost based on pre-flight calibrated values, (2) an oxygen consumption approach with corrections for an assumed rate of suit leakage, and (3) a thermoregulatory approach based on water flow rate and entry and exit temperatures of the liquid cooling garment system worn by the Apollo astronauts [Johnston et al., 1975].

Pre-apollo studies predicted changes in locomotion and energy consumption in sub-gravity environments [Margaria, 1953, Margaria et al. 1964].

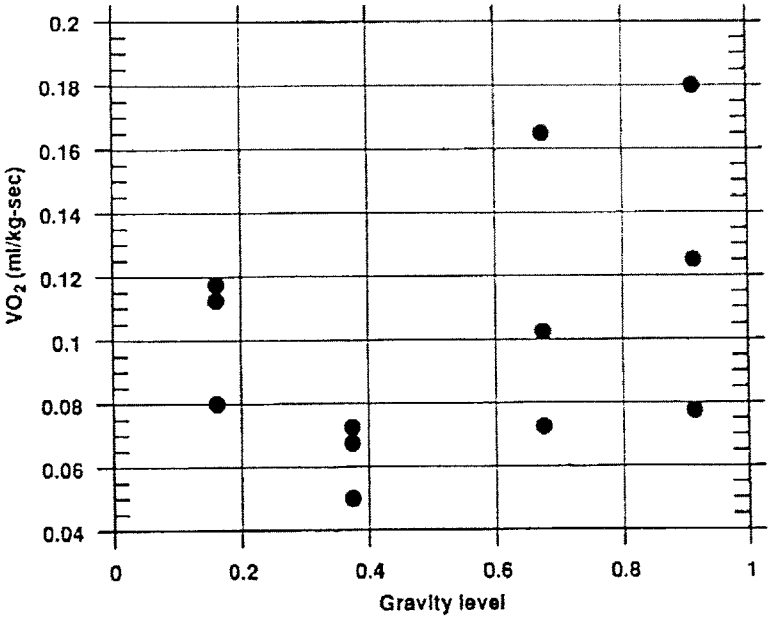


FIGURE 5-3. Oxygen uptake, a measure of metabolic cost, for three subjects walking at 0.5 m/s under partial gravity conditions, simulated using a submersible treadmill. From [Newman, 1992].

These changes were confirmed by the Apollo astronauts [Cavangna, 1972], and more recent studies (Figure 5-3) have demonstrated a possible minimum in metabolic cost of walking at slow speeds (0.5 m/s) in simulated Martian gravity (3/8g) as compared to higher or lower levels of simulated gravity [Newman, 1992].

These results clearly indicate that metabolic cost is not a purely linear relationship as a function of traverse speed or gravitational field strength but instead has non-linearities relating to gait patterns

(walking, running, or loping) and the relatively higher energy cost of maintaining postural stability in <3/8g environments [Newman, 1993].

Load carrying model. Despite these non-linearities, linear models can provide some benefits. For the purposes of this thesis, a simple load carrying model [Santee et al., 2001] will be used to estimate metabolic cost as a function of velocity and surface slope. The load carrying model assumes that the total energy cost of walking while carrying a load can be decomposed into the cost of level walking W_L and the cost of vertical displacement W_V, where:

$$W_L = 3.28 \cdot m + 71.1 \text{ [W]} \quad \textbf{(EQ 5-1)}$$

where m is the mass of the person and load, and the traverse speed is a nominal 1.34 m/s. A linear model can be used to account for off-nominal velocities by writing:

$$W_{L,v} = W_L \cdot R \text{ [W]} \quad \textbf{(EQ 5-2)}$$

with

$$R = 0.661 \cdot v + 0.115 \quad \textbf{(EQ 5-3)}$$

where v is the traverse velocity in m/s. The expression for W_V depends on the direction of climb: the energy cost of uphill vertical work is driven by muscle efficiency, and can be estimated by:

$$W_{V,Up} = k \cdot m \cdot g \cdot h \cdot s^{-1} \text{ [W]} \quad \textbf{(EQ 5-4)}$$

where g is the gravitational acceleration ($1g = 9.8\,\text{m/s}^2$ at Earth sea level), h is the vertical distance ascended in meters in a time period of s seconds, and $k = 3.5$ is an empirical constant representing a muscle efficiency of 29%. Total energy expenditure per unit time while ascending is therefore given by:

$$W_{Up} = W_{L,v} + W_{V,Up} \text{ [W]}. \quad \textbf{(EQ 5-5)}$$

Downhill vertical work is lower than uphill vertical work for low-angle slopes but as the slope angle increases, the increased role of energy absorption into joints and muscles and voluntary braking actions reduce the efficiency of the process, thereby increasing the required downhill vertical work. Slope dependence of vertical work is modeled as:

$$W_{V,Down} = (k \cdot m \cdot g \cdot h \cdot s^{-1}) \cdot 0.3^{(-|\alpha|/7.65)} \text{ [W]} \quad \textbf{(EQ 5-6)}$$

where the efficiency factor $k = 2.4$ corresponds to an efficiency of 42%, and α is the slope angle in degrees. Total energy expenditure per unit time while descending is therefore given by:

$$W_{Down} = W_{L,v} + W_{V,Down} \text{ [W]}. \qquad \text{(EQ 5-7)}$$

This load carrying model was applied in Chapter 4 to estimate the metabolic cost of traverses during field geology (see Section 4.1.4).

Above about 1/2g, running has a significantly higher energy cost than walking, so the load carrying model tends to underestimate the cost of high velocity traverse speeds. Below 1/2g (such as on surface of the Moon or Mars), running and/or loping gaits may be higher velocity and lower power than walking gaits (lower energy per unit distance and per unit time) [Newman, 1993, and Stone, 1974]. Therefore, for Lunar and Martian traverses, this load carrying model will tend to provide a conservatively large estimate of metabolic cost, especially at high traverse velocities, all other factors being equal.

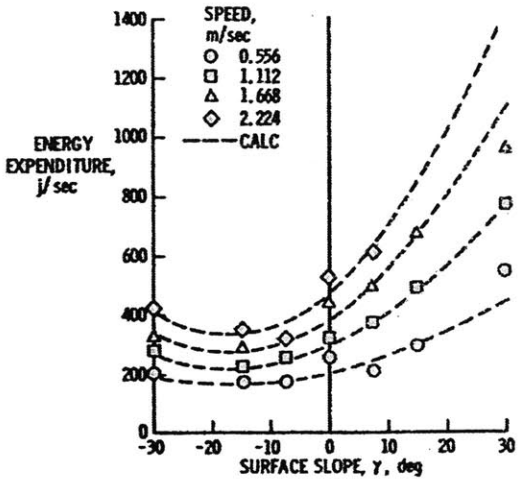


FIGURE 5-4. Effect of surface slope and velocity on energy expenditure during locomotion in simulated lunar gravity for pressure suited subjects (25.5 kN/m^2). From [Stone, 1974], Figure 5.

Figure 5-4 plots measured energy expenditures for suited subjects in simulated lunar gravity as a function of surface slope and velocity. The load-carrying model has a similar form at 1g but the curves flatten out at 1/6g as shown in Figure 5-5.

As expected, the load carrying model overestimates the metabolic cost for higher velocities. However, the load carrying model does not seem to overestimate the metabolic cost for high surface slopes. The additional work of moving in a pressure suit is likely to impose additional inefficiencies during locomotion, especially while ascending steep slopes. This may contribute to the underestimation of the metabolic cost for high surface slopes by the load carrying model.

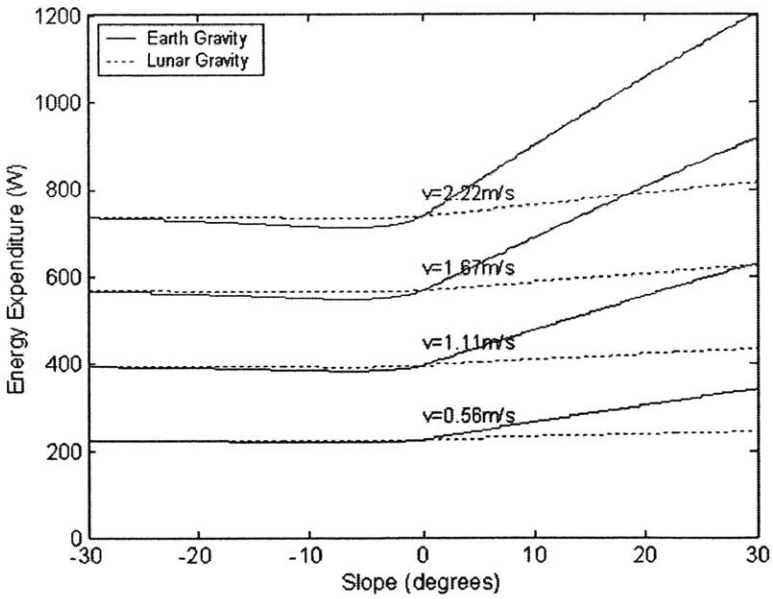


FIGURE 5-5. Effect of surface slope and velocity on energy expenditure in 1g and lunar gravity environments as predicted by the load-carrying model, based on an 80 kg individual carrying a 40 kg load.

In addition to traverse velocity and surface roughness and slopes, soil characteristics such as the trafficability (the ability of the soil to support weight and provide sufficient traction for movement) also significantly affect metabolic cost (Figure 5-6).

5.1.4 Nondimensional Cost of Transport

To compare energy costs of mobility across different conditions or different forms of travel, it is useful to normalize the energy cost of transport. One common approach is to normalize the energy cost per unit time by the velocity to get:

$$\frac{\text{Power[W]}}{\text{Velocity[m/s]}} = \frac{\text{Energy[J]}}{\text{Distance[m]}}, \quad \text{(EQ 5-8)}$$

which can be further normalized by the mass of the mobile object (or the useful payload of the mobile object, in some cases) to give:

$$\frac{\text{Power[W]}}{\text{Velocity[m/s]} \cdot \text{Mass[kg]}} = \frac{\text{Energy[J]}}{\text{Distance[m]} \cdot \text{Mass[kg]}}. \quad \text{(EQ 5-9)}$$

Dividing this quantity by the gravitational acceleration gives:

$$C_T = \frac{\text{Power[W]}}{\text{Velocity[m/s]} \cdot \text{Mass[kg]} \cdot \text{Grav. Accel. [m/s}^2\text{]}}, \quad \text{(EQ 5-10)}$$

a nondimensional parameter that will be referred to as the cost of transport (sometimes called the specific resistance [Newman, 1992]).

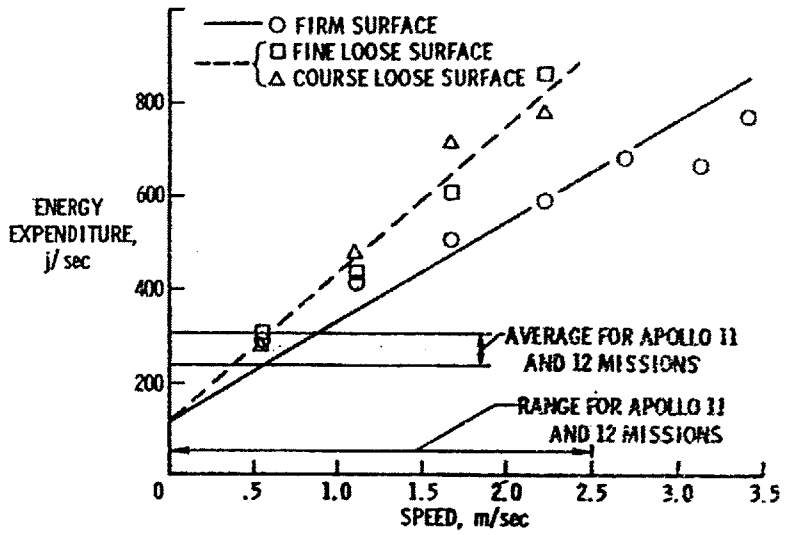


FIGURE 5-6. Effect of surface condition on energy expenditure during locomotion in simulated lunar gravity for pressure suited subjects (25.5 kN/m^2). From [Stone, 1974], Figure 6.

5.1.5 Cost of Transport for Robotic Agents

Many mobile robotic agents have been envisioned for future planetary exploration, but most robotic planetary surface exploration missions have had few, if any, highly mobile components. Mobility during missions such as the Pathfinder technology demonstration (including the Mars rover Sojourner) was limited by significant light-travel-time delays between Earth and Mars, and the short duration of the mission. Recent ideas for mobile explorers include balloons, blimps, hoppers, spider-like or insect-like robots, crawlers, and the traditional wheeled vehicle.

Energy consumption of wheeled vehicles can be accurately modeled given a proper characterization of the wheeled vehicle and environmental operating characteristics including surface characteristics such as soil compaction, roughness, and elevation changes [Heiken et al., 1991]. One particularly salient example is the Apollo Lunar Roving Vehicle: Figure 5-7 demonstrates the excellent agreement between actual Rover cost of transport and the predicted Rover cost of transport from a soil model for each landing site.

Variation in the data that does not agree with the predicted model may be explained by inaccuracies in the soil model or unaccounted-for surface roughness or elevation changes.

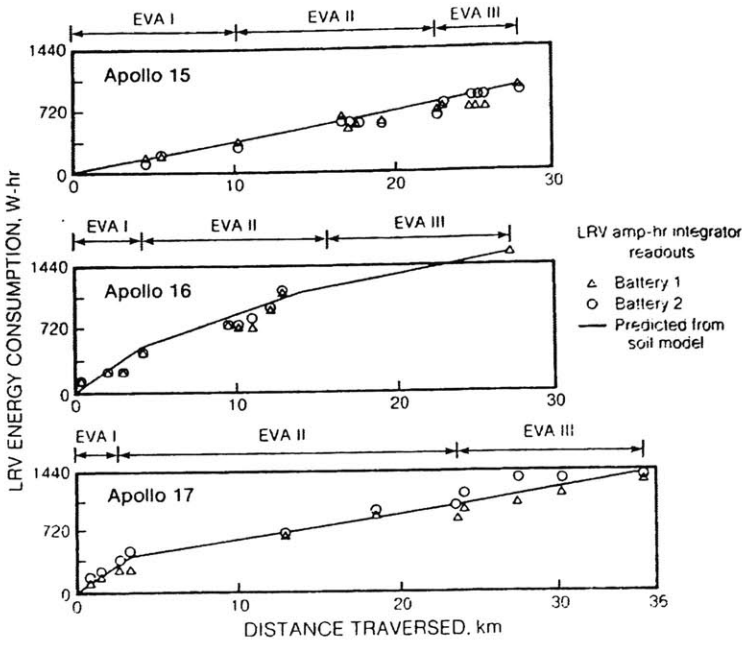


FIGURE 5-7. Unnormalized cost of transport (W-hr) for the Apollo Lunar Roving Vehicle. Cumulative energy consumption is plotted as a function of distance traversed (extravehicular activity [EVA] labels are given at the top of each plot). A soil model for each landing site was developed, and used to (accurately) predict the Rover power consumption. "Mileage" for the Rover averaged 35-56 W-hr/km, or 0.05-0.08 W-hr/km/kg. From [Heiken et al., 1991], p. 528.

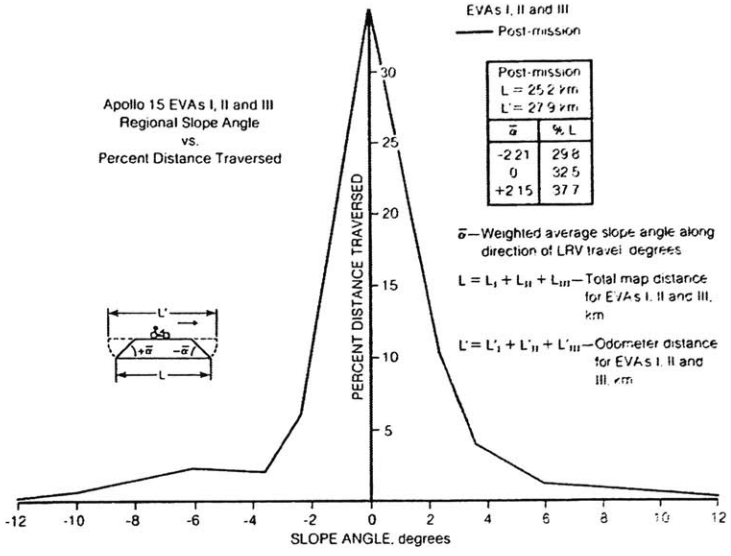


FIGURE 5-8. Surface slopes traversed by the Lunar Roving Vehicle during the three Apollo 15 extravehicular activities (EVAs). The slope distribution is similar in form to the slope distribution of the field geology traverses analyzed in Chapter 4 (see Figure 4-13), but is more-or-less shifted towards lower slopes as compared to the field geology traverses. From [Heiken et al., 1991], p. 530.

The energy per unit distance required to operate the Lunar Roving Vehicle ranged from about 125 to 200 kJ/km. Metabolic cost data from Apollo 14 can be used to estimate the same figure for the second extravehicular activity: the mean cost of transport of the two astronauts was over 500 kJ/km (computed from data in [Johnston et al., 1975]). When one considers that the rover also carried both

astronauts, and did not require other resources such as breathing gases, it seems clear why wheeled transportation was such a success on the moon (at least in terms of energy efficiency during transport). Taking advantage of such efficiency when possible is a good strategy, but exploration of the Martian surface (and other destinations) is likely to require access to locations with more extreme surface roughness than the gently undulating surface of the moon. Matching the transportation method to the surface characteristics will be important regardless of the destination.

5.1.6 Computing the Minimum Cost Traverse

A common problem that emerges when considering how to get from one location to another location is how to find the minimum cost path between the two locations. This problem can be abstractly represented by a graph (Figure 5-9) where edges on the graph correspond to the cost of traversing between adjacent locations. The cost of traversing between adjacent locations can be computing by applying an appropriate energy expenditure model along a path between the two locations. For example, for a human carrying a backpack, the load carrying model might be used to estimate the energy expenditures from one point to another. Dijkstra's shortest (lowest cost) path algorithm (or some other applicable graph algorithm) can then be used to find the minimum cost path between the two locations.

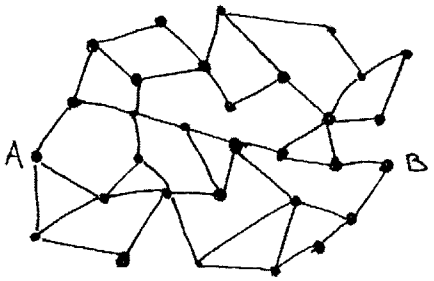


FIGURE 5-9. Graph representation of the least cost traverse problem: Given a set of paths between points A and B, and a cost function $C(i,j)$ for each edge e_{ij} in the graph, find the minimum cost path from A to B.

5.1.7 Cost of System Reconfiguration

An extension of the minimum cost traverse problem is to estimate the cost of a spatial reconfiguration of the elements of a distributed system. The energy cost of such a reconfiguration (from a known initial state to a known final state) might be estimated by summing the cost of traversing from an initial location to a final location for each element of the distributed system that changes position during the reconfiguration. In many cases, the cost of system reconfiguration will not be this simple, because there may be other constraints on the movement of system elements: for example, system elements may need to move in a coordinated fashion such that some particular network structure is maintained.

The energy cost of system reconfiguration does not capture the whole cost of system reconfiguration because of future uncertainty or changes in network capability. Mobility may entail risk: agents may break down or become damaged along the way. There may also be an opportunity cost to system reconfiguration (e.g., data collection is interrupted because of changes in the network topology or reallocation of resources).

Distributed systems may be designed to evolve over time (deployment of the system may involve evolving from a centrally located

to a distributed collection of agents). Hence, estimating the cost of system reconfiguration may be an integral part of the distributed system design process. In summary, the costs related to system reconfiguration not captured by the energy cost of reconfiguration relate to disruption of system activities or risk of loss of capability of the system or of individual agents.

5.1.8 Traverse Cost, Value, and Constraints

The value of a given traverse is driven not only by the costs of the traverse but also largely by the potential benefits of the traverse, such as what sites can be visited or seen along the traverse. A metric that includes benefits and cost (combined according to some appropriate weighting operations) might be computed for each segment of a traverse, and the same approach used to compute the minimum energy cost path from one location to another might be used to compute the maximum value traverse from one location to another. This type of optimization works well for handling "fuzzy" constraints - those that have a continuum of value or cost but not hard limits. To consider hard constraints, a secondary analysis might be used to assess whether all applicable hard constraints are satisfied for a select set of possible traverses (chosen based on the traverse value optimization). This approach to traverse planning and optimization is computationally intensive, and requires a significant amount of knowledge about the environment. While these methods for traverse optimization may provide valuable insight in some cases, other cases will require traverse planning tools able to cope with higher levels of uncertainty in traverse goals or incomplete information about the environment.

5.2 Traverse Heuristics

An alternative approach to traverse optimization is to utilize heuristics to evaluate information and make traverse-related decisions. A heuristic can be defined as:

> A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood...heuristics do not guarantee optimal, or even feasible, solutions and are often used with no theoretical guarantee [Howe, 2001].

While heuristics may not guarantee optimal solutions, they can often yield near-optimal solutions with lower computational burden that strict optimization approaches. Humans commonly use heuristic based planning to evaluate how to get from one location to another, given a set of constraints. Typically, no integration or comprehensive optimization is performed, but past experience and "common sense" rules are used to guide traverse decisions. While

humans can benefit from computational approaches to traverse planning (especially in extreme environments, where margins between success and failure may be small), robotic explorers may benefit from a heuristic approach to traverse planning.

5.2.1 When are Heuristics useful?

Heuristics are especially useful in complex situations in which:

- Incomplete or uncertain data precludes use of optimization approaches,
- Computational complexity of other approaches is high,
- Assessment and decision making is time-critical, or
- Optimal solutions are not required, but good or near-optimal solutions are desired.

Heuristics can be descriptive (what is the situation) or prescriptive (what to do about the situation) and can (following adapted from [Rechtin, 1991], p.20):

- be reminders of past situations of a similar or analogous type,
- help evaluate architectural choices,
- act as sanity checks and first-order assumptions, and
- act as teaching aids.

Table 5-1 outlines the use of different design methodologies for problem solving and when each of them might be useful.

TABLE 5-1. Comparison of Design Methodologies

Parameter	**Nature of the Parameter (N=normative, R=rational, H=heuristic)**					
Value set	Subjective	N	Objective	R	Pragmatic	H
Reasoning	Inductive	N, H	Deductive	R		
Data base	Examples	N, H	Facts	R		
Nature of problem	Structured	N, R	Ill-structured	R, H	Wicked	H
Problem definition	Explicit	N	Implicit	R, H		
System goal	Optimize	N	Satisfy	R, H		
Nature of the rules	Rigid	N	Algorithmic	R	Contextual	H
Dependence on practitioner	Independent	N	Dependent	R, H		
Required expertise	Novice	N	Graduate	R	Expert	H
Problem-solving procedure	Instruction	N	Calculative	R	Deliberative	H
Evaluating multiple options	Poor	N	Good	R	Helpful	H
Handling sociopolitical issues	Poor	N, R	Helpful	H		

Reprinted from [Rechtin, 1991], p. 21, Table 1-1.

TABLE 5-1. Comparison of Design Methodologies

Parameter	Nature of the Parameter (N=normative, R=rational, H=heuristic)			
Validity and verification of solution	Assured	N, R	Limited	H
Replicability of solution	Exactly	N, R	Approximate	H

Reprinted from [Rechtin, 1991], p. 21, Table 1-1.

Based on Rechtin's guidelines, heuristics might be applied to the traverse design process when decisions are based on experience, or are ill-structured or inductive. Heuristics might also be useful for considering multiple traverse designs, or for traverses involving implicit goals and contextual constraints.

5.2.2 Heuristics for Traverse Planning

This section briefly mentions a few possible heuristics for traverse planning. Some of them are discussed in more detail in the examples later in the chapter. Most of the heuristics mentioned here are prescriptive, but often have descriptive components (e.g., this is the situation, and therefore this action should be taken).

Choosing an ordering of sites to visit during a traverse. *If current location is a safe haven (e.g. Mars central base), go to furthest site early in traverse when resources margins are highest.* Implicit in this heuristic is a need to return to the original location at the end of the traverse; obviously if this is not a goal, then this heuristic doesn't apply. This is a specific example of a more general survival-related heuristic that can be stated as: *Plan the traverse to end up (with high probability) in a condition with adequate resources for continued survival.*

Local direction of travel, part 1. *Go around reasonably sized topographic features along the traverse if want to save energy during traverse.* This heuristic is one example of a typical guide used by a human in planning a path from one location to another: features that would impose additional energy costs might generally be avoided, unless scaling the feature has particular value such as the possibility of an aesthetic view, some psychological benefit, or the provision of needed exercise.

Local direction of travel, part 2. *If traveling along features or across features is necessary, try to travel across features in a low topographic power direction.* This heuristic for the local direction of travel can be computed based on a digital elevation model using the height-height correlation function. The direction of preferred travel is given by the direction (for some set of chosen direction vectors) of the minimum of the height-height correlation function.

Overall direction of travel. *To get to a specific location, one must travel in the direction of the location.* This heuristic implies that the

local direction of travel heuristics must sometimes be violated when they are not consistent with the required overall general direction of travel. Deciding when to violate the local direction of travel heuristic is not a trivial matter: it is a complex trade-off to decide when a local deviation is no longer justified in light of the larger traverse goal.

Choosing a low-cost path. *Low-cost paths between two locations may involve traversing through one or more passes (low-points), if significant topography lies between the starting and desired ending locations.* A fairly stupid heuristic for finding a low cost path might involve choosing a bunch of random points between the initial and desired locations, computing the cost of traversing between each pair of nearby points, and computing the minimum cost path from the initial location to the destination using the set of random points as possible intermediate destinations. This is a heuristic because the choice of points is based on a rule of thumb: *to best route from A to B probably involves visiting locations between and around A and B.* The heuristic implementation (see *traverse_compute_min_cost.m* in Appendix D) requires an assumption about the traverse velocity in order to estimate the cost for a segment of the traverse. Other, smarter implementations of this heuristic are certainly possible.

Surface visibility along the traverse. *It is valuable to have big views of the surface along the traverse for line-of-sight communication, spotting navigation landmarks, or for aesthetic views.*

Visibility of points of interest. *It is valuable to view features or points of interest at a variety of scales (e.g., from close up and from far away).Nearby features are easier to see in detail than far away features; value of observing small features should be weighted inversely with the distance at which the feature is observed.*

5.2.3 Limitations of Heuristics

As previously stated, heuristics don't necessarily guarantee an optimal or near-optimal solution (or even a solution consistent with applicable constraints). These and other limitations suggest that non-heuristic approaches to traverse planning should be used where possible, but should be supplemented with heuristics in order to build a process that does not overly burden traverse planners (the agents of a distributed system, in many cases, might be the traverse planners).

5.2.4 Normative Approach

Normative (rule-based) approaches have formed the basis for many types of traverse planning (for example: Apollo lunar surface exploration, pilot flight planning under instrument flight rules) have been used successfully, but during operations these normative

approaches are typically relaxed under certain context. For example, Federal Aviation Regulations tend to be normative, but also can be relaxed in the context of an emergency situation: "the pilot in command of an aircraft is directly responsible for, and is the final authority as to the safe operation of that aircraft [FAR/AIM, 2001]." The more complex the environment, the more impractical exact application of normative approaches becomes. Even when normative approaches are used, heuristics often supplement the normative approach under some conditions.

5.3 Traverse Planning

In most cases, "the name of the game in traverse planning is maximum science return [Muehlberger, 1981]." Traverse planning considerations for Apollo lunar surface traverses included the development of a photo-mosaic and topographic base map, and the identification of potential geological stations for geological sampling, rapid deployment of experiments, and photographic documentation. As part of this process, a "system of priorities was established for both station locations and task performed at each station. [Muehlberger, 1981]" For planetary surface exploration, other considerations may include aesthetic beauty or an opportunity for privacy.

Traverse planning for current missions is largely an activity pursued by Earth-bound scientists and mission controllers. Interviews of the Apollo lunar surface astronauts [Conners et al., 1994] suggest that during future long-duration planetary exploration, explorers will take "a far more active role...in planning and executing their activities." The lunar astronauts also highlighted the need for more flexibility in the traverse planning and execution process, suggesting that less rigid timing and sequencing of activities would be beneficial during long-duration surface missions. Robotic explorers may also benefit from autonomous traverse planning by reducing the delay between different activities, potentially increasing net science return.

Traverse planning in general should include identification of geologic sites and activities of interest (or identification of other traverse goals), and the development of a plan for how to get to and from those sites of interest, including point to point navigation, resource usage, and timing of activities to match appropriate conditions (e.g., sun conditions that provide adequate illumination or thermoregulation, or low atmospheric dust conditions). Other constraints must also be considered.

5.3.1 Process for Traverse Planning

The traverse planning process presented here is not an all-inclusive traverse planning process, but can serve as a starting point for traverse planning. It consists of a structured application of analysis tools and heuristics developed in this thesis, and can certainly be modified and improved. In some steps, optimization approaches are considered, but no attempt at overall traverse optimization is attempted: what is "optimal" is highly dependent upon the relative weighing applied to measures of traverse value and cost.

To apply the traverse planning process:

1. *Evaluate path independent surface conditions and accessibility*: Determine the area of the surface accessible to the explorer or explorers based on path independent accessibility criteria. An *accessibility* or *reachability* map (Section 3.3.3) might be generated based on surface data including slope, thermal inertia (used as a dust prevalence indicator), rock distribution predictions and applicable flight rules (e.g., must stay on slopes less than X degrees). Restricted areas can be incorporated into an accessibility map (e.g., areas with sensitive or potentially dangerous equipment in operation or experiments underway should be avoided). Another accessibility map might be defined by the region where the explorer will have the mandated communications capabilities or access to specific network services (can be computed using the surface coverage algorithm discussed in Section 3.3.2). Accessibility maps can be combined with appropriate logic operations (for example, accessible area has slopes of less than X degrees and does not include a Y meter radius zone centered on the high-power uplink antenna).
2. *Identify sites and activities of interest*: Develop a set of traverse goals including sites of interest and activities required or desired at each site. Under resource or time constrained situations, a system of priorities should be developed for sites and activities. If a site of interest is outside the region of path independent surface accessibility, either reallocate the site to another explorer, modify the capabilities of the explorer (e.g., with tools or other aids), or change the accessibility criteria. Selection of sites and activities should be coordinated with other elements of the exploration system.
3. *Identify initial possible traverse(s)*. There may exist a "natural" or desired ordering of the sites of interest based on science priorities or other considerations. Flight rules may prescribe that the furthest site of interest occur relatively early in the traverse while margins of consumable resources are high. The actual traverse path between points can be computed by considering multiple points along possible traverses, applying an energy expenditure and resource consumption model to compute cost functions for each possible traverse leg, and then applying a shortest (minimum cost) path algorithm to find the minimum

cost path between two points. Note that this requires some time-based parameterization of the traverse: a desired traverse velocity, or desired rate of energy expenditures may need to be specified. A traverse from one location to another may also be based on heuristic information. The directional dependence of surface topography statistics such as the height-height correlation function (see Section 3.2.4) can provide a heuristic for a preferred direction of travel (e.g., if crossing a dune field, this heuristic would suggest travel parallel to dune crests).

4. *Evaluate path dependent surface conditions and accessibility*: Evaluate surface visibility along the route: are features of interest visible, and how well? An overall visibility-score heuristic might be developed to characterize some desired visibility characteristic (e.g., expansive views). Based on the selected traverse(s), evaluate the expected lighting conditions of the surface as a function of time and position to consider the required levels of illumination (and relative direction of the illumination source), the heat balance and any applicable thermoregulation issues, or other light-related constraints. In some cases, locations and/or times associated with adverse weather conditions or other dynamic hazards might also be considered (e.g., likelihood of high wind conditions, blowing dust or dust devils).
5. *Perform flight rule validation*: The proposed traverse(s) should be evaluated with respect to the applicable flight rules. Possible flight rules might include walkback restrictions (e.g., for human explorers, in case of a rover breakdown at a remote site), required resource margins upon traverse completion (power, oxygen, etc.). These flight rules typically consider management of off-nominal events, and might include constraints designed to enable self-rescue or assisted rescue of a disable explorer. Many of the Apollo lunar surface mission constraints involve position, orientation, timing, and environmental exposure [Nute et al., 1970].
6. *Decide whether to modify or accept the traverse plan*: Modify the planned traverse and repeat the process, or accept the planned traverse if it meets the required criteria and no further optimization is necessary. This decision may include results of the next step in the process: approval for acceptance or rejection of the traverse plan may be sought from some other authority such as other explorers or some central planning authority.
7. *Communicate the traverse plan*: Share relevant details of the traverse plan with other elements of the exploration system to enable coordinated activities.

This process could be largely automated, and could be implemented to allow specification of the planned traverse at various levels of fidelity (from "I want to go out, walk over there, look around for a half-hour, and come back" to a detailed Apollo-style traverse plan). Even though many parts of the proposed traverse planning process are computationally intensive, the interface between the traverse

planning process and the traverse planner could be made very simple (computational aspects could be automated, while key design decisions could be exposed across the interface).

From the traverse planners perspective, the process might be analogous to filing a flight plan prior to flying an aircraft under instrument flight rules: calculations ensure that the aircraft will handle appropriately during all phases of flight, a desired routing is identified (requested by the pilot, then accepted or amended by Air Traffic Control authorities). To coordinate aircraft traffic, time restrictions are assigned along with the routing clearance (similarly, spatial or temporal restrictions might apply to the traverse planning process when coordinating activities between explorers - in some cases, no such restrictions need apply).

Such a process is already used in one form during Antarctic exploration: "individuals at U.S. Antarctic stations must file a 'foot plan' (similar to a pilot's flight plan) when they intend to leave a station, even for a relatively brief excursion [Stuster, 1996]." Stuster also suggests that "predictability is the key to enhancing intracrew communication and group task performance."

Developing and using a simple and consistent traverse planning process, and communicating that plan in some standard form to other elements of the exploration system, can help explorers achieve coordination between system elements and enhance explorer safety. Distributed architectures provide the mechanisms for delivery of information required for the traverse planning process, and to enable coordination between explorers by supporting the dissemination of the results of the traverse planning process.

5.4 Examples

The following examples demonstrate how distributed architectures can support individual human or robotic explorers during traverse operations, including how to apply the traverse planning process described in the previous section.

5.4.1 Geologists Trekking in the Mojave Desert

This example illustrates a simple implementation of the minimum cost traverse heuristic and demonstrates how a cost of system reconfiguration can be computed. In this example, a group of ten geologists have just finished assembling a system that will monitor temperature, pressure, humidity, and wind speed over an area of the Soda Lake dry lakebed in the Mojave desert. The geologists communicate on their radios, and decide to spend the afternoon traversing NE to an area near the base of Old Dad Mountain, a distance of

about 12 km. Traverse planning for this simple example consists of answering the questions:

- What route should each individual take?
- What is the metabolic cost of the traverse (how many more calories should I eat tonight...)?

For this example, the only constraint considered is that each traverse route should minimize metabolic cost (clearly, this is not a realistic assumption for ten geologists enjoying themselves in the mojave desert, but the goal here is to illustrate the heuristic).

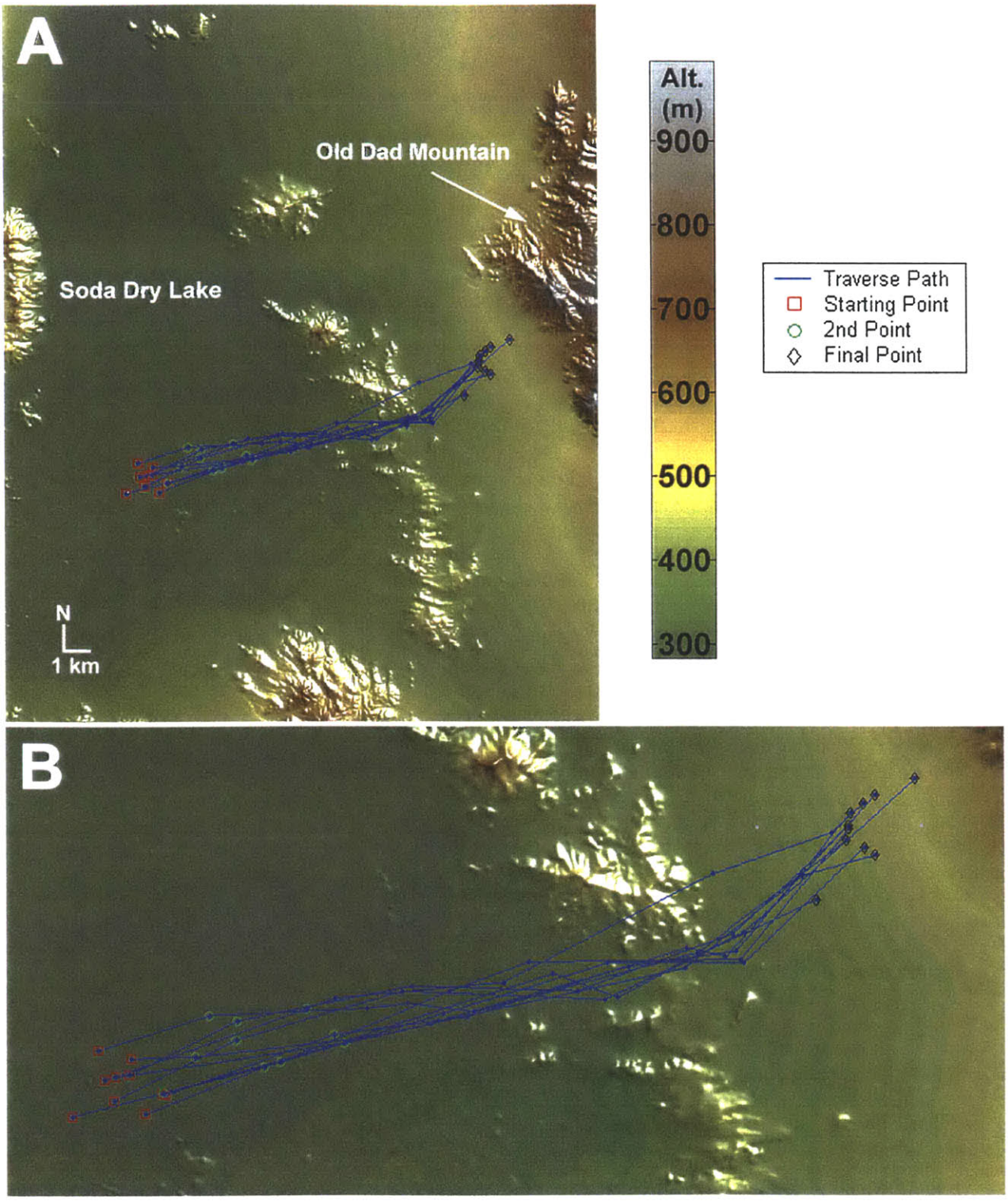
The minimum metabolic cost route is computed using the heuristic for choosing a low-cost path, described in Section 5.2.2 (see the function *traverse_compute_min_cost.m* in Appendix D). It is assumed that each geologist masses 80 kg (body mass, clothes, and equipment). Traverse velocity is assumed to be 0.5 m/s (the geologists want to take their time along the traverse).

Figure 5-10 plots the "optimum" traverses for each geologist, and demonstrates that the minimum cost traverse heuristic does not generate optimal or near-optimal solutions all the time (one of the traverses crosses the ridge of a small mountain). Based on the load-carrying model as applied to the "minimum cost" traverses, the total cost of "system reconfiguration" (the sum of the metabolic cost of all the geologists) is 1.1×10^4 kCal (4.6×10^7 J), for an average metabolic cost per geologist of about 1100 kCal (4.6×10^6 J).In this particular situation, the energy cost of thermoregulation and other field geology activities are likely to be significant: therefore, this computed cost of system reconfiguration is likely to be small compare to the actual energy cost of system reconfiguration.

Figure 5-11 illustrates how the minimum cost traverse algorithm generates a single "minimum cost" traverse. The minimum cost traverse algorithm works by generating a set of trial point (points that might be on or near the optimal traverse) and then computing the cost of traversing between nearby pairs of points.

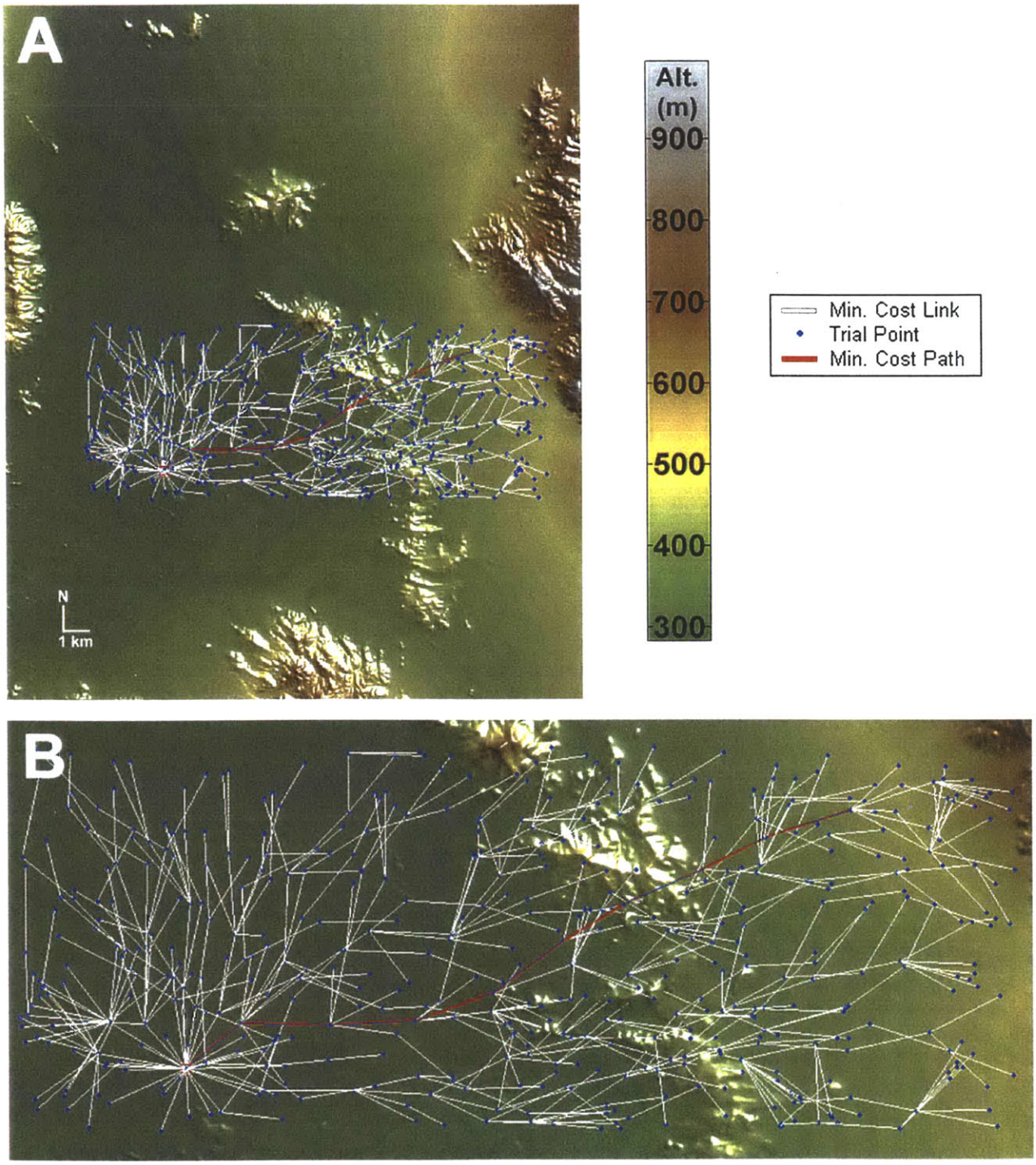
Next, Dijkstra's shortest (minimum cost) path algorithm is used to generate a spanning tree of the trial points where the path to each trial point from the initial position represents the minimum cost path between the two points (white lines in Figure 5-11). The minimal cost path from the initial point to the destination point (red line) is then extracted from the tree (the minimum cost paths to the trial points are necessarily generated as a by-product of computing the minimum cost path from the initial to the final location).

The minimum cost path algorithm is extremely basic, and could be improved in several ways. One key improvement would be to select the trial points in a more intelligent fashion. Once an initial "mini



mum cost" traverse is found, the trial points in the minimum cost traverse could be perturbed to search for a lower cost traverse, or the minimum cost traverse algorithm could be applied to segments

FIGURE 5-10. "Minimum cost" traverses from Soda Dry Lake to the base of Old Dad Mountain in the Mojave Desert.



of the first-pass minimum cost traverse to iteratively refine the traverse.

While this sort of traverse planning is certainly overkill in the situation presented here, a similar traverse planning strategy might be

FIGURE 5-11. The "minimum cost" traverse algorithm generates points on or near the minimum cost traverse, computes costs between nearby points, and finds the minimal cost path to each point.

very useful in highly energy constrained situations where a near-minimum cost traverse is much less obvious that in this simple situation. An extension to this example might consider the (likely) use of off-road vehicles and the gains in efficiency possible through the use of alternative means of transportation. The same process demonstrated here can be used to generate "minimum cost" traverses for robotic agents, given the appropriate energy expenditure model.

5.4.2 Modeling an Apollo Traverse

This example applies the traverse planning process to the second lunar surface extravehicular activity of the Apollo 14 mission using approximations to the planned and actual Apollo 14 traverses. The traverse planning process is applied to the planned and actual traverses. A slope restriction is generated for the area, visibility is analyzed along the traverse, and results of applying the load-carrying model to the actual traverse is compared to actual metabolic cost data from Apollo 14. Finally, it is considered how distributed systems might have contributed to the Apollo lunar surface missions.

Apollo 14 digital elevation model. To serve as the foundation for this analysis, a digital elevation model of the Apollo 14 lunar surface area was constructed (see Chapter 4, page 159).

Evaluate path independent surface conditions and accessibility. The main constraint examined here is the slope constraint imposed on the Apollo 14 lunar surface astronauts. Set by the maximum slope at which an astronaut would be able to safely navigate a slope unassisted, the slope constraint was set to [0 15] degrees [Nute, 1970] (all slopes are considered positive for the purpose of this constraint). A reachability map can be generated based on this constraint.

Identify sites and activities of interest. Planned and actual sites of interest (geological stations and stops) were extracted from a graphical analysis of the Apollo 14 long traverse by Lennie Waugh [Jones, 2000], reprinted in two parts as Figure 5-12 and Figure 5-13. Positions on this oblique-view mosaic were manually correlated with a georeferenced, scanned version of a topographic and aerial map of the Fra Mauro Highlands [Swann et al., Plate 2].

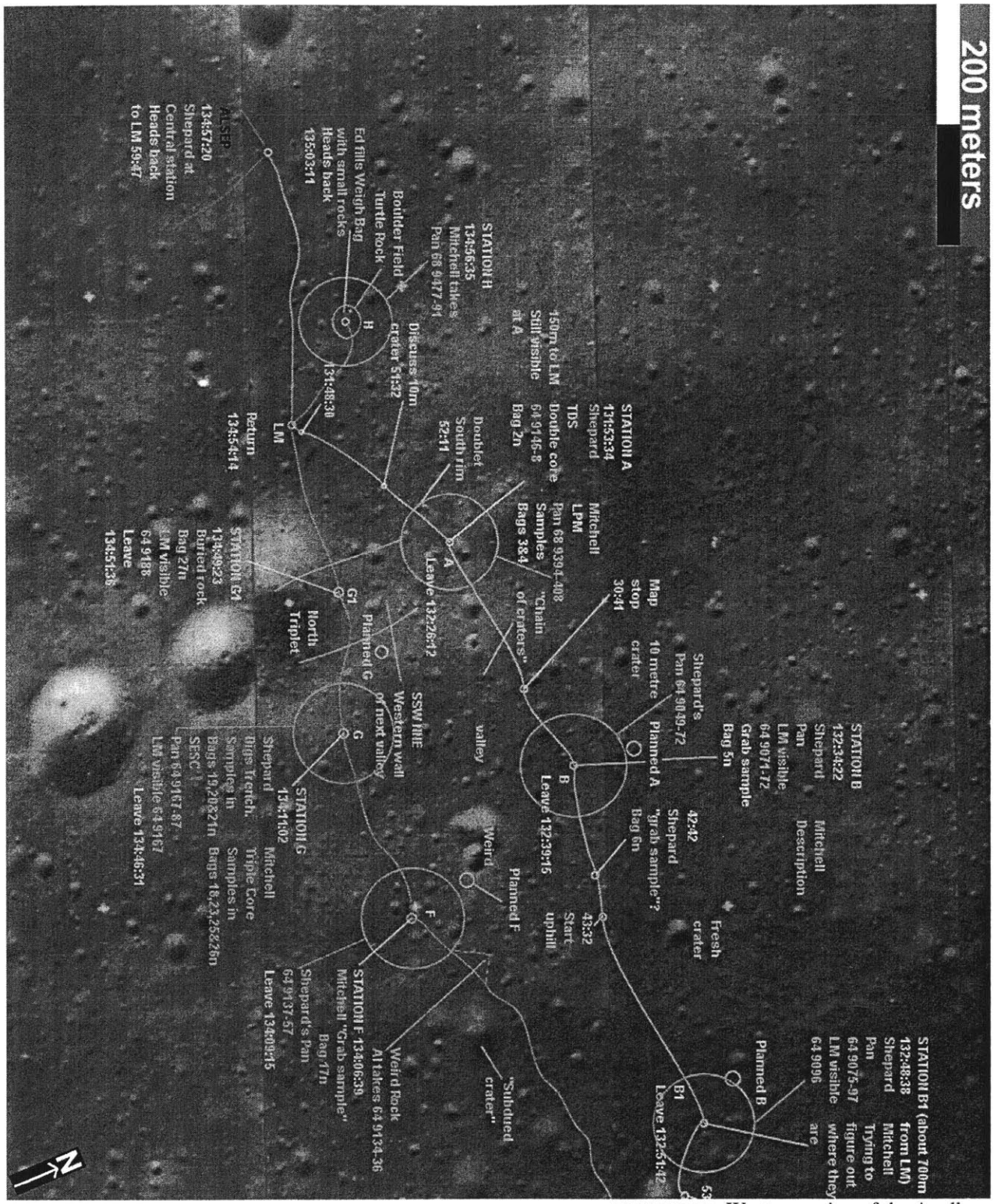


FIGURE 5-12. Western region of the Apollo 14 second traverse. From [Jones, 2001]; graphical reconstruction by Lennie Waugh.

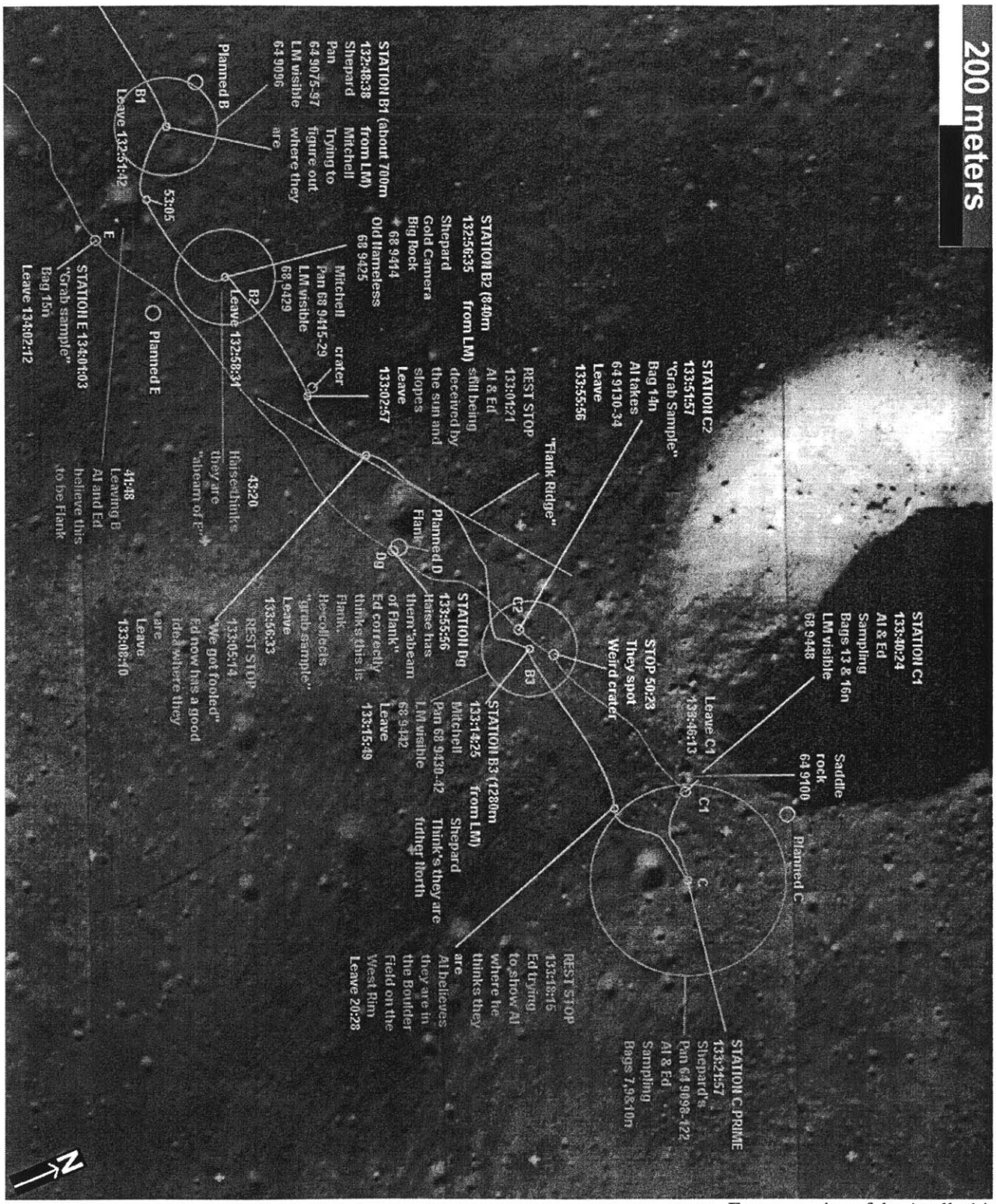


FIGURE 5-13. Eastern region of the Apollo 14 second traverse. From [Jones, 2001]; graphical reconstruction by Lennie Waugh.

Initial Traverse(s). Figure 5-14 illustrates the planned (A) and actual (B) traverses. Time at each geologic station (point of interest) in the planned traverse was carefully considered, and an elaborate system of priorities for geological stations, as well as individual activities at each station, was developed prior to the Apollo long traverse. The structure of the planned traverse (out to cone crater and back) drove the ordering of sites of interest, as did the equipment that the two lunar surface astronauts were to have with them during different stages of the traverse.

While the planned traverse (Figure 5-14 A) crosses a restricted area (an area with slopes > 15 degrees), the paths between waypoints are straight-line segments for convenience, and small areas of steep slopes such as this one could easily be avoided. The actual traverse diverts around some obstacles, including the area of steep slopes. Because of the short duration and distance of this traverse, the main factor driving the traverse optimization process is not energy efficiency, but time.

It should also be noted that relatively steep slopes are probably underrepresented by the Apollo 14 digital elevation model, because of (1) a lack of craters (only some of the more major craters were included in the digital elevation model, and with limited accuracy) and (2) a digital elevation model with reduced high frequency components (few traces of the characteristic "undulations" of the lunar surface were likely captured in the creation of the digital elevation model due to their relatively small amplitude as compared to the 10 meter contour intervals used as the basis for the digital elevation model.

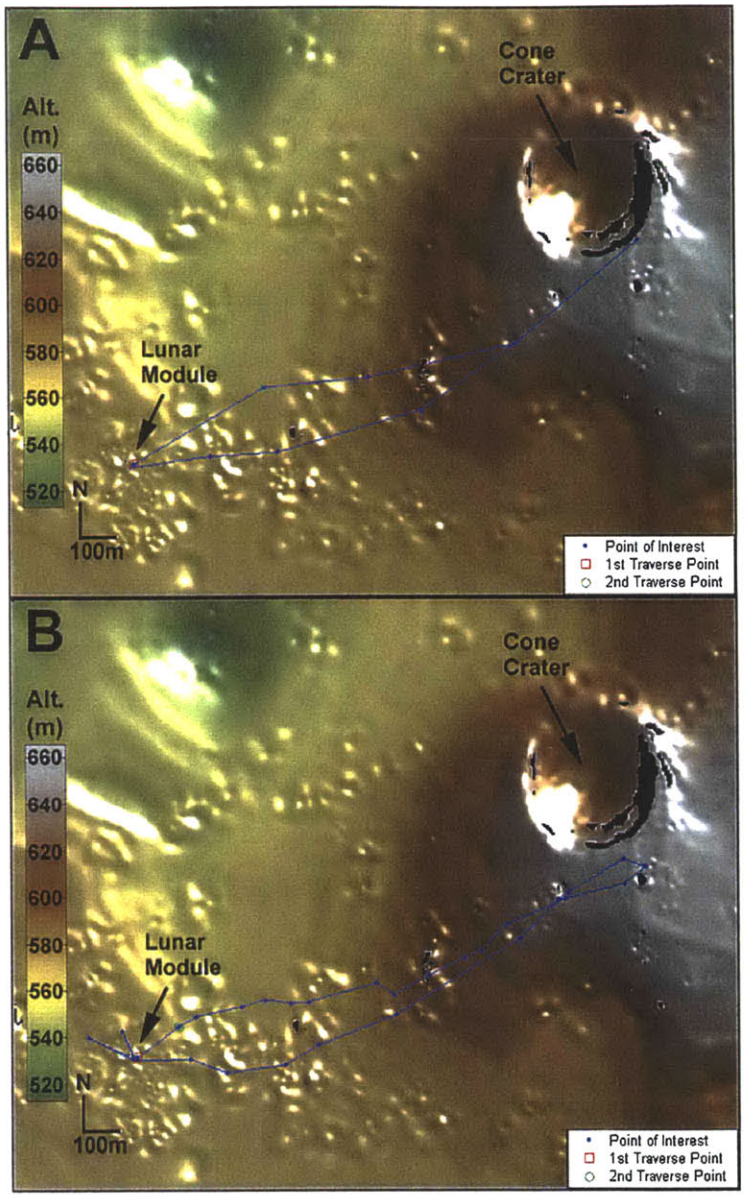


FIGURE 5-14. Planned (A) and actual (B) traverses for the Apollo 14 second extravehicular activity. Prohibited areas (based on a slope constraint of [0 15] degrees) are shown in black.

Path dependent surface conditions. The path dependent surface conditions analyzed here include surface visibility and slope along the traverse path, and overall metabolic cost of the traverse legs between sites of interest.

Overall visibility of the lunar surface and visibility of the lunar surface as a function of time was assessed for the planned and actual traverses. Temporal sampling of each traverse allows visibility analysis to provide a spatial and temporal assessment of surface visibility; near parts of a visible surface are assigned a higher visibility score, but far parts of a surface can be assigned high visibility scores if they are frequently visible.

The overall visibility plots in Figure 5-15 represent one of the challenges with navigation on the moon - the craters frequently block visibility of the surface both nearby and far away (as evidenced by the presence of "visibility holes" - surface locations not seen anywhere along a traverse - both near and far from the traverses).

Time lapse analysis of the planned and actual traverses demonstrates a lack of forward visibility during the climb up the southwest flank of Cone Crater, and a general lack of visibility in all directions to the south of the Cone Crater rim. It was in this area (see Figure 5-13) that the two Apollo 14 lunar surface astronauts encountered difficulties in determining their position accurately.

Only right at the geological station on the south rim of Cone Crater (for the planned traverse) is the interior of Cone Crater clearly visible. This kind of visibility analysis as part of the traverse planning process might have led to changes in the proposed Apollo 14 traverse.

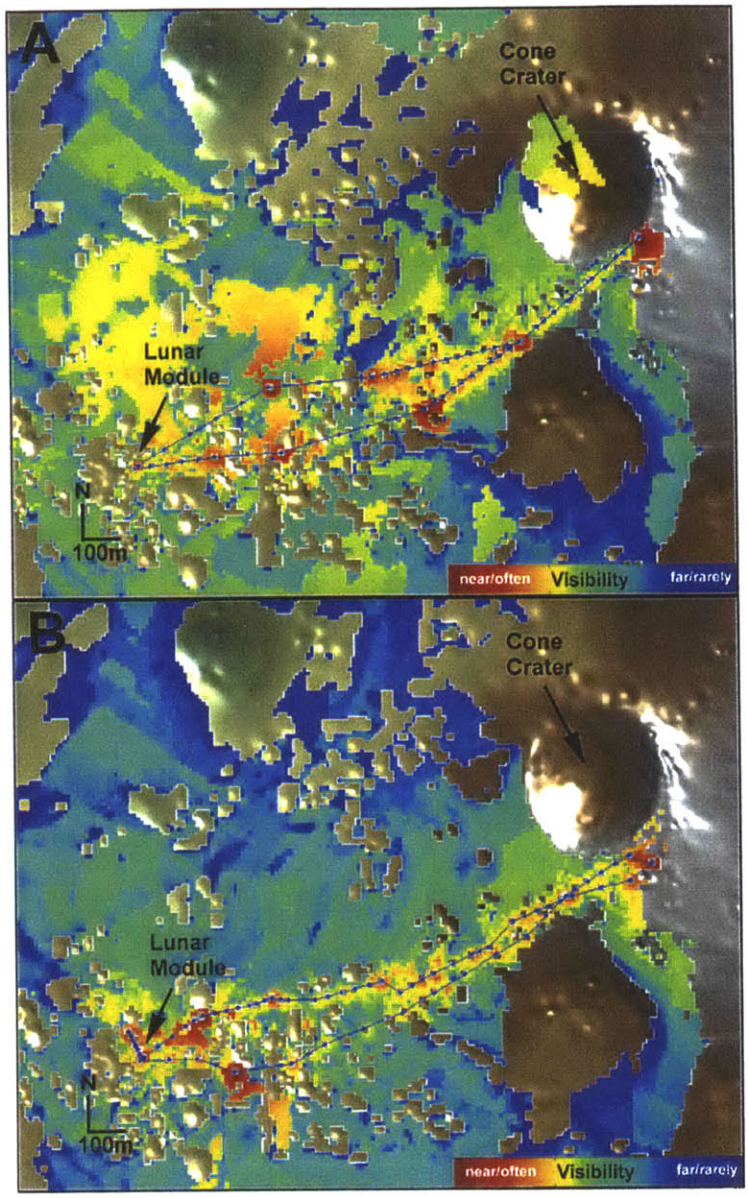


FIGURE 5-15. Surface visibility along the planned (A) and actual (B) long traverse during Apollo 14. Visibility is coded spatially and temporally: traverses were sampled at a temporal interval of 5 minutes, and visibility results at each sampled point were additively combined. For each sampled point, nearby surfaces are assigned a higher value (brighter color). Visibility is computed in all directions, using a height above the surface of 1.8m (a standing astronaut). Note that the interior of Cone Crater is only briefly visible for the planned traverse, and never visible for the actual traverse.

Another important path dependent surface characteristic that needs to be verified is surface slope. To verify the slope requirement, slope statistics were computed for both planned and actual traverses. Figure 5-16 illustrates the slope distribution when the two traverses are spatially sampled, while Figure 5-17 illustrates the slope distribution when the two traverses are temporally sampled.

The spatial slope distributions of the planned and actual traverses are statistically very similar - perhaps the actual traverse distribution is skewed slightly toward gentler slopes, but the use of straight-line paths between points of interest for both traverses suggests that one should disregard such slight differences.

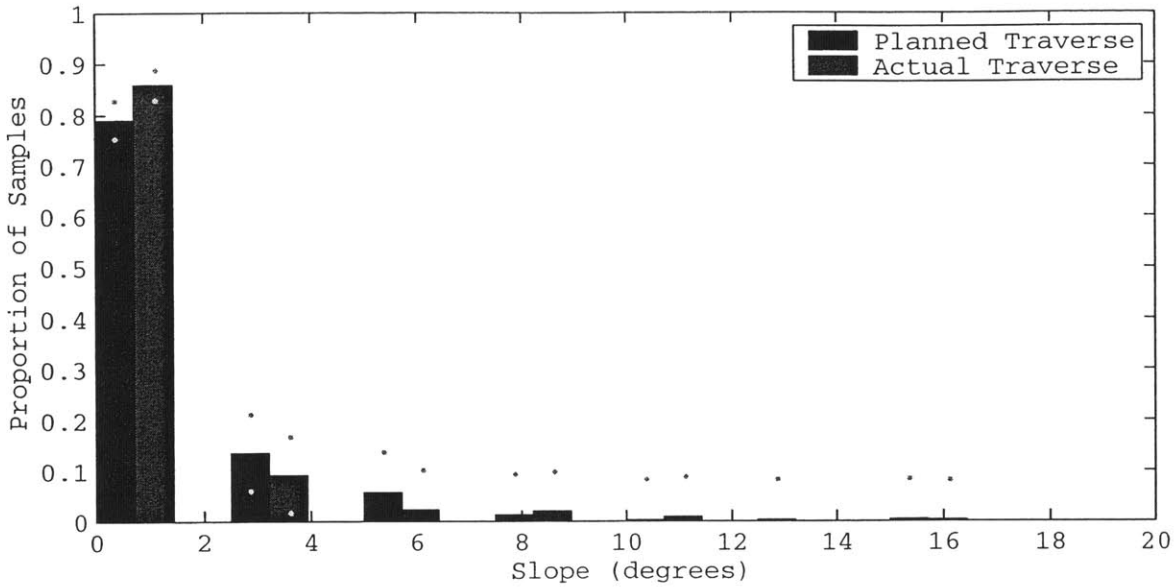


FIGURE 5-16. Slope distributions for the planned and actual Apollo 14 long traverses (traverses spatially sampled every 5 m). Dots indicate 95% confidence interval boundaries.

Differences in temporal slope distributions arise because more time than planned was spent at points of interest or rest stops; the lunar surface astronauts were able to quickly travel from point to point, a scheme that is consistent with the ability of the lunar surface astronauts to achieve a relatively efficient and high-speed loping gait.

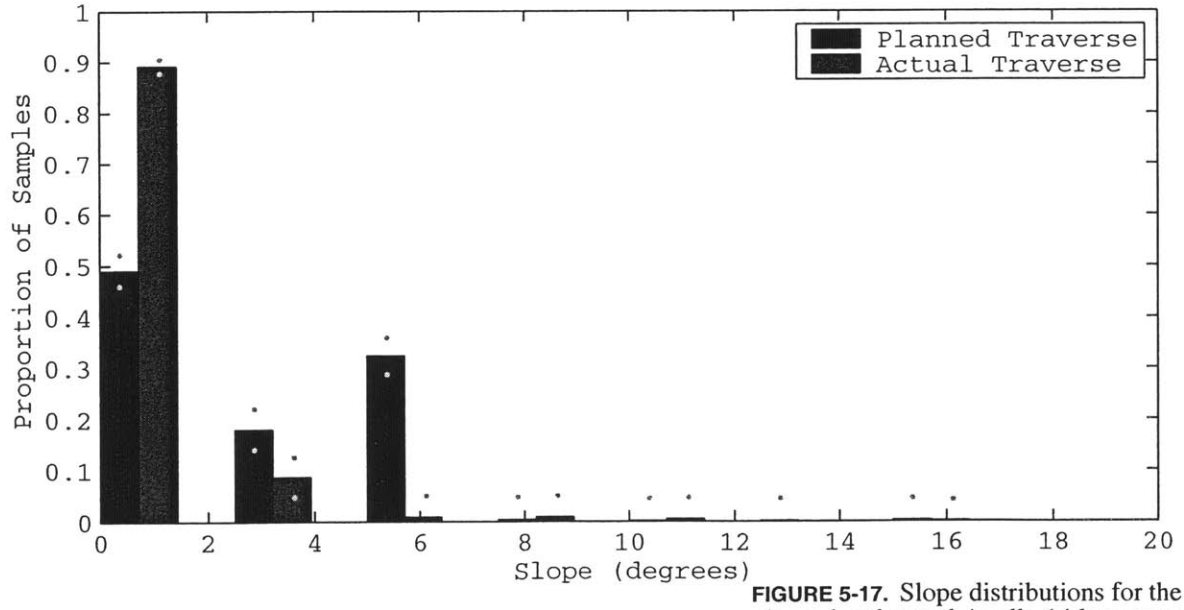


FIGURE 5-17. Slope distributions for the planned and actual Apollo 14 long traverses (traverses temporally sampled every 5 s). Dots indicate 95% confidence interval boundaries.

To test this hypothesis, the metabolic cost load-carrying model can be applied to the actual traverses for the Commander and Lunar Module Pilot, and compared to actual metabolic cost estimates

from the Apollo mission. Metabolic cost estimates are given by Waligora in [Johnston, 1974] only for the specific periods actually spent traversing from one geological station to another. These specific periods were extracted from the traverse and sampled with a temporal interval of 30 seconds. Astronauts were assumed to weigh 74 kg (average of pre- and post-flight means from [Johnston, 1974]), and each suit and portable life support system weighed approximately 48 kg. Table 5-2 summarizes the results of applying the load carrying model to each of these extracted, interpolated legs of the traverse.

TABLE 5-2. Comparison of Metabolic Cost Results

Traverse Parameter	Waligora Results*	Load Carrying Model Results
Duration	72.8 min	76 min
Energy Expenditure	1495 kJ (Commander)	1318 kJ (Both mission roles)
	1599 kJ (Lunar Module Pilot)	

***See [Johnston, 1974], p. 125.**

The model tended to under-predict energy consumption at low velocities on high slopes, and over-predict for high velocities on high negative slopes. This is consistent with the hypothesis that space suit stiffness increases workload during ascent while reducing the relative burden of energy absorption put on muscles during rapid descents (unpublished work, Carr, 2001). Further data at high velocities is necessary to evaluate the loping hypothesis previously mentioned.

No consideration was made in this simple analysis for equipment carried with the astronauts (such as the Mobile Equipment Transporter); this is likely to be a primary factor in the difference in metabolic cost estimates between the Apollo 14 and load carrying model estimates.

Flight rule validation. Many additional flight rules existed for the Apollo 14 mission. None will be discussed here for the sake of brevity, other than to mention that walkback restrictions could be assessed dynamically based on available resource and the generation of a minimum cost path from the current location to the "safe haven." This might allow maximization of science return (by extending the allowable work time) without sacrificing crew safety. Clearly, extensive validation of the walkback traverse algorithm would be required before such a system would be entrusted with the life of an astronaut and the potential success of a mission.

Modifying the planned traverse. One possible modification to the Apollo 14 planned traverse is investigated here.

Instead of trying to sample at the south-east rim of cone crater, it might have been more desirable to location the cone crater geological station on the south-west rim of the crater.

Visibility analysis of this new Cone Crater sampling site is shown in Figure 5-18. This new sampling site is beneficial for several reasons:

- It provides a shorter total traverse to the rim of Cone Crater, with slightly less altitude gain.
- Surface visibility during the traverse to the crater rim is greatly enhanced.
- A fuller view of the interior of Cone Crater is provided, while also providing a view of the area surrounding the Lunar Module in the valley below.

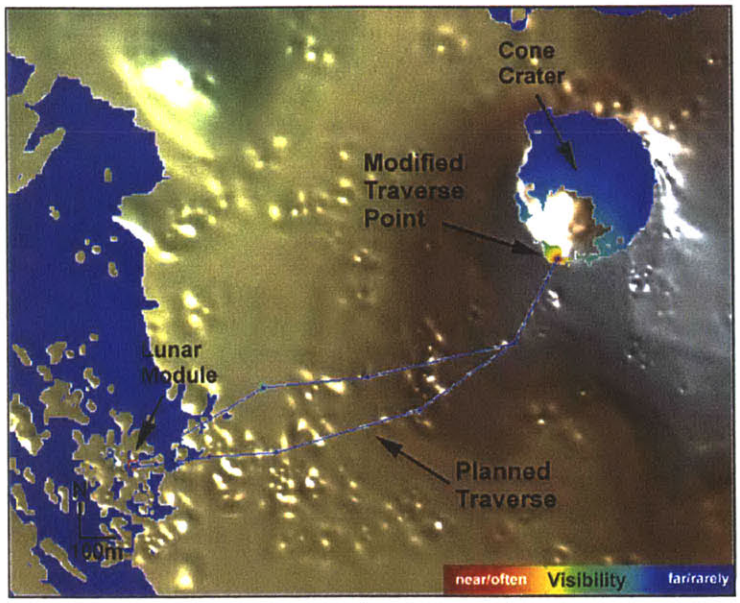


FIGURE 5-18. Visibility analysis from a new possible Cone Crater sampling site: the furthest geological station has been moved to a position on the rim of Cone Crater that allows excellent visibility into the crater and of the area of the Lunar Module to the west. The approach to the rim also provides enhanced visibility and a better sun angle, along with a shorter traverse, as compared to the original planned Cone Crater sampling site.

The approach direction (with respect to the Sun) to the sampling site is also superior: during the Apollo 14 second extravehicular activity, the Sun was basically to the East, and walking primarily north towards the rim of cone crater (as if heading to the new sampling site) would be in the cross-sun direction. This is preferable to the glare and shadowing problems encountered during the attempted traverse to the rim of Cone Crater and back in up-sun or down-sun directions.

Potential uses of distributed systems. Many of the problems encountered during the Apollo 14 second traverse, and many of the similar problems encountered during the other Apollo missions, could easily have been dealt with if the lunar surface astronauts had had access to the right data or a different perspective: distributed systems have the potential to deliver this data or to provide the necessary perspective.

A distributed positioning or direction finding system might have been especially useful during the navigation problems of Apollo 14 (later missions were more successful thanks to the inertial navigation system of the Lunar Roving Vehicle): Radio beacons could have been set out on high points near the Lunar Module or along the traverse - these beacons could also have served as science stations (for example, forming a seismometer network, for sensing cratering events) and could have provided a redundant low bandwidth communications relay from the lunar surface astronauts to the Lunar Module. Given the right hardware and software, setting up this type of network might be similar in effort to the climber who marks his trail by placing bamboo wands in the snow.

Distributed systems might also have been useful to thoroughly examine the lunar surface: as evidenced by the visibility analysis, many areas of the lunar surface even near the Lunar Module and the long traverse, were not visible to the two astronauts. When science goals would benefit from similar data types collected over a large spatial area (for example, studying spatial distribution of soil mechanical properties, reflectivity, electrical properties, temperature, radiation exposure, etc.) clusters of small sensor pods could be produced and easily delivered to an inaccessible site with the flick of a space-suited arm. Mobile agents could provide alternative visual perspectives: this type of information sharing is discussed in the next example in the context of field geology. Mobile agents might also be sent into steep walled craters or other inaccessible or potentially dangerous sites.

Given the stringent constraints of the Apollo missions, both in terms of time and available technology, the Apollo traverses, including the Apollo 14 long traverse, were well planned and effectively executed. Once on the lunar surface, the Apollo lunar surface astronauts were able to accomplish all major mission goals of Project Apollo. However, distributed systems may be able to make future lunar explorers even more effective.

5.4.3 Distributed Field Geology

Chapter 4 discusses the potential benefits of utilizing a distributed system during field geology: short-range wireless networking between members of a field team can enable team coordination and collaboration while supporting such activities as *distributed geology* (see page 142). While for the purposes of this example the agents of the geologic mapping team are human geologists, the approach taken here might certainly be applicable to a group of coordinating robotic explorers, or a group of human and robotic explorers. This example does not explicitly apply the traverse planning process (many aspects of the process were applied in Section 4.1). The goal here is assessing whether line-of-sight connectivity between different members of a field geology team (e.g., group of agents) is adequate for a short-range wireless communication system to be useful during field geology. But first, the direction-of-travel heuristic, based on the height-height correlation function, is compared to the direction of travel during actual field geology traverses.

Direction of Travel Heuristic. The preferred direction of travel heuristic was computed for the Cottonwood, Nevada quadrangle (the field area for the geologic mapping project and analysis of Chapter 4). The preferred direction of travel heuristic has been computed for an approximate scale length of 150 m in the directions [0 11 27 38 45 53 63 78 90] degrees relative to North-South, and is plotted in Figure 5-19. The heuristic seems to approximately

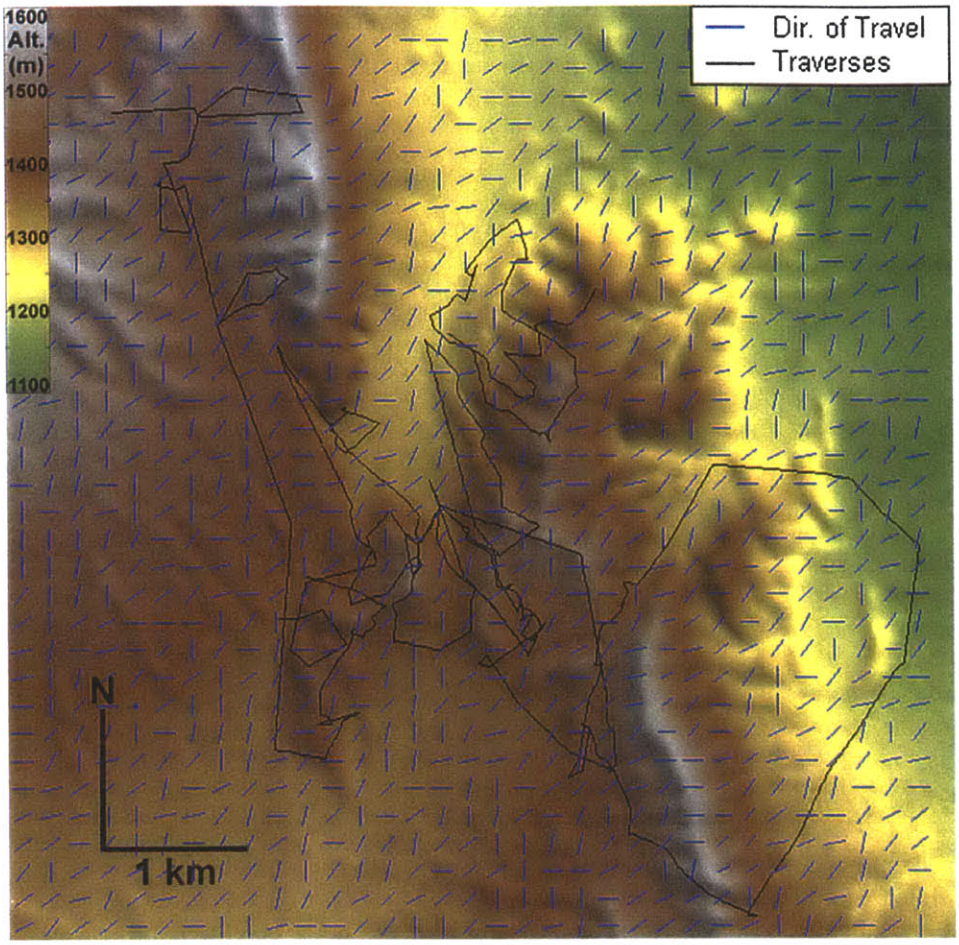


FIGURE 5-19. Comparison of the preferred direction of travel heuristic (short blue lines) and actual field geology traverses (black lines) in the Bird Spring Mountains, Nevada.

agree with the traverse directions in many cases, and especially for longer traverse segments (e.g., a part of a traverse generally in a similar direction). This suggests that the preferred direction of travel heuristic is generally more applicable to long traverses from location to location, rather than a specific approach to a local site of interest. The logic might be as follows: if the path to some desired destination is short, expending additional energy for a short time (traversing in a direction of relatively higher topographic power) is acceptable. On the other hand, if the path to some desired destination is long, choosing a lower-energy path is relatively more important.

Stochastic modeling of a geologic mapping team. During the author's January field geology experience (see Section 4.1), a general geologic mapping team consisted of up to four or five people. Generally members of a mapping team stayed within a couple of hundred meters of one another, but sometimes geologic mapping was done in pairs (the group might split in two) or individuals would map alone for a time.

A simple approach to modeling the movements of the geologic mapping team is considered here. First, traverses from the geologic mapping project in the Bird Spring Mountains are spatially sampled with a sampling interval of 30 m (the spatial resolution of the Cottonwood digital elevation model). For each sampled point, N sets of M random positions (representing N possible configurations of M individual members of the mapping team) are generated from a symmetric gaussian distribution with standard deviation σ, centered on the nominal traverse path.

The dispersion of these positions from the nominal traverse point is based upon the nominal operating range for Bluetooth™ devices [Bluetooth, 2001]. Preliminary subjective field tests with two Bluetooth™ equipped laptops suggest that high power (100 mW radiated power) Bluetooth devices may achieve reliable communication at distances in the field up to 100 m or beyond but performance is highly sensitive to line-of-sight conditions. For longer distances, the 802.11-based devices may need to be used, but similar line-of-sight restrictions may apply (see [IEEE802.11, 2001]).

To examine distributed system connectivity and performance as the mapping team becomes more spatially distributed, the standard deviation has been chosen to be $\sigma = \{30, 60, 90, 120, 150\}$ m.

Effective antenna height above the surface for these devices is assumed to be 2 meters, and the maximum theoretical transmission distance is assumed to be 500 m (beyond Bluetooth™ capabilities but certainly possible for 802.11-based devices). For purposes of comparison, the nominal transmission distance is assumed to be 100 m, and the space loss exponent is taken to be 2. For all results, a team of $M = 5$ individuals (nodes) is assumed, and $N = 10$ trials are performed for each experimental condition (a given traverse point and spatial standard deviation of the group).

Three metrics are used to characterize the distributed system:

- The mean number of reachable (neighbor) nodes, from the perspective of each node (for purposes of simplicity, all reachable nodes from a given node will be referred to as neighbors, even if no direct line-of-sight exists).
- The group-average of the mean cost of message delivery to each agent.
- The mean number of disconnected (isolated) nodes.

To generate these statistics, line-of-sight visibility was first calculated between the nodes to form a connectivity graph. Next, for each node in the graph, a shortest (minimum cost) path spanning tree was constructed for the set of nodes connected to each individual node. The number of nodes in this tree gives the number of

reachable neighbor nodes. If the resulting tree contains only one node, then that node is disconnected. The cost of each edge was weighted by the square of the ratio of the actual distance to the nominal distance between nodes, following the argument on page 109 in Chapter 3. The minimum cost path tree can therefore be used to compute a mean minimum cost of message delivery from reachable nodes to the node under consideration. These statistics are therefore easily computable for each node, and are then averaged over the group of M nodes.

Results. These metrics were computed for 10 trials each of 3 dispersion distributions at 1744 traverse points for 52320 total trials. Figure 5-20 plots results for the mean number of neighbors, and demonstrates that as might be expected, the distribution for the mean number of neighbors shifts towards fewer neighbors as the nodes become physically separated (indicating reduced line-of-sight connectivity). It should be noted that a mean number of neighbors of M-2 is not possible (try to draw a graph for which this is the case, and this will quickly become apparent).

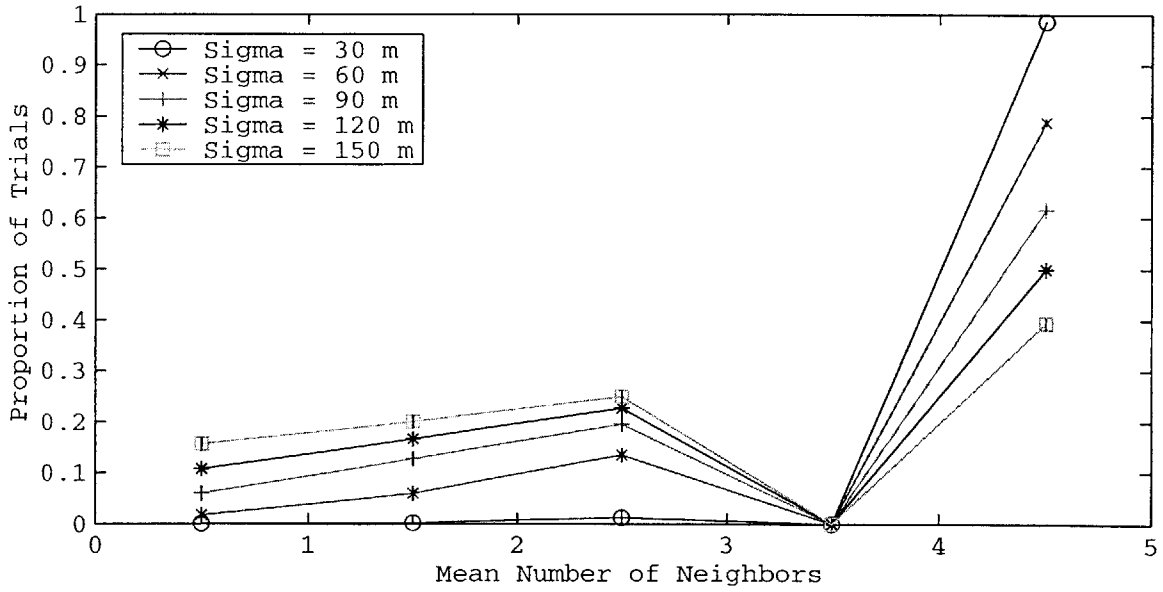


Figure 5-21 plots the mean cost of message delivery as a function of the dispersion distance: as physical separation between nodes increases, the mean cost of message delivery distribution shifts toward higher mean costs of message delivery (as expected).

FIGURE 5-20. Mean number of reachable nodes (from the perspective of a given node) averaged over all traverse positions and trials for a given dispersion distribution.

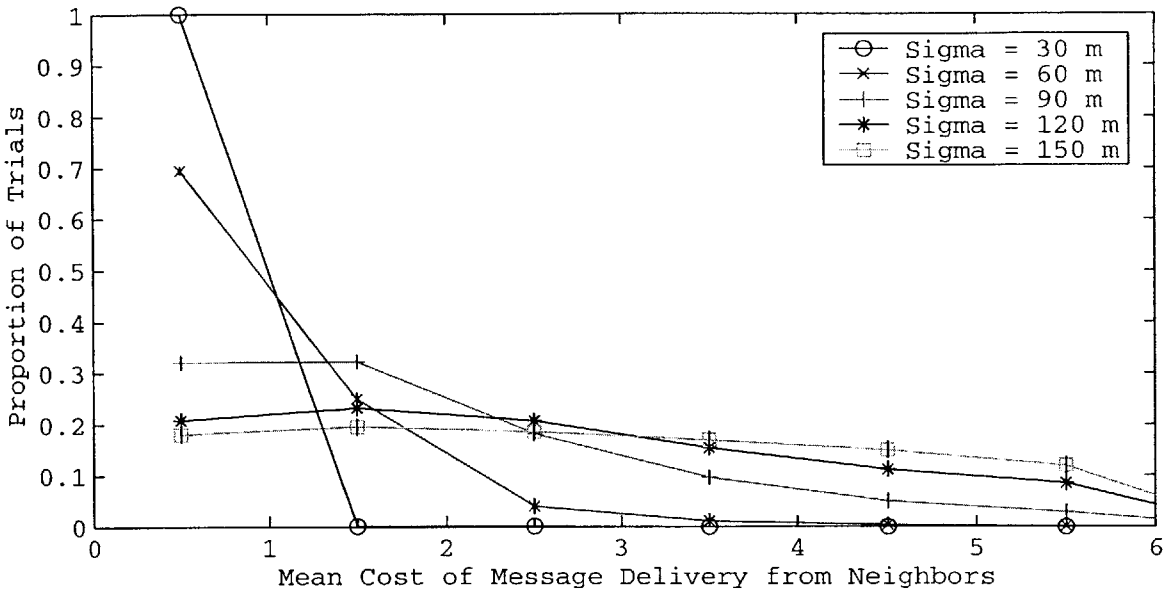


FIGURE 5-21. Mean cost of message delivery (from each node connected to a specific node) averaged over all nodes, trials, and traverse position for a given dispersion distribution.

The distribution of the mean number of disconnected nodes, plotted in Figure 5-22, trends towards higher mean numbers of disconnected nodes, while demonstrating that in a probabilistic sense, few nodes are disconnected.

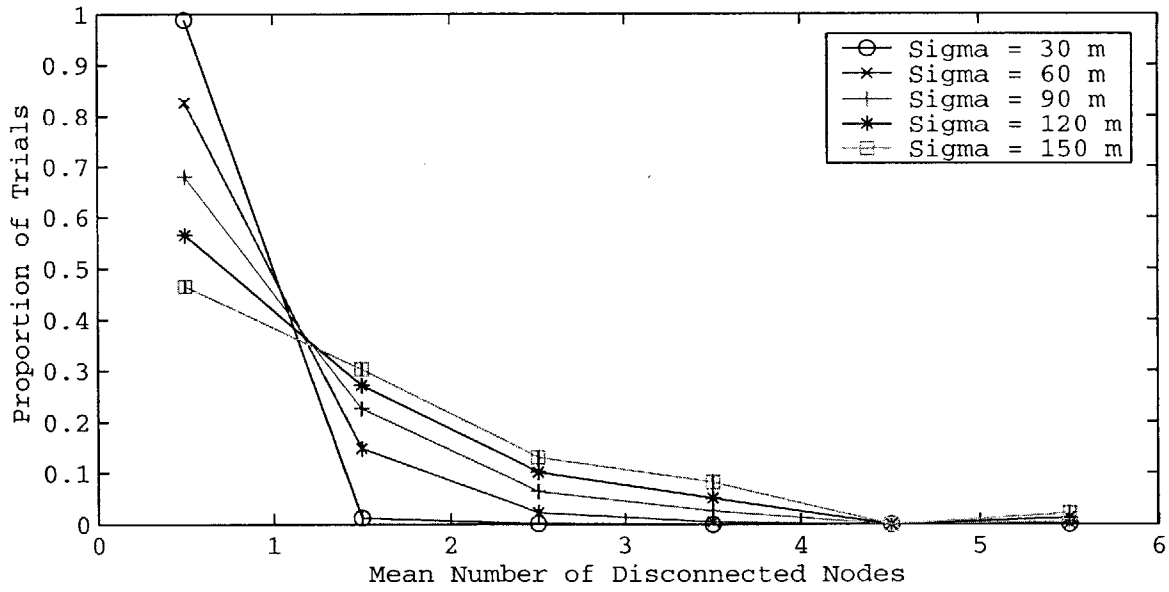


FIGURE 5-22. Number of disconnected nodes, averaged over all trials and traverse positions for a given dispersion distribution.

Discussion. These results imply that a short-range line-of-sight limited wireless system should be capable of achieving acceptable levels of connectivity within the workable range of high-power

Bluetooth™ devices. Furthermore, the results should not be highly sensitive to line-of-sight errors caused by the limited resolution of the digital elevation model because of the reduced sensitivity to he line-of-sight condition at close ranges.

Extensions to this analysis might include the addition of fixed-base wireless networking infrastructure such as a relay placed on a prominent topographic feature, or a vehicle-mounted relay or access point. In addition, it might be useful for the one of the members of the field geology group to be a motorized blimp or balloon (feasible both here on Earth and also on the Martian surface, under low wind conditions). Such a device could be a free flying robotic agent blimp, could might also be simply a tethered balloon with a minimal control system, serving as an "eye in the sky" and communications relay.

On a more general note, this type of analysis could be used in the traverse planning process to assess how widely a group of explorers might be able to physically separate and still stay in contact while executing a traverse. *Analyzing the aforementioned performance metrics along a traverse could provide a heuristic for the extent to which a group of agents need to coordinate their activities to remain in contact* - for example, when connectivity at longer distances is especially poor over some segment of the traverse, robotic agents might plan ahead and congregate closer together.

5.4.4 Rover Assembly of a Martian Sensor Network

The previous example demonstrated how a short-range wireless communication system could support coordination between agents on a traverse. This example will continue this thread by demonstrating how a different kind of distributed system can support agent mobility. This example examines how an autonomous or tele-operated robotic vehicle might assemble a communications and sensing network, while using the assembled network for its own communication needs during the construction phase.

Overview. The Crater Lake digital elevation model, discussed in Section 3.2.6, is used here as an analog to the Martian surface. An remote sensor network has been deployed in a remote valley, but the primary uplink/downlink between the sensor network and any orbiting satellites has failed, leaving the sensor network without communication. The task of the Rover is to autonomously traverse about 15 km around the crater while maintaining continuous communications coverage with the Rover's home base (Figure 5-23). In order to do this, the rover will deploy small "data wands" into the ground along the way: each wand will act as a heat flow probe (to study residual heat flow from the extinct volcano) while providing communication coverage for the Rover and a vital link between the Rover's "home base" and the remote sensor network.

The rover does not need to maintain constant contact with the communication network, but must guarantee that each deployment site has line-of-sight with the previous site, thus continuing the communications chain.

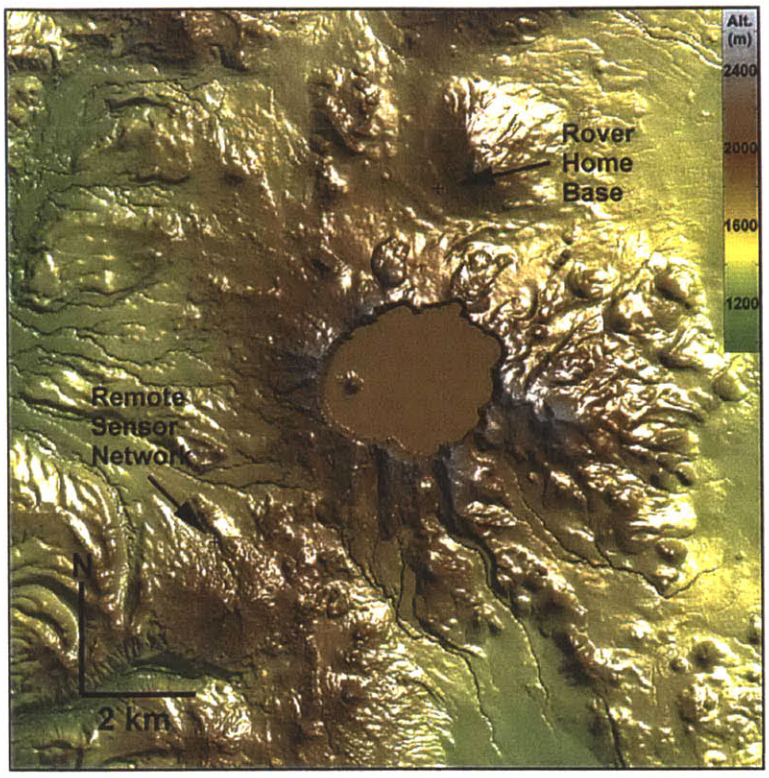


FIGURE 5-23. The Rover's mission: deploy a distributed surface communications and sensing network between "Home Base" and the "Remote Site," while maintaining at least periodic contact with the home base, and ensuring that each communications relay/sensor probe is network accessible. The rover must also plan its traverse so that it navigates around areas of steep terrain in order to stay on slopes less than 10 degrees.

Details of the simulation. The rover will conduct its own traverse planning using the optimization approach illustrated in Section 5.4.1, but in a more dynamic fashion (150 trial points are used in the generation of each minimum cost traverse). A reachability map is computed based on the Rover's initial position and a restriction of surface slopes to [0 20] degrees. The effective antenna height of the Rover and data wands is set to 1.5 m. It is assumed that similar antennas are located at the Rover's home base and at the site of the Remote Sensor Network. A nominal traverse velocity of 0.5 m/sec is assumed, and the mass of the Rover is set to 50 kg (not unrealistic for a rover of this size and capability). Gravitational acceleration is taken to be 3.69 m/s^2 (Mars surface).

For purposes of assessing the mean cost of message delivery, a nominal distance of 1 km is assumed between data wands, and the space loss exponent is taken to be 2.

Rover energy expenditure model. A simple model of Rover energy expenditure was built based on typical performance of the Lunar Roving Vehicle of 0.050 to 0.080 W-hr/km/kg. The cost of traversing on level slopes is assumed to be given by:

$$W_{L,v} = k \cdot v \cdot m + 5 \text{ [W]} \qquad \textbf{(EQ 5-11)}$$

where k = 0.216 Ws/m/kg = 0.060 W-hr/km/kg , v is the traverse velocity in m/s, and m is the vehicle mass in kg. The vertical work for uphill slopes is given by:

$$W_{V,Up} = k_\alpha \cdot v \cdot m \cdot \alpha \cdot \left(\frac{g}{g_{lunar}}\right) \text{ [W]} \qquad \textbf{(EQ 5-12)}$$

where α is the surface slope angle in degrees, g is the local gravitational acceleration, g_{lunar} is the lunar gravitational acceleration, and $k_\alpha \approx 0.0263$ Ws/m/kg characterizes the increase in energy

expenditures for a one degree increase in slope [Heiken et al., 1991]. The vertical work for downhill slopes is given by:

$$W_{V,Down} = \eta \cdot k_\alpha \cdot v \cdot m \cdot \alpha \cdot \left(\frac{g}{g_{lunar}}\right) \text{[W]} \qquad \textbf{(EQ 5-13)}$$

where $\eta \approx 0.3$ is used as an energy recovery efficiency factor, and α is a negative angle. Using $m = 495$ kg and $g = 1.62 \text{m/s}^2$ for a fully loaded Lunar Roving Vehicle on various traverses gives results of 30-60 W-hr/km (depending on the traverse), which agrees well with the actual range from the Apollo missions of 35-56 W-hr/km. The implementation of this simple, somewhat arbitrary model of Rover energy expenditures can be found in Appendix D (see *traverse_compute_rover_cost.m*).

Rover strategy for traverse planning and execution. The basic outline of the Rover planner can be most easily communicated with a bit of pseudocode:

```
do while and(not(mission accomplished), not(give up))
   compute visible region of surface
   compute minimum cost traverse to destination
   if minimum cost traverse contains a visible location
      traverse to visible location
      deploy a data wand
      if and(previous wand visible, target visible)
         mission accomplished
      else if previous wand not visible
         give up
loop
```

FIGURE 5-24. Simplified pseudocode version of the Rover traverse planning and execution strategy.

The actual implementation (see *rover_example.m* in Appendix D) in is somewhat more complicated, but basically implements the same core idea: At each new location, the Rover tries to plan a minimum cost traverse to the final destination, then tries to find a point on the minimum cost traverse that is also visible from the Rover's current position. This ensures that a wand placed at the new location will have line-of-sight with the wand already placed in the current location.

Results. The Crater Lake digital elevation model was sub-sampled by a factor of eight to speed up the simulation, and this had the effect of reducing the reachable area of the digital elevation model, therefore making it more challenging for the rover to successfully traverse to the remote site. Figure 5-25 illustrates a successful traverse *plan* generated by the Rover for the full-resolution (10 meter resolution) digital elevation model. Only iterative planning (planning a traverse part way to the destination) was successful in the sub-sampled version of the digital elevation model.

For the conditions given above, the Rover frequently failed early in the traverse by getting stuck in a high slope area near the North rim of the crater. The execution trace (edited, only for clarity) demonstrates how quickly the Rover can get stuck in high-slope areas, and how terminal this may be to mission success (Figure 5-26).

The simulation was rerun under exactly the same initial conditions, and the next time the Rover successfully placed a data wand within line-of-sight of the remote site antenna, completing a network of 19 data wands with a mean cost of delivery of 0.77. This successful traverse included several runs of repeated failures trying to find a suitable next data wand location, but ultimately triumphed by creating the data wand network shown in Figure 5-27 and Figure 5-28.

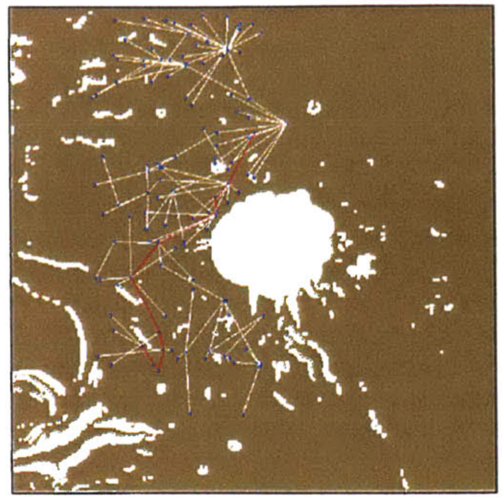


FIGURE 5-25. An example of a successful traverse planning attempt (red line) by the Rover from its current location to the target. Note that this plan does not take into account the required line-of-sight visibility constraint. The white areas indicate slope-restricted areas (>20 degree slopes).

Discussion. These results demonstrated some of the challenges in the deployment of a distributed system, but also demonstrated how a distributed system might be used to support extensive mobility of Rovers, or other vehicles that cannot afford (in terms of mass or power) to carry a ground-to-orbit transceiver. Use of data wands with a taller effective antenna (perhaps held high by small balloons) would greatly improved the surface coverage of the system and made deployment by the Rover much simpler.

```
INITIALIZING ROVER SIMULATION

ROVER STATE: id=1 x=558671 y=4743633 z=1874 kJ=0 t=0
NETWORK STATE: Nodes released=0 SourceVisible=1 TargetVisible=0 NodesConnected=1
MeanCost=NaN
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 556985 Y: 4742685 Z: 1908
ROVER TRAVERSING FROM X: 558671 Y: 4743633 Z: 1874 to X: 556985 Y: 4742685 Z: 1908

ROVER STATE: id=2 x=556985 y=4742685 z=1908 kJ=34 t=0
NETWORK STATE: Nodes released=1 SourceVisible=1 TargetVisible=0 NodesConnected=2
MeanCost=1.25
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 557465 Y: 4741885 Z: 2033
ROVER TRAVERSING FROM X: 556985 Y: 4742685 Z: 1908 to X: 557465 Y: 4741885 Z: 2033

ROVER STATE: id=3 x=557465 y=4741885 z=2033 kJ=65 t=0
NETWORK STATE: Nodes released=2 SourceVisible=1 TargetVisible=0 NodesConnected=3
MeanCost=1.16
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE
POINT
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, GIVING UP
```

FIGURE 5-26. An execution trace of a failed traverse attempt by the Rover.

For reasons of brevity, no extended analysis of energy expenditures by the Rover was performed. However, comparing the estimated energy expenditures for different Rover strategies could be infor-

mative in determining how such a distributed network might be most efficiently assembled, or how such a traverse might be efficiently carried out in an autonomous fashion.

5.5 Discussion

These examples demonstrated how traverse planning techniques can assist human exploration, but also highlighted how very basic are some of the heuristics proposed or implemented here: looking to human explorers for sources of heuristics for robotic planning (and vice-versa) may be fruitful.

The examples also demonstrated that the approach to traverse optimization depends greatly upon the particular limiting factor: during the Apollo missions, time was an extremely limiting factor, therefore the traverse planning approach was not necessarily toward energy expenditures. For very long-duration surface missions, energy expenditures directly relate to mission sustainability - the cost of mobility is likely to be a larger consideration for long duration missions.

As in the earlier examples, the previous example assumed that the explorer (the Rover in this case) had nearly full access to many characteristics of the environment (e.g., an accurate digital elevation model). In further studies it would be useful to model how individual agents acquire and assemble information about their environments, including the role of distributed systems in enhancing that process. Distributed systems will play a key role in how information about the environment is collected and disseminated - different agents ma have different capabilities that give them a comparative advantage in the collection, analysis, or delivery of specific data.

Only in the last example was something approaching a dynamic model of an agent used. In order to develop a better understanding of distributed architectures, a more complete approach to modeling dynamic agents should be considered.

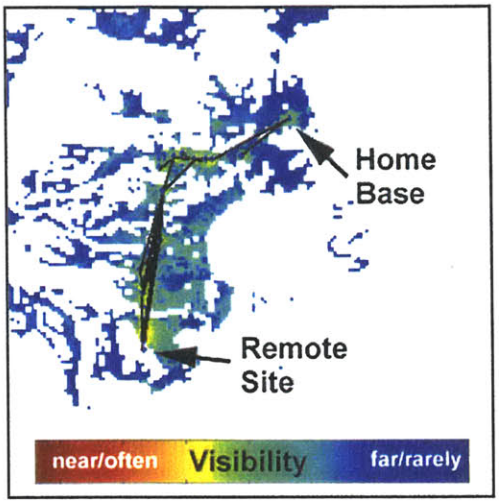


FIGURE 5-27. Surface coverage of the data wand sensing and communication network and remote site link as successfully deployed by the Rover. Dark lines represent line-of-sight visibility between data wands (points).

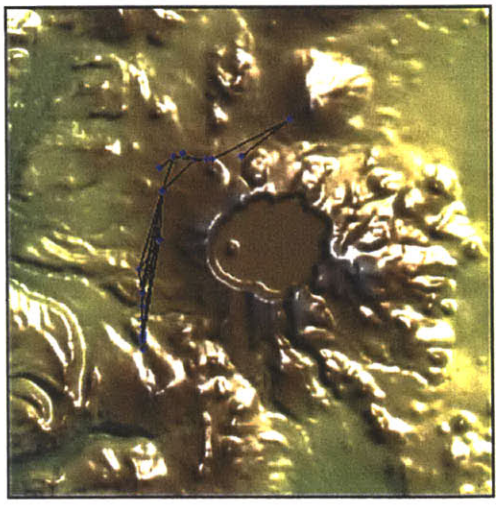


FIGURE 5-28. The data wand sensing and communication network and remote site link as successfully deployed by the Rover, overlaid on the Crater Lake digital elevation model. Dark lines represent line-of-sight visibility between data wands (points).

5.6 Recommendations

This section briefly summarizes recommendations that have been made throughout the thesis relating planetary surface exploration by human and robotic agents and the modeling and implementation of distributed systems. Recommendations of this thesis regarding distributed systems can be roughly organized into three categories including modeling, analysis, and physical real-world implementation of distributed systems. Additional considerations include the

study and improvement of human-robot interactions, and general considerations for the use of distributed systems.

Modeling. A true multi-agent system framework for surface exploration should be developed. This framework should include:

- Dynamic agent models, and dynamic models of interaction (not limited to a static graph structure description of distributed systems).
- Enhanced environment models, including modeling of inaccessible environments and enhanced surface modeling.

Analysis. The analysis tools developed as part of this thesis can be leveraged to a much greater extent than they have been up to now. However, new analysis tools should be developed, including:

- Analysis tools for analyzing graph structures and characterizing the static and dynamic characteristics of distributed systems.
- Additional traverse planning tools, for both virtual improvement in the process of traverse planning, and for actual use in the field.

Physical implementation. Simple, targeted distributed systems should continue to be developed, including sensor networks. Simple targeted distribute systems provide an opportunity to study the costs, benefits, and complexities of working with distributed systems. The short-range wireless connectivity example in this chapter demonstrated the feasibility of developing short-range wireless devices for activities such as communication and coordination during field geology. Development of wireless devices in support of field geology is a topic currently under study by the author. Targeted distributed systems can also be incorporated into analog simulation activities to pursue real science goals while learning more about the challenges of deploying and managing distributed systems in realistic environments.

Humans and robots. Opportunities for human and robot explorers to collaborate are numerous, and should be pursued beyond the traditional "astronaut and rover" team. While previous work in this arena has been productive and useful, there may be comparative advantages for specific activities to other human-robot partnerships. Potential examples might include:

- A field geology team composed of humans and one or more blimps, providing aerial photography or enhanced communication between explorers.
- The routine use of robots to get information (data or physical material) from inaccessible or dangerous localities (small, high, hot, poisonous localities).
- Smart field geology tools that automate parts of the data recording process (analysis of usage might also assist in the better

design of future tools) and provide mechanisms for sharing data between tools or between explorers.

Distributed systems, in general. Distributed systems must be considered from the perspective of how they help to achieve mission goals with a balance of performance, risk, and cost. While many wonderful possibilities exist, few are currently implementable because of our inexperience with distributed systems and lack of understanding of how to control complexity in distributed systems. If they are to be useful, distributed systems for planetary surface exploration will likely need to survive for long periods of time: ensuring adequate hardware lifetimes and consistent simple interfaces will be a challenge.

5.7 Summary and Conclusions

The theoretical basis for distributed architectures for surface exploration was examined in Chapter 2 by considering the modeling of distributed systems and the capabilities, needs, and goals of explorers and exploration.

In Chapter 3, an abstract representation of distributed system network topology was developed based on graph theory, and several tools for analyzing the surface environment and distributed system performance were developed, including line-of-sight connectivity and surface visibility (coverage). Major trades for distributed systems were examined, with an emphasis on efficiency. Multi-hop networks were demonstrated, in limited situations, to be more power efficient than single-hop communication systems. A trade study analysis process was also presented, and an example of a Mars surface sensor network demonstrated the opportunities and challenges in the use of a distributed system for large-spatial-scale surface sensing. The limited characterization of a distributed system as a static graph highlighted the need to treat distributed systems as dynamic networks of individual potentially-mobile agents.

A study of field geology and of voice communications and other aspects of the Apollo missions in Chapter 4 demonstrated some of the ways in which human explorers might benefit from distributed systems: fundamentally, distributed systems can change how explorers collect, analyze, interpret, and disseminate information, potentially enhancing the value of exploration by improving decision making abilities and coordination between explorers.

This chapter illustrated how some aspects of agent mobility can be quantified, and how the traverse planning process can be utilized by individual agents to their benefit and to the benefit of other agents. The examples in Section 5.4 demonstrated how traverse planning can be enhanced by the use of distributed systems, and (the dual)

how performance considerations of distributed system may require specific traverse planning activities.

In summary, the major contributions of this thesis include:

- Identification of major trades of distributed systems for planetary surface exploration,
- Development of a structured approach and a set of analysis tools for trade studies of distributed systems for planetary surface exploration,
- Development of a structured approach to traverse planning for human or robotic planetary surface explorers, and
- Recommendations for future distributed system development and operational concepts for future human planetary surface exploration.

While there are many limitations to the work presented in this thesis, the primary purpose of this thesis is to open doors - to try to elucidate what are the important questions to consider for distributed architectures- while attempting to consider the bottom line: how can distributed architectures support human and robotic surface exploration? This question can now be definitively answered.

Distributed systems fundamentally enhance the exploration process by changing the way in which information is collected and disseminated by exploration agents. This additional access to information, coupled with appropriate decision-making tools, enhances self-sufficiency and autonomy and allows individual explorers to enhance their probability of survival, coordinate their activities with other explorers, and increase the value they add to their overall mission.

Surface-based distributed architectures, and the elements of which they are composed, can be characterized by a quantifiable set of performance metrics including cost of transport of system elements and connectivity metrics. However, the design and optimization of distributed systems is a difficult process, and the benefits of distributed architectures come with the significant burden of added system complexity. Targeted deployment of distributed systems for planetary surface exploration can lead to increases in system performance, in terms of enhanced flexibility and robustness, when system complexity is adequately managed.

5.8 References

Bluetooth, *Bluetooth website (http://www.bluetooth.org/)*, 2001.

Cavagna, G.A., Zamboni, A., Farragiana, T., Margaria, R., *Jumping on the Moon: power output at different gravity values.* Aerospace Medicine, 1972.

Connors, Mary M., Eppler, Dean B., and Morrow, Daniel G., *Interviews with the Apollo Lunar Surface Astronauts in Support of Planning for EVA Systems Design*, Ames Research Center, National Aeronautics and Space Administration, 1994.

Dickerson, P.W., *Exploration Strategies for Human Missions*, Mars Field Geology, Biology and Paleontology Workshop, Concepts and Approaches for Mars Exploration, Lunar and Planetary Institute, Houston, Texas, July 18-20, 2000.

FAR/AIM, *Federal Aviation Regulations & Aeronautical Information Manual*, Aviation Supplies & Academics, Inc., 2001.

Graff, C., and Leibman, E., *Concepts for an Adaptive Network Planner for the Localized Access Area (LAA) for Battlefield Information Systems (BIS) 2015*, IEEE 1994.

Heiken, G., Vaniman, D., and French, B.M., *Lunar Sourcebook: A Users's Guide to the Moon*, Cambridge University Press, Cambridge, England, 1991.

Howe, D., *Free On-Line Dictionary of Computing*, 2001.

IEEE802.11, *IEEE 802.11 wireless LAN specification*, Institute of Electrical and Electronics Engineers, 2001 (*http://standards.ieee.org/catalog/IEEE802.11.html*).

Johnston, R.S., Dietlein, L.F., and Berry, C.A., *Biomedical Results of Apollo (SP-368)*, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1975.

Jones, Eric M. (ed.), *Apollo Lunar Surface Journal*, July 2000 (*http://www.hq.nasa.gov/office/pao/History/alsj/*).

Margaria, R., *La condizione di subgravita e la sottrazione dall'effetto delle accelerzaioni.* Riv. Med. Aer., Vol. 16, p. 469-474, 1953.

Margaria, R., Cavagna, G.A., Saiki, H., *Human locomotion at reduced gravity*, Proceedings of the 2nd Lunar International Symposium. Life Sciences Research and Lunar Medicine. Pergamon Press, Oxford and New York, 1967.

Muelberger, W.R., *Apollo 16 Traverse Planning and Field Procedures*, USGS Professional Paper 1048, Part C, United States Government Printing Office, Washington, D.C., 1981.

Newman, D.J., *Human Locomotion and Energetics in Simulated Partial Gravity*, PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

Newman, D.J. and Alexander, H.L., *Human Locomotion and Workload for Simulated Lunar and Martian Environments*, Acta Astronautica, Vol. 29, No. 8, pp.613-620, 1993.

Nute, R.H., Blevins, R.V., and Koppa, R.J., *Apollo 14 Final Lunar Surface Procedures*, Lunar Surface Operations Office, National Aeronautics and Space Administration, Manned Spacecraft Center, Houston, Texas, December 31, 1970.

Rechtin, Eberhardt, *Systems Architecting*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Santee, W.R., Allison W.F., Blanchard, L.A, and Small, M.G, "A Proposed Model for Load Carriage on Sloped Terrain", *Aviation, Space, and Environmental Medicine*, Vol. 72, No. 6, June 2001.

Shaw, G.B., *The Generalized Information Network Analysis Methodology for Distributed Satellite Systems*, PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1999.

Srinivasan, R. Srini, Leonard, Joel I., and White, Ronald J., *Chapter 26 - Mathematical Modeling of Physiological States*, Space Biology and Medicine, Volume III, Book 2, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1996.

Stone, R.W., *Man's Motor Performance Including Acquisition f Adaptation Effects in Reduced Gravity Environments*, National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia, 1974.

Stuster, Jack, *Bold Endeavors*, Naval Institute Press, Annapolis, Maryland, 1996.

Swann, G.A., Bailey,N.G, Batson, R.M., Eggleton, R.E., Hait, M.H., Holt, H.E., Larson K.B., Reed, V.S., Schaber, G.G, Sutton, R.L., Trask, N.J., Ulrich, G.E., and Wilshire, G.E., *Geology of the Apollo 14 Landing Site in the Fra Mauro Highlands*, USGS Professional Paper 880, United States Geological Survey.

Tutschku, K., Leibnitz, K., Tran-Gia, P., *ICEPT-An Integrated Cellular Network Planning Tool*, IEEE 1997.

Waligora, J.M., *The Use of a Model of Human Thermoregulation During the Apollo and Skylab Programs (N76-11172)*, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1976.

Do not worry about your difficulties in mathematics, I assure you that mine are greater.

Albert Einstein, (1879-1955)

Appendix A - Mathematical Reference

A.1 Power Spectral Density Estimation

A.1.1 Two-Dimensional Discrete Fourier Transform

The two-dimensional discrete fourier transform is a frequency domain representation of a finite two-dimensional sequence $x(n_1, n_2)$ where $0 \le n_1 \le N_1 - 1$ and $0 \le n_2 \le N_2 - 1$. The two-dimensional discrete fourier transform can be defined as [Lim, 1990, p. 141]:

$$X(k_1, k_2) = \sum_{n_1 = 0}^{N_1 - 1} \sum_{n_n = 0}^{N_2 - 1} x(n_1, n_2) e^{-j\left(\frac{2\Pi}{N_1}\right)k_1 n_1} e^{-j\left(\frac{2\pi}{N_2}\right)k_2 n_n} \quad \textbf{(EQ A.1)}$$

for $0 \le k_1 \le N_1 - 1$ and $0 \le k_2 \le N_2 - 1$, and is otherwise 0. In a similar fashion, the inverse transform can be defined as:

$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1 = 0}^{N_1 - 1} \sum_{k_2 = 0}^{N_2 - 1} X(k_1, k_2) e^{j\left(\frac{2\pi}{N_1}\right)k_1 n_1} e^{j\left(\frac{2\pi}{N_2}\right)k_2 n_2} \quad \textbf{(EQ A.2)}$$

for $0 \le n_1 \le N_1 - 1$ and $0 \le n_2 \le N_2 - 1$, and is otherwise 0. Note that these definitions apply only to a first-quadrant finite sequence. Lim notes that this is "not a serious restriction in practice, since a finite-extent sequence can always be shifted to have first-quadrant support, and this shift can easily be taken into account in many application problems."[Lim, 1990, p. 141].

The relationship between $x(n_1, n_2)$ and $X(k_1, k_2)$ can be written in shorthand as $x(n_1, n_2) \leftrightarrow X(k_1, k_2)$.

A.1.2 Two-Dimensional Power Spectral Density

The two-dimensional power spectral density of a finite sequence can be estimated directly from the two dimensional discrete fourier transform as:

$$\hat{P}_x(k_1, k_2) = \frac{1}{N}|X(k_1, k_2)|^2 \qquad \text{(EQ A.3)}$$

where N is the number of points in the finite sequence $x(n_1, n_2)$. From Equation A.1, the units of $X(k_1, k_2)$ are the same as the units of $x(n_1, n_2)$, so if $x(n_1, n_2)$ represents a height field of altitude samples in meters, $\hat{P}_x(k_1, k_2)$ has units m^2.

To compare two or more height field power spectral densities, it is beneficial to renormalize the power spectral densities by the spacing between altitude samples. This renormalized power spectral density can be written as:

$$\hat{P}_x(k_1, k_2) = \frac{w^2}{N}|X(k_1, k_2)|^2 \qquad \text{(EQ A.4)}$$

where w is the spacing between altitude samples for a given height field. This equation will be used to compute the power spectral density when comparing height fields, and will be referred to as *normalized power spectral density*. Figure A-1 illustrates the effect of renormalizing the power spectral density.

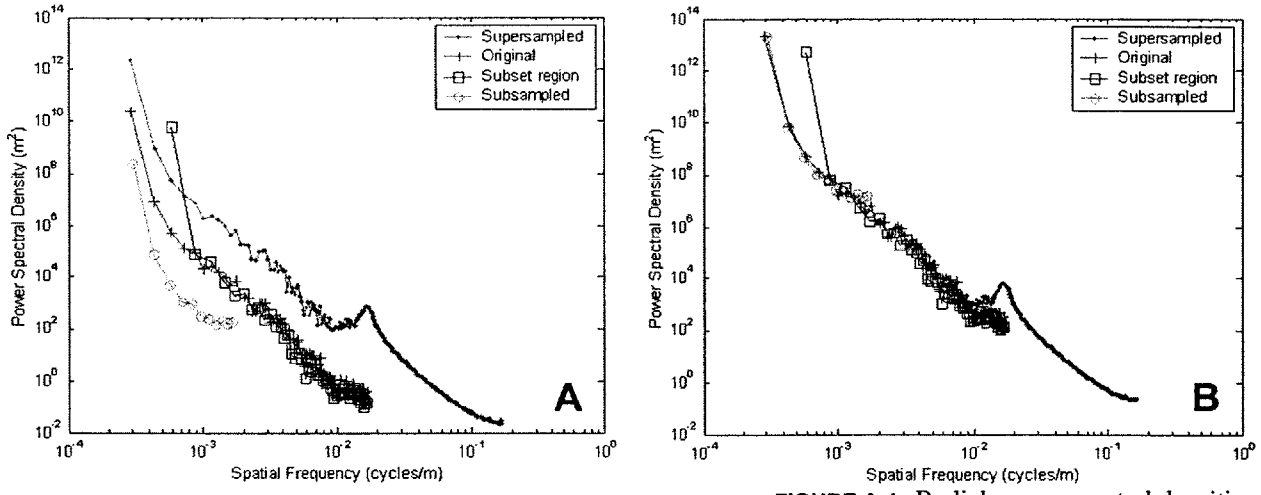


FIGURE A-1. Radial power spectral densities computed for different representations of the same height field, including a sub-region, a sub-sampled version of the height field, and a super-sampled version of the height field. In (A), power spectral densities are plotted as computed with Equation A.3. In (B), normalized power spectral densities are plotted, computed from Equation A.4.

While the discrete fourier transform provides an easy way to estimate the power spectral density, it is by no means the only, the most effective, or the most accurate way to estimate the power spectral density in many cases. Lim (1990) describes many other methods for two-dimensional power spectral estimation including maximum likelihood, autoregressive signal modeling, and maximum entropy methods.

A.1.3 Radial Power Spectral Density

While a radially-symmetric two-dimensional power spectral density $P_x(k_1, k_2)$ is guaranteed to have a real corresponding sequence $x(n_1, n_2)$, a real sequence $x(n_1, n_2)$ does not necessarily have a radially-symmetric two-dimensional power spectral density $P_x(k_1, k_2)$. If the statistics of $x(n_1, n_2)$ are anisotropic, $P_x(k_1, k_2)$ is not likely to be radially symmetric. In this case it may still be desirable to define a representative radial power spectral density profile (e.g. a profile from low to high spatial frequencies) even through no single radial power spectral density profile exists.

Figure A-2 illustrates several methods of estimating a representative radial power spectral density profile for a non-radially-symmetric power spectral density. Low-frequency components produce the horizontal and vertical banding seen in Figure A-2: only sampling the diagonals of the non-radially-symmetric power spectral density avoids the banded areas. Of the methods illustrated in the figure, sampling the diagonals generally provides the best estimate for a radial power spectral density profile.

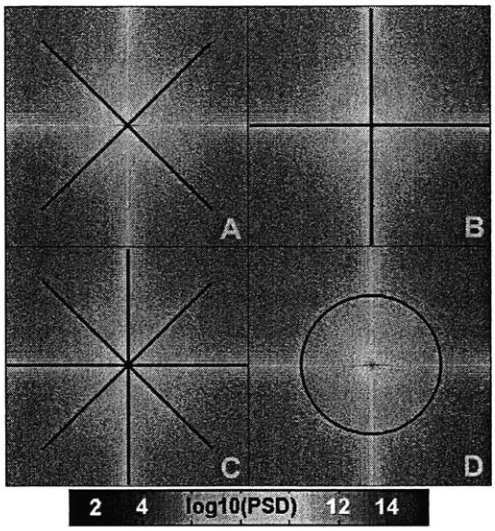


FIGURE A-2. Four methods of estimating a representative radial power spectral density profile for a non-radially-symmetric power spectral density. In (A), the radial power spectral density is estimated from the main diagonals of the two-dimensional power spectral density. The profile for half of each dark diagonal line is averaged with the other three similar profiles. In (B), the central rows and columns are used in the same manner as the diagonals in (A). (C) combines the approaches of (A) and (B). In method (D), a mean value of the power spectral density radial profile is computed for each spatial frequency by "averaging around the circle." Note that the low spatial frequency component is shown centered for the above power spectral densities. For isotropic power spectral densities, all methods are equivalent.

A.2 Entropy Principles

A.2.1 Shannon Measure of Entropy

The Shannon measure of entropy for a probability distribution $p = (p_1, p_2, \ldots, p_n)$ is given by [Shannon and Weaver, 1963]:

$$S(p) = -k \sum_{i=1}^{n} p_i \ln p_i . \qquad \text{(EQ A.5)}$$

where k is an arbitrary positive constant. In addition to being used as a measure of uncertainty, the Shannon measure of entropy can be used as a measure of "equality, disorder, diversity, lack of concentration, similarity, objectivity, unbiasedness, randomness,... and many other characteristics that do not even require probabilistic concepts for their description and have no relationship with uncertainty."[Kapur and Kesavan, 1992, p. 10]

A.2.2 Principle of Maximum Entropy

The principle of maximum entropy states that given a set of constraints for an unknown probability distribution p, the best estimate of p is the probability distribution that maximizes the Shannon entropy $S(p)$, subject to the given constraints. In plain language

this principle can be stated as "Speak the truth and nothing but the truth; Make use of all the information that is given and scrupulously avoid making assumptions about information that is not available."[Kapur and Kesavan, 1992, p. 37]

A.2.3 Principle of Minimum Cross-Entropy

The Minimum Cross-Entropy principle is a statement of directed divergence of one probability distribution from another probability distribution. The Kullback-Leibler measure can be used to characterize the directed divergence of any two distributions p and q, whether or not they represent probability distributions. Given two distributions $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$, the Kullback-Leibler measure is defined as [Kapur and Kesavan, 1992]:

$$D(p, q) = \sum_{i=1}^{n} p_i \ln \frac{p_i}{q_i} \qquad \text{(EQ A.6)}$$

with $p_i = 0$ whenever $q_i = 0$, and defining $0\ln(0/0) = 0$. Here, q represents an a-priori distribution. The goal of the principle of minimum cross-entropy is to choose a single distribution p subject to some set of constraints such that $D(p, q)$ is minimized. Properties of $D(p, q)$ include [Kapur and Kesavan, 1992, p. 153-161]:

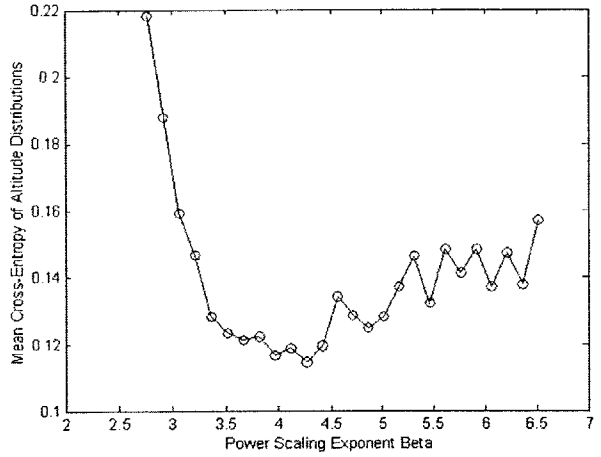
- non-negativity: $D(p, q) \geq 0$,
- identity: $D(p, q) = 0$, if and only if $p = q$,
- continuity: $D(p, q)$ is a continuous and convex function of $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$,
- permutational symmetry: $D(p, q)$ is the same if pairs (p_i, q_i) are re-labelled.

Minimization of the Kullback-Leibler measure of cross-entropy of p with the a-priori distribution q as the uniform distribution, subject to a set of constraints, is identical to maximizing the Shannon measure of entropy for p, subject to the same constraints.

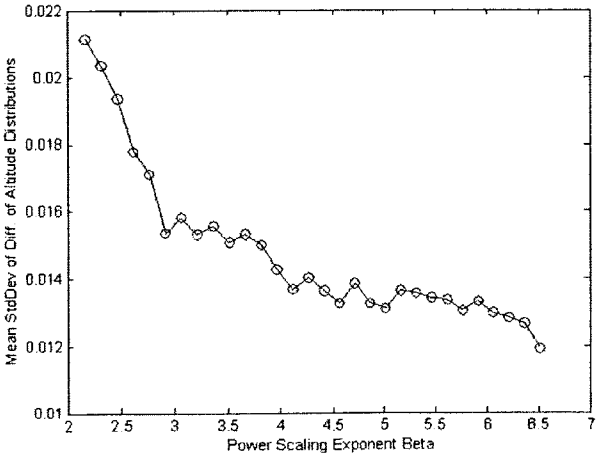
To illustrate why the Kullback-Leibler measure of cross-entropy might be more appropriate in some cases than another metric such as the standard deviation of the difference between two distributions, a simple example is in order.

In Chapter 3, a power spectral density scaling law that scales as $f^{-\beta}$ (where f is the spatial frequency) was used to generate surfaces similar to existing digital elevation models. The problem there was to find the best-fit altitude distribution of the topography, and consequently the best value of β. In this case, the a-priori dis-

tribution q was given by the altitude distribution of the existing digital elevation model, and the problem was to find the value of β that, on average, produces an altitude distribution p that minimizes the divergence of p from q. An initial estimate of $\beta = 4.6$ was determined by estimating a radial power spectral density profile, a set of β_i 's near β was generated, and many (n=100) surfaces were generated for each β_i. The altitude distribution p was computed for each surface, and the mean cross-entropy between p and q computed for each β_i. Figure A-3 plots the mean cross-entropy as a function of the power law scaling exponent, revealing a minimum cross-entropy of approximately 0.115 for $\beta \approx 4.3$. When the standard deviation of the difference in altitude distributions is used instead of cross-entropy of the altitude distributions, no global minimum is clearly identifiable.



(A) Mean Cross-Entropy



(B) Mean Standard Deviation

FIGURE A-3. A plot of mean cross-entropy between the altitude distributions of a real height field and generated height fields as a function of the power law scaling exponent is shown in (A): According to the principle of minimum cross-entropy, the best-fit power law scaling exponent is approximately 4.3. In (B), the standard deviation of the difference in altitude distributions was used instead of the cross entropy. No clear global minimum is visible in the range of power law scaling exponents for which surfaces were generated. In this case, for each value of the power law scaling exponent, 100 different surfaces were generated and analyzed.

A.3 References

Lim, J.S., *Two-Dimensional Signal and Image Processing*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

Kapur, J.N., and Kesavan, H.K., *Entropy Optimization Principles with Applications*, Academic Press, San Diego, California, 1992.

Shannon, Claude E. and Weaver, Warren, *Mathematical Theory of Communication*, University of Illinois Press, 1963.

'The time has come,' good Schmitty said,
'To talk of many things;
Of suits - and ships - and lunar dust -
Of rocks and crater rings -
And when the moon once boiled hot -
and whether it still sings.'

'But wait a bit,' the CapCom cried,
'Before we launch our craft;
For all of us are out of breath,
and soon we'll be unstaffed!'
'No hurry!' said the lunar men,
They thanked him much and clapped.

'A view of Earth,' good Cernan said,
'Is what we chiefly need:
The little sphere of white and blue
Is beautiful indeed -
And now you know, good NASA folk,
that planted is the seed.'

'But not in all,' good NASA cried,
Turning a little sad.
'And after such discovery...
we can't but think it bad!'
'The work is done,' good Cernan said.
The CapCom cried 'we're glad!

It was so nice to be your guide!
The moon is very nice.'
The lunar men said nothing, but
they took another slice.
'I wish you both were not so deaf -
let's go! I've asked you twice!'

'It seems a shame,' the CapCom said,
'To play the game this way
After we've brought them out so far,
and made them leave today!'
The NASA folks said nothing but
'We're going back some day!'

Christopher E. Carr, *Apollo in Wonderland (2001)*

Appendix B - Detailed Results of the Apollo Voice Communications Study

B.1 Introduction

This appendix provides additional results from the study of voice communication transcripts from the lunar surface exploration segments of the Apollo missions, but provides little or no discussion of the results. For details of the database creation and analysis process, or a more detailed discussion of the results, please see Chapter 4.

When results are shown as a function of mission role, results are segmented by CDR, LMP, several CapComs, and other mission roles. Different CapComs were often designated for different mission phases such as landing and post-landing, EVA, wakeup and goodnight, and launch. The function of each CapCom is evident from the MET of voice communications, and no attempt has been made to label specific CapComs by their assigned mission phases.

For each Apollo mission, summary results of the voice communications study are detailed as a function of mission role, including:

- Cumulative communications initiations,
- Time interval between communications histograms,
- Communications duration histograms, and a
- Statistical summary.

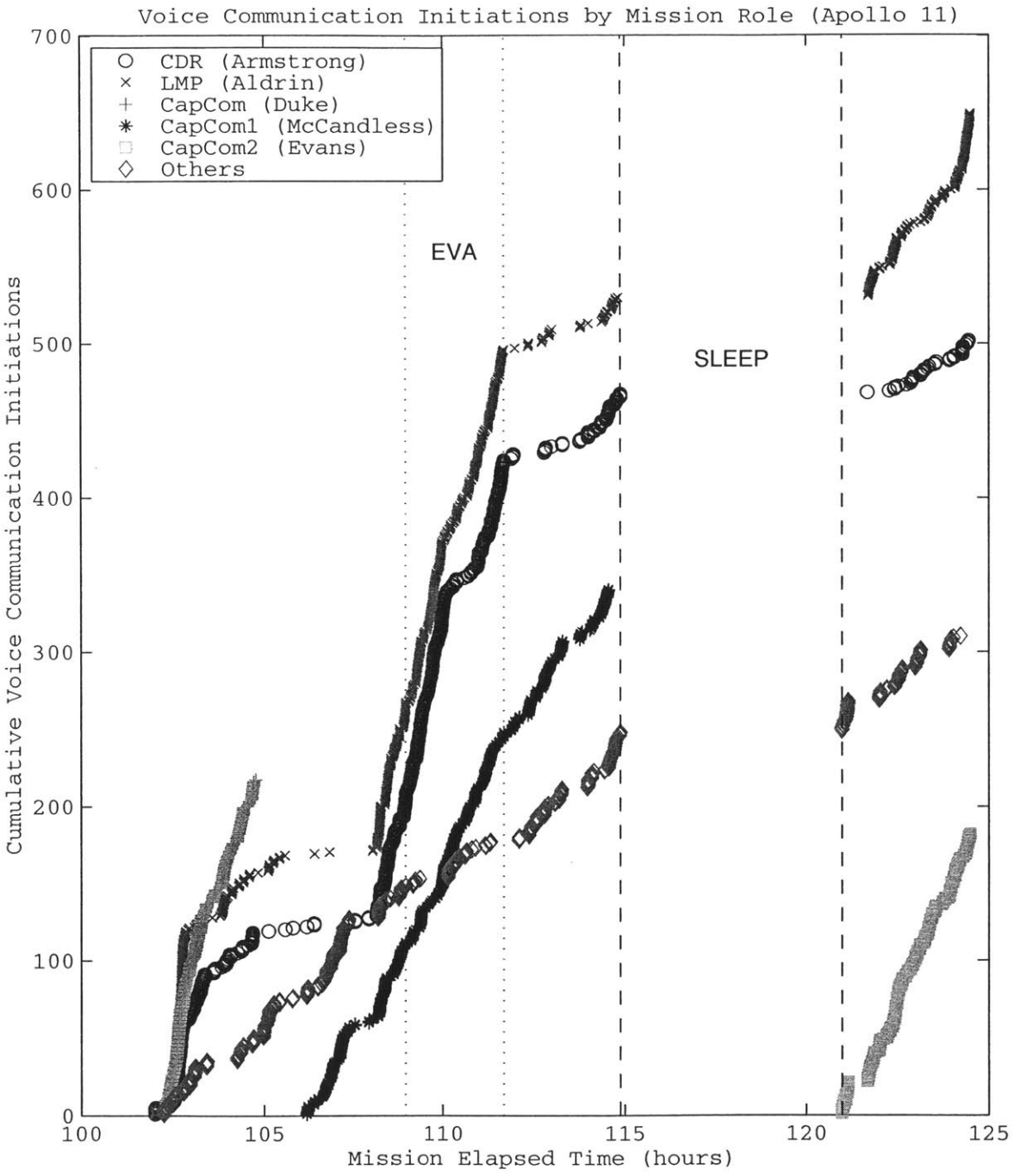
Communications durations are computed as if all available communication time was used for voice communications, and therefore represent an upper bound for the duration of the actual voice communications. The time interval between communications is computed as the time between a communication initiation by an individual until another communication initiation by the same individual.

Comparisons between different Apollo missions are then made, including a comparison of mean voice communication rates, and rates corrected for sleep periods. Mean communication during EVA and non-EVA periods are also analyzed, and mean communication rates are computed as a function of time for each type of surface operation (sleep, non-EVA, EVA). Voice communications, segmented by mission role, are then compared across all missions. Mean voice communication time intervals and durations, segmented by mission role, are compared across all missions.

Several important abbreviations will be used throughout this appendix. They are:

- *Mission Commander (CDR)*
- *Lunar Module Pilot (LMP)*
- *Mission Control Primary Communicator (CapCom)*
- *Extravehicular Activity (EVA)*
- *Mission Elapsed Time (MET)*
- *Command Module Pilot (CMP)*

B.2 Apollo 11



Notice the higher rate of communication initiations during the EVA, and for the CapComs as compared to the CDR and LMP. Many of the communications classified as "other" are due to the command module pilot (CMP).

FIGURE B-1. Voice communication initiations for Apollo 11 lunar surface exploration by mission role as a function of mission elapsed time.

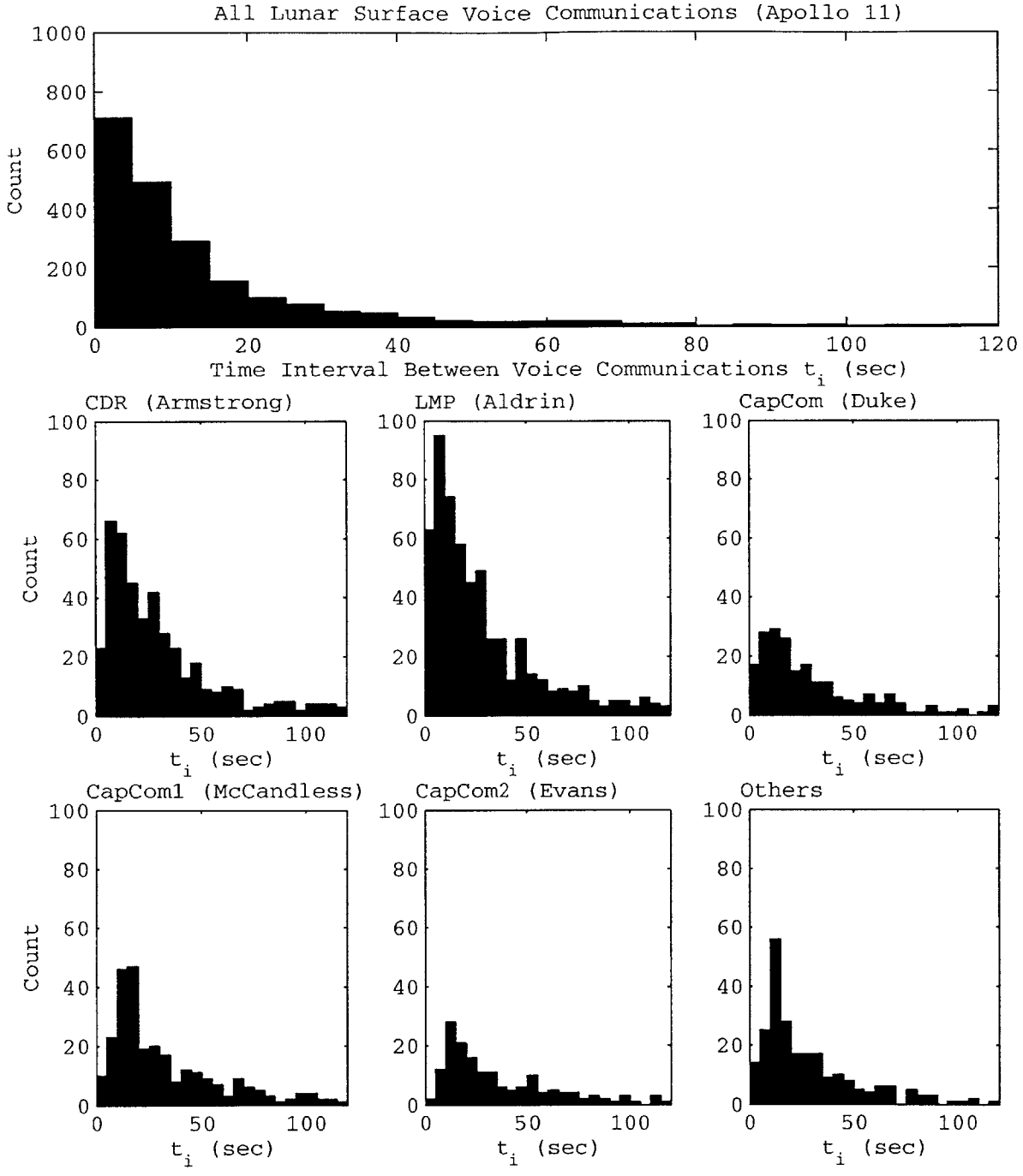


FIGURE B-2. Time intervals between voice communications for Apollo 11 lunar surface exploration by mission role. Bin size is 5 seconds. The few time intervals larger than two minutes are excluded.

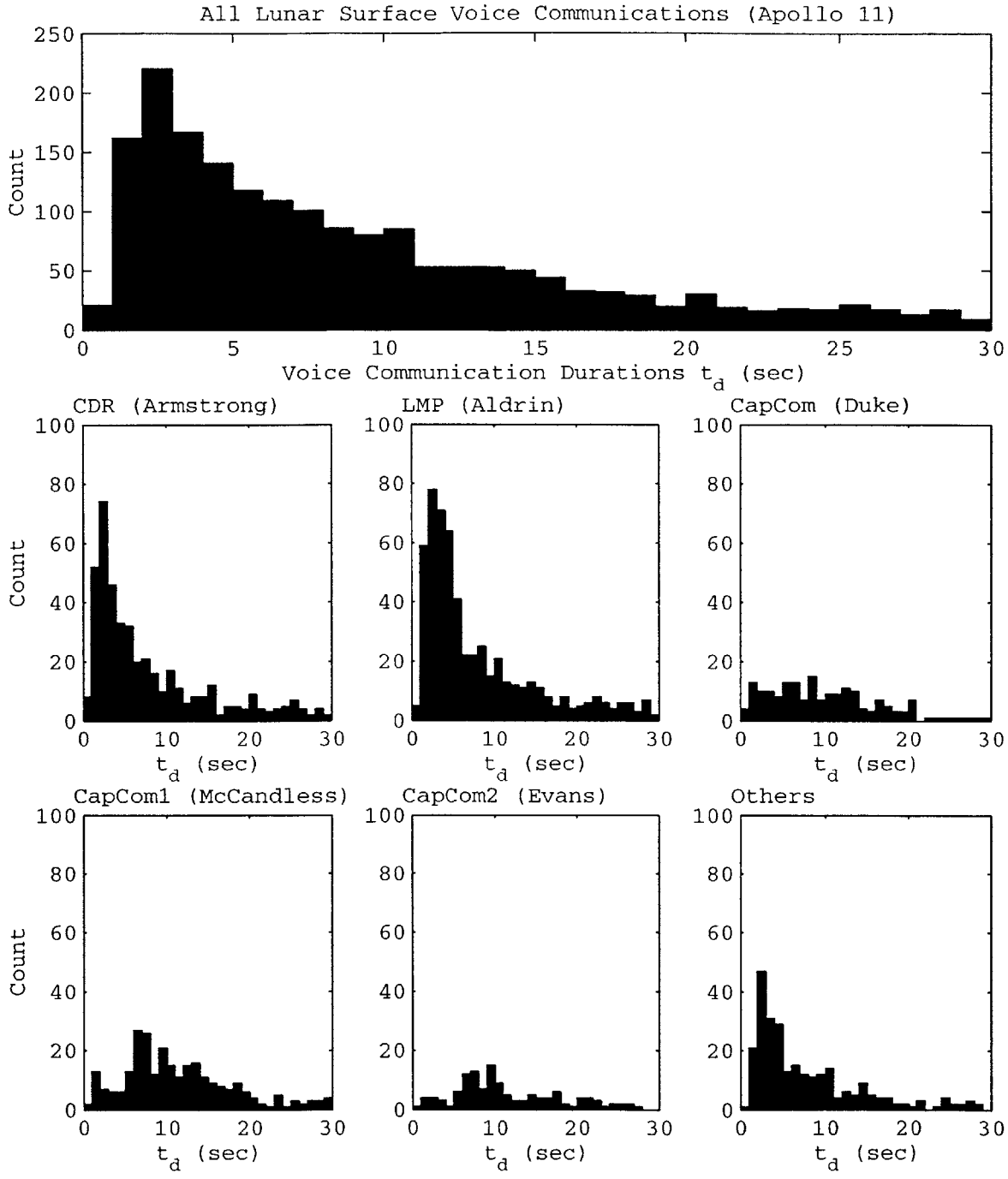


FIGURE B-3. Voice communications durations (upper bounds) for Apollo 11 lunar surface exploration by mission role. Bin sizes are 1 second (accuracy limit from voice transcripts). Voice communication durations greater than 30 seconds are excluded.

TABLE B-1. Voice Communication Analysis (Apollo 11)

Mission Role	# Analyzed	# Corrected	# Not Analyzed	Total	% of Total
CDR	502	6	0	502	22.82%
LMP	649	3	0	649	29.50%
CapCom	217	0	0	217	9.86%
CMP	257	0	0	257	11.68%
CapCom1	340	0	0	340	15.45%
CapCom2	182	0	0	182	8.27%
Others	53	0	0	53	2.41%
Sub-Total (All)	2200	9	0	2200	100.00%
% of Total	**100%**	**0.41%**	**0.00%**	**100%**	

TABLE B-2. Time Interval Statistics (Apollo 11)

Mission Role	Mean Interval (seconds)	5%* Interval (seconds)	25% Interval (seconds)	75% Interval (seconds)	95% Interval (seconds)
CDR	72.45	5	12	58	365
LMP	58.59	3	10	51	247
CapCom	41.79	3	11	47	136
CapCom1	80.13	8	15	81	338
CapCom2	59.4	8	15	64	213
Others	104.56	5	13	75	529
All	23.97	1	3	19	82

TABLE B-3. Communication Duration Statistics (Apollo 11)

Mission Role	Mean Duration (seconds)	5% Duration (seconds)	25% Duration (seconds)	75% Duration (seconds)	95% Duration (seconds)
CDR	20.13	1	2	16	63
LMP	17.53	1	3	16	61
CapCom	23.33	1	5	19	82
CapCom1	33.17	2	7	25	141
CapCom2	35.37	2	8	37	124
Others	27.43	1	3	15	112
All	23.98	1	3	19	82

B.3 Apollo 12

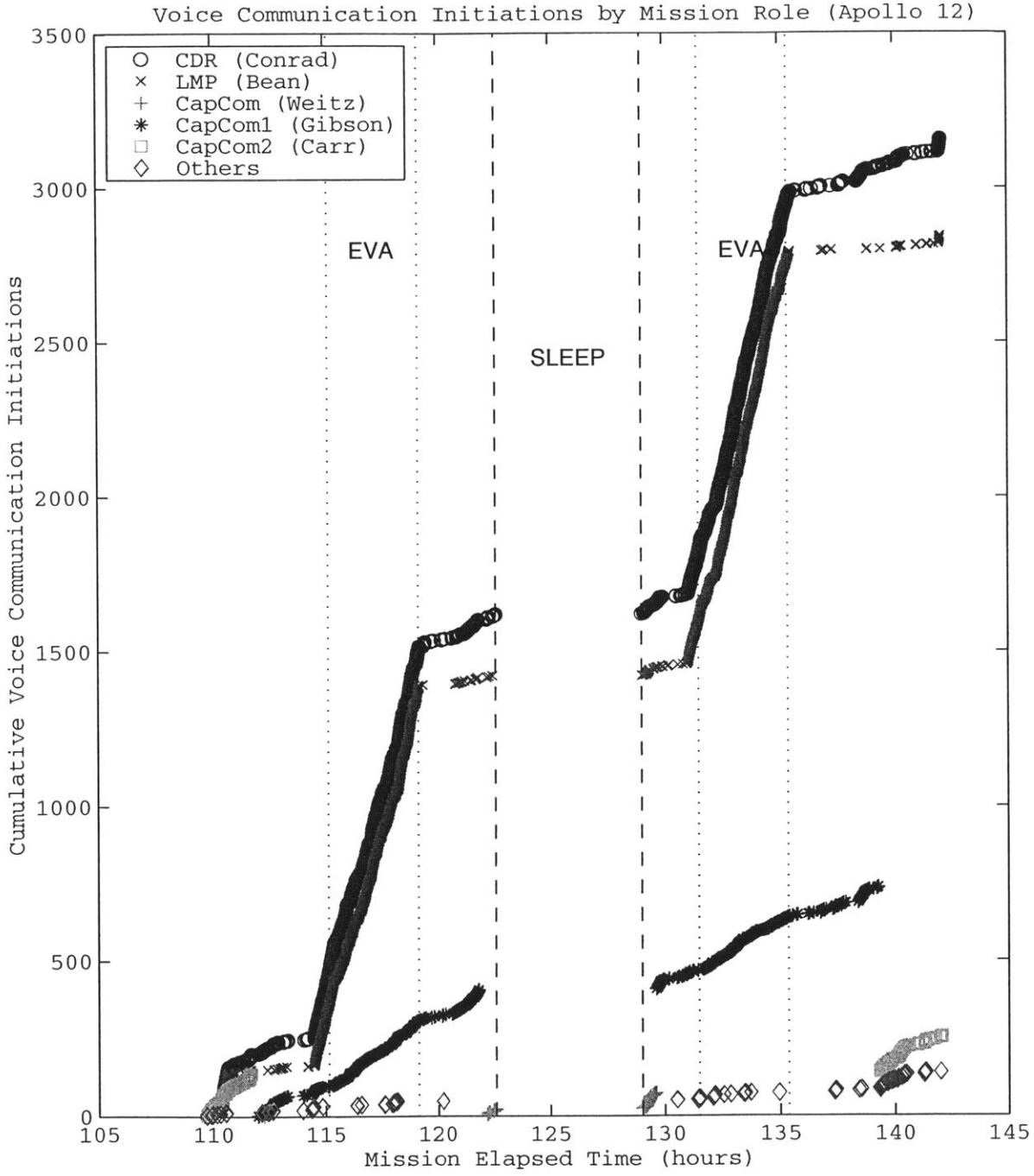


FIGURE B-4. Voice communication initiations for Apollo 12 lunar surface exploration by mission role as a function of mission elapsed time.

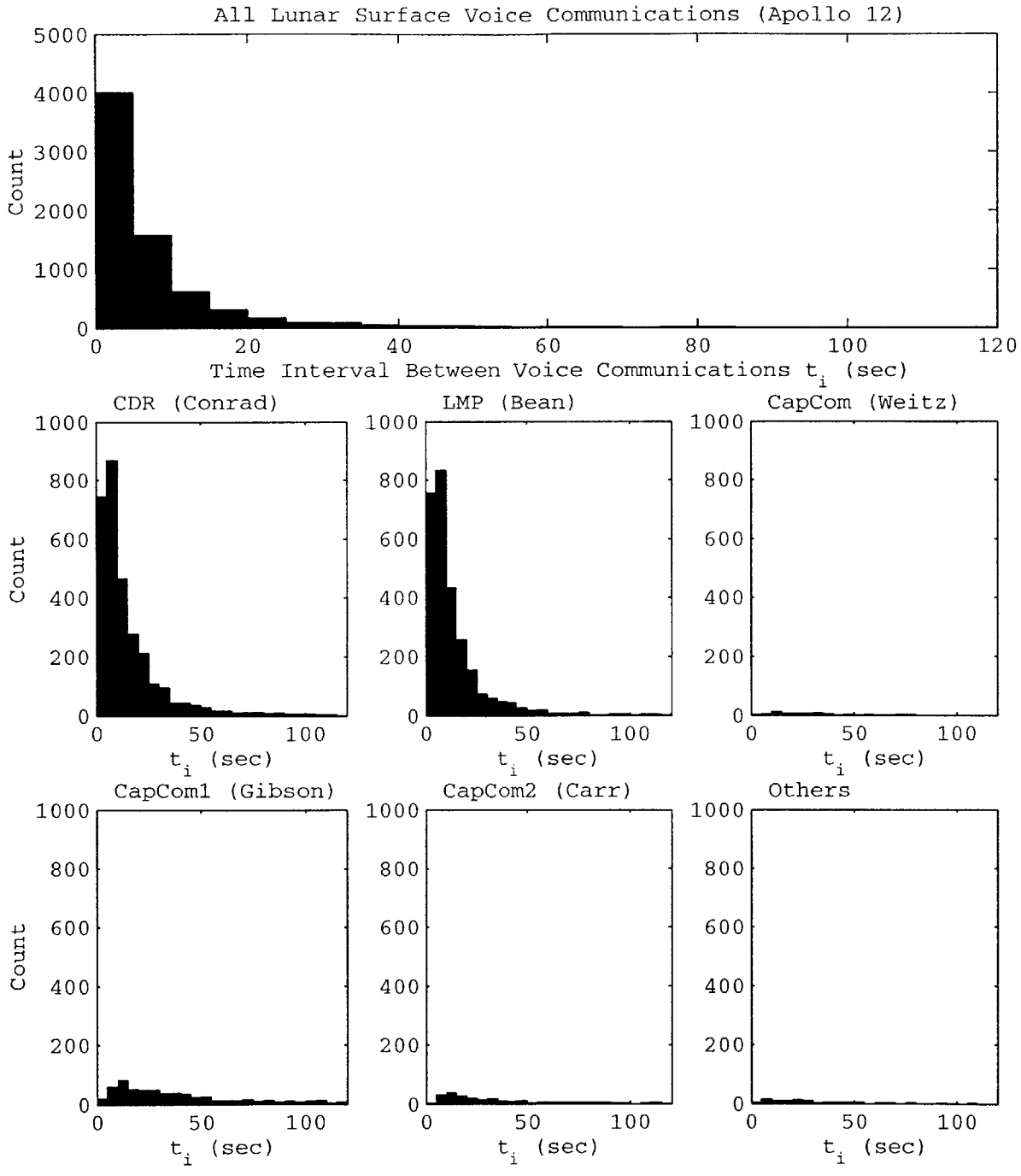


FIGURE B-5. Time intervals between voice communications for Apollo 12 lunar surface exploration by mission role. Bin size is 5 seconds. The few time intervals larger than two minutes are excluded.

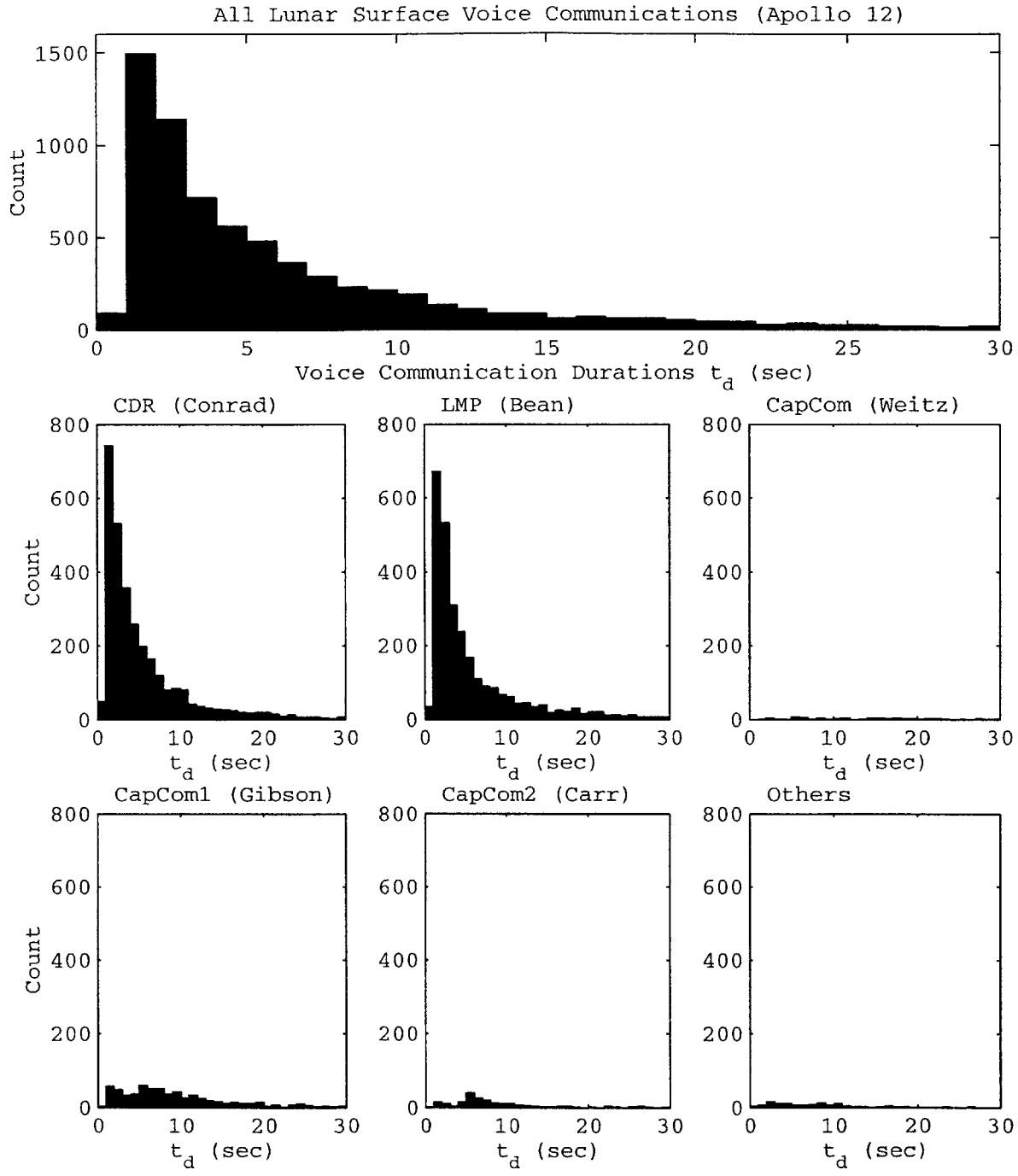


FIGURE B-6. Voice communications durations (upper bounds) for Apollo 12 lunar surface exploration by mission role. Bin sizes are 1 second (accuracy limit from voice transcripts). Voice communication durations greater than 30 seconds are excluded.

TABLE B-4. Voice Communication Analysis (Apollo 12)

Mission Role	# Analyzed	# Corrected	# Not Analyzed	Total	% of Total
CDR	3156	2	0	3156	43.81%
LMP	2843	1	2	2845	39.49%
CapCom	737	0	0	737	10.23%
CMP	118	0	0	118	1.64%
CapCom1	73	0	0	73	1.01%
CapCom2	254	0	0	254	3.53%
Others	21	0	0	21	0.29%
Sub-Total (All)	7202	3	2	7204	100.00%
% of Total	**99.97%**	**0.04%**	**0.03%**	**100.00%**	

TABLE B-5. Time Interval Statistics (Apollo 12)

Mission Role	Mean Interval (seconds)	5% Interval (seconds)	25% Interval (seconds)	75% Interval (seconds)	95% Interval (seconds)
CDR	23.52	2	5	20	74
LMP	19.41	2	4	16	50
CapCom	50.31	7	14	44	172
CapCom1	80.15	6	17	86	297
CapCom2	62.19	7	14	62	248
Others	112	5	15	90	591
All	10.84	1	2	9	32

TABLE B-6. Communication Duration Statistics (Apollo 12)

Mission Role	Mean Duration (seconds)	5% Duration (seconds)	25% Duration (seconds)	75% Duration (seconds)	95% Duration (seconds)
CDR	8.80	1	1	7	26
LMP	7.46	1	2	7	23
CapCom	26.14	2	6	28	69
CapCom1	22.33	1	5	15	66
CapCom2	27.41	1	5	17	95
Others	27.76	1	3	13	131
All	10.84	1	2	9	32

B.4 Apollo 14

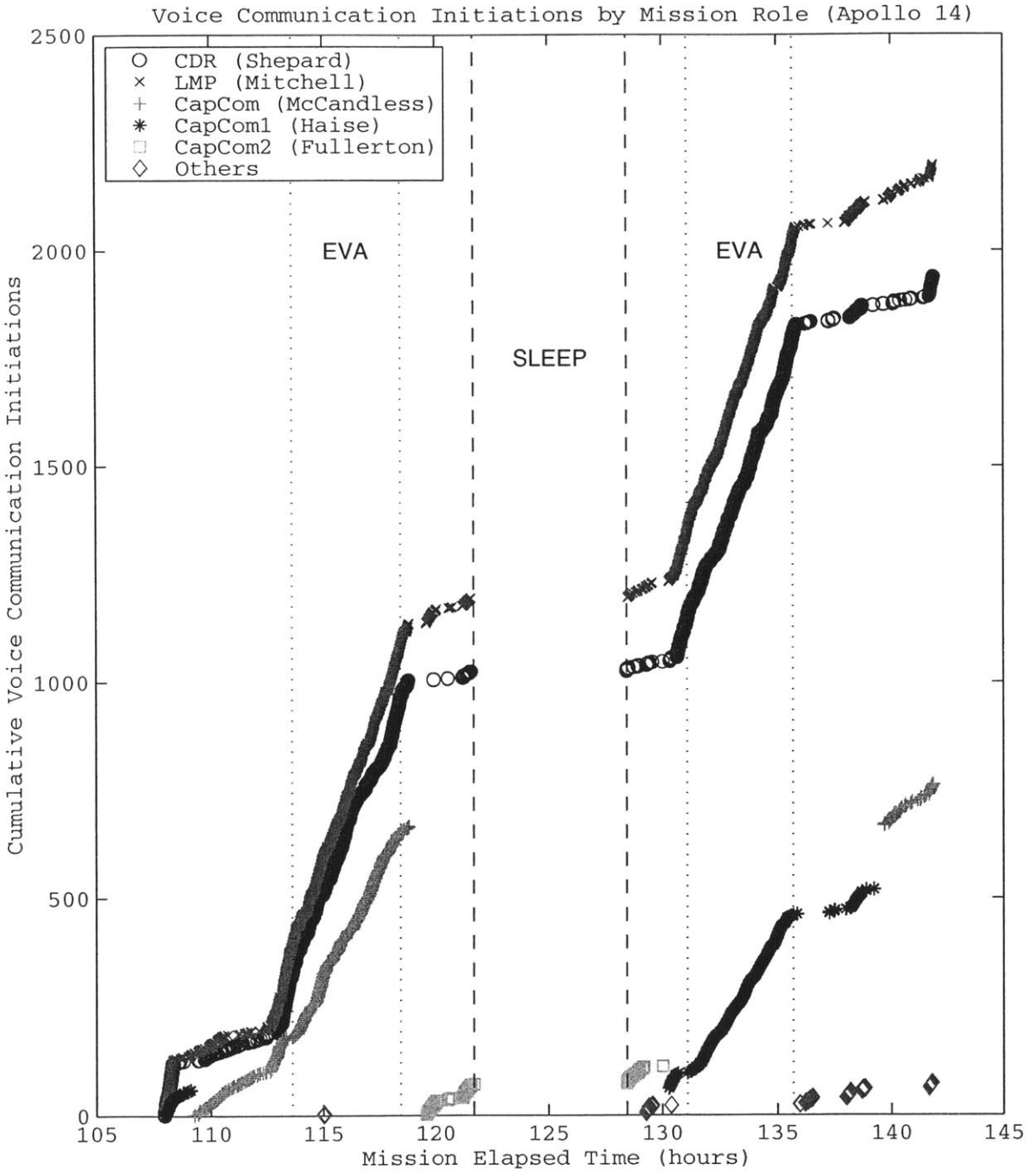


FIGURE B-7. Voice communication initiations for Apollo 14 lunar surface exploration by mission role as a function of mission elapsed time.

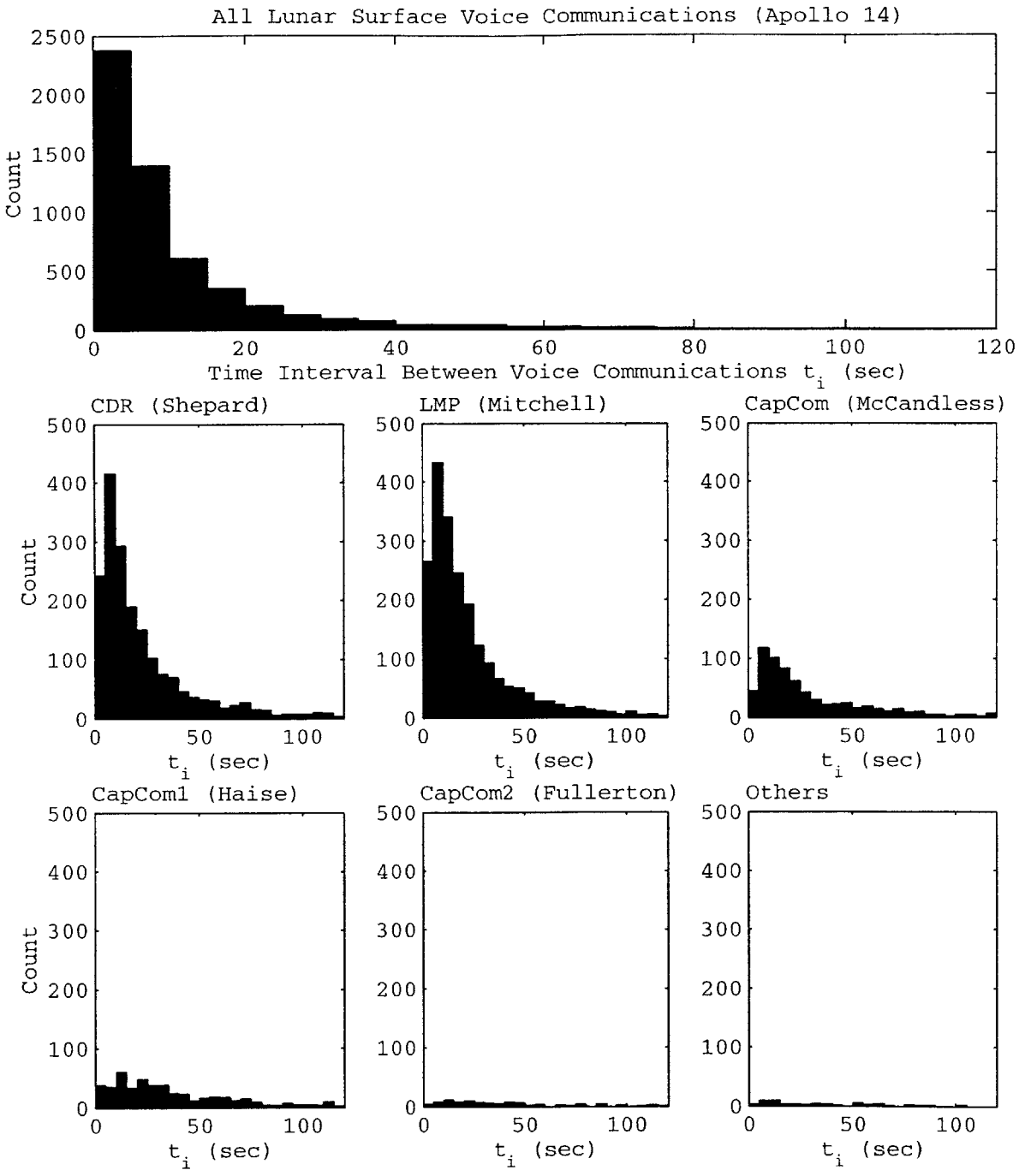


FIGURE B-8. Time intervals between voice communications for Apollo 14 lunar surface exploration by mission role. Bin size is 5 seconds. The few time intervals larger than two minutes are excluded.

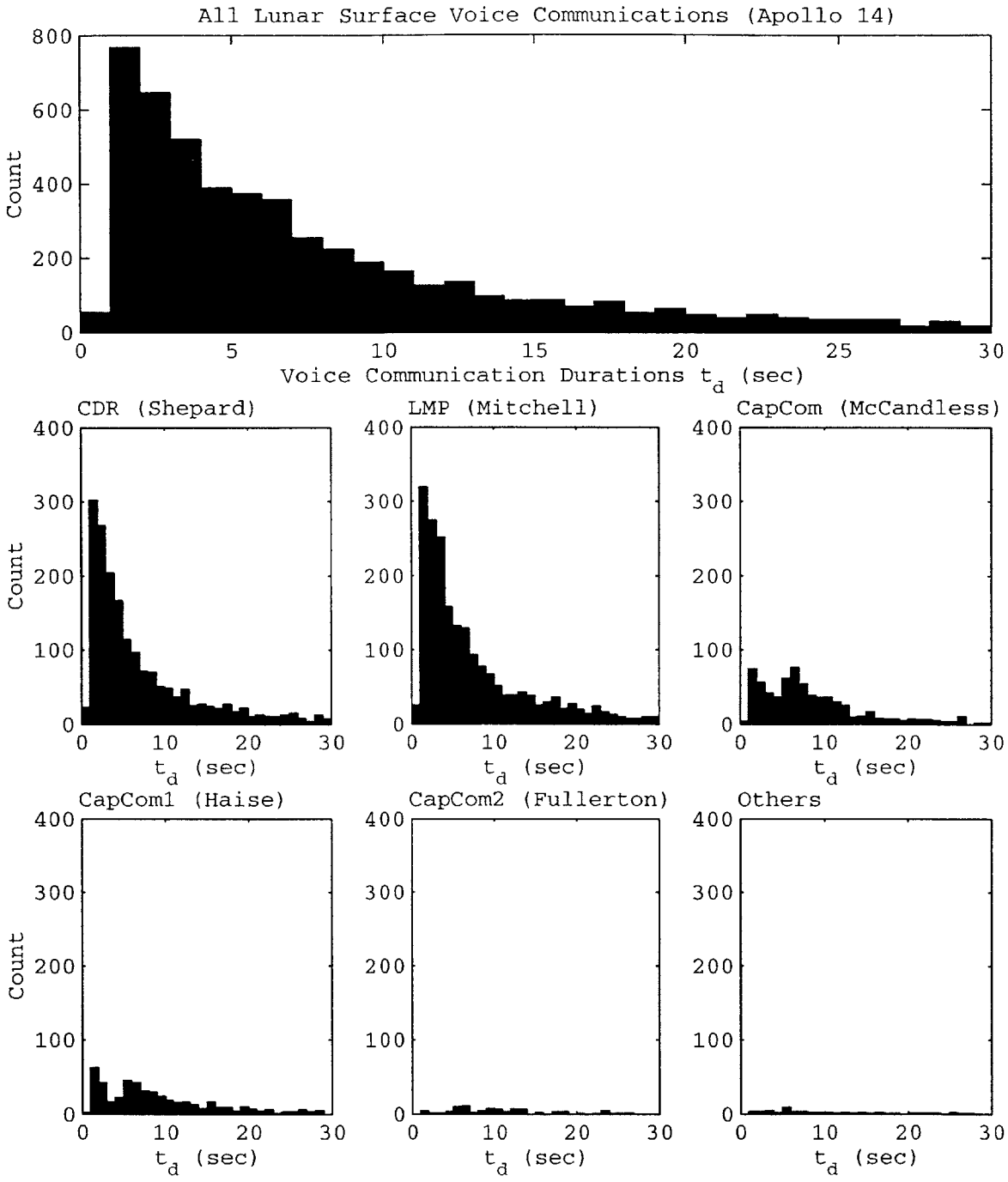


FIGURE B-9. Voice communications durations (upper bounds) for Apollo 14 lunar surface exploration by mission role. Bin sizes are 1 second (accuracy limit from voice transcripts). Voice communication durations greater than 30 seconds are excluded.

TABLE B-7. Voice Communication Analysis (Apollo 14)

Mission Role	# Analyzed	# Corrected	# Not Analyzed	Total	% of Total
CDR	1937	0	0	1937	34.57%
LMP	2196	0	0	2196	39.19%
CapCom	764	0	0	764	13.64%
CMP	11	0	0	11	0.20%
CapCom1	521	0	0	521	9.30%
CapCom2	112	0	0	112	2.00%
Others	61	0	1	62	1.11%
Sub-Total (All)	5602	0	1	5603	100.00%
% of Total	**99.98%**	**0.00%**	**0.02%**	**100.00%**	

TABLE B-8. Time Interval Statistics (Apollo 14)

Mission Role	Mean Interval (seconds)	5% Interval (seconds)	25% Interval (seconds)	75% Interval (seconds)	95% Interval (seconds)
CDR	34.87	3	7	32	113
LMP	34.76	3	8	31	109
CapCom	55.34	4	11	56	221
CapCom1	51.85	3	14	59	160
CapCom2	62.58	5	16	67	263
Others	70.48	4	11	64	229
All	14.79	1	2	12	46

TABLE B-9. Communication Duration Statistics (Apollo 14)

Mission Role	Mean Duration (seconds)	5% Duration (seconds)	25% Duration (seconds)	75% Duration (seconds)	95% Duration (seconds)
CDR	11.19	1	2	11	39
LMP	12.75	1	2	12	40
CapCom	21.38	1	4	13	83
CapCom1	16.87	1	4	14	37
CapCom2	44.17	3	6	24	253
Others	45.07	2	5	27	187
All	14.79	1	2	12	46

B.5 Apollo 15

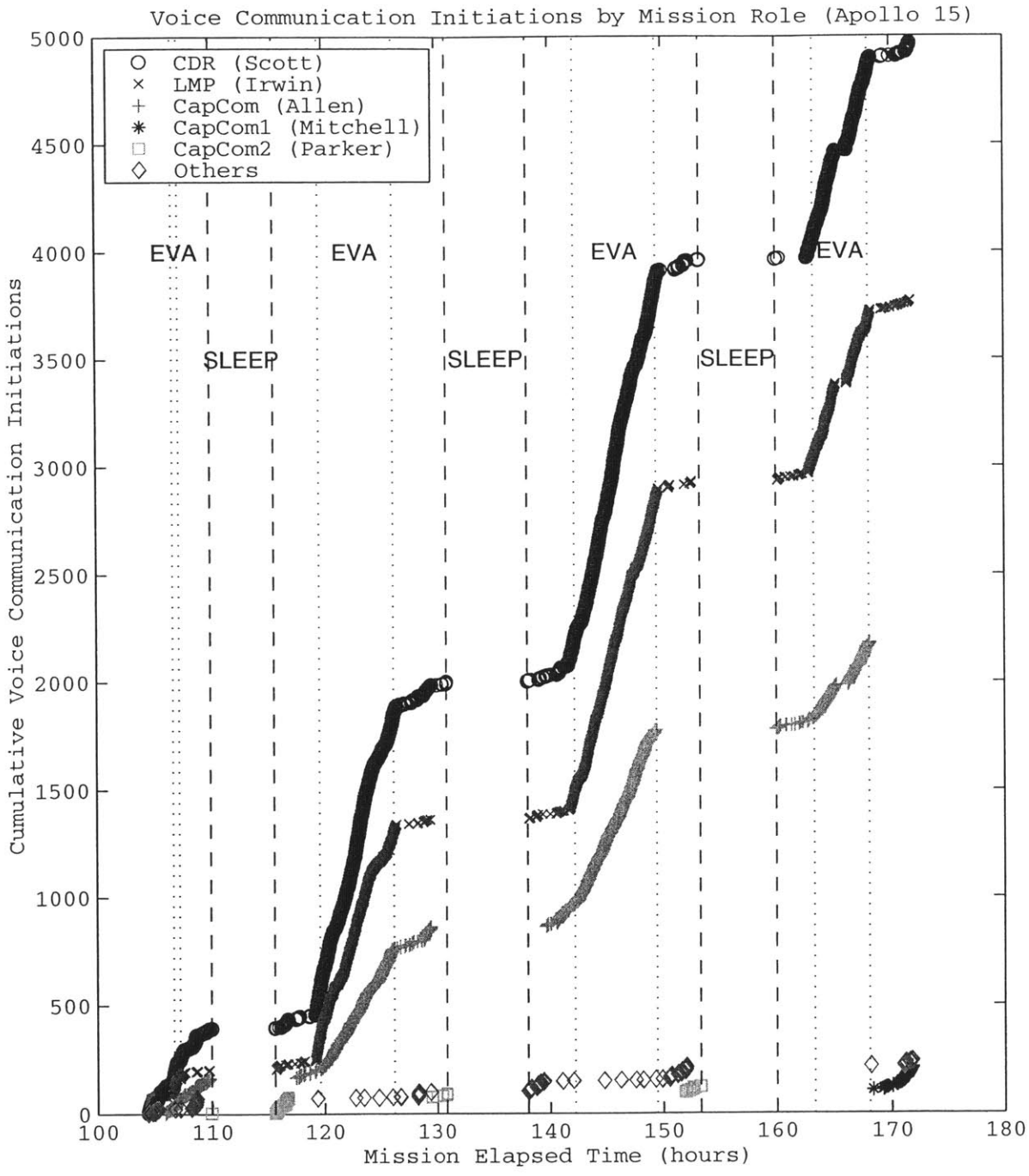


FIGURE B-10. Voice communication initiations for Apollo 15 lunar surface exploration by mission role as a function of mission elapsed time.

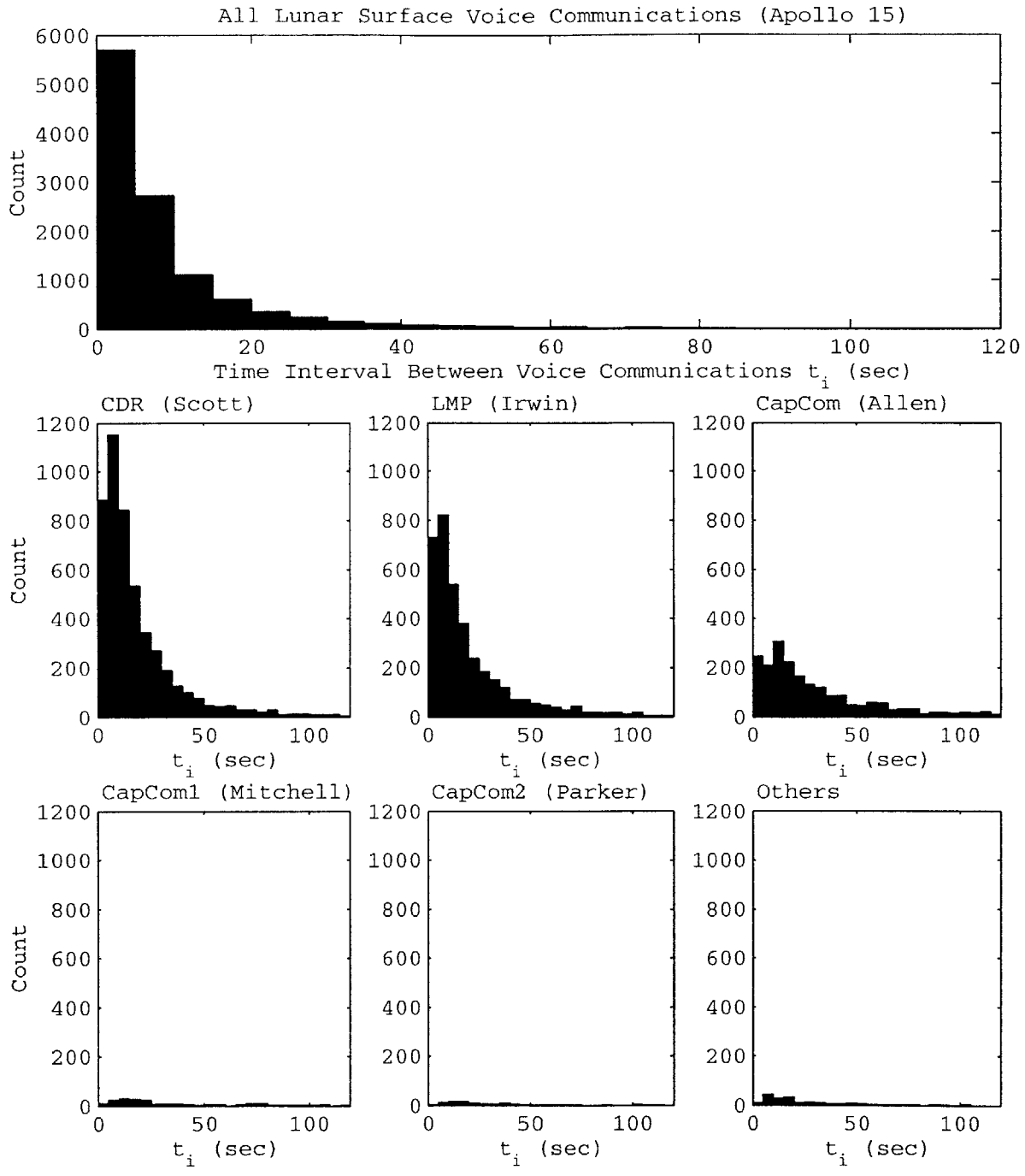


FIGURE B-11. Time intervals between voice communications for Apollo 15 lunar surface exploration by mission role. Bin size is 5 seconds. The few time intervals larger than two minutes are excluded.

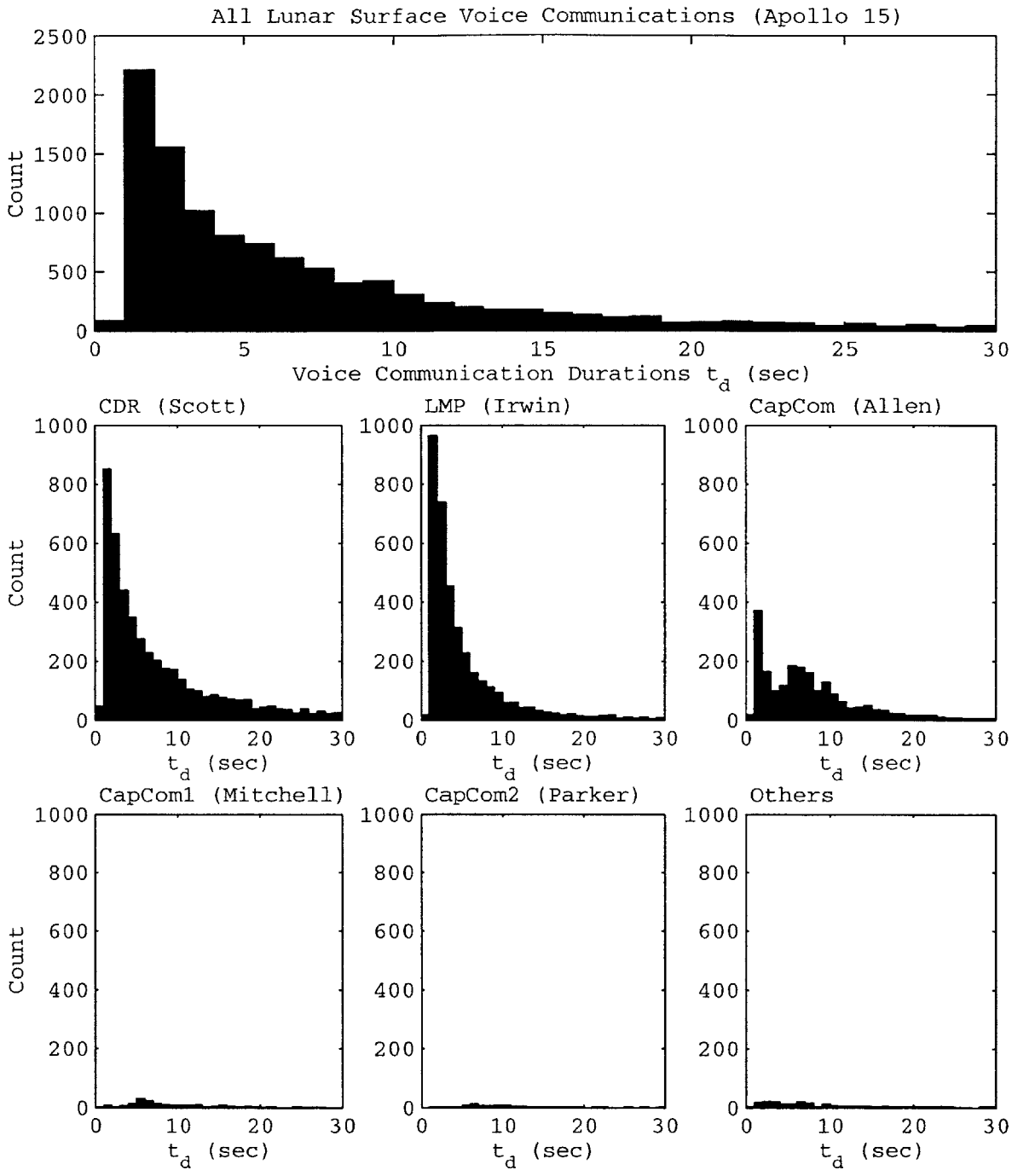


FIGURE B-12. Voice communications durations (upper bounds) for Apollo 15 lunar surface exploration by mission role. Bin sizes are 1 second (accuracy limit from voice transcripts). Voice communication durations greater than 30 seconds are excluded.

TABLE B-10. Voice Communication Analysis (Apollo 15)

Mission Role	# Analyzed	# Corrected	# Not Analyzed	Total	% of Total
CDR	4965	15	4	4969	43.19%
LMP	3769	11	8	3777	32.83%
CapCom	2178	7	0	2178	18.93%
CMP	129	0	0	129	1.12%
CapCom1	212	0	9	221	1.92%
CapCom2	122	0	0	122	1.06%
Others	109	0	0	109	0.95%
Sub-Total (All)	11484	33	21	11505	100.00%
% of Total	**99.82%**	**0.29%**	**0.18%**	**100.00%**	

TABLE B-11. Time Interval Statistics (Apollo 15)

Mission Role	Mean Interval (seconds)	5% Interval (seconds)	25% Interval (seconds)	75% Interval (seconds)	95% Interval (seconds)
CDR	24.70	2	6	24	71
LMP	26.34	2	6	27	83
CapCom	46.11	2	11	50	162
CapCom1	55.70	4	12	67	216
CapCom2	65.77	6	13	55	292
Others	55.30	4	9	45	223
All	11.57	1	2	10	35

TABLE B-12. Communication Duration Statistics (Apollo 15)

Mission Role	Mean Duration (seconds)	5% Duration (seconds)	25% Duration (seconds)	75% Duration (seconds)	95% Duration (seconds)
CDR	12.73	1	2	12	39
LMP	7.16	1	1	6	21
CapCom	12.84	1	2	11	33
CapCom1	27.00	2	5	15	124
CapCom2	32.64	4	7	25	93
Others	21.51	1	3	19	69
All	11.57	1	2	10	35

B.6 Apollo 16

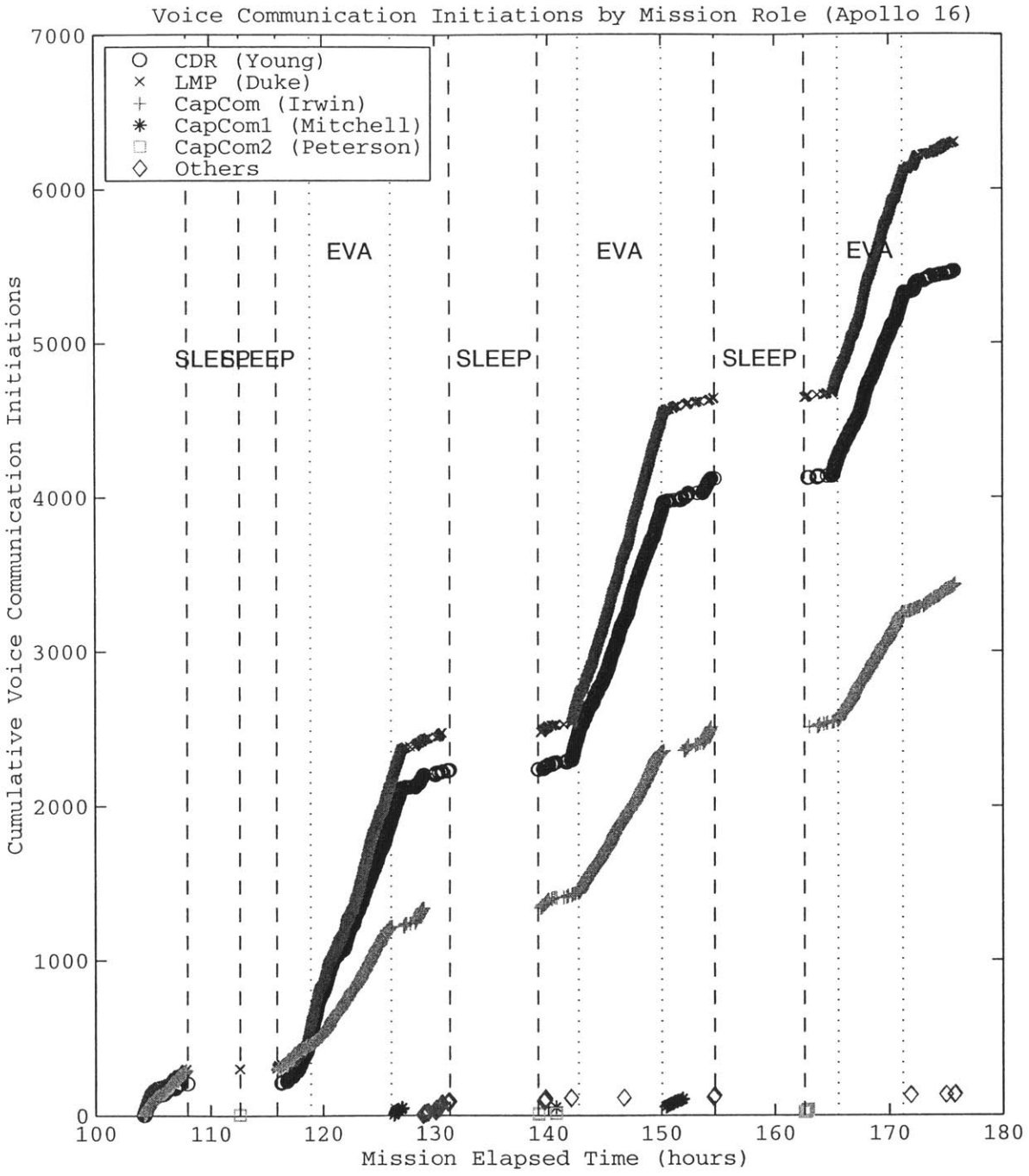


FIGURE B-13. Voice communication initiations for Apollo 16 lunar surface exploration by mission role as a function of mission elapsed time.

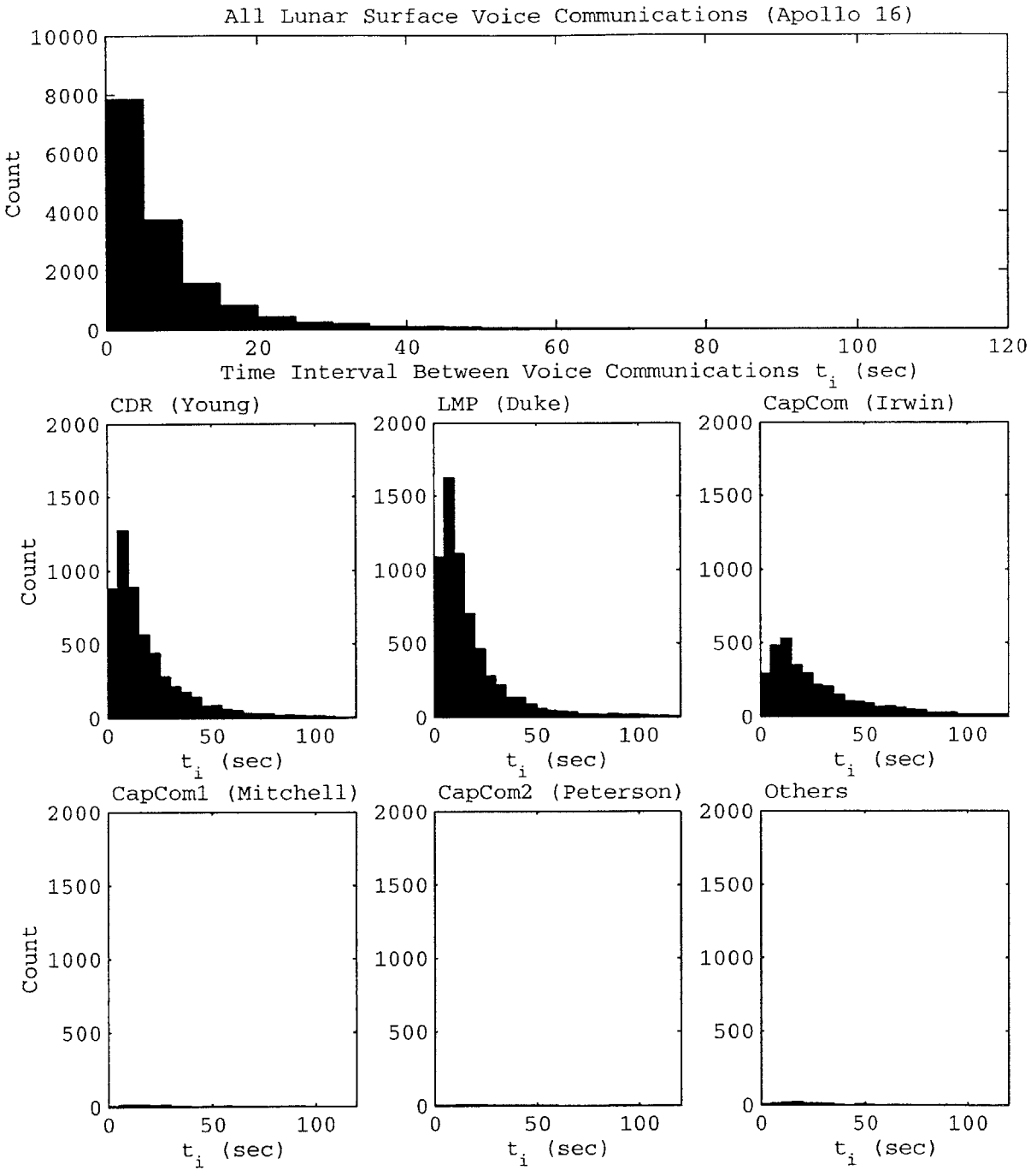


FIGURE B-14. Time intervals between voice communications for Apollo 16 lunar surface exploration by mission role. Bin size is 5 seconds. The few time intervals larger than two minutes are excluded.

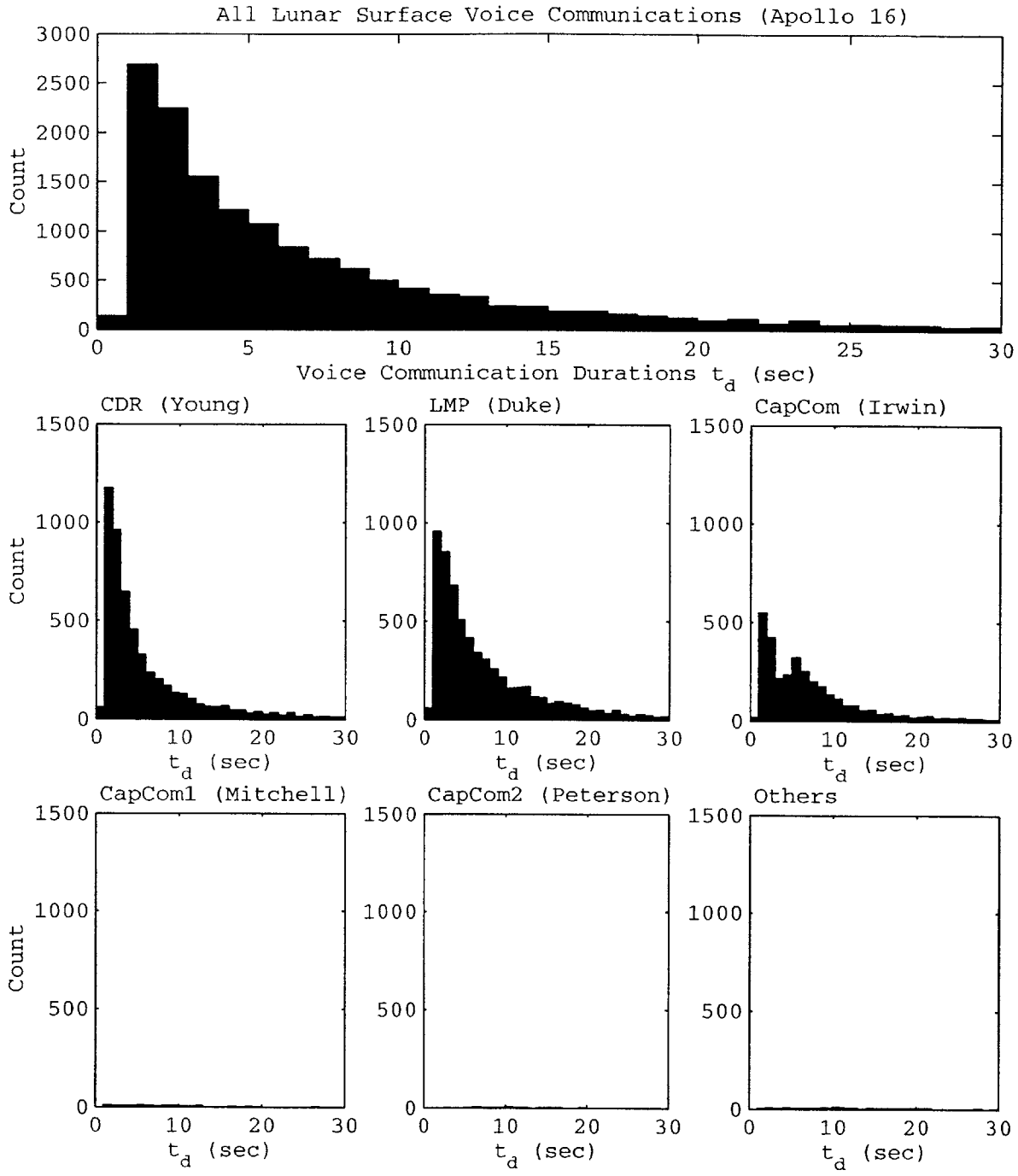


FIGURE B-15. Voice communications durations (upper bounds) for Apollo 16 lunar surface exploration by mission role. Bin sizes are 1 second (accuracy limit from voice transcripts). Voice communication durations greater than 30 seconds are excluded.

TABLE B-13. Voice Communication Analysis (Apollo 16)

Mission Role	# Analyzed	# Corrected	# Not Analyzed	Total	% of Total
CDR	5467	3	1	5468	35.29%
LMP	6304	0	0	6304	40.68%
CapCom	3438	1	0	3438	22.19%
CMP	6	0	0	6	0.04%
CapCom1	106	0	0	106	0.68%
CapCom2	38	0	0	38	0.25%
Others	128	3	8	136	0.88%
Sub-Total (All)	15487	7	9	15496	100.00%
% of Total	**99.4%**	**0.05%**	**0.06%**	**100.00%**	

TABLE B-14. Time Interval Statistics (Apollo 16)

Mission Role	Mean Interval (seconds)	5% Interval (seconds)	25% Interval (seconds)	75% Interval (seconds)	95% Interval (seconds)
CDR	24.11	3	6	25	69
LMP	22.74	2	6	22	64
CapCom	39.69	3	10	43	126
CapCom1	75.90	2	11	46	504
CapCom2	40.97	8	15	41	99
Others	49.53	6	14	36	209
All	10.01	1	2	9	29

TABLE B-15. Communication Duration Statistics (Apollo 16)

Mission Role	Mean Duration (seconds)	5% Duration (seconds)	25% Duration (seconds)	75% Duration (seconds)	95% Duration (seconds)
CDR	7.97	1	2	8	25
LMP	9.24	1	2	10	28
CapCom	12.59	1	2	10	34
CapCom1	37.24	1	4	13	151
CapCom2	58.51	5	7	46	240
Others	28.59	2	7	21	63
All	10.01	1	2	9	29

B.7 Apollo 17

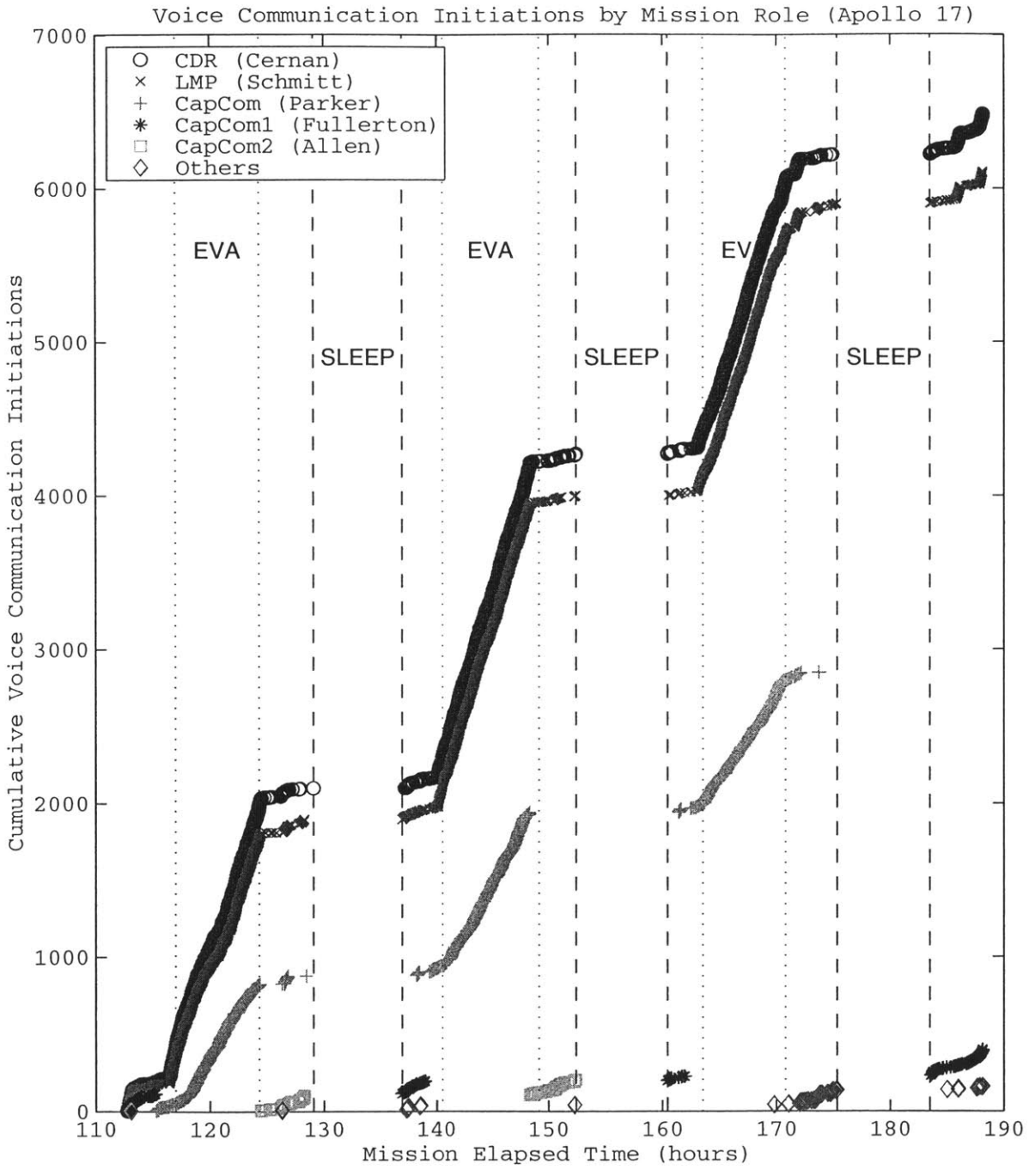


FIGURE B-16. Voice communication initiations for Apollo 17 lunar surface exploration by mission role as a function of mission elapsed time.

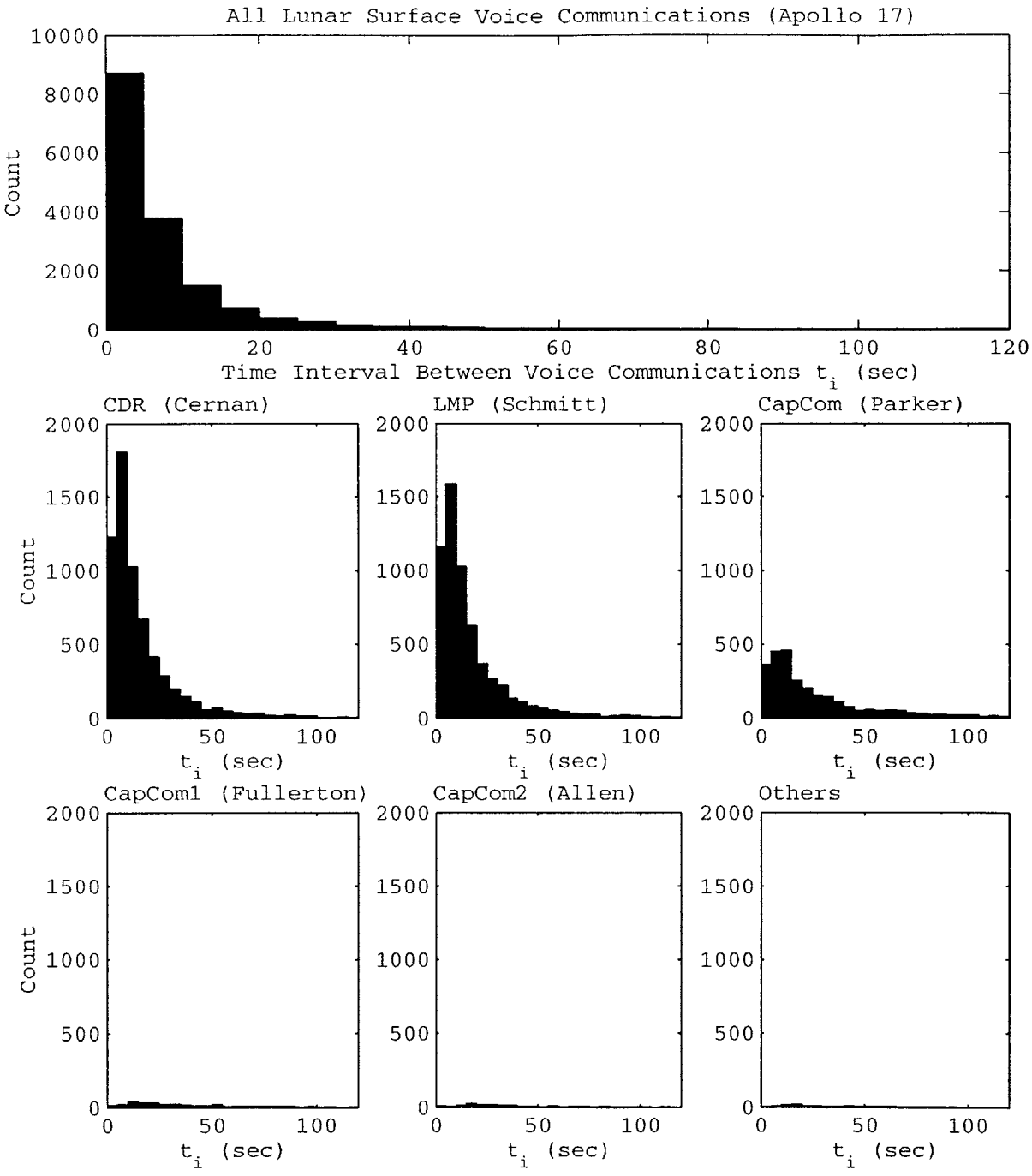


FIGURE B-17. Time intervals between voice communications for Apollo 17 lunar surface exploration by mission role. Bin size is 5 seconds. The few time intervals larger than two minutes are excluded.

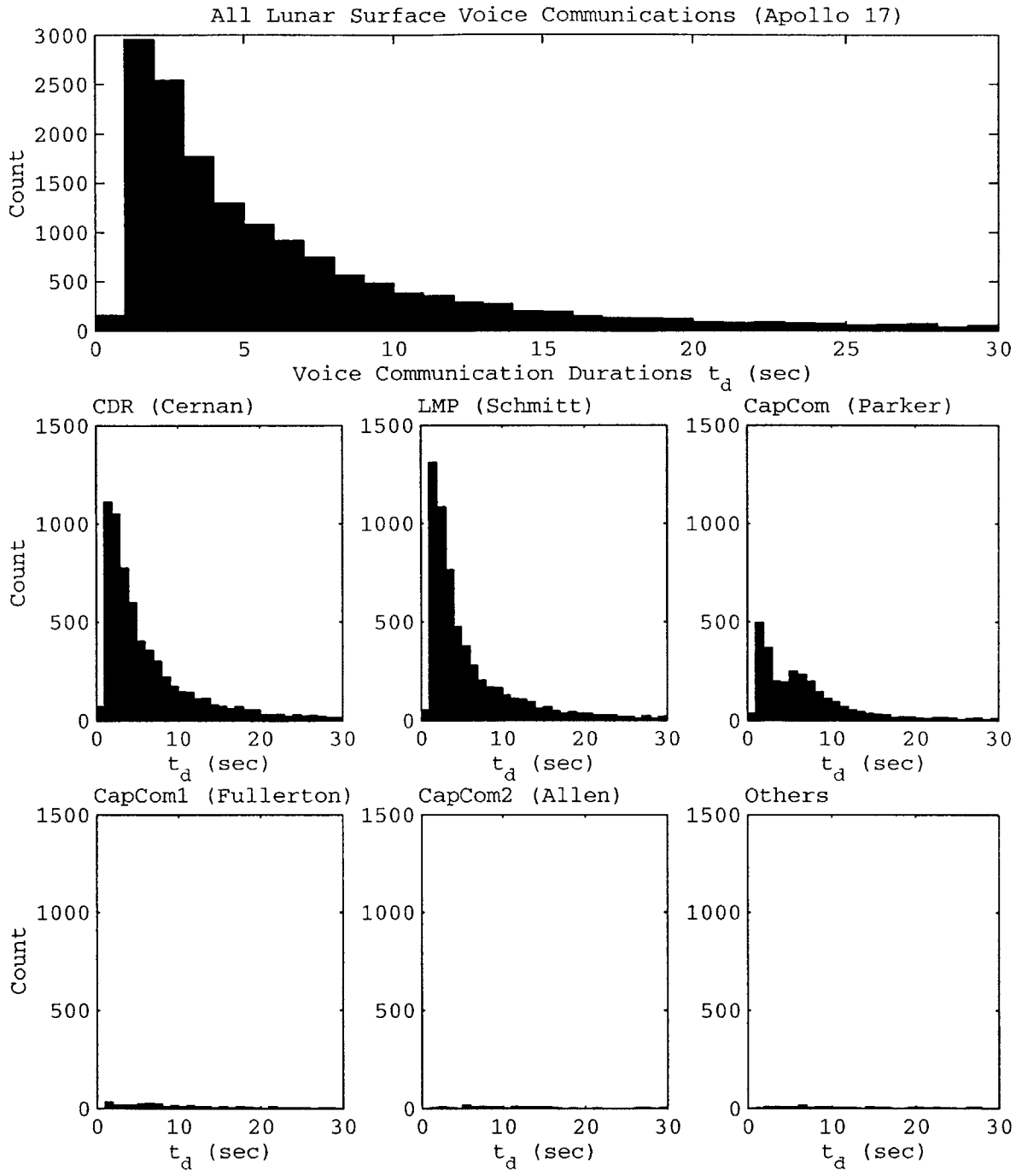


FIGURE B-18. Voice communications durations (upper bounds) for Apollo 17 lunar surface exploration by mission role. Bin sizes are 1 second (accuracy limit from voice transcripts). Voice communication durations greater than 30 seconds are excluded.

TABLE B-16. Voice Communication Analysis (Apollo 17)

Mission Role	# Analyzed	# Corrected	# Not Analyzed	Total	% of Total
CDR	6483	10	27	6510	39.86%
LMP	6108	10	36	6144	37.62%
CapCom	2851	3	0	2851	17.46%
CMP	24	2	0	24	0.15%
CapCom1	396	0	22	418	2.56%
CapCom2	203	0	40	243	1.49%
Others	132	0	8	140	0.86%
Sub-Total (All)	16199	26	133	16332	100.00%
% of Total	**99.19%**	**0.16%**	**0.81%**	**100.00%**	

TABLE B-17. Time Interval Statistics (Apollo 17)

Mission Role	Mean Interval (seconds)	5% Interval (seconds)	25% Interval (seconds)	75% Interval (seconds)	95% Interval (seconds)
CDR	22.16	2	5	21	65
LMP	23.34	2	5	22	68
CapCom	36.43	3	9	39	125
CapCom1	77.06	5	17	73	293
CapCom2	94.86	5	19	80	479
Others	85.82	6	16	59	407
All	10.21	1	2	9	29

TABLE B-18. Communication Duration Statistics (Apollo 17)

Mission Role	Mean Duration (seconds)	5% Duration (seconds)	25% Duration (seconds)	75% Duration (seconds)	95% Duration (seconds)
CDR	8.52	1	2	8	26
LMP	7.75	1	2	8	27
CapCom	9.04	1	2	9	24
CapCom1	38.94	1	5	25	147
CapCom2	69.61	2	7	57	399
Others	49.56	2	5	24	223
All	10.21	1	2	9	29

B.8 Comparison Between Missions

TABLE B-19. Surface Mission Mean Communication Rates

Mission	Mission Start (seconds MET)	Mission End (seconds MET)	Duration (seconds)	Duration (d:hh:mm:ss)	Total Voice Comm. Events	Comm. Rate (comm./min)
Apollo 11	367274	448193	80919	0:22:28:39	2200	1.631
Apollo 12	394997	511597	116600	1:08:23:20	7202	3.706
Apollo 14	388717	510781	122064	1:09:54:24	5602	2.754
Apollo 15	375971	618411	242440	2:19:20:40	11484	2.842
Apollo 16	375189	632936	257747	2:23:35:47	15487	3.605
Apollo 17	406070	677440	271370	3:03:22:50	16199	3.582

(MET=mission elapsed time)

TABLE B-20. Sleep-Corrected Mean Communication Rates

Mission	Sleep Start (seconds MET)	Sleep End (seconds MET)	Sleep Duration (seconds)	Awake Mission Duration (sec.)	Awake Comm Rate (comm./min)
Apollo 11	413577	435544	21967	58952	2.24
Apollo 12	441447	464510	23063	93537	4.62
Apollo 14	438099	462354	24255	97809	3.44
Apollo 15	395916	415839	19923	172503	3.99
	471405	497023	25618		
	551721	576117	24396		
Apollo 16	388815	405165	16350	185046	5.02
	405251	416977	11726		
	473037	500863	27826		
	556987	585512	28525		
Apollo 17	464557	492999	28442	184565	5.27
	48655	577504	28849		
	631085	660599	29514		

TABLE B-21. EVA and Non-EVA Mission Durations

Mission	EVA Start* (seconds MET)	EVA Stop* (seconds MET)	EVA Duration (seconds)	Mission EVA Duration (sec.)	Awake, Non-EVA Duration (sec.)
Apollo 11	392162	402088	9926	9926	49026
Apollo 12	414482	428945	14463	28473	65064
	473367	487377	14010		
Apollo 14	409009	426361	17352	33885	63924
	471992	488525	16533		
Apollo 15	384080	386092	2012	68851	103652
	430626	454234	23608		
	511999	537974	25975		
	587899	605155	17256		
Apollo 16	427920	453830	25910	72948	112098
	513486	540115	26629		
	595828	616237	20409		
Apollo 17	421253	447220	25967	83194	101371
	505969	537069	31100		
	588705	614832	26127		

*** EVA start and stop are defined here roughly by depressurization and repressurization of the Lunar Module. The start and stop times given here are within a few minutes of the official NASA start and stop times, defined by the cabin pressure crossing the threshold of 3.5 pounds per square inch.**

TABLE B-22. EVA and Non-EVA Mean Communication Rates

Mission	Condition	5% Confidence Bound (comms./minute)	Mean (comms./minute)	95% Confidence Bound (comms./minute)
Apollo 11	EVA	3.67	3.74	3.81
	Non-EVA	2.19	2.21	2.23
Apollo 12	EVA	9.94	10.06	10.17
	Non-EVA	2.90	2.93	2.95
Apollo 14	EVA	6.13	6.20	6.26
	Non-EVA	2.57	2.59	2.61
Apollo 15	EVA	7.78	7.84	7.90
	Non-EVA	2.57	2.59	2.61
Apollo 16	EVA	8.91	8.98	9.04
	Non-EVA	3.33	3.35	3.37
Apollo 17	EVA	8.97	9.03	9.09
	Non-EVA	2.69	2.71	2.72

Note: Confidence bounds computed by treating voice communications "arrivals" as a Poisson process. A significance level of 0.05 was used.

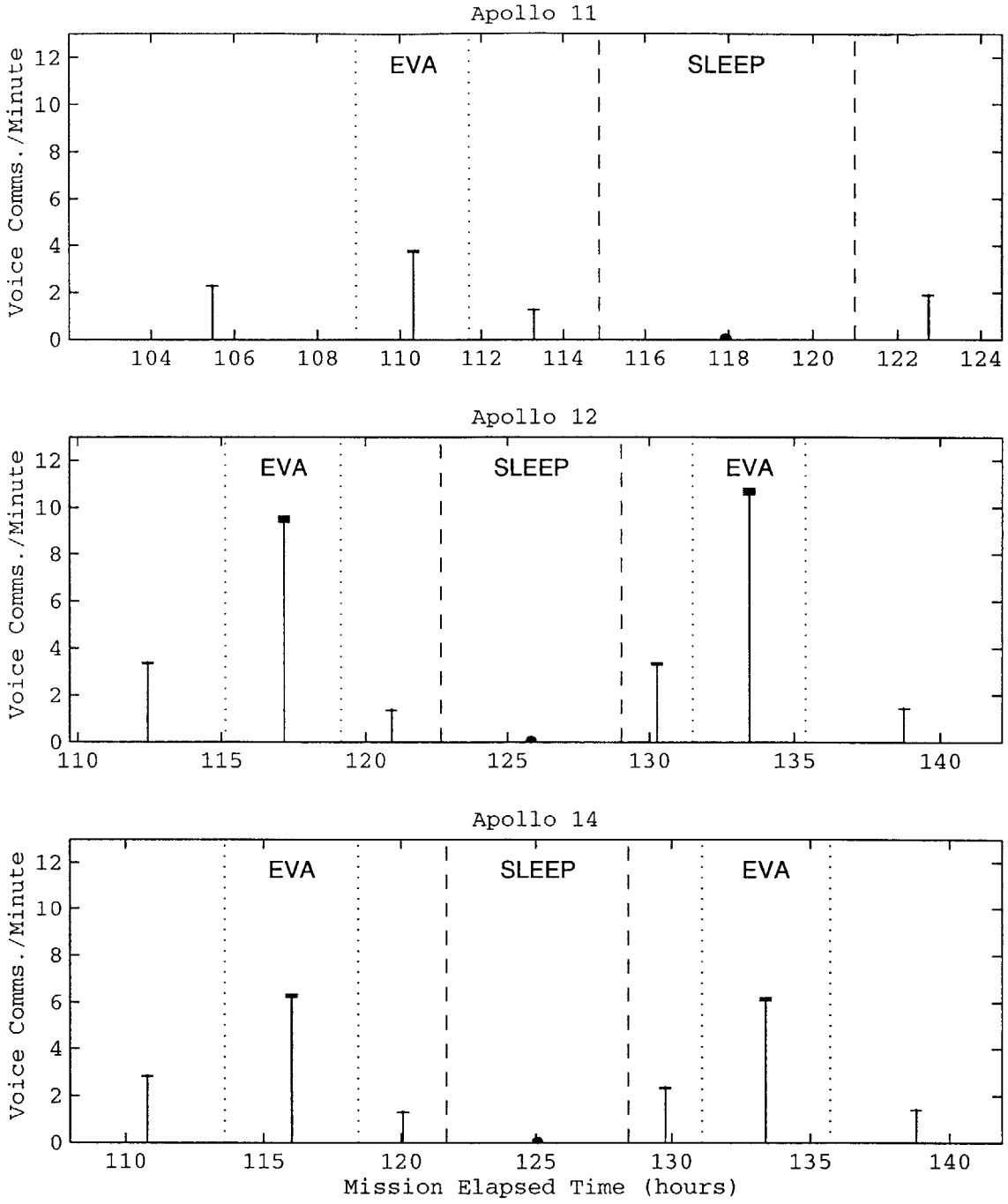


FIGURE B-19. Mean communication rates during extravehicular activity and non-extravehicular activity during Apollo 11, 12, and 14 as a function of Mission Elapsed Time. Bar heights indicate 95% confidence intervals, computed by treating voice communication "arrivals" as a Poisson process.

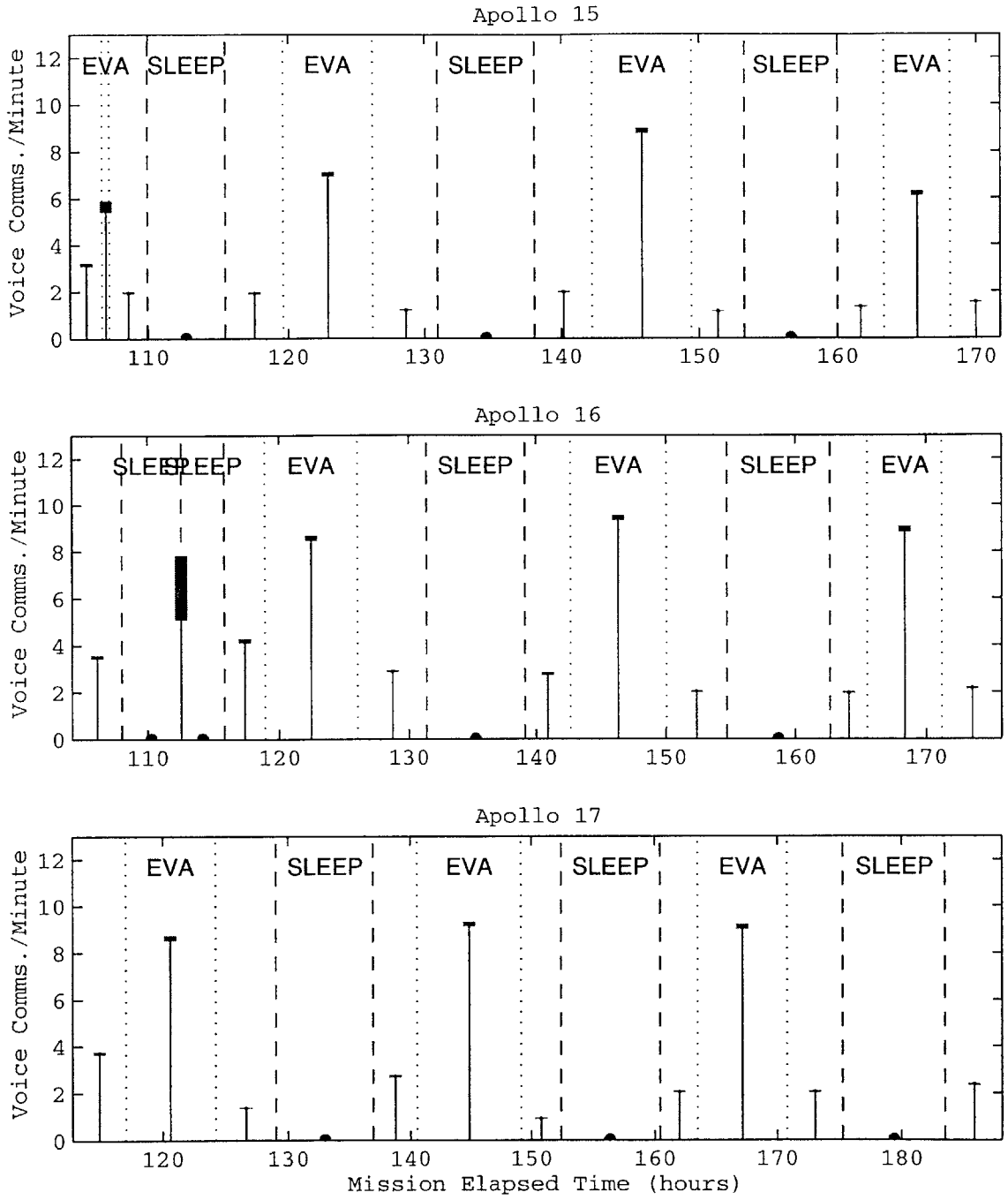


FIGURE B-20. Mean communication rates during extravehicular activity and non-extravehicular activity during Apollo 15, 16, and 17 as a function of Mission Elapsed Time. Bar heights indicate 95% confidence intervals, computed by treating voice communication "arrivals" as a Poisson process.

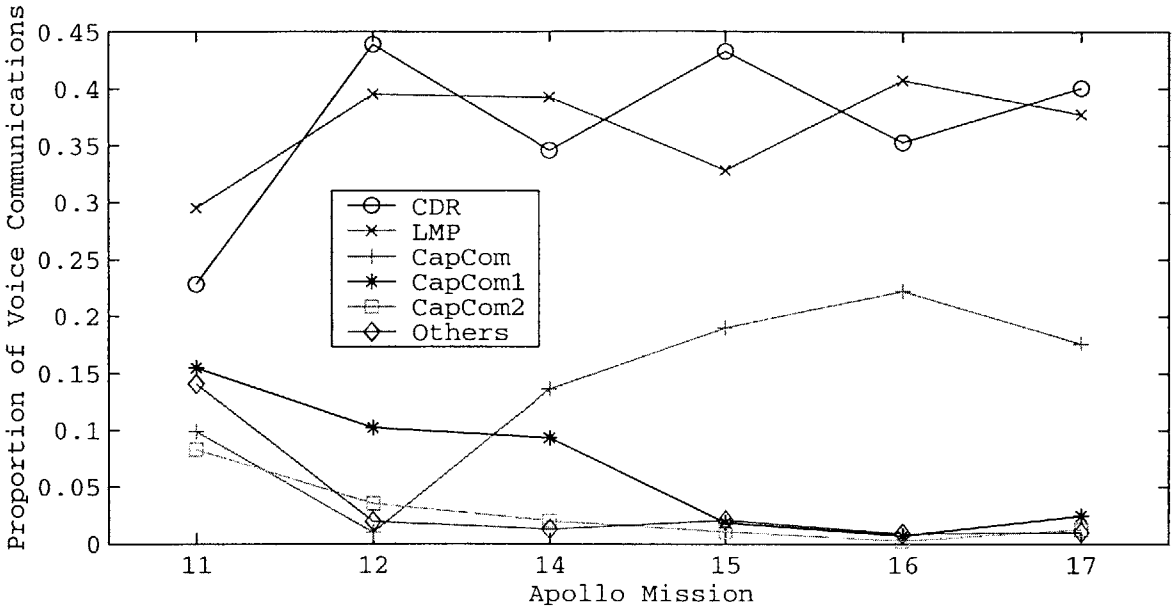


FIGURE B-21. Proportion of voice communications as a function of mission role for all of the Apollo lunar surface missions.

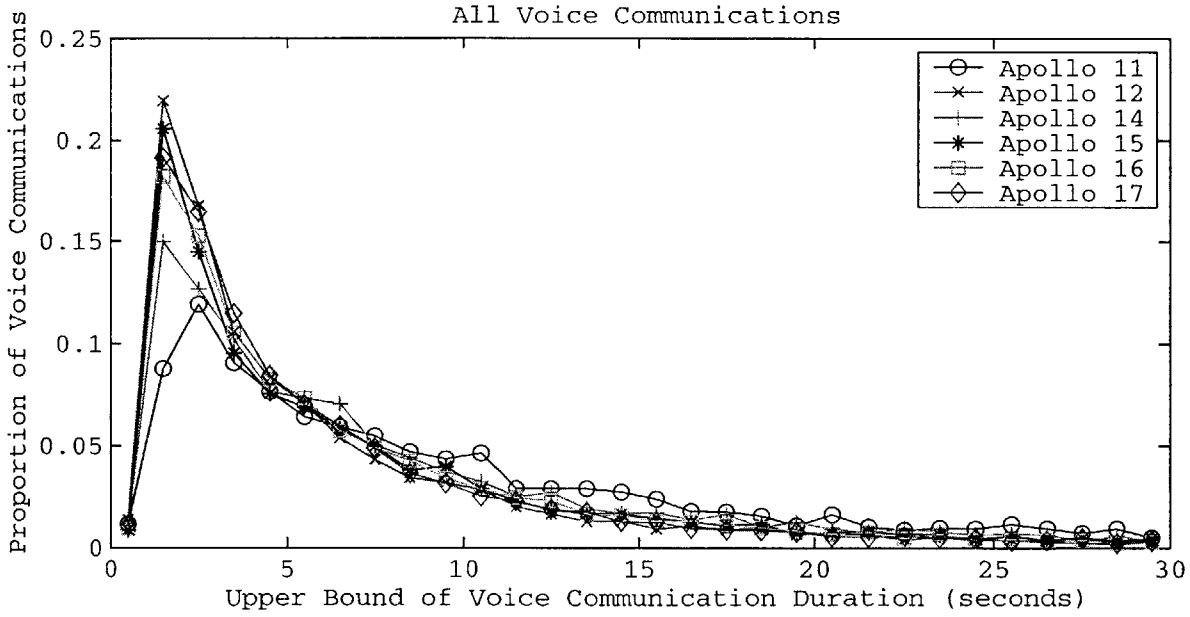


FIGURE B-22. Proportion of voice communications as a function of communication duration for all of the Apollo lunar surface missions.

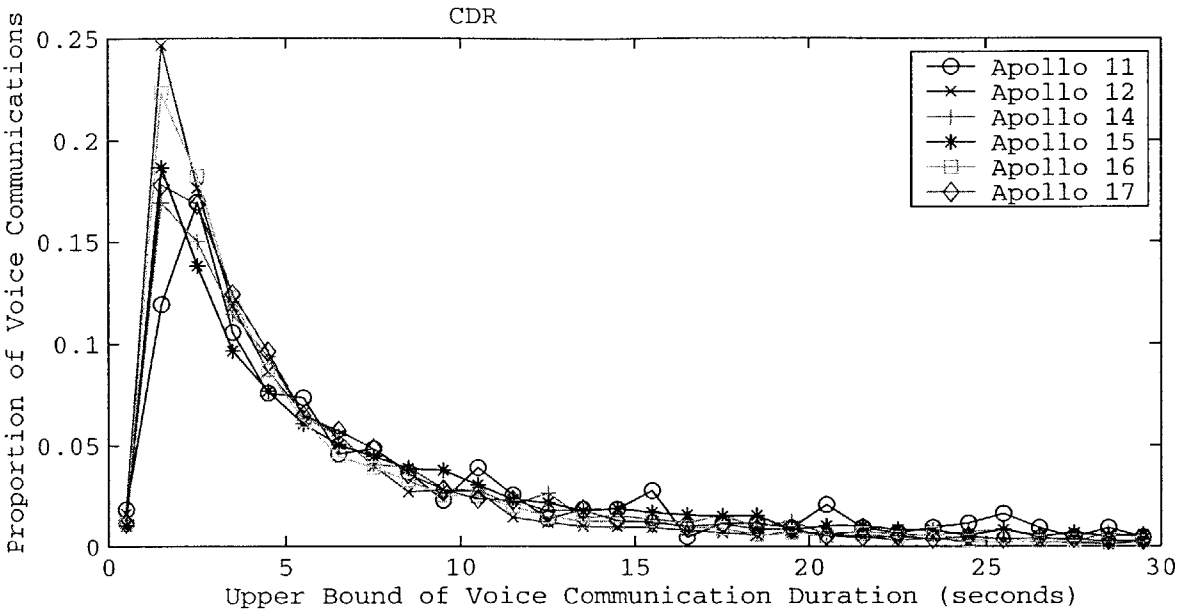


FIGURE B-23. Proportion of voice communications as a function of communication duration for the Commanders of the Apollo lunar surface missions.

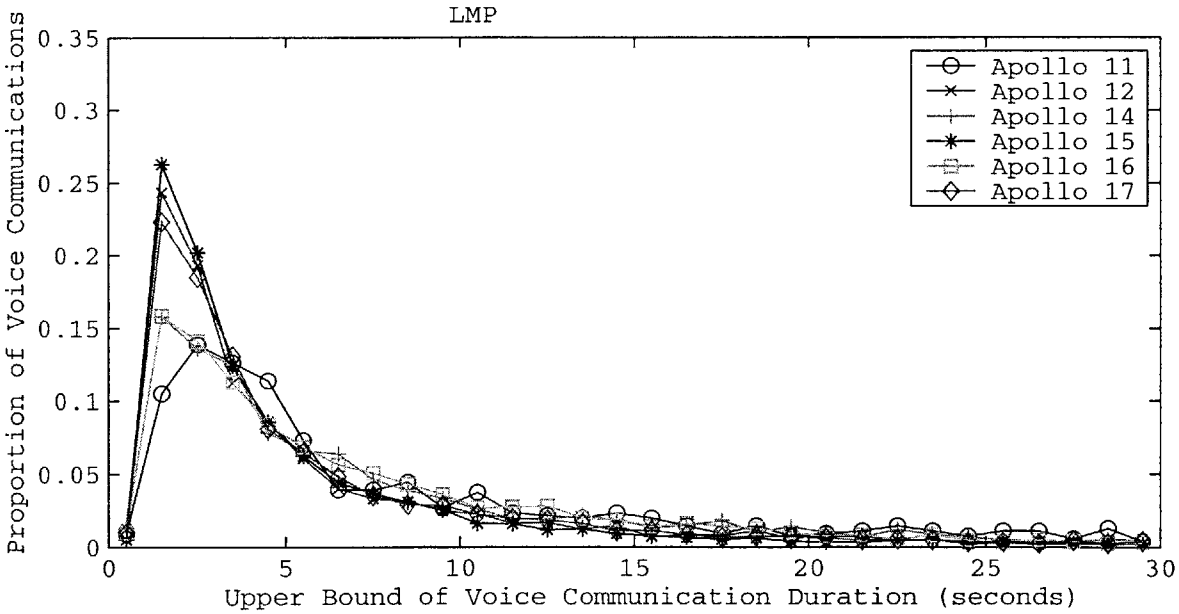


FIGURE B-24. Proportion of voice communications as a function of communication duration for the Lunar Module Pilots of the Apollo lunar surface missions.

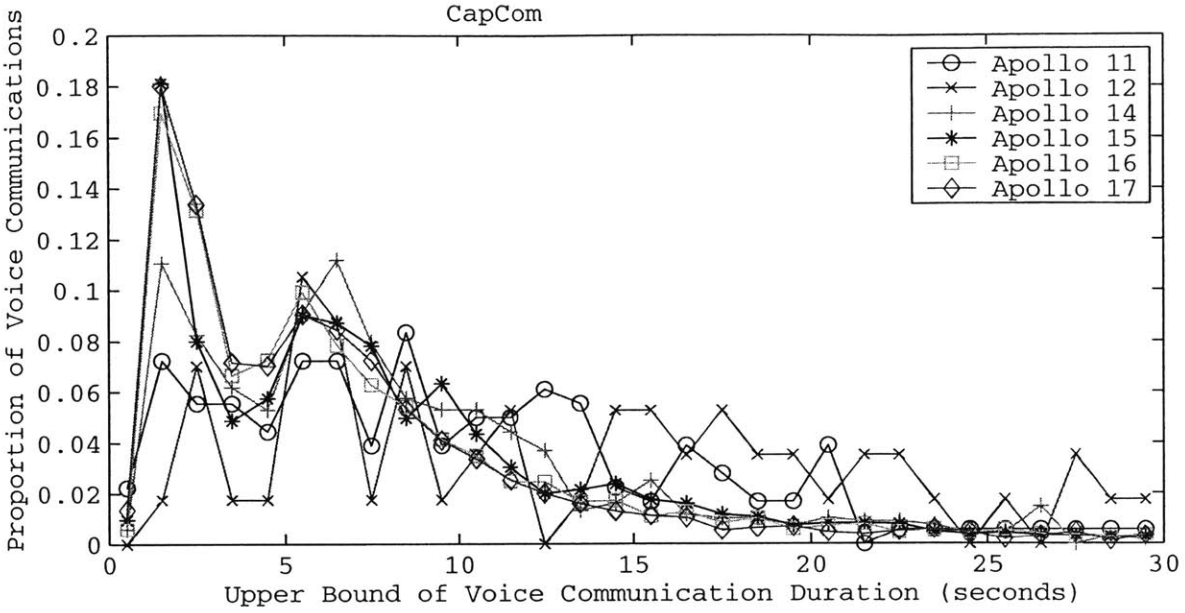


FIGURE B-25. Proportion of voice communications as a function of communication duration for the CapComs of the Apollo lunar surface missions.

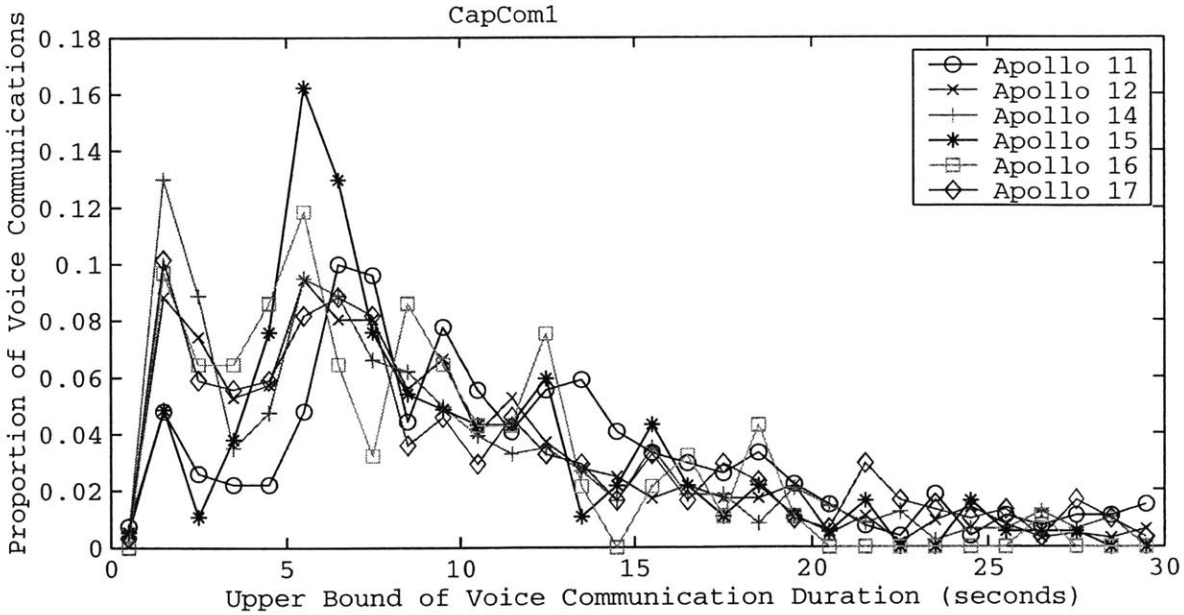


FIGURE B-26. Proportion of voice communications as a function of communication duration for the "CapCom1s" of the Apollo lunar surface missions.

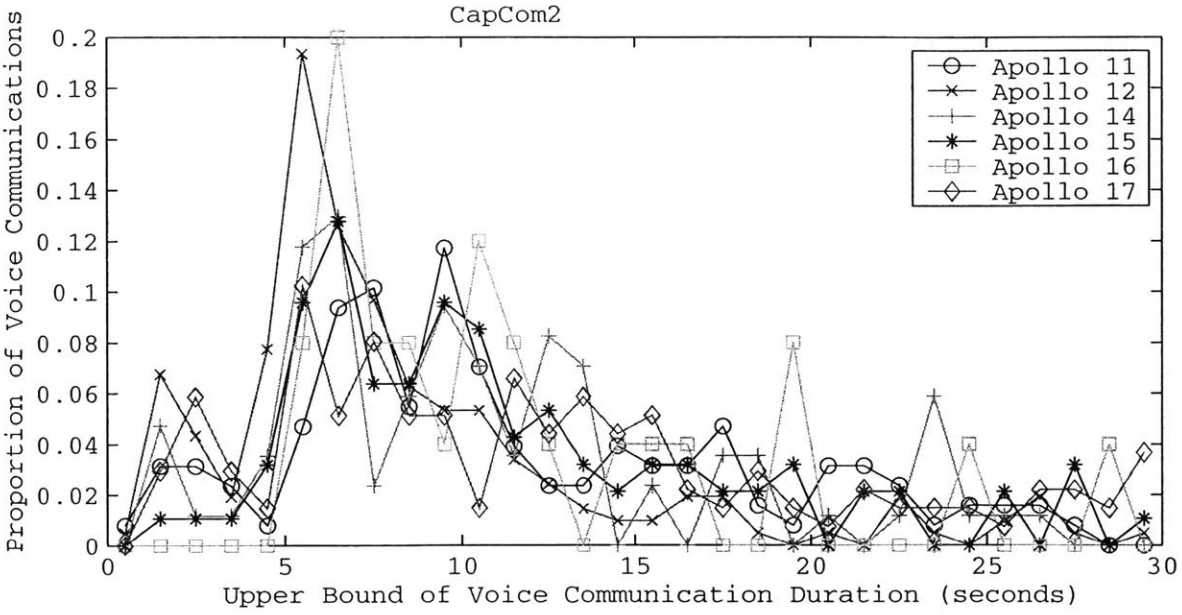


FIGURE B-27. Proportion of voice communications as a function of communication duration for the "CapCom2s" of the Apollo lunar surface missions.

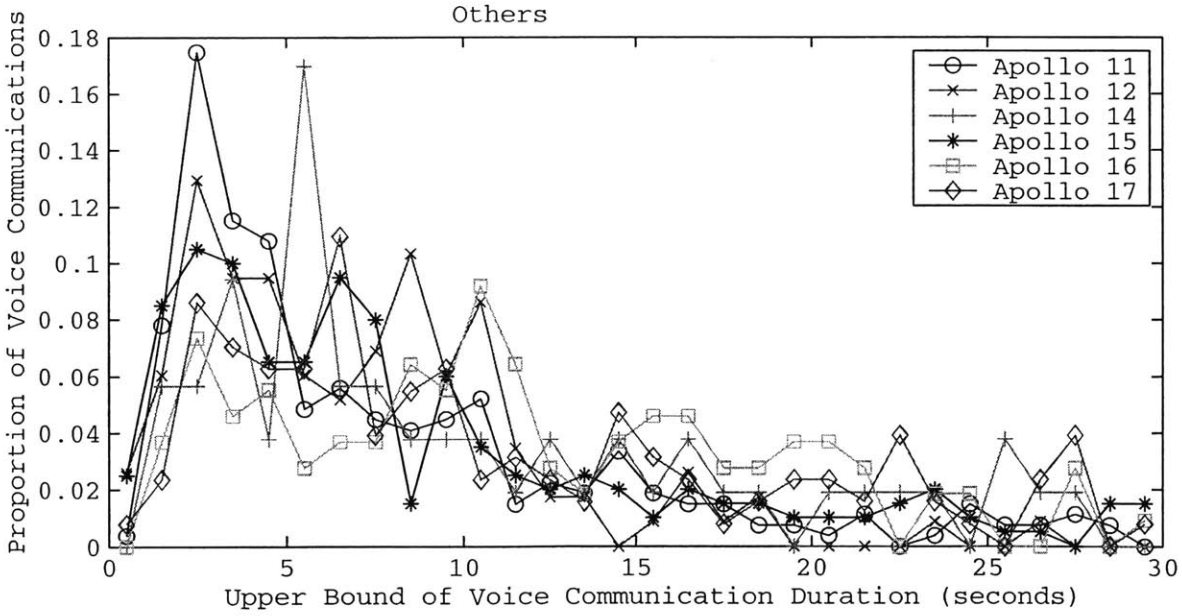


FIGURE B-28. Proportion of voice communications as a function of communication duration for "Others" (not CDR, LMP, or a CapCom) of the Apollo lunar surface missions.

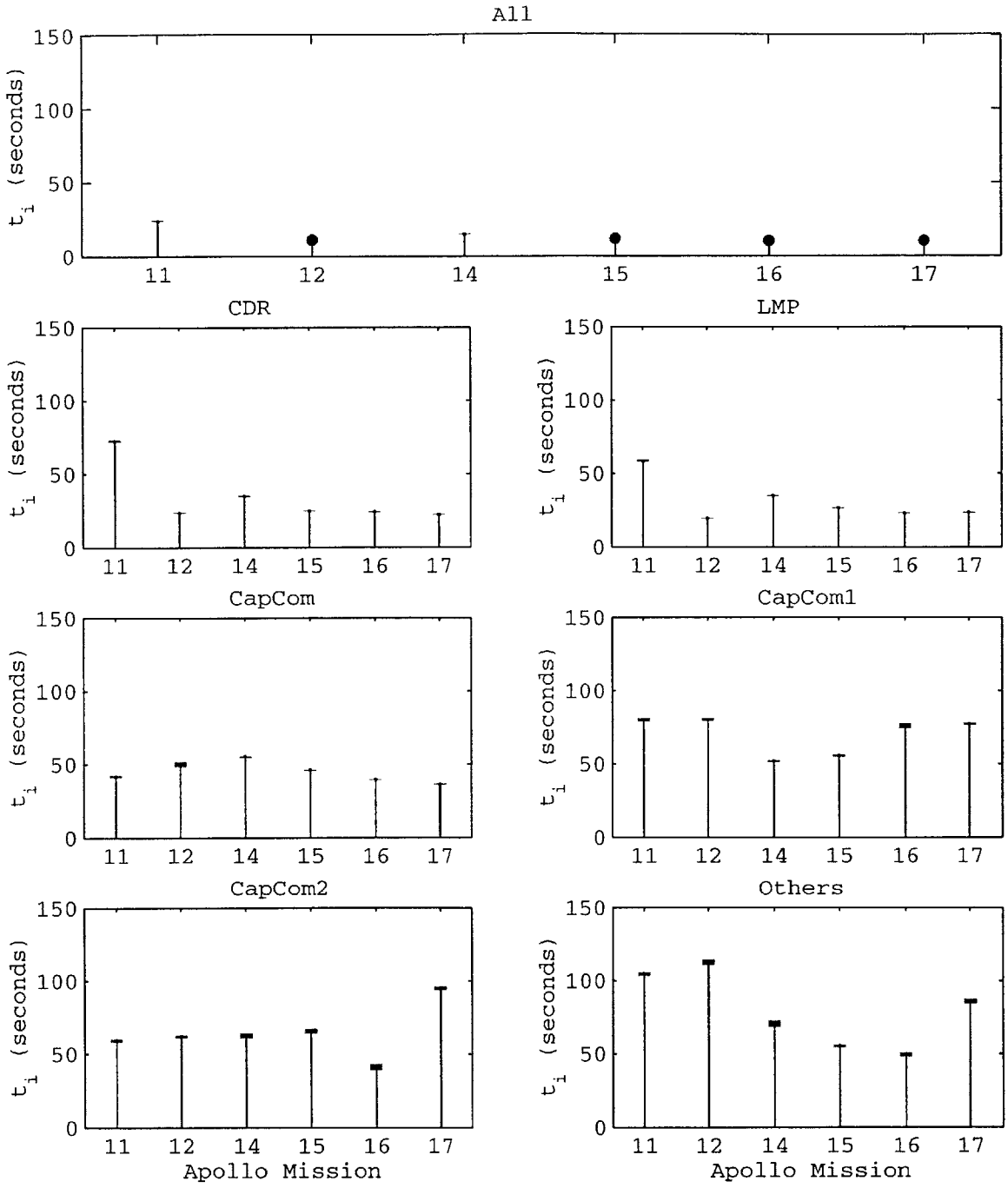


FIGURE B-29. Mean time intervals between voice communications for different mission roles as a function of the Apollo lunar surface mission. Bar heights correspond to a 95% confidence interval as estimated by treating voice communications of a given mission role as a Poisson process.

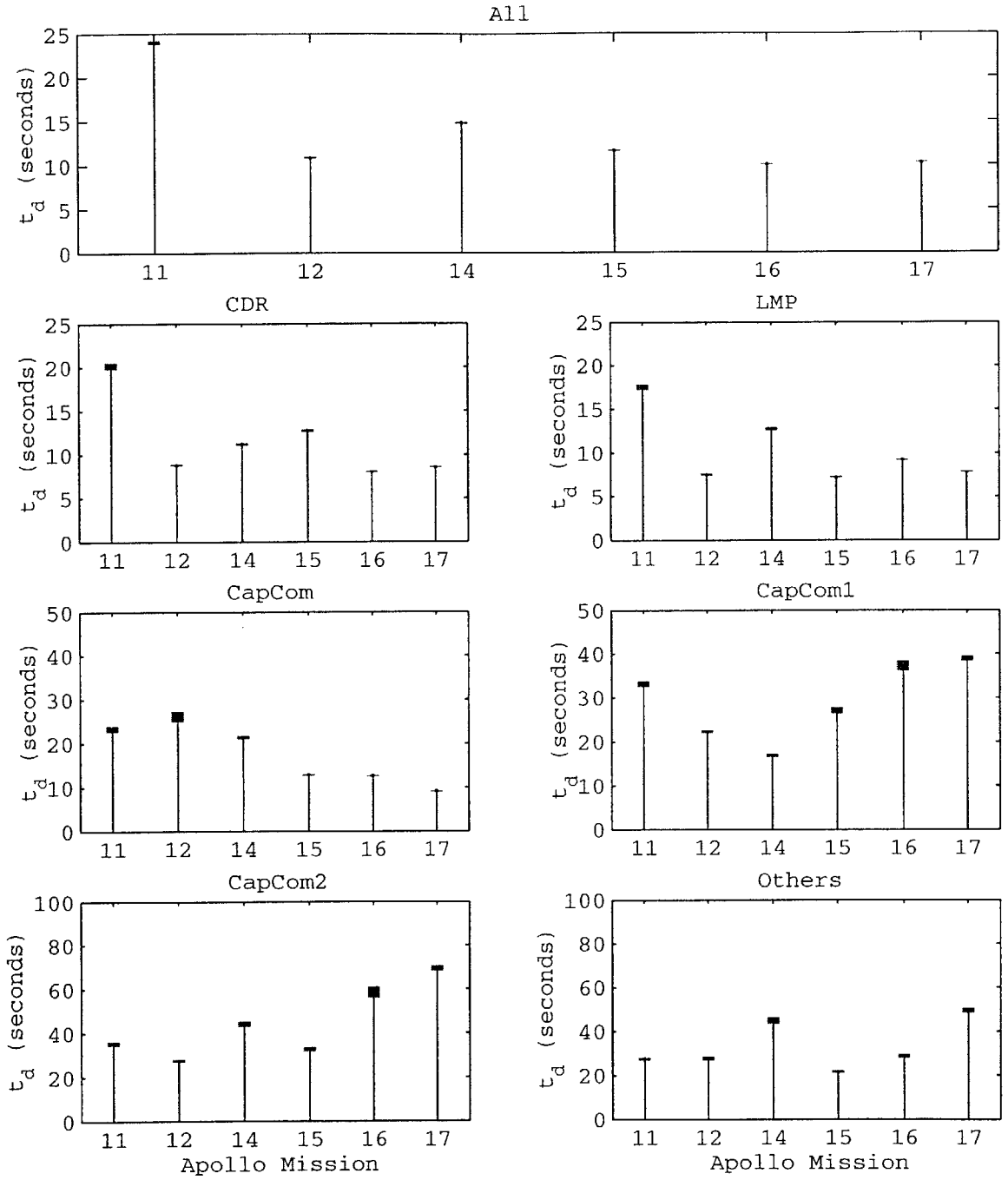


FIGURE B-30. Mean voice communication durations for different mission roles as a function of the Apollo lunar surface mission. Bar heights correspond to a 95% confidence interval as estimated by treating all voice communications collectively as a Poisson process.

'The time has come,' the Walrus said,
'To talk of many things;
Of shoes - and ships - and sealing wax -
Of cabbages - and kings -
And why the sea is boiling hot -
And whether pigs have wings.'

'But wait a bit,' the Oysters cried,
'Before we have our chat;
For some of us are out of breath,
And all of us are fat!'
'No hurry!' said the Carpenter.
They thanked him much for that.

'A loaf of bread,' the Walrus said,
'Is what we chiefly need:
Pepper and vinegar besides
Are very good indeed -
Now if you're ready, Oysters dear,
We can begin to feed.'

'But not on us!' the Oysters cried,
Turning a little blue.
'After such kindness, that would be
A dismal thing to do!'
'The night is fine,' the Walrus said.
'Do you admire the view?

It was so kind of you to come!
And you are very nice!'
The Carpenter said nothing, but
'Cut us another slice:
I wish you were not quite so deaf -
I've had to ask you twice!'

'It seems a shame,' the Walrus said,
'To play them such a trick
After we've brought them out so far,
And made them trot so quick!'
The Carpenter said nothing but
'The butter's spread too thick!'

Lewis Carroll, *"The Walrus and the Carpenter," Through the Looking-Glass and What Alice Found There (1872)*

Appendix C - Data Sources

C.1 Digital Elevation Models

The name, components, and source of the digital elevation models used in this thesis are given in Table C-1.

TABLE C-1. Digital Elevation Model (DEM) Sources

Name	Comments	Source Files	Source
Crater Lake, California	Available online in DEM or ARC/INFO formats.	DEM composed of twelve 7.5 minute quadrangles.	http://craterlake.wr.usgs.gov/dem_download.shtml
Cottonwood, Nevada	Freely available online.	Cottonwood, Nevada 7.5 minute quadrangle.	http://www.gisdatadepot.com/dem/
Soda Lake, California	Individual DEMs available online; merged in ArcView.	Multiple 7.5 minute quadrangles: Soda Lake North, Soda Lake South, Seventeen Mile Point, and Cowhole Mountain.	http://www.gisdatadepot.com/dem/
Kelso Dunes, California	Individual DEMs available online; merged in ArcView.	Multiple 7.5 minute quadrangles: Old Dad Mountain, Glasgow, Kelso, and Kelso Dunes.	http://www.gisdatadepot.com/dem/
Apollo 14 Landing Site	Contours and craters digitized by hand in ArcView; see below.	USGS Lunar Map of Fra Mauro Highlands [Swann et al., Plate 2].	http://www.hq.nasa.gov/office/pao/History/alsj/a14/a14-usgs.jpg.

Digital elevation models were imported into ArcView, exported in ASCII text format, and preprocessed into a slightly different text format that could then be easily imported into MATLAB.

Apollo 14 DEM Creation and Validation. Two versions of the Apollo 14 DEM were created, through different sampling methods of altitude control points. Figure C-1 shows an early attempt to define these altitude control points by heavy sampling along the ten meter contours of the georeferenced USGS map of the Fra Mauro Highlands, while Figure C-2 shows a more refined approach: more even sampling, and an attempt to define many of the larger craters on the map. Crater rim heights and center depths were generally unknown, but an attempt was made to make the crater rim-heights and center-depths consistent with some of the available lunar surface photography or astronaut descriptions. The resulting set of altitude control points was converted to an evenly sampled grid (with 5 meter cell-spacing) using the spline interpolation feature of ArcView.

FIGURE C-1. Selecting altitude control points for the Apollo 14 digital elevation model based on highly sampled ten meter contours intervals produced a poor quality digital elevation model.

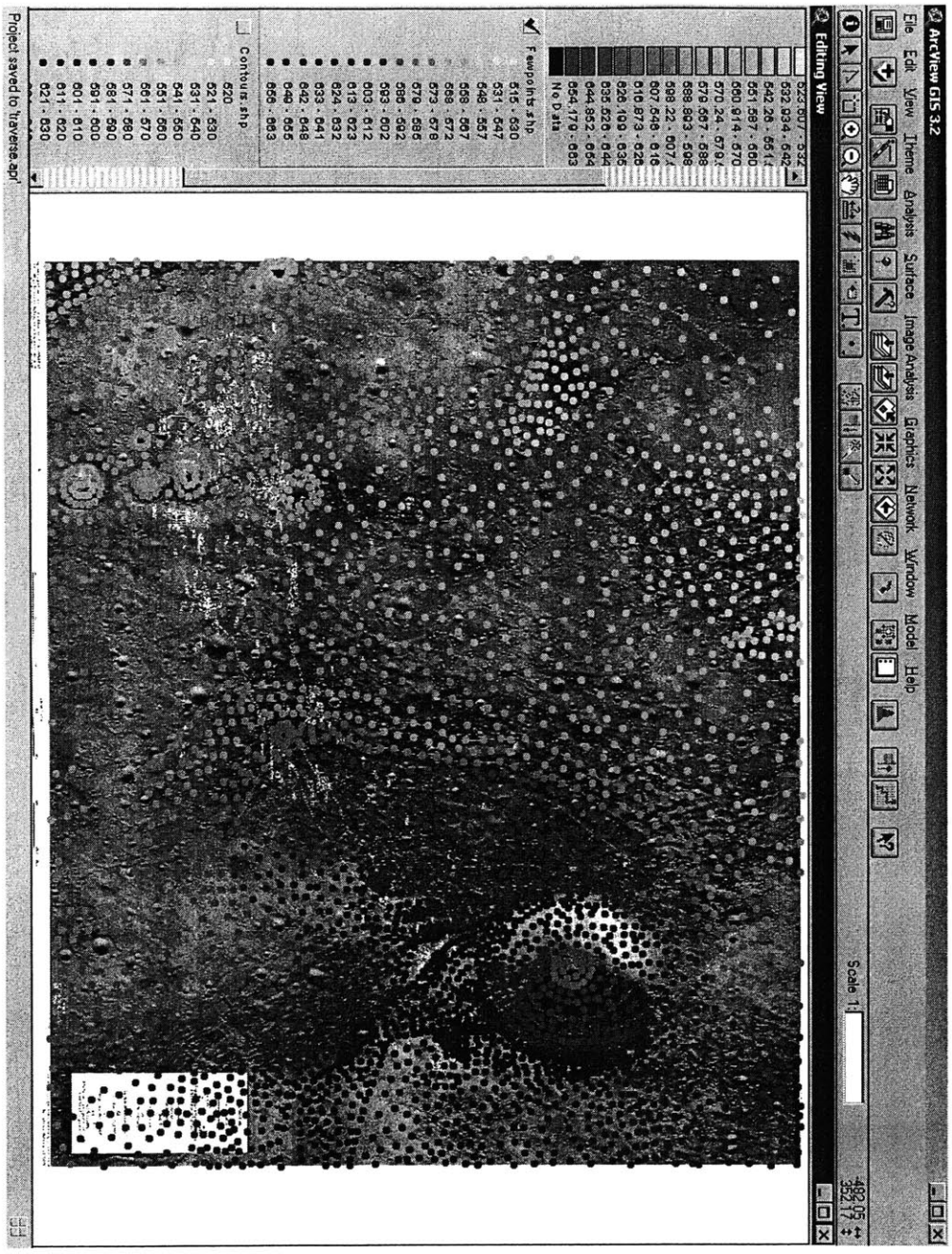


FIGURE C-2. Selecting altitude control points for the Apollo 14 digital elevation model based on contours, contour interpolation, and interpolated craters produced a higher quality digital elevation model. In general, an attempt was made to evenly sample a given region. In many cases, crater rim heights or depths had to be assumed based on general analysis of lunar photographs or astronaut descriptions.

Very limited validation of the Apollo 14 digital elevation model was undertaken. Due to the cratering process, it would be likely that the lunar surface digital elevation model should exhibit a radial power spectral density with a relatively constant scaling exponent. The radial power spectral density computed from the final digital elevation model (based on the more evening sampled altitude control points) is shown in Figure C-3, and does exhibit a relatively constant power law scaling exponent except for very high spatial frequencies.

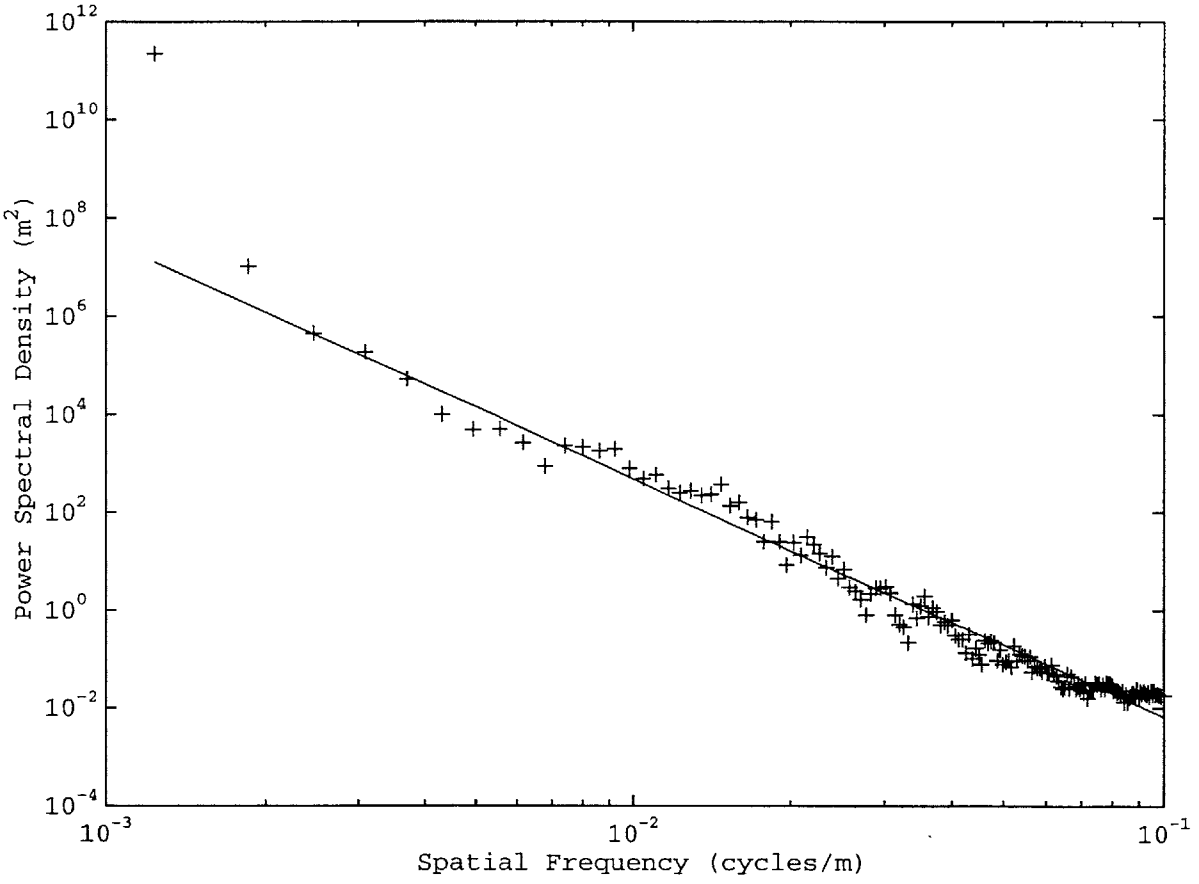


FIGURE C-3. A linear fit of the radial power spectral density of the Apollo 14 digital elevation model gives a power law scaling exponent of 4.86.

C.2 Apollo Biomedical Data

All Apollo metabolic cost data used in this thesis can be found in [Johnston et al., 1975], in Chapter 4 *Metabolism and Heat Dissipation During Apollo EVA Periods* by Waligora and Horrigan. The raw metabolic cost data for the Apollo 14 extravehicular activities is reprinted here as Figure C-4.

Traverse	Duration	Traverse Distance	Net Elevation Change	Net slope Percent	Slope Angle	Traverse Rate	CDR Metabolic Rate		LMP Metabolic Rate	
	min	meters	meters	%	degrees	km/hr	kJ/hr	kcal/hr	kJ/hr	kcal/hr
EVA-1										
ALSEP out	15	172	0	0	0	0.688	1039	248	1159	277
ALSEP return	16	204	0	0	0	0.765	1344	321	1099	262
ALSEP overall	31	376	0	0	0	0.728	1192	285	1130	270
EVA-2										
LM to A	7.4	193	5	2.6	1.49	1.56	593	142	875	209
A to B	7.8	185	-5	-2.7	-1.55	1.42	804	192	887	212
B to B1	7.2	319	15	4.7	2.69	2.65	1009	241	1144	273
B1 to B2	5.1	145	15	10.3	5.88	1.71	1438	343	1218	291
B2 to B3	14.4	399	45	11.3	6.45	1.66	1575	376	1321	315
B3 t C'	6.7	220	15	5.8	3.32	1.97	1911	456	2179	520
C' to C1	1.4	86	3	3.5	2.00	3.69	1024	244	1355	323
C1 to C2	5.6	240	-33	-13.8	-7.86	2.57	998	238	1116	266
C2 to E	6.0	480	-45	-9.4	-5.37	4.8	1314	314	1412	337
E to F	3.9	292	-10	-3.4	-1.95	4.49	1353	323	1545	367
F to G	1.8	172	0	0	0.00	5.7	1180	282	1638	391
G to G1	2.5	110	5	4.5	2.58	2.64	1125	268	1588	379
G1 to LM	3.0	159	-10	-6.3	-3.60	3.18	1277	305	1645	393
Subtotals										
LM to C1	50	1547	93	5	2.86	1.86	1247	298	1262	301
C1 to LM	22.8	1453	-93	-6.4	-3.66	3.82	1205	288	1439	343
Totals										
LM to LM	72.8	3000	0	0	0	2.47	123	294	1318	315

FIGURE C-4. Metabolic expenditures for the two lunar surface astronauts during the Apollo 14 extravehicular activities. From [Johnston et al., 1975], p. 125.

C.3 Apollo Audio Transcripts

Apollo audio transcripts were obtained directly from the Hyper Text Markup Language pages of the Apollo Lunar Surface Journal website [Jones, 2000]. For more on the process of converting these documents to a database of voice communication, see Section 4.2.2.

C.4 References

Johnston, R.S., Dietlein, L.F., and Berry, C.A., *Biomedical Results of Apollo (SP-368)*, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1975.

Jones, Eric M. (ed.), *Apollo Lunar Surface Journal*, July 2000 (*http://www.hq.nasa.gov/office/pao/History/alsj/*).

Swann, G.A., Bailey,N.G, Batson, R.M., Eggleton, R.E., Hait, M.H., Holt, H.E., Larson K.B., Reed, V.S., Schaber, G.G, Sutton, R.L., Trask, N.J., Ulrich, G.E., and Wilshire, G.E., *Geology of the Apollo 14 Landing Site in the Fra Mauro Highlands*, USGS Professional Paper 880, United States Geological Survey.

Alice laughed: "There's no use trying," she said;"one can't believe impossible things."

"I daresay you haven't had much practice," said the Queen. "When I was younger, I always did it for half an hour a day. Why, sometimes I've believed as many as six impossible things before breakfast."

Lewis Carroll, *Alice's Adventures in Wonderland* (1866)

Appendix D - Code Listing

D.1 Introduction

MATLAB functions were written and used to perform most of the analysis in this thesis. Functions were generally written to handle one of five major data types used throughout this work: Surfaces, Points, Graphs, Traverses, and Power Spectral Densities. Miscellaneous functions that do not fall under one of these five structures are assigned a miscellaneous category. Finally, the simulation code for the Rover Assembly of a Martian Sensor Network example near the end of Chapter 5 can be found at the end of this appendix.

D.2 Surface Functions

D.2.1 Surface_Compute_Autocorrelation

```
function [Ca,r] = surface_compute_autocorrelation(X,Y,Z,params,s)

% function [Ca,r] = surface_compute_autocorrelation(X,Y,Z,params,s)
%
% Computes the height autocorrelation function for a x-y vector s
% given a surface defined by X,Y,Z and params.
%
% INPUTS
%  X = x-coordinates of altitude samples (columns of Z)
%  Y = y-coordinates of altitude samples (rows of Z)
%  Z = altitude samples
%  params = 1x6 vector of surface parameters, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  s = 2xn vector with columns of the form [xpoints; ypoints], each column
%      represents a scale length at which the correlation will be computed.
%      xpoints and ypoints must be integer values representing an integer
%      multiple of the cell spacing for the surface.
%
% OUTPUTS
%  Ca = a 1xn vector of height autocorrelation coefficients
%  r = a 2xn vector of scale vectors, derived from s and the surface cell spacing
%
% Author: Chris Carr, June, 2001.

width = params(1);
height = params(2);
nums = size(s,2);
r = zeros(2,nums);
Ca = zeros(1,nums);
xo = 0;
yo = 0;

if (verbose),
    disp('Computing height-height correlation function...');
end
```

```
for i=1:nums,

    % get current scale length
    r(:,i) = s(:,i)*params(5);

    if (verbose),
        disp(sprintf('Scale vector: (x,y)=(%0.2f,%0.2f), %d of %d',r(1,i),r(2,i),i,nums));
    end

    % compute the height autocorrelation for this scale length
    s_x = s(1,i);
    s_y = s(2,i);
    cm = zeros(height-abs(s(2,i)),width-abs(s(1,i)));

    if s_x <0,
        xo = -s_x;
    end
    if s_y <0,
        yo = -s_y;
    end
    for x=1:size(cm,2),
        for y=1:size(cm,1),
            cm(y,x) = (Z(yo+y+s_y,xo+x+s_x)*Z(yo+y,xo+x));
        end
    end

    % compute RMS average of cm to find C
    Ca(i) = sqrt(sum(sum(cm.^2,1),2)/(size(cm,1)*size(cm,2)));
end

return;
```

D.2.2 Surface_Compute_Correlation

```
function [C,r] = surface_compute_correlation(X,Y,Z,params,s)

% function [C,r] = surface_compute_correlation(X,Y,Z,params,s)
%
% Computes the height-height correlation function for a x-y vector s
% given a surface defined by X,Y,Z and params.
%
% INPUTS
%  X = x-coordinates of altitude samples (columns of Z)
%  Y = y-coordinates of altitude samples (rows of Z)
%  Z = altitude samples
%  params = 1x6 vector of surface parameters, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  s = 2xn vector with columns of the form [xpoints; ypoints], each column
%      represents a scale length at which the correlation will be computed.
%      xpoints and ypoints must be integer values representing an integer
%      multiple of the cell spacing for the surface.
%
% OUTPUTS
%  C = a 1xn vector of height-height correlation coefficients
%  r = a 2xn vector of scale vectors, derived from s and the surface cell spacing
%
% Author: Chris Carr, June, 2001.

width = params(1);
```

```
height = params(2);
nums = size(s,2);
r = zeros(2,nums);
C = zeros(1,nums);
xo = 0;
yo = 0;

if (verbose),
    disp('Computing height correlation function...');
end

for i=1:nums,
    % get current scale length
    r(:,i) = s(:,i)*params(5);

    if (verbose),
        disp(sprintf('Scale vector: (x,y)=(%0.2f,%0.2f), %d of %d',r(1,i),r(2,i),i,nums));
    end

    % compute the height-height correlation for this scale length
    s_x = s(1,i);
    s_y = s(2,i);
    cm = zeros(height-abs(s(2,i)),width-abs(s(1,i)));

    if s_x <0,
        xo = -s_x;
    end
    if s_y <0,
        yo = -s_y;
    end
    for x=1:size(cm,2),
        for y=1:size(cm,1),
            cm(y,x) = (Z(yo+y+s_y,xo+x+s_x)-Z(yo+y,xo+x));
        end
    end

    % compute RMS average of cm to find C
    C(i) = sqrt(sum(sum(cm.^2,1),2)/(size(cm,1)*size(cm,2)));
end

return;
```

D.2.3 Surface_Compute_Coverage

```
function C = surface_compute_coverage(Z,params,P,type,offset,merge)
% function C = surface_compute_coverage(Z,params,P,type,offset,merge)
%
% Computes the area of a surface (represented by Z) that is visible from
% a collection of points P.
%
% INPUTS
%  Z      = matrix of altitude samples
%  params = 1x6 vector of surface parameters for Z of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  P      = points at which to compute the coverage
%           P(1,:) = x coordinates
%           P(2,:) = y coordinates
%           P(3,:) = z coordinates
%  type   = 0 compute binary coverage map: entries of C are either 0 or 1
%         = 1 compute distance coverage map: entries of C are distances
%             from (x,y,z) to each point or nodatavalue
```

```
%         = 2 compute slope coverage map: entries of C are dot product between
%           line of sight to a surface location and the slope at that surface
%           location (e.g., a vertical cliff face and a horizontal line of
%           sight would result in an entry of 1, a horizontal surface and a horizontal
%           line of sight would result in an entry of 0). if no line-of-sight
%           then entries are nodatavalue
%         = 3 compute inverse distance coverage map: entries of C are zero if no
%           line of sight exists, and are 1/distance if line of sight exists
%  offset = offset value to use for visibility computations between each point and
%           surface points
%  merge  = 0 merge by addition
%         = 1 merge by multiplication
%
% OUTPUTS
%  C      = coverage matrix (same size as Z) which specifies the visibility
%           of each surface location from point (x,y)

if nargin<4,
    type=0;
end
if nargin<5,
    offset=(max(max(Z))-min(min(Z)))/2000;
end
if nargin<6,
    merge=0;
end

s = size(P,2);
C = ones(size(Z,1),size(Z,2))*NaN; % assign nodatavalue

% for each point in P
for i=1:s,
    x = (P(1,i)-1)*params(5)+params(3);
    y = (size(Z,1)-P(1,i))*params(5)+params(4);
    Ct = surface_compute_coverage_point(Z,params,x,y,P(3,i),type,offset);
    if verbose,
        disp(sprintf('SURFACE_COMPUTE_COVERAGE: %d of %d points (Percentage: %3.1f)',i,s,i*100/
s));
    end
    if i>1,
        %C = surface_merge2(Ct,C,merge);
        C = Ct;
    else
        C = Ct;
    end
end

return;
```

D.2.4 Surface_Compute_Coverage_LOSV

```
function b = surface_compute_coverage_losv(Z,x1,y1,z1,x2,y2,z2)
% function b = surface_compute_coverage_losv(Z,x1,y1,z1,x2,y2,z2)
%
% Computes line of sight visibility between a point (x1,y1,z1) and
% a vector of points specified by (x2,y2,z2).
%
% INPUTS
%  Z = a height field representing a surface that will be used to compute
%      line of sight visibility
%  x1 = x-coordinate of source point
```

```
%  y1 = y-coordinate of source point
%  z1 = z-coordinate of source point
%  x2 = x-coordinate row vector of target points
%  y2 = y-coordinate row vector of target points
%  z2 = z-coordinate row vector of target points
%
% OUTPUTS
%  b = row vector of line-of-sight visibility results (0=no, 1=yes for each entry
%      in b.)
%
% Author: Chris Carr, June 2001.

% initialize position to p1
numsteps = round(sqrt((x2-x1).^2+(y2-y1).^2));
mn = max(numsteps);
msx2 = max(size(x2));
t = zeros(mn,msx2);
for i=1:msx2,
    ni = numsteps(i);
    if ni>0,
        ti = [linspace(0,1,numsteps(i)) ones(1,mn-numsteps(i))]'; % constant step size
    else
        ti = ones(1,mn)'; % constant step size
    end
    t(:,i)=ti;
end
% t is now a matrix with each column representing a single target point

% compute equation of line for all t
p_x = zeros(mn,msx2);
p_y = zeros(mn,msx2);
p_z = zeros(mn,msx2);
for i=1:mn,
    p_x(i,:) = x1 + t(i,:).*(x2-x1);
    p_y(i,:) = y1 + t(i,:).*(y2-y1);
    p_z(i,:) = z1 + t(i,:).*(z2-z1);
end

% check that grid value at current location is below equation of line
zcheck = zeros(mn,msx2);
for i=1:mn,
    idx = (round(p_x(i,:))-1)*size(Z,1)+round(p_y(i,:));
    zcheck(i,:) = Z(idx);
end
g = zeros(size(zcheck,1),size(zcheck,2));
g(find(zcheck>p_z))=1; % condition for visibility is zcheck<=p_z
j = sum(g,1); % sum along columns (each column represents a single visibility computation)
b=not(j); % compute not so that only columns with no failures have line-of-sight condition set to
true
return;
```

D.2.5 Surface_Compute_Coverage_Point

```
function C = surface_compute_coverage_point(Z,params,x,y,z,type,offset)
% function C = surface_compute_coverage_point(Z,params,x,y,z,type,offset)
%
% Computes the area of a surface (represented by Z) that is visible from
% a single point (x,y,z) on the surface.
%
% INPUTS
%  Z      = matrix of altitude samples
```

```
%  params = 1x6 vector of surface parameters for Z of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  x      = x location (not index of Z) for which to compute the coverage
%  y      = y location (not index of Z) for which to compute the coverage
%  z      = z location (not index of Z) for which to compute the coverage
%  type   = 0 compute binary coverage map: entries of C are either 0 or 1
%         = 1 compute distance coverage map: entries of C are distances
%             from (x,y,z) to each point or nodatavalue
%         = 2 compute slope coverage map: entries of C are dot product between
%           line of sight to a surface location and the slope at that surface
%           location (e.g., a vertical cliff face and a horizontal line of
%           sight would result in an entry of 1, a horizontal surface and a horizontal
%           line of sight would result in an entry of 0). if no line-of-sight
%           then entries are nodatavalue
%         = 3 compute inverse distance coverage map: entries of C are zero if no
%           line of sight exists, and are 1/distance if line of sight exists
%  offset = offset used for visibility computations between a point and another point on
%           a surface
%
% OUTPUTS
%  C      = coverage matrix (same size as Z) which specifies the visibility
%           of each surface location from point (x,y)

% define parameters of Z
width = params(1);
height = params(2);
xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5);
nodatavalue = params(6);

if nargin<6,
    type=0;
end
if nargin<7,
    offset=(max(max(Z))-min(min(Z)))/2000;
end

% convert (x,y) to indices of Z
xi = (x-xllcorner)/cellsize+1;
yi = size(Z,1)-(y-yllcorner)/cellsize;

C = ones(size(Z,1),size(Z,2))*nodatavalue;
Ct = ones(size(Z,1),size(Z,2))*nodatavalue;

if or((xi<0),(yi<0)),
    error('(x,y) out of range in SURFACE_COMPUTE_COVERAGE_POINT');
elseif or((xi>size(Z,2)),(yi>size(Z,1))),
    error('(x,y) out of range in SURFACE_COMPUTE_COVERAGE_POINT');
end

if verbose,
    disp('Computing Point Coverage (SURFACE_COMPUTE_COVERAGE_POINT)');
end

% for each point in Z, compute the visibility between (x,y) and Z(i,j)
for i=1:size(Z,1),
    if (verbose)
        disp(sprintf('COVERAGE: Row=%d of %d, Percentage: %3.1f',i,height,(i*width)*100/
(height*width)));
    end
```

```
    for j=1:size(Z,2),
        b = graph_create_NEP_los(Z,xi,yi,z,j,i,Z(i,j)+offset);
        if (type==0),
            Ct(i,j)=b;
        elseif (type==1),
            if (b~=0),
                Ct(i,j)=(sqrt(((j-xi)^2)*(cellsize^2)+((i-yi)^2)*(cellsize^2) + (z-Z(i,j))^2)); %
slant line distance
            else
                Ct(i,j)=nodatavalue;
            end
        elseif (type==2),
            if (b~=0),
                % compute slope
                error('Slope method not yet implemented - SURFACE-COMPUTE-COVERAGE-POINT');
            else
                Ct(i,j)=nodatavalue;
            end
        elseif (type==3),
            if (b~=0),
                Ct(i,j)=1/(sqrt(((j-xi)^2)*(cellsize^2)+((i-yi)^2)*(cellsize^2) + (z-Z(i,j))^2));
% slant line distance
            else
                Ct(i,j)=nodatavalue;
            end
        end
    end
end

C = Ct;
return;
```

D.2.6 Surface_Compute_Coverage_Point_LOSV

```
function C = surface_compute_coverage_pointv(Z,params,x,y,z,type,offset)
% function C = surface_compute_coverage_pointv(Z,params,x,y,z,type,offset)
%
% Computes the area of a surface (represented by Z) that is visible from
% a single point (x,y,z) on the surface.
%
% INPUTS
%  Z      = matrix of altitude samples
%  params = 1x6 vector of surface parameters for Z of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  x      = x location (not index of Z) for which to compute the coverage
%  y      = y location (not index of Z) for which to compute the coverage
%  z      = z location (not index of Z) for which to compute the coverage
%  type   = 0 compute binary coverage map: entries of C are either 0 or 1
%         = 1 compute distance coverage map: entries of C are distances
%             from (x,y,z) to each point or nodatavalue
%         = 2 compute slope coverage map: entries of C are dot product between
%           line of sight to a surface location and the slope at that surface
%           location (e.g., a vertical cliff face and a horizontal line of
%           sight would result in an entry of 1, a horizontal surface and a horizontal
%           line of sight would result in an entry of 0). if no line-of-sight
%           then entries are nodatavalue
%         = 3 compute inverse distance coverage map: entries of C are zero if no
%           line of sight exists, and are 1/distance if line of sight exists
%  offset = offset used for visibility computations between a point and another point on
%           a surface
%
```

```
% OUTPUTS
%  C      = coverage matrix (same size as Z) which specifies the visibility
%           of each surface location from point (x,y)

% define parameters of Z
width = params(1);
height = params(2);
xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5);
nodatavalue = params(6);

if nargin<6,
    type=0;
end
if nargin<7,
    offset=(max(max(Z))-min(min(Z)))/2000;
end

% convert (x,y) to indices of Z
xi = (x-xllcorner)/cellsize+1;
yi = size(Z,1)-(y-yllcorner)/cellsize;

C = ones(size(Z,1),size(Z,2))*nodatavalue;
Ct = ones(size(Z,1),size(Z,2))*nodatavalue;

if or((xi<0),(yi<0)),
    error('(x,y) out of range in SURFACE_COMPUTE_COVERAGE_POINT');
elseif or((xi>size(Z,2)),(yi>size(Z,1))),
    error('(x,y) out of range in SURFACE_COMPUTE_COVERAGE_POINT');
end

if verbose,
    disp('Computing Point Coverage (SURFACE_COMPUTE_COVERAGE_POINT)');
end

% for each point in Z, compute the visibility between (x,y) and Z(i,j)

% package the matrix into the xv,yv, and zv vectors for the visibility computation
for col=1:size(Z,2),
    row = linspace(1,size(Z,1),size(Z,1));
    xv((col-1)*height+1:col*height)=col;
    yv((col-1)*height+1:col*height)=row;
    zv((col-1)*height+1:col*height)=Z(row,col);
end

% make psize visibility computations at a time
psize = 5000;
if verbose,
    disp(sprintf('Computing visibility/coverage computations %d at a time, for this
point...',psize));
end

steps = size(Z,2)*size(Z,1)/psize;
ic = size(Z,2)*size(Z,1);
b = zeros(1,ic);
bcomp = zeros(1,ic);
for i=1:psize:ic,
    ps = min(psize,ic-i+1);
    xt = xv(i:i+ps-1);
    yt = yv(i:i+ps-1);
    zt = zv(i:i+ps-1);
```

```
    bt = surface_compute_coverage_losv(Z,xi,yi,z,xt,yt,zt+offset);
    b(i:i+size(bt,2)-1) = bt;
    if verbose,
        disp(sprintf('Visibility/coverage computations for this point: %10.0f of %10.0f
(%2.1f%%)',i-1+ps,ic,((i-1+ps)/ic)*100));
    end
end

%plot(b-bcomp);

% filter the results vector b
if (type==0),
    Ct=b;
elseif (type==1),
    Ct=(sqrt(((xv-xi).^2).*(cellsize^2)+((yv-yi).^2)*(cellsize^2) + (z-zv).^2)); % slant line dis-
tance
    ndv = find(b==0);
    Ct(ndv)=nodatavalue;
elseif (type==2),
    % compute slope
    error('Slope method not yet implemented - SURFACE-COMPUTE-COVERAGE-POINT');
elseif (type==3),
    Ct=1./(sqrt(((xv-xi).^2)*(cellsize^2)+((yv-yi).^2)*(cellsize^2) + (z-zv).^2)); % slant line
distance
    ndv = find(b==0);
    Ct(ndv)=nodatavalue;
end

% repackage the results Ct into a matrix C
C = ones(size(Z,1),size(Z,2))*nodatavalue;
for col=1:size(Z,2),
    i = linspace(1,size(Z,1),size(Z,1));
    ctr = (col-1)*height + i;
    C(:,col) = Ct(ctr)';
end
return;
```

D.2.7 Surface_Compute_Coverage_Vector

```
function [C,c] = surface_compute_coverage_vector(Z,params,P,type,offset,merge,bMovie,hMovie)
% function [C,c] = surface_compute_coverage_vector(Z,params,P,type,offset,merge,bMovie,hMovie)
%
% Computes the area of a surface (represented by Z) that is visible from
% a collection of points P.
%
% INPUTS
%  Z      = matrix of altitude samples
%  params = 1x6 vector of surface parameters for Z of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  P      = points at which to compute the coverage
%           P(1,:) = x coordinates
%           P(2,:) = y coordinates
%           P(3,:) = z coordinates
%  type   = 0 compute binary coverage map: entries of C are either 0 or 1
%         = 1 compute distance coverage map: entries of C are distances
%             from (x,y,z) to each point or nodatavalue
%         = 2 compute slope coverage map: entries of C are dot product between
%           line of sight to a surface location and the slope at that surface
%           location (e.g., a vertical cliff face and a horizontal line of
%           sight would result in an entry of 1, a horizontal surface and a horizontal
```

```
%           line of sight would result in an entry of 0). if no line-of-sight
%           then entries are nodatavalue
%         = 3 compute inverse distance coverage map: entries of C are zero if no
%           line of sight exists, and are 1/distance if line of sight exists
%  offset = offset value to use for visibility computations between each point and
%           surface points
%  merge  = 0 merge by addition
%         = 1 merge by multiplication
%  bMovie = 0 (default) - don't add frames to movie object
%           1 - add visibility frams to movie object
%  hMovie = handle to a movie object (optional)
%
%
% OUTPUTS
%  C      = coverage matrix (same size as Z) which specifies the visibility
%           of each surface location from point (x,y)
%  c      = count of visible elements for each point in P

if nargin<4,
    type=0;
end
if nargin<5,
    offset=(max(max(Z))-min(min(Z)))/2000;
end
if nargin<6,
    merge=0;
end
if nargin<8,
    bMovie=0;
end

s = size(P,2);
C = ones(size(Z,1),size(Z,2))*NaN; % assign nodatavalue
c = zeros(1,size(P,2));

% for each point in P
for i=1:s,
    x = (P(1,i)-1)*params(5)+params(3);
    y = (size(Z,1)-P(2,i))*params(5)+params(4);
    Ct = surface_compute_coverage_pointv(Z,params,x,y,P(3,i),type,offset);
    c(i) = length(find(isnan(Ct)==0));

    if bMovie~=0,
        % create a frame of the movie and add to hMovie
        h = figure(1);
        X = linspace(params(3),params(3)+(params(1)-1)*params(5),params(1));
        Y = linspace(params(4)+(params(2)-1)*params(5),params(4),params(2));
        surface_plot_minimal(h,X,Y,log(Ct));
        set(gca,'DataAspectRatio',[1 1 1]);
        hMovie(i+1)=getframe(h)
    end

    %Ct = surface_compute_coverage_point(Z,params,P(1,i),P(2,i),P(3,i),type,offset);
    if verbose,
        disp(sprintf('SURFACE_COMPUTE_COVERAGE: %d of %d points (Percentage: %3.1f)',i,s,i*100/
s));
    end
    if i>1,
        C = surface_merge2(Ct,C,merge);
    else
        C = Ct;
    end
```

```
end

if bMovie,
    % save the movie
    save movie.mat hMovie;
end
```

D.2.8 Surface_Compute_Histogram

```
function [n, c] = surface_compute_histogram(Z,params,bins,normalized)

% function [n, c] = surface_compute_histogram(Z,params,bins,normalized)
%
% Computes an altitude histogram for the surface
% represented by Z using the bins specified by bins.
%
% INPUTS
%  Z          = a height field matrix
%  params     = surface parameters for the surface represented by Z, of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  bins       = vector that specifies bins for the altitude histogram. See hist
%               for the format of bins.
%  normalized = optional argument; non-zero specifies that output counts should be
%               normalized to proportions
%
% OUTPUTS
%  n      = count vector for output histogram
%  c      = bin centers for output histogram
%
% Author: Chris Carr, May 2001.

if max(size(bins))>1,
    % assume user wants to specify bin edges
    n = zeros(1,max(size(bins))-1);
    binedges = bins;
    numb = max(size(bins))-1;
    offset = (binedges(numb+1)-binedges(1))/10^20; % ensures that values at max are included in
count
    for i=1:size(Z,1),
        [nt,ct]=histc(Z(i,:),[binedges(1:numb) binedges(numb+1)+offset]);
        n = n+nt(1:max(size(nt))-1);
    end
    ct = (binedges(1:numb) + binedges(2:numb+1))/2;
else
    % assume bins is number of bins to use
    n = zeros(1,bins);

    % create appropriately sized bins
    binedges = linspace(min(min(Z)),max(max(Z)),bins+1);
    offset = (binedges(bins+1)-binedges(1))/10^20; % ensures that values at max are included in
count
    for i=1:size(Z,1),
        [nt,ct]=histc(Z(i,:),[binedges(1:bins) binedges(bins+1)+offset]);
        n = n+nt(1:max(size(nt))-1);
    end
    ct = (binedges(1:bins) + binedges(2:bins+1))/2;
end

c = ct;
if nargin==4,
```

```
    if (normalized~=0),
        n = n/sum(n,2);
    end
end

return;
```

D.2.9 Surface_Compute_Histogram_Cumulative

```
function [n, c] = surface_compute_histogram_cumulative(Z,params,bins,normalized)

% function [n, c] = surface_compute_histogram(Z,params,bins,normalized)
%
% Computes an cumulative altitude histogram for the surface
% represented by Z using the bins specified by bins. Each bin will
% contain a count that represents the number of samples of Z less than
% or equal to the right bin boundary.
%
% INPUTS
%  Z          = a height field matrix
%  params     = surface parameters for the surface represented by Z, of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  bins       = vector that specifies bins for the altitude histogram. See hist
%               for the format of bins.
%  normalized = optional argument; non-zero specifies that output counts should be
%               normalized to proportions
%
% OUTPUTS
%  n      = count vector for output histogram
%  c      = bin centers for output histogram
%
% Author: Chris Carr, May 2001.

if nargin<4,
    normalized = 0;
end

[n,c] = surface_compute_histogram(Z,params,bins, normalized);

for i=1:size(n,2),
    if n~=1,
        n(i) = n(i) + n(i-1);
    end
end

return;
```

D.2.10 Surface_Compute_PSD

```
function [psd,f,w]=surface_compute_psd(Z,params,option)

% function [psd,f,w]=surface_compute_psd(Z,params,option)
%
% Computes the Power Spectral Density of a height field Z
% using the surface parameters params.
%
% INPUTS
%  Z = square height field
```

```
%  params = 1x6 vector of surface parameters, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  option = optional parameter that indicates how PSD should be normalized
%           0 = normalized by 1/number of points N in Z and by w^2
%           1 = normalized by 1/number of points N in Z
%           2 = normalized by 1/number of points N in Z and by 1/w^2 where
%               w is the cellsize for the surface Z.
%           3 = unnormalized
%
% OUTPUTS
%  psd = power spectral density of height field Z
%  f   = vector of spatial frequency corresponding to rows or columns of psd
%  w   = vector of spatial wavelengths corresponding to rows or columns of psd
%
% Note: psd has low frequencies near the center of the matrix. The height field
% Z must also have even dimensions.
%
% Author: Chris Carr, May 2001.

if nargin<3,
    option=0;
end

d = params(5);
dft_z = fft2(Z);
if (option==0),
    N = size(Z,1)*size(Z,2)/(params(5)^2);
elseif (option==1),
    N = size(Z,1)*size(Z,2);
elseif (option==2),
    N = size(Z,1)*size(Z,2)*params(5)^2;
else
    N = 1;
end

psd = abs(dft_z).^2 * (1/N);

ftmp = linspace(1/(size(psd,2)*d/2),1/(2*d),size(psd,2)/2);
f = [ftmp fliplr(-ftmp)];
w = 1./f;
```

D.2.11 Surface_Compute_Reachability

```
function Ro = surface_compute_reachability(Zo,params,P,lim,type,merge)
%
% function R = surface_compute_reachability(Z,params,P,lim,type,merge)
%
% Computes the set of points reachable from a set of initial points
% for a surface represented by Z and params.
%
% In order for a point to be reachable, it must be connected to
% other reachable points (the initial points are by definition reachable)
% and have values within the range specified by the value limits vector
% lim.
%
% INPUTS
%  Z      = matrix of altitude samples
%  params = 1x6 vector of surface parameters for Z of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  P      = initially reachable points
```

```
%           P(1,:) = x coordinates
%           P(2,:) = y coordinates
%           P(3,:) = z coordinates
%  lim    = [lim_min lim_max], defines range criterion for reachability
%  type   = 0 (default), connected defined by up/down and left/right
%         = 1, connected defined by up/down, left/right, and diagonals
%  merge  = 0 merge by addition (default)
%         = 1 merge by multiplication
%
% OUTPUTS
%  R      = coverage matrix (same size as Z) which specifies the visibility
%           of each surface location from point (x,y)

set(0,'RecursionLimit',10000);

global Z;
global M;
global Rp;
global limits;
global mytype;

Z=Zo;
limits = lim;
mytype = type;

R = zeros(size(Z,1),size(Z,2))*NaN;
sx = size(Z,2);
sy = size(Z,1);

% for each point in P, compute the reachability
for i=1:size(P,2),
    % compute reachability for point (xi,yi)
    complete = 0;
    xi = P(1,i);
    yi = P(2,i);
    Rp = zeros(size(Z,1),size(Z,2))*NaN;
    M = zeros(size(Z,1),size(Z,2));

    %debug=0;
    no = [];
    while(not(complete)),
        %debug=debug+1
        cz = length(find(M==0));
        for k=1:max(size(xi)),
            flood_fill(yi(k),xi(k));
        end
        %ni = find(M==-1);
        %if isempty(setdiff(no,ni)),
        %    complete=1;
        %end
        yi = mod(find(M==-1),sy);
        %yi = mod(find(M==-1),sx);
        yzi = find(yi==0);
        if not(isempty(yzi)),
            yi(yzi)=sx;
        end
        xi = ceil(find(M==-1)/sy);
        %xi = ceil(find(M==-1)/sx);
        yg = find(yi>size(Z,1));
        yi(yg)=size(Z,1);
        xg = find(xi>size(Z,2));
        xi(xg)=size(Z,2);
```

```
        if length(xi)==0,
            complete=1;
        end
        %no = ni;
        cz2 = length(find(M==0));
        if cz==cz2,
            complete=1;
        end

    end

    % merge the reachable areas
    R = surface_merge2(R,Rp,merge);
end



Ro=R;
return;
```

D.2.12 Surface_Compute_Reconfig_Cost

```
function [C,c] = surface_compute_reconfig_cost(Z,params,Xs,Xf,m,t,g,op1,op2)
% function [C,c] = surface_compute_reconfig_cost(Z,params,Xs,Xf,m,t,g,op1,op2)
%
% Computes the cost of a system reconfiguration of a series of nodes in
% a distributed system.
%
% INPUTS
%  Z - a matrix of altitude samples
%  params - a 1x6 vector of height field parameters; of the form
%            [width height xllcorner yllcorner cellsize nodatavalue]
%  Xs - initial node positions, of the form
%       Xs(1,:) = x-coordinates of points x in Z
%       Xs(2,:) = y-coordinates of points y in Z
%       Xs(3,:) = z-coordinates of points x and y
%  Xf - final node positions (same form as Xs)
%  m  - vector of masses or other parameter for the nodes
%  t  - length of reconfiguration time period in seconds
%  g  - gravitational acceleration (scalar)
% op1 - traverse optimization approach (op1=0, straight line traverse)
%                                      (op1=1, min cost path traverse)
% op2 - traverse cost function option (op2=0, use metabolic cost load carrying model)

C = zeros(1,size(Ps,2));
c = 0;

% define traverse sampling interval - spatial sampling
s = params(5);

% for each node
for i=1:size(Ps,2),
    % build a traverse from starting position to ending position
    if op1==0,
        % build a straight line traverse
        % use starting and ending points...
        Ti = [[1 Xs(1,i) Xs(2,i) Xs(3,i) 0 1 0 0]' [2 Xf(1,i) Xf(2,i) Xf(3,i) 0 2 t 0]'];
        % ... and just interpolate between them.
        Ti = traverse_interpolate(Ti,s,0); % spatial sampling
    elseif op1==1,
```

```
        % build a min cost traverse between points
        error('Min Cost Traverse between points not yet implemented in SURFACE_COMPUTE_RECONFIG');
    else
        error('Invalid value for option op1 in SURFACE_COMPUTE_RECONFIG_COST');
    end

    % compute cost of that traverse
    if op2==0,
        % use metabolic cost model
        m = traverse_compute_metabolic_cost(Ti,m(i),g);
        pause;
        rc = 0;
    end

    % store cost in C
    C(1,i)=rc;

% compute total cost of reconfiguration
c = sum(sum(C,1),2);
```

D.2.13 Surface_Compute_Slope

```
function [Xn,Yn,Zn,np]=surface_compute_slope(X,Y,Z,params)

% function [Xn,Yn,Zn,np]=surface_compute_slope(X,Y,Z,params)
%
% Computes the slope of a surface in degrees and returns a new
% surface that represents the slope field of the input surface.
% Slopes are computed using a weighted centered-difference
% algorithm and a 3x3 kernel.
%
% INPUTS
%  X = x-coordinates of altitude samples (columns of Z)
%  Y = y-coordinates of altitude samples (rows of Z)
%  Z = altitude samples
%  params = 1x6 vector of surface parameters, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% OUTPUTS
%  Xn = x-coordinates of slope samples (columns of Zn)
%  Yn = y-coordinates of slope samples (rows of Zn)
%  Zn = slope samples
%  np = 1x6 vector of surface parameters for new surface, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% Note: Slope computations for the edges of the surface are based
% on the local slope trend.
%
% Author: Chris Carr, May 2001.

width = params(1);
height = params(2);
w = params(5); % cell spacing

for i=1:height,
    for j=1:width,
        % compute slope Zn(i,j) for grid point Z(i,j)
        if (j==1),
            % left side
            if (i==1),
                % upper left corner
```

```
            dx = (2*Z(i,j+1)+Z(i+1,j+1)-3*Z(i,j))/(3*w);
            dy = (2*Z(i+1,j)+Z(i+1,j+1)-3*Z(i,j))/(3*w);
            Zn(i,j) = 180/pi*atan(dx^2 + dy^2);
        elseif (i==height),
            % lower left corner
            dx = (2*Z(i,j+1)+Z(i-1,j+1)-3*Z(i,j))/(3*w);
            dy = (2*Z(i-1,j)+Z(i-1,j+1)-3*Z(i,j))/(3*w);
            Zn(i,j) = 180/pi*atan(dx^2 + dy^2);
        else
            % middle left side
            dx = (2*Z(i,j+1)+Z(i-1,j+1)+Z(i+1,j+1)-4*Z(i,j))/(4*w);
            dy = (2*Z(i+1,j)+Z(i+1,j+1)-2*Z(i-1,j)-Z(i-1,j+1))/(6*w);
        end
    elseif (j==width),
        % right side
        if (i==1),
            % upper right corner
            dx = (2*Z(i,j-1)+Z(i+1,j-1)-3*Z(i,j))/(3*w);
            dy = (2*Z(i+1,j)+Z(i+1,j-1)-3*Z(i,j))/(3*w);
            Zn(i,j) = 180/pi*atan(dx^2 + dy^2);
        elseif (i==height),
            % lower right corner
            dx = (2*Z(i,j-1)+Z(i-1,j-1)-3*Z(i,j))/(3*w);
            dy = (2*Z(i-1,j)+Z(i-1,j-1)-3*Z(i,j))/(3*w);
            Zn(i,j) = 180/pi*atan(dx^2 + dy^2);
        else
            % right side middle
            dx = (2*Z(i,j-1)+Z(i-1,j-1)+Z(i+1,j-1)-4*Z(i,j))/(4*w);
            dy = (2*Z(i+1,j)+Z(i+1,j-1)-2*Z(i-1,j)-Z(i-1,j-1))/(6*w);
        end
    else
        % in middle
        if (i==1),
            % top
            dx = (2*Z(i,j-1)+Z(i+1,j-1)-2*Z(i,j+1)-Z(i+1,j+1))/(6*w);
            dy = (2*Z(i+1,j)+Z(i+1,j-1)+Z(i+1,j+1)-4*Z(i,j))/(4*w);
        elseif (i==height),
            % bottom
            dx = (2*Z(i,j-1)+Z(i-1,j-1)-2*Z(i,j+1)-Z(i-1,j+1))/(6*w);
            dy = (2*Z(i-1,j)+Z(i-1,j-1)+Z(i-1,j+1)-4*Z(i,j))/(4*w);
        else
            % middle; use standard slope computation
           dx = ((Z(i-1,j-1)+2*Z(i,j-1)+Z(i+1,j-1))-(Z(i-1,j+1)+2*Z(i,j+1)+Z(i+1,j+1)))/(8*w);
           dy = ((Z(i-1,j-1)+2*Z(i-1,j)+Z(i-1,j+1))-(Z(i+1,j-1)+2*Z(i+1,j)+Z(i+1,j+1)))/(8*w);
            Zn(i,j) = 180/pi*atan(dx^2 + dy^2);
        end
    end
end

Xn = X;
Yn = Y;
np = params;
return;
```

D.2.14 Surface_Compute_Subsample

```
function [Xn,Yn,Zn,np] = surface_compute_subsample(X,Y,Z,params,samp)

% function [Xn,Yn,Zn,np] = surface_compute_subsample(X,Y,Z,params,samp)
```

```
%
% Given a surface defined by X,Y,Z, subsamples that surface (decimates the
% surface by a factor of samp).
%
% INPUTS
%  X      = x-coordinates of height field samples for columns of Z
%  Y      = y-coordinates of height field samples for rows of z
%  Z      = height field samples
%  params = 1x6 vector of surface parameters, of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  samp   = decimation factor for subsampling
%
% OUTPUTS
%  Xn     = x-coordinates of height field samples for columns of Zn
%  Yn     = y-coordinates of height field samples for rows of Zn
%  Zn     = height field samples within the region specified by region
%  np     = 1x6 vector of surface parameters for Zn, of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% Author: Chris Carr, May, 2001.

xs = floor(size(Z,2)/samp);
ys = floor(size(Z,1)/samp);

for i=1:ys,
    for j=1:xs,
        Zn(i,j)=Z(i*samp,j*samp);
    end
end
xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5)*samp;
height = ys;
width = xs;

Xn = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
Yn = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);
np = [width height xllcorner yllcorner cellsize params(6)];

return
```

D.2.15 Surface_Compute_Supersample

```
function [Xn,Yn,Zn,np] = surface_compute_supersample(X,Y,Z,params,samp)

% function [Xn,Yn,Zn,np] = surface_compute_supersample(X,Y,Z,params,samp)
%
% Given a surface defined by X,Y,Z, supersamples that surface (using
% spline interpolation).
%
% INPUTS
%  X      = x-coordinates of height field samples for columns of Z
%  Y      = y-coordinates of height field samples for rows of z
%  Z      = height field samples
%  params = 1x6 vector of surface parameters, of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  samp   = decimation factor for subsampling
%
% OUTPUTS
%  Xn     = x-coordinates of height field samples for columns of Zn
```

```
%  Yn      = y-coordinates of height field samples for rows of Zn
%  Zn      = height field samples within the region specified by region
%  np      = 1x6 vector of surface parameters for Zn, of the form
%            [width height xllcorner yllcorner cellsize nodatavalue]
%
% Author: Chris Carr, May, 2001.

Zn = resample(Z,samp,1);
Zn = resample(Zn',samp,1)';

xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5)/samp;
height = params(2)*samp;
width = params(1)*samp;

Xn = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
Yn = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);
np = [width height xllcorner yllcorner cellsize params(6)];

return
```

D.2.16 Surface_Compute_TravelDir

```
function [Dx,Dy,Dm,DD,deg]=surface_compute_traveldir(X,Y,Z,params,dvectors,segmentsize)

% function [Dx,Dy,Dm,DD,deg]=surface_compute_traveldir(X,Y,Z,params,dvectors,segmentsize)
%
% Provides a heuristic for the preferred direction of travel in a given surface
% region (blocks of segmentsize x segmentsize in map units) using the
% height-height correlation function.
%
% INPUTS
%  X - x coordinates of surface columns
%  Y - y coordinates of surface rows
%  Z - a matrix of altitude samples
%  params - a 1x6 vector of height field parameters; of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  dvectors - list of directions vectors along which to compute the height-height
%           correlation function for a given segment, of the form [[a b]' [c d]' ...]
%           where a and c are x coordinates of the direction vector and b and d are
%           y coordinates of the direction vector and a,b,c,d are all integers.
%  segmentsize - scalar integral value specifying the dimension for a given surface
%          segment, specified in map units; must be larger than all dirdeg vectors
%
% OUTPUTS
%  Dx = x coordinates of segment centers
%  Dy = y coordinates of segment centers
%  Dm = minimum value of height-height correlation function
%  DD = best direction in degrees
%  deg = degrees analyzed
%
% Author:Chris Carr
% Date: August, 2001

sg_size = segmentsize/params(5);

% compute number and position of segment blocks
D_x = floor((params(1,1)-1)/sg_size); % number of x segment blocks
D_y = floor((params(1,2)-1)/sg_size); % number of y segment blocks
Dx = params(3)+(0:D_x-1)*sg_size*params(1,5)+sg_size*params(1,5)/2;
```

```
Dy = params(4)+(0:D_y-1)*sg_size*params(1,5)+sg_size*params(1,5)/2;

% compute directions in degrees
for i=1:size(dvectors,2),
    deg(i) = atan2(dvectors(2,i),dvectors(1,i));
end
deg(size(dvectors,2)+1)=NaN;

DD = zeros(D_y,D_x);
Dm = zeros(D_y,D_x);
% for each segment
for idx_x = 1:D_x,
    for idx_y = 1:D_y,
        % extract box of size segmentsize x segmentsize centered on this position
        sgs = sg_size*params(1,5)/2;
        region = [Dx(idx_x)-sgs Dy(idx_y)-sgs Dx(idx_x)+sgs Dy(idx_y)+sgs];
        [Xseg,Yseg,Zseg,pseg]=surface_extract_region(X,Y,Z,params,region);
        % subtract mean altitude from Z for height-height correlation function analysis
        %mz = sum(sum(Z,1),2)/(size(Z,1)*size(Z,2));
        mz = min(min(Z))-1;
        Zseg = Zseg-mz;
        % for each applicable direction compute height height correlation function over segment
        v = verbose;
        verbose(0);
        [Ca,r]=surface_compute_autocorrelation(Xseg,Yseg,Zseg,pseg,dvectors/params(5));
        verbose(v);
        % find the minimum value and its index
        [hh_min idx_hh_min] = min(Ca);
        DD(idx_y,idx_x)=deg(idx_hh_min);
        Dm(idx_y,idx_x)=hh_min;
        % check if min and max are really close

        if (abs(min(Ca)-max(Ca))/abs(min(Ca))<0.001)
            DD(idx_y,idx_x)=NaN;
        end
    end
    if verbose,
        disp(sprintf('Computing Traveldirection for Segment %d of %d (%0.2f
%%)',idx_x*D_y,D_x*D_y,((idx_x)*D_y)/(D_x*D_y)*100));
    end
end

return;
```

D.2.17 Surface_Convert_Crd_To_Pts

```
function P = surface_convert_crd_to_pts(params,X)
% function P = surface_convert_crd_to_pts(params,X)
%
% Converts points indices to point coordinates.
%
% INPUTS
%  X = point coordinates in map units, of the form:
%      X(1,:) = x-coordinates of points x in Z
%      X(2,:) = y-coordinates of points y in Z
%      X(3,:) = z-coordinates of points z in Z
%
% OUTPUTS
%  P = points to convert, of the form:
%      P(1,:) = x-indicies of points x in Z
%      P(2,:) = y-indices of points y in Z
```

```
%       P(3,:) = z-coordinates of points z in Z
%
% Author: Chris Carr
% Date: August, 2001

xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5);
height = params(2);

P(1,:) = round((X(1,:)-xllcorner)/cellsize+1);
P(2,:) = round(( -X(2,:)+yllcorner+(height-1)*cellsize)/cellsize+1);
P(3,:) = X(3,:);
```

D.2.18 Surface_Convert_Pts_To_Crd

```
function X = surface_convert_pts_to_crd(params,P)
% function X = surface_convert_pts_to_crd(params,P)
%
% Converts points indices to point coordinates.
%
% INPUTS
%  P = points to convert, of the form:
%       P(1,:) = x-indicies of points x in Z
%       P(2,:) = y-indices of points y in Z
%       P(3,:) = z-coordinates of points z in Z
%
% OUTPUTS
%  X = point coordinates in map units, of the form:
%       X(1,:) = x-coordinates of points x in Z
%       X(2,:) = y-coordinates of points y in Z
%       X(3,:) = z-coordinates of points z in Z
%
% Author: Chris Carr
% Date: August, 2001

xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5);
height = params(2);

X = zeros(3,size(P,2));

X(1,:)=(P(1,:)-1)*cellsize + xllcorner;
X(2,:)=-((P(2,:)-1)*cellsize - yllcorner - (height-1)*cellsize);
X(3,:)=P(3,:);
```

D.2.19 Surface_Create

```
function [x,y,z] = surface_create(surfparams,type,beta,scale,altminmax)

% function [x,y,z] = surface_create(surfparams,type,beta,scale,altminmax)
%
% Creates an artificially generated surface using surface parameters
% surfparams, of type type. Beta and scale are optional parameters
% for surfaces created using an exponential power density function
% scaling law.
%
% INPUTS
```

```
%  surfparams = 1x6 vector of surface parameters of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  type       = 1 (for power spectral density exponential scaling law
%                  based height field)
%               2 (for zero height field),
%               3 (for uniform random height field), on interval [0,1]
%               4 (for alternating height field)
%  beta       = power spectral density scaling exponent (required input
%               only if type=1
%  scale      = prefactor for power spectral density scaling law
%  altminmax  = 1x2 vector containing desired min and max altitude of
%               newly generated height field (optional parameter)
%
% OUTPUTS
%  x = x-coordinates of altitude samples of columns of z
%  y = y-coordinates of altitude samples of rows of z
%  z = altitude samples of generated height field
%
% Author: Chris Carr, May 2001.

width = surfparams(1);
height = surfparams(2);
xllcorner = surfparams(3);
yllcorner = surfparams(4);
cellsize = surfparams(5);

if type==2,
    if (verbose),
        disp('Creating Zero Height Field (SURFACE_CREATE)');
    end

    % zero height field
    z = zeros(height,width);
    x = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
    y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);
elseif type==3,
    if (verbose),
        disp('Creating Random Height Field (SURFACE_CREATE)');
    end

    % random height field
    z = rand(height,width);
    x = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
    y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);
elseif type==4,
    if (verbose),
        disp('Creating Alternating Height Field (SURFACE_CREATE)');
    end

    % 1 and 0 alternating
    for i=1:height,
        for j=1:width,
            z(i,j)=mod(i,2) & mod(j,2);
        end
    end
    x = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
    y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);
elseif and(nargin>=4,type==1),
    if (verbose),
        disp('Creating Exponential PSD Height Field (SURFACE_CREATE)');
    end
```

```
    % note: synth_spect uses a scaling law to BUILD THE SPECTRUM, NOT
    % THE POWER SPECTRA. We want a POWER LAW EXPONENT, so beta/2 is
    % passed to synth_spect so that the POWER of the DFT built by
    % synth_spect corresponds to the POWER LAW EXPONENT beta.
    [sspec,z]=synth_spect([height,width],sprintf('%d^2*kr.^(-%f)',scale,beta/2));
    x = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
    y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);

    % rescale to similar altitudes
    if nargin==5,
        minz = min(min(z));
        maxz = max(max(z));
        z = z-minz; % move to zero
        z = z.*(altminmax(2)-altminmax(1))/(maxz-minz); % scale
        z = z+altminmax(1); % move to minimum altitude
        sspec = fft2(z);
    end

else
    % default
    if (verbose),
        disp('Invalid Arguments (SURFACE_CREATE)');
    end
    x = [];
    y = [];
    z = [];
end
```

D.2.20 Surface_Extract_Altitudes

```
function P = surface_extract_altitudes(Z,params,x,y)

% function P = surface_extract_altitudes(Z,params,x,y)
%
% Given a surface defined by Z, extracts the altitude samples defined
% by vectors x and y. Returns heights specified by points (xi,yi) in z.
%
% INPUTS
%  Z      = height field
%  params = 1x6 vector of surface parameters of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  x      = x-coordinates of points to extract
%  y      = y-coordinates of points to extract
%
% OUTPUTS
%  P     = P(1,:) = x-indicies of points x in Z
%          P(2,:) = y-indices of points y in Z
%          P(3,:) = z-coordinates of points x and y
%
% Author: Chris Carr, May, 2001.

% convert x and y to indices for Z

xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5);

% make out of bounds entries have NaN
xn = x;
```

```
yn = y;
zn = zeros(1,max(size(x)));
P = zeros(3,max(size(x)));

out_right = find(x>(xllcorner + (size(Z,2)-1)*cellsize));
if max(size(out_right)>0),
    xn(out_right) = NaN;
    yn(out_right) = NaN;
    zn(out_right) = NaN;
end
out_left = find(x<xllcorner);
if max(size(out_left)>0),
    xn(out_left) = NaN;
    yn(out_left) = NaN;
    zn(out_left) = NaN;
end
out_top = find(y>(yllcorner + (size(Z,1)-1)*cellsize));
if max(size(out_top)>0),
    xn(out_top) = NaN;
    yn(out_top) = NaN;
    zn(out_top) = NaN;
end
out_bot = find(y<yllcorner);
if max(size(out_bot)>0),
    xn(out_bot) = NaN;
    yn(out_bot) = NaN;
    zn(out_bot) = NaN;
end

xi = round((xn-xllcorner)/cellsize+1);
yi = round((-yn+yllcorner+(size(Z,1)-1)*cellsize)/cellsize+1);

for i=1:max(size(x));
    if not(isnan(xi(i))),
        zn(i)=Z(yi(i),xi(i));
    end
end

P(1,:) = xi;
P(2,:) = yi;
P(3,:) = zn;

return
```

D.2.21 Surface_Extract_Even

```
function [Xn,Yn,Zn,np]=surface_extract_even(X,Y,Z,params)

% function [Xn,Yn,Zn,np]=surface_extract_even(X,Y,Z,params)
%
% Extracts the largest region of the current surface that has
% an even number of samples in the x and y directions.
%
% INPUTS
%  X = x-coordinates of altitude samples (columns of Z)
%  Y = y-coordinates of altitude samples (rows of Z)
%  Z = altitude samples
%  params = 1x6 vector of surface parameters, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
```

```
% OUTPUTS
%  Xn = x-coordinates of new altitude samples (columns of Zn)
%  Yn = y-coordinates of new altitude samples (rows of Zn)
%  Zn = new altitude samples
%  np = 1x6 vector of surface parameters for new surface, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% Note: If a row or column is removed, it is removed from the southern
% or eastern margin of the surface by default.
%
% Author: Chris Carr, May 2001.

a = mod(size(Z,1),2);
b = mod(size(Z,2),2);

Xn = X;
Yn = Y;
Zn = Z;
np = params;

if a~=0,
    % remove last row
    Yn = Yn(1:size(Yn,2)-1);
    Zn = Zn(1:size(Zn,1)-1,:);
    np(2)=np(2)-1;
end

if b~=0,
    % remove last column
    Xn = Xn(1:size(Xn,2)-1);
    Zn = Zn(:,1:size(Zn,2)-1);
    np(1)=np(1)-1;
end

return;
```

D.2.22 Surface_Extract_Normalized

```
function [X,Y,Z,np] = surface_extract_normalized(Z,params,normp)

% function [X,Y,Z,np] = surface_extract_normalized(Z,params,normp)
%
% Normalizes a surface.
%
% INPUTS
%  Z      = matrix of altitude samples
%  params = parameters of the surface, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  normp = [minxynorm, minznorm, maxxynorm, maxznorm] - specifies to what dimensions
%          the surface should be normalized in each direction
%
% OUTPUTS
%  X = x-coordinates of altitude samples (columns of Z)
%  Y = y-coordinates of altitude samples (rows of Z)
%  Z = altitude samples
%
% Note: Only works for square height fields.
%
% Author: Chris Carr, May 2001.

if size(Z,1)~=size(Z,2),
```

```
    error('Error: SURFACE_EXTRACT_NORMALIZED only works for square height fields');
end

if nargin<2,
    normp = [1 1];
end

X = linspace(normp(1),normp(3),size(Z,2));
Y = linspace(normp(1),normp(3),size(Z,1));
Z = (Z-min(min(Z)))*(normp(4)-normp(2))/(max(max(Z))-min(min(Z)))+normp(2);
np = [params(1) params(2) normp(1) normp(2) params(1)/size(Z,2) params(6)];
```

D.2.23 Surface_Extract_Points

```
function P = surface_extract_points(Z,params,x,y)

% function P = surface_extract_points(Z,params,x,y)
%
% Given a surface defined by Z, extracts the altitude samples defined
% by vectors x and y. Returns points in matrix P.
%
% INPUTS
%  Z      = height field
%  params = 1x6 vector of surface parameters of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  X      = x-coordinates of points to extract
%  Y      = x-coordinates of points to extract
%
% OUTPUTS
%  P     = 3 x n matrix where n is the number of points being extracted.
%          P(1,:) = x coordinates of points
%          P(2,:) = y coordinates of points
%          P(3,:) = z coordinates of points
%
% Author: Chris Carr, May, 2001.

P(1,:)=x;
P(2,:)=y;
for i=1:max(size(x));
    P(3,i)=Z(y(i),x(i));
end

return
```

D.2.24 Surface_Extract_Profile

```
function P = surface_extract_profile(Z,params,x1,y1,x2,y2)
% function P = surface_extract_profile(Z,params,x1,y1,x2,y2)
%
% Given a surface defined by Z, extracts the altitude samples along the
% line between x1,y1 and x2,y2. Returns heights specified by points (xi,yi)
% in Z.
%
% INPUTS
%  Z      = height field
%  params = 1x6 vector of surface parameters of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  x1      = x-coordinates of end point 1
%  y1      = y-coordinates of end point 1
%  x2      = x-coordinates of end point 2
%  y2      = y-coordinates of end point 2
```

```
%
% OUTPUTS
%  P     = P(1,:) = x-indicies of points x in Z
%          P(2,:) = y-indices of points y in Z
%          P(3,:) = z-coordinates of points x and y
%
% Author: Chris Carr, May, 2001.

spacing = params(5);
steps = sqrt(((x2-x1)/spacing)^2 + ((y2-y1)/spacing)^2)+1;

% create a line from x1,y1 to x2,y2 using the appropriate number of samples
xn = linspace(x1,x2,steps);
yn = linspace(y1,y2,steps);
% get the altitude samples for this line using surface_extract_altitudes
P = surface_extract_altitudes(Z,params,xn,yn);
```

D.2.25 Surface_Extract_Region

```
function [Xn,Yn,Zn,np] = surface_extract_region(X,Y,Z,params,region)

% function [Xn,Yn,Zn,np] = surface_extract_region(X,Y,Z,params,region)
%
% Given a surface defined by X,Y,Z, extracts a region of the surface
% defined by region and returns the region as the surface defined by
% Xn, Yn, and Zn. Empty Xn, Yn, Zn are returned if region is not fully
% within the surface specified by X,Y,Z.
%
% INPUTS
%  X      = x-coordinates of height field samples for columns of Z
%  Y      = y-coordinates of height field samples for rows of z
%  Z      = height field samples
%  params = 1x6 vector of surface parameters, of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  region = bounding box for new surface, of the form [xmin ymin xmax ymax]
%
% OUTPUTS
%  Xn     = x-coordinates of height field samples for columns of Zn
%  Yn     = y-coordinates of height field samples for rows of Zn
%  Zn     = height field samples within the region specified by region
%  np     = 1x6 vector of surface parameters for Zn, of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% Author: Chris Carr, May, 2001.

a = min(region(1)>=min(X));
b = min(region(2)>=min(Y));
c = max(region(3)<=max(X));
d = max(region(4)<=max(Y));
np = params;

if and(a,and(b,and(c,d))),

    % region specified by rect is within source surface
    xi = intersect(find(X>=region(1)),find(X<=region(3)));
    yi = intersect(find(Y>=region(2)),find(Y<=region(4)));

    Xn = X(xi);
    Yn = Y(yi);
    Zn = Z(min(yi):max(yi),min(xi):max(xi));
    np(1) = max(size(xi));
```

```
    np(2) = max(size(yi));

    % debug
    if isempty(Xn),
        region
        min(X)
        max(X)
        min(Y)
        max(Y)
        params
    end

    np(3) = Xn(1);
    np(4) = Yn(max(size(yi)));
else
    Xn = [];
    Yn = [];
    Zn = [];
    np = [];
end

return
```

D.2.26 Surface_Extract_Square

```
function [Xn,Yn,Zn,np] = surface_extract_square(X,Y,Z,params,option)

% function [Xn,Yn,Zn,np] = surface_extract_square(X,Y,Z,params,option)
%
% Extracts the largest square region of a height field Z. If option is
% nonzero, extracts the largest square region of Z that has a width
% and height that is a power of 2. Option is an optional parameter.
%
% INPUTS
%  X = x-coordinates of altitude samples (columns of Z)
%  Y = y-coordinates of altitude samples (rows of Z)
%  Z = altitude samples
%  params = 1x6 vector of surface parameters, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  option = 0 (largest square region extracted)
%           1 (largest square region of width 2^n, n=integer, extracted)
%
% OUTPUTS
%  Xn = x-coordinates of new altitude samples (columns of Zn)
%  Yn = y-coordinates of new altitude samples (rows of Zn)
%  Zn = new altitude samples
%  np = 1x6 vector of surface parameters for new surface, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% Author: Chris Carr, May 2001.

grid=Z;
w = size(grid,2); % cols
h = size(grid,1); % rows

if nargin<5,
    option=0;
end

if option~=0,
    % make new grid dimensions a power of 2
```

```
    pwr = min(floor(log10(w)/log10(2)), floor(log10(h)/log10(2)));
    snew = 2^pwr;
    Zn = grid(1:snew,1:snew);
    Xn = X(1:snew);
    Yn = Y(1:snew);
    np = [snew snew params(3) params(4)+params(5)*(h-snew) params(5) params(6)];
else
    if (w<h),
        % new grid should be w by w
        Zn = grid(1:w,1:w);
        Xn = X(1:w);
        Yn = Y(1:w);
        np = [w w params(3) params(4)+params(5)*(h-w) params(5) params(6)];
    else
        % new grid should be h by h
        Zn = grid(1:h,1:h);
        Xn = X(1:h);
        Yn = Y(1:h);
        np = [h h params(3) params(4)+params(5) params(5) params(6)];
    end
end

return
```

D.2.27 Surface_Extract_Valid

```
function [Xn,Yn,Zn,np]=surface_extract_valid(X,Y,Z,params)

% function [Xn,Yn,Zn,np]=surface_extract_valid(X,Y,Z,params)
%
% Extracts the largest valid region of a height field Z that
% contains no NaN entries.
%
% INPUTS
%  X = x-coordinates of altitude samples (columns of Z)
%  Y = y-coordinates of altitude samples (rows of Z)
%  Z = altitude samples
%  params = 1x6 vector of surface parameters, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% OUTPUTS
%  Xn = x-coordinates of new altitude samples (columns of Zn)
%  Yn = y-coordinates of new altitude samples (rows of Zn)
%  Zn = new altitude samples
%  np = 1x6 vector of surface parameters for new surface, of the form:
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% Note: It is assumed that NaN entries are only found on the border of
% the original height field.
%
% Author: Chris Carr, May 2001.

if verbose,
    disp('Extracting valid surface region (SURFACE_EXTRACT_VALID)');
end

g = Z;
Xn = X;
Yn = Y;

ctr=0;
```

```
bHasNaN = 1;  % assume contains NaN entries
while(bHasNaN>0),
   top = g(1,:);
   bottom = g(size(g,1),:);
   left = g(:,1);
   right = g(:,size(g,2));

   a = sum(isnan(top),2);
   b = sum(isnan(bottom),2);
   c = sum(isnan(left),1);
   d = sum(isnan(right),1);

   if a>0,
       % remove top
       g = g(2:size(g,1),:);
       if verbose,
           disp('Removing north row of surface (SURFACE_EXTRACT_VALID)');
       end
        Yn = Yn(2:size(Yn,2));
   end
   if b>0,
       % remove bottom
       g = g(1:size(g,1)-1,:);
       if verbose,
           disp('Removing south row of surface (SURFACE_EXTRACT_VALID)');
       end
       Yn = Yn(1:size(Yn,2)-1);
   end
   if c>0,
       % remove left
       g = g(:,2:size(g,2));
      if verbose,
          disp('Removing west row of surface (SURFACE_EXTRACT_VALID)');
      end
       Xn = Xn(2:size(Xn,2));
   end
   if d>0,
       % remove right
       g = g(:,1:size(g,2)-1);
       if verbose,
           disp('Removing east row of surface (SURFACE_EXTRACT_VALID)');
       end
       Xn = Xn(1:size(Xn,2)-1);
   end

   bHasNaN = a+b+c+d;
   if or(size(g,1)<=2,size(g,2)<=2),
        % grid too small
        Xn=[];
        Yn=[];
        Zn=[];
        np=[];
        bHasNaN=0;
   end
   ctr = ctr+1;
end

Zn = g;

%np = [size(Zn,2), size(Zn,1), Xn(1),Yn(1), params(5), params(6)];
np = [size(Zn,2), size(Zn,1), min(Xn),min(Yn), params(5), params(6)];
```

D.2.28 Surface_Georeference

```
function [X,Y]=surface_georeference(params)

% function [X,Y]=surface_georeference(params)
%
% Georeferences a height field using the parameters params.
%
% INPUTS
%  params = 1x6 vector of surface parameters of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% OUTPUTS
%  X      = x-coordinates of altitude samples for columns of Z
%  Y      = y-coordinates of altitude samples for rows of Z
%
% Author: Chris Carr, May 2001.

width = params(1);
height = params(2);
xllcorner = params(3);
yllcorner = params(4);
cellsize = params(5);

X = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
Y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);
```

D.2.29 Surface_Import

```
function [X,Y,Z,params]=surface_import(fname,filter)
% function [X,Y,Z,params]=surface_import(fname,filter)
%
% Loads an ASCII file containing a height field. The filename
% fname must be in MATLAB .mat ASCII format. The first six
% entries in the file must be the width, height, xllcorner,
% yllcorner, cellsize, and nodatavalue. These entries must be
% followed by width*height data entries that represent samples
% on the height field grid.
%
% INPUTS
%  fname - a file name to load; should be of the form 'mysurface.mat'
%  filter - if non-zero, output grid Z will contain NaN entries for
%           each height sample equal to nodatavalue as specified in
%           the file fname.
%
% OUTPUTS
%  X - the x-coordinates of the columns of Z
%  Y - the y-coordinates of the rows of Z
%  Z - a matrix of altitude samples
%  params - a 1x6 vector of height field parameters; of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%
% Author: Chris Carr

% read in the file
eval(sprintf('load -ascii %s;',fname));

% get variable name
```

```
va = fname(1:length(fname)-4);

% read in the parameters
eval(sprintf('width = %s(1);',va));
eval(sprintf('height = %s(2);',va));
eval(sprintf('xllcorner = %s(3);',va));
eval(sprintf('yllcorner = %s(4);',va));
eval(sprintf('cellsize = %s(5);',va));
eval(sprintf('nodatavalue = %s(6);',va));

% filter the array of altitude samples if necessary
if filter~=0,
    % filter nodatavalue entries
    eval(sprintf('f = find(%s==nodatavalue);',va));
    eval(sprintf('%s(f)=NaN;',va));
    eval('nodatavalue = NaN;');
end

% build the height field data
Z = zeros(height,width);
for i=1:height,
    if (eval(sprintf('(i*width+5) <= size(%s,2)',va))),
        eval(sprintf('Z(i,:)=%s((i-1)*width+6:i*width+5);',va));
    end
end
X = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
Y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);
params = [width height xllcorner yllcorner cellsize nodatavalue];

return
```

D.2.30 Surface_Merge2

```
function Zn=surface_merge2(Z1,Z2,option)
% function Zn = surface_merge2(Z1,Z2,option)
%
% Merges the surfaces Z1 and Z2. Entries in Zn that have value nodatavalue
% in either Z1 or Z2 but not both will contain valid data. Entries in Zn
% that have nodatavalue in both Z1 and Z2 will have value nodatavalue.
% Entries in Zn for which both Z1 and Z2 contain valid data will be assigned
% according to the option input (see below). Invalid data in Z1 and Z2 must
% be indicated by NaN entries.
%
% INPUTS
%  Z1 = a matrix representing a surface
%  Z2 = a matrix representing a surface (must be same size as Z1)
%  option = 0 (default value, ADD elements of Z1 and Z2 to produce Zn)
%           1 (MULTIPLY elements of Z1 and Z2 to produce Zn)
%
% OUTPUTS
%  Zn = the new merged surface
%
% Author: Chris Carr, June 2001.

if nargin<3,
    option=0;
end

Zn = ones(size(Z1,1),size(Z1,2))*NaN;

Z1i_valid = find(isnan(Z1)==0);
```

```
Z1i_invalid = find(isnan(Z1)==1);
Z2i_valid = find(isnan(Z2)==0);
Z2i_invalid = find(isnan(Z2)==1);

Zni_valid_both = intersect(Z1i_valid,Z2i_valid); % valid entries in both surfaces
Zni_invalid_both = intersect(Z1i_invalid,Z2i_invalid); % invalid entries in both surfaces

Z1i_valid_only = intersect(Z1i_valid,Z2i_invalid);
Z2i_valid_only = intersect(Z2i_valid,Z1i_invalid);

% merge the data
if (option==0),
    Zn(Zni_valid_both) = Z1(Zni_valid_both)+Z2(Zni_valid_both);
elseif (option==1),
    Zn(Zni_valid_both) = Z1(Zni_valid_both).*Z2(Zni_valid_both);
end

Zn(Z1i_valid_only) = Z1(Z1i_valid_only);
Zn(Z2i_valid_only) = Z2(Z2i_valid_only);
```

D.2.31 Surface_Plot

```
function surface_plot(hFig,X,Y,Z,sp)

% function surface_plot(hFig,X,Y,Z,sp)
%
% Plots a surface on an existing figure with handle hFig.
%
% INPUTS
%  hFig = a handle to an existing figure
%  X    = x-coordinates of altitude samples (columns of Z)
%  Y    = y-coordinates of altitudes samples (rows of Z)
%  Z    = height field of altitude samples
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig
%
% Author: Chris Carr, May 2001.

figure(hFig);
if nargin==5,
    subplot(sp(1),sp(2),sp(3));
end

surf(X,Y,Z);
shading flat;
view(2);
set(gca,'PlotBoxAspectRatio',[1 1 1]);
xlabel('Meters');
ylabel('Meters');
```

D.2.32 Surface_Plot_Add_Scale

```
function surface_plot_add_scale(hFig,Z,params,scale,sp,op)
%function surface_plot_add_scale(hFig,Z,params,scale,sp,op)
%
% Adds a scale to a surface plot.
% INPUTS
%  hFig = a handle to an existing figure
%  params = surface parameters, of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
```

```
%  scale = scale vector of the form [sx sy sz]. Each entry of the
%          scale vector represents the scale length in the x, y, and z directions
%  sp    = an optional argument of form [a b c] that specifies a subplot triple
%          for the figure specified by hFig
%  op    = optional argument, =0 (default) 3d scale
%                             =1 2d scale (xy plane)
%
% Author: Chris Carr, May 2001.

figure(hFig);
if nargin<5,
    sp = [1 1 1];
end
if nargin<4,
    scale = [1000 1000 1000];
end
if nargin<6,
    op = 0;
end

% position at lower left corner
f = 10;
offset = (max(max(Z))-min(min(Z)))/10;
x0 = params(3)+size(Z,2)/f*params(5);
y0 = params(4)+size(Z,1)/f*params(5);
z0 = Z(size(Z,1)-ceil(size(Z,1)/f),ceil(size(Z,2)/f))+offset;
x1 = x0+scale(1);
y1 = y0+scale(2);
z1 = z0+scale(3);

np = get(gca,'nextplot');
set(gca,'nextplot','add');
set(gca,'linewidth',2);
plot3([x0 x1],[y0 y0],[z0 z0],'k-');
plot3([x0 x0],[y0 y1],[z0 z0],'k-');
if (op==0),
    plot3([x0 x0],[y0 y0],[z0 z1],'k-');
end
set(gca,'nextplot',np);
```

D.2.33 Surface_Plot_Minimal

```
function surface_plot_minimal(hFig,X,Y,Z,sp)

% function surface_plot_minimal(hFig,X,Y,Z,sp)
%
% Plots a surface on an existing figure with handle hFig.
% Tick labels are eliminated, border is enlarged slightly,
% and no x and y labels are added. Ticks are also eliminated.
%
% INPUTS
%  hFig = a handle to an existing figure
%  X    = x-coordinates of altitude samples (columns of Z)
%  Y    = y-coordinates of altitudes samples (rows of Z)
%  Z    = height field of altitude samples
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig
%
% Author: Chris Carr, May 2001.

figure(hFig);
```

```
if nargin==5,
    subplot(sp(1),sp(2),sp(3));
end

surf(X,Y,Z);
shading flat;
view(2);
set(gca,'xlim',[min(X) max(X)]);
set(gca,'ylim',[min(Y) max(Y)]);
set(gca,'PlotBoxAspectRatio',[1 1 1]);
set(gca,'xticklabel',[]);
set(gca,'xtick',[]);
set(gca,'yticklabel',[]);
set(gca,'ytick',[]);
set(gca,'LineWidth',2);
set(gca,'box','on');
```

D.2.34 Surface_Plot_Minimal_3d

```
function surface_plot_minimal_3d(hFig,X,Y,Z,sp,v,C)

% function surface_plot_minimal_3d(hFig,X,Y,Z,sp,v,C)
%
% Plots a surface on an existing figure with handle hFig.
% Tick labels are eliminated and ticks are eliminated.
% 3D view is shown.
%
% INPUTS
%  hFig = a handle to an existing figure
%  X    = x-coordinates of altitude samples (columns of Z)
%  Y    = y-coordinates of altitudes samples (rows of Z)
%  Z    = height field of altitude samples
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig
%  v    = vertical exaggeration factor, based on x axis dimensions
%  C    = colordata matrix (optional parameter)
%
% Author: Chris Carr, May 2001.

figure(hFig);
if nargin<5,
    sp = [1 1 1];
end
if nargin<6,
    v = 1;
end
if nargin<7,
    C = Z; % color by Z by default
end

subplot(sp(1),sp(2),sp(3));
h = surf(X,Y,Z);
shading interp;
view([36.1356 29.8137]);
set(gca,'CameraPositionMode','manual');
set(gca,'CameraPosition',[675007 3.93394e+006 32495.3]);
set(gca,'CameraTargetMode','manual');
set(gca,'CameraTarget',[643128 3.9776e+006 1517.74]);
set(gca,'CameraViewAngleMode','manual');
set(gca,'CameraViewAngle',4.67766);
```

```
set(gca,'xlim',[min(X) max(X)]);
set(gca,'ylim',[min(Y) max(Y)]);
set(gca,'PlotBoxAspectRatio',[1 1 1]);
set(gca,'xticklabel',[]);
set(gca,'xtick',[]);
set(gca,'yticklabel',[]);
set(gca,'ytick',[]);
set(gca,'zticklabel',[]);
set(gca,'ztick',[]);
set(gca,'LineWidth',0.01);
set(gca,'box','off');
zc = mean(get(gca,'zlim'));
dx = max(X)-min(X);
set(gca,'zlim',[zc-dx/(2*v) zc+dx/(2*v)]);
grid off;
set(gca,'visible','off');

set(h,'cdata',C);
set(h,'facelighting','phong');
set(h,'edgelighting','phong');
```

D.2.35 Surface_Plot_Waypoints

```
function surface_plot_waypoints(h,wp_x,wp_y,wp_z,sbl,sp,op)
% function surface_plot_waypoints(h,wp_x,wp_y,wp_z,sbl,sp,op)
%
% Plots waypoints on an existing surface plot, with figure handle specified
% by figure handle h.
%
% INPUTS
%  h = handle of a valid figure of a surface plot
%  wp_x = x coordinates of waypoints, 1xn vector for n waypoints
%  wp_y = y coordinates of waypoints, 1xn vector for n waypoints
%  wp_z = z coordaintes of waypoints, 1xn vector for n waypoints
%  sbl = symbol to use for plotting
%  sp = optional parameter; specifies a subplot, of the form [a b c]
%  op = optional parameter; op=0 (default) draws waypoints only
%                           op=1 draws waypoints with connecting straight lines
%                           op=2 draws waypoints with surface lines (projection onto surface)
%
% Author: Chris Carr, June 2001.

if nargin<5,
    sbl='.';
end
if nargin<6,
    sp = [1 1 1];
end
if nargin<7,
    op = 0
end

figure(h);
np = get(gca,'nextplot');
set(gca,'nextplot','add');
subplot(sp(1),sp(2),sp(3));

if (op==0),
    % draw waypoints only
```

```
    plot3(wp_x,wp_y,wp_z,sbl);
elseif (op==1),
    % draw straight connecting lines
    plot3(wp_x,wp_y,wp_z,sbl);
    plot3(wp_x,wp_y,wp_z,'-');
elseif (op==2),
    % draw waypoints
    plot3(wp_x,wp_y,wp_z,sbl);
    %then draw surface lines of projections of waypoints onto surface
end

set(gca,'nextplot',np);
```

D.2.36 FloodFill

```
function flood_fill(i,j)

global Z;
global M;
global Rp;
global limits;
global mytype;

a = (i<=size(Z,1));
b = (j<=size(Z,2));

if and(a,b),

% mark point as handled
M(i,j)=1;

if Z(i,j) >= limits(1),
    if Z(i,j) <= limits(2),

        Rp(i,j)=1;

        %sx = size(Rp,1)-1;
        %sy = size(Rp,2)-1;
        sy = size(Rp,1)-1;
        sx = size(Rp,2)-1;


        % 4-connected regions
        if i<=sy,
            if (M(i+1,j)==0),
                M(i+1,j)=-1; % down
            end
        end
        if i>=2,
            if (M(i-1,j)==0),
                M(i-1,j)=-1; % up
            end
        end
        if j<=sx,
            if (M(i,j+1)==0), % if not marked yet...
                M(i,j+1)=-1; % right
            end
        end
        if j>=2,
            if (M(i,j-1)==0),
                M(i,j-1)=-1; % left
```

```
            end
        end

        if mytype==1,
            % 8-connected regions
            if and(i<=sy,j<=sx),
                if (M(i+1,j+1)==0),
                    M(i+1,j+1)=-1; % down and right
                end
            end
            if and(i<=sy,j>=2),
                if (M(i+1,j-1)==0),
                    M(i+1,j-1)=-1; % down and left
                end
            end
            if and(i>=2,j<=sx),
                if (M(i-1,j+1)==0),
                    M(i-1,j+1)=-1; % up and right
                end
            end
            if and(i>=2,j>=2),
                if (M(i-1,j-1)==0),
                    M(i-1,j-1)=-1; % up and left
                end
            end
        end
    end
end
end
return;
```

D.3 Point Functions

D.4 Graph Functions

D.4.1 Graph_Compute_Adjacency

```
function [n,A] = graph_compute_adjacency(N,E,ut)

% function [n,A] = graph_compute_adjacency(N,E,ut)
%
% Computes an adjacency matrix given a set of nodes N and
% a set of edges E
%
% INPUTS
%  N = 1 x k vector of node ids where k is the number of nodes
%  E = 2 x e vector of edges where e is the number of edges
%  ut = (0 = default, digraph adjacency matrix)
%       (1 = adjacency matrix is upper triangular)
```

```
%       (2 = put edge in adjacency matrix twice)
%
% OUTPUTS
%  n = 1 x k vector of node ids, same as N
%  A = n x n matrix with nonzero entries if A(i,j) corresponds to an edge in E
%
% author: Chris Carr, May 2001.

if (verbose),
    disp('Computing Adjacency Matrix (GRAPH_COMPUTE_ADJACENCY)');
end

if nargin<3,
    ut=0;
end

A = zeros(size(N,2),size(N,2));

for i=1:size(E,2),
    e_index_a = find(N==E(1,i));
    e_index_b = find(N==E(2,i));

    if ut==1,
        if e_index_a < e_index_b,
            A(e_index_a,e_index_b) = 1;
        else
            A(e_index_b,e_index_a) = 1;
        end
    elseif ut==2,
        A(e_index_a,e_index_b)=A(e_index_a,e_index_b)+1;
        A(e_index_b,e_index_a)=A(e_index_b,e_index_a)+1;
    else
        A(e_index_a,e_index_b)=1;
    end
end
n=N;
```

D.4.2 Graph_Compute_Degree

```
function [n,d]=graph_compute_degree(N,E)

% function [n,d] = graph_compute_degree(N,E)
%
% Computes the degree of a set of nodes N given a set of edges E
%
% INPUTS
%  N = 1 x k vector of node ids where k is the number of nodes
%  E = 2 x e vector of edges where e is the number of edges
%
% OUTPUTS
%  n = 1 x k vector of node ids, same as N
%  d = 1 x k vector of node degrees
%
% author: Chris Carr, May 2001.

if (verbose),
    disp('Computing Graph Degree (GRAPH_COMPUTE_DEGREE)');
end

% compute node half-edges
if min(size(E))~=0,
```

```
    Ea = E(1,:);
    Eb = E(2,:);

    for i=1:size(N,2),
        % compute degree for each node
        a = find(Ea==N(i));
        b = find(Eb==N(i));
        d(i) = length(a)+length(b);
    end
    n = N;
else
    n = N;
    d = zeros(size(n,1),size(n,2));
end
```

D.4.3 Graph_Compute_Dsp

```
function [Vm,Em,D,Cm]=graph_compute_dsp(V,E,C,n)

% function [Vm,Em,D,Cm]=graph_compute_dsp(V,E,C,n)
%
% Computes Dijkstra's Shortest Path for a weighted connected graph
% specified by G={V,E} and an initial vertex specified by n.
%
% INPUTS
%  V = 1 x k vector of vertices in the graph
%  E = 1 x e vector of edges of the graph
%  C = 1 x e vector of cost associated with each edge
%  n = index into vector of vertices V(n) = starting node
%
% OUTPUTS
%  Vm = shortest path cost set of vertices (same as V)
%  Em = minimum cost set of edges
%  D  = distance vector (contains distance to each node in V starting
%       from initial vertex specified by n)
%  Cm  = distance vector (contains distance for each edge in Em)
%
% Author: Chris Carr, August 2001.
% Based on Dijkstra's Shortest Path algorithm as described in Gross & Yellen, 1999.

% initialize the Dijkstra tree
Vm = [n];
Em = [];
D = [0];
Cm = [];

% test if input graph has edges
if isempty(E),
    return;
end

% initialize the set of frontier edges
F = [];
Fv = [];
Fc = [];

Fn = [];
Fcn = [];
Fvn = [];

% other initializations
```

```
give_up = 0;
iter=0;
maxiter = size(V,2);

% while Dijkstra tree {Vm,Em} does not span G={V,E}
while and(size(Vm,2)~=size(V,2),not(give_up)),
    % find new frontier edges of T={Vm,Em}
    for idx_n=1:size(Vm,2),
        % find neighbors of Vm(idx_n)
        my_node = Vm(idx_n);
        % find cost of/distance to current node
        my_node_D = D(idx_n);
        % find ids of neighbors
        idx_neigh1 = find(E(1,:)==my_node);
        idx_neigh2 = find(E(2,:)==my_node);

        % if neighbors are not already in Vm, add neighbors to frontier list
        for n=1:size(idx_neigh1,2),
            % get the neightbor id
            neigh1 = E(2,idx_neigh1(n));
            % check if neigh is in Vm
            if isempty(find(Vm==neigh1)),
                % add edge to frontier list
                F(1,size(F,2)+1)=idx_neigh1(n);
                Fc(1,size(Fc,2)+1)=C(idx_neigh1(n))+my_node_D;
                Fv(1,size(Fv,2)+1)=neigh1;
            end
        end
        for n=1:size(idx_neigh2,2),
            % get the neighbor id
            neigh2 = E(1,idx_neigh2(n));
            % check if neigh is in Vm
            if isempty(find(Vm==neigh2)),
                % add edge to frontier list
                F(1,size(F,2)+1)=idx_neigh2(n);
                Fc(1,size(Fc,2)+1)=C(idx_neigh2(n))+my_node_D;
                Fv(1,size(Fv,2)+1)=neigh2;
            end
        end
    end

    % let e be a frontier edge for T that has the smallest P-value (cost or distance)
    if isempty(Fc),
        give_up = 1;
    else
        idx_min_D = find(Fc==min(Fc));
        idx_min_D = idx_min_D(1,1);
        e = F(idx_min_D);
        v = Fv(idx_min_D);
        d = Fc(idx_min_D);
        % add edge e (and vertex v) to tree T
        Vm(1,size(Vm,2)+1)=v;
        Em(1:2,size(Em,2)+1)=E(1:2,e);
        Cm(size(Em,2)+1)=C(e);
        % dist[v]=P(e)
        D(1,size(D,2)+1)=d;

        % remove all edges that include v from the frontier edge list
        idx_keep = find(Fv~=v);
        for i=1:size(idx_keep,2),
            Fn(i)=F(idx_keep(i));
            Fcn(i)=Fc(idx_keep(i));
```

```
            Fvn(i)=Fv(idx_keep(i));
        end
        F = Fn;
        Fc = Fcn;
        Fv = Fvn;

        % give up if no progress is being made
        % (likely that G={V,E} is not connected)
        iter = iter+1;
        if (iter>=maxiter),
            give_up=1;
        end
    end
    F = [];
    Fc =[];
    Fv =[];
end
% return Dijkstra tree T and its vertex labels
return;
```

D.4.4 Graph_Compute_MCST

```
function [Vm, Em]=graph_compute_mcst(V,E,C,n);

% function [Vm Em]=graph_compute_mcst(V,E,C,n)
%
% Computes the minimum cost spanning tree of the graph G={V,E}
% given a edge cost vector C and an initial node n.
%
% INPUTS
%  V = 1 x k vector of vertices in the graph
%  E = 1 x e vector of edges of the graph
%  C = 1 x e vector of cost associated with each edge
%  n = index into vector of vertices V(n) = starting node
%
% OUTPUTS
%  Vm = minimum cost set of vertices (same as V)
%  Em = minimum cost set of edges
%
% Author: Chris Carr, August 2001.
% Based on Prim's algorithm as described in Gross & Yellen, 1999.

% initialize T = {Vm,Em}
Vm = [n];
Em = [];
% initialize frontier nodes
F =[];
Fc=[];
Fv=[];
% initialize other variables
give_up = 0;
iter = 0;
maxiter = size(V,2);

% iterate until tree spans the set or give up
while and(size(Vm,2)~=size(V,2),not(give_up)),
    % find new frontier edges of T
    for idx_n=1:size(Vm,2),
        % find neighbors of Vm(idx_n)
        my_node = Vm(idx_n);
        idx_neigh1 = find(E(1,:)==my_node);
```

```
        idx_neigh2 = find(E(2,:)==my_node);
        % if neighbors are not already in Vm, add neighbors to frontier list
        for n=1:size(idx_neigh1,2),
            % get the neightbor id
            neigh1 = E(2,idx_neigh1(n));
            % check if neigh is in Vm
            if isempty(find(Vm==neigh1)),
                % add edge to frontier list
                F(1,size(F,2)+1)=idx_neigh1(n);
                Fc(1,size(Fc,2)+1)=C(idx_neigh1(n));
                Fv(1,size(Fv,2)+1)=neigh1;
            end
        end
        for n=1:size(idx_neigh2,2),
            % get the neighbor id
            neigh2 = E(1,idx_neigh2(n));
            % check if neigh is in Vm
            if isempty(find(Vm==neigh2)),
                % add edge to frontier list
                F(1,size(F,2)+1)=idx_neigh2(n);
                Fc(1,size(Fc,2)+1)=C(idx_neigh2(n));
                Fv(1,size(Fv,2)+1)=neigh2;
            end
        end
    end

    % find frontier edge e with smallest edge weight
    mc = min(Fc);
    idx_mc = find(Fc==mc);
    idx_mc = idx_mc(1,1); % choose first edge if multiple edges have equal weights

    % find non-tree vertex v of that edge
    v = Fv(idx_mc);
    % add v to Vm, and add edge e to Em
    Vm(1,size(Vm,2)+1)=v;
    Em(1:2,size(Em,2)+1)=E(1:2,F(idx_mc));
    % remove all edges that include v from the frontier edge list
    idx_keep = find(Fv~=v);
    for i=1:size(idx_keep,2),
        Fn(i)=F(idx_keep(i));
        Fcn(i)=Fc(idx_keep(i));
        Fvn(i)=Fv(idx_keep(i));
    end
    F = Fn;
    Fc = Fcn;
    Fv = Fvn;

    iter = iter+1;
    if iter==maxiter,
        give_up=1;
    end
end
```

D.4.5 Graph_Create_NEP

```
function [N,E,P]=graph_create_NEP(Z,N,P,op1,op2,op3)

% function graph_create_NEP(Z,N,P,op1,op2)
%
% builds a G_NEP graph given a height field Z,
% a set of node IDs N, and a set of points P
```

```
%
% INPUTS
%  Z = height field (i x j matrix)
%  N = 1 x n vector of node ids, where n is number of nodes
%  P = 3 x n vector, where P(1,:)= x coords of nodes
%                         P(2,:)= y coords of nodes
%                         P(3,:)= z coords of nodes
%  op1 = option (0 = default: normal line of sight visibility)
%              (1 = distance limited line of sight visibility)
%  op2 = option (if op1=1, specifies maximum line-of-sight distance in cells)
%  op3 = specifies cellspacing for surface; required if op2=1.
%
% OUTPUTS
%  N = 1 x n vector of node ids (same as input)
%  E = 2 x e vector of edges, where E(1,:) = id of originating node
%                                   E(2,:) = id of completing node
%  P = 3 x n vector of points (same as input vector)
%
% Author: Chris Carr, May, 2001.

if (verbose),
    disp('Creating Nodes/Edges/Points Graph (GRAPH_CREATE_NEP)');
end
if nargin<6,
    op3=30;
end
if nargin<5,
    op2 = 3;
end
if nargin<4,
    op1 = 0;
end

n = size(P,2);
ctr=0;
E=[];

for i=1:n,
    for j=(i+1):n,
        b = graph_create_NEP_los(Z,P(1,i),P(2,i),P(3,i),P(1,j),P(2,j),P(3,j));
        if (b~=0),
            % visibility exists
            % check options
            if op1==0,
                ctr=ctr+1;
                E(1,ctr)=N(i);
                E(2,ctr)=N(j);
            elseif op1==1,
                dd = sqrt(((P(3,j)-P(3,i))/op3)^2+(P(2,j)-P(2,i))^2+(P(1,j)-P(1,i))^2);
                if dd<=op2,
                    % visible and within specified distance limit
                    ctr=ctr+1;
                    E(1,ctr)=N(i);
                    E(2,ctr)=N(j);
                end
            else
                % do nothing
            end
        end
    end
end
```

```
if (verbose),
    disp(sprintf('%d edges created (GRAPH_CREATE_NEP)',ctr));
end
```

D.4.6 Graph_Compute_NEP_los

```
function b = graph_create_NEP_los(zz,x1,y1,z1,x2,y2,z2)

% initialize position to p1
numsteps = round(sqrt((x2-x1)^2+(y2-y1)^2));

if numsteps==0,
    % points are within one cell on the surface; visibility is guaranteed
    % due to points being below the resolution of the surface model
    b=1;
else
    t=linspace(0,1,numsteps);

    % compute equation of line for all t
    p_x = x1 + t.*(x2-x1);
    p_y = y1 + t.*(y2-y1);
    p_z = z1 + t.*(z2-z1);

    zcheck = zeros(1,numsteps);
    % check that grid value at current location is below equation of line
    for i=1:numsteps,
        zcheck(i) = zz(round(p_y(i)),round(p_x(i)));
    end
    j = sum(find(zcheck>p_z));

    if (j>0),
        b = 0;
    else
        b=1;
    end
end
```

D.4.7 Graph_Create_Points

```
function [n, P] = graph_create_points(Z,type,number,offset)

% function [n, P] = graph_create_points(Z, type, number, offset)
%
% Makes number of points distributed on a grid which has
% height i=size(Z,1) and width j=size(Z,2).
%
% INPUTS
%  Z      = matrix of size i x j
%  type   = 1 (UNIFORM distribution),
%           2 (Gaussian Distribution),
%  number = number of points to generate
%  offset = z-coordinate of points relative to z-coordinate of surface
%           (i.e. height of point above surface)
%
% OUTPUTS
%  n      = ids for points
%  P      = vector of points, P(1,:) is 1 x number vector of x-coordinates
%                             P(2,:) is 1 x number vector of y-coordinates
%                             P(3,:) is 1 x number vector of z-coordinates
```

```
%
% Author: Chris Carr, May 2001.

width = size(Z,2);
height = size(Z,1);

P=[];

if type==1,
    % uniform random distribution
    x = floor(rand(1,number)*(width))+1;
    y = floor(rand(1, number)*(height))+1;
    n = linspace(1,number,number); % ids
elseif type==2,
    % gaussian random distribution
    valid = 0;
    ctr=0;
    while(~valid),
        x = floor(randn(1, number)*(width)/4 + width/2)+1;
        y = floor(randn(1, number)*(height)/4 + height/2)+1;
        n = linspace(1,number,number); % ids

        a = isempty(find(x>size(Z,2)));
        b = isempty(find(x<1));
        c = isempty(find(y>size(Z,1)));
        d = isempty(find(y<1));

        valid = and(and(a,b),and(c,d));
        ctr = ctr+1;
        if verbose,
            ctr
        end
    end
else
    %
    disp('Invalid option for MakePoints');
    x=[];
    y=[];
    z=[];
    n=[];
end

if size(n,1)~=0,
    for i=1:number,
        P(1,i)=x(i);
        P(2,i)=y(i);
        P(3,i)=Z(y(i),x(i))+offset;
    end
end
return;
```

D.4.8 Graph_Create_Points_Restricted

```
function [n, P] = graph_create_points_restricted(Z,type,number,offset,R)

% function [n, P] = graph_create_points_restricted(Z, type, number, offset, R)
%
% Makes number of points distributed on a grid which has
% height i=size(Z,1) and width j=size(Z,2). Points created in restricted zones
```

```
% are cast out and additional points created until enough valid points are created.
%
% INPUTS
%  Z      = matrix of size i x j
%  type   = 1 (UNIFORM distribution),
%           2 (Gaussian Distribution),
%  number = number of points to generate
%  offset = z-coordinate of points relative to z-coordinate of surface
%           (i.e. height of point above surface)
%  R      = reachability map of size i x j. R(j,i)=0 is a restricted area, while
%           R(j,i)=1 is an accessible or reachable area.
%
% OUTPUTS
%  n      = ids for points
%  P      = vector of points, P(1,:) is 1 x number vector of x-coordinates
%                             P(2,:) is 1 x number vector of y-coordinates
%                             P(3,:) is 1 x number vector of z-coordinates
%
% Author: Chris Carr, May 2001.

width = size(Z,2);
height = size(Z,1);

P=[];

if type==1,
    % uniform random distribution
    ctr=1;
    while (ctr<=number),
        x(ctr) = floor(rand(1,1)*(width))+1;
        y(ctr) = floor(rand(1,1)*(height))+1;
        n(ctr) = ctr; % ids
        if R(y,x)==1,
            ctr=ctr+1;
        end
    end
elseif type==2,
    % gaussian random distribution
    ctr=1;
    while(ctr<=number),
        x(ctr) = floor(randn(1,1)*(width)/4 + width/2)+1;
        y(ctr) = floor(randn(1,1)*(height)/4 + height/2)+1;
        n(ctr) = ctr; % ids

        a = isempty(find(x>size(Z,2)));
        b = isempty(find(x<1));
        c = isempty(find(y>size(Z,1)));
        d = isempty(find(y<1));
        in_bounding_box = and(and(a,b),and(c,d));
        if and(in_bounding_box,R(y,x)==1),
            ctr = ctr+1;
        end
    end
else
    %
    disp('Invalid option for parameter type in GRAPH_CREATE_POINTS_RESTRICTED');
    x=[];
    y=[];
    z=[];
    n=[];
end
```

```
if size(n,1)~=0,
    for i=1:number,
        P(1,i)=x(i);
        P(2,i)=y(i);
        P(3,i)=Z(y(i),x(i))+offset;
    end
end

return;
```

D.4.9 Graph_Plot_Edges

```
function h = graph_plot_edges(hFig,N,E,P,params,sym,sp,lw)

% function graph_plot_edges(hFig,N,E,P,params,sym,sp,lw)
%
% Plots the points of an augmented graph on the figure with handle hFig.
%
% INPUTS
%  hFig = a handle to an existing figure
%  N    = a 1 x n vector of nodes
%  E    = a 2 x e vector of edges, where e is the number of edges; entries refer to nodes
%  P    = a 3 x n vector of n points, P(1,:)=xcoord, P(2,:)=ycoord, P(3,:)=zcoord
%  params = 1x6 vector of surface parameters of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  sym  = a symbol to use for plotting (optional argument)
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig
%  lw   = linewidth for edges
%
% OUTPUTS
%  h = vector of handles to plotted edges
%
% Author: Chris Carr, August 2001

figure(hFig);
if nargin<7,
    lw = 0.5;
end
if nargin<6,
    sym='-k';
end
if nargin==7,
    subplot(sp(1),sp(2),sp(3));
end

for i=1:size(E,2),
    e_index_a(i) = find(N==E(1,i));
    e_index_b(i) = find(N==E(2,i));
    ep_a(1,i) = P(1,e_index_a(i));
    ep_a(2,i) = P(2,e_index_a(i));
    ep_a(3,i) = P(3,e_index_a(i));
    ep_b(1,i) = P(1,e_index_b(i));
    ep_b(2,i) = P(2,e_index_b(i));
    ep_b(3,i) = P(3,e_index_b(i));

    ex = [ep_a(1,i) ep_b(1,i)];
    ey = [ep_a(2,i) ep_b(2,i)];
    ez = [ep_a(3,i) ep_b(3,i)];
```

```
    px = (ex-1)*params(5)+params(3);
    py = (params(2)-ey)*params(5)+params(4);
    pz = ez;

    h(i) = plot3(px,py,pz,sym);
    set(h(i),'LineWidth',lw);
    set(gca,'nextplot','add');
end
```

D.4.10 Graph_Plot_Points

```
function h = graph_plot_points(hFig,P,params,sym,sp)

% function graph_plot_points(hFig,P,params,sym,sp)
%
% Plots the points of an augmented graph on the figure with handle hFig.
%
% INPUTS
%  hFig = a handle to an existing figure
%  P    = a 3 x n vector of n points, P(1,:)=xcoord, P(2,:)=ycoord, P(3,:)=zcoord
%  params = 1x6 vector of surface parameters of the form
%               [width height xllcorner yllcorner cellsize nodatavalue]
%  sym  = a symbol to use for plotting (optional argument)
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig

figure(hFig);
if nargin<5,
    sp = [1 1 1];
end
if nargin<4,
    sym='.';
end
if nargin==5,
    subplot(sp(1),sp(2),sp(3));
end

px = (P(1,:)-1)*params(5)+params(3);
py = (params(2)-P(2,:))*params(5)+params(4);
h = plot3(px,py,P(3,:),sym);
```

D.4.11 Graph_Remove_Leaves

```
function [Vn,En,Cn]=graph_remove_leaves(V,E,C,vk,op)
% function [Vn,En,Cn]=graph_remove_leaves(V,E,C,vk,op)
%
% Removes leaves of a graph except for vertex ids in the keep list.
%
% INPUTS
%  V = set of graph vertices
%  E = set of graph edges
%  C = set of edge costs
%  vk = set of graph vertices, keep list
%  op = 0, apply leaf removal once
%     = 1, apply leaf removal until all leaves gone except those in vk
%
% OUTPUTS
%  Vn = a new set of graph vertices
%  En = a new set of edges
```

```
%  Cn = a new set of edge costs
%
% Author: Chris Carr
% Date: August 2001
%

% turn graph into an adjacency matrix
Vi=sort(V);
Ei=E;
Ci=C;
maxiter = 500;
[n,A] = graph_compute_adjacency(Vi,Ei,2);
sa = diag(A*(A'))';
nodes = n(find(sa<=1));
leaves_to_remove = setdiff(nodes,vk);
give_up=0;
ctr=0;

while and(not(isempty(leaves_to_remove)),not(give_up)),
    % for each leaf in leaves_to_remove
    for i=1:size(leaves_to_remove,2),
        % get vertex id for this leaf
        %vid = Vi(leaves_to_remove(i))
        vid = leaves_to_remove(i);
       % remove all edges involving vid (there can be only one for a tree, given that it is a leaf)
        ekt1 = find(Ei(2,:)~=vid);
        ekt2 = find(Ei(1,:)~=vid);
        ekt = intersect(ekt1,ekt2);
        vkt = find(Vi~=vid);
        Ei = Ei(1:2,ekt);
        Ci = Ci(ekt);
        Vi = Vi(vkt);
    end
    % compute a new adjacency matrix
    [n,A] = graph_compute_adjacency(Vi,Ei,2);
    sa = diag(A*(A'))';
    nodes = n(find(sa<=1));
    leaves_to_remove = setdiff(nodes,vk);

    if op==0,
        give_up=1;
    end

    ctr = ctr+1;
    if ctr>=maxiter,
        give_up=1;
    end
end

Vn = Vi;
Cn = Ci;
En = Ei;
```

D.5 Traverse Functions

D.5.1 Traverse_Compute_Distance

```
function d = traverse_compute_distance(T,op1,op2)
```

```
% function d = traverse_compute_distance(T,op1,op2)
%
% Computes the distance along a traverse.
%
% INPUTS
%  T = a traverse, of the form,
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%  op1 = distance sum optional parameter
%        0 -> (default value) compute distance along traverse referenced to
%             1st waypoint in traverse (integrated distance)
%        1 -> compute distance between a waypoint and the next waypoint. The
%             last waypoint is assigned a distance of 0.
%  op2 = distance computation parameter
%        0 -> (default value) compute distance between waypoints using x-y-z coordinates
%        1 -> compute distance between waypoints using only xy coordinates
%        2 -> compute distance between waypoints using only z coordinates
%
% OUTPUTS
%  d = a column vector of distances, one per waypoint (row) in T
%
% Author: Chris Carr, June 2001.

if nargin<2,
    op1 = 0;
end
if nargin<3,
    op2 = 0;
end

s = size(T,1);
for i=1:s-1,
    if (op2==0),
        dist(i)=sqrt((T(i+1,2)-T(i,2))^2+(T(i+1,3)-T(i,3))^2+(T(i+1,4)-T(i,4))^2);
    elseif (op2==1),
        dist(i)=sqrt((T(i+1,2)-T(i,2))^2+(T(i+1,3)-T(i,3))^2);
    elseif (op2==2),
        dist(i)=T(i+1,4)-T(i,4);
    end
end
dist(s)=0;

if op1==0,
    d(1)=0;
    for i=2:s,
        d2(i)=sum(dist(1:i-1),2);
    end
elseif op1==1,
    d2 = dist;
else
    error('Invalid option op1 in TRAVERSE_COMPUTE_DISTANCE');
end

d = d2';
if verbose,
```

```
    disp(sprintf('Traverse maximum distance: %8.2f',max(d2)));
end
return;
```

D.5.2 Traverse_Compute_Metabolic_Cost

```
function m = traverse_compute_metabolic_cost(T, mass,g)

% function m = traverse_compute_metabolic_cost(T, mass,g)
%
% Computes the metabolic cost in Joules of a traverse T based on
% the human load carriage model from Santee et al. from
% Aviation, Space and Environmental Medicine, Vol 72, No. 6, June
% 2001.
%
% INPUTS
%  T = a matrix of waypoint data with one row per waypoint (a traverse)
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%  mass = total mass of person making the traverse (including clothing, backpack, etc)
%  g = gravitational acceleration, uses 9.8 m/sec^2 by default.
%
% OUTPUTS
%  m = column vectors of metabolic cost data for each waypoint segment.
%      m(:,1) = column of times
%      m(:,2) = column of power expenditures (J/sec) due to level walking
%      m(:,3) = column of power expenditures (J/sec) due to vertical displacement
%      m(:,4) = column of total estimated power expenditures (J/sec)
%      m(:,5) = column of total estimated energy expenditures (J)
%      m(:,6) = column of velocities (m/sec)
%
% Note1: size(m,1) = size(T,1)
% Note2: The traverse T should be expanded (see traverse_expand) prior to
% using this function.
% Note3: Let s = size(T,1). The last (s-1) entries in m are computed from
% the (s-1) differences in position (horizontal and vertical) as specified
% by the traverse T. The 1st entry of m is set to zero (basically this is
% a padding of the metabolic cost vector). Setting the first element to
% zero is consistant with the timing of each traverse point: metabolic cost
% starts at zero, and after reaching the second traverse point, metabolic
% cost is based upon the altitude differences from the first two traverse
% points.
%
% Author: Chris Carr, June 2001.

if nargin<3,
    g = 9.8;
end

s = size(T,1);
W_Lv = zeros(s,1);
W_V = zeros(s,1);
W_total = zeros(s,1);
E_total = zeros(s,1);
vel = zeros(s,1);
```

```
deltat = zeros(s,1);

for i=2:s,
    % compute height h
    h = T(i,4) - T(i-1,4); % delta z
    % compute delta distance
    dx = sqrt(((T(i,4)-T(i-1,4))^2 + (T(i,3)-T(i-1,3))^2 + (T(i,2)-T(i-1,2))^2));
    % compute delta time
    dt = T(i,7)-T(i-1,7);
    deltat(i-1)=dt;
    % compute velocity
    v = dx/dt;
    vel(i-1)=v;
    % compute level power and energy
    R = 0.661*v + 0.115; % ratio from Santee et al.
    W_L = 3.28*mass + 71.1; % watts for level walking at 1.34 m/sec
    W_Lv(i-1) = W_L*R; % watts for level walking, corrected for actual velocity

    % compute vertical work
    if h < 0,
        % descending... use descending work model
        grad = -h/sqrt((T(i,3)-T(i-1,3))^2+(T(i,2)-T(i-1,2))^2);
        alpha = atan(grad); % slope angle
        alpha = alpha*180/pi; % convert to degrees
        W_V(i-1)=2.4*mass*g*(-h)/dt * 0.3^(alpha/7.65); % watts due to vertical displacement
    elseif h>0,
        % ascending... use ascending work model
        k = 3.5; % corresponds to 28.6% muscle efficiency
        W_V(i-1) = k * mass * g * h / dt; % watts due to vertical displacement
    else
        % h = 0
        W_V(i-1) = 0;
    end

    W_total(i-1) = W_Lv(i-1)+W_V(i-1);

    % multiply by time interval of this segment to get energy instead of power
    E_total(i) = W_total(i-1)*dt; % energy in Joules for this segment of the traverse
end

m = [T(:,7) W_Lv W_V W_total E_total vel deltat];

return;
```

D.5.3 Traverse_Compute_Min_Cost

```
function [T,c,fail]=traverse_compute_min_cost(X,Y,Z,params,xs,xf,v,m,g,op,param,R,C)
% function [T,c,fail]=traverse_compute_min_cost(X,Y,Z,params,xs,xf,v,m,g,op,param,R,C)
%
% Computes the minimum cost traverse (moving at a constant velocity v)
% from the point xs to xf on the surface specified by Z and params.
%
% INPUTS
%  X - x coordinates of the columns of Z
%  Y - y coordinates of the rows of Z
%  Z - a matrix of altitude samples
%  params - a 1x6 vector of height field parameters; of the form
%            [width height xllcorner yllcorner cellsize nodatavalue]
%  xs - initial node position of the form [x y z]'
%  xf - final node position of the form [x y z]'
%  v  - const velocity v of the traverse
```

```
%  m  - mass or other parameter for the cost model
%  g  - gravitational acceleration
%  op - cost option (0=metabolic cost load carrying model)
%                   (1=rover energy cost model)
%  param - parameter for the min cost optimization (number of nodes
%          to use for the min cost graph computation)
%  R  - (optional) reachability map to use in computing this min-cost traverse
%  C  - (optional) cost map to use in computing this min-cost traverse
%
% OUTPUTS
%  T - minimum cost traverse
%  c - cost of minimum cost traverse
%  fail - boolean flag, =1 if no path found between source and target, 0 otherwise
%
% Author: Chris Carr
% Date: August, 2001.
%

fail=0;
draw_on_plot=1;
use_cost_surface = 1;
use_reachability_surface = 1;

if nargin<13,
    use_cost_surface = 0;
    C=[];
end
if nargin<12,
    use_reachability_surface = 0;
    R=[];
end

width = params(1);
height = params(2);
cellsize = params(5);
s = cellsize;

% create 'param' number of points on the surface Z

if verbose,
    disp('Creating trial points in (TRAVERSE_COMPUTE_MIN_COST)');
end

%[n,P] = graph_create_points(Z,1,param-2,0);
% create points by using exploration factor
ef = 2;
a=0; %a = 10*params(5);
dx = abs(xf(1)-xs(1))+a;
dy = abs(xf(2)-xs(2))+a;
%if dx<dy,
%    dx = (dx+dy)/2;
%elseif dy<dx,
%    dy = (dx+dy)/2;
%end
cx = (xf(1)+xs(1))/2;
cy = (xf(2)+xs(2))/2;
minx = max([min(X) cx-(dx/2*ef)]);
maxx = min([max(X) cx+(dx/2*ef)]);
miny = max([min(Y) cy-(dy/2*ef)]);
maxy = min([max(Y) cy+(dy/2*ef)]);
region = [minx miny maxx maxy];
[Xr,Yr,Zr,pr]=surface_extract_region(X,Y,Z,params,region);
```

```
if use_reachability_surface,
    [n,P]=graph_create_points_restricted(Zr,1,param-2,0,R);
else
    [n,P]=graph_create_points(Zr,1,param-2,0);
end
X = surface_convert_pts_to_crd(pr,P);
P = surface_convert_crd_to_pts(params,X);
% end of code to create points using exploration factor

X = surface_convert_pts_to_crd(params,P);
X = [xs X xf];
P = surface_extract_altitudes(Z,params,X(1,:),X(2,:));
N = 1:size(X,2); % node ids

% compute point density metric (helps reduce number of edges as number of points grows large)
metric = 1/sqrt(param)*2*max([pr(1) pr(2)])*cellsize;

% connect points if the distance between them is less than metric, and compute edge cost
if verbose,
    disp('Computing traverse costs between trial points (TRAVERSE_COMPUTE_MIN_COST)');
end
ctr = 0;
E=[];
for i=1:param,
    for j=i+1:param,
        % check distance between two points
        dist = sqrt((X(1,i)-X(1,j))^2+(X(2,i)-X(2,j))^2);
        % compute reachability along edge, if necessary
        reachable=0;
        if use_reachability_surface,

            % compute reachability along edge
            numpts = j-i+1;
            t = linspace(0,1,numpts);
            Px = round(P(1,i)+(P(1,j)-P(1,i))*t);
            Py = round(P(2,i)+(P(2,j)-P(2,i))*t);
            %Px = P(1,i):1:P(1,j);
            %Py = P(2,i):1:P(2,j);
            for kk=1:size(Px,2),
                Ryx(kk) = R(Py(kk),Px(kk));
            end
            reachable = and(isempty(find(Ryx==0)),isempty(find(isnan(Ryx)==1)));
        end
        % conditions for creating edge include:
        % (1) distance < metric
        % (2) path between two points must be reachable (R(j,i)=1 for all (j,i) along the edge
        a = (dist<metric);
        b = or(not(use_reachability_surface),and(use_reachability_surface,reachable));
        if and(a,b),
            % create an edge
            ctr=ctr+1;
            E(1:2,ctr)=[i j]';
            % build a traverse for this edge
            segd = sqrt((X(1,i)-X(1,j))^2 + (X(2,i)-X(2,j))^2 + (X(3,i)-X(3,j))^2);
            t = segd/v;
            TEi = [[1 X(1,i) X(2,i) X(3,i) 0 1 0 0]; [2 X(1,j) X(2,j) X(3,j) 0 2 t 0]];
            % ... and just interpolate between them.
            TEi = traverse_interpolate(TEi,s,0); % spatial sampling
            TEi = traverse_project(Z,params,TEi); % project onto surface
            % compute the cost of the traverse for this edge
            if use_cost_surface,
                % compute cost from cost surface
```

```
                Px = P(1,i):1:P(1,j);
                Py = P(2,i):1:P(2,j);
                Cyx=0;
                for kk=1:size(Px,2),
                    Cyx = Cyx+C(Py(kk),Px(kk));
                end
                C(ctr)=Cyx;
            else
                if op==0,
                    mEi = traverse_compute_metabolic_cost(TEi,m,g);
                    % extract cost in J for this edge
                    C(ctr)=sum(mEi(:,5),1);
                elseif op==1,
                    mEi = traverse_compute_rover_cost(TEi,m,g);
                    % extract cost in J for this edge
                    C(ctr)=sum(mEi(:,5),1);
                else
                    error('Invalid option op in TRAVERSE_COMPUTE_MIN_COST');
                end
            end
        end
    end
end

%  now find minimum cost path between xs and xf given G={N,E,P,C}
node=1; % source node is node 1
if verbose,
    disp('Computing minimum cost path tree (TRAVERSE_COMPUTE_MIN_COST)');
end

% debug code
if draw_on_plot==1,
    set(gca,'nextplot','add');
    graph_plot_points(1,P,params);
    graph_plot_edges(1,N,E,P,params,'-k');
end

[Vm,Em,D,Cm]=graph_compute_dsp(N,E,C,node);

% check and see if source and target nodes are in the connected set
if isempty(find(Vm==param)),
    % no path to target node found...
    fail = 1;
    % instead, find the nearest point to the target point and return a path to that point
    for kk=1:size(Vm,2),
        % compute distance to target for each point in Vm
        nid = Vm(kk);
        dtt(kk) = sqrt((X(1,nid)-X(1,param))^2+(X(2,nid)-X(2,param))^2+(P(3,nid)-P(3,param))^2);
    end
    [dt idx_dt]=min(dtt);
    vk = [1 Vm(idx_dt)]; % keep source and node closest to target node
else
    % remove the leaves except for source and target nodes
    vk = [1 size(X,2)]; % keep source and target nodes
end

if verbose,
    disp('Extracting traverse path from min cost path tree (TRAVERSE_COMPUTE_MIN_COST)');
end
[Vn,En,Cn]=graph_remove_leaves(Vm,Em,Cm,vk,1);

% debug code
```

```
if draw_on_plot==1,
    Pm = P(1:3,Vm);
    set(gca,'nextplot','add');
    graph_plot_points(1,Pm,params);
    graph_plot_edges(1,Vm,Em,Pm,params,'-w');
    Pn = P(1:3,Vn);
    graph_plot_points(1,Pn,params);
    graph_plot_edges(1,Vn,En,Pn,params,'-r',[1 1 1],2);
end

% turn the min cost path into a traverse
vn = 1; % initialize current node to starting node
to = 0;
T = [1 X(1,1) X(2,1) X(3,1) 0 1 0 0];
% for each edge in the path
vid = [];
for i=1:size(En,2),
    % find the next node
    eid1 = find(En(1,:)==vn);
    eid2 = find(En(2,:)==vn);
   eid = setdiff(union(eid1,eid2),vid); % edge containing nodes with node id vn, excluding already
visited edges
    vo = vn; % set old vertex
    tv = [En(2,eid) En(1,eid)];
    vn = setdiff(tv,vo); % vn = vertex id for next node - pick the node id not already visited
    vid = [vid eid]; % update the list of visited edges
    % compute distance to new node
    dist = sqrt((X(1,vn)-X(1,vo)).^2+(X(2,vn)-X(2,vo)).^2+(X(3,vn)-X(3,vo)).^2);
    % compute time to new node at constant velocity v
    t = dist/v+to;
    to = t;
    T = [T; [i+1 X(1,vn) X(2,vn) X(3,vn) 0 i+1 t 0]];
end

% compute cost of the final min cost traverse
if op==0,
    m = traverse_compute_metabolic_cost(T,m,g);
    c = sum(m(:,5),1);
elseif op==1,
    m = traverse_compute_rover_cost(T,m,g);
    c = sum(m(:,5),1);
else
    c=0;
end
return;
```

D.5.4 Traverse_Compute_Rover_Cost

```
function m = traverse_compute_rover_cost(T, mass,g)

% function m = traverse_compute_rover_cost(T, mass,g)
%
% Computes the energy cost in Joules of a traverse T based on
% a simple made-up rover energy expenditure model.
%
% NOTE: LRV mass = 210 kg, max payload = 495 kg
% Vehicle Mileage ~ 35-56 W-hr/km
% Mass Mileage ~ 0.050 - 0.080 W-hr/km/kg
%
%
```

```
% INPUTS
%  T = a matrix of waypoint data with one row per waypoint (a traverse)
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%  mass = total mass of rover including empty weight and payload
%  g = gravitational acceleration, uses 9.8 m/sec^2 by default.
%
% OUTPUTS
%  m = column vectors of metabolic cost data for each waypoint segment.
%      m(:,1) = column of times
%      m(:,2) = column of power expenditures (J/sec) due to level movement
%      m(:,3) = column of power expenditures (J/sec) due to vertical displacement
%      m(:,4) = column of total estimated power expenditures (J/sec)
%      m(:,5) = column of total estimated energy expenditures (J)
%      m(:,6) = column of velocities (m/sec)
%
% Note1: size(m,1) = size(T,1)
% Note2: The traverse T should be expanded (see traverse_expand) prior to
% using this function.
% Note3: Let s = size(T,1). The last (s-1) entries in m are computed from
% the (s-1) differences in position (horizontal and vertical) as specified
% by the traverse T. The 1st entry of m is set to zero (basically this is
% a padding of the metabolic cost vector). Setting the first element to
% zero is consistant with the timing of each traverse point: metabolic cost
% starts at zero, and after reaching the second traverse point, metabolic
% cost is based upon the altitude differences from the first two traverse
% points.
%
% Author: Chris Carr, August 2001.

if nargin<3,
    g = 9.8;
end

lg = 1.62;
s = size(T,1);
W_Lv = zeros(s,1);
W_V = zeros(s,1);
W_total = zeros(s,1);
E_total = zeros(s,1);
vel = zeros(s,1);
deltat = zeros(s,1);

for i=2:s,
    % compute height h
    h = T(i,4) - T(i-1,4); % delta z
    % compute delta distance
    dx = sqrt(((T(i,4)-T(i-1,4))^2 + (T(i,3)-T(i-1,3))^2 + (T(i,2)-T(i-1,2))^2));
    % compute delta time
    dt = T(i,7)-T(i-1,7);
    deltat(i-1)=dt;
    % compute velocity
    if dt==0,
        v=0;
    else
        v = dx/dt;
```

```
    end
    vel(i-1)=v;
    % compute level power and energy
    k = 0.060; % W-hr/km/kg
    kp = k*3.6; % W-s/m/kg
    W_Lv(i-1) = kp * v * mass+5;

    % compute vertical work
    if h < 0,
        % descending... use descending work model
        grad = -h/sqrt((T(i,3)-T(i-1,3))^2+(T(i,2)-T(i-1,2))^2);
        alpha = atan(grad); % slope angle
        alpha = abs(alpha*180/pi); % convert to degrees
        k = 0.0073; % W-hr/km/kg/deg
        kp = k*3.6; % W-s/m/kg/deg
        ef = 0.3; % efficiency factor of energy recovery
        W_V(i-1) = - ef * kp * mass * alpha * (g/lg) * v; % watts due to vertical displacement
    elseif h>0,
        % ascending... use ascending work model: Lunar Sourcebook: 1 degree = 0.0073 W-hr/km/kg
        grad = -h/sqrt((T(i,3)-T(i-1,3))^2+(T(i,2)-T(i-1,2))^2);
        alpha = atan(grad); % slope angle
        alpha = abs(alpha*180/pi); % convert to degrees
        k = 0.0073; % W-hr/km/kg/deg
        kp = k*3.6; % W-s/m/kg/deg
        W_V(i-1) = kp * mass * alpha * (g/lg) * v; % watts due to vertical displacement
    else
        % h = 0
        W_V(i-1) = 0;
    end

    W_total(i-1) = W_Lv(i-1)+W_V(i-1);

    % multiply by time interval of this segment to get energy instead of power
    E_total(i) = W_total(i-1)*dt; % energy in Joules for this segment of the traverse
end

m = [T(:,7) W_Lv W_V W_total E_total vel deltat];

return;
```

D.5.5 Traverse_Expand

```
function Tn = traverse_expand(T)
% function Tn = traverse_expand(T)
%
% Expands a traverse by creating additional traverse segments for
% waypoints in the traverse T that have non-zero stay times. The
% new traverse, Tn, has zero duration stay times for all waypoints.
%
% INPUTS
%  T = a traverse, of the form,
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%
% OUTPUTS
```

```
%  Tn = a new traverse
%
% Author: Chris Carr, June 2001.

s = size(T,1);
ctr = 1;

for i=1:s,
    % see if current waypoint has a non-zero stay time
    if (T(i,8)~=0)
        % non-zero stay time: create a new waypoint (same location)
        % with new time of arrival at the waypoint
        Tc = T(i,:);
        Tc(1,7) = T(i,7)+T(i,8);
        Tc(1,8) = 0;

        % copy original waypoint (with zero-stay time) into Tn
        Tn(ctr,:) = T(i,:);
        Tn(ctr,8)=0;

        % copy new waypoint (with zero-stay time) into Tn
        Tn(ctr+1,:) = Tc;

        ctr = ctr+2;
    else
        Tn(ctr,:) = T(i,:);
        ctr = ctr+1;
    end
end

return;
```

D.5.6 Traverse_From_Points

```
function T = traverse_from_points(x,y,z,op,param)
%function T = traverse_from_points(x,y,z,op,param)
%
% Converts a series of (x,y,z) points into a traverse T.
%
% INPUTS
%  x = x-coordinates of points in the traverse
%  y = y-coordinates of points in the traverse
%  z = z-coordinates of points in the traverse
%  op = 0, default, parameter defines velocity for parameterization of path
%  param = parameter for creating the traverse (when op=0, velocity to use for
%          construction of the traverse for computing time between waypoints)
%
% OUTPUTS
%  T = a matrix of waypoint data with one row per waypoint
%  T(:,1) = column of waypoint ids
%  T(:,2) = column of waypoint x-coordinates
%  T(:,3) = column of waypoint y-coordinates
%  T(:,4) = column of waypoint z-coordinates
%  T(:,5) = column of waypoint day identifiers (field day)
%  T(:,6) = column of waypoint ids for this field day
%  T(:,7) = column of waypoint times of arrival (sec since midnight)
%  T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%
% Author: Chris Carr, June 2001.

s = size(x,2); % number of points in traverse
```

```
T = zeros(s,8);

if nargin<5,
    param = 1; % 1 meter/sec
end
if nargin<4,
    op =0; % velocity defined traverse
end

if (op==0),

    for i=1:s,
        T(i,1) = i; % default waypoint id is count of waypoints so far
        T(i,2) = x(i);
        T(i,3) = y(i);
        T(i,4) = z(i);
        T(i,5) = 0;
        T(i,6) = i; % default waypoint day id is count of waypoints so far
        if i==1,
            T(i,7) = 0;
        else
            d = sqrt((x(i)-x(i-1))^2+(y(i)-y(i-1))^2+(z(i)-z(i-1))^2);
            t = d/param;
            T(i,7) = t;
        end
        T(i,8) = 0;
    end
else
    error('Invalid parameter value OP');
end

return;
```

D.5.7 Traverse_Import

```
function T = traverse_import(file)
% function T = traverse_import(file)
%
% Imports a traverse data file and returns the result in T.
%
% INPUTS
%  file = filename for traverse data to be imported
%
% OUTPUTS
%  T = a matrix of waypoint data with one row per waypoint
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)

T = csvread(file);
```

D.5.8 Traverse_Interpolate

```
function Tn = traverse_interpolate(T,s,op)
% function Tn = traverse_interpolate(T,s,op)
```

```
%
% Interpolates a traverse at sampling intervals of s by
% interpolating additional waypoints between the existing
% waypoints of the traverse T.
%
% INPUTS
%  T = a matrix of waypoint data with one row per waypoint
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%  s = sampling interval for the new traverse Tn
%  op = sampling option (op=0, default -> spatial sampling)
%                       (op=1, -> temporal sampling)
%
% OUTPUTS
%  Tn = a matrix of waypoint data with one row per waypoint, the new traverse
%
% Author: Chris Carr, June 2001.

if nargin<3,
    op = 0;
end

siz = size(T,1)-1; % number of waypoint segments in the traverse T

Tn = T(1,:);

% for each waypoint segment in T
for i=1:siz,
    % interpolate the segment
    Ts = traverse_interpolate_segment(T(i:i+1,:),s,op);
    z = size(Tn,1);
    Tn(z:z+size(Ts,1)-1,:) = Ts;
end
```

D.5.9 Traverse_Interpolate_Segment

```
function Tn = traverse_interpolate_segment(T,s,op)
% function Tn = traverse_interpolate_segment(T,s,op)
%
% Interpolates a segment of a traverse with sampling interval s.
% For spatial sampling of a given surface, s should be at least
% equal to the grid spacing of the surface in order to guarantee
% an adequate number of traverse samples for any traverse segment T.
% For temporal sampling, the time interval should be chosen so
% that sample points are spaced at most a distance apart equal to
% the grid size of the given surface.
%
% INPUTS
%  T = a traverse segment, of size 2x8.
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
```

```
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%  s = the sampling interval for the traverse segment
%  op = (0=default, spatial sampling, 1=temporal sampling)
%
% OUTPUTS
%  Tn     = the new traverse, a projection of T onto the surface specified by Z and params
%
% NOTE: T(:,8) waypoint stay times should be zero before using this function. For information
% on time resampling of traverses to zero waypoint stay times (by creating additional
% traverse segments from a given waypoint to the same waypoint) see the function
% traverse_expand.m.
%
% Author: Chris Carr, June 2001.

% size of T should be 2x8 - by default only read first two rows
d = sqrt((T(1,2)-T(2,2))^2+(T(1,3)-T(2,3))^2+(T(1,4)-T(2,4))^2);
dt = T(2,7)-T(1,7);
if (op==0),
    n = ceil(d/s);
else
    n = ceil(dt/s);
end
x = linspace(T(1,2),T(2,2),n);
y = linspace(T(1,3),T(2,3),n);
z = linspace(T(1,4),T(2,4),n);
t = linspace(T(1,7),T(2,7),n);

Tn = zeros(n,8);
Tn(:,1) = [ones(n-1,1)*T(1,1);T(2,1)];
Tn(:,2) = x';
Tn(:,3) = y';
Tn(:,4) = z';
Tn(:,5) = [ones(n-1,1)*T(1,5);T(2,5)];
Tn(:,6) = linspace(T(1,6),T(2,6),n)';
Tn(:,7) = t';
Tn(:,8) = 0;

return;
```

D.5.10 Traverse_Plot_On_Surface

```
function traverse_plot_on_surface(h,Z,params,T,offset,sp,clr,ns)
% function traverse_plot_on_surface(h,Z,params,T,offset,sp,clr,ns)
%
% Plots a traverse on a surface.
%
% INPUTS
%  h = handle of a valid figure of a surface plot
%  Z = surface height field matrix
%  params = surface parameters for the height field Z, of the form
%           [width height xllcorner yllcorner cellsize nodatavalue]
%  T = traverse matrix, of the form:
%      T(:,1) = column of waypoint ids
%      T(:,2) = column of waypoint x-coordinates
%      T(:,3) = column of waypoint y-coordinates
%      T(:,4) = column of waypoint z-coordinates
%      T(:,5) = column of waypoint day identifiers (field day)
%      T(:,6) = column of waypoint ids for this field day
%      T(:,7) = column of waypoint times of arrival (sec since midnight)
%      T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
```

```
%  offset = offset in surface units for plotting of z coordinate of waypoints
%  sp = optional parameter; specifies a subplot, of the form [a b c]
%  clr = color characters; of the form 'abc'
%  ns = no symbols option: (0=show symbols [default], 1=show no symbols,
%                           only line); optional parameter

if nargin<5,
    offset = 2; % 2 meter default offset
end
if nargin<6,
    sp = [1 1 1];
end
if nargin<7,
    clr = 'brg';
end
if nargin<8,
    ns = 0;
end

% process the traverse
s = params(5);
Tp = traverse_project(Z,params,T);
Te = traverse_expand(Tp);
Te = traverse_interpolate(Te,s);
Te = traverse_project(Z,params,Te);

figure(h);
subplot(sp(1),sp(2),sp(3));
% plot traverse
np = get(gca,'nextplot');
set(gca,'nextplot','add');
%plot3(Tp(:,2)',Tp(:,3)',Tp(:,4)',strcat(clr(1),'o')); % plot just the original (projected)
traverse waypoints
if (ns==0),
    plot3(Tp(:,2)',Tp(:,3)',Tp(:,4)',strcat(clr(1),'.')); % plot just the original (projected)
traverse waypoints
end
plot3(Te(:,2)',Te(:,3)',Te(:,4)'+offset,strcat(clr(1),'-')); % plot interpolated waypoints
if (ns==0),
    plot3(Tp(1,2),Tp(1,3),Tp(1,4),strcat(clr(2),'s')); % plot red waypoint square at starting way-
point
    plot3(Tp(2,2),Tp(2,3),Tp(2,4),strcat(clr(3),'o')); % plot green waypoint circle at 2nd way-
points
end
set(gca,'nextplot',np);
```

D.5.11 Traverse_Project

```
function Tn = traverse_project(Z,params,T)
% function Tn = traverse_project(Z,params,T)
%
% Projects the traverse T onto the surface represented by Z and params.
%
% INPUTS
%  Z      = a matrix of altitude samples that defines the projection surface
%  params = 1x6 vector of surface parameters of the form
%                [width height xllcorner yllcorner cellsize nodatavalue]
%  T      = a matrix of waypoint data with one row per waypoint that defines a traverse
%           T(:,1) = column of waypoint ids
%           T(:,2) = column of waypoint x-coordinates
%           T(:,3) = column of waypoint y-coordinates
```

```
%           T(:,4) = column of waypoint z-coordinates
%           T(:,5) = column of waypoint day identifiers (field day)
%           T(:,6) = column of waypoint ids for this field day
%           T(:,7) = column of waypoint times of arrival (sec since midnight)
%           T(:,8) = column of waypoint time of stay at waypoint, if known (sec)
%
% OUTPUTS
%  Tn     = the new traverse, a projection of T onto the surface specified by Z and params
%
% This function may be most useful for displaying traverses on a surface plot. Where
% traverse waypoints are not significantly further apart than the grid spacing of the
% surface, projecting the traverse onto the surface will reduce the resolution of the
% traverse data.
%
% Author: Chris Carr, June 2001.

% for each waypoint in the traverse, look up the altitude of the waypoint in the digital
% elevation model specified by Z and params, and replace the z coordinate of the traverse
% with the altitude of the digital elevation model.

s=size(T,1);

x = T(:,2);
y = T(:,3);

% get x-y-z points on surface from traverse x-y points
P = surface_extract_altitudes(Z,params,x',y');

% replace z coordinates of traverse points with z coordinates of surface model
Tn = T;
Tn(:,4) = P(3,:)';
```

D.6 Power Spectral Density Functions

D.6.1 PSD_Compute_Avg_Cross

```
function [psd_r, q] = psd_compute_avg_cross(grid, cellsize)

% function [psd_r, q] = psd_compute_avg_cross(grid, cellsize)
%
% Extracts average from central cross in a grid (to estimate radial
% PSD profile from 2d PSD).
%
% INPUTS
%  grid = m x m matrix of samples of the power spectral density
%  cellsize = spatial separation of the samples of the height
%             field for which grid is the power spectral density,
%             corresponds to 1/max spatial frequency
%
% OUTPUTS
%  psd_r = radial profile of the power spectral density
%  q     = spatial frequency vector
%
% Author: Chris Carr, May 2001.

%maxsteps = floor(sqrt((size(grid,1)/2-1)^2*2));
maxsteps = floor(size(grid,1)/2);
```

```
w = size(grid,2);
h = size(grid,1);
tmp = grid;

st = [1, 1];
ed = [1,h/2];
[q psd_r1] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [1, 1];
ed = [w/2,1];
[q psd_r2] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w 1];
ed = [w/2+1,1];
[q psd_r3] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w 1];
ed = [w,h/2];
[q psd_r4] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

psd_r = (psd_r1+psd_r2+psd_r3+psd_r4)/4;

return
```

D.6.2 PSD_Compute_Avg_CrossDiag

```
function [psd_r, q] = psd_compute_avg_crossdiag(grid, cellsize)

% function [psd_r, q] = psd_compute_avg_crossdiag(grid, cellsize)
%
% Extracts average from diagonals and central cross in a
% grid (to estimate radial PSD profile from 2d PSD).
%
% INPUTS
%  grid = m x m matrix of samples of the power spectral density
%  cellsize = spatial separation of the samples of the height
%             field for which grid is the power spectral density,
%             corresponds to 1/max spatial frequency
%
% OUTPUTS
%  psd_r = radial profile of the power spectral density
%  q     = spatial frequency vector
%
% Author: Chris Carr, May 2001.

%maxsteps = floor(sqrt((size(grid,1)/2-1)^2*2));
maxsteps = floor(size(grid,1)/2);
w = size(grid,2);
h = size(grid,1);
tmp = grid;

st = [1, 1];
ed = [1,h/2];
[q psd_r1] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [1, 1];
ed = [w/2,1];
[q psd_r2] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w 1];
```

```
ed = [w/2+1,1];
[q psd_r3] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w 1];
ed = [w,h/2];
[q psd_r4] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [1 1];
ed = [w/2,h/2];
[q psd_r5] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [1 h];
ed = [w/2,h/2+1];
[q psd_r6] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w 1];
ed = [w/2+1,h/2];
[q psd_r7] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w h];
ed = [w/2+1,h/2+1];
[q psd_r8] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

psd_r = (psd_r1+psd_r2+psd_r3+psd_r4+psd_r5+psd_r6+psd_r7+psd_r8)/8;

return
```

D.6.3 PSD_Compute_Avg_Diag

```
function [psd_r, q] = psd_compute_avg_diag(grid, cellsize)

% function [psd_r, q] = psd_compute_avg_diag(grid, cellsize)
%
% Extracts average from diagonals in a grid (to estimate radial
% PSD profile from 2d PSD).
%
% INPUTS
%  grid = m x m matrix of samples of the power spectral density
%  cellsize = spatial separation of the samples of the height
%             field for which grid is the power spectral density,
%             corresponds to 1/max spatial frequency
%
% OUTPUTS
%  psd_r = radial profile of the power spectral density
%  q     = spatial frequency vector
%
% Author: Chris Carr, May 2001.

%maxsteps = floor(sqrt((size(grid,1)/2-1)^2*2));
maxsteps = floor(size(grid,1)/2);
w = size(grid,2);
h = size(grid,1);
tmp = grid;

st = [1 1];
ed = [w/2,h/2];
[q psd_r1] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [1 h];
ed = [w/2,h/2+1];
```

```
[q psd_r2] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w 1];
ed = [w/2+1,h/2];
[q psd_r3] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

st = [w h];
ed = [w/2+1,h/2+1];
[q psd_r4] = psd_extract_line(tmp,cellsize,st,ed,maxsteps);

psd_r = (psd_r1+psd_r2+psd_r3+psd_r4)/4;

return
```

D.6.4 PSD_Compute_Avg_Radial

```
function [psd_r,q] = psd_compute_avg_radial(grid, cellsize)

% function [psd_r,q] = psd_compute_avg_radial(grid, cellsize)
%
% Extracts radial profile (assumes grid is PSD grid) from PSD
% by averaging over entire grid.
%
% INPUTS
%  grid = m x m matrix of samples of the power spectral density
%  cellsize = spatial separation of the samples of the height
%             field for which grid is the power spectral density,
%             corresponds to 1/max spatial frequency
%
% OUTPUTS
%  psd_r = radial profile of the power spectral density
%  q     = spatial frequency vector
%
% Author: Chris Carr, May 2001.

%maxsteps = floor(size(grid,1)/2-1);
maxsteps = floor(size(grid,1)/2);
w = size(grid,2);
h = size(grid,1);
tmp = grid;

psd_r = zeros(1,maxsteps);
n = zeros(1,maxsteps);

% upper left quadrant
qd = [1 1 w/2 h/2];
step = [(qd(3)-qd(1))/abs(qd(3)-qd(1)) (qd(4)-qd(2))/abs(qd(4)-qd(2))];
ref = [1 1];
for i=qd(1):step(1):qd(3),
    for j=qd(2):step(2):qd(4),
        r = floor(sqrt((ref(1)-i)^2+(ref(2)-j)^2));
        if r<maxsteps,
            % r is valid index
            psd_r(r+1) = psd_r(r+1)+grid(j,i);
            n(r+1) = n(r+1)+1;
        end
    end
end

% upper right quadrant
```

```
qd = [w/2+1 1 w h/2];
step = [(qd(3)-qd(1))/abs(qd(3)-qd(1)) (qd(4)-qd(2))/abs(qd(4)-qd(2))];
ref = [w 1];
for i=qd(1):step(1):qd(3),
    for j=qd(2):step(2):qd(4),
        r = floor(sqrt((ref(1)-i)^2+(ref(2)-j)^2));
        if r<maxsteps,
            % r is valid index
            psd_r(r+1) = psd_r(r+1)+grid(j,i);
            n(r+1) = n(r+1)+1;
        end
    end
end

% lower left quadrant
qd = [1 h/2+1 w/2 h];
step = [(qd(3)-qd(1))/abs(qd(3)-qd(1)) (qd(4)-qd(2))/abs(qd(4)-qd(2))];
ref = [1 h];
for i=qd(1):step(1):qd(3),
    for j=qd(2):step(2):qd(4),
        r = floor(sqrt((ref(1)-i)^2+(ref(2)-j)^2));
        if r<maxsteps,
            % r is valid index
            psd_r(r+1) = psd_r(r+1)+grid(j,i);
            n(r+1) = n(r+1)+1;
        end
    end
end

% lower right quadrant
qd = [w/2+1 h/2+1 w h];
step = [(qd(3)-qd(1))/abs(qd(3)-qd(1)) (qd(4)-qd(2))/abs(qd(4)-qd(2))];
ref = [w h];
for i=qd(1):step(1):qd(3),
    for j=qd(2):step(2):qd(4),
        r = floor(sqrt((ref(1)-i)^2+(ref(2)-j)^2));
        if r<maxsteps,
            % r is valid index
            psd_r(r+1) = psd_r(r+1)+grid(j,i);
            n(r+1) = n(r+1)+1;
        end
    end
end

psd_r = psd_r./n;
u = linspace(1,maxsteps,maxsteps);
q = 1/(maxsteps*cellsize)*u;
```

D.6.5 PSD_Compute_Radial

```
function [psdn,fn,wn,beta,a] = psd_compute_radial(psd,f,w,method)

% function [psdn,fn,beta,a] = psd_compute_radial(psd,f,w,method)
%
% Estimates the radial power spectral density profile for
% a 2D power spectral density using one of several methods
% specified by the 'method' parameter.
%
% INPUTS
%  psd = power spectral density
```

```
%  f = spatial frequency vector
%  w = wavelength vector
%  method = 1, diagonal method
%           2, cross method
%           3, diagonal and cross method
%           4, average around the circle
%
% OUTPUTS
%  psdn = radial power spectral density profile
%  fn   = new spatial frequncy vector
%  wn   = mew wavelength vector
%  beta = power law exponent for least-squares fit power law
%  a    = power factor
%
%
% Author: Chris Carr, May 2001.

% compute cellsize from wavelength vector
cellsize = min(w);

% extract radial trace of PSD
if method==1,
    [psd_r r] = psd_compute_avg_diag(psd,cellsize);
elseif method==2,
    [psd_r r] = psd_compute_avg_cross(psd,cellsize);
elseif method==3,
    [psd_r r] = psd_compute_avg_crossdiag(psd,cellsize);
elseif method==4,
    [psd_r r] = psd_compute_avg_radial(psd,cellsize);
else
    error('Invalid method in PSD_COMPUTE_RADIAL');
end

% compute trend line
% ignore 1st point
b = log10(psd_r(2:size(psd_r,2)));
t = log10(f(2:size(psd_r,2)));
t_f=log10(f);

% Implement least squares
b = [sum(b,2);sum(t.*b,2)];
A = [size(t,2) sum(t,2); sum(t,2) sum(t.^2,2)];
k = real(inv(A)*b);
yf = k(1)+ k(2)*t_f;
beta = -k(2);

psdn = psd_r;
fn = fftshift(f);
wn = fftshift(w);
fn = fn(size(fn,2)/2+1:size(fn,2));
wn = wn(size(wn,2)/2+1:size(wn,2));
a = 10^k(1);
```

D.6.6 PSD_Extract_Line

```
function [q,z] = psd_extract_line(Z, spacing, mystart, myend, maxsteps);

% function [q,z] = psd_extract_line(Z, spacing, mystart, myend, maxsteps);
%
% Extracts the values in grid given starting and ending positions in grid
```

```
%
% INPUTS
%  Z        = a 2D grid of values
%  spacing  = spacing of values in the time domain (of which Z is the 2D power spectral density)
%  mystart  = 1x2 matrix of starting index in height field, of form [x1 y1]
%  myend    = 1x2 matrix of ending index in height field, of form [x2 y2]
%  maxsteps = maximum number of elements of extracted line
%
% OUTPUTS
%  q        = frequency vector assuming Z is a power spectral density
%  z        = values along the extracted line
%
% Author: Chris Carr, May 2001.

% compute function across grid
% compute distance to get number of steps

if (verbose),
    disp('Extracting line from grid...');
end

d = sqrt(((myend(1)-mystart(1))^2 + (myend(2)-mystart(2))^2))+1;
steps = floor(d);
if steps>maxsteps,
    steps=maxsteps;
end
t = linspace(0,1,steps);
if (verbose),
    disp(sprintf('Using %d steps.',steps));
end
xp = floor(mystart(1)+t*(myend(1)-mystart(1)));
yp = floor(mystart(2)+t*(myend(2)-mystart(2)));

for i=1:size(xp,2),
    z(i) = Z(yp(i),xp(i));
end

%q = t*steps; % freq vector
u = linspace(1,steps,steps);
q = 1/(steps*spacing)*u;
return
```

D.6.7 PSD_Plot

```
function psd_plot(hFig,psd,f,w,sp)

% function psd_plot(hFig,psd,f,w,sp)
%
% Plots psd as a power spectral density function.
%
% INPUTS
%  hFig = a handle to an existing figure
%  psd  = 2d power spectral density
%  f    = frequency vector for 2d power spectral density
%  w    = wavelength vector for 2d power spectral density
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig
%
% Author: Chris Carr, May 2001.
```

```
figure(hFig);
if nargin==5,
    subplot(sp(1),sp(2),sp(3));
end

ftmp = [fliplr(f(1:size(f,2)/2)),f(1:size(f,2)/2)];
wtmp = [fliplr(w(1:size(w,2)/2)),w(1:size(w,2)/2)];

a = sprintf('-1/%d',min(abs(wtmp)));
b = sprintf('1/%d',max(wtmp));
c = sprintf('1/%d',min(abs(wtmp)));
s = '          ';

a = [a s(1:10-size(a,2))];
b = [b s(1:10-size(b,2))];
c = [c s(1:10-size(c,2))];

d = -min(abs(wtmp));
e = max(wtmp);
f = min(abs(wtmp));

surf(fftshift(log10(psd)));
view(2);
shading flat;
set(gca,'PlotBoxAspectRatio',[1 1 1]);
set(gca,'xlim',[1 size(psd,2)]);
set(gca,'ylim',[1 size(psd,1)]);
set(gca,'xtick',[1 size(psd,2)/2 size(psd,2)]);
set(gca,'ytick',[1 size(psd,1)/2 size(psd,2)]);
set(gca,'xticklabel',[a;b;c]);
set(gca,'yticklabel',[d e f]);
xlabel('Spatial Frequency (cycles/m)');
ylabel('Wavelength (m)');
```

D.6.8 PSD_Plot_Minimal

```
function psd_plot_minimal(hFig,psd,f,w,sp)

% function psd_plot_minimal(hFig,psd,f,w,sp)
%
% Plots psd as a power spectral density function.
%
% INPUTS
%  hFig = a handle to an existing figure
%  psd  = 2d power spectral density
%  f    = frequency vector for 2d power spectral density
%  w    = wavelength vector for 2d power spectral density
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig
%
% Author: Chris Carr, May 2001.

figure(hFig);
if nargin==5,
    subplot(sp(1),sp(2),sp(3));
end

surf(fftshift(f),fftshift(f),fftshift(log10(psd)));
view(2);
shading flat;
```

```
set(gca,'xlim',[min(f) max(f)]);
set(gca,'ylim',[min(f) max(f)]);
set(gca,'PlotBoxAspectRatio',[1 1 1]);
set(gca,'xticklabel',[]);
set(gca,'xtick',[]);
set(gca,'yticklabel',[]);
set(gca,'ytick',[]);
set(gca,'LineWidth',2);
set(gca,'box','on');
```

D.6.9 PSD_Plot_Radial

```
function psd_plot_radial(hFig,psd_r,f,w,sym,sp)

% function psd_plot_radial(hFig,psd_r,f,w,sym,sp)
%
% Plots psd as a power spectral density function.
%
% INPUTS
%  hFig = a handle to an existing figure
%  psd  = radial power spectral density
%  f    = frequency vector for 2d power spectral density
%  w    = wavelength vector for 2d power spectral density
%  sym  = symbol to use for plotting (optional argument)
%  sp   = an optional argument of form [a b c] that specifies a subplot triple
%         for the figure specified by hFig
%
% Author: Chris Carr, May 2001.

figure(hFig);
if nargin<5,
    sym = '.-';
end
if nargin==6,
    subplot(sp(1),sp(2),sp(3));
end

ftmp = f;
wtmp = w;

loglog(f,psd_r,sym);
set(gca,'FontName','Courier New');
set(gca,'FontSize',10);
xlabel('Spatial Frequency (cycles/m)');
ylabel('Power Spectral Density (m^2)');
```

D.6.10 PSD_Plot_Radial_Minimal

```
function psd_plot_radial_minimal(hFig,psd_r,f,w,sym,sp)

% function psd_plot_radial_minimal(hFig,psd_r,f,w,sym,sp)
%
% Plots psd as a power spectral density function.
%
% INPUTS
%  hFig = a handle to an existing figure
%  psd  = radial power spectral density
%  f    = frequency vector for 2d power spectral density
```

```
%  w     = wavelength vector for 2d power spectral density
%  sym   = symbol to use for plotting (optional argument)
%  sp    = an optional argument of form [a b c] that specifies a subplot triple
%          for the figure specified by hFig
%
% Author: Chris Carr, May 2001.

figure(hFig);
if nargin<5,
    sym = '.-';
end
if nargin==6,
    subplot(sp(1),sp(2),sp(3));
end

ftmp = f;
wtmp = w;

loglog(f,psd_r,sym);
set(gca,'PlotBoxAspectRatio',[1 1 1]);
set(gca,'xticklabel',[]);
set(gca,'xtick',[]);
set(gca,'yticklabel',[]);
set(gca,'ytick',[]);
set(gca,'LineWidth',2);
set(gca,'box','on');
```

D.6.11 Synth_Spect

```
function [sspec,fld]=synth_spect(siz,law)
% function [sspec,fld]=synth_spect(siz,law)
%
% Synthesizes a spectrum for a height field based on power law that specifies
% the radial profile of the spectrum; this symmetry is required to ensure that
% the height field will be real.
%
% INPUTS:
%  siz   - height and width of the desired field, of the form [n n] for 2-d fields
%          (note: height and width must be the same)
%  law   - a power law for the radial profile of the synthetic spectrum, written
%          in terms of "kr" for 2d fields or "k" for 1d fields
%
% OUTPUTS:
%  sspec - the synthetic spectrum (the DFT of the height field)
%  fld   - the (real) height field (the IDFT of the spectrum)
%
% Function written by Oded Aharonson, Aug. 2000
% Modified slightly by Chris Carr, Spring 2001.

if length(siz)==1,

  n=siz(1);
  nh=n/2;
  nh1=nh+1;
  k=fftaxis(n);
  rp=2.*pi.*rand(size(k)); % random phases
  sspec=eval(law).*exp(i*rp); % compute spectrum

  % ensure spectrum has required symmetry properties
  sspec(1)=real(sspec(1));
```

```
  sspec(nh1:n)=conj(flipud(fliplr(sspec(2:nh1))));
  sspec(~isfinite(sspec))=0;
  sspec=ifftshift(sspec);
  sf=ifft(sspec);
  fld=real(sf); % make sure field is real
  sspec=fft(fld); % recompute spectrum based on actual field created

elseif length(siz)==2,
  if siz(1)~=siz(2)
    error('synth_spect: only square 2D fields are implemented!');
  end

  [kx,ky]=fftaxis(siz);
  kr=sqrt(kx.^2+ky.^2);
  rp=2*pi*rand(siz);  % use random phases
  sspec=zeros(siz);

  n=siz(1);
  nh=n/2;
  nh1=nh+1;

  negs=1:nh;
  poss=nh+2:n;
  npos=1:(nh1);
  nneg=(nh1):n;

  sspec=eval(law).*exp(i*rp); % generate spectrum
  sspec=fftshift(sspec); % shift low freq to center

  % ensure symmetry properties for the generated field
  sspec(nh1:n,nh1:n)=conj(flipud(fliplr(sspec(2:nh1,2:nh1))));
  sspec(nh1:n,2:nh1)=conj(flipud(fliplr(sspec(2:nh1,nh1:n))));
  sspec(1,nh1:n)=conj(flipud(fliplr(sspec(1,2:nh1))));
  sspec(nh1:n,1)=conj(flipud(fliplr(sspec(2:nh1,1))));
  sspec(nh1,nh1)=real(sspec(nh1,nh1));
  sspec(1,1)=real(sspec(1,1));
  sspec(1,nh1)=real(sspec(1,nh1));
  sspec(nh1,1)=real(sspec(nh1,1));
  sspec(~isfinite(sspec))=0;
  sspec(1,1)=min(real(sspec(:)));

  % compute inverse fourier transform
  sf=ifft2(sspec);
  %disp(diff(minmax(imag(sf)))./diff(minmax(real(sf))));
  fld=real(sf); % make sure field is real
  sspec=fft2(fld); % recompute spectrum

else
  error('synth_spect; only 1 or 2 D fields are implemented!');
end
```

D.6.12 FFTAxis

```
function [varargout]=fftaxis(siz,Fs,use_toolbox)
% fftaxis: generate frequency axis for an fft.
%
% Optional argument Fs is 1/dt, the sampling frequency which scales the result.
% By default, uses freqspace function in the signal toolbox, but can be
% overridden with use_toolbox=0.
%
% Written by Oded Aharonson.
```

```
defaultval('Fs',1,1);
defaultval('use_toolbox',1,1);

if use_toolbox,

  ndim=length(siz);
  k1=cell(ndim,1);

  if ndim==2,
    [kx,ky]=freqspace(siz,'meshgrid');
    kx=kx./2; ky=ky./2;
    kx=kx.*Fs; ky=ky.*Fs;
    varargout={kx,ky};
  elseif ndim==1,
    k=freqspace(siz,'whole');
    k=(k-1)./2;
    k=k.*Fs;
    varargout{1}=k;
  else
    error('fftaxis: only 1 or 2D implemented using freqspace!');
  end

else

  ndim=length(siz);
  k1=cell(ndim,1);

  for idim=1:ndim,
    sizi=siz(idim);
    if ~iseven(sizi),
      error('fftaxis: only even size implemented!');
    end
    sizh=sizi/2;
    k1{idim}=((-sizh):(sizh-1))*Fs/sizi;
  end

  if ndim>1,
    k=cell(ndim,1);
    [k{:}]=meshgrid(k1{:});
  else
    k=k1;
  end
  varargout=k;
end
```

D.7 Miscellaneous Functions

D.7.1 BivariateNormal

```
function [pdf]=bivariatenormal(num,ex,ey,sx,sy,exy,x,y,)
% function [pdf]=bivariatenormal(num,ex,ey,sx,sy,exy,x,y)
%
% Returns the value of the PDF at the location x,y.
%
% INPUTS
%  num = number of experimental values sampled from the PDF
```

```
%  ex  = expected value of x, E[x]
%  ey  = expected value of y, E[y]
%  sx  = standard deviation of x
%  sy  = standard deviation of y
%  exy = expected value of xy, E[xy]
%  x   = 1 x num vector of x coordinates
%  y   = 1 x num vector of y coordinates
%
% OUTPUTS
%  pdf = value of the probability density function at (x,y)
%
% Author: Chris Carr
% Date: August, 2001

rho = (exy-(ex*ey))/(sx*sy);
pre = 1/(2*pi*sx*sy*sqrt(1-rho^2));
frac = ((x.^2/sx^2)-(2*rho*((x/sx).*(y/sy)))+(y.^2/sy^2))/(2*(1-rho^2));
pdf = pre*exp(-frac);
```

D.7.2 BivariateNormal_Ellipse

```
function [p1,p2,a,x,y]=bivariatenormal_ellipse(num,ex,ey,sx,sy,rho)
% function [p1,p2,a,x,y]=bivariatenormal_ellipse(num,ex,ey,sx,sy,rho)
%
% Returns the error ellipse principal directions for the specified
% bivariate normal distribution, along with num (x,y) points along
% the 3-sigma error ellipse.
%
% INPUTS
%  num = number of x,y samples along the 3-sigma ellipse
%  ex  = expected value of x, E[x]
%  ey  = expected value of y, E[y]
%  sx  = standard deviation of x
%  sy  = standard deviation of y
%  rho = correlation coefficient rho(x,y)
%
% OUTPUTS
%  p1,p2 = principle axes of the error ellipse
%  a = angle between x axis and principle direction of error ellipse
%  x = points on the 3-sigma error ellipse
%  y = points on the 3-sigma error ellipse
%
% Author: Chris Carr
% Date: August, 2001

tan2a = 2*rho*sx*sy/(sx^2-sy^2);
a = 0.5*atan(tan2a);

p1=sqrt((((sx^2)*(sy^2))*(1-rho^2))/( ((sy^2)*(cos(a)^2)-
2*rho*sx*sy*sin(a)*cos(a)+(sx^2)*(sin(a)^2))));
p2=sqrt((((sx^2)*(sy^2))*(1-rho^2))/(
((sy^2)*(sin(a)^2)+2*rho*sx*sy*sin(a)*cos(a)+(sx^2)*(cos(a)^2))));

% draw 3 sigma envelope
theta = linspace(0,2*pi,num);
X = 3*p1*cos(theta)-3*p1*sin(theta); % unrotated 3-sigma x
Y = 3*p2*sin(theta)+3*p2*cos(theta); % unrotated 3-sigma y
theta = 0.5*acot((sx^2-sy^2)/(sx*sy*rho));
% convert X and Y to cylindrical coords
```

```
[th,r]=cart2pol(X,Y);
[x,y]=pol2cart(th+theta,r);

function [x,y]=bivariatenormal2(num,ex,ey,sx,sy,rho)
% function [x,y]=bivariatenormal2(num,ex,ey,sx,sy,rho)
%
% Returns num values sampled from a bivariate normal probability
% density function.
%
% INPUTS
%  num = number of experimental values sampled from the PDF
%  ex  = expected value of x, E[x]
%  ey  = expected value of y, E[y]
%  sx  = standard deviation of x
%  sy  = standard deviation of y
%  rho = correlation coefficient of x and y, rho(x,y)
%
% OUTPUTS
%  x   = 1 x num vector of x coordinates
%  y   = 1 x num vector of y coordinates
%
% Author: Chris Carr
% Date: August, 2001

Z1 = randn(1,num);
Z2 = randn(1,num);
a = rho/sqrt(1-rho^2);
W = a*Z1+Z2;
W_mean = 0;
W_std = sqrt(a^2+1+2*a*rho);
W_n = (W-W_mean)/W_std;

x = sx*Z1+ex;
y = sy*W_n+ey;
```

D.7.3 Colormap_Create

```
function c_out = colormap_create(c_in)

% function c_out = colormap_create(c_in)
%
% Creates a color map using the RGB colors specified
% by c_in.
%
% INPUT
%  c_in = n x 3 matrix of R-G-B color values; c_in(:,1) = R values;
%         c_in(:,2)= G values, c_in(:,3) = B values
%
% OUTPUTS
%  c_out = 64 x 3 matrix of R-G-B color values, interpolated
%          based on c_in. Can be used as an input value to
%          the MATLAB "colormap" m function.
%
% Author: Chris Carr, May, 2001.

xi = linspace(1,size(c_in,1),64);

ri = interp1(c_in(:,1),xi)';
gi = interp1(c_in(:,2),xi)';
```

```
bi = interp1(c_in(:,3),xi)';

c_out = [ri/255 gi/255 bi/255];
```

D.7.4 CrossEntropy

```
function D = crossentropy(p,q)

% function D = crossentropy(p,q)
%
% Computes the cross entropy between two probability distributions p and q,
% where q is assumed to be the a-priori distribution.
%
% INPUTS
%  p = 1xn vector representing samples of a probability distribution
%  q = 1xn vector representing samples of a probability distribution
%
% OUTPUTS
%  D = a scalar value, the cross-entropy between the two probability distributions
%      p and q.
%
% Author: Chris Carr, May 2001.

D = 0;

for (i=1:size(p,2)),
    if q(i)~=0, % let Di = 0 for q(i)=0
        Di = p(i)*log(p(i)/q(i));
        D = D + Di;
    end
end
```

D.7.5 DefaultVal

```
function defaultval(arg,defval,quiet)
% defaultval: sets a variable to a default value if the variable is undefined.
%
% Checks if a variable arg exists in the caller's workspace,
% if nonexistent or if empty arg is set to value defval in caller's workspace
% If quiet does not exist or is 1, then will print value being used to screen.
%
% Example:
%
% function y=pow2(x,n)
% defaultval('n',2)
% y=x.^n;
%
%
% Written by Keli, 5/1/00
% Modified by Oded Aharonson, 5/1/00

if ~exist('quiet'),
  quiet=0;
end

flag=1;
if evalin('caller',[ 'exist(''' arg ''')']);,
```

```
  flag=evalin('caller',[ 'isempty(' arg ')']);
end
if flag,
  if ~quiet,
    if (ischar(defval) | length(defval)<3)
      [path,prog,ext,ver]=star69;
      verbose([prog,': Using ',arg,'=',num2str(defval)]);
    end
  end
  assignin('caller',arg,defval);
end
```

D.7.6 Std_Dev_Range

```
function s = std_dev_range(x,range)
% function s = std_dev_range(x,range)
%
% Computes the standard deviation of the
% set of samples of x in the range specified
% by the range parameter.
%
% INPUTS
%  x = a row vector of data
%  range = [min max], where [min,max) specifies the range
%          of the set whose standard deviation will be computed.
%
% OUTPUTS
%  s = standard deviation of the subset of x in the interval [min,max)
%
% Author: Chris Carr, June 2001.

less_than = find(x<range(2));
greater_or_equal = find(x>=range(1));
i = intersect(less_than,greater_or_equal); % desired set has both properties
s = std(x(i));
```

D.7.7 Verbose

```
function vn=verbose(v)

% function vn=verbose(v)
%
% Sets or returns the verbose setting.
%
% INPUTS
%  v = 1 (turns on verbose mode)
%    = 0 (turns off verbose mode)
%
% OUTPUTS
%  vn = new verbose mode setting
%
% usage: verbose(1); verbose(0); verbose;
% Using verbose with no arguments returns the current verbose mode.
%
% Author: Chris Carr, May 2001. Modeled after similar function by Oded Aharonson.

persistent vb;

if nargin<1,
```

```
    vn=vb;
else
    vb = v;
    vn = vb;
end
```

D.8 Rover Example

D.8.1 Rover_Example

```
%
% Rover_Example.m
%

%clear all;
close all;
clc;
verbose(1);

% mission:
% deploy a sensor network consisting of N sensor/comm pods
% have nominal traverse plan
% question: how often and where drop sensor pods along the
% way to have continuous network coverage

mode = 2; % 1=pick points, 2 = setup, 3=run
subsample = 1; % factor for subsampling of surface for testing purposes
trial_points = 100; % number of trial points to use when computing a minimum cost traverse
nom_dist =1000; % nom distance between sensors for comparison
mm = 2; % space loss exponent
maxplanits = 10;

% number of communication sensors has already been deployed in a lakebed
% goal is to build a bridge of communication sensors to the existing
% distributed system
N = []; % node id list number of available comm nodes/sensor nodes
N_h = 1.5; % effective altitude above surface of sensor node
N_P = []; % node position list
X_N = []; % node position list
R_ch = 1.5; % effective altitude above surface of rover for communications
R_vh = 1.5; % effective altitude above surface of rover for visibility
S_ch = 1.5; % effective altitude above surface of source node
T_ch = 1.5; % effective altitude above surface of target node
maxslope = 20; % maximum slope in degrees navigable by the rover
vel = 0.5; % nominal velocity of 0.5 m/sec
mass = 50; % 50 kg rover
g = 3.69; % gravitational acceleration (mars); lunar = 1.62 m/s^2
% rover position and traverse
R_x = 558671;
R_y = 4743633;
R_z = 1900; % will be initialized later
S_x = R_x;
S_y = R_y;
S_z = R_z;
% target position and traverse
T_x = 554676;
T_y = 4737260;
T_z = 1700; % will be initialized later
```

```
if mode==1,
    disp('ROVER SIMULATION: POINT PICKING MODE');
    % load CRLA dem
    disp('Loading Crater Lake DEM');
    [X,Y,Z,p]=surface_import('crla.mat',1);
    p(5)=10;
    % recompute x and y
    width = p(1);
    height = p(2);
    xllcorner = p(3);
    yllcorner = p(4);
    cellsize = p(5);
    X = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
    Y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);

    % extract valid area
    [X,Y,Z,p]=surface_extract_valid(X,Y,Z,p);
    load cmap_alt.mat;
    h = figure;
    surface_plot_minimal(h,X,Y,Z);
    set(gca,'DataAspectRatio',[1 1 1]);
    colormap(cmap_altitude);
    [X,Y]=ginput(1);
    disp(sprintf('X: %0.2f, Y %0.2f',X,Y));
elseif mode==2,
    % load CRLA dem
    disp('ROVER SIMULATION: SETUP MODE - PRECOMPUTING SIMULATION COMPONENTS');
    disp('Loading Crater Lake DEM');
    [X,Y,Z,p]=surface_import('crla.mat',1);
    p(5)=10; % wrong in file
    % recompute x and y
    width = p(1);
    height = p(2);
    xllcorner = p(3);
    yllcorner = p(4);
    cellsize = p(5);
    X = linspace(xllcorner,xllcorner+(width-1)*cellsize,width);
    Y = linspace(yllcorner+(height-1)*cellsize,yllcorner,height);

    % extract valid area
    [X,Y,Z,p]=surface_extract_valid(X,Y,Z,p);
    load cmap_alt.mat;
    % compute subsample
    disp('Subsampling the Crater Lake DEM');
    [X,Y,Z,p]=surface_compute_subsample(X,Y,Z,p,subsample);
    % compute slope map
    disp('Computing a slope map for the Crater Lake DEM');
    [Xs,Ys,Zs,ps]=surface_compute_slope(X,Y,Z,p);
    % compute reachability map
    disp(sprintf('Computing surface reachability based on [0 %d] slope constraint',maxslope));
    R_P = surface_extract_altitudes(Z,p,R_x,R_y);
    R_z = R_P(3,1);
    S_P = R_P; % initialize source node
    S_z = R_P(3,1)+S_ch;
    S_Pc = surface_extract_altitudes(Z,p,S_x,S_y);
    S_Pc(3,1)=S_Pc(3,1) + S_ch;
    T_Pc = surface_extract_altitudes(Z,p,T_x,T_y);
    T_Pc(3,1)=T_Pc(3,1) + T_ch;
    T_P = surface_extract_altitudes(Z,p,T_x,T_y);
    R = surface_compute_reachability(Zs,ps,R_P,[0 maxslope],1,0);
    disp('Saving results for Rover Simulation Setup');
```

```
    save exp47_setup.mat;
else

diary('exp47_diary.txt');
disp('INITIALIZING ROVER SIMULATION');
load exp47_setup.mat;

% initial conditions
% traverse entry id and time
ctr=1;
t = 0; % 9 AM
R_T = [ctr R_x R_y R_z 0 ctr t 0];
R_E = 0; % initialize rover energy expenditures to zero
% initialize goal states
mission_accomplished = 0;
give_up = 0;
nodes_released = 0;
source_visible = 1;
target_visible = 0;
nodes_connected = 1;
mean_cost = NaN;
Data(ctr,:) = [nodes_released source_visible target_visible nodes_connected mean_cost];
% overall visibility map
V = zeros(size(Z,1),size(Z,2))*NaN;

% deployment activities
while and(not(mission_accomplished),not(give_up)),
    % publish rover current state
    disp(sprintf('ROVER STATE: id=%d x=%d y=%d z=%d kJ=%d t=%d',ctr,R_x,R_y,R_z,round(R_E/
1000),round(t)));
    disp(sprintf('NETWORK STATE: Nodes released=%d SourceVisible=%d TargetVisible=%d NodesCon-
nected=%d Mean-
Cost=%0.2f',nodes_released,source_visible,target_visible,nodes_connected,mean_cost));
    % convert current position to a point
    R_P = surface_extract_altitudes(Z,p,R_x,R_y);
    R_Pv = R_P;
    R_Pc = R_P;
    R_Pv(3,1)=R_Pv(3,1)+R_vh;
    R_Pc(3,1)=R_Pc(3,1)+R_ch;

    % debug: plot the traverse
    figure(1);
    surface_plot_minimal(1,X,Y,R);
    colormap(cmap_altitude);
    set(gca,'DataAspectRatio',[1 1 1]);

    % compute traverse to the destination

[T,c,fail]=traverse_compute_min_cost(X,Y,Z,p,[R_x;R_y;R_P(3,1)],[T_x;T_y;T_P(3,1)],vel,mass,g,1,t
rial_points,R);

    % compute what is visible from current location
    [C,c]=surface_compute_coverage_vector(Z,p,R_Pv,3,0,0,0);
    % update visibility of world
    V = surface_merge2(V,C,0);

    found_next_sensor_position = 0;
    look_for_point_count = 0;
    while and(not(found_next_sensor_position),not(give_up)),
        % what is the furthest point on the traverse in my zone of current visibility?
        no_connected_points = 0;
        Xt = T(2:size(T,1),2:4)'; % extract traverse point positions
```

```
        if isempty(Xt),
            idx_visible = [];
            no_connected_points = 1;
        else
            Pt = surface_extract_altitudes(Z,p,Xt(1,:),Xt(2,:)); % convert positions to points
            Xt(3,:)=Pt(3,:);
            Vt = zeros(1,size(Xt,2));
            for pp=1:size(Pt,2),
                Vt(pp) = V(Pt(2,pp),Pt(1,pp)); % extract visibility for this traverse point
            end
            % find all visible points
            idx_visible = find(Vt>=0);
        end

        if not(isempty(idx_visible)),
            % pick point with greatest index
            idx_new = max(idx_visible);
            Xnew = Xt(:,idx_new);
            found_next_sensor_position = 1;
            disp(sprintf('ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: %d Y: %d Z:
%d',Xnew(1,1),Xnew(2,1),Xnew(3,1)));
        else
            if look_for_point_count > maxplanits,
                % give up
               disp(sprintf('ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, GIVING
UP'));
                give_up = 1;
            else
                look_for_point_count = look_for_point_count+1;
                if no_connected_points,
                    % try another traverse directly to the target
                    disp(sprintf('ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION,
TRYING FOR TARGET'));
                    % no points on the traverse are in my current zone of visibility
                  % plan a traverse to the first point of the old traverse, and try this loop again

[T,c,fail]=traverse_compute_min_cost(X,Y,Z,p,[R_x;R_y;R_P(3,1)],[T_x;T_y;T_P(3,1)],vel,mass,g,1,t
rial_points,R);
                else
                    disp(sprintf('ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION,
TRYING FOR 1ST TRAVERSE POINT'));
                    % no points on the traverse are in my current zone of visibility
                  % plan a traverse to the first point of the old traverse, and try this loop again

[T,c,fail]=traverse_compute_min_cost(X,Y,Z,p,[R_x;R_y;R_P(3,1)],[T(2,2);T(2,3);T(2,4)],vel,mass,g
,1,trial_points,R);
                end
            end
        end
    end

    if not(give_up),
        % traverse to the new sensor location Xnew, and place a sensor there
        disp(sprintf('ROVER TRAVERSING FROM X: %d Y: %d Z: %d to X: %d Y: %d Z:
%d',R_x,R_y,R_z,Xnew(1,1),Xnew(2,1),Xnew(3,1)));
        % build a traverse to the new location
        dist = sqrt((Xnew(1,1)-R_x)^2+(Xnew(2,1)-R_y)^2+(Xnew(3,1)-R_z)^2);
        t_new = vel/dist+t;
        T = [[ctr R_x R_y R_z 0 ctr t 0];[ctr+1 Xnew(1,1) Xnew(2,1) Xnew(3,1) 0 ctr+1 t_new 0]];
        T = traverse_project(Z,p,traverse_interpolate(T,p(5),0));
        m = traverse_compute_rover_cost(T,mass,g);
        % update the position of the rover and the energy expenditure of the rover
```

```
ctr = ctr+1;
t=t_new;
R_x = Xnew(1,1);
R_y = Xnew(2,1);
R_z = Xnew(3,1);
R_E = R_E + sum(m(:,5),1);
R_P = surface_extract_altitudes(Z,p,R_x,R_y);
R_Pc = R_P;
R_Pc(3,1)=R_Pc(3,1)+R_ch;

% compute LOS for the whole system - see if can see destination node and source node
nodes_released = nodes_released+1;
N(nodes_released)=nodes_released;
X_N(1,nodes_released)=R_x;
X_N(2,nodes_released)=R_y;
X_N(3,nodes_released)=R_z+N_h;
N_P = surface_convert_crd_to_pts(p,X_N);
total_nodes = nodes_released+3; % rover, source, and target
% format for Vm: [nodes released, rover, target, source]
VT = [N total_nodes-2 total_nodes-1 total_nodes];
PT = [N_P R_Pc T_Pc S_Pc];
[VT,ET,PT]=graph_create_NEP(Z,VT,PT,0);

% compute cost of edges
CT = zeros(1,size(ET,2));
XT = [X_N [R_x;R_y;R_z] [T_x;T_y;T_z] [S_x;S_y;S_z]];
for idx_e=1:size(ET,2),
    ni = ET(1,idx_e);
    nj = ET(2,idx_e);
    % compute cost of edge ij
    dist_e = sqrt((XT(1,ni)-XT(1,nj))^2+(XT(2,ni)-XT(2,nj))^2+(XT(3,ni)-XT(3,nj))^2);
    CT(idx_e)=((dist_e/nom_dist))^mm;
end

% compute shortest path tree
[Vm,Em,D,Cm]=graph_compute_dsp(VT,ET,CT,total_nodes-2); % total_nodes-2 is rover = root of
tree

nodes_connected = size(Vm,2)-1; % subtract the rover
source_visible = not(isempty(find(Vm==total_nodes)));
target_visible = not(isempty(find(Vm==total_nodes-1)));
mean_cost = mean(Cm);

% debug
pause;
close(1);
figure(1);
surface_plot_minimal(1,X,Y,log(V));
%colormap(cmap_altitude);
set(gca,'DataAspectRatio',[1 1 1]);
set(gca,'nextplot','add');
graph_plot_points(1,PT,p);
h = graph_plot_edges(1,VT,ET,PT,p);
for jjj=1:size(h,2),
    set(h(jjj),'LineWidth',2);
end
pause;
close(1);

% store data for summary figures, etc.
% add traverse point to official Rover traverse
R_T(ctr,:)=[ctr R_x R_y R_z 0 ctr t 0];
```

```
        Data(ctr,1:5) = [nodes_released source_visible target_visible nodes_connected mean_cost];

        % if can't see source node, output rover and network state and then end
        if not(source_visible),
            disp('SOURCE NODE NO LONGER CONNECTED! QUITTING SIMULATION');
            disp(sprintf('FINAL ROVER STATE: id=%d x=%d y=%d z=%d kJ=%d
t=%d',ctr,R_x,R_y,R_z,R_E,t));
           disp(sprintf('FINAL NETWORK STATE: Nodes released=%d SourceVisible=%d TargetVisible=%d
NodesConnected=%d Mean-
Cost=%0.2f',nodes_released,source_visible,target_visible,nodes_connected,mean_cost));
            give_up = 1;
        elseif and(source_visible,target_visible),
            % if can see destination and source node, then mission accomplished
            % show rover and network state and end
            disp('SOURCE AND TARGET NODE ARE CONNECTED! MISSION ACCOMPLISHED');
            disp(sprintf('FINAL ROVER STATE: id=%d x=%d y=%d z=%d kJ=%d
t=%d',ctr,R_x,R_y,R_z,R_E,t));
           disp(sprintf('FINAL NETWORK STATE: Nodes released=%d SourceVisible=%d TargetVisible=%d
NodesConnected=%d Mean-
Cost=%0.2f',nodes_released,source_visible,target_visible,nodes_connected,mean_cost));
            give_up = 1;
        end
    end
end

diary off;

end
```

D.8.2 Example Simulation Listing for Failure

(Edited for Clarity)

```
INITIALIZING ROVER SIMULATION

ROVER STATE: id=1 x=558671 y=4743633 z=1874 kJ=0 t=0
NETWORK STATE: Nodes released=0 SourceVisible=1 TargetVisible=0 NodesConnected=1 MeanCost=NaN
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 556985 Y: 4742685 Z: 1908
ROVER TRAVERSING FROM X: 558671 Y: 4743633 Z: 1874 to X: 556985 Y: 4742685 Z: 1908

ROVER STATE: id=2 x=556985 y=4742685 z=1908 kJ=34 t=0
NETWORK STATE: Nodes released=1 SourceVisible=1 TargetVisible=0 NodesConnected=2 MeanCost=1.25
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 557465 Y: 4741885 Z: 2033
ROVER TRAVERSING FROM X: 556985 Y: 4742685 Z: 1908 to X: 557465 Y: 4741885 Z: 2033

ROVER STATE: id=3 x=557465 y=4741885 z=2033 kJ=65 t=0
NETWORK STATE: Nodes released=2 SourceVisible=1 TargetVisible=0 NodesConnected=3 MeanCost=1.16
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, GIVING UP
```

D.8.3 Example Simulation Listing for Success

(Edited for Clarity)

```
INITIALIZING ROVER SIMULATION

ROVER STATE: id=1 x=558671 y=4743633 z=1874 kJ=0 t=0
NETWORK STATE: Nodes released=0 SourceVisible=1 TargetVisible=0 NodesConnected=1 MeanCost=NaN
```

```
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 557385 Y: 4742605 Z: 1907
ROVER TRAVERSING FROM X: 558671 Y: 4743633 Z: 1874 to X: 557385 Y: 4742605 Z: 1907

ROVER STATE: id=2 x=557385 y=4742605 z=1907 kJ=30 t=0
NETWORK STATE: Nodes released=1 SourceVisible=1 TargetVisible=0 NodesConnected=2 MeanCost=0.90
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 556585 Y: 4742525 Z: 1920
ROVER TRAVERSING FROM X: 557385 Y: 4742605 Z: 1907 to X: 556585 Y: 4742525 Z: 1920

ROVER STATE: id=3 x=556585 y=4742525 z=1920 kJ=42 t=0
NETWORK STATE: Nodes released=2 SourceVisible=1 TargetVisible=0 NodesConnected=3 MeanCost=2.07
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 556425 Y: 4742525 Z: 1901
ROVER TRAVERSING FROM X: 556585 Y: 4742525 Z: 1920 to X: 556425 Y: 4742525 Z: 1901

ROVER STATE: id=4 x=556425 y=4742525 z=1901 kJ=43 t=0
NETWORK STATE: Nodes released=3 SourceVisible=1 TargetVisible=0 NodesConnected=4 MeanCost=1.66
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 556105 Y: 4742525 Z: 1869
ROVER TRAVERSING FROM X: 556425 Y: 4742525 Z: 1901 to X: 556105 Y: 4742525 Z: 1869

ROVER STATE: id=5 x=556105 y=4742525 z=1869 kJ=45 t=0
NETWORK STATE: Nodes released=4 SourceVisible=1 TargetVisible=0 NodesConnected=5 MeanCost=1.40
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 555785 Y: 4742685 Z: 1836
ROVER TRAVERSING FROM X: 556105 Y: 4742525 Z: 1869 to X: 555785 Y: 4742685 Z: 1836

ROVER STATE: id=6 x=555785 y=4742685 z=1836 kJ=47 t=0
NETWORK STATE: Nodes released=5 SourceVisible=1 TargetVisible=0 NodesConnected=6 MeanCost=1.22
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 555545 Y: 4742605 Z: 1828
ROVER TRAVERSING FROM X: 555785 Y: 4742685 Z: 1836 to X: 555545 Y: 4742605 Z: 1828

ROVER STATE: id=7 x=555545 y=4742605 z=1828 kJ=50 t=0
NETWORK STATE: Nodes released=6 SourceVisible=1 TargetVisible=0 NodesConnected=7 MeanCost=1.17
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 555145 Y: 4742285 Z: 1797

ROVER STATE: id=8 x=555145 y=4742285 z=1797 kJ=54 t=0
NETWORK STATE: Nodes released=7 SourceVisible=1 TargetVisible=0 NodesConnected=8 MeanCost=1.07
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR 1ST TRAVERSE POINT
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 555225 Y: 4741645 Z: 1892
ROVER TRAVERSING FROM X: 555145 Y: 4742285 Z: 1797 to X: 555225 Y: 4741645 Z: 1892

ROVER STATE: id=9 x=555225 y=4741645 z=1892 kJ=78 t=0
NETWORK STATE: Nodes released=8 SourceVisible=1 TargetVisible=0 NodesConnected=9 MeanCost=1.14
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 555145 Y: 4740285 Z: 1812
ROVER TRAVERSING FROM X: 555225 Y: 4741645 Z: 1892 to X: 555145 Y: 4740285 Z: 1812

ROVER STATE: id=10 x=555145 y=4740285 z=1812 kJ=99 t=0
NETWORK STATE: Nodes released=9 SourceVisible=1 TargetVisible=0 NodesConnected=10 MeanCost=1.20
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554585 Y: 4739485 Z: 1698
ROVER TRAVERSING FROM X: 555145 Y: 4740285 Z: 1812 to X: 554585 Y: 4739485 Z: 1698

ROVER STATE: id=11 x=554585 y=4739485 z=1698 kJ=111 t=0
NETWORK STATE: Nodes released=10 SourceVisible=1 TargetVisible=0 NodesConnected=11 MeanCost=1.19
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4739085 Z: 1732
ROVER TRAVERSING FROM X: 554585 Y: 4739485 Z: 1698 to X: 554665 Y: 4739085 Z: 1732

ROVER STATE: id=12 x=554665 y=4739085 z=1732 kJ=122 t=0
NETWORK STATE: Nodes released=11 SourceVisible=1 TargetVisible=0 NodesConnected=12 MeanCost=1.22
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4738845 Z: 1878
ROVER TRAVERSING FROM X: 554665 Y: 4739085 Z: 1732 to X: 554665 Y: 4738845 Z: 1878

ROVER STATE: id=13 x=554665 y=4738845 z=1878 kJ=152 t=0
NETWORK STATE: Nodes released=12 SourceVisible=1 TargetVisible=0 NodesConnected=13 MeanCost=1.17
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
ROVER FAILED TO FIND SUITABLE NEXT COMM/SENSOR NODE LOCATION, TRYING FOR TARGET
```

```
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4738445 Z: 1683
ROVER TRAVERSING FROM X: 554665 Y: 4738845 Z: 1878 to X: 554665 Y: 4738445 Z: 1683

ROVER STATE: id=14 x=554665 y=4738445 z=1683 kJ=146 t=0
NETWORK STATE: Nodes released=13 SourceVisible=1 TargetVisible=0 NodesConnected=14 MeanCost=1.10
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4738125 Z: 1683
ROVER TRAVERSING FROM X: 554665 Y: 4738445 Z: 1683 to X: 554665 Y: 4738125 Z: 1683

ROVER STATE: id=15 x=554665 y=4738125 z=1683 kJ=150 t=0
NETWORK STATE: Nodes released=14 SourceVisible=1 TargetVisible=0 NodesConnected=15 MeanCost=1.04
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4737885 Z: 1704
ROVER TRAVERSING FROM X: 554665 Y: 4738125 Z: 1683 to X: 554665 Y: 4737885 Z: 1704

ROVER STATE: id=16 x=554665 y=4737885 z=1704 kJ=156 t=0
NETWORK STATE: Nodes released=15 SourceVisible=1 TargetVisible=0 NodesConnected=16 MeanCost=1.00
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4737725 Z: 1761
ROVER TRAVERSING FROM X: 554665 Y: 4737885 Z: 1704 to X: 554665 Y: 4737725 Z: 1761

ROVER STATE: id=17 x=554665 y=4737725 z=1761 kJ=168 t=0
NETWORK STATE: Nodes released=16 SourceVisible=1 TargetVisible=0 NodesConnected=17 MeanCost=0.94
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4737645 Z: 1773
ROVER TRAVERSING FROM X: 554665 Y: 4737725 Z: 1761 to X: 554665 Y: 4737645 Z: 1773

ROVER STATE: id=18 x=554665 y=4737645 z=1773 kJ=171 t=0
NETWORK STATE: Nodes released=17 SourceVisible=1 TargetVisible=0 NodesConnected=18 MeanCost=0.89
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4737565 Z: 1784
ROVER TRAVERSING FROM X: 554665 Y: 4737645 Z: 1773 to X: 554665 Y: 4737565 Z: 1784

ROVER STATE: id=19 x=554665 y=4737565 z=1784 kJ=174 t=0
NETWORK STATE: Nodes released=18 SourceVisible=1 TargetVisible=0 NodesConnected=19 MeanCost=0.84
ROVER IDENTIFIED NEXT COMM/SENSOR NODE at X: 554665 Y: 4737485 Z: 1806
ROVER TRAVERSING FROM X: 554665 Y: 4737565 Z: 1784 to X: 554665 Y: 4737485 Z: 1806
SOURCE AND TARGET NODE ARE CONNECTED! MISSION ACCOMPLISHED
FINAL ROVER STATE: id=20 x=554665 y=4737485 z=1806 kJ=1.785951e+005 t=4.067217e-002
FINAL NETWORK STATE: Nodes released=19 SourceVisible=1 TargetVisible=1 NodesConnected=21 MeanCost=0.77
```

Begin at the beginning and go on till you come to the end;
then stop.

Lewis Carroll, *Alice in Wonderland*

Bibliography

[1] *Advanced Technology for Human Support in Space*, National Academy Press, Washington D.C., 1997 (*http://www.nap.edu/catalog/5826.html*).

[2] Aharonson, O., Zuber, M.T., and Rothman, D.H., *Statistics of Mars' Topography from the Mars Orbiter Laser Altimeter: Slopes, Correlations, and Physical Models*, submitted to J. Geophysical Research, 2000.

[3] Allen, John L., *Passage Through the Garden*, University of Illinois Press, Urbana, Illinois, 1975.

[4] Allton, J.H., *Catalog of Apollo Lunar Surface Geological Sampling Tools and Containers* (JSC-23454), National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1989.

[5] Apollo Experience Report: Assessment of Metabolic Expenditures (NASA TN D-7883/N75-18271), National Aeronautics and Space Administration, 1975.

[6] Appelbaum, J., and Steiner, A., *Spectral Content of Solar Radiation on Martian Surface Based on Mars Pathfinder*, J. Propulsion and Power, Vol. 17, No. 3, May-June 2001.

[7] Baecher, Gregory B., *Site Exploration: A Probabilistic Approach*, PhD Thesis, Massachusetts Institute of Technology, 1972.

[8] Baldwin, J., Fisher, P., Wood, J., and Langford, M., *Modelling environmental cognition of the view with GIS*, Proceedings, Third International Conference on Integrated GIS and Environmental Modelling, 1996 (*http://www.geog.le.ac.uk/jwo/research/dem_char/SantaFe/index.html*).

[9] Barbosa, Valmir C., *An Introduction to Distributed Algorithms*, MIT Press, Cambridge, Massachusetts, 1996.

[10] Bluetooth, *Bluetooth website* (*http://www.bluetooth.org/*), 2001.

[11] Braak, Laurent, and Clement, Gilles, *Mathematical Model of Human Physiological Responses During Extra-Vehicular*

Activity, European Space Agency Purchase Order 120639, 1992.

[12] Burchfiel, B. Clark, Professor of Geology in the Department of Earth, Atmospheric, and Planetary Sciences at the Massachusetts Institute of Technology, *Personal Communication*, 2001.

[13] Burton, John H., Project Manager, and Yoha, Robert, Principal, *GeoSAR: A Radar Based Terrain Mapping Project, Year 2 Research and Development Status Report*, California Department of Conservation, 1996 (*http://www.consrv.ca.gov/radar/geosar/year2rpt/toc.html*).

[14] Carr, C.E., and Newman, D.J., *Applications of Wearable Computing to Exploration in Extreme Environments*, 3rd Annual Mars Society Convention, Toronto, Canada, 2000 (submitted).

[15] Carr, C.E., and Schwartz, S.J., and Newman, D.J., *Preliminary Considerations for Wearable Computing in Support of Astronaut Extravehicular Activity*, Massachusetts Institute of Technology, 2001(unpublished).

[16] Cavagna, G.A., Zamboni, A., Farragiana, T., Margaria, R., *Jumping on the Moon: power output at different gravity values.* Aerospace Medicine, 1972.

[17] Chaikin, Andrew, *A Man on the Moon*, Penguin Books, New York, 1994.

[18] Chakrabarti, S., Mishra, A., *QoS Issues in Ad Hoc Wireless Networks*, IEEE Communication Magazine, February 2001.

[19] Chen, S., *Routing Support for Providing Guaranteed End-To-End Quality of Service*, PhD Thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1999.

[20] *Chronology of Mars Exploration*, National Space Science Data Center, Goddard Space Flight Center, National Aeronautics and Space Administration, Greenbelt, Maryland (*http://nssdc.gsfc.nasa.gov/planetary/chronology_mars.html*).

[21] Clapp, W. Mitchell, *Dirigible Airships for Martian Surface Exploration (AAS 84-176)*, The Case for Mars II, p. 489-496, Science and Technology Series, Volume 62, American Astronautical Society, Univelt, San Diego, California, 1985.

[22] CMEX: Center for Mars Exploration, Ames Research Center, National Aeronautics and Space Administration, Moffet Field, California, 2001 (*http://cmex.arc.nasa.gov*).

[23] Cohen, Paul R., *Empirical Methods for Artificial Intelligence*, MIT Press, Cambridge, Massachusetts, 1995.

[24] Compton, Robert R., *Geology in the Field*, John Wiley & Sons, New York, 1985.

[25] Connors, Mary M., Eppler, Dean B., and Morrow, Daniel G., *Interviews with the Apollo Lunar Surface Astronauts in Support of Planning for EVA Systems Design*, Ames Research Center, National Aeronautics and Space Administration, 1994.

[26] Delin, K.A., *JPL Sensor Webs Project*, Jet Propulsion Laboratory, Pasadena, California, 2001 (*http://sensorwebs.jpl.nasa.gov*).

[27] Dickerson, P.W., *Exploration Strategies for Human Missions*, Mars Field Geology, Biology and Paleontology Workshop, Concepts and Approaches for Mars Exploration, Lunar and Planetary Institute, Houston, Texas, July 18-20, 2000.

[28] Diestel, R., *Graph Theory*, Springer-Verlag, New York, New York, 1997.

[29] Dodds, P.S., and Rothman, D.H., *Scaling, Universality, and Geomorphology*, Annu. Rev. Earth Planet. Sci. 2000. 28:571-610.

[30] Drezner, Z., *Facility Location - A Survey of Applications and Methods*, Springer, New York, Berlin, 1995.

[31] *Engineering Constraints for Mars Surveyor 2003 Landing Site*, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 2001 (*http://mars.jpl.nasa.gov/2001/landingsite/EngConstr.html*).

[32] *EX13-98-036, Reference Mission Version 3.0, Addendum to the Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team*, Exploration Office, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, June, 1998.

[33] FAR/AIM, *Federal Aviation Regulations & Aeronautical Information Manual*, Aviation Supplies & Academics, Inc., 2001.

[34] Gershenfeld, N., *The Nature of Mathematical Modeling*, Cambridge University Press, Cambridge, United Kingdom, 1999.

[35] Gershenfeld, N., *The Physics of Information Technology*, Cambridge University Press, Cambridge, United Kingdom, 2000.

[36] Goddard Workshop, *Science and the Human Exploration of Mars Workshop*, National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland, January 11-12, 2001 (*http://www.lpi.usra.edu/publications/reports/CB-1089/CB-1089.intro.html*).

[37] Graff, C., and Leibman, E., *Concepts for an Adaptive Network Planner for the Localized Access Area (LAA) for Battlefield Information Systems (BIS) 2015*, IEEE 1994.

[38] Gross, J., and Yellen, J., *Graph Theory and Its Applications*, CRC Press, Boca Raton, Florida, 1999.

[39] Hass, Kenneth W., Jr., *Invention and Exploration in Discovery*, PhD Thesis, Massachusetts Institute of Technology, 1992.

[40] Havinga, P., Smit, G., and Bos, M., *Energy-Efficient Adaptive Wireless Network Design*, IEEE 2000.

[41] Heiken, G., Vaniman, D., and French, B.M., *Lunar Sourcebook: A Users's Guide to the Moon*, Cambridge University Press, Cambridge, England, 1991.

[42] Heisenberg, Werner, *Zeitschrift für Physik*, 43 (1927), 172-198, received 23 March 1927.

[43] Hodges, Kip V., *Personal communications*, January, 2001.

[44] Hoffman, S.J., and Kaplan, D.L., *Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team*, Lyndon B. Johnson Space Center, National Aeronautics and Space Administration, July 1997 (*http://spaceflight.nasa.gov/mars/reference/hem/hem1.html*).

[45] Horrigan, David J., Waligora, James M., Beck, Bradley, and Trevino, Robert C., *Chapter 24 - Extravehicular Activities*, Space Biology and Medicine, Volume III, Book 2, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1996.

[46] Howe, D., *Free On-Line Dictionary of Computing*, 2001.

[47] Hutchison, William E., *Personal communications*, January, 2001.

[48] *IEEE 1220 Standard: Application and Management of the Systems Engineering Process*, IEEE Standards Dept., NY, December 1998.

[49] IEEE802.11, *IEEE 802.11 wireless LAN specification*, Institute of Electrical and Electronics Engineers, 2001 (*http://standards.ieee.org/catalog/IEEE802.11.html*).

[50] *INCOSE System Engineering Handbook, Release 1.0*, Internaional Council on Systems Engineering, San Francisco By Area Chapter, January, 1998.

[51] Jha, S., Chalmers, M., Lau, W., Hassan, J., Yap, S., Hassan, M., *Universal Network of Small Wireless Operators (UNSWo)*, IEEE 2001.

[52] Jilla, C.D., Sedwick, R.J., and Miller, D.W., *Application of Multidisciplinary Design Optimization Techniques to Distributed Satellite Systems (AIAA-99-4632)*, AIAA Space Technology Conference & Exposition, Albuquerque, New Mexico, September 28-30, 1999.

[53] Johnston, R.S., Dietlein, L.F., and Berry, C.A., *Biomedical Results of Apollo (SP-368)*, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1975.

[54] Jones, Eric M. (ed.), *Apollo Lunar Surface Journal*, July 2000 (*http://www.hq.nasa.gov/office/pao/History/alsj/*).

[55] Kapur, J.N., and Kesavan, H.K., *Entropy Optimization Principles with Applications*, Academic Press, San Diego, California, 1992.

[56] Kieffer, H.H., Jakosky, B.M., Snyder, C.W., Matthews, M.S. (editors), *Mars*, University of Arizona Press, Tucson, Arizona, 1992.

[57] Kosmo, Joseph J., and Ross, Amy, *Results and Findings of the Representative Planetary Surface EVA Deployment Task Activities, Flagstaff, Arizona* (CTSD-ADV-470), National Aeronautics and Space Administration, Crew and Thermal Systems Division, Lyndon B. Johnson Space Center, Houston, Texas, 2000.

[58] Kosmo, Joseph J., Trevino, Robert, and Ross, Amy, *Results and Findings of the Astronaut-Rover (ASRO) Remote Field Site Test, Silver Lake, California* (CTSD-ADV-360), National Aeronautics and Space Administration, Crew and Thermal Systems Division, Lyndon B. Johnson Space Center, Houston, Texas, 1999.

[59] Kreslavsky, M.A. and Head, J.W. III, *Kilometer-scale slopes on Mars and their correlation with geologic units: Initial results from Mars Orbiter Laser Altimeter (MOLA) data*, J.

Geophysical Research, Vol. 104, No. E9, p. 21911-21924, September 25, 1999.

[60] Kreslavsky, M.A. and Head, J.W. III, *Kilometer-scale roughness of Mars: Results from MOLA data analysis*, J. Geophysical Research, Vol. 105, No. E11, p. 26695-26711, November 25, 2000.

[61] Lewicki, S. A., and Zong, J., *Multi-angle imaging spectroradiometer, Level 1 Ancillary Geographical Product Algorithm Theoretical Basis* (JPL D-13400, Rev. A), Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 1999 (*http://eospso.gsfc.nasa.gov/ftp_ATBD/REVIEW/MISR/ATBD-MISR-05/atbd-misr-05.pdf*).

[62] Lim, J.S., *Two-Dimensional Signal and Image Processing*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[63] Lin,Y., Hsu, Y., *Multihop Cellular: A New Architecture for Wireless Communications*, IEEE InfoComm 2000.

[64] Lu, J., Chuang, J., and Letaief, K.B., Architecture and Signaling for Rapid Infrastructure Deployment Using Wireless Communication Networks, IEEE, 1997.

[65] Margaria, R., *La condizione di subgravita e la sottrazione dall'effetto delle accelerzaioni.* Riv. Med. Aer., Vol. 16, p. 469-474, 1953.

[66] Margaria, R., Cavagna, G.A., Saiki, H., *Human locomotion at reduced gravity*, Proceedings of the 2nd Lunar International Symposium. Life Sciences Research and Lunar Medicine. Pergamon Press, Oxford and New York, 1967.

[67] Mars Global Surveyor Website, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California (http://mars.jpl.nasa.gov/mgs/).

[68] MGS TES Website, Mars Global Surveyor Thermal Emission Spectrometer Website, 2001 (http://tes.la.asu.edu/).

[69] Mohandas, S. U. and Sandgren, E., 1989, *Multiobjective Optimization Dealing with Uncertainty*, Advances in Design Automation - Design Optimization (Ravani, B., ed.), ASME, Vol. 19-2, pp. 241-248, Montreal, Canada, September 17-21, 1989.

[70] Muelberger, W.R., *Apollo 16 Traverse Planning and Field Procedures*, USGS Professional Paper 1048, Part C, United States Government Printing Office, Washington, D.C., 1981.

[71] *NASA Systems Engineering Handbook*, SP-6105, National Aeronautics and Space Administration, June, 1995.

[72] Newman, Dava J., *Human locomotion and energetics in simulated partial gravity*, Ph.D. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

[73] Newman, D.J. and Alexander, H.L., *Human Locomotion and Workload for Simulated Lunar and Martian Environments*, Acta Astronautica, Vol. 29, No. 8, pp.613-620, 1993.

[74] Nute, R.H., Blevins, R.V., and Koppa, R.J., *Apollo 14 Final Lunar Surface Procedures*, Lunar Surface Operations Office, National Aeronautics and Space Administration, Manned Spacecraft Center, Houston, Texas, December 31, 1970.

[75] Nicogossian, Arnauld E., Huntoon, Carolyn L., and Pool, Sam L., *Space Physiology and Medicine, 3rd Edition*, Lea & Febiger, Malvern, Pennsylvania, 1993.

[76] Oxford English Dictionary, Oxford University Press, 2001 (*http://dictionary.oed.com*).

[77] Portee, D.S.F, and Trevino, R.C., *Walking to Olympus: An EVA Chronology*, NASA History Office, NASA Headquarters, Washington DC, 1997.

[78] Rechtin, Eberhardt, *Systems Architecting*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[79] Robbins, Tim, *Even Cowgirls Get the Blues*, Bantam Books, New York, 1976.

[80] Roberto, V., and Chiaruttini, C., *Modeling and Reasoning Techniques in Geologic Interpretation*, IEEE Transactions of Systems, Man, and Cybernetics, Vol. 29, No. 5, September 1999.

[81] Rover Team (Moore, H.J., point of contact), *Characterization of the Martian Surface Deposits by the Mars Pathfinder Rover, Sojourner*, Science, Vol. 278, 5 December 1997.

[82] Russell, S. and Norvig, P., *Artificial Intelligence, A Modern Approach*, Prentice-Hall, 1995.

[83] Saleh, Joseph H., *Unpublished work*, Cambridge, MA, 2001.

[84] Santee, W.R., Allison W.F., Blanchard, L.A, and Small, M.G., "A Proposed Model for Load Carriage on Sloped Terrain", *Avi-*

ation, Space, and Environmental Medicine, Vol. 72, No. 6, June 2001.

[85] Schoening, J.R.; Christian, J.R., *Soldier's Radio,* Military Communications Conference, 1992. MILCOM '92, Conference Record. Communications - Fusing Command, Control and Intelligence., IEEE, Page(s): 497 -499 vol. 2., 1992.

[86] Shannon, Claude E. and Weaver, Warren, *Mathematical Theory of Communication*, University of Illinois Press, 1963.

[87] Shaw, G.B., *The Generalized Information Network Analysis Methodology for Distributed Satellite Systems*, PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1999.

[88] Sheridan, Thomas B., *Telerobotics, Automation, and Human Supervisory Control*, MIT Press, Cambridge, Massachusetts, 1992.

[89] Sheridan, Thomas B., and Ferrell, William R., *Man-Machine Systems*, MIT Press, Cambridge, Massachusetts, 1974.

[90] Simon, H.A., *Sciences of the Artificial*, MIT Press, Cambridge, Massachusetts, 1981.

[91] Smith, Zuber, Frey, Garvin, Head, Mhleman, Pettengill, Phillips, Solomon, Zwally, Banerdt, Duxbury, Golombek, Lemoine, Neumann, Rowlands, Aharonson, Ford, Ivanov, McGovern, Abshire, Afzal, and Sun, *Mars Orbiter Laser Altimeter (MOLA): Experiment Summary after the First Year of Global Mapping of Mars*, Journal of Geophysical Research, accepted September 16, 2000.

[92] Srinivasan, R. Srini, Leonard, Joel I., and White, Ronald J., *Chapter 26 - Mathematical Modeling of Physiological States*, Space Biology and Medicine, Volume III, Book 2, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1996.

[93] Stone, R.W., *Man's Motor Performance Including Acquisition f Adaptation Effects in Reduced Gravity Environments*, National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia, 1974.

[94] Stuster, Jack, *Bold Endeavors*, Naval Institute Press, Annapolis, Maryland, 1996.

[95] Swann, G.A., Bailey,N.G., Batson, R.M., Eggleton, R.E., Hait, M.H., Holt, H.E., Larson K.B., Reed, V.S., Schaber, G.G., Sutton, R.L., Trask, N.J., Ulrich, G.E., and Wilshire, G.E., *Geol-*

ogy of the Apollo 14 Landing Site in the Fra Mauro Highlands, USGS Professional Paper 880, United States Geological Survey.

[96] Talbot-Stern, J., *Design of an Integrated Mars Communication, Navigation, and Sensing System*, AIAA Paper 2000-0011, 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 10-13, 2000.

[97] Thwaites, R.G., editor, *Original Journals of the Lewis and Clark Expedition 1804-1806*, Volumes 1-7 and Atlas, Antiquarian Press Ltd, New York, 1959.

[98] Tufte, Edward R., *Envisioning Information*, Graphics Press, Cheshire Connecticut, 1990.

[99] Tufte, Edward R., *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, 1983.

[100]Tufte, Edward R., *Visual Explanations*, Graphics Press, Cheshire, Connecticut, 1997.

[101]Tutschku, K., Leibnitz, K., and Tran-Gia, P., *ICEPT - An Integrated Cellular Network Planning Tool*, IEEE, 1997.

[102]Ulrich, G.E., Hodges C.A., and Muehlberger, W.R., *Geology of the Apollo 16 Area, Central Lunar Highlands*, Geological Survey Professional Paper 1048, United States Government Printing Office, Washington D.C., 1981.

[103]United States Geological Survey Digital Raster Graphics (*http://terraserver.microsoft.com*).

[104]Waligora, J.M., *The Use of a Model of Human Thermoregulation During the Apollo and Skylab Programs (N76-11172)*, National Aeronautics and Space Administration, Lyndon B. Johnson Space Center, Houston, Texas, 1976.

[105]Webbon, Bruce, *Human Requirements for Life Support and Extravehicular Activity in Space*, Presentation at the MIT/NASA/Industry EVA and Life Support Workshop, Cambridge, Massachusetts, 1994.

[106]Weiss, Gerhard, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 2000.

[107]Wertz, J.R., and Larson, W.J., *Space Mission Analysis and Design, 2nd Edition*, Kluwer Academic Publishers, Norwell, Massachusetts, 1992.

[108]Wilde, Richard C. *100 Years of EVA - Progress and Challenges*, Presentation at the MIT/NASA/Industry EVA And Life Support Workshop, Cambridge, Massachusetts, 1994.

[109]Wilhelms, D.E., *To a Rocky Moon: A geologist's history of lunar exploration*, University of Arizona Press, Tucson, Arizona, 1993.

[110]Wood, J., *The Geomorphological Characterisation of Digital Elevation Models*, Ph.D. Thesis, Department of Geography, University of Leicester, United Kingdom, 1996 (*http://www.geog.le.ac.uk/jwo/research/dem_char/thesis/index.html*).

[111]Woolridge, Michael, and Jennings, Nick, *Intelligent Agents: Theory and Practice*, Knowledge Engineering Review, Volume 10, No 2, Cambridge University Press, June 1995.

[112]Zubrin, R.M., Baker, D.A., Gwynne, O., *Mars Direct: A Simple, Robust, and Cost Effective Architecture for the Space Exploration Initiative*, AIAA-91-0328, American Institute of Aeronautics and Astronautics, 1991.