Terrain Relative Localization for Lunar Entry, Descent, and Landing

by

Matthew J. Hale

B.S., Astronautical Engineering
United States Air Force Academy, 2005

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

JUNE 2007

Author ....
Department of Aeronautics and Astronautics
25 May 2007

Certified by..
Dr. Richard W. Madison
Senior Member of the Technical Staff, Charles Stark Draper Laboratory
Thesis Supervisor

Certified by..
Dr. Brent D. Appleby
Lecturer in Aeronautics and Astronautics
Tactical ISR Division Leader, Charles Stark Draper Laboratory
Thesis Advisor

Accepted by ....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

[This page intentionally left blank.]

Terrain Relative Localization for Lunar Entry, Descent, and Landing

by

Matthew J. Hale

Submitted to the Department of Aeronautics and Astronautics on 25 May 2007
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

As exploration of the solar system continues, the need for the capability to land a spacecraft very accurately on a planetary body has become apparent. Due to limitations on the achievable accuracy of inertial navigation systems over long distances, relative navigation techniques are better suited to fill this need. Chief among these techniques is terrain relative navigation. Terrain relative navigation uses the surface features of a planetary body and an onboard map of the planetary surface to determine the spacecraft's position and update the inertial navigation system. One of the tasks of terrain relative navigation is terrain relative localization, which entails matching a sensor image of the planetary surface to the stored map of the surface. This correlation allows a position match to be determined, which is used to update the spacecraft's inertial navigation system.

This thesis focuses upon two terrain relative localization techniques and their applicability to lunar entry, descent, and landing. These localization techniques are terrain contour matching (TERCOM) and a crater matching routine. Both methods are tested using simulation environments that mimic expected conditions in lunar navigation. The ability of each algorithm to generate a position match with no noise is evaluated, as well as the performance of each algorithm under various sensor noise conditions.

The terrain contour matching algorithm generates a high level of error in the position match and is found to be unsuitable for lunar terrain relative navigation. The crater matching routine performs quite well, with low processing speeds, moderate memory requirements, and a high level of position match fidelity under moderate noise conditions. The crater matching routine is recommended for continued work and potential application to lunar navigation during entry, descent, and landing.

Thesis Supervisor:	Dr. Richard W. Madison
Title:	Senior Member of the Technical Staff, Charles Stark Draper Laboratory
Thesis Advisor:	Dr. Brent D. Appleby
Title:	Lecturer in Aeronautics and Astronautics Tactical ISR Division Leader, Charles Stark Draper Laboratory

Acknowledgments

My deep gratitude to all whose help has allowed me to complete this thesis, especially:

To my bride, Carly. It is in your support and encouragement that I find my strength. And it is in your love that my work has value. And to our child, affectionately 'the Little Guy' until November of '07.

To Sam Schweighart, who provided invaluable guidance as my thesis supervisor for many months, and whose enthusiastic support has been crucial in my completion of this project.

To Rich Madison and Brent Appleby, my thesis advisors, whose advice, suggestions, and comments have greatly refined this paper. I thank you for the time and energy you have dedicated to this thesis.

To the many other mentors I have had at Draper whose help has been pivotal in my years here in Boston. And to Piero Miotto, for the opportunity to come to Draper as a research fellow and to study at the Institute for my degree.

To those instructors whose passion has kindled my thirst for academic knowledge. Especially Dr. Paul "Dr. Evil" Vergez of the United States Air Force Academy and Mrs. Melinda Michaels of St. Xavier High School.

To my homebrew crew. Without you guys and your 'help', those late thesis nights would have found me much less motivated to burn the midnight oil.

And to my Lord, for the opportunity to pursue Him in my work during these years.

TABLE OF CONTENTS

ABSTRACT **3**

ACKNOWLEDGMENTS **5**

RESEARCH PROJECT SPONSORSHIP .... ERROR! BOOKMARK NOT DEFINED.

ASSIGNMENT .... ERROR! BOOKMARK NOT DEFINED.

TABLE OF CONTENTS **7**

LIST OF FIGURES **15**

LIST OF TABLES **21**

CHAPTER 1 INTRODUCTION **23**

CHAPTER 2 TERRAIN CONTOUR MATCHING **30**

2.1 The TERCOM Algorithm **32**

2.1.1 Algorithm Overview 33

2.1.2 The TERCOM Process 34

2.1.2.1 The Long Sample Short Matrix MAD Algorithm....37

2.1.2.2 The Short Sample Long Matrix MAD Algorithm....39

2.1.2.3 MAD Algorithm Notes....41

2.1.3 The VAR Correlation Algorithm....42

2.1.3.1 The Long Sample Short Matrix VAR Algorithm....44

2.1.3.2 The Short Sample Long Matrix VAR Algorithm....46

2.1.3.3 VAR Algorithm Notes....48

2.2 The CraterMaker and CraterMap Utilities....48

2.2.1 The CraterMaker Tool....49

2.2.2 The CraterMap Tool....51

2.3 Analysis Assumptions & Description....52

2.3.1 Setup of the Analysis....53

2.3.1.1 Simulating the Reference Map....56

2.3.1.2 Reference Map Characteristics....57

2.4 Analysis Results & Conclusions....60

CHAPTER 3 PRELIMINARIES TO THE CRATER MATCHING ROUTINE....67

3.1 Utility of Crater Matching in Terrain Relative Navigation....68

3.1.1 Vision Based Operations for Terrain Relative Localization....69

3.1.2 Crater Matching and Lunar Vision Based Terrain Localization....70

3.2 Crater Matching Problem Statement and Characterization....73

3.2.1 Problem Statement....73

3.2.1.1 Requirements....75

3.2.1.2 Definition of Success....77

3.2.2 Star Tracking Algorithms & the Crater Matching Problem....77

3.2.2.1 The 'Lost In Space' Problem 78

3.2.2.2 Star Tracker Algorithms for Crater Matching 78

3.2.2.2.1 The Geometric Approach 79

3.2.2.2.2 Pattern-Based Approach 80

3.2.2.2.3 Comparative Cost-Benefit Assessment 81

3.2.2.3 Star Tracker Algorithm Selected for Application 82

3.2.3 Initialization of the Crater Matching Routine 82

3.2.3.1 Terminology 82

3.2.3.2 Unit Systems 83

3.2.3.3 Unit Conversions 86

3.2.3.3.1 Sensor Input Image Plane Units $(q \quad r)$ to Pixelated Image Plane Units $(m \quad n)$ 86

3.2.3.3.2 Pixelated Image Plane Units $(m \quad n)$ to True Image Plane Units $(u \quad v)$ 86

3.2.3.4 Inputs, Outputs, & Algorithm Constants 87

3.2.3.5 Assumptions 88

3.2.3.5.1 Camera is an ideal camera 88

3.2.3.5.2 Craters are circular 89

3.2.3.6 Algorithm Notes 89

CHAPTER 4 THE PATTERN VECTOR AND THE CRATER MATCHING ROUTINE 93

4.1 Establish the Translational Reference 97

4.1.1 Euclidean Translation 98

4.1.2 Perspective Transformation 102

4.1.3 Data Point Translation Example 104

4.2 Establish the Rotational Reference 106

4.2.1 Reference Point Selection 109

4.2.2 Rotation of Data Points 110

4.2.3 Data Point Rotation Example 112

4.2.3.1 Select the closest neighbor. 113

4.2.3.2 Rotate the Data Points. 114

4.3 Generate the Grid-Based Pattern Vector 115

4.3.1 Derivation of a Pattern from the Image 115

4.3.2 Pattern Vector Format 118

4.3.3 Sparsity Considerations 120

CHAPTER 5 THE PERSPECTIVE TRANSFORM AND THE CRATER MATCHING ROUTINE 125

5.1 Problem Statement and Characterization 126

5.1.1 Problem Statement 126

5.1.2 Coordinate Frame Definitions 129

5.1.3 Unit Systems 131

5.2 Initialization of the Perspective Transform Algorithm 132

5.2.1 Terminology 133

5.2.2 Inputs, Outputs, & Algorithm Constants 133

5.2.3 Assumptions 134

5.2.3.1 Moon is spherical 134

5.2.3.2 Camera is nadir-pointing. 135

5.2.3.3 Spacecraft altitude does not change over transform. 135

5.2.3.4 Image plane is unit distance from camera. 136

5.2.3.5 The shape of any given crater is constant over the transform. 136

5.2.3.6 Craters are circular. 136

5.2.3.7 Camera is an ideal camera 137

5.2.4 Algorithm Notes 137

5.3 The Perspective Transform Rotation Matrix Calculation 137

5.3.1 Rotation Matrix Calculation Initialization 138

5.3.2 Rotation Angle Derivation 139

5.3.3 Rotation Axis Derivation 142

5.3.4 Rotation Matrix Calculation 143

5.4 The Perspective Transform Translation Vector Calculation 145

5.4.1 Translation Vector Calculation Initialization 146

5.4.2 Translation Magnitude Derivation 147

5.4.3 Translation Direction Derivation 148

5.4.4 Translation Vector Calculation 150

5.5 The Perspective Transform Point-wise Depth Calculation 151

5.5.1 Point-wise Depth Calculation Initialization 151

5.5.2 Point-wise Depth Derivation 152

5.6 The Perspective Transform Equation 153

5.6.1 Perspective Transform Initialization 155

5.6.2 Perspective Transform Derivation 156

5.6.2.1 Generalized World Frame to Camera Frame Mapping 156

5.6.2.2 Application of Generalized Mapping to Perspective Transform Camera Model and Coordinate Frames 157

5.6.2.3 Application of Generalized Mapping to this Perspective Transform 159

5.6.3 The Perspective Transform Equation for this Problem 161

5.7 Perspective Transform Implementation and Results 163

5.7.1 Analysis Initialization 164

5.7.2 Accuracy of Centering the Desired Boresight 166

5.7.3 Image Translation vs. Image Transformation 169

5.7.4 Change in the Normalized Distances between Crater Centers 172

5.7.5 Coding of the Perspective Transform Algorithm 179

CHAPTER 6 THE MAIN PROCESSES OF THE CRATER MATCHING ROUTINE 183

6.1 The Sensor Image Signature 184

6.1.1 Signature Generation Process 185

6.1.2 Example Signature Formation 189

6.2 The Crater Database 195

6.2.1 Generation of the Database 195

6.2.2 The Look-Up Table 198

6.2.2.1 Storing the Data 198

6.2.2.2 Advantages & Disadvantages 200

6.2.2.3 Utilizing Sparsity 201

6.2.3 Database Characteristics 203

6.3 The Correlation Algorithm 205

CHAPTER 7 IMPLEMENTATION AND ANALYSIS OF THE CRATER MATCHING ROUTINE 211

7.1 Coding of the Crater Matching Routine 212

7.2 Crater Matching Routine Analysis Initialization 216

7.3 Crater Matching Routine Analysis Results 219

7.3.1 Inconclusive Algorithm Results: Causes for No Match Possible 219

7.3.2 Failure of the Algorithm: Incorrect Position Correlation 225

7.3.3 Processing Speed Analysis 227

7.3.4 Error Analysis 235

7.4 Crater Matching Routine Conclusions 248

CHAPTER 8 CONCLUSIONS 253

8.1 Summary of Results 254

8.1.1 TERCOM 254

8.1.2 The Crater Matching Routine 256

8.2 Observations 256

8.2.1 TERCOM 256

8.2.2 The Crater Matching Routine 256

8.3 Recommendations for Future Work 256

8.3.1 TERCOM 256

8.3.2 The Crater Matching Routine 256

8.3.2.1 Required Improvements for Operability 256

8.3.2.2 Translation of the Routine into Operational Code 256

8.3.2.3 Possible Modifications to the Routine 256

APPENDIX A: PERSPECTIVE TRANSFORM PROOFS 256

APPENDIX B: ABBREVIATIONS AND ACRONYMS 256

APPENDIX C: MATLAB CODE FOR THE CRATER MATCHING ROUTINE 256

REFERENCES 256

LIST OF FIGURES

Figure 1.1. Process Overview of Vision-Based Terrain Relative Localization. 28

Figure 2.1. Overview of the Calculation of Terrain Profiles. 34

Figure 2.2. An Example of the Use of TERCOM for Position Fix During Missile Flight. 35

Figure 2.3. TERCOM Correlation Process Overview..36

Figure 2.4. The Long Sample Short Matrix (LSSM) and Short Sample Long Matrix (SSLM) TERCOM Scenarios. 37

Figure 2.5. An Example of an LSSM Correlation Attempt. 39

Figure 2.6. An Example of an SSLM Correlation Attempt. 41

Figure 2.7. Description of the VAR Correlation Algorithm. 43

Figure 2.8. An Example Position Fix Using the VAR Algorithm. 46

Figure 2.9. An Example Terrain Output of the CraterMap Utility..52

Figure 2.10. Overview of the Generation of Simulated Terrain Profiles for TERCOM Analysis............54

Figure 2.11. Overview of the Correlation of Simulated Terrain Profiles to the Reference Map for TERCOM Analysis. 55

Figure 2.12. An Example Reference Map with 100m Cells for TERCOM Analysis. 57

Figure 2.13. Expected Error Trends for the TERCOM Reference Maps..60

Figure 2.14. TERCOM LSSM VAR Analysis: 0.3 craters/km^2 with standard dev. of 0.1 craters/km^2. ...61

Figure 2.15. TERCOM LSSM VAR Analysis: 0.6 craters/km^2 with standard dev. of 0.1 craters/km^2. ...62

Figure 2.16. TERCOM LSSM VAR Analysis: 0.3 craters/km^2 with standard dev. of 0.1 craters/km^2, Forced Large Craters of [1.0 1.0 1.5] km Radius..62

Figure 2.17. TERCOM LSSM VAR Analysis: 0.6 craters/km^2 with standard dev. of 0.1 craters/km^2, Forced Large Craters of [1.0 1.0 1.5] km Radius..63

Figure 2.18. TERCOM LSSM VAR Analysis: 0.3 craters/km^2 with standard dev. of 0.1 craters/km^2, Forced Large Craters of 2.5 km Radius. 63

Figure 2.19. TERCOM LSSM VAR Analysis: 0.6 craters/km^2 with standard dev. of 0.1 craters/km^2, Forced Large Craters of 2.5 km Radius. 64

Figure 3.1. Reprint of the Process Overview of Vision-Based Terrain Relative Localization. 71

Figure 3.2. An Overview of the Interactions Between the Crater Matching Routine Processes....75

Figure 3.3. The Unit Systems of the Image Plane....83

Figure 3.4. A Detail of the UV (True) Image Plane....85

Figure 3.5. Example Data Points for Pattern Vector Generation. 89

Figure 4.1. Translational and Rotational Invariance. 95

Figure 4.2. Overview of the Pattern Vector Generation Process. 97

Figure 4.3. Euclidean Translation: An Example. 98

Figure 4.4. Euclidean Translation: An Explanation of Its Usefulness. 100

Figure 4.5. Euclidean Translation: An Explanation of Its Usefulness cont'd. 101

Figure 4.6. Overview of the Perspective Transform. 103

Figure 4.7. Euclidean Translation of the Example Data Points. 104

Figure 4.8. Perspective Transform of the Example Data Points. 105

Figure 4.9. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points....106

Figure 4.10. Set-up of the Rotational Reference Brain Teaser Example. 107

Figure 4.11. Reoriented Rotational Reference Brain Teaser Example. 108

Figure 4.12. Establishing the Rotational Reference. 111

Figure 4.13. Establishing the Rotational Reference: Closest Neighbor Selection for the Example Data Points....113

Figure 4.14. Establishing the Rotational Reference: Rotation of the Example Data Points. 114

Figure 4.15. Pattern Vector Generation for the Example Data Points. 117

Figure 5.1. Reprinted Overview of the Perspective Transform. 127

Figure 5.2. Overview of the Processes of the Perspective Transform. 128

Figure 5.3. Definition of the Coordinate Frames Used for the Perspective Transform Calculations. 130

Figure 5.4. Reprinted Unit Systems of the Image Plane. 132

Figure 5.5. Topography of the Lunar Surface. 135

Figure 5.6. Image Rotation: Alpha Angle Calculation 140

Figure 5.7. Image Rotation: Rotation Angle Calculation. 141

Figure 5.8. Image Rotation: Rotation Axis Calculation. 143

Figure 5.9. Image Rotation: The Sign of the Rotation Axis 145

Figure 5.10. Translation Magnitude Calculation. 147

Figure 5.11. Translation Direction Calculation. 149

Figure 5.12. Translation Vector Calculation. 150

Figure 5.13. Calculation of the Point-wise Depth. 153

Figure 5.14. Reprinted Overview of the Perspective Transform. 155

Figure 5.15. Reprint Example Data Points for Pattern Vector Generation. 165

Figure 5.16. Example Data Points with Potential Boresight Craters Highlighted. 166

Figure 5.17. Euclidean Translation of the Example Data Points. 167

Figure 5.18. Perspective Transformation of the Example Data Points. 168

Figure 5.19. Reprint Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points. 170

Figure 5.20. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points for Boresight Crater # 2 170

Figure 5.21. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points for Boresight Crater # 3 171

Figure 5.22. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points for Boresight Crater # 4 171

Figure 5.23. Characteristics of the Distances Between Crater Centers for Boresight Crater #1 176

Figure 5.24. Characteristics of the Distances Between Crater Centers for Boresight Crater #2 176

Figure 5.25. Characteristics of the Distances Between Crater Centers for Boresight Crater #3............177

Figure 5.26. Characteristics of the Distances Between Crater Centers for Boresight Crater #4............177

Figure 6.1. Reprint Overview of the Pattern Vector Generation Process 186

Figure 6.2. Overview of the Signature Generation Process. 189

Figure 6.3. Reprint Example Data Points with Potential Boresight Craters Highlighted. 190

Figure 6.4. Signature Generation for Example Data Point Boresight #1. 191

Figure 6.5. Signature Generation for Example Data Point Boresight #2. 192

Figure 6.6. Signature Generation for Example Data Point Boresight #3. 193

Figure 6.7. Signature Generation for Example Data Point Boresight #4. 194

Figure 6.8. Overview of the Database Generation Process..198

Figure 6.9. Database Compression Using the Circular Nature of the Pattern in the Grid Matrix...........202

Figure 6.10. Row Size Analysis of the Look-Up Table. 204

Figure 6.11. Overview of the Crater Matching Process. 206

Figure 7.1. Reprint Overview of the Interactions Between the Crater Matching Routine Processes. 213

Figure 7.2. Processing Time vs. the Number of Potential Boresight Craters in the Sensor Image.........229

Figure 7.3. Processing Time vs. the Number of Total Craters in the Sensor Image. 231

Figure 7.4. Number of Total Craters vs. Number of Potential Boresight Craters...................................233

Figure 7.5. Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels........................237

Figure 7.6. Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels, Detail #1.......237

Figure 7.7. Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels, Detail #2.......238

Figure 7.8. Analysis of the Routine: Response of the Routine to Sensor Image Noise. 239

Figure 7.9. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Detail. 240

Figure 7.10. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Position Match Success and Failure...241

Figure 7.11. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Signature Generation Failure..242

Figure 7.12. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Signature Generation Success with Component Failures 243

Figure 7.13. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Inconclusive Correlation Attempts 244

Figure 7.14. Analysis of the Routine: Response of the Routine to Sensor Image Noise, "E2 & E3 Threshold Not Met" Attempts 245

Figure 7.15. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Signature Matching Success with Component PV Matching Failures 246

Figure 8.1. Reprint TERCOM LSSM VAR Analysis: 0.6 craters/km^2 with standard dev. of 0.1 craters/km^2, Forced Large Craters of [1.0 1.0 1.5] km Radius 256

Figure 8.2. Reprint TERCOM LSSM VAR Analysis: 0.6 craters/km2 with standard dev. of 0.1 craters/km2 256

Figure 8.3. Reprint Analysis of the Routine: Response of the Routine to Sensor Image Noise, Position Match Success and Failure 256

Figure 8.4. Reprint Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels (detail). 256

Figure A-1. Proof that the Normalized Distance Between Two Boresight Vectors is Constant Over the Transform 256

Figure A-2. Demonstration that the Normalized Distance Between Any Two Points in the Image Plane is not Constant Over the Transform 256

Figure A-3. Demonstration that the Normalized Distance Between Any Two Points in the Image Plane is not Constant Over the Transform, Detail 256

LIST OF TABLES

Table 3-1. An Overview of the Costs and Benefits of Star Tracker Algorithms 81

Table 4-1. Grid Matrix Entries for the Example Data........ 123

Table 5-1. Comparison of Euclidean Translation and Perspective Transform Error........ 169

Table 5-2. Distance between the Translated and Transformed Crater Centers........ 172

Table 5-3. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #1. 173

Table 5-4. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #2. 174

Table 5-5. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #3. 174

Table 5-6. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #4. 175

Table 5-7. Change in Position of Crater Centers Relative to All Other Craters Over the Transform........ 179

Table 7-1. General Success / Failure Statistics from the Analysis of the Routine........ 222

Table 7-2. Reasons and Frequencies the Routine Failed to Generate a Position Match........ 223

Table 7-3. Reasons and Frequencies the Routine Generated a Signature but a Component Patter Vector Generation Failed........ 224

Table 7-4. Reasons and Frequencies the Routine Successfully Determined a Position Match but a Component Patter Vector Matched Incorrectly. 224

Table 7-5. Correct and Incorrect Position Matches........ 227

Table 7-6. Processing Time vs. the Number of Potential Boresight Craters in the Sensor Image........ 230

Table 7-7. Processing Time vs. the Number of Total Craters in the Sensor Image........ 232

Table 7-8. Number of Total Craters vs. Number of Potential Boresight Craters........ 234

Table 8-1. Reprint General Success / Failure Statistics from the Analysis of the Routine....................256

Table 8-2. Reprint Processing Time vs. the Number of Potential Boresight Craters in the Sensor Image. 256

Table 8-3. Reprint Processing Time vs. the Number of Total Craters in the Sensor Image...................256

Table B-1. Acronyms....................256

Table B-2. Abbreviations for the Crater Matching Routine Analysis....................256

Chapter 1

Introduction

As planetary exploration continues to become more accurate and to demand more detailed scientific data, the need for the capability to land precisely has been recognized by the space community. In the past, planetary landing has been imprecise at best, usually involving large landing error dispersion ellipses which encompassed tens of square kilometers of the planetary surface. More accurate autonomous exploration, as well as the potential for manned planetary exploration, demands landing precision on the order of ten meters.

In conjunction with the directives of the 2004 Vision for Space Exploration,[1] the current planetary exploration aims of the National Aeronautics and Space Administration (NASA) are focused upon returning man to the Moon. In support of this goal, NASA has broadcast that several new

[1] A set of priorities developed by then President George W. Bush and NASA which was announced in 2004, providing a roadmap for NASA. The main goals of the VSE are to return man to the Moon in preparation for manned exploration of Mars and other planetary bodies in the solar system.

technological capabilities will be necessary in the near future. Among these is the capability for precision lunar landing. NASA has requested the capability to precisely land a spacecraft on the lunar surface, and to do so with a landing dispersion ellipse with maximum dimensions of $10\ m$ by $10\ m$, with a preferred accuracy of $1\ m$ by $1\ m$.

This is a tremendous increase in desired landing accuracy. Consider a few historical examples. In 1969, the Apollo 11 lunar landing had a landing error dispersion ellipse of $19\ km$ by $5\ km$.[1] Also in 1969, Apollo 12's landing error dispersion ellipse was somewhat less, at approximately $13\ km$ by $5\ km$.[2] Even modern spacecraft attempting planetary landing have landing dispersion ellipses quite drastically larger than that desired by NASA. For example, the Mars Science Laboratory, scheduled for launch in 2009, will have a dispersion ellipse similar in size to that of Apollo 11, with a $20\ km$ diameter.[3]

These three landing dispersion ellipses are not atypical for a modern planetary landing. However, NASA has requested a precision landing capability that is many times more accurate than what is currently the norm – which is attested to by the fact that the desired error is measured in meters rather than kilometers. Consider that a $20\ km$ by $5\ km$ dispersion ellipse has an area of approximately $78.5\ km^2$. On the other hand, the dispersion ellipse desired by NASA has an area of $1\ m^2$. *That is, the landing accuracy desired by NASA is approximately six billion* $(6.2 * 10^9)$ *times more accurate than the dispersion ellipses of Apollo-type missions!*

To realize this drastic improvement in accuracy, seasoned entry, descent, and landing (EDL) techniques for planetary landing are being revised. Rather than relying solely upon inertial guidance and navigation techniques, many precision planetary landing efforts currently focus upon relative navigation techniques as a means to achieve greater landing accuracy.

Inertial navigation and relative navigation are similar in their purpose, but distinct from each other in the means used to achieve that end. Both types of navigation use measurements to update the position knowledge of a spacecraft, but each uses a different type of measurement. Inertial navigation measures the spacecraft's pose in relation to the inertial frame[i], and updates the guidance computers with this measurement. On the other hand, relative navigation techniques measure the spacecraft's pose in relation to a non-inertial frame, and use these values to update the spacecraft's pose knowledge.[ii]

Consider the situation of a spacecraft navigating from Earth to the Moon. To allow the spacecraft to reach its desired destination, an inertial navigation system attempts to determine the spacecraft's position relative to an expected position on the trajectory. First, sensor input concerning the location of the spacecraft is processed to determine the 'measured position' of the spacecraft. This 'measured position' can then be used to update the inertial guidance system, allowing the guidance system to correct for the effects of drift error and initialization errors. Either inertial measurements or relative measurements can be used to provide a 'measured position' for updates. Possible inertial measurements include accelerometer, gyroscope, and star sensor measurements, to name a few. Examples of relative measurements include those that reference the location of the Earth, the Sun, another planetary body, or even a signal from another spacecraft.

Inertial navigation systems have been applied very successfully to a wide variety of space navigation problems. However, current inertial navigation systems are unable to achieve the accuracy necessary for precision planetary landing, as desired by NASA. Relative navigation techniques have been used much less widely in space navigation, but they have been successful at allowing greater accuracy when they are used. Therefore, engineers attempting to design a system to achieve a more precise landing capability often turn to relative navigation as the most viable guidance option.

---

[i] The inertial frame is an absolute, non-rotating frame. Often, the celestial sphere is considered inertial for space navigation applications.
[ii] The author uses the term 'pose' to indicate both the position and orientation of the spacecraft.

For lunar landing, perhaps the most analyzed relative navigation technique is terrain relative navigation (TRN). TRN can be described as the process of sensing some terrain surface beneath a suspended vehicle, and using these sensor inputs to update the vehicle's knowledge of its own position, orientation, or velocity. Therefore, TRN allows a vehicle to navigate more precisely based upon measurements of the terrain surface beneath the suspended vehicle. In the case of lunar landing, the spacecraft will approach and eventually land upon the lunar surface. Therefore, lunar TRN requires a spacecraft to navigate relative to the lunar surface. During this process, TRN has the potential to enable a much more accurate landing.

TRN can take any of a number of forms; however, all TRN techniques will include the following steps:

1. Obtain measurements of terrain surface using an onboard sensor.
2. Correlate the sensor input to a database containing terrain characteristics for the type of sensor being used.
3. Use the information from the correlation to update the vehicle's navigational computers.

For example, cruise missiles use TRN to increase their accuracy. In order to achieve precision targeting capability, a cruise missile navigates to its target area by 'looking' at the ground beneath it. As the missile senses features of the ground that it is flying over, the missile attempts to correlate these features to an onboard database of terrain features that the missile is expected to encounter. If a correlation is successful, the missile is able to update the knowledge of its own position in relation to the target, thereby increasing the missile's overall targeting accuracy. By relying upon TRN, cruise missiles were able to achieve precision targeting capabilities long before the advent of the global position system (GPS) based targeting.[1]

[1] GPS now allows for highly accurate targeting for many types of munitions, including cruise missiles, without the use of TRN techniques. However, many cruise missiles retain their TRN capabilities although they have GPS

This thesis will focus upon increasing the achievable accuracy of a lunar landing by relying upon TRN techniques in addition to traditional inertial navigation operations. The author will address a very specific problem encountered in applying TRN to lunar landing. As a fundamental requirement for TRN, data from the sensors must be correlated with a stored reference map to allow for a position fix. (Step 2 above). In the case of lunar landing, this entails correlating sensor data of the lunar surface with a specific portion of a stored map of the lunar surface. This correlation process – which is vital to lunar TRN – will be the topic of this thesis.

This paper will investigate two alternative correlation schemes for lunar TRN and attempt to determine the applicability of each of these schemes to lunar EDL missions. The first of these correlation schemes is based upon a process used by early cruise missiles to update the missile's position knowledge. This correlation process uses elevation data gathered by altimeters on the spacecraft and attempts to match this data with a database of elevation data for the lunar surface. This correlation technique is known as terrain contour matching (TERCOM). This algorithm tends to be simple, reliable, and fast, as evidenced by its long tenure on many cruise missiles.[1] These characteristics make the TERCOM algorithm a good option for implementation and testing for its applicability to the lunar environment.

The second correlation scheme that will be addressed uses digital imagery and attempts to match the craters in the sensor images with a database of lunar craters. This correlation technique falls into the category of vision-based navigation (VBN). VBN encompasses a wide field of technologies that employ optical sensors to determine the state of the sensor body. Successful application of VBN technologies has been realized in many fields, including robotics, unmanned aerial vehicles, deep-sea

sensors, as the TRN capability is useful in GPS-deprived environments and because TRN enables the terrain – following capability of cruise missiles.

[1] TERCOM, although modified from its original form, still exists on many US missile platforms. Several of these include the AGM-86B, AGM-129 ACM, and the BGM-109 Tomahawk. Refer to http://www.af.mil/factsheets/ , referenced on 20 May 2007. TERCOM-like systems are also in use on the missile platforms of other countries. Refer to http://en.wikipedia.org/wiki/Tercom , referenced on 24 May 2007.

operations, autonomous spacecraft maneuvers, and a host of others. For planetary EDL, terrain-relative VBN presents the opportunity for high-precision navigation and the possibility of achieving a planetary landing with precision on the order of tens of meters, as desired.

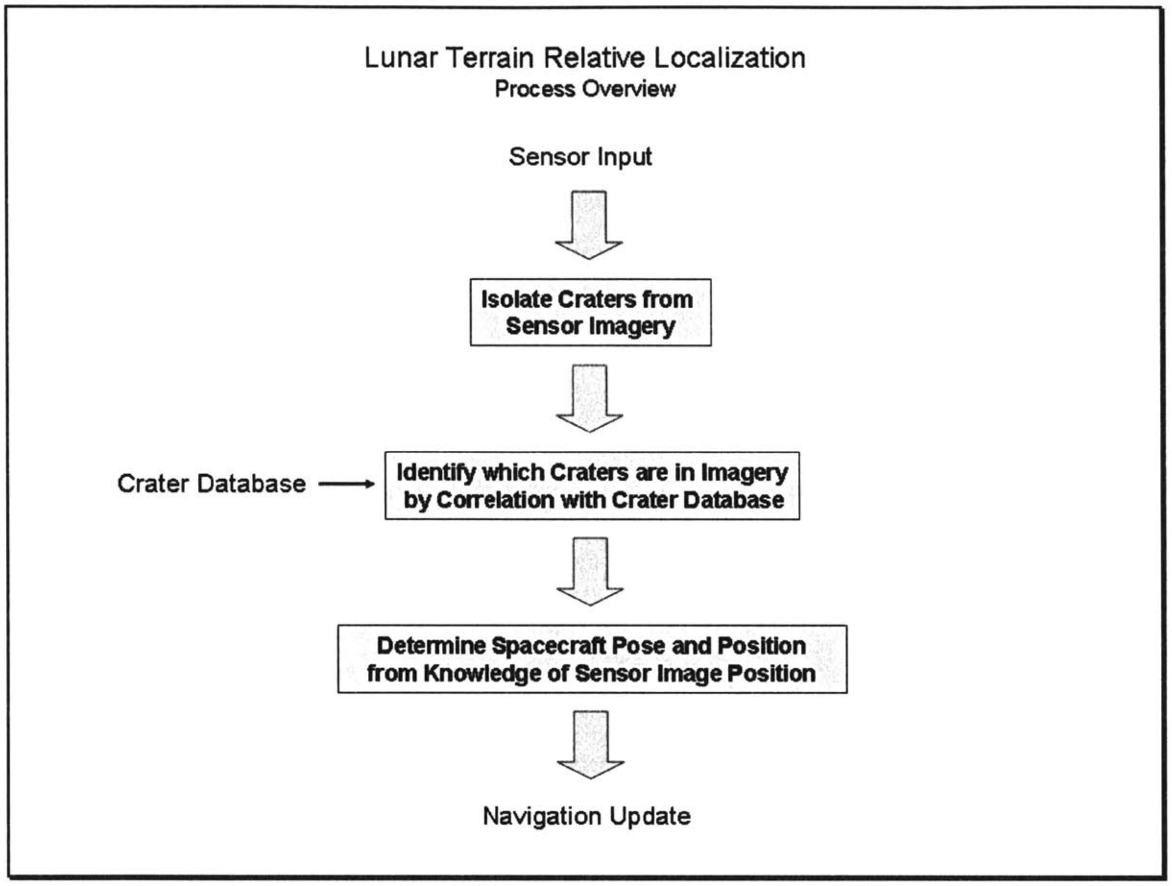

Lunar Terrain Relative Localization
Process Overview
Sensor Input
Isolate Craters from Sensor Imagery
Crater Database
Identify which Craters are in Imagery by Correlation with Crater Database
Determine Spacecraft Pose and Position from Knowledge of Sensor Image Position
Navigation Update

Figure 1.1. Process Overview of Vision-Based Terrain Relative Localization.

Figure 1.1 gives an overview of the stages of lunar VBN. In the second half of this thesis, the author will address the positive identification of optically-sensed craters by correlation with a lunar reference map, which is the second step depicted in Figure 1.1.

For both TERCOM and the crater matching routine, this thesis will only address the process of matching sensor data to a lunar reference database. Both correlation algorithms require sensor input, match the sensor measurements to the stored database, and output the position of the measurements as determined by the matching process.

This paper will not present an extensive analysis of the navigation process; rather, the focus of this paper is upon generating a successful position match. Generating this match is the process of terrain relative localization (TRL). However, the author will use TRN and TRL loosely throughout this document to refer to the process of generating a position update and using that update to aid in spacecraft navigation.

In Chapter 2, this paper will present the TERCOM algorithm and an analysis of its applicability to the lunar EDL environment. Following the TERCOM discussion, Chapter 3, Chapter 4, Chapter 5, and Chapter 6 address the 'crater matching routine' correlation process which is based upon digital imagery. Chapter 3 discusses the basis for the crater matching routine and presents an overview of the processes of the routine. Chapter 4 and Chapter 5 focus upon two sub-routines of the crater matching routine. Chapter 4 details the pattern vector generation process, while Chapter 5 addresses the perspective transformation. Finally, Chapter 6 describes the main processes of the crater matching routine in detail. The implementation and results of the crater matching routine are presented in Chapter 7. Chapter 8 presents an overview of the results of both TERCOM and the crater matching routine, and draws conclusions from both. The author also makes several recommendations for future work in Chapter 8.

Chapter 2

Terrain Contour Matching

Terrain contour matching (TERCOM) is a method of terrain relative navigation (TRN) that attempts to match the profile of the terrain beneath a suspended vehicle with a database of terrain profiles. If a match is made, the position of the vehicle can be determined and used as a navigational update. This chapter details the TERCOM algorithm as well as the simulation environment used to test TERCOM for potential applicability to TRN on the Moon. This chapter will also present the results of analysis conducted using this simulation environment.

TERCOM is one of the earliest and simplest of relative navigation techniques. It has been in development and/or use since 1958, when the concept was first put forth by LTV Electrosystems and the Aeronautical Systems Division at Wright-Patterson AFB as a potential aid to cruise missile guidance systems. Since then, TERCOM has been in use in a variety of capacities, among which its long tenure as a vital cruise missile component is most notable.

Systems that are based upon the TERCOM concept are in use today in both commercial and military applications. For example, Honeywell recently developed an Interferometric Synthetic Aperture Radar Sensor with Integrated Map Correlation system called the Precision Terrain Aided Navigation (PTAN) system. PTAN is currently used on the Tactical Tomahawk Cruise Missile. Since 2003, it has also been marketed as a low-cost autonomous navigation solution in GPS-deprived environments which boasts GPS quality accuracy.[1] The TERCOM concept has also been extensively and successfully applied to the underwater navigation of submarines, among other areas.[4]

From Chapter 1, TRN can be described as the process of sensing some terrain surface beneath a suspended vehicle and using these sensor inputs to update the vehicle's knowledge of its own position, orientation, or velocity. The following steps are essential to TRN:

1. Obtain measurements of terrain surface using an onboard surface.
2. Correlate the sensor input to a database containing terrain characteristics for the type of sensor being used.
3. Use the information from the correlation to update the vehicle's navigational computers.

The second step – the correlation between the sensor input and the surface – provides information about the vehicle's relative position, orientation, or velocity. The focus of the TERCOM algorithm is to match elevation profiles recorded by an onboard altimeter with a database of terrain elevation profiles that corresponded to the surface beneath the vehicle. After a successful match, the vehicle's position can be determined and is fed to the navigational computers to allow for update.

More rigorously, TERCOM operates by using air-to-ground level (AGL) and mean sea level (MSL) altitude measurements in order to obtain a profile of the overflown terrain, and then attempts to correlate this profile with the columns of an onboard 'reference matrix' of elevation data. This correlation enables TERCOM to determine cross-track (lateral) and down-track (translational)

[1] 100 foot accuracy at altitudes of up to 30,000 feet; 10 foot accuracy at altitudes of up to 5,000 feet.

position knowledge error, which is then used to update the INS and make navigational corrections. The TERCOM algorithm requires foreknowledge of the terrain data that is overflown to form the reference map.

Sections as presented below rely heavily upon an internal document produced at Draper Laboratory that explores the TERCOM concept as a potentially viable option for lunar entry, descent, and landing.[1] The author is also the first author on the internal document, and a great deal of the algorithm development and implementation in said document are the result of the author's efforts. The lunar surface simulator and the actual analyses were conducted by other Draper Laboratory staff who are listed as the other authors of the internal document. This document is included here with permission of the authors that provided significant contributions to the original internal document.

The remainder of this chapter discusses the feasibility of applying TERCOM to lunar navigation. This discussion will include a description of the TERCOM process will be presented, followed by an overview of the utilities used to simulate terrain on the lunar surface, the analysis of TERCOM's applicability to navigation on the Moon, and finally results and conclusions drawn from these analyses.

2.1 The TERCOM Algorithm

The description of TERCOM outlined below is summarized from Baker and Clem's "A Terrain Contour Matching (TERCOM) Primer."[5] For the algorithm description included herein, the author

[1] Hale, M., Silvestro, M., Slagowski, S., and Wasileski, B. "Analysis of the Use of Terrain Contour Matching (TERCOM) For Navigation In The Lunar Environment." The Charles Stark Draper Laboratory, Internal Research and Design Technical Paper. August 2006.

will present the basic TERCOM process as described by Baker and Clem. After this description, the author will discuss the modifications made to this algorithm to facilitate lunar navigation.

TERCOM is a form of TRN that was first developed to aid the inertial navigation systems of cruise missiles during flight to a target. Like GPS, TERCOM is able to provide a position update capability in order to correct errors in the inertial navigation system of the spacecraft. There are two distinct advantages of TERCOM over GPS; it cannot be jammed and it does not depend on existing navigation infrastructure to operate. This second advantage is important for lunar navigation since GPS cannot be used on the Moon. Other references found in literature have referred to TERCOM as the Terrain Aided Inertial Navigation System or TERCOM Aided Inertial Navigation System (TAINS).

2.1.1 Algorithm Overview

The TERCOM algorithm is based on a relatively simple concept. Using various sensors, the spacecraft is able to determine the terrain profile[1] of the terrain beneath the suspended vehicle. By comparing the surface elevation features measured beneath the vehicle with similar elevation data stored onboard the vehicle, the position of the measured data can be determined. This position can then be used to update the inertial guidance systems of the vehicle.

In order to compare the terrain profile of the terrain beneath the vehicle, an accurate way of determining the terrain profile is necessary. Two separate sensors are traditionally used to generate the terrain profiles for use by TERCOM. A barometric altimeter is used to obtain the MSL altitude of the vehicle, and a radar altimeter is used to obtain the AGL altitude. The subtraction of the AGL from the vehicle's MSL results in the terrain profile altitude. Refer to Figure 2.1.

[1] A terrain profile is the contour of the surface overflown by the vehicle, reference to mean-sea-level.

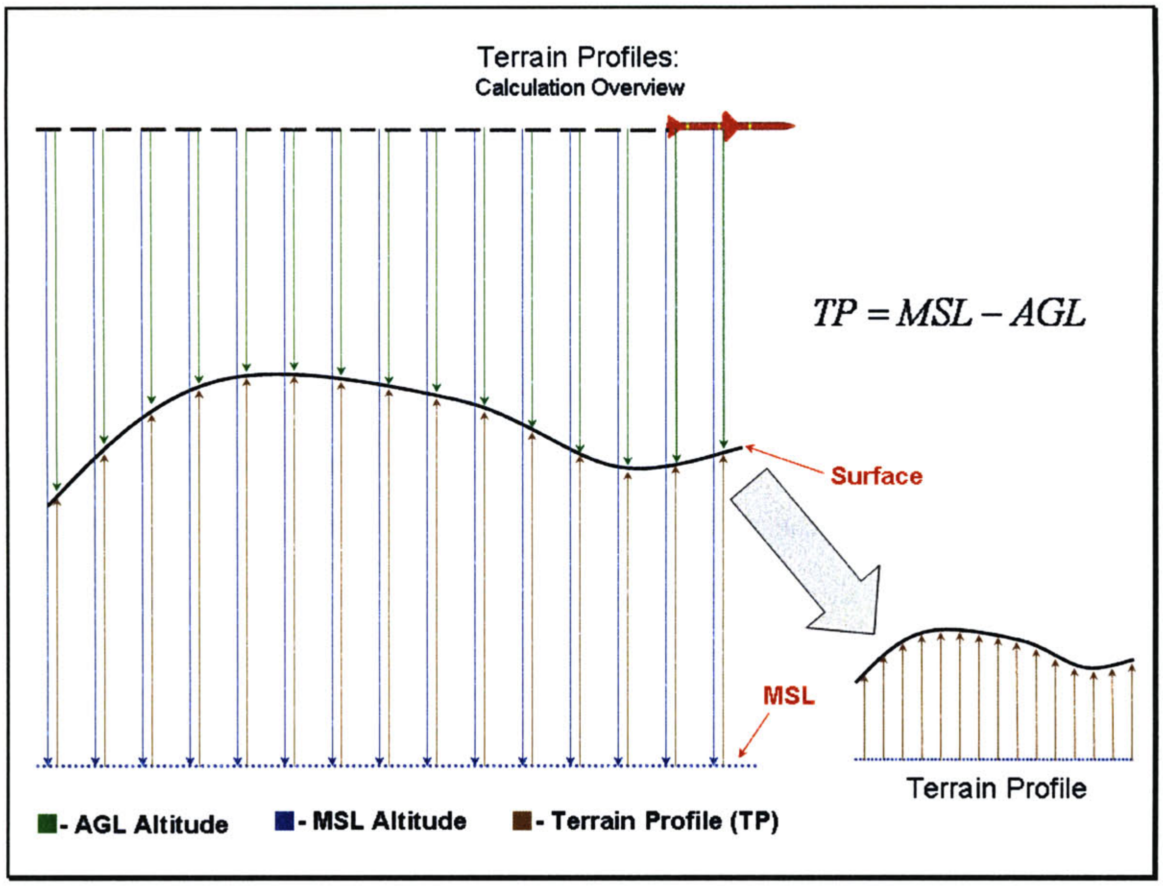


Figure 2.1. Overview of the Calculation of Terrain Profiles.

From Figure 2.1, the terrain profile is a series of these terrain profile altitudes that are produced by several measurements from the suspended vehicle. Therefore, the terrain profile is essentially a column of data that specifies the height of the terrain above mean sea level.

2.1.2 The TERCOM Process

The first step in the TERCOM process is the generation of a database of elevation maps of the surface to be overflown. These maps will be used in the correlation process to determine the vehicle's absolute position. However, the database size can be prohibitive (especially for operations in space) due to the memory space required to store elevation data for the entire overflown surface. Therefore, certain areas along the planned trajectory of the vehicle will be selected prior to flight as position 'fix areas'. Using its INS, the vehicle will travel between these 'fix areas.' After flight over one of these 'fix areas', the INS will be updated with a position fix generated from the TERCOM process. This update will reduce INS error and allow for more precise navigation capabilities.

Maps of the pre-planned fix areas are generated prior to flight and stored on board the vehicle. These maps are stored as reference matrices, with each matrix defining the terrain profiles for a fix area. Figure 2.2 below shows a missile's planned trajectory and the fix areas along the missile's route.

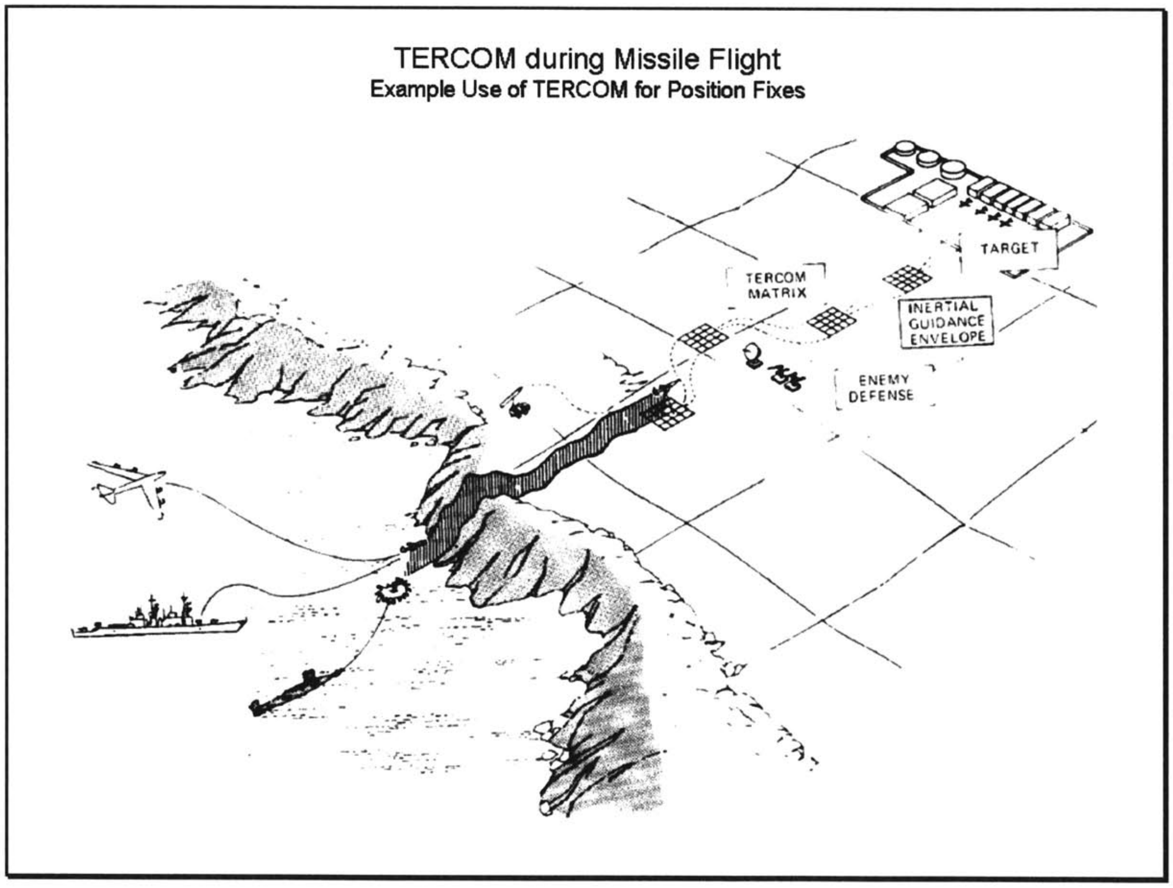

TERCOM during Missile Flight
Example Use of TERCOM for Position Fixes
TARGET
TERCOM MATRIX
INERTIAL GUIDANCE ENVELOPE
ENEMY DEFENSE

Figure 2.2[6]. An Example of the Use of TERCOM for Position Fix During Missile Flight.

When the vehicle flies over a TERCOM fix area, it measures its altitude above the terrain using an altimeter (such as a radar, LIDAR, etc.). This altitude is referred to as the AGL. In addition to the AGL, the vehicle also measures its MSL altitude with a barometric altimeter. The difference between these two values produces a profile of the terrain height above MSL that the vehicle has flown over.

This profile is then compared to the stored reference matrix using a correlation algorithm. Figure 2.3 below shows a graphical representation of the TERCOM process.

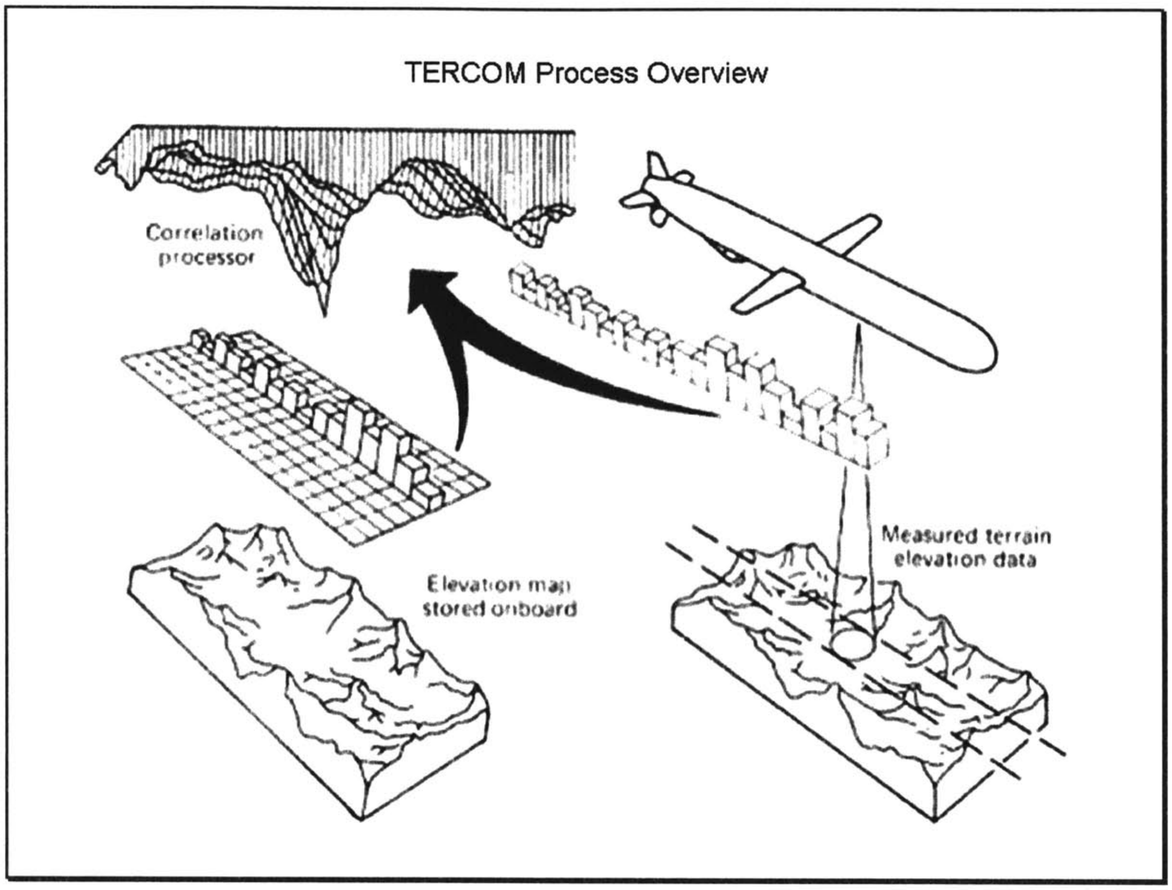


Figure 2.3[7]. TERCOM Correlation Process Overview.

The primary task of the TERCOM algorithm is to correlate the terrain profiles from a 'fix area' with the database of terrain profiles. This correlation must process the data quickly and allow for a position fix with a high level of confidence.

An example of the simple correlation algorithm used in the original TERCOM process is the Mean Absolute Difference (MAD) algorithm. There are two primary MAD scenarios that might be encountered which involve the relative size of the terrain profile and the reference matrix. The first of these scenarios is where the terrain profile has more elements than the reference matrix. This situation is referred to as the Long Sample Short Matrix (LSSM) scenario. The second is where the terrain profile has fewer elements than the reference matrix. This is referred to as the Short Sample Long Matrix (SSLM) scenario. Refer to Figure 2.4.

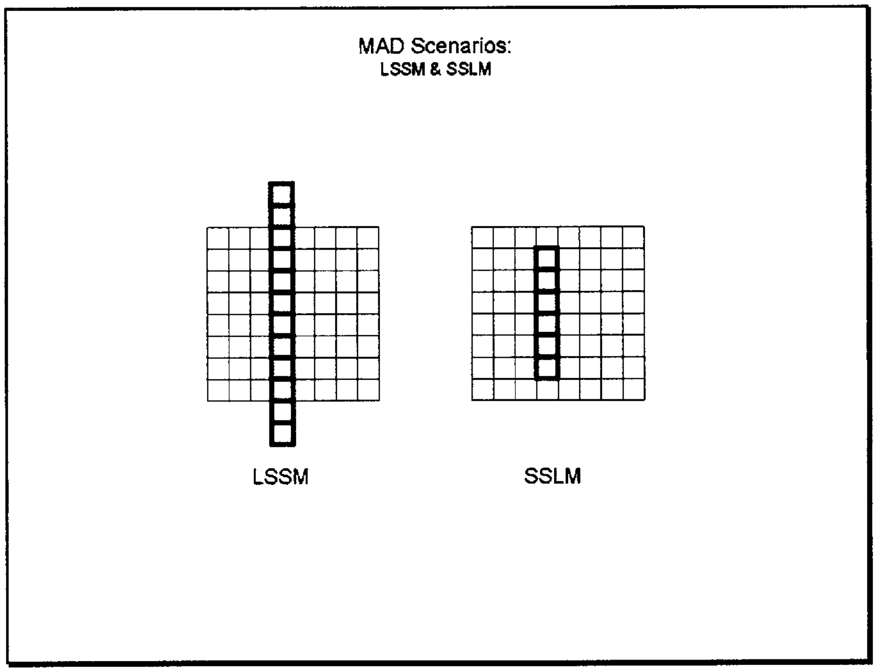


Figure 2.4. The Long Sample Short Matrix (LSSM) and Short Sample Long Matrix (SSLM) TERCOM Scenarios.

The MAD algorithm will be present for each of these cases.

2.1.2.1 The Long Sample Short Matrix MAD Algorithm

The LSSM technique begins sampling at an estimated time prior to the expected arrival of the vehicle over the fix area and the number of samples is greater than the number of rows in the reference matrix. The main calculation of the MAD algorithm for the LSSM case is:

$$MAD_{k,n} = \frac{1}{M}\sum_{m=1}^{M}\left|h_{k+m} - H_{m,n}\right|$$

Equation 2.1

where:

1. $H_{m,n}$ is the $(m,n)^{th}$ element of the pre-stored $M \;\; x \;\; N$ reference matrix
2. K is the difference between the number of rows in the terrain profile and the number of rows in the reference matrix[1]. Given $k \in [0 \quad K]$, this will allow one dimension of variation between the terrain profile and the reference matrix.
3. h_{k+m} is the $(k+m)^{th}$ measured terrain elevation data point
4. $MAD_{k,m}$ is the mean absolute difference between the $(k+m)^{th}$ terrain elevation data point and the $(m,n)^{th}$ reference matrix data point

The MAD calculation presented in Equation 2.1 will determine the mean absolute difference between the n^{th} column of the reference matrix and M elements of the terrain profile. However, the object of the MAD calculation is to determine the best fit between the terrain profile and the reference matrix. Therefore, the terrain profile must be iteratively 'slid' over the reference matrix such that the MAD of all the data points in the terrain profile is calculated. In addition, the terrain profile must be 'slid' across the columns of the reference matrix in a similar manner.

In other words, the MAD must be calculated for all possibilities of (k,n) given that $k \in [0 \quad K]$ and $n \in [1 \quad N]$. Refer to Figure 2.5.

[1] Assuming the terrain profile has more rows than the reference matrix. This is the LSSM scenario.

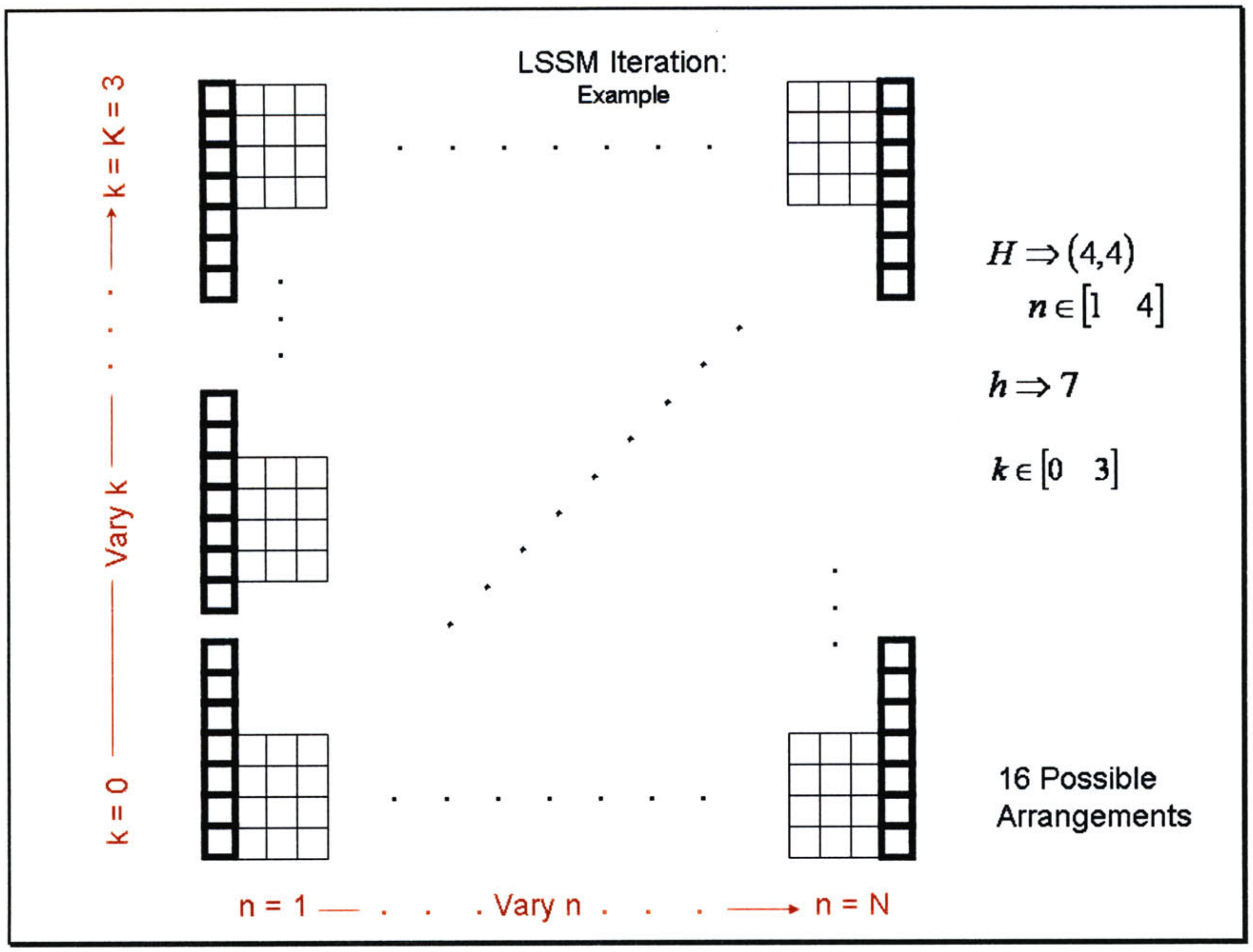


Figure 2.5. An Example of an LSSM Correlation Attempt.

2.1.2.2 The Short Sample Long Matrix MAD Algorithm

The SSLM technique generates a single measurement profile when the vehicle is determined to be within the fix area. This terrain profile has fewer elements than there are rows in the reference matrix. The main calculation of the MAD algorithm for the SSLM case is:

$$MAD_{k.n} = \frac{1}{M-K}\sum_{m=1}^{M-K}\left|h_m - H_{k+m.n}\right|$$

Equation 2.2

where:

1. $H_{k+m.n}$ is the $(k+m,n)^{th}$ element of the pre-stored $M \;\; x \;\; N$ reference matrix
2. K is the difference between the number of rows in the terrain profile and the number of rows in the reference matrix[1]. Given $k \in [0 \quad K]$, this will allow one dimension of variation between the terrain profile and the reference matrix.
3. h_m is the m^{th} measured terrain elevation data point
4. $MAD_{k.m}$ is the mean absolute difference between the m^{th} terrain elevation data point and the $(k+m,n)^{th}$ reference matrix data point

As in the MAD algorithm for the LSSM case, the MAD calculation presented in Equation 2.2 will determine the mean absolute difference between the n^{th} column of the reference matrix and M elements of the terrain profile. This algorithm must be applied to all arrangements of the terrain profile and the reference matrix in the SSLM scenario.

In other words, for the SSLM the MAD must be also calculated for all possibilities of (k,n) given that $k \in [0 \quad K]$ and $n \in [1 \quad N]$. Refer to Figure 2.6.

---

[1] Assuming the terrain profile has fewer rows than the reference matrix. This is the SSLM scenario.

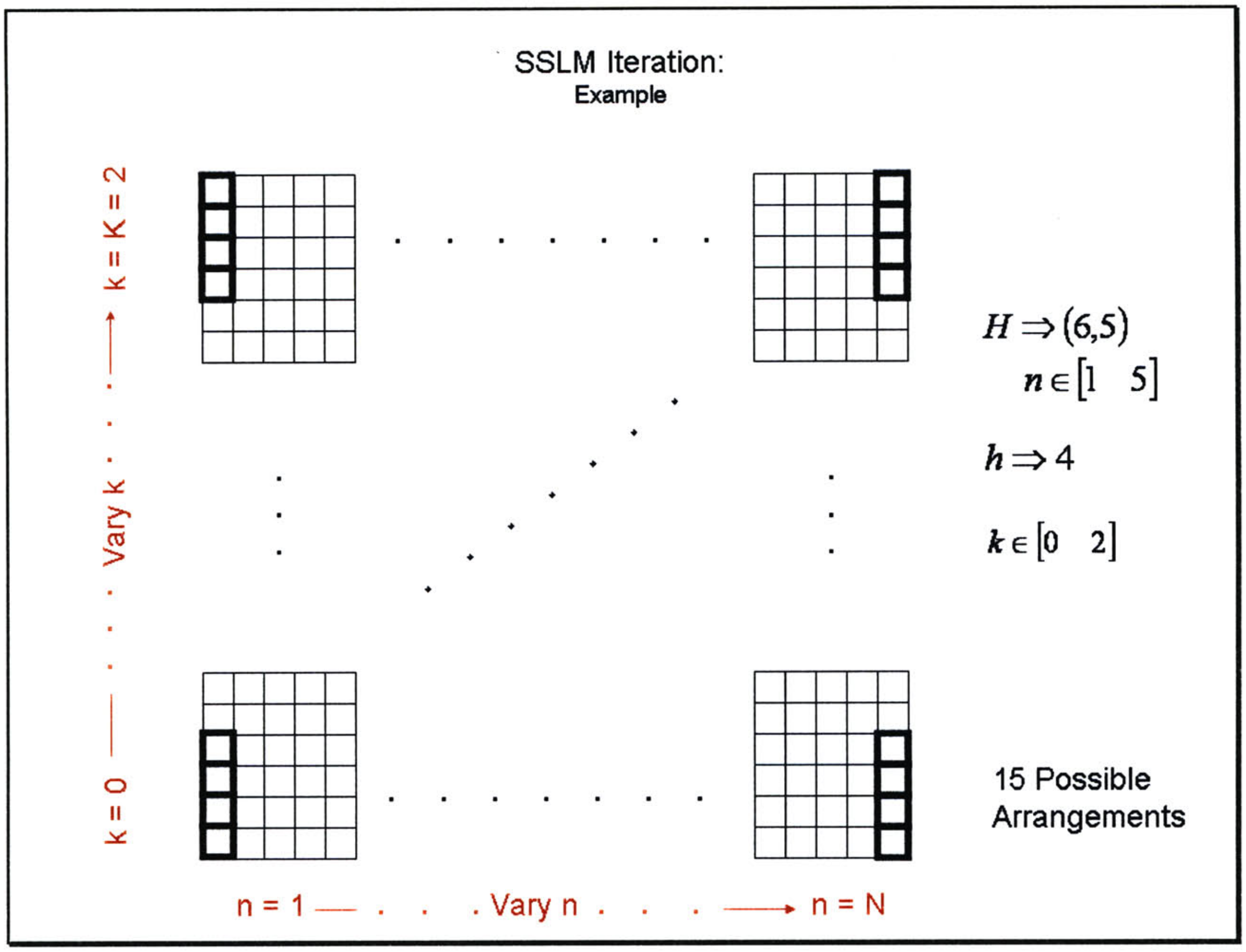


Figure 2.6. An Example of an SSLM Correlation Attempt.

For this analysis of the TERCOM algorithm, only the LSSM technique was implemented.

2.1.2.3 MAD Algorithm Notes

The MAD algorithm assumes that the vehicle measures the overflown terrain along a path that aligns with a single column of the map and maintains straight flight within that column. Any cross-track skewing as a result of cross-track velocity errors of the vehicle will result in errors in the position fix. In addition, any downrange velocity error will result in uneven sampling within the TERCOM cells.

That is, if the velocity of the vehicle is not precisely known, the sampling of the terrain profile will not be able to properly determine where one terrain profile cell should end and another begin.[1]

Several reference map characteristics can be derived to give an estimate of that map's feasibility as a fix area for use with TERCOM. However, map characteristics should not be used as the sole determination of a map's feasibility. It should also be noted that the performance of the MAD algorithm with a map along the matrix columns does not guarantee the same performance along the matrix rows and vice versa. Simulations of reference matrix overflight must be performed to validate that map's use for TERCOM. Proper selection of the maps should provide the lowest estimated Circle Error Probability (CEP) value as well as reduce or eliminate the probability of a false fix. Refer to section 2.1.3.3 for an explanation of the CEP.

2.1.3 The VAR Correlation Algorithm

As discussed above, TERCOM uses a barometric altimeter in order to determine the vehicle's altitude above MSL for terrestrial applications. The measurement of the MSL altitude allows the AGL altitude to be translated into terrain profiles. However, the mean sea level has no analogous quantity on the Moon. Without some analogue to the MSL altitude, terrain profiles cannot be produced and the MAD algorithm is unable to generate a position fix.

Considering this situation, the author developed a variance-based (VAR) algorithm to eliminate the need for barometric altimeter measurements in obtaining a position fix. Considering the apparent impossibility of obtaining an MSL altitude measurement near the lunar surface, this modification to the MAD algorithm enhances the applicability of TERCOM to lunar TRN.

Conceptually, the VAR algorithm is quite simple. Consider the case in which the overflight path of a vehicle is straight, level flight. Assume this vehicle is able to obtain only a terrain-referenced altitude profile (AGL), using a LIDAR, radar altimeter, or similar sensor. Stored onboard is a terrain

[1] Typically, several measurements are averaged to determine the value of a terrain profile cell.

reference map comprised of a grid of elevation data referenced to MSL. The VAR algorithm sums ('stacks') the profile of altimetry measurements with the values of the reference topography map as depicted in Figure 2.7. When the AGL terrain profile is stacked atop the correct swath of the reference map, the stacked altitude will be nearly constant, reflecting straight and level flight. Therefore, if the variance value for each 'stacking' is determined, the smallest variance will produce the best position fix.

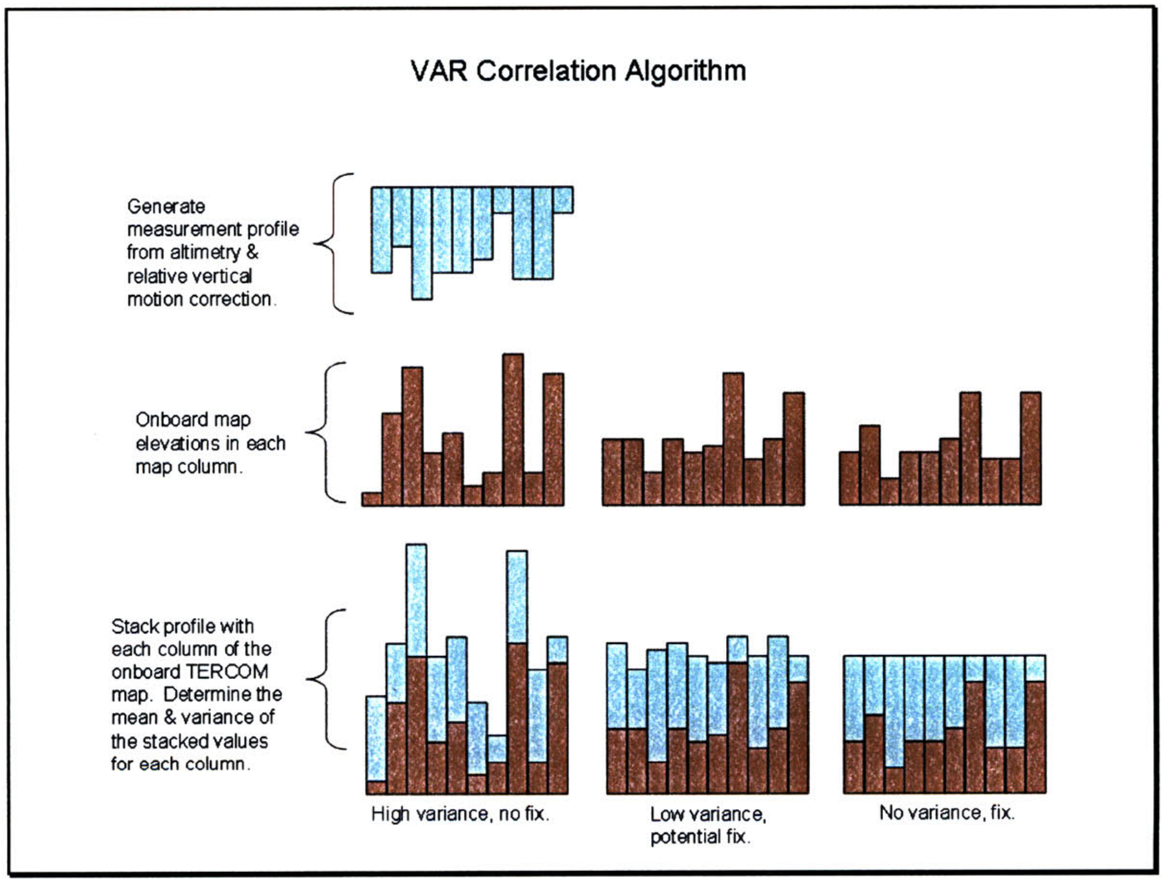

VAR Correlation Algorithm
Generate measurement profile from altimetry & relative vertical motion correction.
Onboard map elevations in each map column.
Stack profile with each column of the onboard TERCOM map. Determine the mean & variance of the stacked values for each column.
High variance, no fix.
Low variance, potential fix.
No variance, fix.

Figure 2.7. Description of the VAR Correlation Algorithm.

This algorithm operates under the assumption that the vehicle is at a constant altitude relative to MSL. If the vehicle's altitude is changing, relative measurements of the vertical displacements need to be determined. These vertical displacements can then be added to the altimetry measurements to produce a set of measurements relative to a constant altitude.[1]

As in the case of the MAD algorithm, the VAR algorithm is also dependent upon whether the LSSM or the SSLM scenario is being implemented. Both VAR algorithms will be presented below. Note, however, that only the LSSM VAR algorithm will be used in the analysis.

2.1.3.1 The Long Sample Short Matrix VAR Algorithm

The VAR algorithm for the LSSM is shown below:

$$VAR_{k,n} = \sigma_{k,n}{}^{2} = \frac{1}{M-1}\sum_{m=1}^{M}\left(x_{k,m,n} - \bar{x}_{k,n}\right)^{2}$$

Equation 2.3

where:

1. $\bar{x}_{k,n} = \frac{1}{M}\sum_{m=1}^{M} x_{k,m,n}$ is the average over the column of 'stacked' reference data and measured data (k^{th} displacement of the elevation profile, n^{th} column, M reference matrix rows).
2. $VAR_{k,n}$ is the average variance of the 'stacked' data column (k^{th} displacement along the elevation profile, n^{th} column).
3. $\sigma_{k,n}$ is the standard deviation of the stacked data column (k^{th} displacement along the elevation profile, n^{th} column).
4. $x_{k,m,n} = Terrain_{k+m} + Map_{m,n}$ is the $(k,m,n)^{th}$ element of stacked data (k^{th} displacement along the elevation file, n^{th} column).

[1] The concept of this algorithm was developed by Piero Miotto and Matthew Hale without reference to other sources. Piero Miotto is an engineer at the Charles Stark Draper Laboratory.

5. M is the number of rows in the reference matrix. M is also the number of 'stacked' data points for a VAR LSSM calculation.
6. N is the number of reference matrix columns.
7. K is the maximum displacement along the elements of the measured terrain elevation file.

The VAR algorithm presented in Equation 2.3 will determine the variance of the n^{th} column of the reference matrix stacked with M elements of the terrain elevation column, displaced by k elements along the terrain elevation column. As in the MAD algorithm, the VAR algorithm will be iteratively 'slid' across the reference matrix similar to the process in Figure 2.5.

In other words, the VAR algorithm must be calculated for all possibilities of (k,n) given that $k \in [0 \quad K]$ and $n \in [1 \quad N]$.

The LSSM VAR algorithm is implemented as the TERCOM method used to evaluate the potential performance of TERCOM on the lunar surface. Therefore, an example of a position fix of the LSSM VAR algorithm as plotted by Matlab is shown in Figure 2.8.

The portion highlighted in red indicates the portion of the terrain profile (the profile is in yellow) that correlates to the reference matrix column (the matrix is in blue). For this figure, only the best fit is plotted. This implies that this terrain profile has already been varied across all of the columns of the reference matrix, and for all possible row displacements of the profile, to determine the location of the best fit.

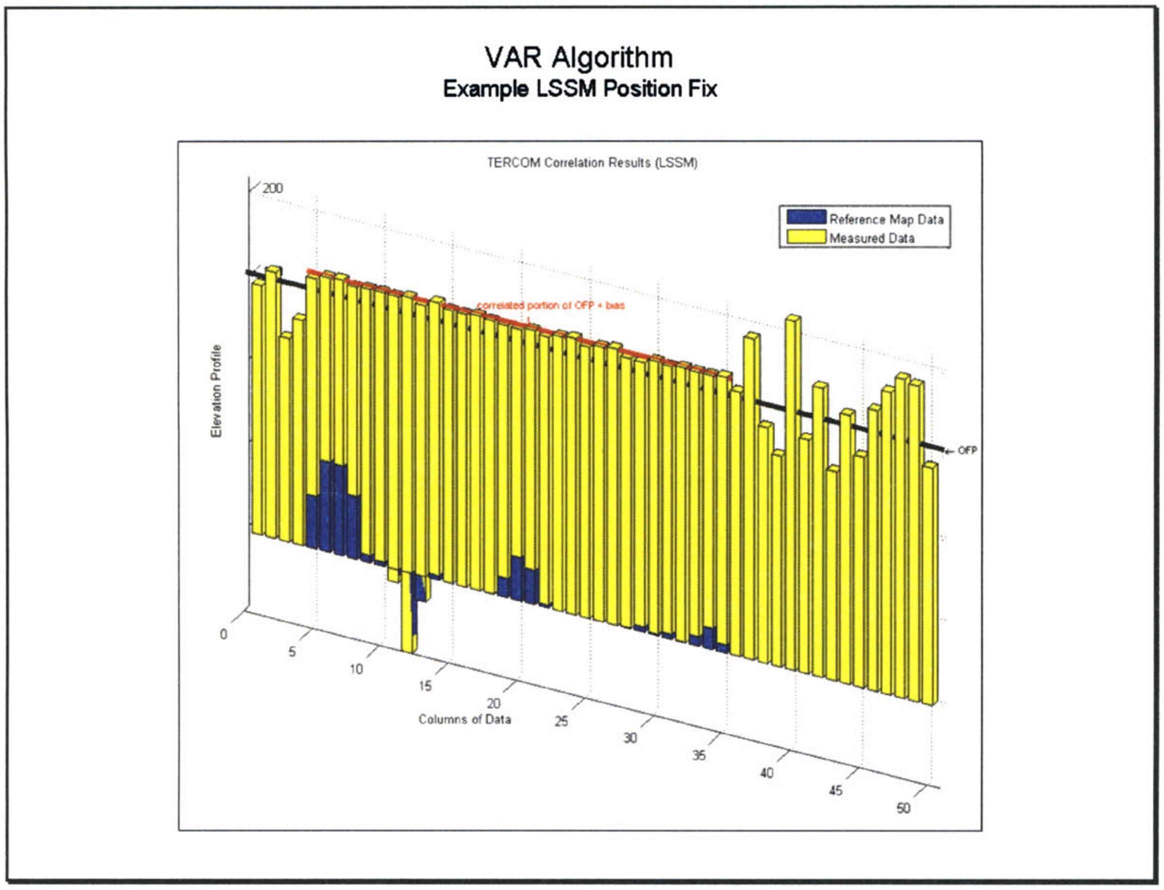


Figure 2.8. An Example Position Fix Using the VAR Algorithm.[i]

2.1.3.2 The Short Sample Long Matrix VAR Algorithm

The VAR algorithm for the SSLM case is shown below:

[i] OFP refers to the overflight path of the vehicle

$$VAR_{k,n} = \sigma_{k,n}{}^2 = \frac{1}{M-K-1}\sum_{m=1}^{M-K}\left(x_{k,m,n} - \bar{x}_{k,n}\right)^2$$

Equation 2.4

where:

1. $\bar{x}_{k,n} = \frac{1}{M-K}\sum_{m=1}^{M-K} x_{k,m,n}$ is the average over the column of 'stacked' reference data and measured data (k^{th} displacement along the reference matrix rows, n^{th} column, M reference matrix rows).
2. $VAR_{k,n}$ is the average variance of the 'stacked' data column (k^{th} displacement along the reference matrix rows, n^{th} column).
3. $\sigma_{k,n}$ is the standard deviation of the stacked data column (k^{th} displacement along the reference matrix rows, n^{th} column).
4. $x_{k,m,n} = Terrain_m + Map_{m+k,n}$ is the $(k,m,n)^{th}$ element of stacked data (k^{th} displacement along the reference matrix rows, n^{th} column).
5. M is the number of rows in the reference matrix.
6. N is the number of columns in the reference matrix.
7. K is the maximum displacement along the rows of the reference matrix.

The VAR algorithm presented in Equation 2.4 will determine the variance of $M-K$ elements of the n^{th} column of the reference matrix stacked with the entire terrain elevation column, displaced by k elements along the rows of the reference matrix. As in the MAD algorithm, the VAR algorithm will be iteratively 'slid' across the reference matrix similar to the process in Figure 2.6.

In other words, the VAR algorithm must be calculated for all possibilities of (k,n) given that $k \in [0 \quad K]$ and $n \in [1 \quad N]$.

2.1.3.3 VAR Algorithm Notes

The limitations of the MAD algorithm also apply to the VAR algorithm. That is, the vehicle must fly at a known velocity over the reference matrix and maintain straight flight within the column of the reference matrix.

In addition, the Circle Error Probability can also be computed for the VAR algorithm. For both the MAD and VAR algorithms, the minimum CEP is a function of the cell dimensions of the reference map. This minimum CEP value is determined with the following equation.

$$CEP_{\min imum} = 0.389(d/\sqrt{12} + d/\sqrt{12}) = .339d$$

Equation 2.5

where:

1. d is the map cell dimension.

The CEP equation is taken from Baker and Clem.[8] It defines the radius from the correct position match within which 50% of position match attempts will fall given the terrain on which the matching occurs.

2.2 The CraterMaker and CraterMap Utilities

The resolutions of current lunar topography maps are insufficient for use in verifying the feasibility of TERCOM for use on the moon. The best lunar topography maps available today are from the

Clementine mission of 1994[i]. These maps provide horizontal resolutions on the order of kilometers. TERCOM requires much higher resolutions than are available with Clementine data for any type of realistic testing or operations.

However, the lack of lunar elevation data will be blunted with the launch of the Lunar Reconnaissance Orbiter (LRO) in late 2008.[ii] The Lunar Orbiter Laser Altimeter (LOLA) instrument on board the LRO will provide global topography maps for the Moon with a 100 meter horizontal resolution and 1 meter vertical resolution. In some regions, these maps will be able to provide horizontal and vertical resolutions as good as 25 and 0.1 meters respectively. It is assumed these maps will be available prior to the execution of a lunar landing mission, and therefore available to use to build TERCOM reference matrices.

In order to test the TERCOM algorithms without the data expected to be gathered by the LRO, a simple lunar surface simulator tool was developed to provide topographic maps with sufficient resolution to match the expected LOLA maps. Two Matlab utilities named the CraterMaker and the CraterMap together provide this capability.

The CraterMaker routine generates a list of randomly varying craters across a given surface. The CraterMap routine then builds off of the CraterMaker functionality by creating a virtual terrain from the craters produced by the CraterMaker.[iii]

2.2.1 The CraterMaker Tool

The CraterMaker routine is supplied with a user-defined crater density, latitude and longitude ranges, and an average radius for some given planet. CraterMaker then recreates the typical geometry of an

[i] Officially the Deep Space Program Science Experiment (DSPSE), more commonly known as the 'Clementine' mission, was a NASA mission to map the Moon, among other things.

[ii] A NASA spacecraft scheduled for launch in late 2008 to characterize the lunar environment for entry, descent, and landing. http://lunar.gsfc.nasa.gov/index.html

[iii] Neither of these routines were created by the author. However, an understanding of them is essential to the analysis that will be conducted on the author's TERCOM algorithms.

impact crater, generating several physical characteristics. These features include crater depth, crater radius, and rim height.

Using the user-defined crater density and the area of the surface under consideration, the CraterMaker then computes the number of craters to be generated. Next, the CraterMaker code defines ranges for the crater radii, depth, and height. The CraterMaker then assigns a random radius, depth and rim height for each crater given the range for these quantities. For this step, it is important to note that the crater radii distribution is not completely random like the depth and rim height. Instead, a probability distribution function (PDF) of crater radii was defined using the New Neukum Crater Production Function of Neukum, Ivanov, and Hatmann.[9] This PDF is heavily weighted towards smaller radius craters, with very large (greater than 1 km diameter) craters occurring very infrequently.[1]

The shape of the 'inside' of the crater, as well as the shape of a outside, are both assumed to follow a generic profile. From the center of the crater cavity to the rim, the profile is defined to follow a hyperbolic curve. From the rim of the crater outward, the profile is one of exponential decay. To make these profiles more realistic, a select amount of randomness is applied along these profiles. To ensure that the two profiles form a piece-wise continuous curve, boundary value constraints are defined for the profiles. This also ensures that the craters are of the desired shape and that the crater surface height ultimately goes to zero at its outer edge.

After it is defined, the crater profile is incrementally transformed 360° about the vertical axis through the crater center, creating a surface of revolution. In order to prevent the circular contours inherent in these types of surfaces, random variation is again applied at each azimuth increment in the transformation. This same procedure is then repeated for each desired crater until an entire list of craters has been processed.

[1] This PDF is similar to a chi-squared distribution, but more heavily weighted towards the low end.

The Neukum crater distribution function discussed above provides a relatively low overall crater density, which in turn makes it difficult to generate a TERCOM fix. This scenario is discussed in the results section below. An additional function of the CraterMaker utility allows the user to force a greater distribution of craters of a given size in the crater list. This capability represents the real-world process of selecting a TERCOM fix area with the correct type of surface features to allow for a position fix.

2.2.2 The CraterMap Tool

As mentioned previously, the CraterMap routine builds upon the functionality of the CraterMaker. The CraterMap utility creates a virtual terrain using the list of craters generated by the CraterMaker. CraterMap is capable of generating both flat surfaces and spherical projection maps.

The CraterMap routine begins to build a terrain by generating a mesh of points over the given surface area. All the points in the mesh initially have a z-value (height) of 0. CraterMap then iteratively superimposes the surface mesh associated with any given crater onto the surface mesh representing the simulated terrain. Using bi-directional interpolation, the CraterMap utility then determines the crater height that corresponds with each relevant terrain grid location. These crater heights are then stored as the z-value (height) of the terrain mesh. As craters are continually added, the local terrain elevation mesh is updated with each crater's height data. The end result is a very sophisticated and realistic looking simulation of the lunar topography. It is worth noting that one of the main benefits of this process is that the method provides an inherent ability to create overlapping impact craters, as well as craters within previously created impacts. In addition, this functionality does not significantly impact the computational effort required for simulating the terrain. Figure 2.9 gives an example output of the CraterMap utility.

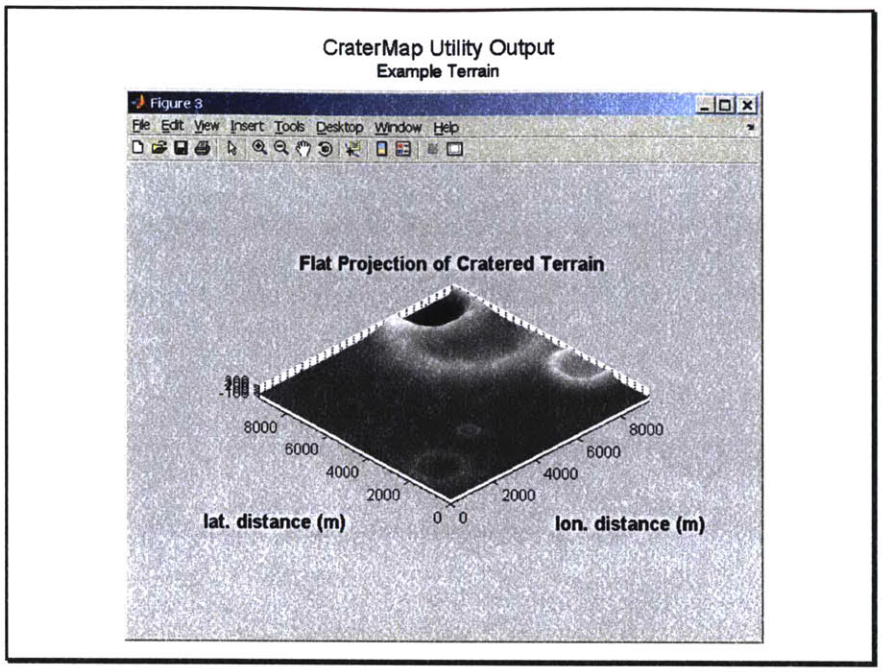


Figure 2.9. An Example Terrain Output of the CraterMap Utility.

2.3 Analysis Assumptions & Description

In order to characterize the performance of the VAR algorithm in a simulated lunar environment, the analysis included herein was initialized and conducted. The analysis was performed by executing a series of position fix attempts. Each attempt generated a set of maps that were then used as input for the TERCOM process. Multiple maps were utilized to increase the sample size for the analyses. For each attempt, the following user-defined values are required:

1. 1-σ sensor error term.
2. 1-σ map error term.
3. Mean crater density value in craters per square kilometer.
4. 1-σ crater density term.
5. Initial state of the random number generator.
6. Number of maps to generate.
7. List of crater radius values to force on each map of the run (optional).

The values of these user-defined quantities will be discussed with the results of the analyses as necessary.

2.3.1 Setup of the Analysis

For each map, a set of terrain profiles is generated as simulated sensor input. The generation of these profiles is accomplished by first creating a terrain profile vector that consists of random entries. Next, part of the profile vector is overlaid with the data from a reference matrix (map) column with noise added. From any given map, a series of terrain profile vectors can be created in this manner by varying the map column to overlay onto the profile as well as by varying the profile location of the map column. This generates a larger sample size for the analysis. The generation of the TERCOM profiles is shown in Figure 2.10 below. The correlation process implements the LSSM technique described in section 2.1.3.1.

The correlation of these simulated terrain profiles to the onboard reference map is outlined in the Figure 2.11 below. This figure shows the correlation for only the profile set generated from the first column. This same process is repeated for the profile sets generated with data from each map column. As a result, a large number of terrain profiles can be determined and matched with the reference matrix for any given initial map. This greatly increases the practical number of iterations of the analysis.

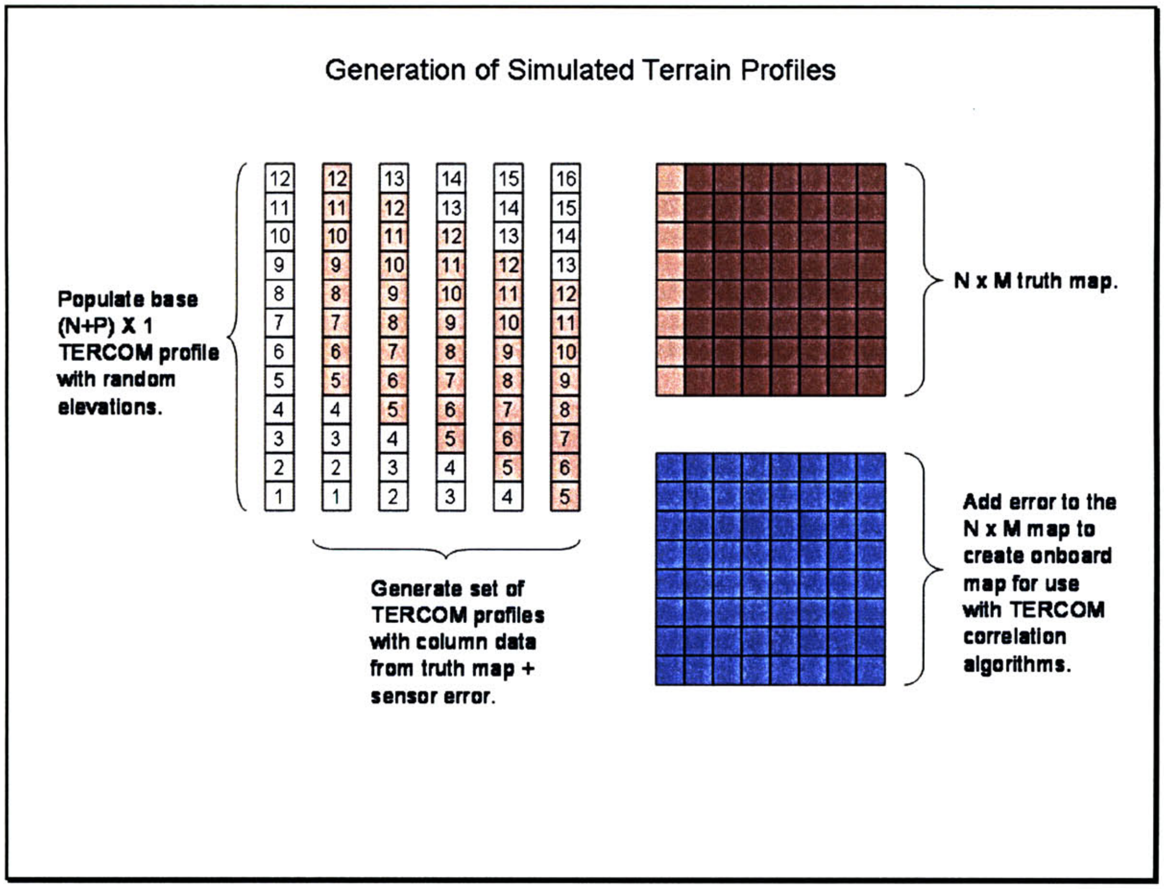

Generation of Simulated Terrain Profiles
Populate base (N+P) X 1 TERCOM profile with random elevations.
Generate set of TERCOM profiles with column data from truth map + sensor error.
N x M truth map.
Add error to the N x M map to create onboard map for use with TERCOM correlation algorithms.

Figure 2.10. Overview of the Generation of Simulated Terrain Profiles for TERCOM Analysis.

It is assumed that cell measurements in any given terrain profile are perfectly aligned with cells in the map. This is an idealized case in order to simplify the analysis. In reality, the values of the measured terrain profiles might be skewed due to velocity errors or initial knowledge error.

For this study, the maximum sample rate is assumed to be $100\ Hz$ and the maximum vehicle ground velocity is assumed to be 1700 meters per second. This correlates to an altimeter sample every 17 meters. It is assumed that this sample rate is sufficient to characterize the average height in a TERCOM cell. Since the velocity of the vehicle will decrease during breaking and approach, requiring that there be more samples per cell during breaking to ensure that the terrain profile cells cover the same area of the lunar surface whether or not the spacecraft is breaking. Any error during this breaking maneuver is assumed to be included in the single sensor error term outlined below.

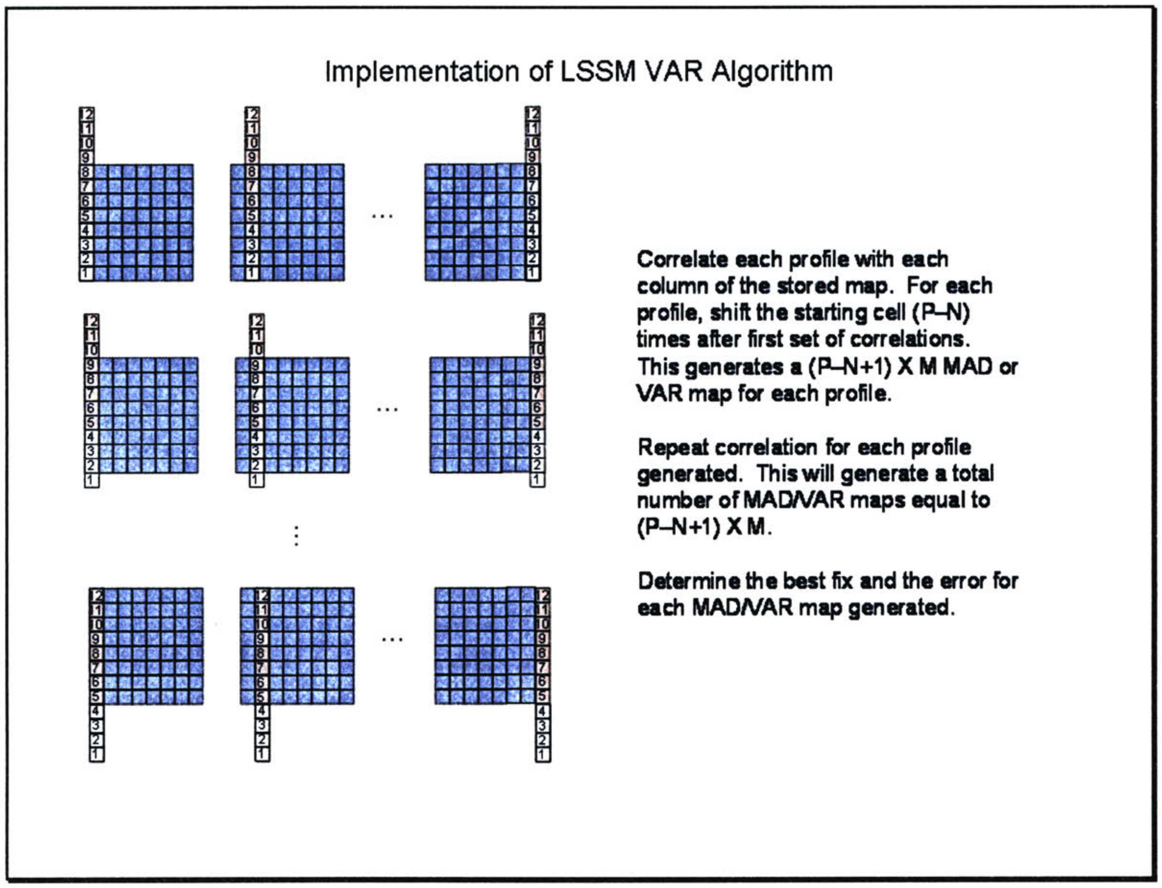

Implementation of LSSM VAR Algorithm
Correlate each profile with each column of the stored map. For each profile, shift the starting cell (P–N) times after first set of correlations. This generates a (P–N+1) X M MAD or VAR map for each profile.
Repeat correlation for each profile generated. This will generate a total number of MAD/VAR maps equal to (P–N+1) X M.
Determine the best fix and the error for each MAD/VAR map generated.

Figure 2.11. Overview of the Correlation of Simulated Terrain Profiles to the Reference Map for TERCOM Analysis.

A separate analysis should be performed to determine the validity of these assumptions and the required number of measurement samples to properly characterize an elevation cell. Other than these basic assumptions, the velocity of the vehicle and the sampling rate of the altimeter are not addressed explicitly in this analysis.

Only two sources of error were introduced to the execution of the TERCOM algorithm. Both errors are modeled with a zero mean Gaussian distribution that has an adjustable standard deviation value. These error sources were the map error and the sensor error. The map error represents the error between the map and the true terrain. The sensor error is the error that is added to the measured terrain profile cells and has multiple potential sources. These sensor error sources include sensor inaccuracies, imprecise attitude knowledge of the vehicle, and error caused by under-sampling the altitudes for a given map cell. For simplification, all sensor error sources are modeled as a single error term.

2.3.1.1 Simulating the Reference Map

For the generation of the reference matrix, craters are added to a flat base map with a nominal elevation of zero as discussed above. This is assumed to be a worst case scenario, as it will require the TERCOM algorithm to determine a position fix based only upon craters and not on any underlying features such as hills, slopes, cliff faces, etc.

The CraterMap utility is used to generate 100 meter resolution TERCOM reference maps for this analysis. Initially, the horizontal resolution of the maps created from the CraterMap utility is roughly 20 meters. These 20 meter resolution values are then averaged into 100 meter cells to provide a lower resolution topographic map. Recall that the LRO mission will also provide lunar maps with 100 meter resolution. Therefore, these maps mimic the quality of the data expected to be available for lunar navigation. Figure 2.12 provides an elevation plot of an example map created by the CraterMap utility for this analysis.

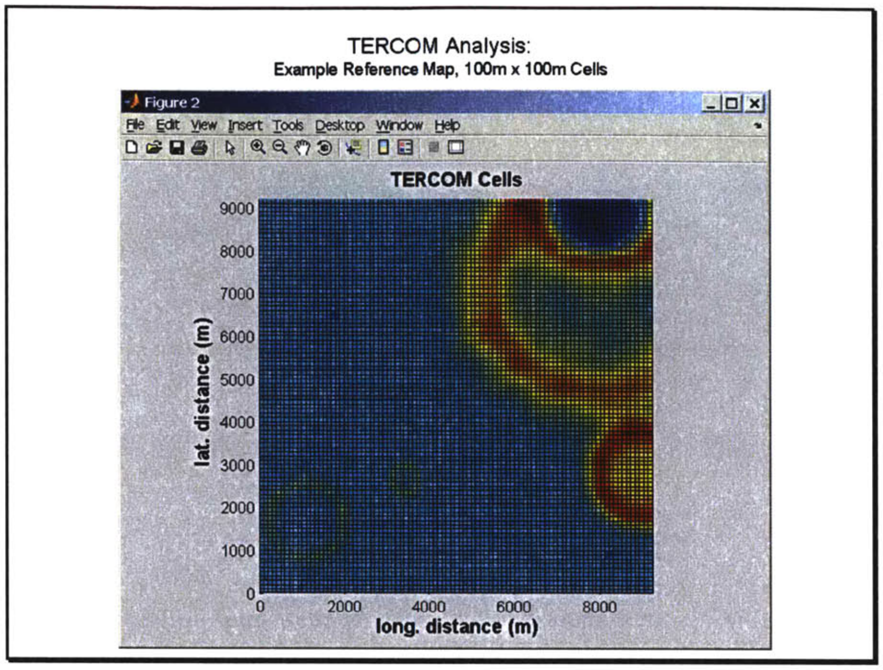


Figure 2.12. An Example Reference Map with 100m Cells for TERCOM Analysis.

2.3.1.2 Reference Map Characteristics

Several of the characteristic values of these maps were derived in order to help measure the maps' effectiveness in allowing a TERCOM fix. As mentioned previously, these characteristics should not be used as the sole measure of a map's effectiveness at generating a TERCOM position fix. The characteristics include σ_T and σ_Z.[10] The characteristic σ_T is the standard deviation of the terrain elevations in a map and represents the roughness of the map surface. It is calculated as

$$\sigma_T = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(H_i - \overline{H}\right)^2}$$

Equation 2.6

where:

$$\overline{H} = \frac{1}{N}\sum_{i=1}^{N} H_i$$

Equation 2.7

A higher value for σ_T indicates a rougher terrain.

The other characteristic σ_Z is the standard deviation of the point-to-point changes in the terrain elevation and also provides an indicator of the roughness of the map surface along either the rows or columns of the map. The value of σ_Z will differ whether TERCOM is attempting to correlate the terrain profile to the map rows or the map columns. For this study, the profiles are always matched with the map columns. The value of σ_Z is determined with the following equations:

$$\sigma_Z = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}\left(D_i - \overline{D}\right)^2}$$

Equation 2.8

where:

$$D_i = H_i - H_{i+1}$$
$$\overline{D} = \frac{1}{N-1}\sum_{i=1}^{N-1} D_i$$

Equation 2.9

A higher value of σ_Z also indicates a rougher terrain.

A final map characteristic is the value of the terrain correlation length, X_T. This parameter is the distance between map rows or columns so the value of their normalized auto-correlation function is minimized to e^{-1}. In other words, any two map rows or columns selected from the reference matrix are assumed to be independent of one another if they are separated by a distance equal to X_T. It is derived using the values of σ_T and σ_Z. The equation relating σ_T, σ_Z, and X_T is shown below.

$$\sigma_Z^2 = 2\sigma_T^2\left(1 - e^{-\left(\frac{d}{X_T}\right)^2}\right)$$

Equation 2.10

The smaller the value of the X_T parameter, the more independent the columns (or rows) of the reference matrix. As the independence of the columns (or rows) increases, the likelihood of false position match decreases.

For each map in this analysis, the mean and standard deviation of the position fix errors are determined for each terrain profile. Each individual position fix error is determined by noting the number of row and column cells between the calculated and true position for a given terrain profile. This number of cells is then multiplied by the length of a cell. Finally, the position fix error is calculated as the magnitude of the vector made up by these row and column error components. The mean and standard deviation of the error are then plotted against the corresponding map values of σ_T and σ_Z for all maps in a given run.

2.4 Analysis Results & Conclusions

Figure 2.13 below shows the expected trends for errors and the number of maps suitable for TERCOM based on underlying surface types and crater distributions of a map of interest. For this analysis, only the map types in the first row have been tested.

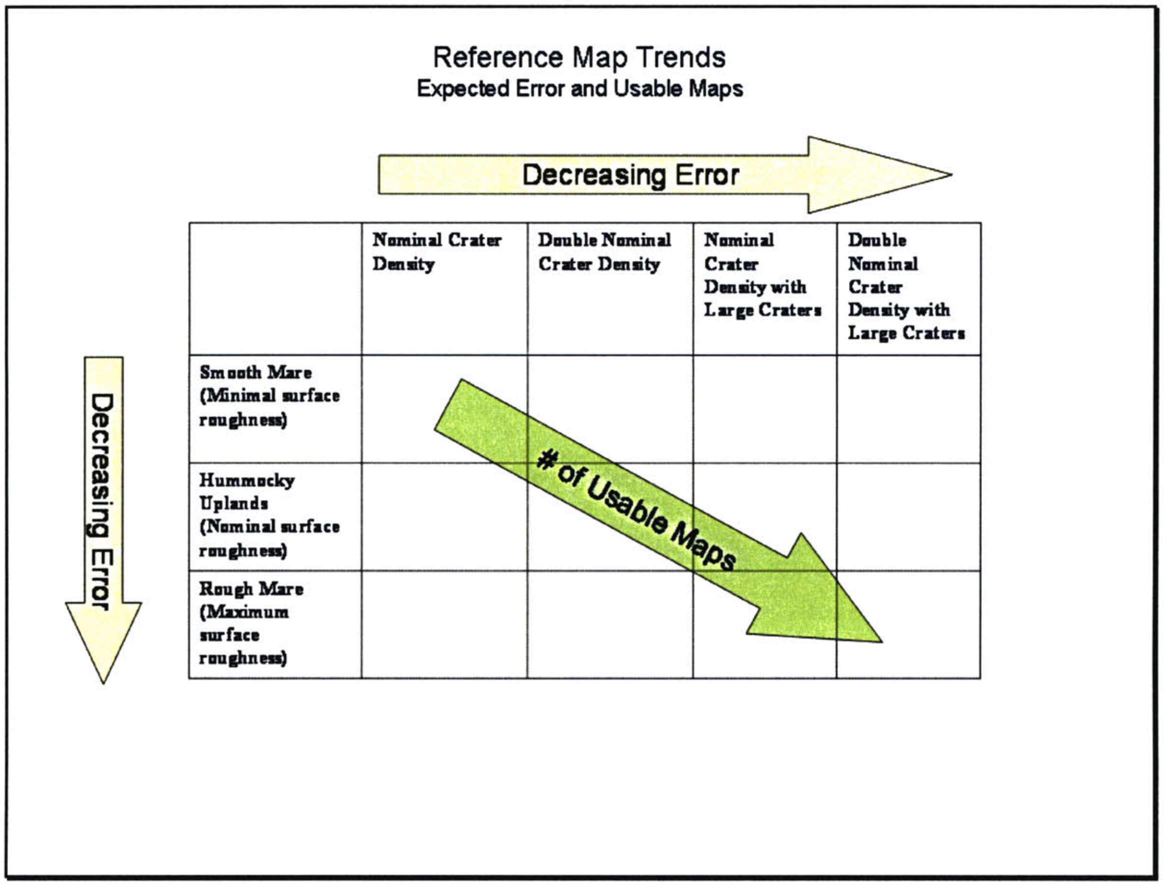

Reference Map Trends
Expected Error and Usable Maps
Decreasing Error
Nominal Crater Density
Double Nominal Crater Density
Nominal Crater Density with Large Craters
Double Nominal Crater Density with Large Craters
Smooth Mare (Minimal surface roughness)
Hummocky Uplands (Nominal surface roughness)
Rough Mare (Maximum surface roughness)
# of Usable Maps
Decreasing Error

Figure 2.13. Expected Error Trends for the TERCOM Reference Maps.

Figure 2.14 through Figure 2.19 below show the results from several runs of the TERCOM algorithm and shows the average fix errors for each map plotted against the corresponding σ_T value. The sigSensor term represents the error added to the sensor measurements in meters and the sigMap term represents the amount of error added to the reference map in meters. As outlined in the Figure 2.13

above, these plots show that the error decreases as the density of craters increases. The error also decreases with the addition of several larger, deeper craters.

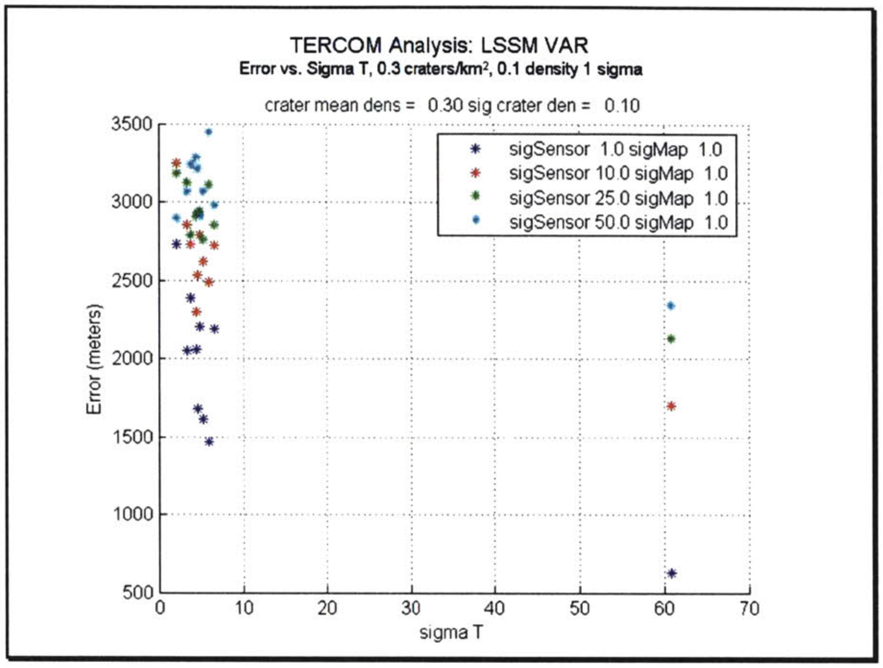


**Figure 2.14. TERCOM LSSM VAR Analysis:
0.3 craters/km^2 with standard dev. of 0.1 craters/km^2.**

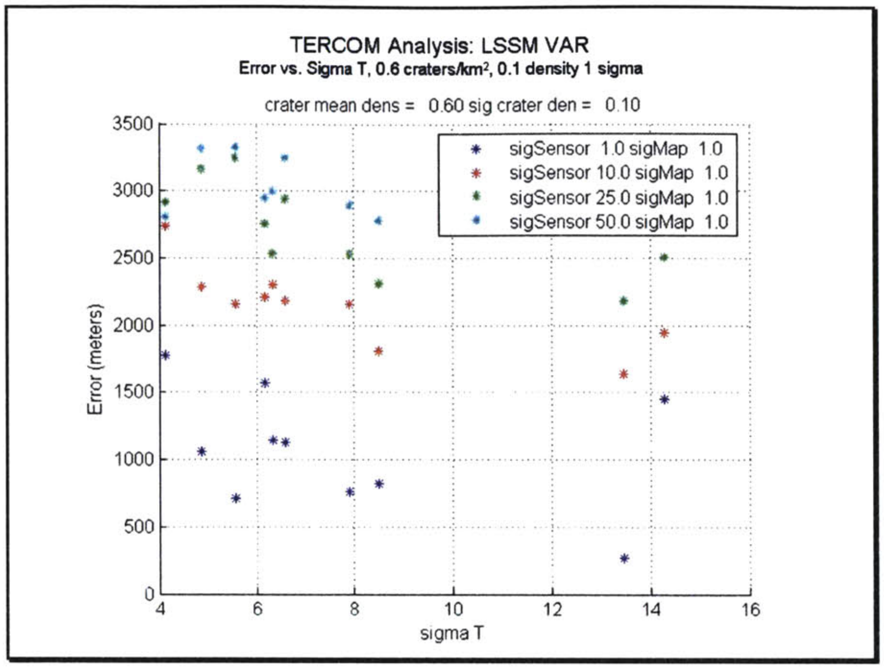


**Figure 2.15. TERCOM LSSM VAR Analysis:
0.6 craters/km^2 with standard dev. of 0.1 craters/km^2.**

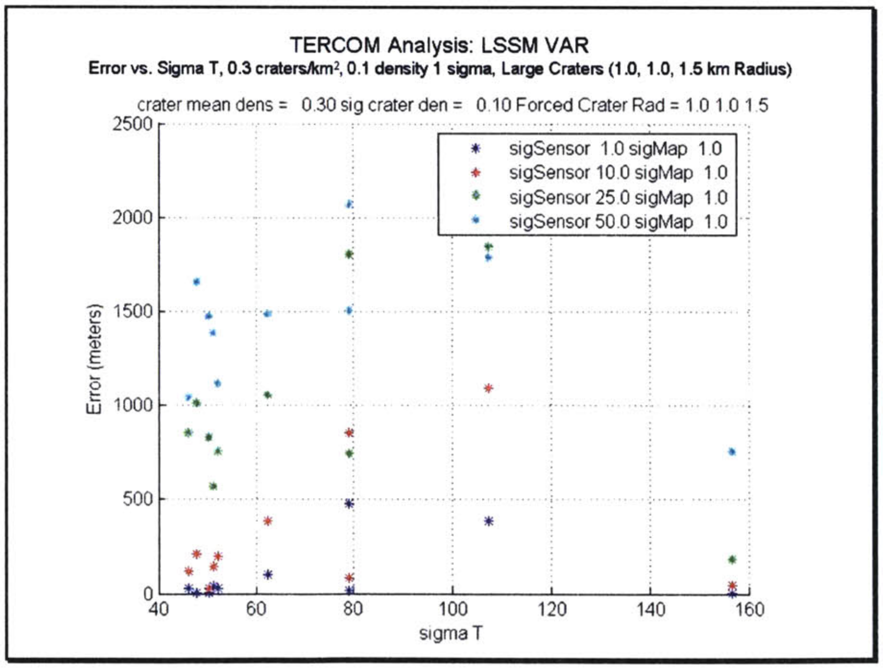


**Figure 2.16. TERCOM LSSM VAR Analysis:
0.3 craters/km^2 with standard dev. of 0.1 craters/km^2,
Forced Large Craters of [1.0 1.0 1.5] km Radius.**

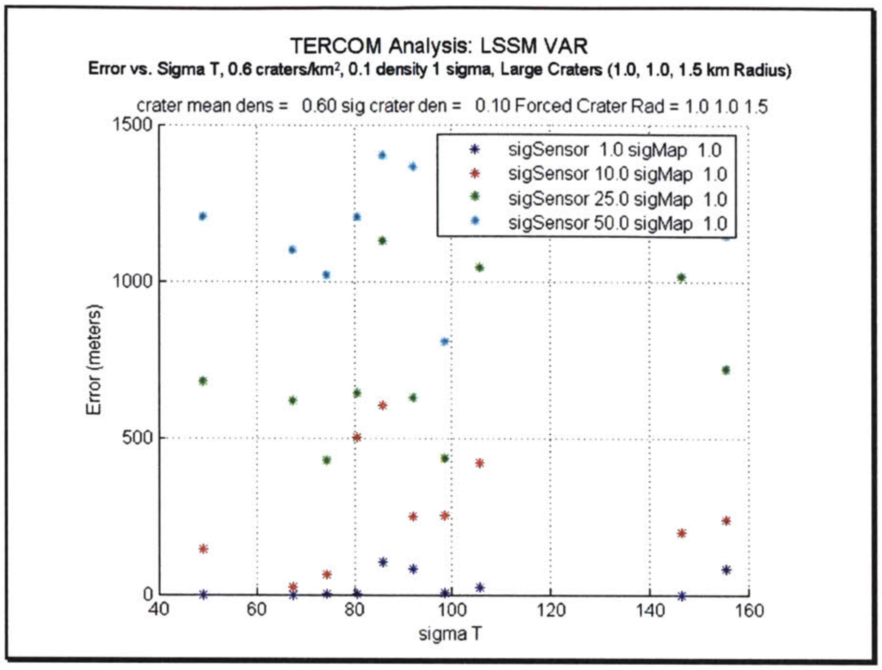


Figure 2.17. TERCOM LSSM VAR Analysis:
0.6 craters/km^2 with standard dev. of 0.1 craters/km^2,
Forced Large Craters of [1.0 1.0 1.5] km Radius.

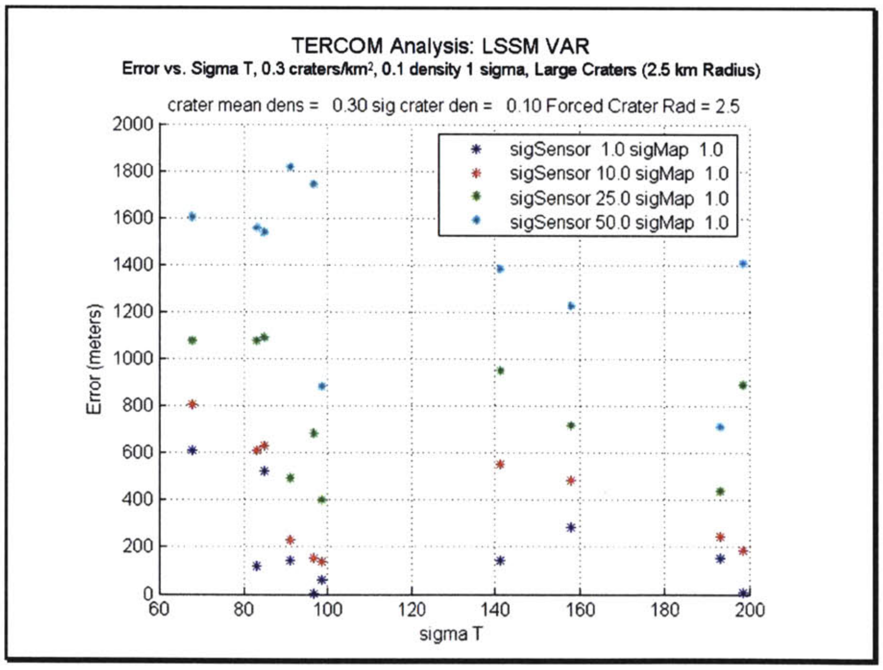


Figure 2.18. TERCOM LSSM VAR Analysis:
0.3 craters/km^2 with standard dev. of 0.1 craters/km^2,
Forced Large Craters of 2.5 km Radius.

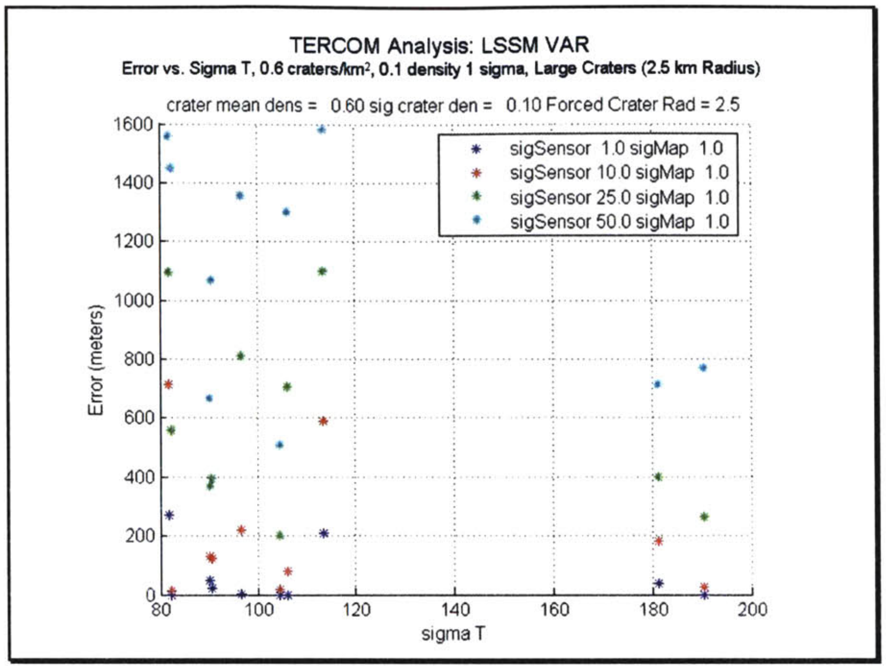


Figure 2.19. TERCOM LSSM VAR Analysis: 0.6 craters/km^2 with standard dev. of 0.1 craters/km^2, Forced Large Craters of 2.5 km Radius.

From these analyses, it appears that TERCOM can produce correct position fixes – but only on a very limited number of cratered surfaces. The algorithm tends to generate lower overall errors as the 'roughness' of a map increases. However, as discussed above, the 'roughness' measurement of a map is not a clear indicator of its 'uniqueness' for generating a correct position fix. In specific, the TERCOM algorithm does not seem to work well in the absence of large craters. Even with low noise levels, the no large crater cases of Figure 2.14 and Figure 2.15 show an unacceptably high level of error. Otherwise, the expected trends of Figure 2.13 are found to hold throughout the analyses.

The lack of a barometric altitude measurement which is required for the MAD algorithm appears to be sufficiently addressed by the VAR algorithm implementation. The results of the VAR algorithm nearly match the results from the MAD algorithm on the same set of data.[i] This indicates that the

[i] These results are not presented here; however, they were found during the course of this analysis.

VAR algorithm is a valid alternative to the MAD algorithm for the lunar case – and potentially for other uses.

Overall, the initial performance of the TERCOM algorithm is somewhat disappointing. Since there must be a high level of confidence in any position fix to allow that position to update the inertial navigation system, the probability of false fix must be very low. However, for all cases there are false position fixes demonstrated by the outliers in the graph. Even with a nominal amount of sensor and map noise, there were still false matches generated.

In addition, limitations upon the overall accuracy of the simulation are apparent in the analyses presented above. For these analyses, the maximum achievable accuracy is the dimension of a terrain profile cell, $100\ m$. However, in order to achieve precision landing, the position of the spacecraft must be known much more accurately than $100\ m$ by $100\ m$. Greater accuracy will require that the reference matrix cells each cover a smaller surface area of the Moon. However, the position error demonstrated exceeded the levels expected from the terrain cell size by hundreds – if not thousands – of meters.[1] This position fix error is unacceptable for the purposes of lunar navigation – and implies that the error is not due to the reference map, but primarily the result of the correlation algorithm.

The initial analyses presented here demonstrate that TERCOM is not necessarily robust enough to be able to generate position matches in the variety of conditions that will be encountered on the lunar surface. Combined with the limited accuracy of the initial TERCOM correlation attempts, the author has decided to look to other methods to use TRN to update the inertial navigation system of the spacecraft.

[1] Sub-pixel interpolation might increase the error due to the terrain cell size. However, the large errors indicate that the terrain cell size is not the causal factor in the TERCOM error levels.

Chapter 3

Preliminaries to the Crater Matching Routine

The crater matching routine is a method of identifying craters in an image of the lunar surface by matching the craters to a database of lunar craters. This chapter details both the matching routine and the creation of the crater database.

The matching process involves several steps. First, a crater that must be identified will be chosen from a sensor image. This crater will be referred to as the 'boresight crater.' Next, a feature vector describing the pattern of nearby craters will be built for the boresight crater. Finally, the boresight crater will be identified by searching a database for the known crater whose feature vector is the closest match to the feature vector of the boresight crater.

The creation of the database of lunar feature vectors will be accomplished by building a feature vector for each crater that may be visible in a sensor image. The feature vector itself is independent of the location of the boresight crater within the input image. It is also independent of the in-plane rotation of the image. This is true for the feature vectors built for the matching routine as well for the creation of the database.

This chapter (Chapter 3) discusses the details that are necessary before a more focused discussion of the crater matching routine is undertaken. The first section of this chapter, section 3.1, addresses the use of the crater matching routine as a potential component of terrain relative navigation architectures. The next section, section 3.2, goes on to further characterize the problem that the crater matching routine is intended to solve.

3.1 Utility of Crater Matching in Terrain Relative Navigation

As mentioned in Chapter 1 , TRN can be described as the process of sensing some terrain surface beneath a suspended vehicle, and using these sensor inputs to update the vehicle's knowledge of its own position, orientation, or velocity. TRN can take any of a number of forms[1]; however, all TRN techniques will include the following steps:

1. Obtain measurements of terrain surface using an onboard sensor.
2. Correlate the sensor input to a database containing terrain characteristics for the type of sensor being used.

[1]TERCOM, described in Chapter 2, and the crater matching routine described in this chapter, are two examples.

3. Use the information from the correlation to update the vehicle's navigational computers.

The second of these steps (TRL) is the heart of many TRN methods. The correlation between the sensor input and the surface provides information about the vehicle's relative position, orientation, or velocity. This information can be used, in turn, to aid in navigation over the surface. For example, Chapter 2 explains that the sensor in use for TERCOM was an altimeter. The focus of the TERCOM algorithm was to match elevation profiles recorded by the sensor with a database of terrain elevations that corresponded to the surface beneath the vehicle. After a successful match, the match information is fed to the navigational computers to allow for update.

The crater matching routine is intended to fill this role as well. It is designed to match craters from visual imagery of the lunar surface with a database of relative crater positions. A successful match by the crater matching routine will allow information on the spacecraft's pose in relation to the lunar surface to be determined. The determined pose will then be used to update the spacecraft's navigational computers.

3.1.1 Vision Based Operations for Terrain Relative Localization

Computer vision first became an area of significant study and application in the 1970s when computers had sufficiently matured, allowing large data sets to be processed. Since then, computer vision has been applied to many fields – fields which are as varied as medical imaging, missile targeting, and robotic maneuvering. Computer vision has also been applied very successfully to autonomous vehicle guidance and control on Earth. For example, computer vision has been applied to the navigation of military unmanned aerial vehicles extensively, achieving high levels of autonomy.[11]

The application of computer vision to space operations is a reality as well. For years, star trackers, sun sensors, and altimeters have been in use to complement inertial guidance systems. Autonomous vehicle guidance has also been achieved with computer vision in space applications. In 1997, the

Mars Pathfinder used computer vision to allow it to autonomously navigate the Martian surface.[12] Since then, Opportunity and Spirit arrived on the Martian surface in 2004 and have traveled miles over the red planet's surface using computer vision to navigate the often complex terrain it found.[13] Computer vision has found many other uses in space as well. One use of computer vision that is currently being developed is the application of vision based operations to terrain relative navigation for a landing spacecraft.

The National Aeronautics and Space Administration (NASA) is currently funding the Space Technology 9 Project (ST9) as part of the New Millennium Program.[14] Under ST9, the Jet Propulsion Laboratory is developing a "terrain-relative guidance system, which integrates computer vision and inertial sensing to perform terrain-relative navigation, hazard detection and landing-site targeting."[15] This system will be based upon computer vision and will accomplish terrain relative navigation. That is, one of the objectives of the ST9 Project incorporates vision based navigation to achieve terrain relative localization.

3.1.2 Crater Matching and Lunar Vision Based Terrain Localization

As ST9 demonstrates, developing a precision landing capability for planetary missions is currently of importance to space exploration. In the past, planetary landing has been at best imprecise, usually involving relatively large landing error dispersion ellipses encompassing tens of square kilometers of planetary surface[i]. However, if a landing precision of ten meters is achieved as desired, the accuracy of the lander will be approximately two hundred and fifty thousand times more accurate than the landing error dispersion area of Apollo 11. To realize this drastic improvement, current precision planetary landing efforts focus upon using relative navigation techniques to update more traditional inertial guidance and navigation techniques.

[i] Apollo 11's lunar landing in 1969 had a landing error dispersion ellipse of 19 km by 5 km. Apollo 12's 1969 landing error dispersion ellipse was 4 km by 2.5 km. The Mars Science Laboratory, scheduled for launch in 2009, has a dispersion ellipse similar in size to that of Apollo 11 (20 km diameter).

As mentioned above, computer vision presents an opportunity for both high-precision navigation and the possibility of achieving a planetary landing precision on the order of tens of meters, as desired. Many image processing problems are encountered in vision based terrain navigation. These include optical flow considerations, feature tracking, hazard avoidance, and many others. However, one of the most basic terrain navigation steps is terrain localization. Terrain localization can be described as determining where the spacecraft is by comparing sensor inputs to a database.

The process of terrain localization is laid out in three steps in Figure 3.1.

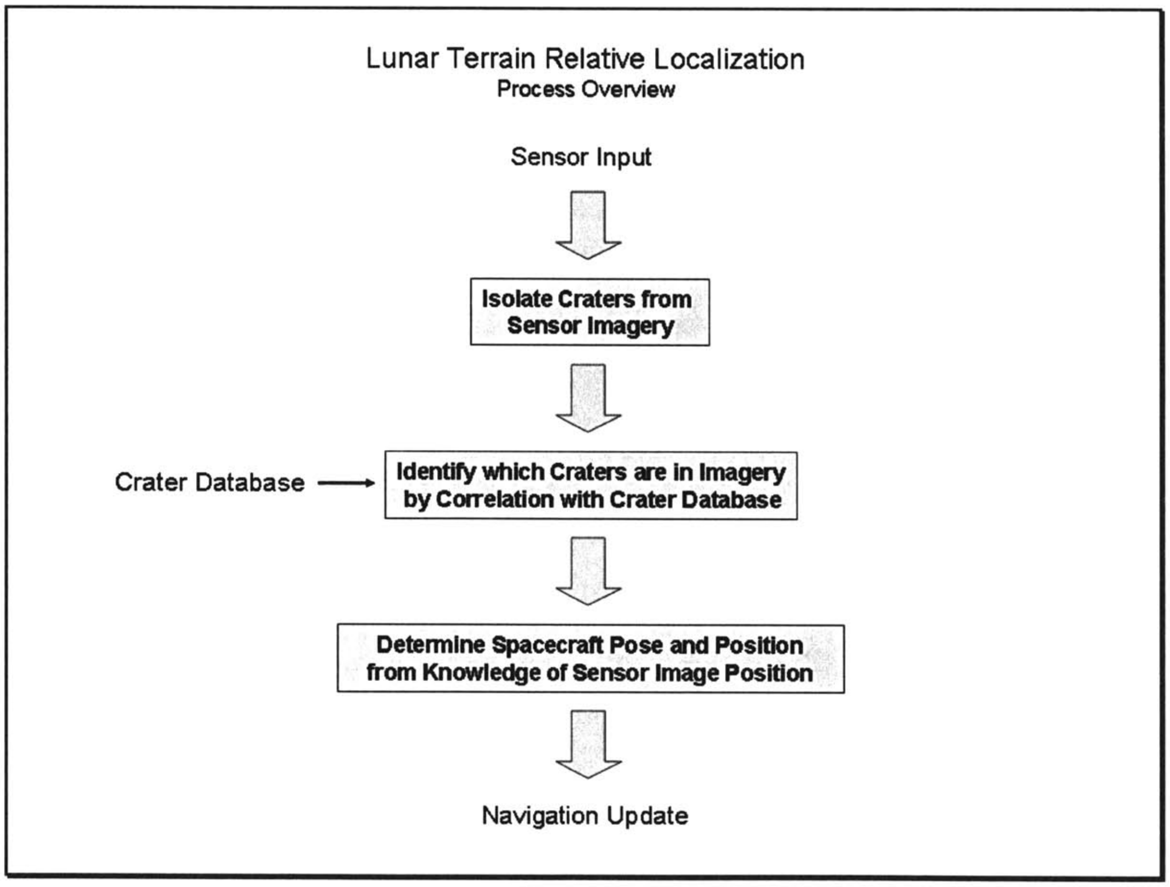

Lunar Terrain Relative Localization
Process Overview
Sensor Input
Isolate Craters from Sensor Imagery
Crater Database
Identify which Craters are in Imagery by Correlation with Crater Database
Determine Spacecraft Pose and Position from Knowledge of Sensor Image Position
Navigation Update

Figure 3.1[i]. Reprint of the Process Overview of Vision-Based Terrain Relative Localization.

[i] Also Figure 1.1.

In Figure 3.1, the first significant step in terrain localization is the isolation of craters from a given sensor image. The author will refer to this step as crater identification. The second step in the localization is to correlate the craters in the image with craters from a crater database. This step will be referred to as crater matching. The final step before updating the spacecraft's navigational computers is to determine the pose of the spacecraft based upon the correlation between the sensor image and the crater database. This final step will be referred to as pose estimation.[1] The calculated pose of the spacecraft can then be used to update the inertial guidance systems, typically using a covariance-based integration method such as the Kalman filter. The three steps depicted in Figure 3.1 will allow a spacecraft to use imagery of the lunar surface to update its inertial guidance systems. Repeated systematically as the spacecraft travels over the lunar surface, these terrain localizations will be critical in achieving precision landing capability.

The crater matching algorithm addresses a very specific problem encountered in lunar vision based terrain localization. The algorithm attempts to correlate an image of the lunar surface with a specific portion of a stored map of the lunar surface. This chapter will focus on this correlation process, specifically utilizing lunar cratering as the object of correlation between imagery and reference maps.

[1] The author will imply both orientation and position estimation, and will refer to this final step as 'pose estimation.' This is due to the fact that the algorithms for pose estimation include position estimation.

3.2 Crater Matching Problem Statement and Characterization

As mentioned earlier, the crater matching routine must identify the absolute position of the craters in a given image of the lunar surface. This is accomplished by building a feature vector based upon the pattern of nearby craters and matching this feature vector to a database of lunar craters.

Before examining the crater matching routine in more detail, several topics will be addressed. First the crater matching problem will be defined and characterized more fully in section 3.2.1. Then an overview of the star tracking problem and the generic solution algorithms to the star tracking algorithm will be presented in section 3.2.2. These star tracker algorithms will be studied due to the similarity between the star tracking and crater matching problems. A comparison between the star tracking methods will allow the author to identify a potential solution model for the crater matching problem. Finally, section 3.2.3 will present the preliminaries to the crater matching routine, such as essential terminology, units, and algorithm inputs and outputs.

3.2.1 Problem Statement

The statement of the crater matching problem has been made rather casually by the author in previous sections. This section will define the crater matching problem more rigorously.

The statement of the crater matching problem is rather simple. Given the relative center positions and radii of all the visible craters in an image of the lunar surface, determine the absolute position of the craters in the image. A database of the absolute crater center positions and radii of all lunar craters is provided. This database will allow the correlation between the image of the lunar surface and the absolute position of the landmarks in that image.

This correlation will be accomplished by a routine that is comprised of several steps:

1. Generate a database of patterns of nearby craters for any given crater on the lunar surface. This database will be stored and queried during the correlation of the image to the database.
2. Determine the pattern of nearby craters for a given crater in the sensor image about which the pattern is generated.
3. Generate a 'signature' for a given sensor image by repeating step 2 for every possible crater in an image.
4. Correlate the patterns in the image signature with the database of patterns. Successful correlation will allow the absolute location of the craters within the sensor image to be determined.

Each of these steps is a lengthy series of operations in itself, and so each will be addressed in Chapter 4 through Chapter 6.

Figure 3.2 presents a flow diagram for the steps enumerated above. These steps are clearly divided into two subsets – those that occur on the ground, and those that occur on the spacecraft. Both of these subsets share the process of generating a signature for a given image in common with each other. The ground operations consist of all those necessary to generate a crater database. The operations that occur onboard the spacecraft aim to generate an image signature for a given sensor image and to correlate that signature with the crater database. This correlation will determine the inertial position of the image. The crater matching process presented in Figure 3.2 will be examined in more details throughout the remainder of this chapter.

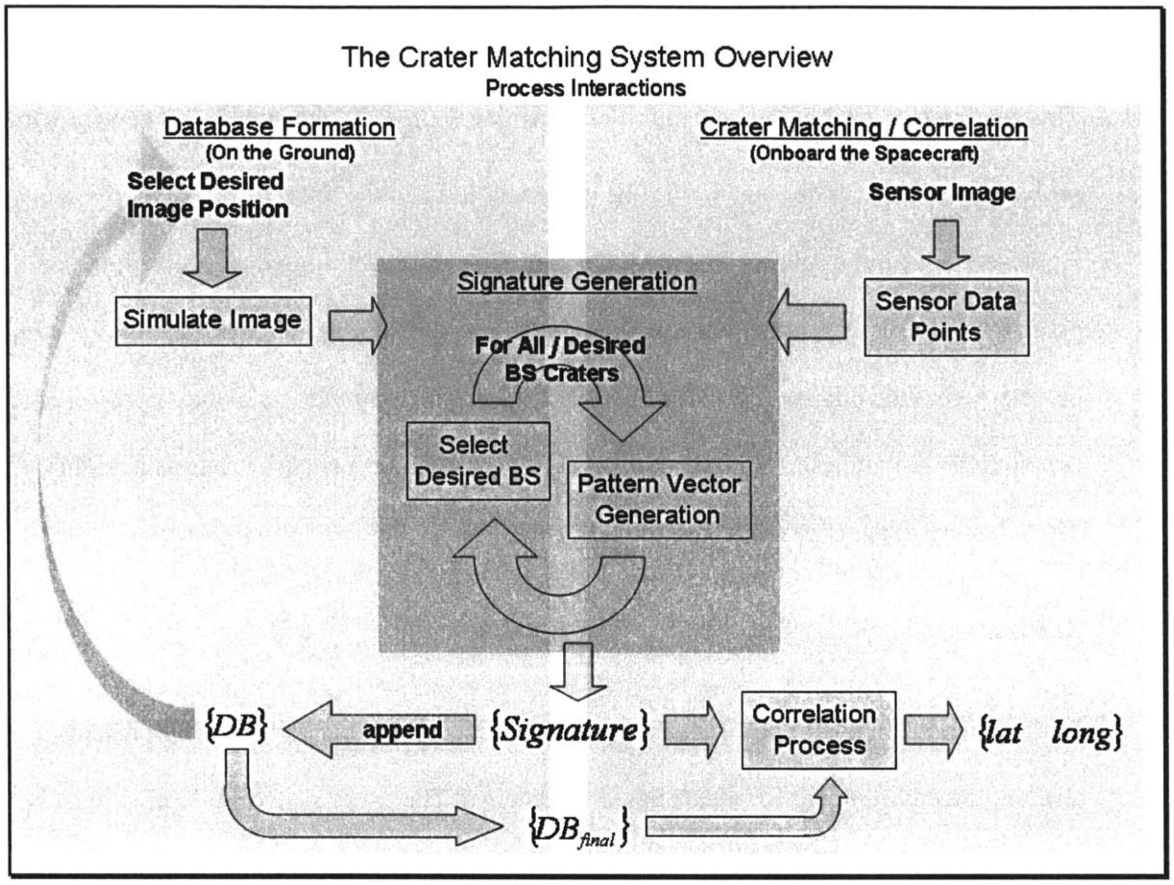


Figure 3.2. An Overview of the Interactions Between the Crater Matching Routine Processes.

The remainder of this section will continue to analyze the crater matching problem in more detail.

3.2.1.1 Requirements

As stated in the initial problem statement of section 3.2.1, the general requirement of this routine is to determine the absolute position of the craters in a sensor image without any prior position knowledge.[i] Because the crater matching routine is intended for eventual use on a lunar lander, there are additional requirements that this intended purpose imposes. Several of these are explained below:

[i] Save that the image was taken at a defined altitude above the lunar surface

1. Accuracy

This routine is crucial to the generation of a position fix for TRL operations. Therefore, the probability of false crater match must be extremely low, on the order of 10^{-3} or 0.1%. However, since the routine is intended as an initial position determination algorithm, the precision requirements are rather lax. In other words, the dependability of this routine must be very high, which is achieved by allowing the determination of the absolute position of the craters to be less precise. Note that the precision of the crater matching routine is highly dependent upon the crater database as well.

2. Robustness

With changing shadows and limited surface knowledge, the lunar environment presents a challenging environment for vision based operations. The crater matching routine must be able to adequately perform at any sun angle that may be encountered, allowing for the possibility of newly formed craters, and be responsive to the variety of terrain characteristics likely to be encountered on the Moon.

3. Processing Speed

This routine must be able to generate a positive crater match within three seconds. This will allow adequate time for processing before and after the crater matching routine in order to generate a position fix for the camera within ten seconds of sensor input. This processing speed requirement will be derived from a 3.60 GHz processor.[1]

[1] On a Pentium 4 CPU

4. Memory Requirements

   This routine must be operable using 1.00 GB RAM and must have a maximum size of 5 MB.[i] This memory requirement will be especially crucial in considering the size of the stored crater database.

These requirements were relevant design parameters in developing and refining the crater matching routine.

3.2.1.2 Definition of Success

A successful crater match is defined as a correlation between the craters in a given sensor image and their absolute positions with greater than 75% of the craters being properly matched to the absolute position.[ii] The correlation for an individual crater is successful when the crater in the original sensor image is matched with the crater of which that sensor image was taken.

A successful crater match does not necessarily equate to a successful position fix. Rather, a successful match only underscores that the sensor image craters were accurately matched to the same craters in the database. In addition, these measures of success only make sense for simulation operations.

3.2.2 Star Tracking Algorithms & the Crater Matching Problem

If the crater matching problem is generalized, it requires a correlation between an image containing distinct data features and a database of these features. Likewise, star tracker algorithms attempt to correlate image features (stars) to a database of these features. In the process of addressing this problem, the author noted this similarity between the crater matching task and the attitude

[i] These performance parameters are not analogous to the capabilities of spacecraft computers.

[ii] Notice that the frequency of false matches is on the order of 0.1%, while the frequency of successful matches must be more than 75%. The remainder of the matches are inconclusive match attempts.

initialization problem for star trackers. Initial research demonstrated that the solution methods of the star tracker algorithms had possible application to the crater matching routine.[1]

This section will characterize the attitude initialization problem, discuss its similarity to the crater matching problem, and present the star tracker algorithm upon which the author chose to base this crater matching algorithm.

3.2.2.1 The 'Lost In Space' Problem

One of the algorithms that a star tracker typically uses to determine the orientation of a spacecraft is an attitude initialization algorithm, often known as a 'lost in space' algorithm. These algorithms are designed to operate without any prior orientation knowledge, and their task is to determine the attitude of a spacecraft to a fairly accurate degree.

Stated more rigorously, the attitude initialization problem is as follows: given a spacecraft with no information about its current attitude other than a series of images taken of star patterns visible from an onboard star tracker, determine the attitude of the spacecraft. For this problem, an onboard database of stars is available for image correlation.

Due to the similarity of this 'lost in space' problem and the crater matching problem, the author researched current solutions to the star tracker attitude initialization problem. The results of this research will be briefly presented in the next section.

3.2.2.2 Star Tracker Algorithms for Crater Matching

The author limited his review of current star tracker algorithms to those that met the following requirements:

[1] Star trackers are attitude sensors frequently used to determine a spacecraft's three dimensional orientation in space.

1. Requires no a priori attitude knowledge.
2. Suitable for relatively small field of view.
3. Memory requirements are modest and processing time is fast (several seconds or less).

As noted by Padgett, Kreutz-Delgado, and Udomkesmalee, the algorithms that met these requirements generally fell into two broad categories; geometric algorithms and pattern-based algorithms.[16]

3.2.2.2.1 The Geometric Approach

One set of attitude initialization algorithms treats the stars in the sensor image as vertices, comparing the angular separation and distance between sets of stars. In other words, geometric attitude initialization algorithms used the angular separation and distances as the features to be compared between the image and the database. While there are many variations of this geometric approach to the attitude initialization problem, the fundamental correlation technique is quite simple:

1. Determine the angular separation and distance between sets of stars in the sensor image.
2. Compare these angular separation and distances to an onboard database of pre-calculated distances and angles.
3. Determine the area of sky with the highest number of correlations between sensor image and database sets. If this number of correlations reaches a certain threshold in relation to the total number of sets in the sensor image, a positive image correlation is generated.

A subset of this geometric approach is a polygonal algorithm, in which the angles and distances are recorded as polygons. Some of the most widely used attitude initialization algorithms are variants of this polygonal approach, typically using sets of triangles to define the arrangement of stars in the sensor image.

One of the more applicable geometric algorithms was one developed by Mortari, Junkins, and Samaan which used a pyramid-based geometric algorithm to solve the lost in space problem.[17] This algorithm uses only the angular separation of sets of four or more stars to form geometric pyramids and attempts to match these pyramids to a star catalogue. In addition, this pyramid approach utilizes a '*k*-vector' approach for accessing the catalogue that does not require searching.[18] As a result, this algorithm is sufficiently fast to be a viable star tracking algorithm. In addition, a confidence is analytically determined for each match, allowing the star matching to be used only within confidence bounds.

3.2.2.2.2 Pattern-Based Approach

The second set of attitude initialization algorithms treats the stars in the sensor image as a cohesive set and generates a pattern based upon the relative arrangement of these stars. These algorithms typically form a pattern by the following process:

1. Select a guide star and determine its closest neighboring star.
2. Orient a grid with a specified cell size on the guide star with the closest neighboring star on the positive x-axis.
3. Generate a pattern from all stars within a certain distance of the guide star. If a star falls within a grid cell, that cell is considered 'full'; otherwise, the cell is considered 'empty.' A simple bit vector is typically utilized to store this information. This vector is termed the 'pattern vector.'
4. Repeat steps 1 through 3 for all possible guide stars in the sensor image, forming a set of image pattern vectors.
5. Compare the set of image pattern vectors to a database of pattern vectors. The area of sky with the highest number of correlations, above some threshold, generates an position match.

This approach is quite different from the geometric approach because each time a correlation occurs, all the stars within a defined radius of the guide star are considered.

3.2.2.2.3 Comparative Cost-Benefit Assessment

There are both costs and benefits to the geometric and pattern-based approaches. The most important aspects of each method are presented in Table 3-1.

Based upon the advantages enumerated in Table 3-1, the author chose to pursue the pattern-based approach to inform a solution to the crater matching problem. Refer to the article "Evaluation of Star Identification Techniques" by Padgett, Kreutz Delgado, and Udomkesmalee for a more detailed comparison and analysis of the attitude initialization approaches presented in this section.[19]

Table 3-1. An Overview of the Costs and Benefits of Star Tracker Algorithms

	Benefits	Costs
Geometric Methods	1 Geometric algorithms are widely proven and implemented 2 Many modifications available that can mitigate some of the downsides to these algorithms	1 Utilizes very limited # of stars from the image to generate correlations 2 Database size is often prohibitive 3 Database size and processing time both increase quickly as sources of error are introduced / greater accuracy demanded 4 Algorithm generates spurious matches and performance degrades noticeably with any significant noise
Pattern-Based Methods	1 Pattern vector includes most of the stars from an image 2 The database can be stored efficiently, resulting in a small database size 3 The correlation can be performed very quickly using a look-up table 4 Initial analysis indicates that the pattern matching approach is quite robust to image noise	1 The pattern-based approach is not as widely implemented as geometric algorithms Therefore not as operationally proven

3.2.2.3 Star Tracker Algorithm Selected for Application

In the comparative studies of these star tracker approaches, specific attention was paid to the robustness of each approach in relation to image noise. This concern with robustness reflects the fact that high levels of image noise are expected in the lunar entry, descent, and landing scenario. As in any software intended for space application, memory requirements, processing speed, and dependability were key considerations in weighing both of these types of algorithms as well.

3.2.3 Initialization of the Crater Matching Routine

3.2.3.1 Terminology

Several terms will be used throughout the remainder of this chapter which will require further explanation:

1. Image Plane – synonymous with the focal plane of the camera. Refer to section 3.2.3.2.
2. Principle Axis – the line through the camera center and perpendicular to the image plane.
3. Principle Point – the point of intersection of the image plane and the principle axis.
4. Data Points – the data points consist of the position of all crater centers and their respective radii for all craters under consideration. The crater centers are in pixels and the crater radii are in units of distance (typically kilometers). These points are either coupled with a sensor image or are those used in the formation of a crater database.
5. Image Data Points – the image data points consist of the position of all crater centers and their respective radii visible in a given sensor image. The crater centers are in pixels and the crater radii are in units of distance (typically kilometers). The author will also refer to the image data points as the 'image'.
6. Pattern Vector – a vector containing information to identify the pattern of nearby craters for a given boresight crater.

3.2.3.2 Unit Systems

The crater matching routine will require only a single frame of reference, the camera's focal plane. This plane will also be referred to as the 'image plane.' Since the given sensor image lies in this plane, and all crater matching routine operations are image operations, this is the only frame required for the routine. The perspective transform algorithm, discussed more fully in Chapter 5, will require additional coordinate frames. These frames will be introduced in section 5.1.2.

Although only one frame of reference is used in the crater matching routine, several different unit systems for the image plane will be utilized in the solution to this problem. Figure 3.3 depicts these unit systems.

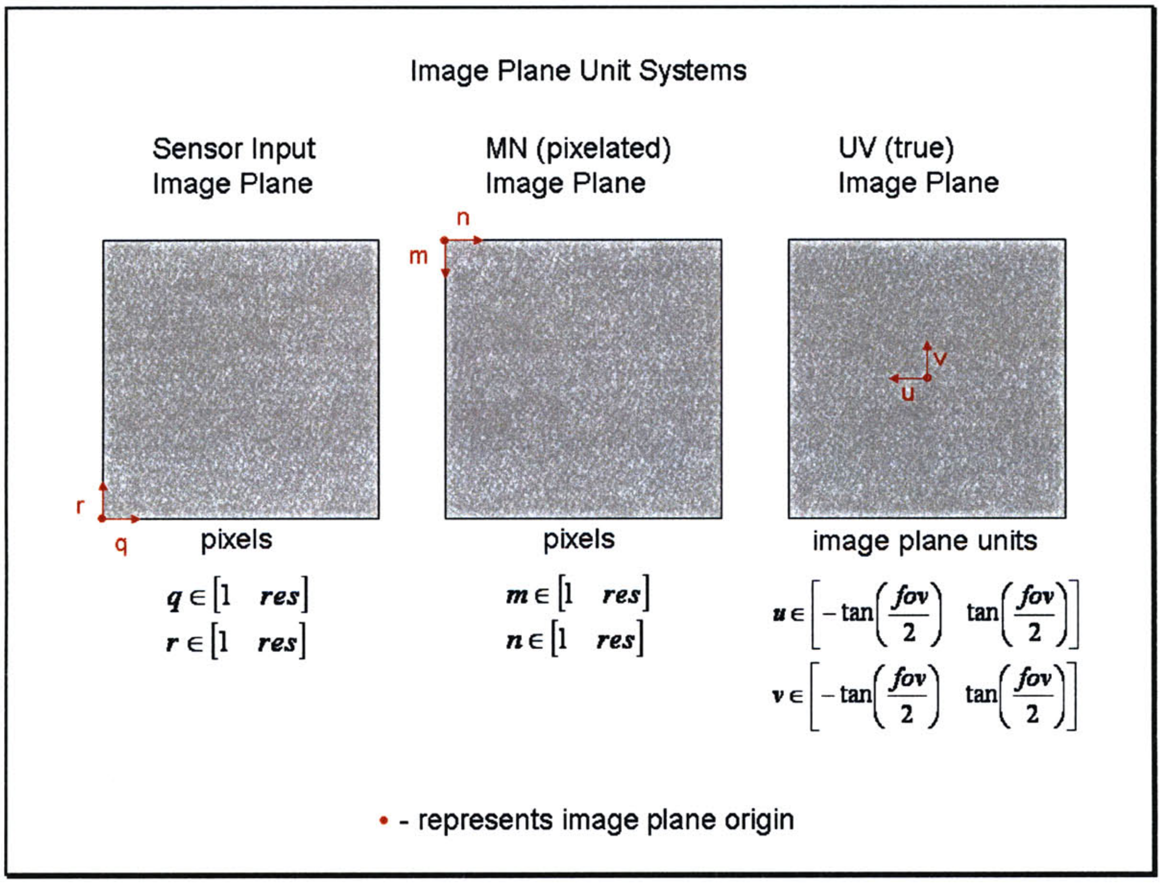


Image Plane Unit Systems
Sensor Input Image Plane
MN (pixelated) Image Plane
UV (true) Image Plane
r
q
n
m
v
u
pixels
pixels
image plane units
$q \in [1 \quad res]$
$r \in [1 \quad res]$
$m \in [1 \quad res]$
$n \in [1 \quad res]$
$u \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right)\right]$
$v \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right)\right]$
● - represents image plane origin

Figure 3.3. The Unit Systems of the Image Plane.

Two abbreviations utilized in Figure 3.3 require further explanation: res refers to the resolution of the original sensor image in pixels, and fov is the camera's field of view in radians.

The initial sensor input 'image' is provided as a data set with the crater centers in pixels and radii in units of distance (typically kilometers). This data is recorded with the origin at the lower left of the image, with the $+q$ axis to the right and the $+r$ axis toward the top of the image. This frame is awkward to use with matrices due to the alignment of the 'rows' (r) and 'columns' (q) in relation to the way rows and columns are stored and referenced in Matlab matrices. As a result, the sensor input is initially converted from the sensor input frame to a more useable unit system.[1] In this unit system, the physical area covered by each pixel is affected by the altitude of the spacecraft.

The pixelated image frame is useful because image data can be intuitively stored in Matlab matrices using this frame. The origin for this frame is at the top left, with the $+m$ direction downward and the $+n$ direction to the right. Any ordered pair $(m \quad n)$ in this frame can also be interpreted as the $(row \quad column)$ location of the image data stored in Matlab matrix form.

The true image frame is analogous to the camera focal plane. It is located at unit distance from the camera along the principle axis of the nadir-pointing camera. The origin of this frame is located at the center of the image with the $+u$ and $+v$ directions aligned left and upward, respectively. Distances in this frame are measured in units of focal length; therefore, the bounds of the true image frame are defined by the field of view of the camera, (fov). See Figure 3.4 . These bounds are:

[1] The sensor input frame is used only because this is the format of the input data. Otherwise, the pixelated and true image planes are much more useful for the crater matching routine.

$$u \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right) \right]$$
$$v \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right) \right]$$

Equation 3.1

The units of the true image plane are defined by the units of distance being used throughout the problem (the 'unit focal distance' of the image plane from the camera will be in the same units as the altitude and the radius of the moon). For the crater matching routine, the focal length will therefore be 1 km. Although this focal length is not representative of a practical camera focal length, the perspective transform equations of Equation 5.27 can be re-derived for any focal length.

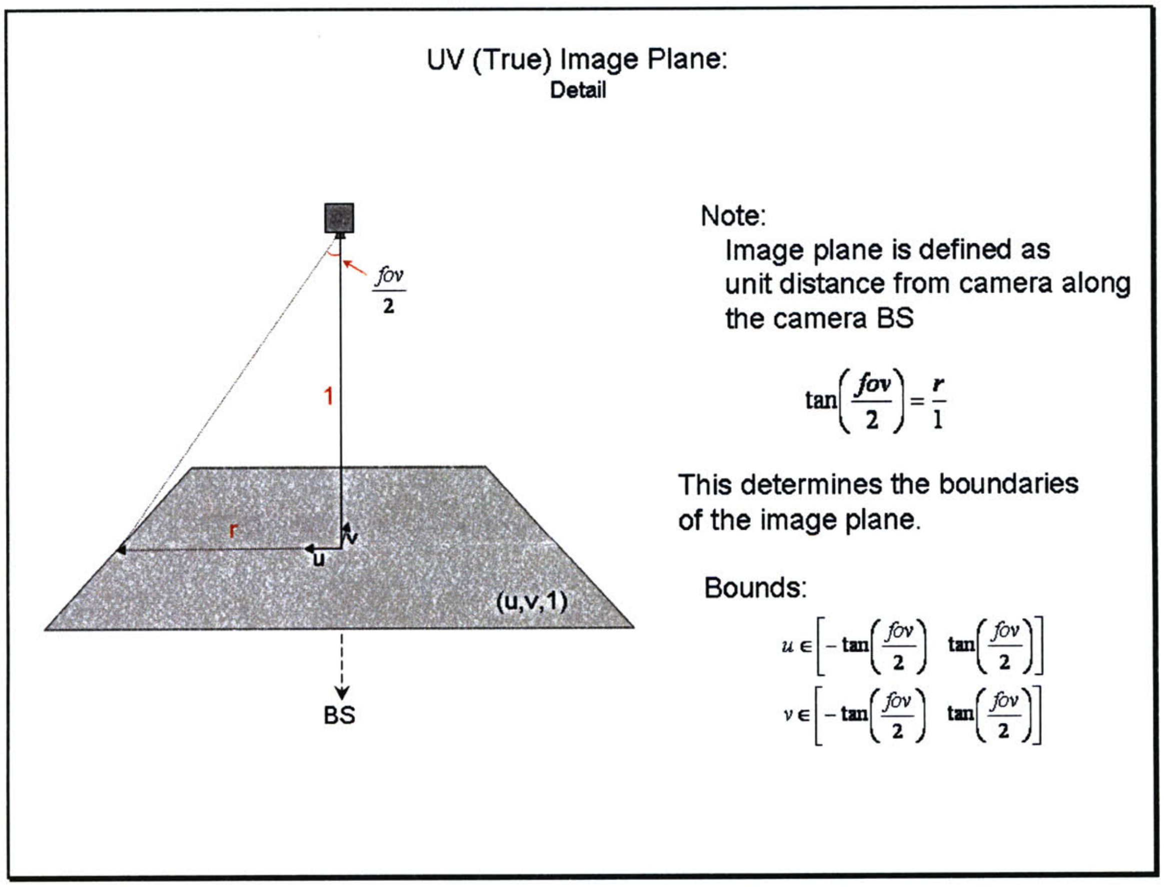

UV (True) Image Plane:
Detail
fov/2
1
r
u
v
(u,v,1)
BS
Note:
Image plane is defined as unit distance from camera along the camera BS
$\tan\left(\frac{fov}{2}\right) = \frac{r}{1}$
This determines the boundaries of the image plane.
Bounds:
$u \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right) \right]$
$v \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right) \right]$

Figure 3.4. A Detail of the UV (True) Image Plane.

3.2.3.3 Unit Conversions

Based on the image plane unit systems described in section 3.2.3.2, conversions between these units can be implemented.

3.2.3.3.1 Sensor Input Image Plane Units $(q \quad r)$ to Pixelated Image Plane Units $(m \quad n)$

The unit conversion from the sensor input image plane to the pixelated image plane entails a rotation of the axes counter-clockwise by -90° about $[0 \quad 0 \quad 1]$ and a translation of the origin from the lower left to the upper left of the image. That is,

$$\begin{bmatrix} m \\ n \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} q \\ r \\ 0 \end{bmatrix} + \begin{bmatrix} res \\ 0 \\ 0 \end{bmatrix}$$

Equation 3.2

This rotation and translation is identical to modifying the sensor input image plane order pairs $(q \quad r)$ as follows:

$$\begin{aligned} m &= res - r \\ n &= q \end{aligned}$$

Equation 3.3

3.2.3.3.2 Pixelated Image Plane Units $(m \quad n)$ to True Image Plane Units $(u \quad v)$

The unit conversion from the pixelated image plane to the true image plane requires a 180° counter-clockwise rotation of the axes about $[1 \quad -1 \quad 0]$ and a translation of the origin from the upper left to the center of the image. That is,

$$\begin{bmatrix} u \\ v \\ 0 \end{bmatrix} = \tan\left(\frac{fov}{2}\right) \cdot \left(\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} * \left(\frac{2}{res}\right) \cdot \begin{bmatrix} m \\ n \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right)$$

Equation 3.4

The $\{m \quad n\}$ image points are converted from pixels to true image frame units before the rotation is applied. This unit conversion (partially factored out on the right side of Equation 3.4) is

$$pix2uv = \frac{2 \cdot \tan\left(\frac{fov}{2}\right)}{res}$$

Equation 3.5

The rotation and translation for this unit conversion is identical to modifying the pixelated image plane ordered pairs $(m \quad n)$ as follows:

$$u = \tan\left(\frac{fov}{2}\right) - n \cdot pix2uv$$
$$v = \tan\left(\frac{fov}{2}\right) - m \cdot pix2uv$$

Equation 3.6

3.2.3.4 Inputs, Outputs, & Algorithm Constants

The inputs, outputs, and constants for the crater matching routine are listed below. Several of these parameters have not yet been explained in full. These parameters will be detailed as they are necessary throughout the crater matching process. For now, it will suffice to say that these are the inputs, outputs, and constants necessary to the crater matching routine.

Inputs:

1. Sensor image data points in the original image frame.
2. Database of stored pattern vectors for lunar craters.
3. Camera altitude (alt) above the lunar surface.
4. Camera field-of-view (fov) and resolution (res).
5. Camera attitude (att).

User Defined Constants:

1. Pattern Radius (PR).
2. Buffer Radius (BR).
3. Grid Number (g).

Algorithm Constants:

1. Radius of the Moon, $r_m = 1737.4\ km$

Outputs:

1. Latitude and longitude of correlation between the sensor image and the crater database.

3.2.3.5 Assumptions

The author made various assumptions in the implementation of the crater matching routine. Each assumption, the necessity of the assumption, and the relative validity of the assumption are discussed here.

3.2.3.5.1 Camera is an ideal camera

The camera is assumed to be a point camera without noise or aberration. This assumption does not accurately model physical hardware. However, the validity of this assumption is based on the understanding that a camera can be modeled and a significant amount of the image noise can be filtered out. If this filtering occurs before the sensor image is inputted into the perspective transform algorithm, the ideal camera assumption is relatively valid. At the same time, analyses examining algorithm performance with varying amounts of pixel error added to the original sensor image are certainly warranted to better understand the robustness of the overall crater matching technique.

3.2.3.5.2 Craters are circular

The generation of a pattern vector for the crater matching routine assumes that all craters are circular. This allows a crater to be uniquely identified by the position of the crater center and its radius. In general, lunar craters are elliptical, not circular. However, image pre-processing will allow elliptical craters to be modeled as circular craters for the purpose of crater matching.

In the future, the capability to model caters as ellipses may prove beneficial to the accuracy of the crater matching algorithm. In this case, the storage of the crater center in a defined grid cell may not be the most beneficial means of pattern generation.

3.2.3.6 Algorithm Notes

The explanation of the crater matching routine that is contained in Chapter 4, Chapter 5, and Chapter 6 will benefit from the use of an example data set to be processed. For the remainder of this chapter, the data set plotted in Figure 3.5 will be used as an example data set.

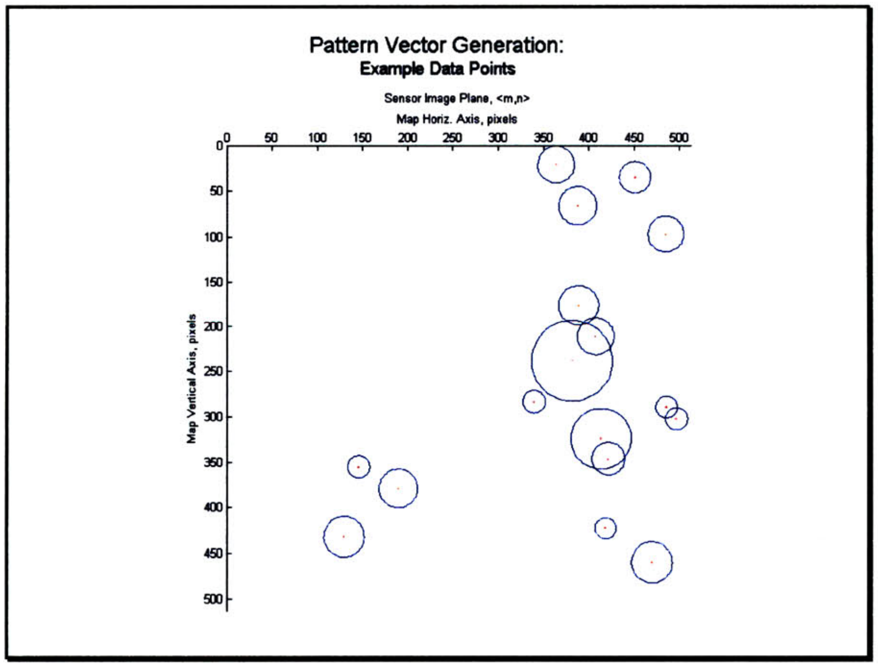


Figure 3.5. Example Data Points for Pattern Vector Generation.

The data points in Figure 3.5 are craters that were recognized by image processing software in use by the Charles Stark Draper Laboratory for lunar mission design and were derived from Clementine images.[1] This data was obtained by a simulated camera with the following initial conditions:

$$\begin{aligned} &fov = 90^\circ \\ &res = 512 \quad pixels \\ &alt = 100 \quad km \\ &att = \begin{bmatrix} 0^\circ & 0^\circ & 0^\circ \end{bmatrix} \\ &lat = 80^\circ \\ &long = 85^\circ \end{aligned}$$

Equation 3.7

where lat and $long$ are the latitude and longitude of the camera location. Other initial conditions that will be needed for these example problems are:

$$\begin{aligned} &PR = 50 \quad km \\ &BR = 10 \quad km \\ &g = 24 \end{aligned}$$

Equation 3.8

where g is the number of rows and columns of the g x g grid used to create the pattern vector. Reference section 4.3.1 for a more detailed explanation for g. The pattern radius (PR) and buffer radius (BR) have not yet been explained. However, these parameters will be introduced in more detail as they are needed in the pattern vector generation process.

Pattern vector generation will also require a crater upon which the remainder of the pattern generation process can be based. This crater will be centered at:

[1] The Deep Space Program Science Experiment (DSPSE), more commonly known as 'Clementine', was a 1994 mission to test small sensors in prolonged space exposure. This mission was also tasked with mapping the lunar surface. The data sets and imagery from this mapping process is widely available online. Reference http://nssdc.gsfc.nasa.gov/planetary/clementine.html

$$BS_{uv} = \begin{bmatrix} -0.328 & -0.105 \end{bmatrix}$$

Equation 3.9

in true image frame units. The crater at this location in the image is highlighted by a filled green dot in Plot A of Figure 4.1, and will be referred to as the ‘boresight crater’.

Chapter 4

The Pattern Vector and the Crater Matching Routine

The crater matching routine is tasked with matching the craters in an image of the lunar surface to a database of lunar craters. This matching is accomplished by examining the pattern of nearby craters for any given crater in the image. A feature vector – called the pattern vector – will define the pattern of nearby craters for a given crater. This pattern vector will enable the correlation between the crater in the sensor image and a database of lunar craters. This correlation will be accomplished using a process similar to that in use by pattern-based star tracker algorithms described in section 3.2.2.2.2. The remainder of this chapter will describe the generation of the pattern vector, while a specification of the process of correlation between the pattern vector and the crater database will be reserved for Chapter 6.

A pattern vector is generated for a given sensor image so that the pattern of nearby craters can be matched to a database of patterns. However, the data points in the sensor image are not referenced to an inertial coordinate system. As a result, these image data points can be, and often are, rotated from data points stored in a database of lunar craters. This rotation of the image data points from a nominal value is the 'rotational offset' of the image data points. In a similar manner, the image data points may also be translated from a nominal position. This difference between the image data points' position and the nominal position is the 'translational offset'.

The pattern generation process must be invariant to rotational and translational offsets in the data points. If two sets of data are given and the second set is identical to the first except that the second set is rotated and/or translated in relation to the first, the pattern vector generated for both sets of data points must be the same. Therefore, it is necessary to reference the image data points in some way so that any translational and rotational offset in the image will not cause an incorrect correlation of a pattern vector to the database.

Take the data points plotted in Figure 4.1, for example.

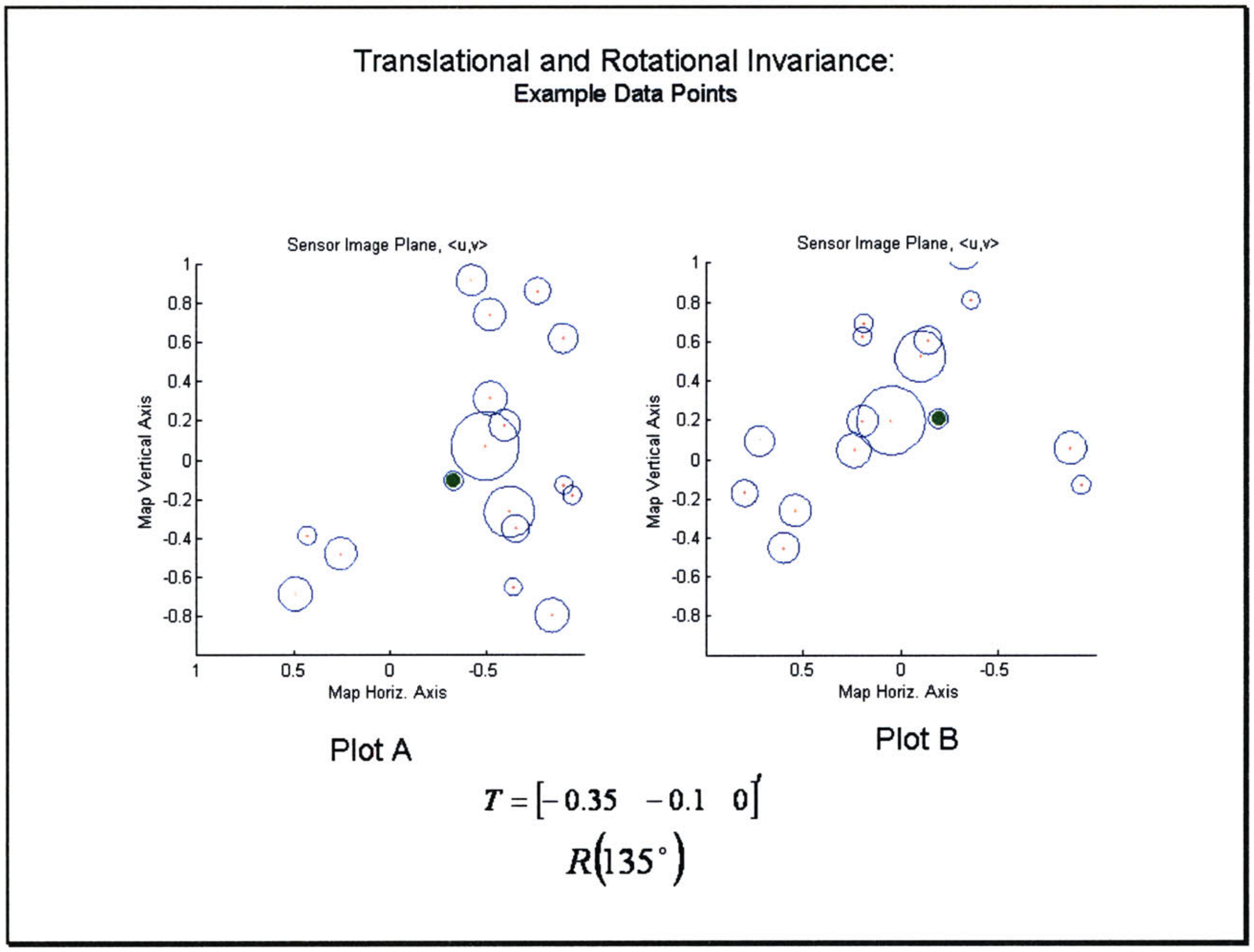


Figure 4.1. Translational and Rotational Invariance.

The data points displayed in both Plot A and B are identical to each other save for a rotation and a translation between the sets of data. Specifically, the rotation angle between the first and second data sets is a rotation of 135° clockwise about the origin, and a translation of $[-0.35 \quad -0.1 \quad 0]'$ true image frame units between the data sets. The new data set is calculated as follows:

$$Data_2 = R * Data_1 + T$$

Equation 4.1

where R represents a rotation matrix generated from the rotation angle and T represents the translation vector.

The situation presented in Figure 4.1 is expected to be encountered frequently in the crater matching task. Therefore, the process of generating a pattern for a desired boresight crater must account for the variation in the orientation and translation of the data points in relation to the desired boresight. Establishing the translational and rotational reference will comprise the first two steps in the pattern vector generation process.

The generation of a pattern vector will require three steps:

1. Establish a translational reference for the data points.
2. Establish a rotational reference for the data points.
3. Generate a pattern vector based upon the rotational and translational references.

These steps are shown in Figure 4.2, with some of the steps broken into component steps. The flow diagram of Figure 4.2 also presents the inputs and outputs of each step. These inputs and outputs will be discussed in more detail as the pattern vector generation is explained in the remainder of this section.

The remainder of this section will present the pattern vector generation in its component steps. Section 4.1 examines a means to establish a translation reference, section 4.2 determines a rotational reference, and section 4.3 generates a pattern vector based upon a given set of data points. This section will describe the pattern generation process, but correlation algorithms will not be addressed until Chapter 6.

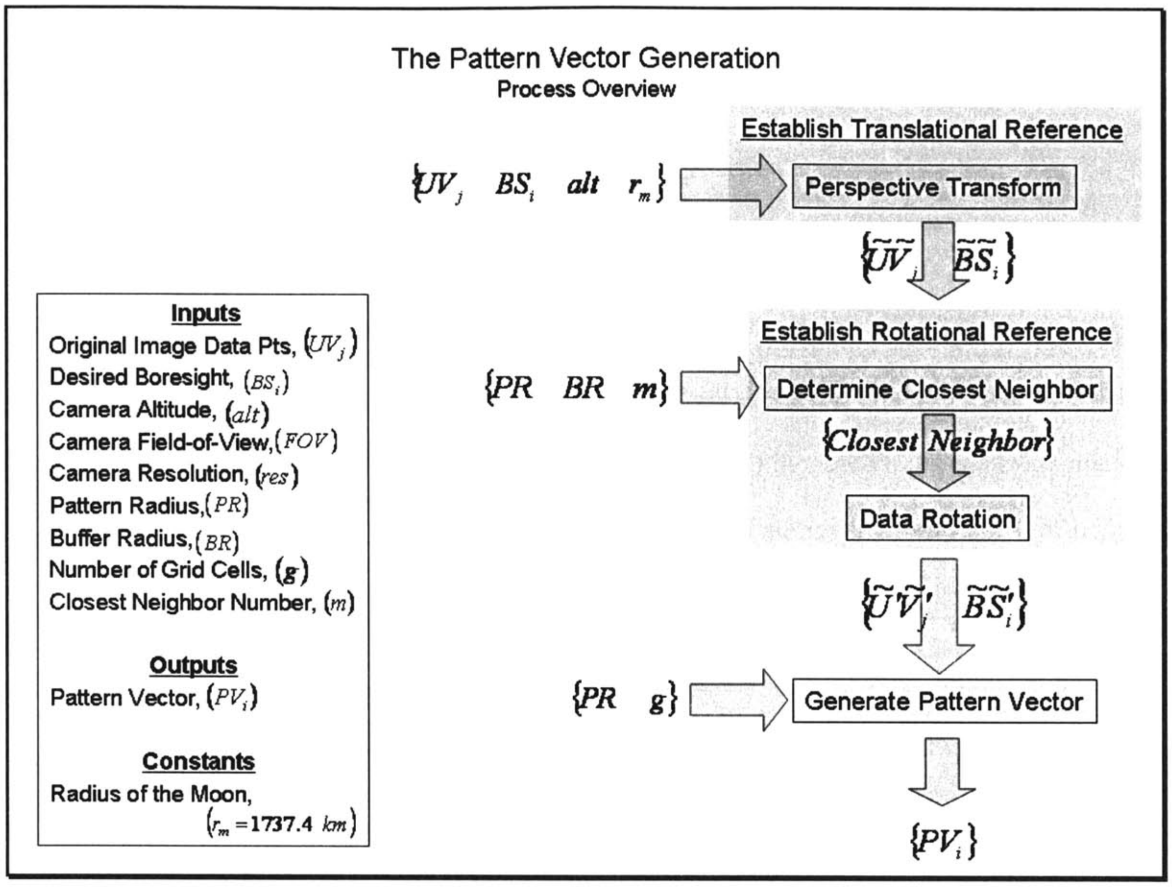

The Pattern Vector Generation
Process Overview
Establish Translational Reference
$\{UV_j \quad BS_i \quad alt \quad r_m\}$
Perspective Transform
$\{\widetilde{UV}_j \quad \widetilde{BS}_i\}$
Establish Rotational Reference
$\{PR \quad BR \quad m\}$
Determine Closest Neighbor
$\{Closest\ Neighbor\}$
Data Rotation
$\{\widetilde{UV}'_j \quad \widetilde{BS}'_i\}$
$\{PR \quad g\}$
Generate Pattern Vector
$\{PV_i\}$
Inputs
Original Image Data Pts, (UV_j)
Desired Boresight, (BS_i)
Camera Altitude, (alt)
Camera Field-of-View, (FOV)
Camera Resolution, (res)
Pattern Radius, (PR)
Buffer Radius, (BR)
Number of Grid Cells, (g)
Closest Neighbor Number, (m)
Outputs
Pattern Vector, (PV_i)
Constants
Radius of the Moon,
$(r_m = 1737.4\ km)$

Figure 4.2. Overview of the Pattern Vector Generation Process.

4.1 Establish the Translational Reference

A pattern vector must be invariant to the Euclidean translation of a set of data points. To allow this invariance, a point of reference will need to be identified such that data point translation will not affect the pattern generation outcome. The simplest choice for this point of reference is the boresight crater itself. If the boresight crater is always translated to the center of the image before generating the pattern of nearby craters, the generation of this pattern will be invariant to any translational offsets.

This process of translating the boresight crater to the center of an image before generating a pattern vector will be referred to as establishing a translational reference. Two methods of establishing such a translational reference will be presented, a Euclidean translation and the perspective transform.

4.1.1 Euclidean Translation

One method of establishing a translational reference is to use a simple Euclidean translation to center the desired boresight in the image. A Euclidean translation simply adds (or subtracts) a horizontal and vertical distance to all ordered pairs. This has the effect of translating the entire data set by a specified step in a certain direction. Reference Figure 4.3.

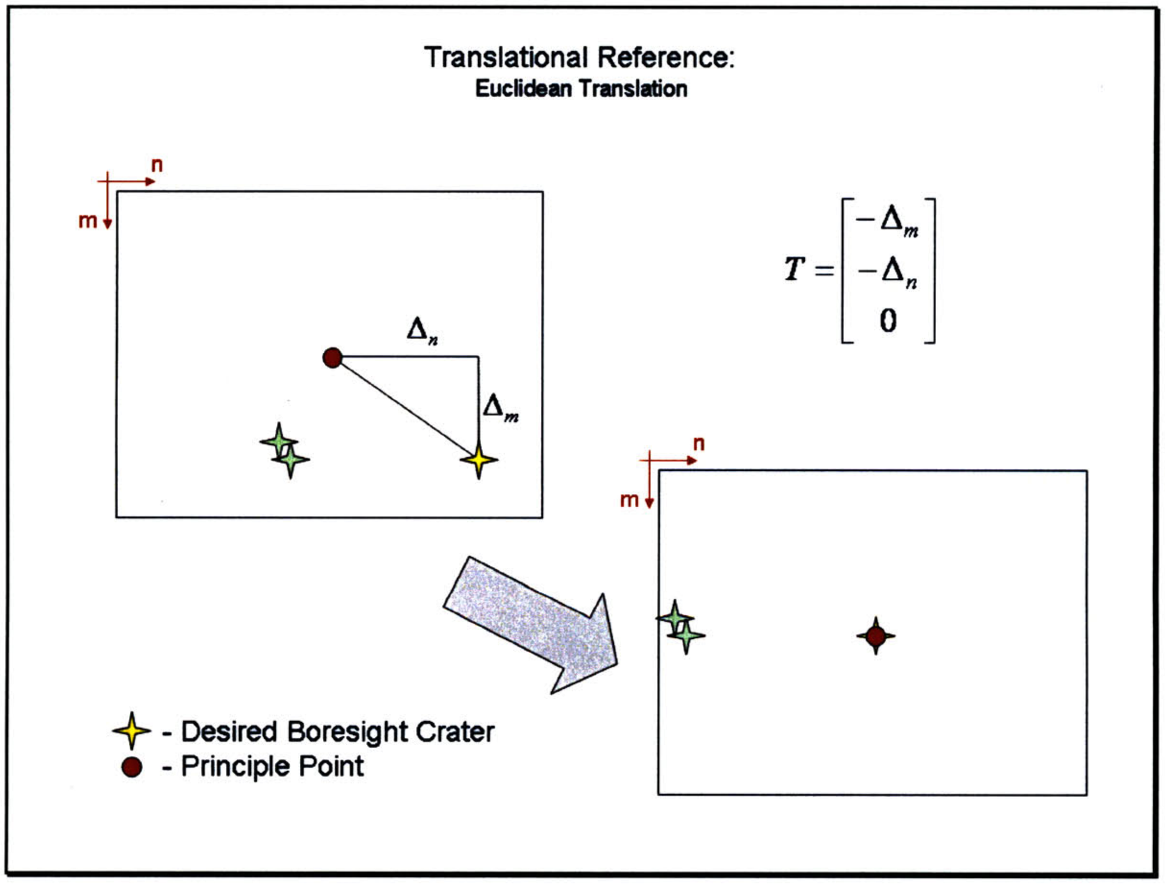

Translational Reference:
Euclidean Translation
n
m
Δ_n
Δ_m
$T = \begin{bmatrix} -\Delta_m \\ -\Delta_n \\ 0 \end{bmatrix}$
n
m
- Desired Boresight Crater
- Principle Point

Figure 4.3. Euclidean Translation: An Example.

Figure 4.3 shows the translation of a desired boresight crater to the principle point. As mentioned, this translation can be determined geometrically, and is implemented by simply moving all data points by a specified amount in a specified direction. In other words, the desired translation T is

$$T = \begin{bmatrix} -\Delta_m & -\Delta_n & 0 \end{bmatrix}'$$

Equation 4.2

where $\{\Delta_m \quad \Delta_n\}$ represent the original offset of the desired boresight crater from the principle point in the m and n directions respectively. This desired translation is applied to the existing data set by

$$Translated\ \ Data = Data + T$$

Equation 4.3

If the given set of data points was derived from an image on a plane parallel to the camera's focal plane, this Euclidean translation is an acceptable image translation. The Euclidean translation of the desired boresight crater to the center of the image will be representative of an image where the camera's principle point coincides with the desired boresight crater.[1] However, if the original image includes any perspective effects, the set of data points resulting from a Euclidean translation will not match the correct positions of the craters 'seen' by the nadir-pointing camera when its principle point is aligned with the desired boresight. The Euclidean translation results will not 'look' like they should because the perspective of the problem has been ignored.

In general, the image is affected by perspective in the crater matching routine. The camera in the crater matching routine takes an image of the lunar surface, which has significant curvature. Due to this curvature, the lunar surface cannot usually be considered planar and parallel to the camera's focal

[1] Given a nadir-pointing camera, which is assumed for this algorithm.

plane.[i] In the case of the crater matching routine, a Euclidean translation will only provide a rough approximation of the image if it had been taken with the desired boresight at the principle point.

Consider the illustrations in Figure 4.4 and Figure 4.5.

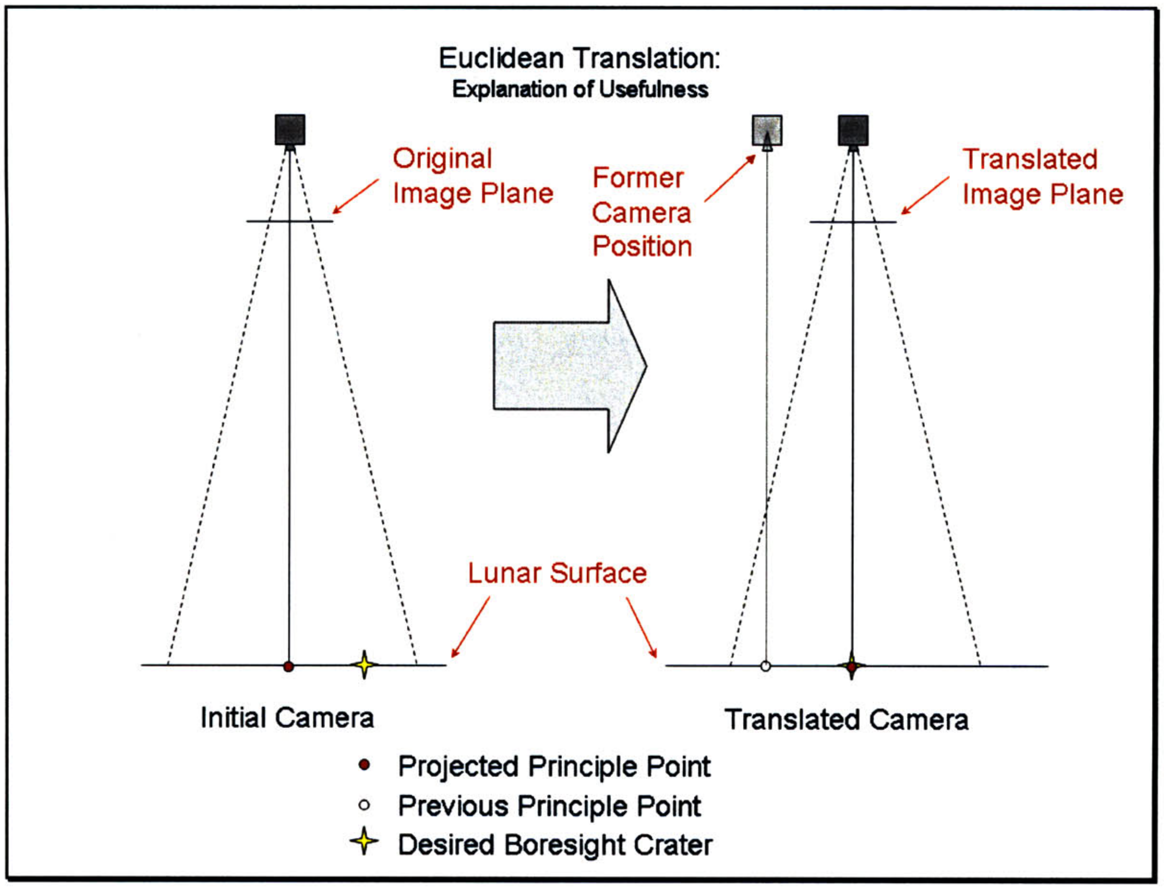


Figure 4.4. Euclidean Translation: An Explanation of Its Usefulness.

Figure 4.4 depicts the situation where a Euclidean translation is useful. The lunar surface in Figure 4.4 is parallel to the image plane. For this case, the relative position of craters in the original and translated image planes will accurately correlate to the position of the craters on the lunar surface. That is, the projection of the craters on the lunar surface onto the image plane is merely a scaling of the crater positions. As a result, there are no warping or skewing effects on the image data points

[i] There are certain cases where the Euclidean translation may be applicable to the crater matching routine. These cases occur when the curvature of the lunar surface has minimal impact upon the sensor image. For example, low altitudes and small fields-of-view both decrease the impact of the Moon's curvature on a sensor image.

from the camera's perspective. Therefore, a Euclidean translation will accurately simulate the sensor image as if it had been taken over another point on the lunar surface.

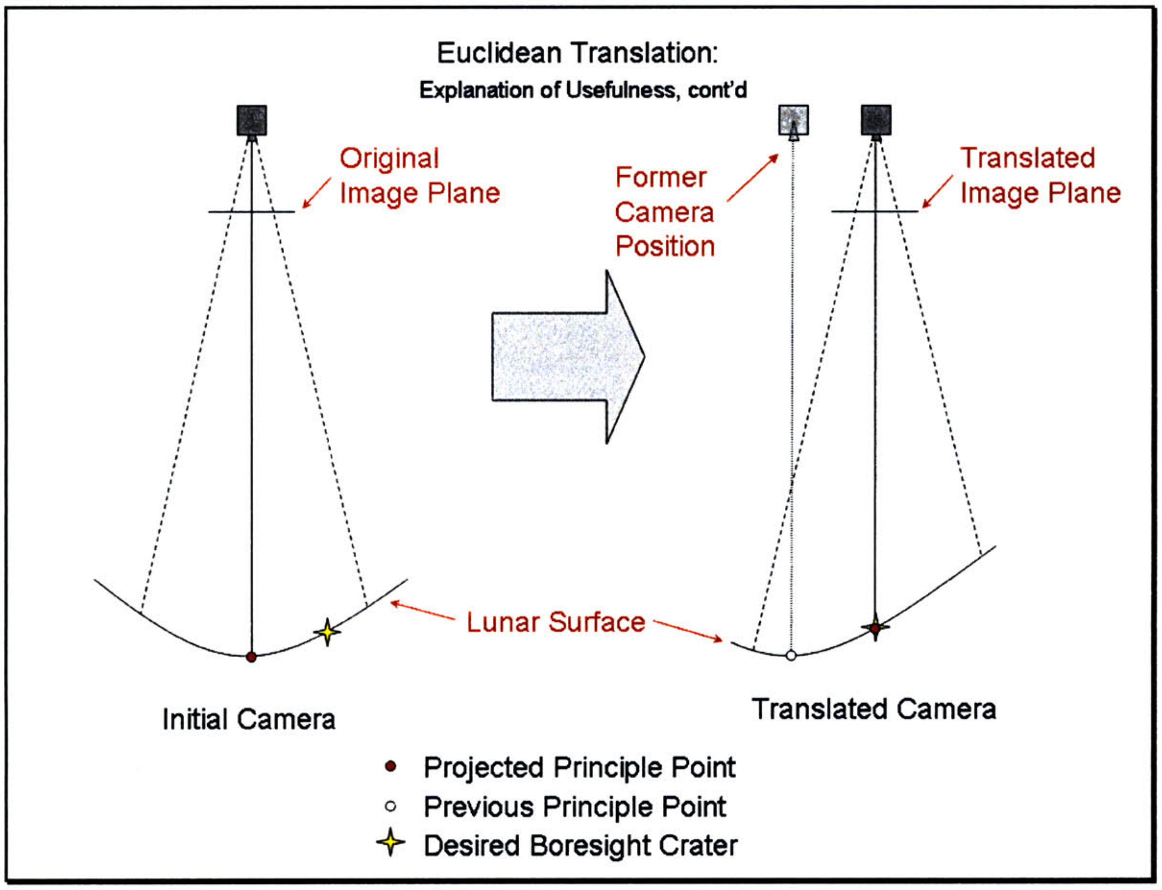


Figure 4.5. Euclidean Translation: An Explanation of Its Usefulness cont'd.

Figure 4.5 depicts a situation where the image will include effects from the camera's perspective in the data points. For this situation, images taken at different location over the lunar surface will appear skewed or warped. The images appear different because the lunar surface and the camera's image plane are not parallel to one another. As a result, any image taken by the camera in this situation will include the effects of perspective.

For the situation presented in Figure 4.5, a Euclidean translation of the original data points will not accurately describe the image generated by the translation of the camera position. As mentioned above, simply translating the data points from the original image does not account for the change in effects caused by the camera's perspective of the ground.

The perspective transformation which is introduced in the following section, section 4.1.2 allows a solution to the problem of data point translation while accounting for perspective.

4.1.2 Perspective Transformation

The perspective transformation is a process that allows image data points to be modified so that the principle point of the image appears shifted to coincide with the desired boresight. The transform accounts for the perspective of the problem, which in this case is the spherical lunar surface. The perspective transformation is derived from the relationship between the points on the lunar surface and their three dimensional position relative to the camera's image plane.

Figure 4.6 gives an overview of what the perspective transform will accomplish.

The perspective transform converts data points in the original image plane $(u \quad v)$ to data points in the transformed image plane $(\tilde{u} \quad \tilde{v})$, as shown in Figure 4.6. As a result, the transform will effectively modify the original data points so that they appear as if they were 'seen' by the camera when it was centered over the desired boresight. The perspective transform equation is:

$$c_j \cdot \begin{bmatrix} \tilde{u}_j \\ \tilde{v}_j \\ 1 \end{bmatrix} = \begin{bmatrix} R_i & -T_i \end{bmatrix} * \begin{bmatrix} d_j \cdot \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} \\ 1 \end{bmatrix}$$

Equation 4.4

where c_j and d_j are the z' components[i] of the crater positions on the lunar surface in the transformed and original camera frames, respectively. R_i and T_i are the rotation and translation matrices between the original and transformed camera frames. The set of points $\{u_j \quad v_j \quad 1\}$ are the original image plane points, whereas the points $\{\tilde{u}_j \quad \tilde{v}_j \quad 1\}$ are the transformed image plane points.

The perspective transform problem, its solution, and the utility of the perspective transform in the crater matching task will be discussed in more detail in Chapter 5.

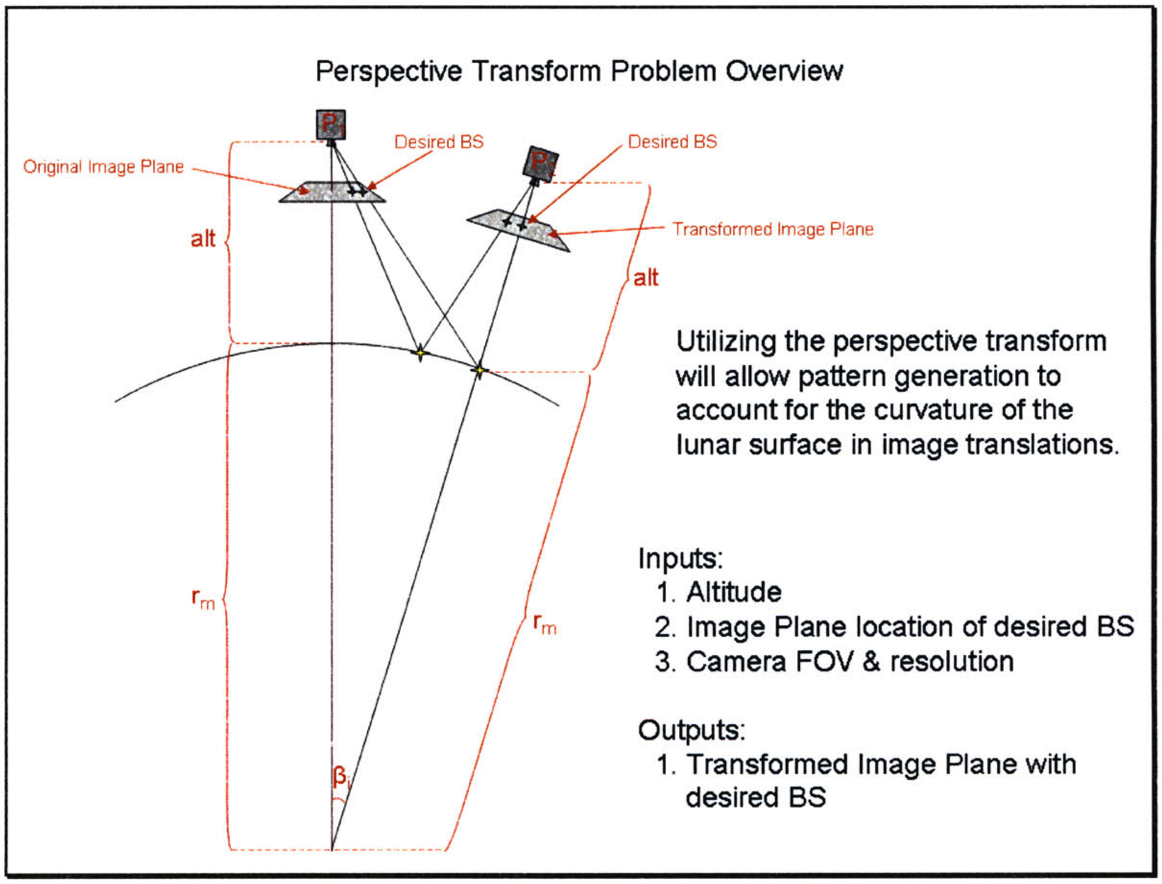

Perspective Transform Problem Overview
Original Image Plane
Desired BS
Desired BS
Transformed Image Plane
alt
alt
r_m
r_m
β_i
Utilizing the perspective transform will allow pattern generation to account for the curvature of the lunar surface in image translations.
Inputs:
1. Altitude
2. Image Plane location of desired BS
3. Camera FOV & resolution
Outputs:
1. Transformed Image Plane with desired BS

Figure 4.6. Overview of the Perspective Transform.

[i] For an explanation of the camera frame and the z' axis, see section 5.1.2 .

4.1.3 Data Point Translation Example

It is instructive to present an example of the translation of the desired boresight to the principle point of the image. An example of both Euclidean translation and the perspective transform will be given.

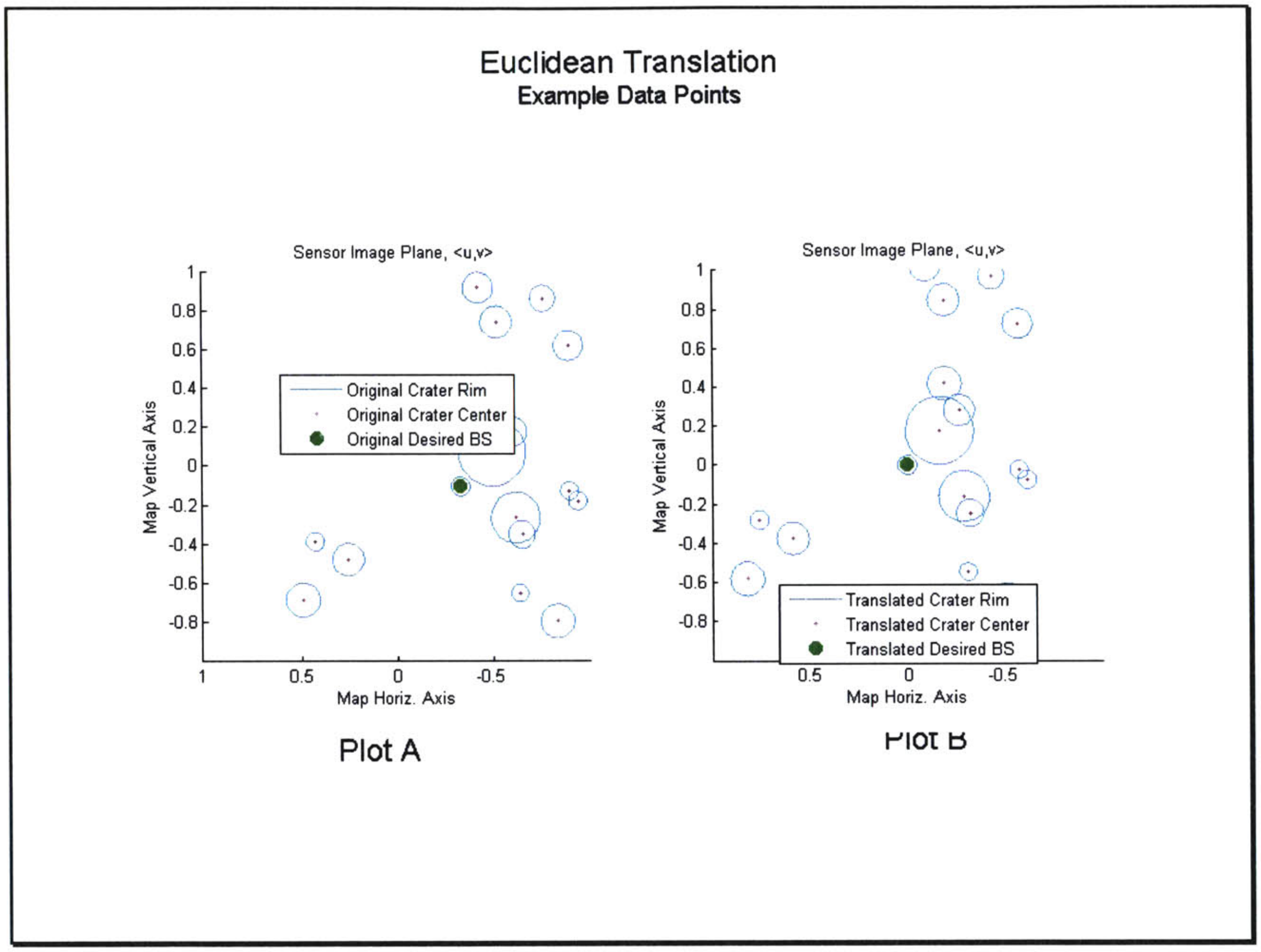


Figure 4.7. Euclidean Translation of the Example Data Points.

Figure 4.7 presents the Euclidean translation of the desired boresight, originally located at $BS_{uv} = [-0.328 \quad -0.105]$ (according to Equation 3.9). The desired boresight is translated to the principle point of the image by a Euclidean translation. This translation does not account for any perspective effects. In this plot and the following plots, the desired boresight is highlighted in green.

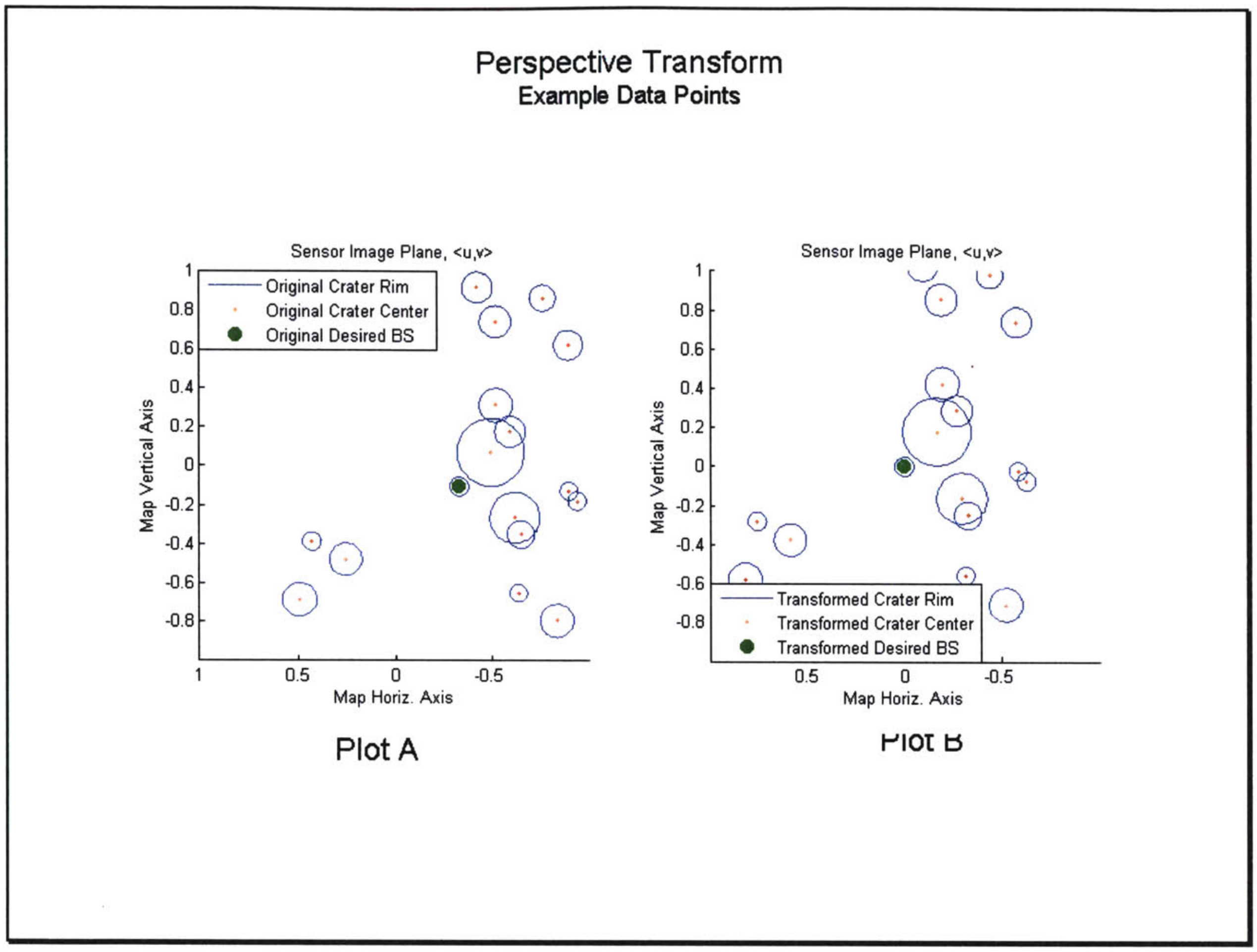


Figure 4.8. Perspective Transform of the Example Data Points.

Figure 4.8 presents the perspective transform that translates the same desired boresight crater to the principle point of the image. In this case, the translation does account for the perspective of the problem. Although it is hardly noticeable, the relative position of the craters after the transform is slightly modified. In order to highlight the differences, Plot B from both Figure 4.7 and Figure 4.8 are overlaid in Figure 4.9.

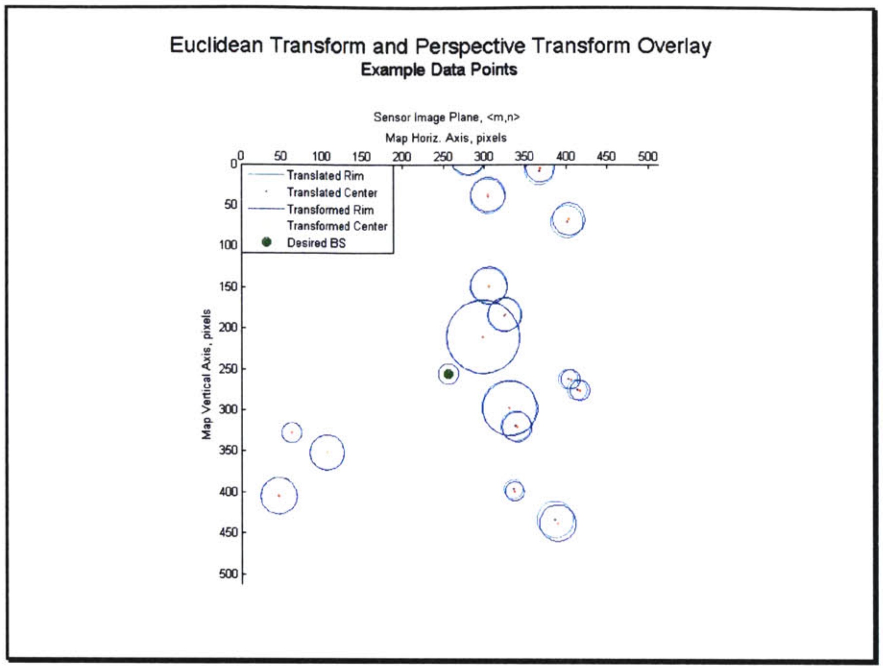


Figure 4.9. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points.

The slight difference between these two plots is due to the fact that Plot B from Figure 4.8 accounts for the perspective of the problem, whereas Plot B from Figure 4.7 does not.

See section 5.7 for a more detailed comparative analysis of these two translation methods.

4.2 Establish the Rotational Reference

In addition to the translational reference established in section 4.1, a rotational reference for the pattern vector is also required. The rotational reference defines a method of orienting the data points such that the pattern formed from nearby craters is always formed at the same rotational orientation about the boresight crater. This section will characterize the rotational reference, and introduce a method of establishing and utilizing this rotational reference in the crater matching routine.

Figure 4.10 applies the concept of rotational reference to a simple spatial orientation brain teaser.

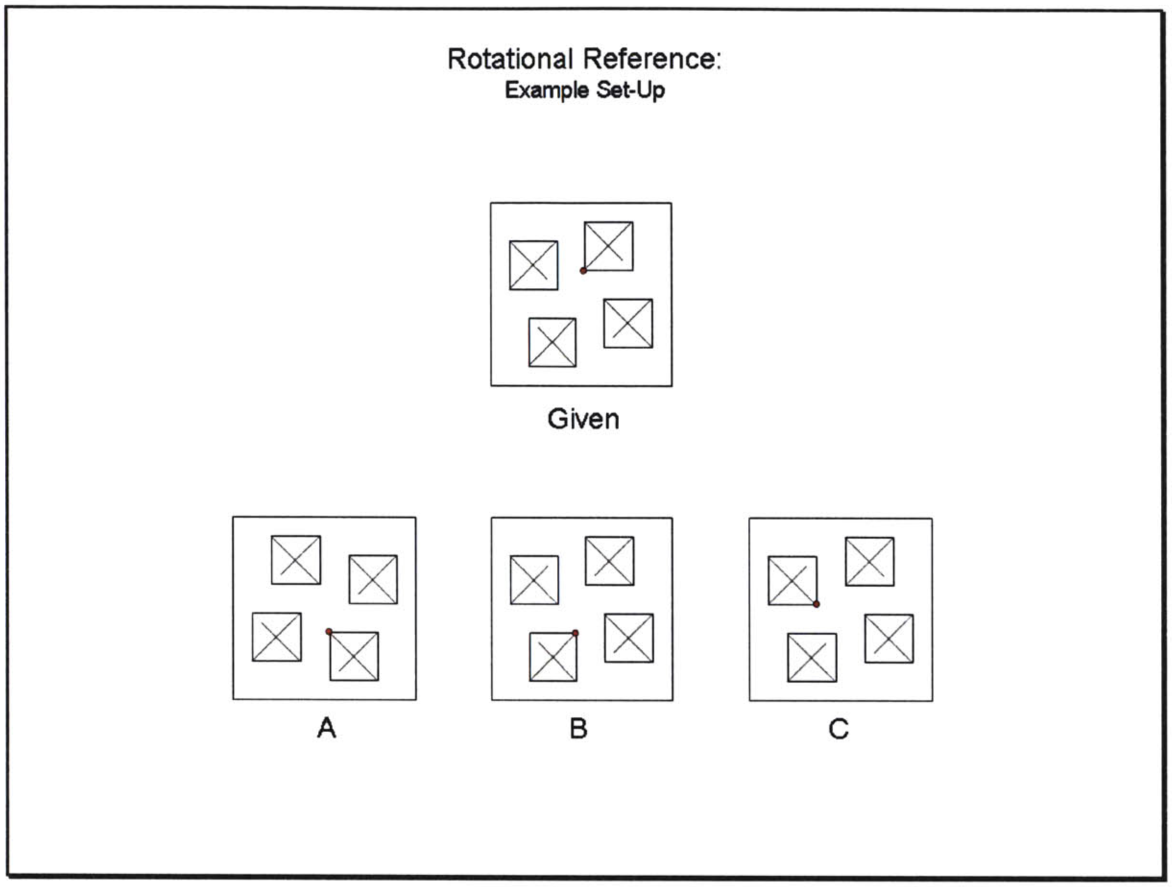


Figure 4.10. Set-up of the Rotational Reference Brain Teaser Example.

Based upon the model given in Figure 4.10, the reader must determine which of the three choices is a rotated – but identical – copy of the model. This problem is somewhat difficult as presented. However, it becomes quite rudimentary if each of the choices is rotated so that they have the same orientation relative to one another. See Figure 4.11.

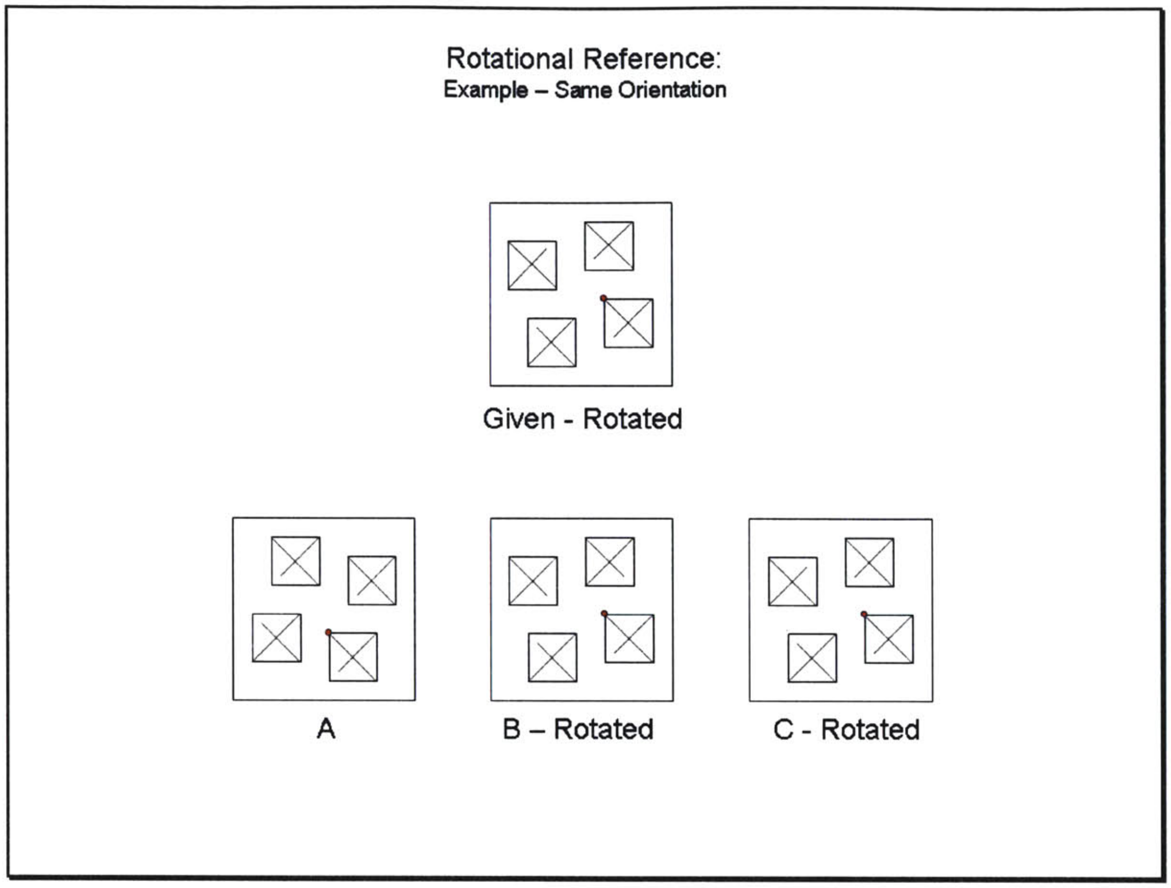


Figure 4.11. Reoriented Rotational Reference Brain Teaser Example.

It is quite simple to determine that choice C is the correct answer.

By rotating the choices in this example, the pattern of squares and lines within each choice is able to be directly compared to the original. The process of rotating each choice in the example to a nominal orientation greatly simplifies the task of matching one of the choices with the original. Similarly, the establishment of a rotational reference in the crater matching problem allows for data points with different original orientations to be compared directly with each other much more easily.

The combination of a defined translational and rotational reference will ensure that each time a pattern of nearby craters is generated for a given crater, that it is done so in a standardized and repeatable manner. It is crucial for the pattern generation to be repeatable – as the patterns generated for sensor images must be able to be compared to patterns stored in the database.

In the crater matching problem, the 'direction of travel' of the camera relative to the lunar surface is unknown. Therefore, although the orientation of the camera and image in relation to the spacecraft is known, the orientation of the camera and image in relation to any absolute frame is unknown. If the absolute orientation of the camera were precisely known, a rotational reference would be unnecessary, as the inertial frame of reference for the sensor image would be known. In this case, all images (for both the database and the sensor image) could be rotated such that the lunar north is at the top of the image, and the pattern generated with this absolute orientation.

4.2.1 Reference Point Selection

In order to establish a rotation that can be applied to pattern generation in general, some point of reference in each image must be found that allows this image rotation to occur in a predictable fashion. For the example in Figure 4.10 and Figure 4.11, the point of reference selected was the red dot in the corner of each choice. For each choice, the dot is unique (there is only one dot per choice) and easily identifiable. Each choice in this example was re-oriented so that the red dots all appeared in the same relative position. As a result, the final comparison between the choices and the original was very simple.

Although there are a variety of possibilities for this point of reference in the crater matching problem, the author has chosen the closest neighboring crater as the simplest choice for a point of reference. The data points will be rotated so that this point of reference is always on the same horizontal level as the boresight crater and to the right of the boresight crater. In other words, the crater nearest the boresight crater will serve as the reference point in establishing a rotational reference for a given boresight crater. The closest neighboring crater is a unique and easily identifiable point of reference within the image. The choice of the closest neighboring crater as the rotational reference was based in part on work presented in the article "A Grid Algorithm for Autonomous Star Identification" by Padgett and Kreutz-Delgado.[20] In order to determine the closest neighboring crater, the normalized distance between the boresight crater and all other crater centers is calculated. Based upon these normalized distances, the nearest neighbor to the boresight crater is selected as the rotational reference point for a given set of data points.

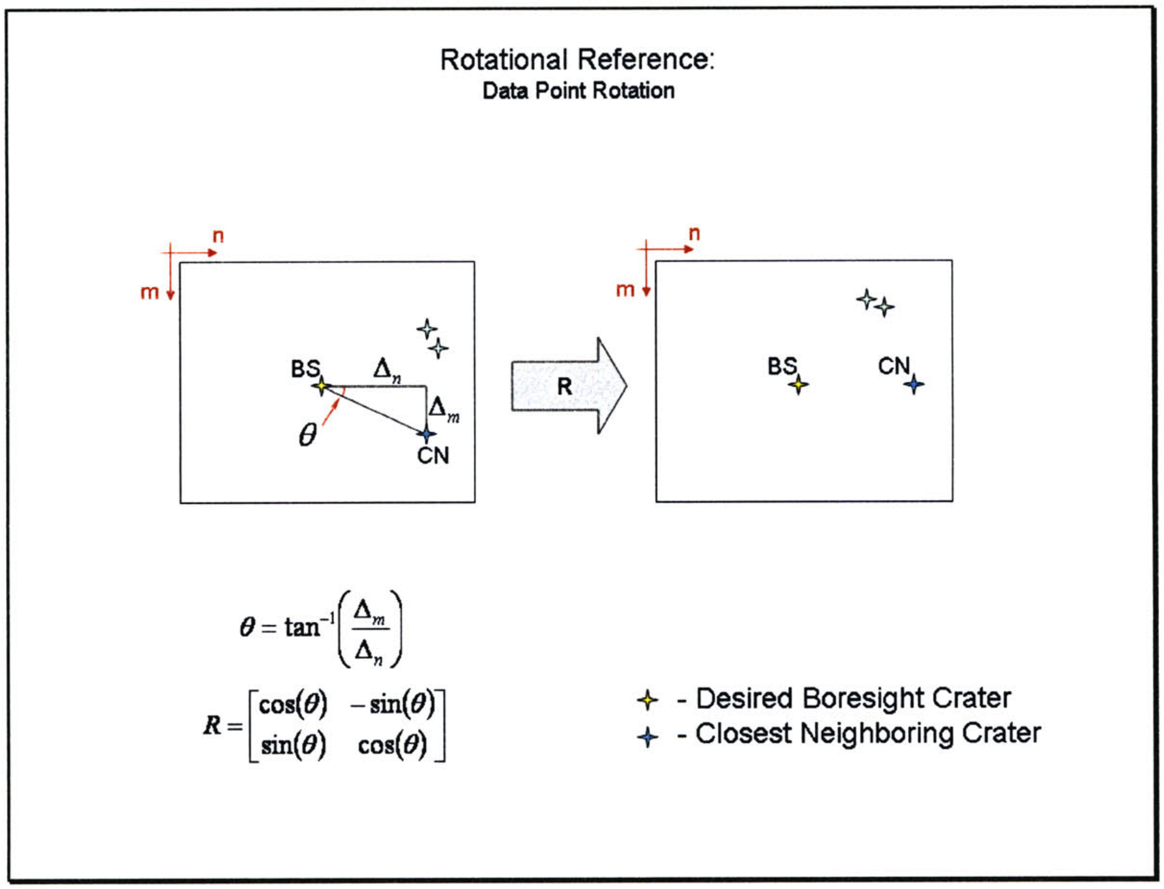
This choice of the closest nearby crater as a point of reference is easy to determine for any given set of data points. However, if the closest neighboring crater is very close to the boresight, the possibility of a significant amount of error being introduced by rotating the data points based upon this point is heightened. As a result, a buffer radius (BR) is introduced. This buffer radius defines the minimum allowable distance between the closest neighboring crater and the boresight crater and serves to prevent excessive rotational error.

If the closest neighboring crater is not selected properly (for whatever reason), the pattern vector generated for that boresight crater may be completely wrong. Therefore, the selection of the proper boresight is critical to the correlation of a crater to its corresponding location from the database. Reference section 8.3 for the author's recommendations on methods to mitigate the probability of misidentifying the closest neighbor crater.

4.2.2 Rotation of Data Points

Once the closest neighboring crater has been identified, the rotation of the data points is accomplished in a fairly standard manner. First, the desired angle of rotation about the boresight in the image plane is determined. Next, a two-dimensional rotation matrix is formed, and finally the rotation is applied to the data points by matrix multiplication.

The rotation angle (θ) is the counter-clockwise angle about the boresight through which the data points must be rotated. A rotation of this nature will position the closest neighboring crater to the right of the boresight on the same horizontal axis as the boresight crater. To determine this rotation angle, a right triangle is defined with the hypotenuse the distance between the boresight crater and closest neighboring crater and one of the legs as the horizontal axis to the right of the boresight. Reference Figure 4.12.



Rotational Reference:
Data Point Rotation
n
m
BS
Δ_n
Δ_m
θ
CN
R
BS
CN
$\theta = \tan^{-1}\left(\frac{\Delta_m}{\Delta_n}\right)$
$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
✦ - Desired Boresight Crater
✦ - Closest Neighboring Crater

Figure 4.12. Establishing the Rotational Reference.

Simple trigonometry allows the determination of this rotation angle.

$$\theta = \tan^{-1}\left(\frac{\Delta_m}{\Delta_n}\right)$$

Equation 4.5

Depending upon the position of the closest neighbor in relation to the boresight, this angle may need to be checked to ensure that the tangent properly accounted for the original quadrant location of the closest neighbor.

The determination of this rotation angle allows the formation of the standard rotation matrix to rotate data points counter-clockwise.[1]

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Equation 4.6

Finally, simple matrix multiplication allows the data points to be rotated appropriately. Allow a representative data point to be $\{a \quad b\}$ and the same rotated data point $\{a' \quad b'\}$. Then

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix}$$

Equation 4.7

If this rotation is applied to every data point, the resulting data will be oriented properly about the boresight with the closest neighbor to the right of and on the same horizontal axis as the boresight crater.

The rotation angle defined here is not the same as the rotation angle (β_i) that will be defined in Chapter 5.

4.2.3 Data Point Rotation Example

To demonstrate both the reference point selection and the rotation of data points, the example of Figure 4.8 will be continued here. This example assumes the desired boresight crater is already properly translated to the principle point of the image, as in Plot B of Figure 4.8.

[1] Which can be found in almost any textbook dealing with the rotation of axes or data points.

4.2.3.1 Select the closest neighbor.

Figure 4.13 demonstrates the determination of the closest neighbor for the given data points.

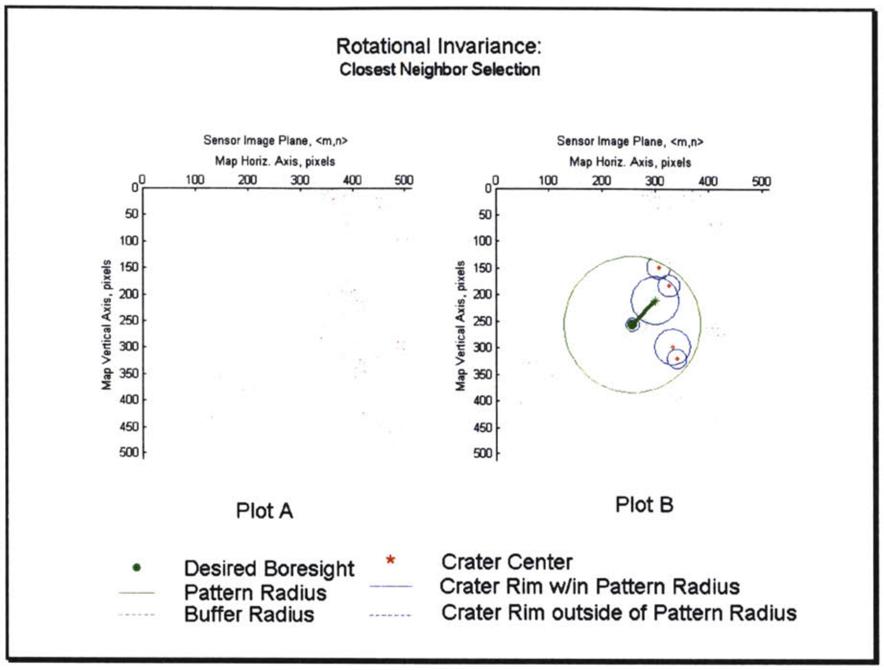


Figure 4.13. Establishing the Rotational Reference: Closest Neighbor Selection for the Example Data Points.

Plot A in Figure 4.13 displays the crater centers and rims visible in the original sensor image (after the perspective transform centered the desired boresight crater). This is the same initial data as that used in the example in section 4.1.3. Plot B highlights the selection of the closest neighboring crater. Once again, the desired boresight crater is a filled green point. The outer limit of the buffer radius (BR) is overlaid on the plot with a dashed green circle. The patter radius (PR) is also shown as a solid green circle. Only the data within the pattern radius will be considered in the formation of the pattern vector. Therefore, the pattern radius must be the same for both the sensor image and the database. Otherwise, the pattern vector in the database and that of the sensor image will include different amounts of data. The closest neighboring crater is indicated by a green asterisk in Figure 4.13.

4.2.3.2 Rotate the Data Points.

Figure 4.14 demonstrates the rotation of the data points to the desired orientation for pattern generation.

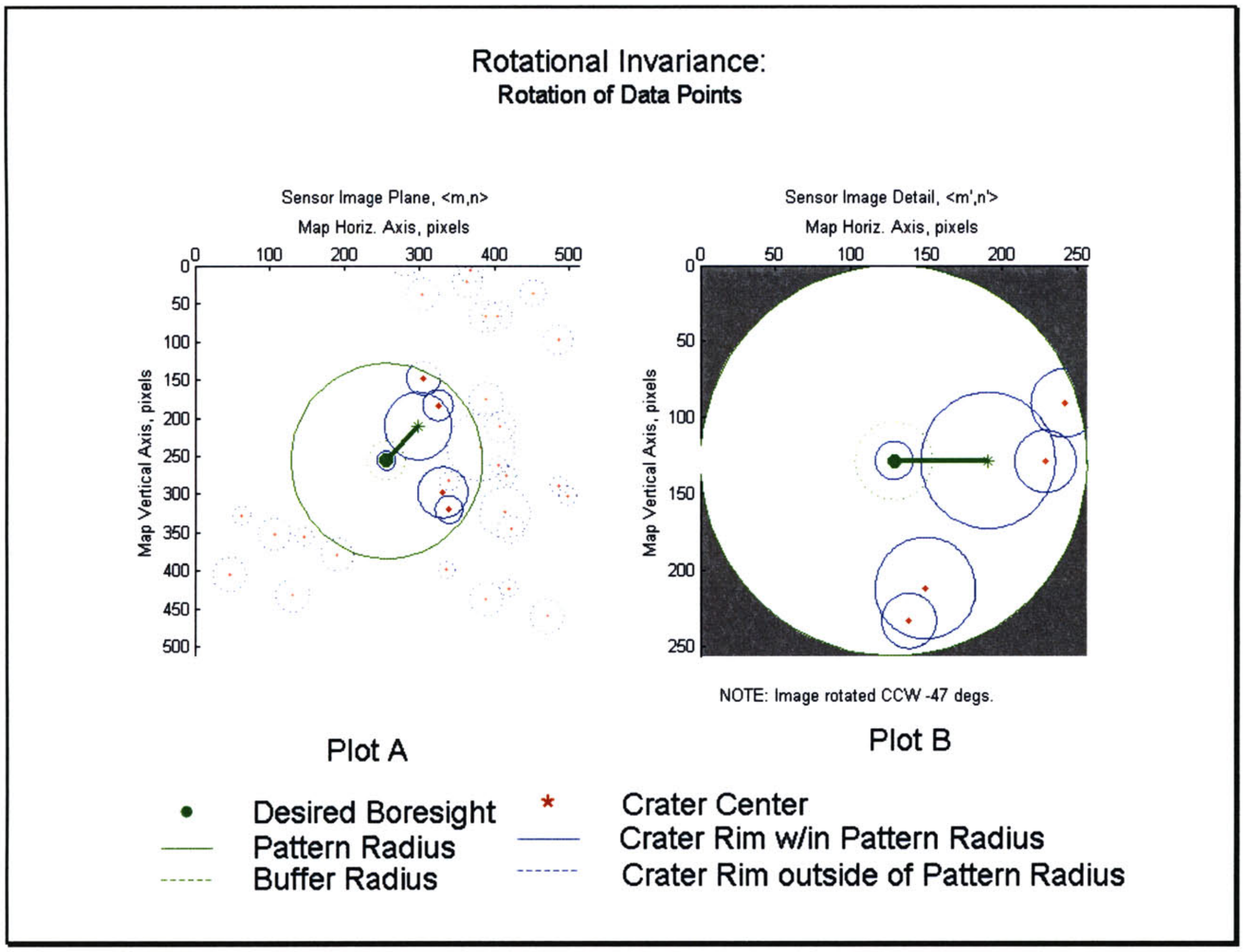


Figure 4.14. Establishing the Rotational Reference: Rotation of the Example Data Points.

Plot A in Figure 4.14 is the transformed data set with the closest neighbor, buffer radius, and pattern radius identified. Like Plot A and B from Figure 4.13, this is also a plot of the data after the perspective transform has been applied to center the data points about the desired boresight. Note that Plot A in Figure 4.14 is identical to Plot B in Figure 4.13. The boresight crater, closest neighboring crater, buffer radius, and pattern radius are all identified according to the standard described in the previous section. Plot B in Figure 4.14 'zooms in' on the original data points so that the pattern generated only includes those craters within the pattern radius. The portions of this plot that are not included in the pattern generation are indicated by shading these areas gray. In addition, the data in Plot B has been rotated clockwise by 47° about the desired boresight so that the closest neighboring crater is rotated to the horizontal level of the boresight and to the right of the boresight.

Plot B in Figure 4.14 is an example of data points that have an established translational and rotational reference. These data points are properly processed for pattern generation.

4.3 Generate the Grid-Based Pattern Vector

Sections 4.1 and 4.2 detail a method to determine a reliable translational and rotational frame of reference. Once this frame of reference is established for a given image, the pattern of neighboring craters for the selected boresight crater must be established. The pattern will then be used to correlate the sensor image with the crater database, and thereby determine the absolute location of the image.

The next several sections will detail the process of determining the crater pattern and generating a pattern vector. The pattern of nearby craters is recognized by overlaying a grid onto the image. Then the crater pattern referenced to the grid is stored in a bit vector that specifies whether or not a crater center occupies a given grid cell.

4.3.1 Derivation of a Pattern from the Image

Before the pattern of craters for a specified boresight crater can be determined, the author will again emphasize that the data points must be translationally and rotationally referenced. Plot B in Figure 4.14 is an image that has an established translational and rotational reference. Reference sections 4.1

and 4.2 for a procedure to establish these frames of reference. By establishing these references, a given image may be compared to the database regardless of the orientation of the original image in relation to the inertial lunar frame.

Given a set of data points with an established translational and rotational reference, the pattern of surrounding craters can be generated for a given boresight crater. This pattern will be determined in several steps:

1. Overlay a $g \ x \ g$ grid on the image, where g is the user-defined number rows and columns of the grid overlay.
2. For each grid cell, determine if a crater center lies within the cell.
3. Record the grid pattern in a $g \ x \ g$ 'grid matrix'. For every grid cell containing a crater center, input the radius of the crater into the matrix. For empty cells, enter a zero into the matrix.

An example of these steps will serve to demonstrate the pattern generation in more detail. The example presented below is a continuation of the example that has been used throughout Chapter 4.

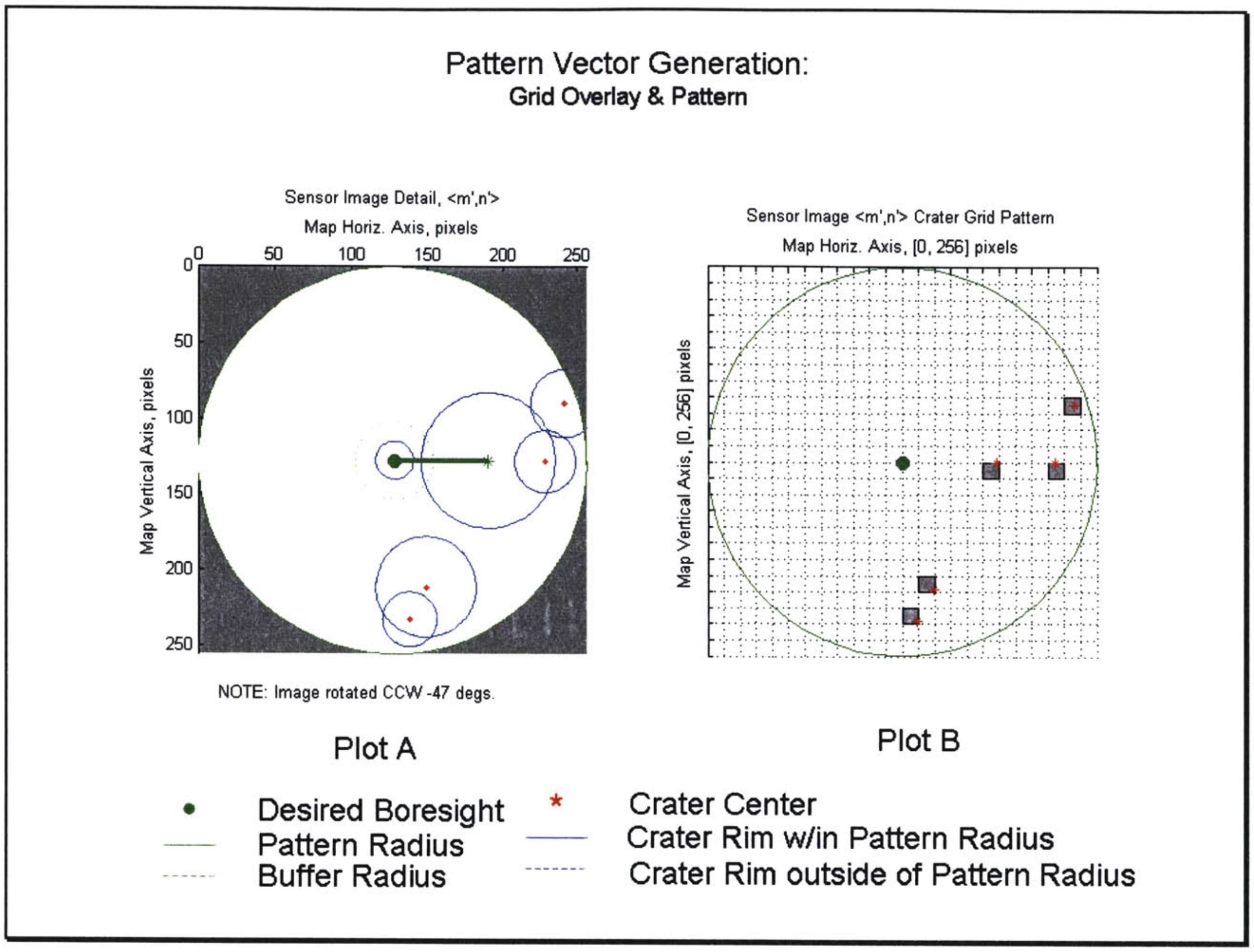


Figure 4.15. Pattern Vector Generation for the Example Data Points.

Plot A in Figure 4.15 is identical to Plot B in Figure 4.14. Plot A in Figure 4.15 has been rotationally and translationally referenced and data outside the pattern radius has been ignored, in preparation for the grid overlay and pattern generation.

Plot B in Figure 4.15 combines the grid overlay and the initial cell status determination (Steps 1 and 2 above). The crater positioning is exactly the same in Plot A and Plot B. However, Plot B has forgone displaying the crater rims. For the first step in creating the pattern of nearby craters, a grid has been established in Plot B. This g x g grid is based on $g = 24$, the initial condition stated in section 3.2.3.6. The second step in determining the crater pattern is to determine if any crater center lies in a given grid cell. These 'full' cells have been indicated in Plot B by filling the grid cell in dark gray. Every full grid cell has a corresponding crater center that lies within the bounds of the cell in Plot B.

From the pattern of 'full' and 'empty' cells in Plot B, the grid matrix can be populated. The grid matrix is the 24×24 (that is, $g \times g$) matrix that contains zeros in the grid cells where no crater centers were found, and the crater radius in the cells where there was a crater center. This grid matrix will contain all the pattern information for the specified desired boresight crater and the initial conditions given.

If more than one crater center falls within a given grid cell, there is ambiguity in which radius is recorded for that grid cell. This situation will not be addressed in the implementation of the transform. However, several possibilities for resolving this ambiguity will be discussed in the recommendations for future work, section 8.3.

4.3.2 Pattern Vector Format

Once the grid matrix is populated, it can be stored and accessed more efficiently in the form of a vector. This section will present a method for converting the grid matrix into the pattern vector.

From Figure 4.15, the grid matrix can be determined.

Example Grid Matrix =

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22\,288 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20\,384 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 33\,376 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18\,144 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Equation 4.8

Perhaps the simplest method for storing the grid matrix as a vector is to concatenate the rows of the grid matrix into a single row vector. In other words,

$$\begin{bmatrix} row\ 1 \\ row\ 2 \\ \vdots \\ row\ n \end{bmatrix} \Rightarrow \begin{bmatrix} row\ 1 & row\ 2 & \cdots & row\ n \end{bmatrix}$$

Equation 4.9

This 'pattern vector' format of the grid matrix is desirable for a number of reasons. Perhaps most importantly, by simply appending other pattern vectors as additional rows in a matrix of pattern vectors, many pattern vectors can be stored in one data structure. This is extremely useful for the formation and storage of a database of pattern vectors. The crater database will be addressed more extensively in section 6.2. In addition, this pattern vector format allows for simple conversion between the grid matrix and the pattern vector without the loss of any data. The author has decided that the benefits of converting the grid matrix into pattern vector merit the data format conversion.

4.3.3 Sparsity Considerations

The pattern vector format presented in section 4.3.2 is an effective means of storing grid matrices when they are fully populated. However, the grid matrices generated by this crater matching routine are typically quite sparsely populated. For sparse matrices, the pattern vector format in Equation 4.9 is highly inefficient because it unnecessarily stores many empty grid cells.

This section will demonstrate the inadequacy of storing the pattern vector according to Equation 4.9, present several common methods for storing sparse matrices, and develop the method chosen by the author to store the sparse grid matrices.

From Equation 4.8, the first row with a nonzero element in this example is the 9th row. Once translated into the format in Equation 4.9, the first 215 elements of the concatenated pattern vector are zeros, followed after one full cell by 90 zeros, and so on. For the entire grid matrix, there are only four nonzero elements – which means that less than 0.7% of the grid cells are full! The sparsity of the data as evidenced in this example is common for images of the lunar surface. Therefore, significant processing and memory savings can be realized by taking advantage of the sparsity of the pattern vector.

The most typical storing technique for general sparse matrices is the compressed row storage format.[1] This storage technique makes no assumptions as to the sparsity of the data structures or symmetry of the matrices being stored. The compressed row storage method does not lose any data, as the method stores every nonzero element in the original matrix. However, if matrices are stored in this format,

[1] The compressed column format is equally as common. However, the process is nearly the same, with no significant disadvantages or advantages. Therefore, the author will present only the compressed row format.

any matrix operations will become highly inefficient. For the crater matching routine, this inefficiency with operations is irrelevant, as no matrix operations will be conducted upon the stored database besides a search routine. However, the author has chosen not to utilize the compressed row format for data storage, and it will be instructive to analyze the method in more detail to understand why the author chose not to use this storage format.

The compressed row format stores three arrays to completely characterize the data in a given matrix. The first array (A) is a row vector of all nonzero elements in the original matrix. The second array (B) is a row vector of the column index for each nonzero element in the original matrix. The third array (C) records the element number of each nonzero data point that begins a new row in the original matrix. By storing the third array in this manner, it eliminates the need to repeat the row location for nonzero elements that are located on the same row in the original matrix.

In compressed row format, the example matrix from Equation 4.8 is

$$\begin{aligned} A &= [22.288 \quad 44.576 \quad 20.384 \quad 33.376 \quad 18.144] \\ B &= [23 \quad 18 \quad 22 \quad 14 \quad 13] \\ C &= [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 2 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 4 \ 0 \ 5 \ 0 \ 0 \ 6] \end{aligned}$$

Equation 4.10

For this example, the compressed row format results in having to store the row index for every row-column ordered pair. The benefit of this method is in storing very large matrices, where it is likely that each row has multiple entries. For entries on the same row, the row number will only be recorded once for the entire row, which could result in significant savings. However, the cost of the compressed row format is the need to store placeholders if a row contains no crater centers.

For a typical grid matrix produced by the crater matching routine, there are usually more rows that have no ‘full’ cells than there are rows that do have ‘full’ cells. This situation makes the need to store zeros as row placeholders prohibitive for memory size. In addition, the compressed row format would create the need to access three separate matrices in order to correlate a sensor image to the database. This need to load three separate matrices could have a significant cost in processing speed. The author has concluded that the potential benefit in the reduction of the size of the database would be negligible in light of the grid matrix sparsity patterns and the potential cost in correlation processing speed for the compressed row format storage technique.

Due to the costs of storing the pattern vector using the compressed row format, the author decided to generate his own method. An explanation of this method and its costs and benefits follows.

Consider the uncompressed pattern vector which has been generated by the horizontal concatenation of the rows of the grid matrix. (Equation 4.9) This vector will be a row vector with g^2 elements, and many of these elements will be zero. To eliminate the storage of zero elements, record the column index and the grid cell value (which will be the crater radius) of all the 'full' cells in the pattern vector. In other words, the ordered pair $\langle pattern\_vector\_column, \quad crater\_radius \rangle$ is recorded for each 'full' pattern vector cell. Store these ordered pairs as consecutive elements in a compressed pattern vector. In this storage scheme, the odd elements in the compressed pattern vector will contain the column indices of the nonzero elements in the decompressed pattern vector. The i^{th} even element of the compressed pattern vector will contain the crater radii for the crater that corresponds to the $(i-1)^{th}$ location in the decompressed pattern vector.

For example,[1] the decompressed pattern vector

$$\begin{bmatrix} 0 & 0 & 5 & 0 & 0 & 0 & 7 & 0 & 0 & -3 & 9 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 4.11

can be stored as the compressed pattern vector

[1] This example does not correspond to the example in use throughout Chapter 4.

$$\begin{bmatrix}3 & 5 & 7 & 7 & 10 & -3 & 11 & 9 & 16 & 2 & 20 & 1\end{bmatrix}$$

Equation 4.12

which is comprised of the index-value ordered pairs

$$\left\langle \begin{pmatrix}3 & 5\end{pmatrix} \begin{pmatrix}7 & 7\end{pmatrix} \begin{pmatrix}10 & -3\end{pmatrix} \begin{pmatrix}11 & 9\end{pmatrix} \begin{pmatrix}16 & 2\end{pmatrix} \begin{pmatrix}20 & 1\end{pmatrix} \right\rangle$$

Equation 4.13

This format will be referred to as the consecutively indexed form. The consecutively indexed form will only save memory if less than half of the elements in the decompressed pattern vector are nonzero. In addition, this form requires only one data structure to store the compressed pattern vector rather than three. Due to the relatively random dispersion of nonzero elements in the decompressed pattern vector and the low number of nonzero elements, the consecutively indexed form will be used to store the pattern vectors.

For the example grid matrix in Equation 4.8, the formation of the condensed pattern vector is presented in Table 4-1 and Equation 4.14.

Table 4-1. Grid Matrix Entries for the Example Data.

Grid Cell	Pattern Vector	Crater Radius	Ordered Pair
(row, column)	Cell	(pixels)	
(9, 23)	215	22.288	(215, 22.288)
(13, 22)	310	20.384	(310, 20.384)
(20, 14)	470	33.376	(470, 33.376)
(22, 13)	517	18.144	(517, 18.144)

$$\left\langle 215 \quad 22.288 \quad 310 \quad 20.384 \quad 470 \quad 33.376 \quad 517 \quad 18.144 \right\rangle$$

Equation 4.14

The grid matrix rows and columns are labeled in Matlab matrix notation, with the first row and column located at the upper left of the grid.

For the remainder of this paper, the term 'pattern vector' will refer to the compressed pattern vector stored in consecutively indexed form, as shown in Equation 4.14.

Chapter 5
The Perspective Transform and the Crater Matching Routine

As described briefly in section 4.1.2, the object of the perspective transform is to modify an image to appear as if it was taken with the principle point focused over a different location in the original image. Specifically, this new principle point will be over the boresight crater center in the original image. By 'centering' the original sensor image this way, a reliable and repeatable context is established to allow the crater matching algorithm to generate a pattern from the other craters in the sensor image.

The transformation will be presented in detail throughout the remainder of this section. The algorithm initializations will first be presented, including the inputs, outputs, assumptions, and other critical algorithm details. Next, the calculation of the rotation matrix between the original and desired camera frames will be detailed. The rotation matrix calculation will be followed by the calculation of the translation vector and the perpendicular depth to each crater center in the original camera frame. Finally, the results of these calculations will allow the perspective transform to be presented and calculated. Each of these steps will be detailed below.

5.1 Problem Statement and Characterization

5.1.1 Problem Statement

Given a nadir-pointing sensor image of the lunar surface from a specified altitude with crater centers and radii identified, modify the pattern of crater centers in the image in such a way as to simulate the image as if it had been taken at another point over the lunar surface. Allow this other point to be defined as the point over an arbitrary crater center in the given image from the same altitude. This arbitrary crater will be referred to as the desired boresight crater. Refer to Figure 5.1 for a pictorial problem overview.

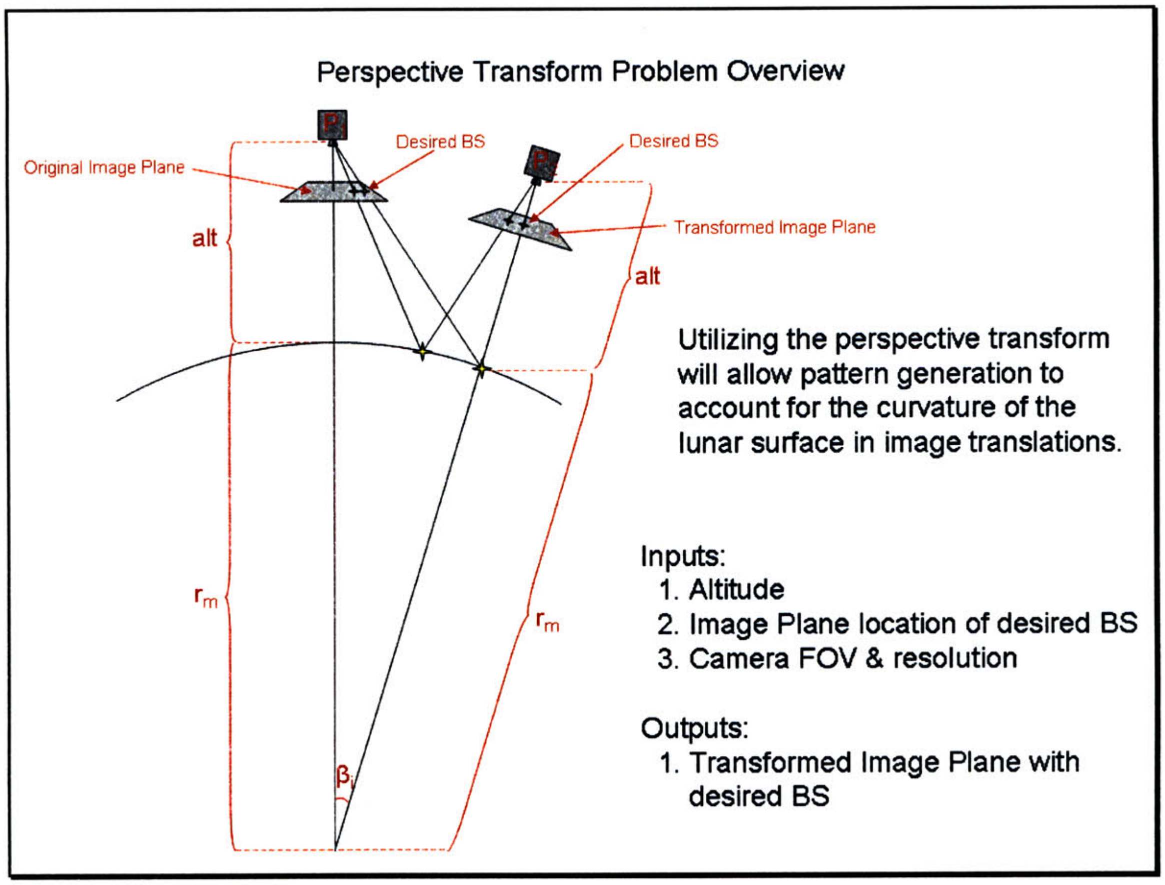

Perspective Transform Problem Overview
Original Image Plane
Desired BS
Desired BS
Transformed Image Plane
alt
alt
r_m
r_m
β_i
Utilizing the perspective transform will allow pattern generation to account for the curvature of the lunar surface in image translations.
Inputs:
1. Altitude
2. Image Plane location of desired BS
3. Camera FOV & resolution
Outputs:
1. Transformed Image Plane with desired BS

Figure 5.1[i]. Reprinted Overview of the Perspective Transform.

In Figure 5.1, the term 'BS' refers to the boresight crater. This abbreviation will be used throughout the remainder of this chapter. The original camera location is designated as $P1$ and the desired camera position as $P2$.

The transformation between these two image frames will be accomplished by implementing a perspective transformation. The perspective transformation process will entail several steps:

1. Calculate the rotation matrix between the image frames.
2. Calculate the translation vector between the image frames.
3. Calculate the perpendicular depth to each crater center.
4. Execute the perspective transform to generate the transformed image plane.

[i] Also Figure 4.6

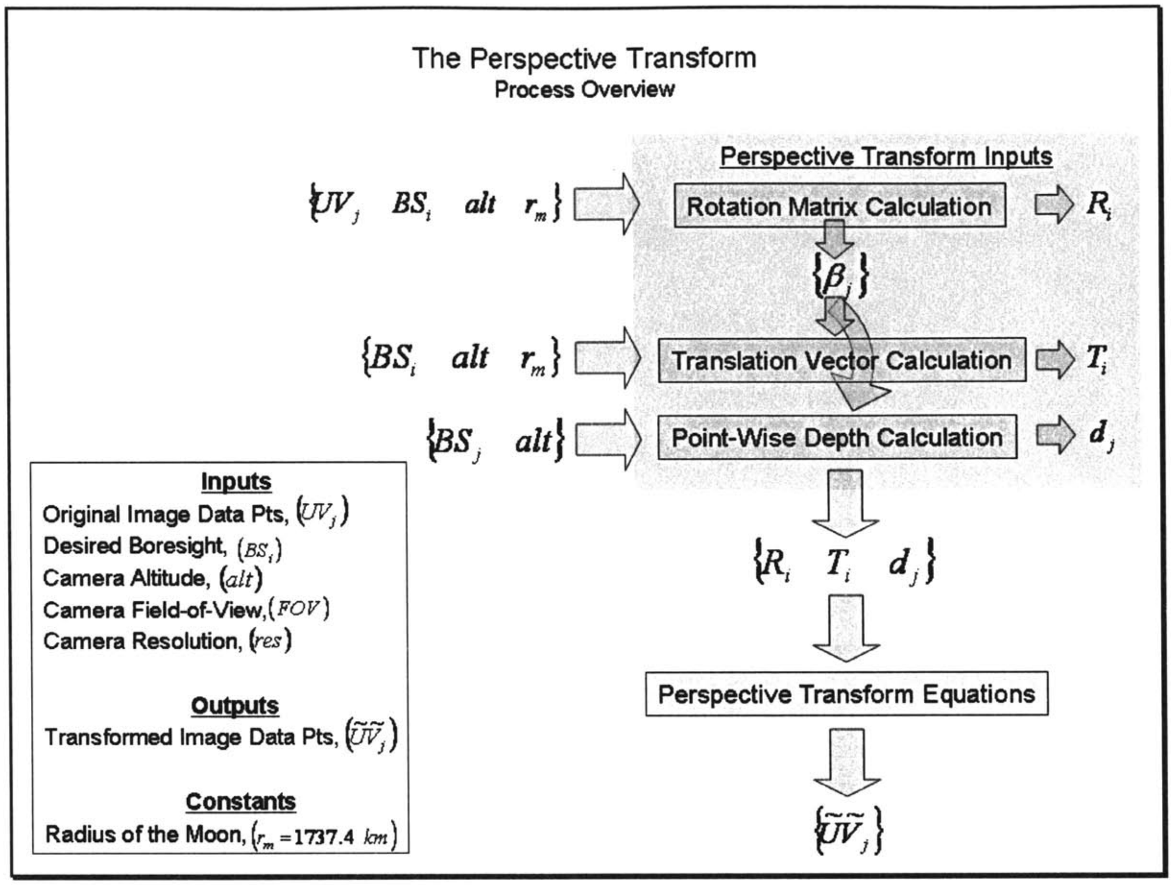

The Perspective Transform
Process Overview
Perspective Transform Inputs
$\{UV_j \quad BS_i \quad alt \quad r_m\}$
Rotation Matrix Calculation
R_i
$\{\beta_j\}$
$\{BS_i \quad alt \quad r_m\}$
Translation Vector Calculation
T_i
$\{BS_j \quad alt\}$
Point-Wise Depth Calculation
d_j
$\{R_i \quad T_i \quad d_j\}$
Perspective Transform Equations
$\{\tilde{U}\tilde{V}_j\}$
Inputs
Original Image Data Pts, (UV_j)
Desired Boresight, (BS_i)
Camera Altitude, (alt)
Camera Field-of-View, (FOV)
Camera Resolution, (res)
Outputs
Transformed Image Data Pts, $(\tilde{U}\tilde{V}_j)$
Constants
Radius of the Moon, $(r_m = 1737.4\ km)$

Figure 5.2. Overview of the Processes of the Perspective Transform.

Figure 5.2 presents an overview of the perspective transform process, demonstrating the interaction between the four main steps in the perspective transform. In addition, Figure 5.2 demonstrates the inputs and outputs of each step. The four steps presented in list form above and in Figure 5.2 will be analyzed in more detail in the remainder of this chapter. As each step is presented, the necessary inputs and outputs which are visible in Figure 5.2 will be defined more precisely.

The remainder of this chapter will demonstrate the coordinate frames and unit conventions in use for this algorithm, detail the transformation algorithm itself, and present several proofs concerning the functionality of the algorithm. Hereafter, the author's use of 'the transform' will be in reference to the perspective transform characterized herein. Refer to section 5.6.3 for more details on the definition of the transform.

5.1.2 Coordinate Frame Definitions

Three separate coordinate frames will be utilized in the derivation of this perspective transform. All are right-handed frames. Refer to Figure 5.3 which shows the relationship between these frames. Two terms will assist in describing these frames. The principle axis is defined as the vector beginning at the camera center and perpendicular to the camera's image plane. The principle point is the point at which the principle axis intersects the image plane.

The lunar frame is the reference frame for this problem and is assumed to be inertial. The origin is located at the center of the Moon. The $+x$ component points toward 0° lunar longitude and the $+z$ component points toward the lunar north pole. The $+y$ component is defined by the right hand rule since the $+x$ and $+z$ directions are given. The lunar frame is necessary for the geometric derivation of the perspective transform algorithm, but is not required in any actual calculations for the transform.

The camera frame is centered upon the camera with the $+z'$ axis along the principle axis. Due to the assumption that the camera is nadir-pointing, the $+z'$ axis is therefore oriented 'downward' toward the lunar surface. The $+x'$ axis is oriented 'forward', in the direction of motion of the spacecraft. The $+y'$ axis is defined by the right hand rule. This frame is non-inertial due to the movement of the spacecraft, which defines the origin of this frame.

The camera's image plane is defined by $\{u \quad v\}$. The origin is located at the center of the focal plane (the principle point) with the $+u$ component pointing toward the left and the $+v$ component toward the top of plane. The third component is not listed for the image plane. Strictly speaking, this component of the image plane is actually defined by being unit distance in the $+z'$ direction from the origin of the camera frame (because the camera's focal distance is also defined to be unit distance, see section 5.1.3). As a result,

$$\{x' \quad y' \quad 1\} = \{u \quad v\}$$

Equation 5.1

For the remainder of this paper, the terms *image frame* and *image plane* will both describe this coordinate frame.

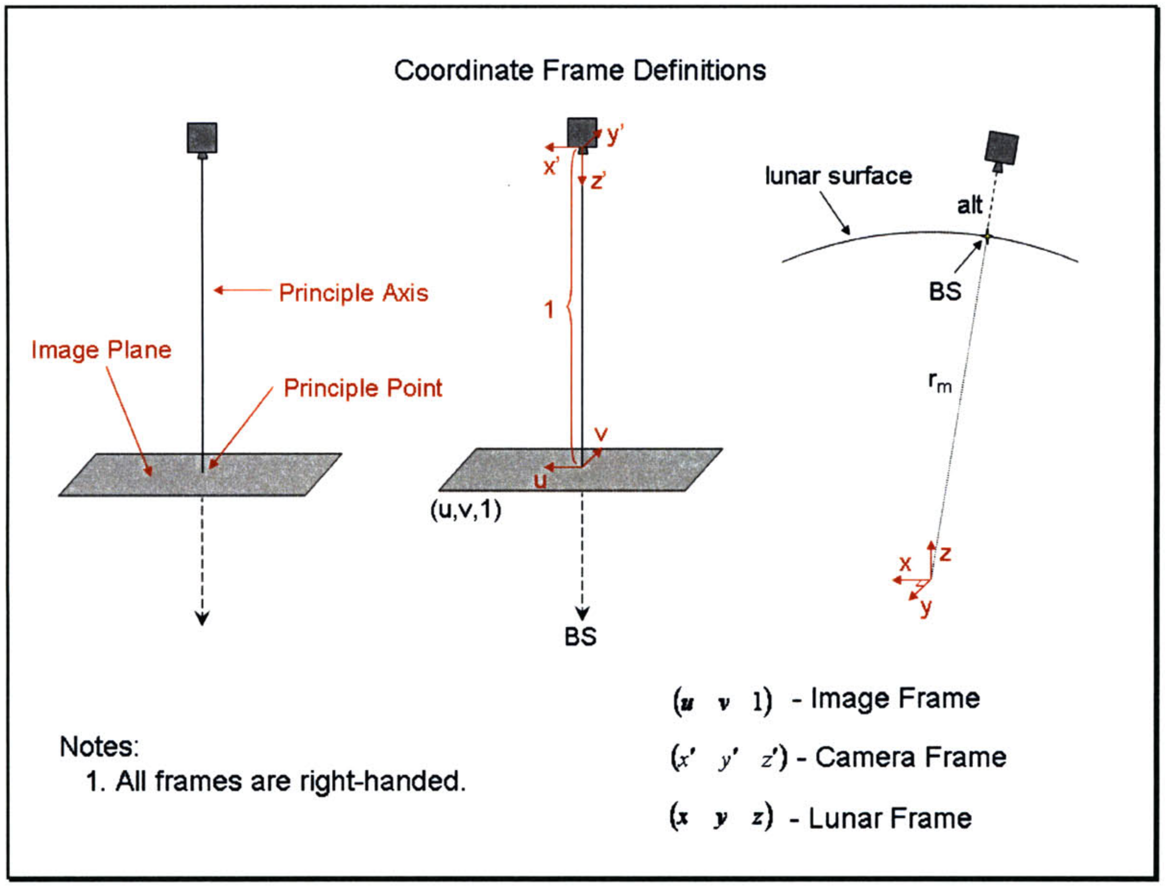

Coordinate Frame Definitions
Principle Axis
Image Plane
Principle Point
y'
x'
z'
1
v
u
(u,v,1)
BS
lunar surface
alt
BS
r_m
x
z
y
$(u \quad v \quad 1)$ - Image Frame
$(x' \quad y' \quad z')$ - Camera Frame
$(x \quad y \quad z)$ - Lunar Frame
Notes:
1. All frames are right-handed.

Figure 5.3. Definition of the Coordinate Frames Used for the Perspective Transform Calculations.

Although all three coordinate frames detailed here are necessary for the derivation and understanding of the perspective transform, only the image frame and camera frame are utilized in the implementation of the equations derived herein.

Strictly speaking, the coordinate frame conversion between the camera frame and the image frame is simply a translation along the z' axis. Although this translation will align the axes and origins, the conversion from the camera frame to the image plane involves a projective frame conversion from

$\Re^3 \rightarrow \Re^2$. This conversion is inherent in the perspective transform process. For simplicity, the ordered pair $\{u \quad v\}$ represents a vector in the image plane, the ordered triplet $\{u \quad v \quad 0\}$ represents a vector in the camera frame that also lies in the image plane, and the ordered triplet $\{u \quad v \quad 1\}$ is a vector in the camera frame with an endpoint in the image plane.

The camera frame and the image plane are dependent upon the location of the camera in relation to the lunar frame. In addition, the perspective transform will simulate the image as if it had been taken at another point over the lunar surface (see Figure 5.1). As a result, it will be useful to refer to the original and transformed camera frame, as well as the original and transformed image plane. The original camera and image frame will describe the frames that characterize the original sensor image. The transformed camera and image frame will be characterized by the simulated camera position after the perspective transform.

5.1.3 Unit Systems

Section 3.2.3 presents a fairly comprehensive overview of the units systems that will also be utilized throughout the perspective transform problem.

Figure 5.4 is repeated here for ease of reference. Two abbreviations utilized in Figure 5.4 will be highlighted again: res refers to the resolution of the original sensor image in pixels, and fov is the camera's field of view in radians.

As mentioned previously, the true image frame in this problem is also the camera focal plane, located a focal length from the camera along the $+z'$ axis with the origin at the center of the image and the $+u$ and $+v$ directions aligned left and upward, respectively.[1]

[1] This definition of the $+u$ and $+v$ directions is defined such that they are aligned with the $+x'$ and $+y'$ axes of the camera frame, respectively. The author chose to define the image plane axes in this manner to allow the

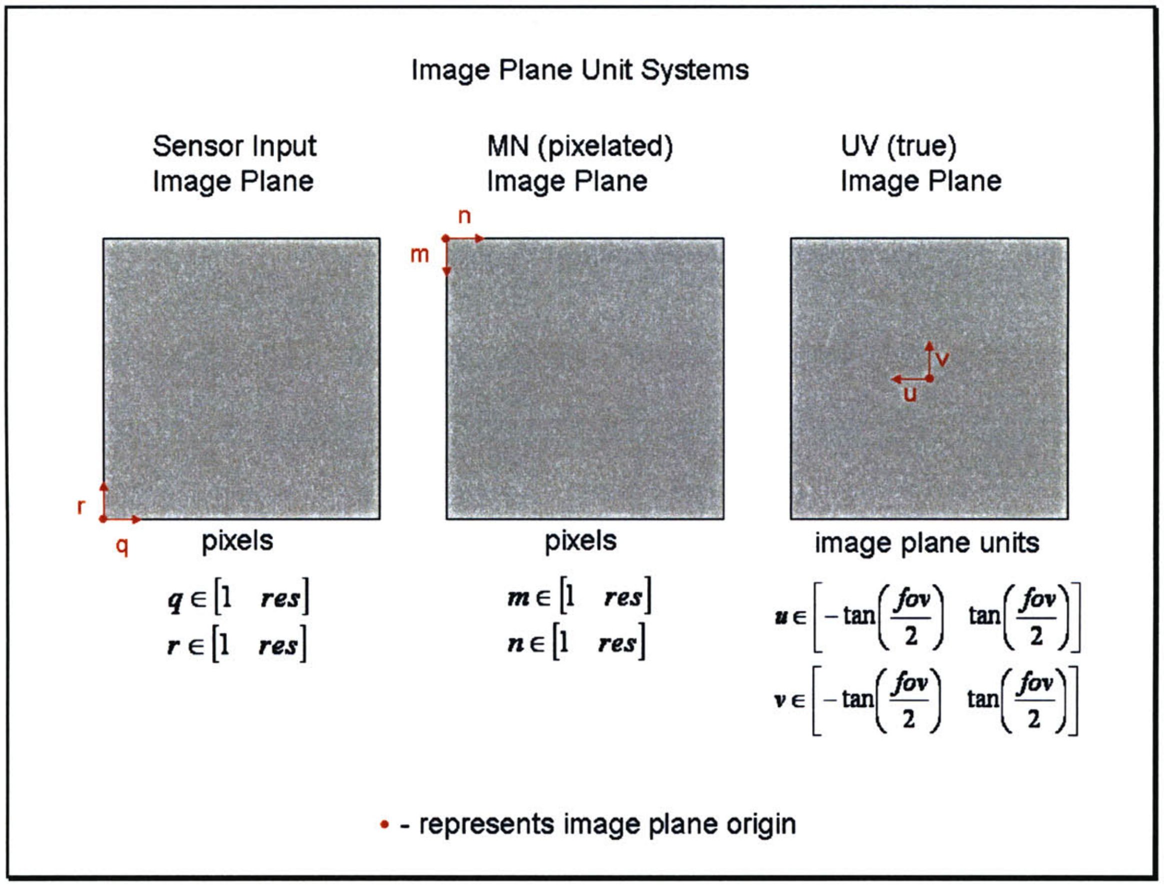


Image Plane Unit Systems
Sensor Input Image Plane
MN (pixelated) Image Plane
UV (true) Image Plane
n
m
v
u
r
q
pixels
pixels
image plane units
$q \in [1 \quad res]$
$r \in [1 \quad res]$
$m \in [1 \quad res]$
$n \in [1 \quad res]$
$u \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right)\right]$
$v \in \left[-\tan\left(\frac{fov}{2}\right) \quad \tan\left(\frac{fov}{2}\right)\right]$
• - represents image plane origin

Figure 5.4[i]. Reprinted Unit Systems of the Image Plane.

5.2 Initialization of the Perspective Transform Algorithm

This section will introduce the terminology, inputs, outputs, constants, and assumptions that will be crucial to the discussion of the perspective transform.

R matrix from Equation 5.14 to be identity, $R = I_{3x3}$. Therefore, this definition of the image plane axes allows further simplification of the final perspective transform equations.

[i] Also Figure 3.3

5.2.1 Terminology

It is necessary to introduce several terms that will be applied throughout the discussion of this algorithm.

1. Principle Axis – the line from the camera center and perpendicular to the image plane.
2. Principle Point – the point at which the principle axis intersects the image plane.
3. Perspective Transform – the image transformation described briefly in sections 4.1.2 and 5.1.1.
4. Desired Boresight Crater – the desired principle point of the transformed image which coincides with a crater in the original image plane.
5. Transformed Image Plane – the image plane in which the perspective transform simulates the transformed image was taken. See Figure 5.1.
6. Image Data Points – the positions of all crater centers and their respective radii visible in a given image. Craters are assumed to be circular.

5.2.2 Inputs, Outputs, & Algorithm Constants

Inputs:

1. Sensor image data points in the original image frame.
2. Image frame coordinates of the crater center which is desired to be the principle point in the transformed image frame.
3. Spacecraft altitude.
4. Camera field-of-view and resolution.

Constants:

1. Radius of the Moon, $r_m = 1737.4\ km$

Outputs:

1. Transformed image frame data points with the desired crater center located at the principle point.

5.2.3 Assumptions

Several assumptions have been made in the implementation of this algorithm. These assumptions, the necessity of each assumption, and their relative validity are discussed here.

5.2.3.1 Moon is spherical

For the perspective transform, a spherical model of the Moon is assumed. This greatly simplifies calculations but still allows the center of gravity and center of mass to remain in their actual positions. This assumption is not expected to have a significant impact upon the transform results since the oblateness of the Moon is very small. In actuality, $J2_{moon} \cong 2.027 \cdot 10^{-4}$ for the Moon compared to $J2_{earth} \cong 1.083 \cdot 10^{-3}$[1] for the Earth.[21] The spherical moon model also assumes that the topography of the lunar surface does not significantly affect calculations. In general, lunar surface elevations vary by as much as 12 km and are an average of about 1.9 km higher on the far side of the Moon than on the near side. Refer to Figure 5.5.[22]

As a result, topography may have a significant effect on the accuracy of the perspective transform. This may necessitate the effects of elevation being accounted for in practical application of this algorithm.

[1] The $J2$ coefficients are used in determining the effect of the oblateness of a spheroid on its gravity potential. Therefore, the magnitude of these coefficients is generally utilized as a measure of the oblateness of the body.

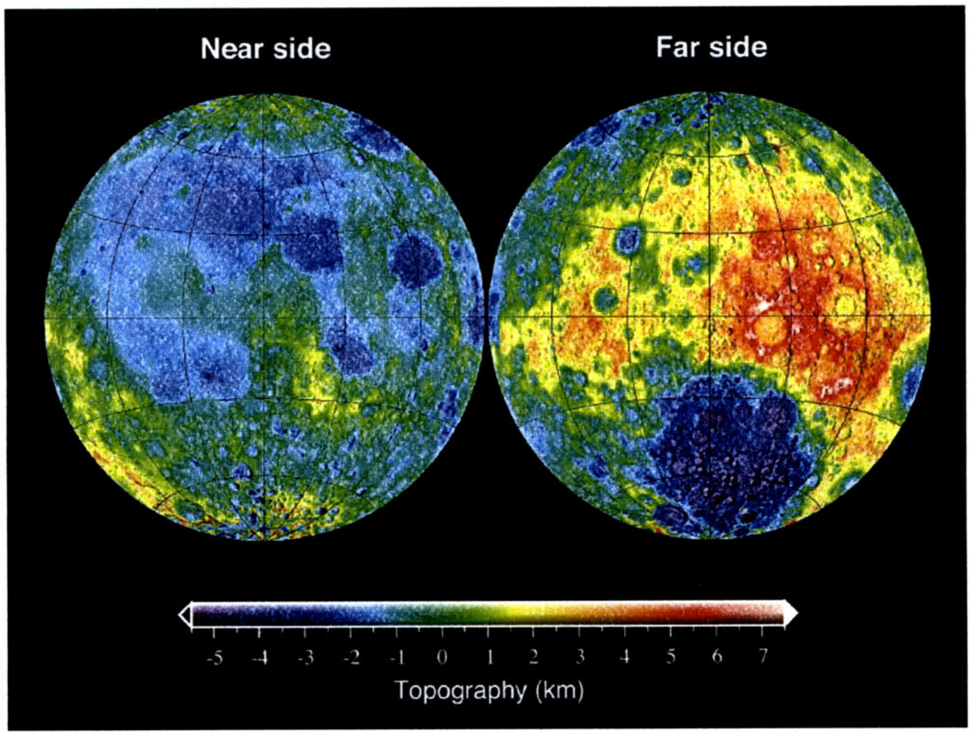


Figure 5.5[i]. Topography of the Lunar Surface.

5.2.3.2 Camera is nadir-pointing.

This algorithm assumes sensor images that were obtained from a nadir-pointing camera. This assumption is a simplifying assumption. With the addition of another perspective transform – or perhaps of its inclusion within the current transform – this algorithm could allow for non-nadir-pointing cameras as long as accurate attitude information is known.

5.2.3.3 Spacecraft altitude does not change over transform.

The transform is intended to transform an image taken at point $P1$ to appear as if it were taken at point $P2$ at the same altitude. See Figure 5.1. However, with simple modification of the translation vector calculation in section 5.4, the point $P2$ could theoretically be allowed to be any point above the lunar surface. For the purposes of this perspective transform, constant altitude will be assumed.

[i] This image is a topographic model based off of the spherical harmonic model developed by the US Geographical Survey, USGS359. The lunar geoid was obtained from the lunar gravity model LP150Q.

5.2.3.4 Image plane is unit distance from camera.

The algorithm also assumes that the image plane is located at unit distance from the camera along the $+z'$ axis. In other words, the focal length is unit length in the units of distance utilized in the problem. To change this assumption, the perspective transform equation, Equation 5.27, must be derived again from the projection equations in section 5.6.2.1. In general, however, the image plane being unit distance from the camera does not affect the validity or applicability of the perspective transform process; rather, it merely scales the results.

5.2.3.5 The shape of any given crater is constant over the transform.

It is assumed that perspective only has a significant effect on crater centers, not upon the crater's shape. This assumption is not valid. However, the crater matching algorithm depends upon precise crater center knowledge, and only loosely upon the crater radius. Therefore, this assumption can be made to simplify the algorithm. To take the effects of the perspective transform upon the crater's shape into effect, many points along any given crater rim would need to be identified and the perspective transform equations applied to these points. The result would be a slight change in the relative position of these points over the transform.

5.2.3.6 Craters are circular.

Craters are assumed to be circular in order that the radius can be used for matching purposes. This assumption complements the assumption of section 5.2.3.5 in allowing the crater radius to be stored rather than having to identify many points along the crater rim in order to identify the shape of the crater.

To implement some more sophisticated model of crater shape, the parameters of that model must be stored for use in crater matching purposes. In addition, a database must be generated based upon the same crater model, and the model parameters pertinent to each crater must be stored for that crater. For example, craters could be modeled as ellipses. A focus and the semi-major and minor axes would be stored in this case. Either the explicit center of the ellipse or the two foci would be used to define the position of the crater.

Note that this assumption was also made for the crater matching routine, noted in section 3.2.3.5.2.

5.2.3.7 Camera is an ideal camera

The perspective transform assumes an ideal camera, with no sensor noise. This assumption was also made for the crater matching routine, noted in section 3.2.3.5.1.

5.2.4 Algorithm Notes

Several indices in use in this algorithm allow for cyclic processing of craters within a given image. These indices require explanation:

1. i iterates on the potential boresight selected as the desired boresight selected from craters in the sensor image. Therefore $\{u_i \quad v_i\}$ represents the desired boresight in the original sensor image.
2. j iterates through all craters in the sensor image for a given desired boresight i. Therefore $\{u_j \quad v_j\}$ represents an arbitrary crater center in the sensor image.

5.3 The Perspective Transform Rotation Matrix Calculation

As is explained in section 5.1.1, the perspective transformation will modify an image so that the transformed image has the desired boresight crater at the principle point. This process will require the rotation matrix between the original image plane and the desired image plane. This rotation is completely characterized by a rotation axis and angle. Therefore, the calculation of the rotation matrix will require the following steps:

1. Determine the desired rotation angle.
2. Determine the axis of rotation.
3. Use the rotation axis and angle to determine the rotation matrix.

This process is based primarily upon the geometry of the problem. The convention of the right hand rule is used throughout these calculations; therefore, a counter-clockwise angle is, by convention, positive. This section will present the rotation matrix calculation in the three steps listed above.

5.3.1 Rotation Matrix Calculation Initialization

Inputs:

1. Sensor image data points in the original image frame.
2. Image frame coordinates of the i^{th} crater center, which is the desired boresight.
3. Spacecraft altitude.

Outputs:

1. Crater angle β_j for every sensor image crater j. The j^{th} crater angle is the angle between the principle axis and the radial line from the center of the Moon to the j^{th} crater center in the original sensor image. The i^{th} crater angle, β_i, is also the rotation angle for the i^{th} potential boresight. Refer to section 5.3.2 for a more detailed explanation.
2. Rotation matrix R_i for every selected boresight i. The rotation matrix is a function of the rotation angle, β_i.

Constants:

1. Radius of the Moon, $r_m = 1737.4\ km$

Assumptions:

1. Moon is spherical.
2. Camera is nadir-pointing.
3. Spacecraft altitude does not change over transform.
4. Image plane is unit distance from the camera along $+z'$ axis.

5.3.2 Rotation Angle Derivation

The rotation angle (β_i) is the angular separation between the original and the transformed boresight vectors (or principle axes). Since the camera is assumed to be nadir-pointing, these boresight vectors are also radial vectors pointing from the center of the Moon to points on the lunar surface. Therefore, the angular separation of the radial vector to the current boresight and the radial vector to the crater center of the desired boresight *on the lunar surface* can also be understood to be the rotation angle. Refer to Figure 5.7.

In the geometric determination of the rotation angle, the apparent camera angle (α_i) to the i^{th} crater is calculated first using the Pythagorean Theorem and simple trigonometry. Refer to Figure 5.6.

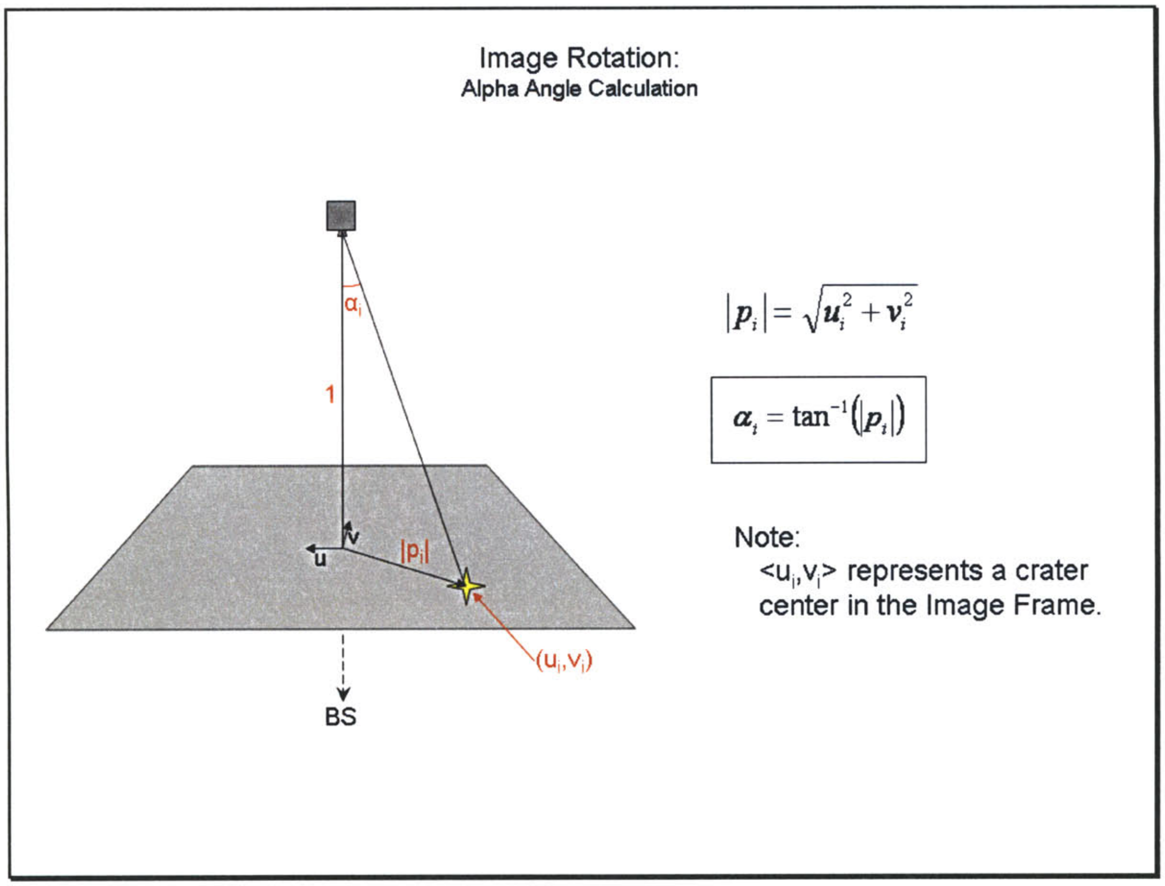


Image Rotation:
Alpha Angle Calculation
α_i
1
v
u
$|p_i|$
(u_i,v_i)
BS
$|\boldsymbol{p}_i| = \sqrt{\boldsymbol{u}_i^2 + \boldsymbol{v}_i^2}$
$\boxed{\boldsymbol{\alpha}_i = \tan^{-1}(|\boldsymbol{p}_i|)}$
Note:
$<u_i,v_i>$ represents a crater center in the Image Frame.

Figure 5.6. Image Rotation: Alpha Angle Calculation.

By geometry,

$$\alpha_i = \tan^{-1}\left(\frac{\sqrt{u_i^2 + v_i^2}}{1}\right)$$

Equation 5.2

Relying upon the geometry of the problem and utilizing the Law of Sines, the angle γ_i can be determined. Refer to Figure 5.7.

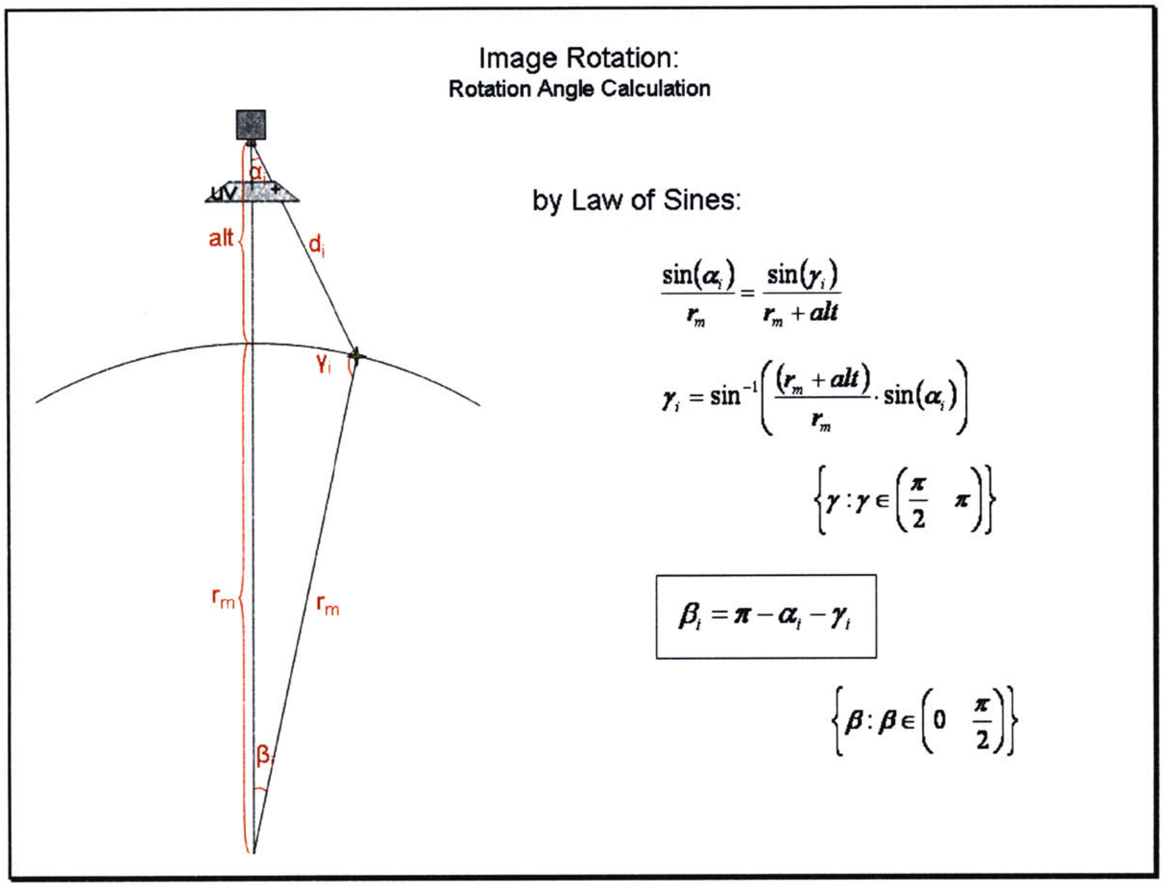


Image Rotation:

Rotation Angle Calculation

by Law of Sines:

$$\frac{\sin(\alpha_i)}{r_m} = \frac{\sin(\gamma_i)}{r_m + alt}$$

$$\gamma_i = \sin^{-1}\left(\frac{(r_m + alt)}{r_m} \cdot \sin(\alpha_i)\right)$$

$$\left\{\gamma : \gamma \in \left(\frac{\pi}{2} \quad \pi\right)\right\}$$

$$\boxed{\beta_i = \pi - \alpha_i - \gamma_i}$$

$$\left\{\beta : \beta \in \left(0 \quad \frac{\pi}{2}\right)\right\}$$

Figure 5.7. Image Rotation: Rotation Angle Calculation.

$$\gamma_i = \sin^{-1}\left(\frac{r_m + alt}{r_m} \cdot \sin(\alpha_i)\right), \quad \left\{\gamma : \gamma \in \left(\frac{\pi}{2} \quad \pi\right)\right\}$$

Equation 5.3

Finally, the rotation angle (β_i) follows from the sum of the interior angles of a triangle, considering that the values for α_i and γ_i already determined. Refer to Figure 5.7.

$$\beta_i = \pi - \alpha_i - \gamma_i$$

Equation 5.4

The rotation angle (and therefore α and γ as well) need only be calculated once for each desired boresight, as explained section 5.3.4. However, the calculation of the perpendicular distance to each crater center in the image requires the crater angle to each crater center, as stated in section 5.5. This is the reason that the crater angle calculation will be completed for every crater before the boresight is selected. For the i^{th} desired boresight, the i^{th} crater angle will also be the rotation angle. That is, if $j = i$ then $\beta_j = \beta_i$.

5.3.3 Rotation Axis Derivation

The rotation axis is the axis about which the rotation of data points will occur according to the right hand rule. In conjunction with the rotation angle, this three-component vector will uniquely define the desired rotation. The rotation axis will not only be used in the calculation of the rotation matrix, but also in the calculation of the translation vector. Both of these calculations are necessary once per desired boresight.

The rotation axis is defined as the vector perpendicular to the plane of rotation. This plane of rotation is the plane of the triangle in Figure 5.7. The direction of the rotation axis is defined by the right hand rule as follows: the rotation of the desired boresight vector to the original boresight vector is in the 'positive' right hand rule direction (that is, counter-clockwise).

Therefore, define

$$z = [0 \quad 0 \quad 1]$$

Equation 5.5

$$UV_i = [u_i \quad v_i \quad 0]$$

Equation 5.6

The cross product between z and UV_i will generate a vector that is perpendicular to the plane of rotation and in the proper direction. Therefore, this cross product yields the rotation axis. Refer to Figure 5.8.

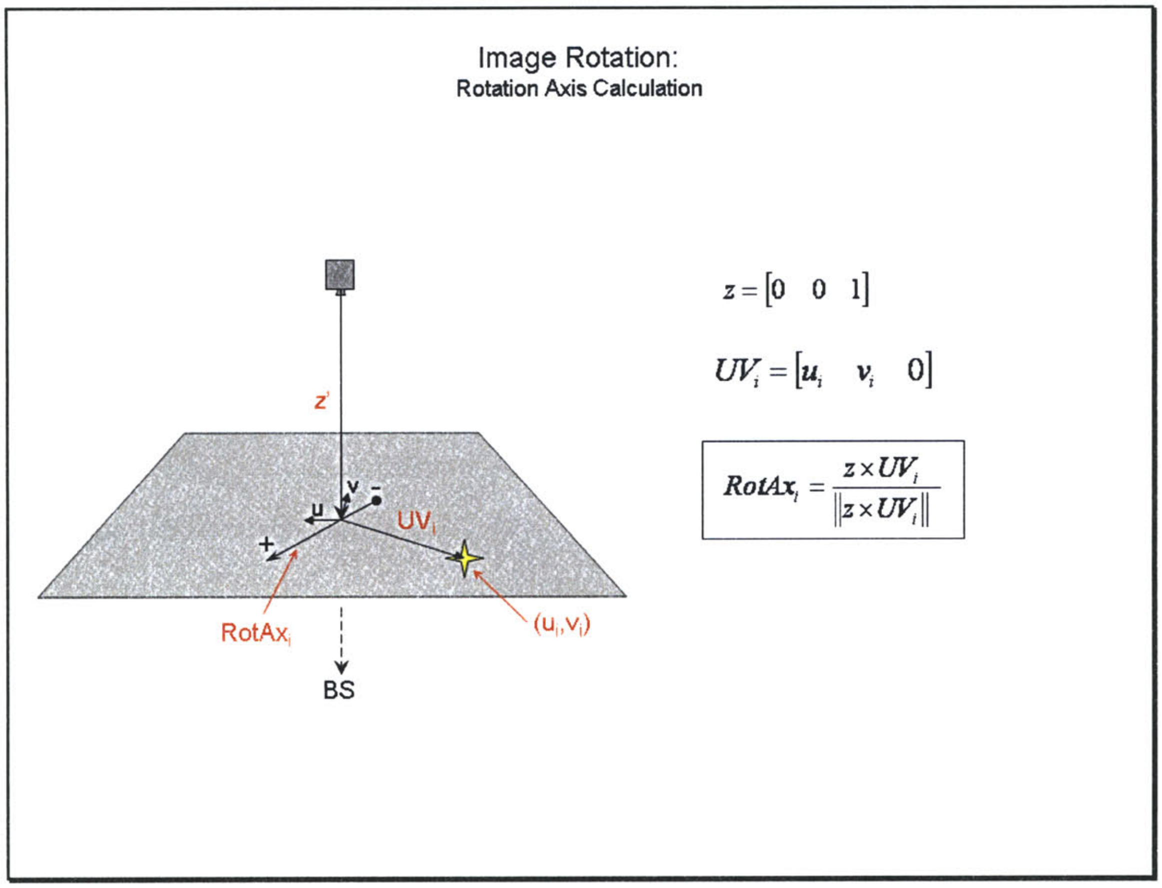


Image Rotation:
Rotation Axis Calculation
z'
v
u
-
+
UV_i
RotAx_i
(u_i,v_i)
BS
$z = [0 \quad 0 \quad 1]$
$UV_i = [u_i \quad v_i \quad 0]$
$RotAx_i = \frac{z \times UV_i}{\|z \times UV_i\|}$

Figure 5.8. Image Rotation: Rotation Axis Calculation.

$$RotAx_i = \frac{z \times UV_i}{\|z \times UV_i\|}$$

Equation 5.7

The rotation axis is normalized for ease of future use.

5.3.4 Rotation Matrix Calculation

The rotation matrix is calculated next from the rotation angle and axis. The rotation matrix is calculated once per desired boresight crater.

There are a variety of methods for calculating a rotation matrix from its associated axis and angle. The following method was chosen by the author for its simplicity and ease of implementation in any matrix-friendly computer language.[1]

Procedurally,

$$N_t = \begin{bmatrix} 0 & -RotAx_t(3) & RotAx_t(2) \\ RotAx_t(3) & 0 & -RotAx_t(1) \\ -RotAx_i(2) & RotAx_t(1) & 0 \end{bmatrix}$$

Equation 5.8

$$N2_t = RotAx_t' * RotAx_i - I_{3x3}$$

Equation 5.9

$$R_t = I_{3x3} + N_t \cdot \sin(\beta_t) + N2_t \cdot (1 - \cos(\beta_t))$$

Equation 5.10

In application, it is quite easy to confuse the sign of the rotation angle and the direction of the rotation axis, which can cause the generation of a rotation matrix that does not accomplish the desired rotation. To ensure the veracity of these calculations, Figure 5.9 breaks the angle-axis combination down into four cases, depending upon the image plane quadrant in which the desired boresight lies.

The z vector (which is into the page) and UV_t are given, the rotation axis can be calculated, as shown. Based upon the diagram, it is then ascertained that a positive rotation angle will indeed rotate the boresight vector from UV to the origin. This holds for all cases.

[1] Equations to determine a rotation matrix from a given axis and angle were taken from the notes of Gerhard Besold http://www.memphys.sdu.dk/~bcsold/INDEX/axis-angle.pdf. The algorithm was verified against other, more traditional algorithms.

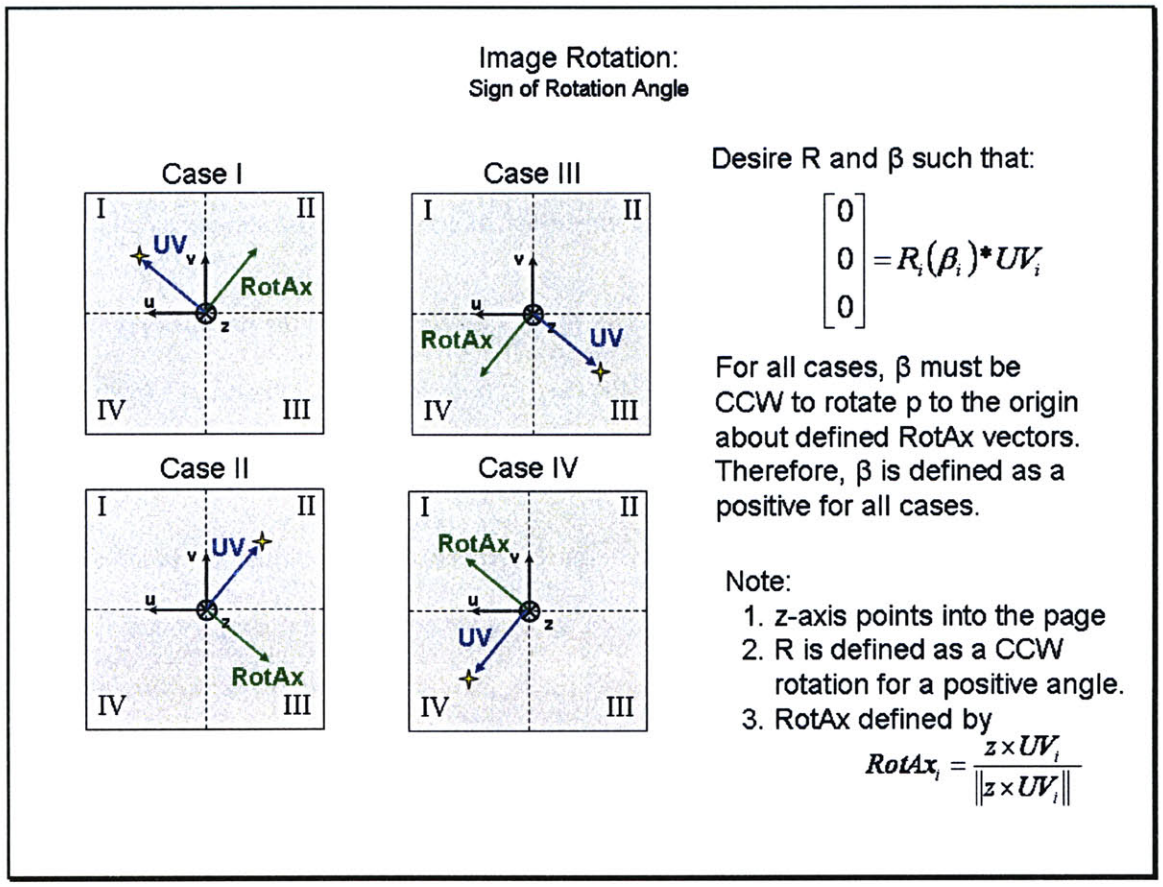


Image Rotation:
Sign of Rotation Angle

Desire R and β such that:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = R_i(\beta_i) * UV_i$$

For all cases, β must be CCW to rotate p to the origin about defined RotAx vectors. Therefore, β is defined as a positive for all cases.

Note:

1. z-axis points into the page
2. R is defined as a CCW rotation for a positive angle.
3. RotAx defined by

$$RotAx_i = \frac{z \times UV_i}{\left\| z \times UV_i \right\|}$$

Figure 5.9. Image Rotation: The Sign of the Rotation Axis.

5.4 The Perspective Transform Translation Vector Calculation

The previous section, section 5.3, detailed the calculation of the rotation matrix between the original camera frame and the transformed camera frame, which is necessary for the execution of the perspective transform algorithm. The translation vector between these two frames is also necessary to execute the perspective transform. The calculation of the translation vector between these frames will require the following steps:

1. Determine the translation vector magnitude.
2. Determine the translation vector direction.
3. Combine the translation direction and magnitude to generate the translation vector.

These calculations are derived from the geometry of the transform. Since the translation vector is uniquely defined by the current boresight and the desired boresight, its calculation is executed once for every boresight crater.

The translation vector must be calculated as the translation between the original and transformed camera frames, and not as the translation between the original and desired principle points.

5.4.1 Translation Vector Calculation Initialization

Inputs:

1. Image frame coordinates of the i^{th} crater center, which is the desired boresight.
2. Spacecraft altitude.
3. Rotation angle β_i for desired boresight crater i.

Outputs:

1. Translation vector T_i between the original and i^{th} desired camera frames.

Constants:

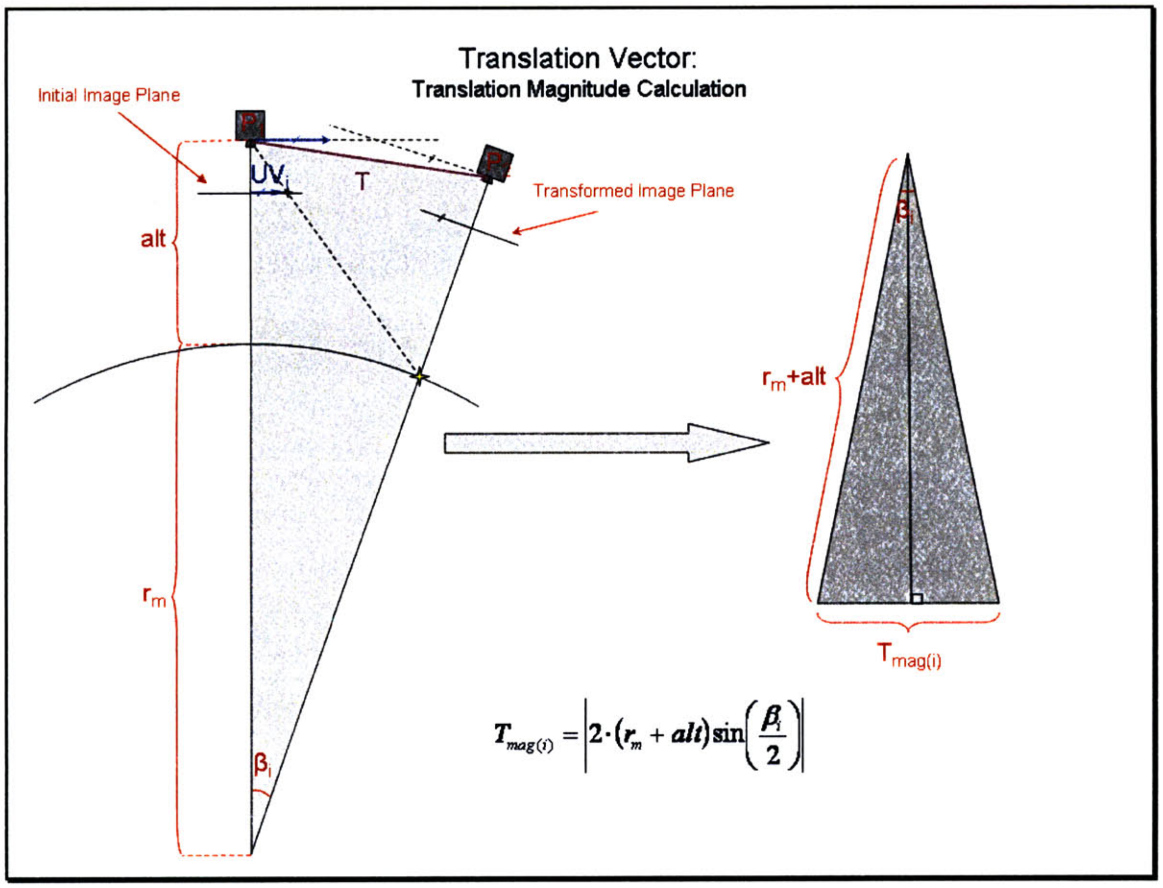
1. Radius of the Moon, $r_m = 1737.4$ km.

Assumptions:

1. Moon is spherical.
2. Camera is nadir-pointing.
3. Spacecraft altitude does not change over transform.

5.4.2 Translation Magnitude Derivation

The translation magnitude is calculated geometrically by utilizing an isosceles triangle. The long legs to the triangle highlighted in Figure 5.10 are equal by the assumption of constant altitude over the transform, which makes the triangle in question an isosceles triangle. As a result, the bisector of the angle between the two legs of equal length will also be perpendicular to the 'base' leg of the same triangle. Therefore, simple trigonometry allows the length of T to be determined using the definition of sine.



Translation Vector:
Translation Magnitude Calculation
Initial Image Plane
UV_i
T
Transformed Image Plane
alt
r_m
β_i
r_m+alt
$T_{mag(i)}$
$T_{mag(i)} = \left|2 \cdot (r_m + alt)\sin\left(\frac{\beta_i}{2}\right)\right|$

Figure 5.10. Translation Magnitude Calculation.

$$T_{mag(t)} = \left| 2 \cdot (r_m + alt) \cdot \sin\left(\frac{\beta_t}{2}\right) \right|$$

Equation 5.11

5.4.3 Translation Direction Derivation

The geometric derivation of the translation direction $T_{dir(t)}$ is somewhat more rigorous than that of the translation magnitude. For simplicity, the derivation will be presented procedurally:

1. From the triangle highlighted in Figure 5.10, the angle between T_t and the original boresight vector can be determined to be $\frac{\pi - \beta_t}{2}$ radians by the sum of interior angles to a triangle and the nature of an isosceles triangle. This angle is also the same angle as that between T_t and the desired boresight vector.

2. In addition, the boresight vector is normal to the image plane. Therefore, vector $UV_t = \begin{bmatrix} u_t \\ v_t \\ 0 \end{bmatrix}$ is perpendicular to original boresight vector. Consequently, the angle between UV_t and T_t can be determined to be $\frac{\beta_t}{2}$ as shown in the detail of Figure 5.11.

3. If UV_t is rotated in a clockwise direction by $\frac{\beta_t}{2}$ radians in the plane of the highlighted triangle in Figure 5.11, the resulting vector will be in the direction of translation. The rotation axis will be defined as it was in Equation 5.7, (the rotation axis will point out of the page in this case). Since the necessary rotation is clockwise, the angle of rotation will be negative. Therefore, the rotation matrix can be determined by Equation 5.8, Equation 5.9, and Equation 5.10, with the caveat that the rotation matrix will be calculated for a rotation angle of $\frac{-\beta_t}{2}$ radians.

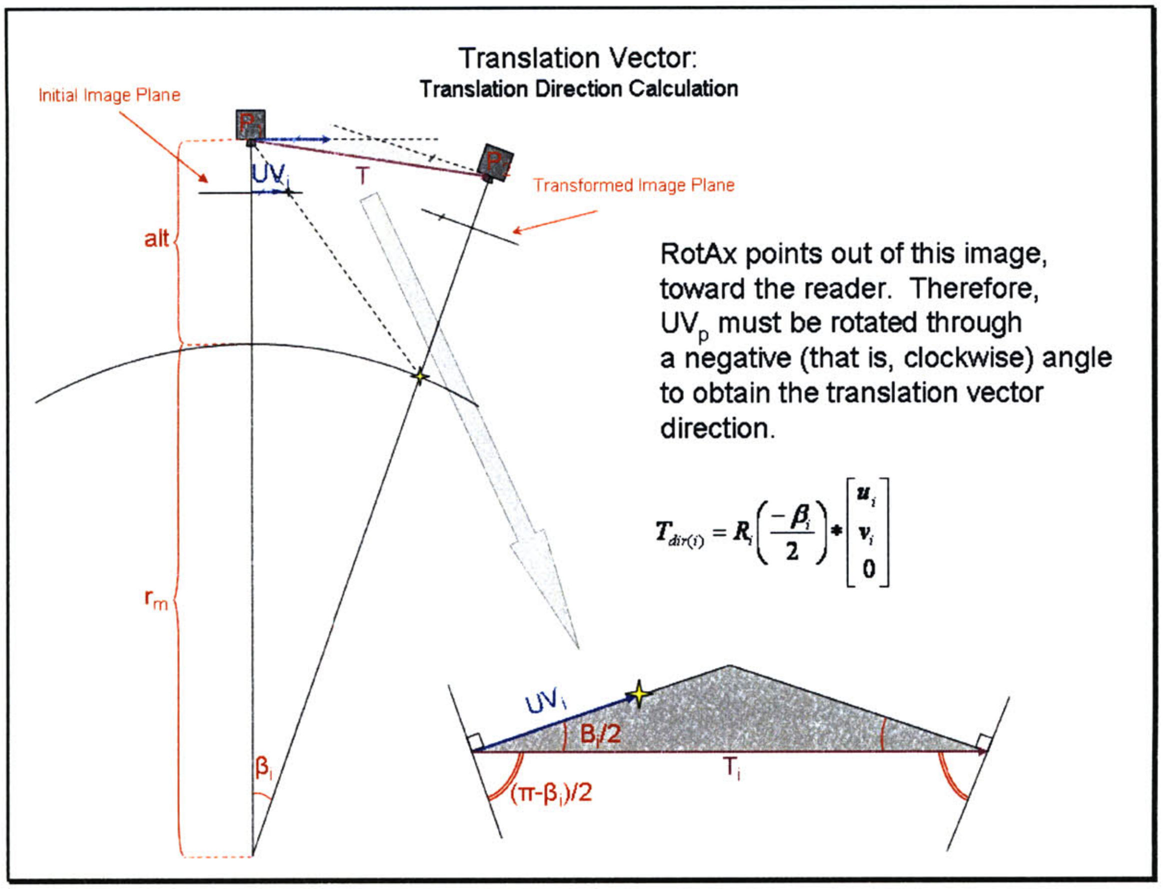

Translation Vector:
Translation Direction Calculation
Initial Image Plane
Transformed Image Plane
UV$_i$
T
alt
r_m
β_i
RotAx points out of this image, toward the reader. Therefore, UV$_p$ must be rotated through a negative (that is, clockwise) angle to obtain the translation vector direction.
$T_{dir(i)} = R_i\left(\frac{-\beta_i}{2}\right) * \begin{bmatrix} u_i \\ v_i \\ 0 \end{bmatrix}$
UV$_i$
$B_i/2$
T_i
$(\pi-\beta_i)/2$

Figure 5.11. Translation Direction Calculation.

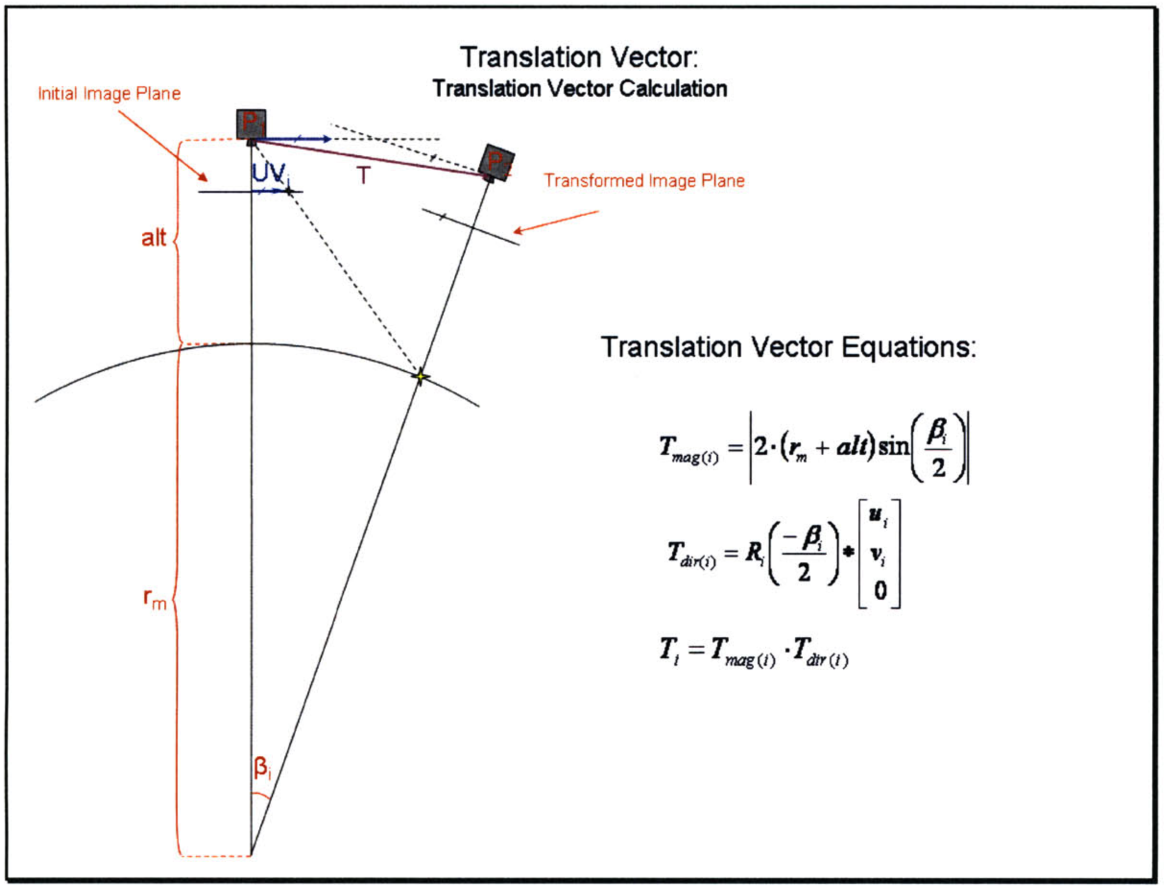
$$T_{dir(i)} = R_i\left(\frac{-\beta_i}{2}\right) * \begin{bmatrix} u_i \\ v_i \\ 0 \end{bmatrix}$$

Equation 5.12

where $R_i\left(\frac{-\beta_i}{2}\right)$ indicates that the rotation angle for the rotation matrix calculation is $\frac{\beta_i}{2}$ radians in a clockwise direction. The resultant vector, $T_{dir(i)}$ is a unit vector in the direction of the translation vector.

5.4.4 Translation Vector Calculation

After $T_{dir(i)}$ is normalized, a simply scalar multiplication of the vector $T_{dir(i)}$ by the magnitude $T_{mag(i)}$ will yield the desired translation vector, T_i.



$$T_{mag(i)} = \left| 2 \cdot (r_m + alt) \sin\left(\frac{\beta_i}{2}\right) \right|$$

$$T_{dir(i)} = R_i\left(\frac{-\beta_i}{2}\right) * \begin{bmatrix} u_i \\ v_i \\ 0 \end{bmatrix}$$

$$T_i = T_{mag(i)} \cdot T_{dir(i)}$$

Figure 5.12. Translation Vector Calculation.

$$T_i = \left| 2 \cdot (r_m + alt) \cdot \sin\left(\frac{\beta_i}{2}\right) \right| \cdot R_i\left(\frac{-\beta_i}{2}\right) * \begin{bmatrix} u_i \\ v_i \\ 0 \end{bmatrix}$$

Equation 5.13

5.5 The Perspective Transform Point-wise Depth Calculation

The previous two sections, sections 5.3 and 5.4, detail the calculation of the rotation matrix and translation vector necessary for the perspective transform. The transform also requires one other piece of information; the point-wise depth of each crater center. The j^{th} point-wise depth refers to the perpendicular (i.e. parallel to the principle axis) distance from the camera center to the altitude of the j^{th} crater center on the lunar surface. In other words, the point-wise depth is the z' component of the position vector of the actual crater centers in the camera frame. Refer to Figure 5.13, where the point-wise depth is labeled as d_j. This depth can be obtained in a single algebraic step after geometric derivation.

5.5.1 Point-wise Depth Calculation Initialization

Inputs:

4. Image frame coordinates of the j^{th} crater center.
5. Crater angle β_j for j^{th} crater center.
6. Spacecraft altitude.

Outputs:

1. Perpendicular depth to each j^{th} crater center in the original sensor image.

Constants:

1. Radius of the Moon, $r_m = 1737.4$ km.

Assumptions:

1. Moon is spherical.
2. Camera is nadir-pointing.

5.5.2 Point-wise Depth Derivation

The point-wise depth can be obtained by geometric derivation in a single step and requires only the use of simple trigonometry. The point-wise depth, d_j, only needs to be found once for each crater in the original sensor image. The selection of the desired boresight crater has no impact upon this calculation.

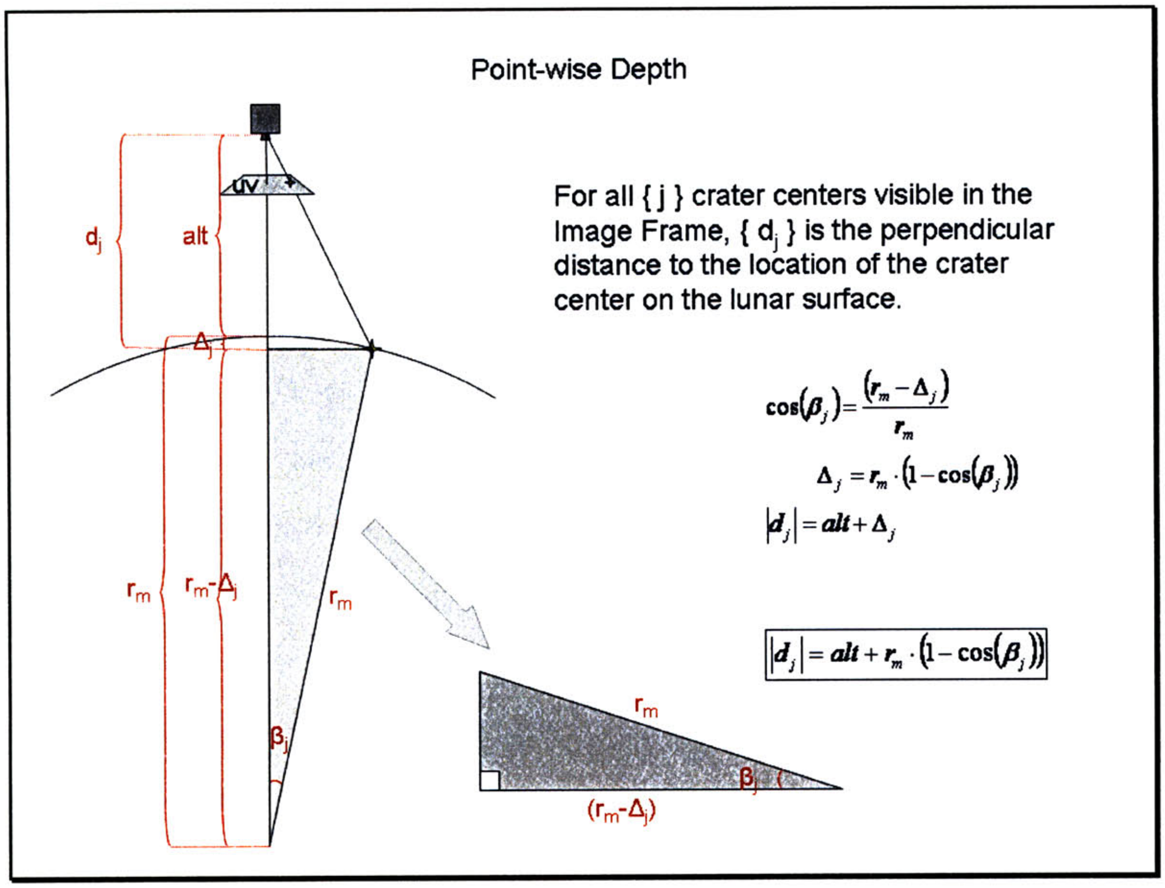
From Figure 5.13, if Δ_j can be determined, the point-wise depth will follow with ease.

From Figure 5.13, the j^{th} crater angle β_j defines the quantity Δ_j. As a result, Δ_j is calculated the trigonometric definition of β_j. With Δ_j known, the point-wise depth $|d_j|$ follows as

$$\begin{aligned} |d_j| &= alt + \Delta_j \\ |d_j| &= alt + r_m \cdot (1 - \cos(\beta_j)) \end{aligned}$$

Equation 5.14

The point-wise depth is a scalar value.



For all { j } crater centers visible in the Image Frame, { d_j } is the perpendicular distance to the location of the crater center on the lunar surface.

$$\cos(\beta_j)=\frac{(r_m-\Delta_j)}{r_m}$$

$$\Delta_j=r_m\cdot(1-\cos(\beta_j))$$

$$|d_j|=alt+\Delta_j$$

$$\boxed{|d_j|=alt+r_m\cdot(1-\cos(\beta_j))}$$

Figure 5.13. Calculation of the Point-wise Depth.

5.6 The Perspective Transform Equation

As explained in the perspective transform problem statement (see section 5.1.1), the transform entails four steps. These steps are:

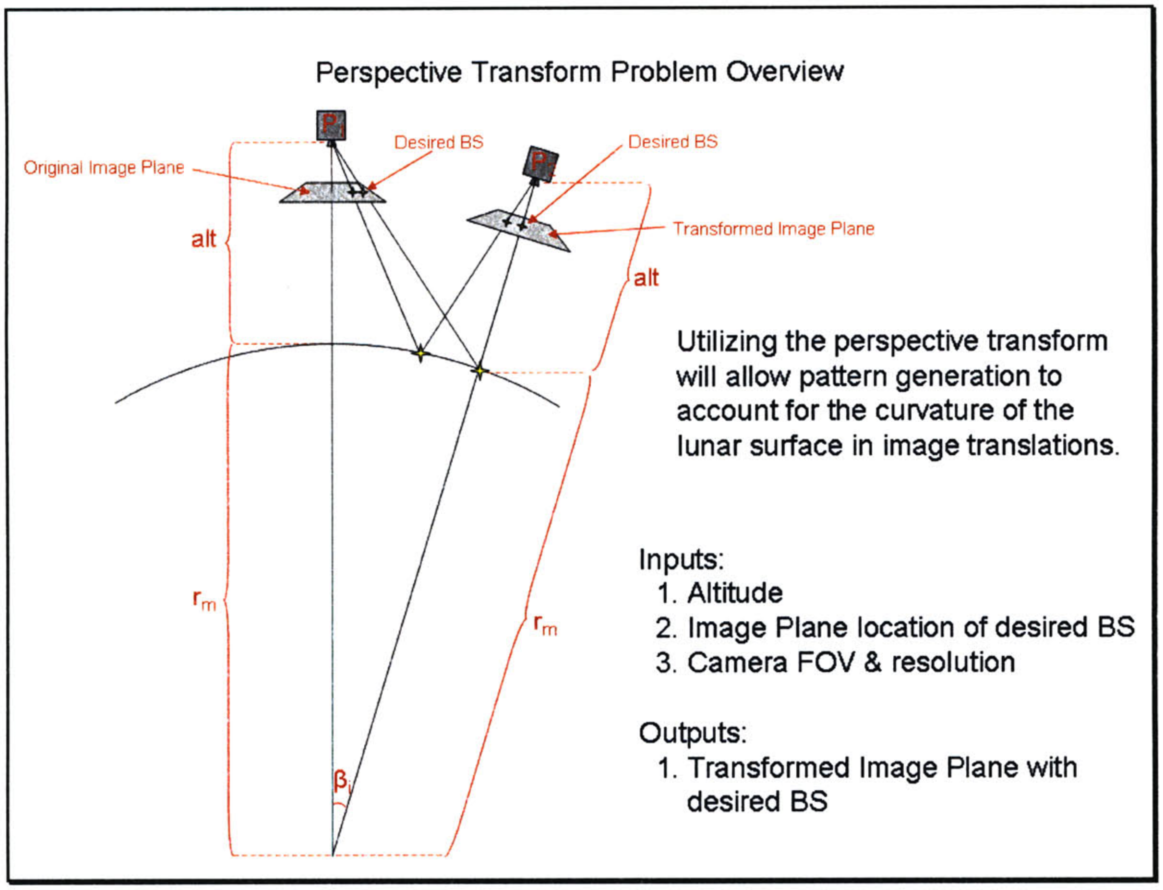
1. Calculate the rotation matrix between the image frames.
2. Calculate the translation vector between the image frames.
3. Calculate the perpendicular depth to each crater center.
4. Execute the perspective transform to generate the transformed image plane.

The first three of these steps are all preparatory for the fourth step, the implementation of the perspective transform. Sections 5.3, 5.4, and 5.5 have addressed these preparatory steps. This section will combine the results of those steps in the execution of the perspective transform.

The perspective transform is necessary to the crater matching algorithm because it allows for the modification of a given sensor image in such a way as to simulate the relocation of the principle point over various crater centers in the original image (for example, from point $P1$ to $P2$ in Figure 5.14). The perspective transform allows this simulated principle point relocation to account for the perspective differences between the two viewpoints. This perspective difference is caused by the projection of the spherical lunar surface and its features onto the flat image plane. The differences that will occur in the image plane are those apparent when 'looking' at the spherical lunar surface from two different points in space. The perspective transform will allow the transformed image plane to 'look' like the camera is centered over the desired crater center and 'looking' down upon a portion of the spherical lunar surface.

Each time the perspective transform is executed, it will be applied to all of the j craters in a given sensor image. For a given image, the transform will be executed i times, once for each potential boresight. Therefore i will identify the desired boresight for a given iteration of the transform, and j will iterate through all the craters in the sensor image for a given boresight. Reference section 5.2.4 for more details on these indices.

Figure 5.14 is included here again as pertinent reference.



Utilizing the perspective transform will allow pattern generation to account for the curvature of the lunar surface in image translations.

Inputs:
1. Altitude
2. Image Plane location of desired BS
3. Camera FOV & resolution

Outputs:
1. Transformed Image Plane with desired BS

Figure 5.14[i]. Reprinted Overview of the Perspective Transform.

5.6.1 Perspective Transform Initialization

Inputs:

1. Sensor image data points in the original image frame, $\{u_j \quad v_j\}$.

2. Rotation matrix for i^{th} desired boresight, R_i.

3. Translation matrix for i^{th} desired boresight, T_i.

4. Point-wise depth for all j craters in the sensor image, d_j.

Outputs:

[i] Also Figure 4.6

1. Transformed image data points, $\{\tilde{u}_j \quad \tilde{v}_j\}$ for all j craters in the sensor image.

Constants:

1. None

Assumptions:

1. Moon is spherical.
2. Camera is nadir-pointing.
3. Spacecraft altitude does not change over transform.
4. Image plane is unit distance from the camera.
5. Camera is an ideal camera.

5.6.2 Perspective Transform Derivation

This section will derive the perspective transform as a mapping between the original and transformed camera frames. In order to do this, a generalized perspective transform will first be derived. After the derivation of the generalized transform, it will be modified so that it will apply to the crater matching routine and will be simplified accordingly.

5.6.2.1 Generalized World Frame to Camera Frame Mapping

Define the generic camera frame to be a Euclidean frame centered at the center of projection[1] of a camera. Define the generic world frame to be a specific right-handed three-space different from the generic camera frame.

For sections 5.6.2.1 and 5.6.2.2, the generic camera frame will always be referred to using the word 'generic' to differentiate it from the camera frame specified in section 5.1.2. (After these sections, the generic camera frame will no longer be referenced.) The generic world frame will be referred to as

[1] The center of projection for a camera is the point at which the lens focuses all incoming rays of light.

the world frame because it does not overlap terminology previously used in coordinate frame definitions.

When considering a pinhole camera,[1] the general mapping from the world frame into the generic camera frame is given by:[23]

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = M * R * \begin{bmatrix} I_{3x3} & -T \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Equation 5.15

where

1. $\{X \quad Y \quad Z\}$ represents an ordered triplet in the world frame,
2. $\{X' \quad Y' \quad Z'\}$ represents an ordered triplet in the generic camera frame
3. R is the 3 x 3 rotation matrix from the world frame to the generic camera frame
4. T is the 3 x 1 translation vector to align the world and generic camera frame origins
5. $M = \begin{bmatrix} f_{X'} & 0 & p_{X'} \\ 0 & f_{Y'} & p_{Y'} \\ 0 & 0 & 1 \end{bmatrix}$

Equation 5.16

6. $\{f_{X'} \quad f_{Y'}\}$ is the X' and Y' units in terms of focal length
7. $\{p_{X'} \quad p_{Y'}\}$ is the principal point location in the generic camera frame coordinates.

5.6.2.2 Application of Generalized Mapping to Perspective Transform Camera Model and Coordinate Frames

The application of Equation 5.15 will allow the transform from the camera frame to the image frame as defined in section 5.1.2 to be found. Allow the following conditions to be made:

[1] A pinhole camera is a camera with a small aperture and no lenses. It is useful due to the relative simplicity of the projection of the image onto the image plane.

1. Allow the generic world frame to be the camera frame defined in section 5.1.2; that is,

$$\{X \quad Y \quad Z\} = \{x' \quad y' \quad z'\}$$

Equation 5.17

2. Allow the generic camera frame to be the image frame defined in section 5.1.2;[1] that is,

$$\{X' \quad Y' \quad Z'\} = W \cdot \{u \quad v \quad 1\}$$

Equation 5.18

where

$$\{U \quad V \quad W\} = W \cdot \{u \quad v \quad 1\}$$

Equation 5.19

Equation 5.18 highlights that W is the perpendicular distance from the camera to the crater center on the lunar surface in the transformed camera frame.

3. Since the image frame is a planar subspace of the three-space camera frame (defined in section 5.1.2). Refer to Equation 5.1. Therefore $R = I_{3x3}$ and $T = [0 \quad 0 \quad 0]'$. This definition of $\{R \quad T\}$ means that the $\{X \quad Y\} = \{x' \quad y'\}$ coordinate axes are aligned. Refer to Figure 5.3.
4. Allow the focal length $f_{X'} = f_{Y'} = f = 1$. This implies that scaling is identical in both the X' and Y' directions.
5. Allow the principal point to be the origin of the image frame. This means that the origin of the image plane lies in the center of the image. Refer to Figure 5.3 and Figure 5.4.

Applying these conditions to Equation 5.15 yields:

[1] The author realizes that specifying the three dimensional generic camera frame as the planar image frame may introduce added complexity. Other simplifications made will cause this difficulty to become irrelevant. Reference the remainder of the derivation for more details.

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = I_{3x3} * I_{3x3} * \begin{bmatrix} I_{3x3} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} * \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$W \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Equation 5.20

This equation defines the projection of points in the camera frame into points on the image frame for the given camera assumptions and pertinent coordinate frames.

5.6.2.3 Application of Generalized Mapping to this Perspective Transform

Allow two equations of the form of Equation 5.20 to be constructed such that

$$\begin{bmatrix} u_{j,1} \\ v_{j,1} \\ 1 \end{bmatrix} = \frac{1}{W_{j,1}} \cdot \begin{bmatrix} x'_{j,1} \\ y'_{j,1} \\ z'_{j,1} \end{bmatrix}$$

Equation 5.21

$$\begin{bmatrix} u_{j,2} \\ v_{j,2} \\ 1 \end{bmatrix} = \frac{1}{W_{j,2}} \cdot \begin{bmatrix} x'_{j,2} \\ y'_{j,2} \\ z'_{j,2} \end{bmatrix}$$

Equation 5.22

Let Equation 5.21 represent the image plane at point $P1$ from Figure 5.1, and Equation 5.22 to represent the image plane at point $P2$ from the same figure. Since the perspective transform must transform the original image frame into the transformed image plane. In other words, the transform must generate the ordered triplet $\{u_{j,2} \quad v_{j,2} \quad 1\}$ given the ordered triplet $\{u_{j,1} \quad v_{j,1} \quad 1\}$. Therefore, a relationship between these two ordered triplets must be found to allow for the derivation of the perspective transform.

The transformation between any two Euclidean three-spaces can be defined as

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = R * \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} - T$$

Equation 5.23

From the derivation of R_t and T_t, the relationship between the original and transformed camera frames is

$$\begin{bmatrix} x'_{j,2} \\ y'_{j,2} \\ z'_{j,2} \end{bmatrix} = R_t * \begin{bmatrix} x'_{j,1} \\ y'_{j,1} \\ z'_{j,1} \end{bmatrix} - T_t$$

Equation 5.24

where the ordered triplet $\{x'_{j,1} \quad y'_{j,1} \quad z'_{j,1}\}$ is a point in the original camera frame, and $\{x'_{j,2} \quad y'_{j,2} \quad z'_{j,2}\}$ a point in the desired camera frame.

By substitution of Equation 5.24 and Equation 5.21 into Equation 5.22,

$$\begin{bmatrix} u_{j,2} \\ v_{j,2} \\ 1 \end{bmatrix} = \frac{1}{W_{j,2}} \cdot \begin{bmatrix} x'_{j,2} \\ y'_{j,2} \\ z'_{j,2} \end{bmatrix}$$

$$W_{j,2} \cdot \begin{bmatrix} u_{j,2} \\ v_{j,2} \\ 1 \end{bmatrix} = R_t * \begin{bmatrix} x'_{j,1} \\ y'_{j,1} \\ z'_{j,1} \end{bmatrix} - T_t$$

$$W_{j,2} \cdot \begin{bmatrix} u_{j,2} \\ v_{j,2} \\ 1 \end{bmatrix} = R_t * W_{j,1} \cdot \begin{bmatrix} u_{j,1} \\ v_{j,1} \\ 1 \end{bmatrix} - T_t$$

Equation 5.25

This can be rewritten as

$$W_{j,2} \cdot \begin{bmatrix} u_{j,2} \\ v_{j,2} \\ 1 \end{bmatrix} = \begin{bmatrix} R_t & -T_t \end{bmatrix} * \begin{bmatrix} W_{j,1} \cdot \begin{bmatrix} u_{j,1} \\ v_{j,1} \\ 1 \end{bmatrix} \\ 1 \end{bmatrix}$$

Equation 5.26

This equation is the perspective transform that characterizes the relationship between points in the original image plane and the transformed image plane.

5.6.3 The Perspective Transform Equation for this Problem

If Equation 5.26 is rewritten to utilize the notation in use throughout the rest of this paper, the result is

$$c_j \cdot \begin{bmatrix} \tilde{u}_j \\ \tilde{v}_j \\ 1 \end{bmatrix} = \begin{bmatrix} R_t & -T_t \end{bmatrix} * \begin{bmatrix} d_j \cdot \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} \\ 1 \end{bmatrix}$$

Equation 5.27

where $\{\tilde{u}_j \quad \tilde{v}_j \quad 1\}$ represents the transformed image plane coordinates and c_j is the point-wise depth of the transformed image points. In other words, c_j is the point-wise depth of the image points if the principle axis passed through the desired camera location $(P2)$ and the desired boresight crater.

Equation 5.27 defines the relationship between the original image frame and the transformed image frame. It requires the rotation matrix R_t and translation vector T_t between both of these frames, as well as the point-wise depth d_j to each point in the original camera frame.

The perspective transform equation will be utilized in the crater matching routine to modify the original sensor image such that a selected 'boresight' crater will coincide with the principle point of the image plane after the transformation. From this transformed image, the pattern of craters surrounding the centered boresight crater will be used to uniquely identify the boresight crater. That is, the crater pattern in the original sensor image will be compared with the patterns in the database of lunar crater patterns and a match determined. If most of the boresight craters selected from the original image match to a cluster of craters in the same general area from the database, the image can be correlated to that geographic area of the lunar sphere. This information will then allow the spacecraft pose to be estimated.

In order for the transformation to be realistic (and therefore useful), it must accurately model the shift in location of the craters relative to one another across the transform. This shift in the relative crater pattern is due to the curvature of the moon, which changes the apparent crater pattern as the camera boresight is shifted from one point to another on the lunar surface.

The perspective transform given in Equation 5.27 proves useful precisely because it accounts for the expected shift in the relative pattern of craters in the sensor image across a boresight shift. In other words, it allows the sensor image to be modified in such a way as to simulate that the image had been taken with a different principle point. Due to this characteristic, this perspective transform will be utilized in the crater matching routine to 'center' the image upon various crater centers, allowing the pattern of surrounding craters to be identified for each selected 'boresight' crater.

For the remainder of this paper, the entire process of generating the proper $\{R_i \quad T_i \quad d_i\}$ components as well as executing the perspective transform equation (Equation 5.27) on a given set of data will be referred to as the perspective transform.

5.7 Perspective Transform Implementation and Results

After the derivation of the transform, it was implemented and the overall performance of the algorithm was tested in preparation for its use in the crater matching routine. This section will present several analyses conducted to better characterize the perspective transform algorithm. These analyses are intended to allow a better understanding of the algorithm and its role in application to the crater matching routine.

For several of these analyses, a Euclidean translation of the image was implemented as well to serve as a basis of comparison. This Euclidean translation simply shifted the image such that the desired boresight crater was relocated to the principle point of the image, and all other craters shifted to maintain their relative position.[1]

The following is a brief description of the analyses presented here:

1. Accuracy of Centering the Desired Boresight

   An analysis of how accurately the transform centers the desired boresight in the transformed image.

2. Image Translation vs. Image Transformation

[1] It is interesting to note that this simple Euclidean translation of the image was used by the star tracker algorithms on which this crater matching routine was based. This follows from the assumption that the stars are at an infinite distance from the star tracker.

A comparative analysis between Euclidean translation of the image and the perspective transform of the image.

3. Change in the Normalized Distances between Crater Centers

   An analysis of the change in the normalized distance between any two given craters in the image over the perspective transform.

In addition, an overview of the coding of the algorithm will follow these result sections. The algorithm was coded in Matlab version 7.1 to allow for ease of testing and analysis. This code is not intended for use onboard the spacecraft.

5.7.1 Analysis Initialization

For the following analyses, the algorithm initialization parameters will be as follows:

Inputs:

1. Spacecraft Altitude: $100\ km$
2. Spacecraft attitude: 0° roll, 0° pitch, and 0° yaw
3. Camera Field of View: 90°
4. Camera Resolution: $512\ x\ 512\ pixels$

5. Original Sensor Image (as data points):

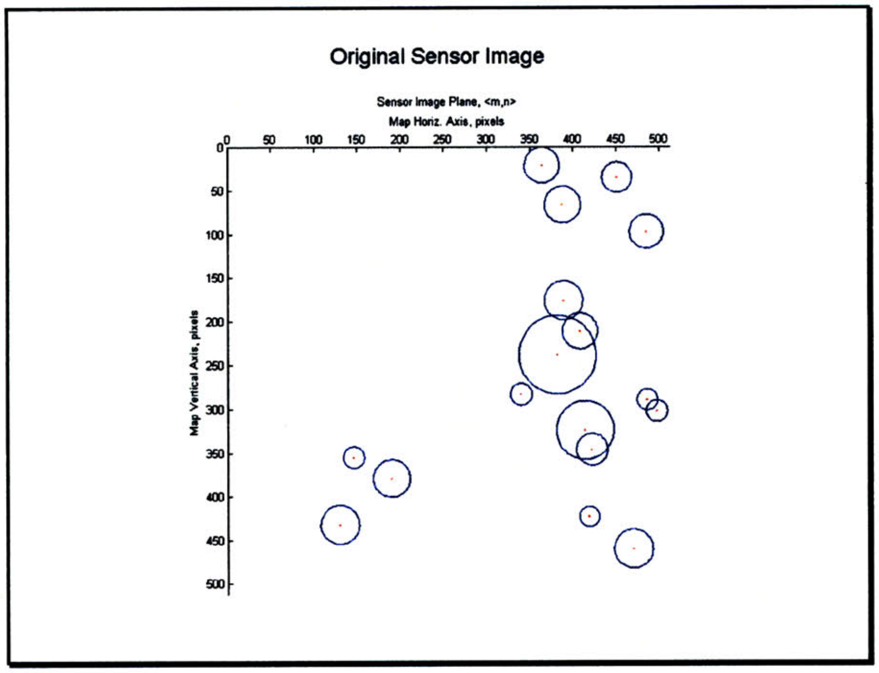


Figure 5.15[i]. Reprint Example Data Points for Pattern Vector Generation.

(image taken at 80° latitude and 85° longitude)

[i] Also Figure 3.5

6. Potential Boresight Craters:

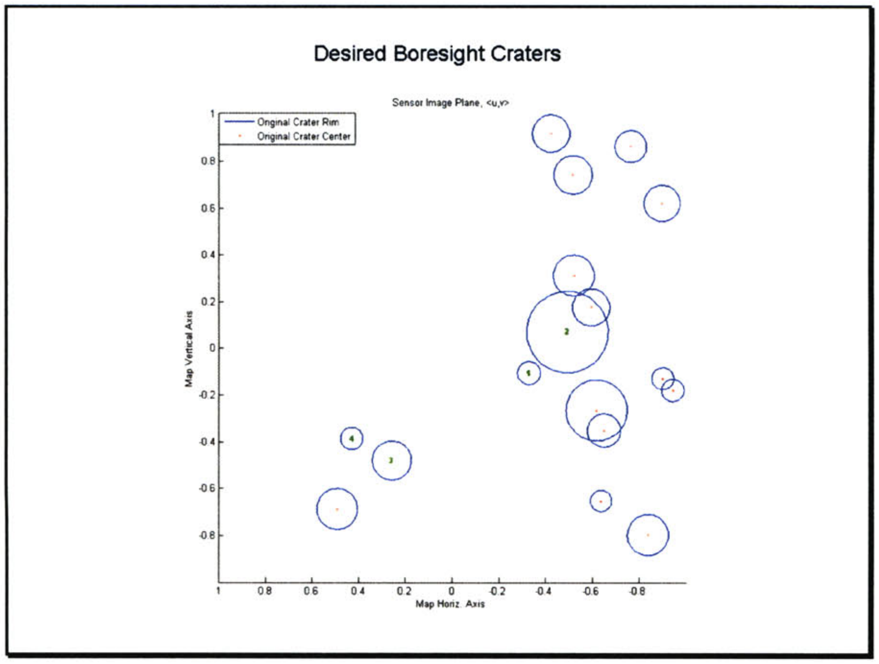


Figure 5.16. Example Data Points with Potential Boresight Craters Highlighted.

The potential boresight craters are numbered in this image. The examples presented throughout these analyses will refer to the boresight craters in this order.

5.7.2 Accuracy of Centering the Desired Boresight

One expected effect of the perspective transform is to center the transformed image on the desired boresight crater. As a result, a complete analysis of the transform must verify that the desired boresight crater is actually the principle point in the transformed image.

See Figure 5.17 and Figure 5.18, which presents the centering for the first potential boresight crater. The desired boresight crater is shown first in both the original and translated images, and secondly in the original and transformed images. Both of these figures demonstrate the effective centering of the boresight.

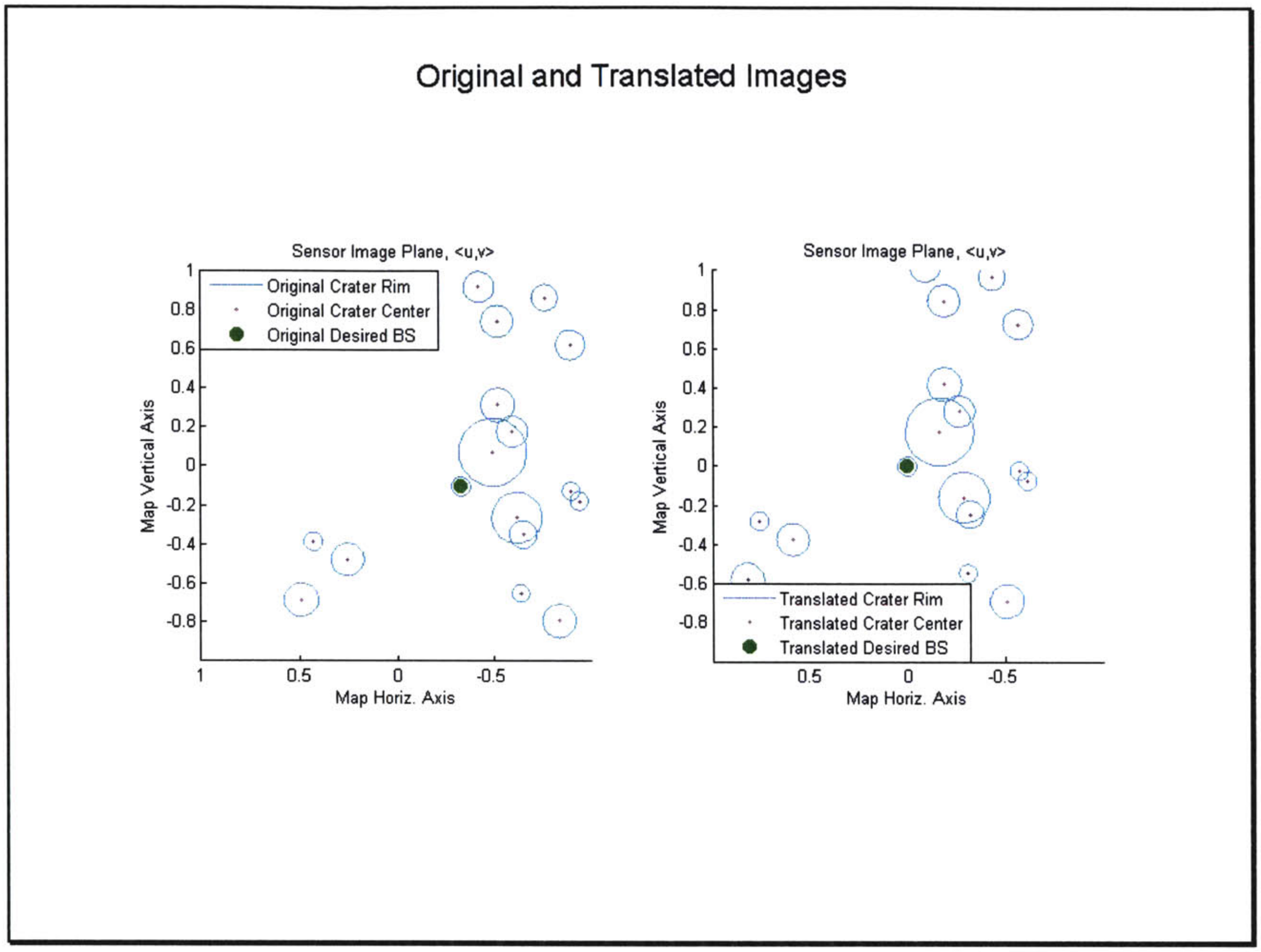


Figure 5.17. Euclidean Translation of the Example Data Points.

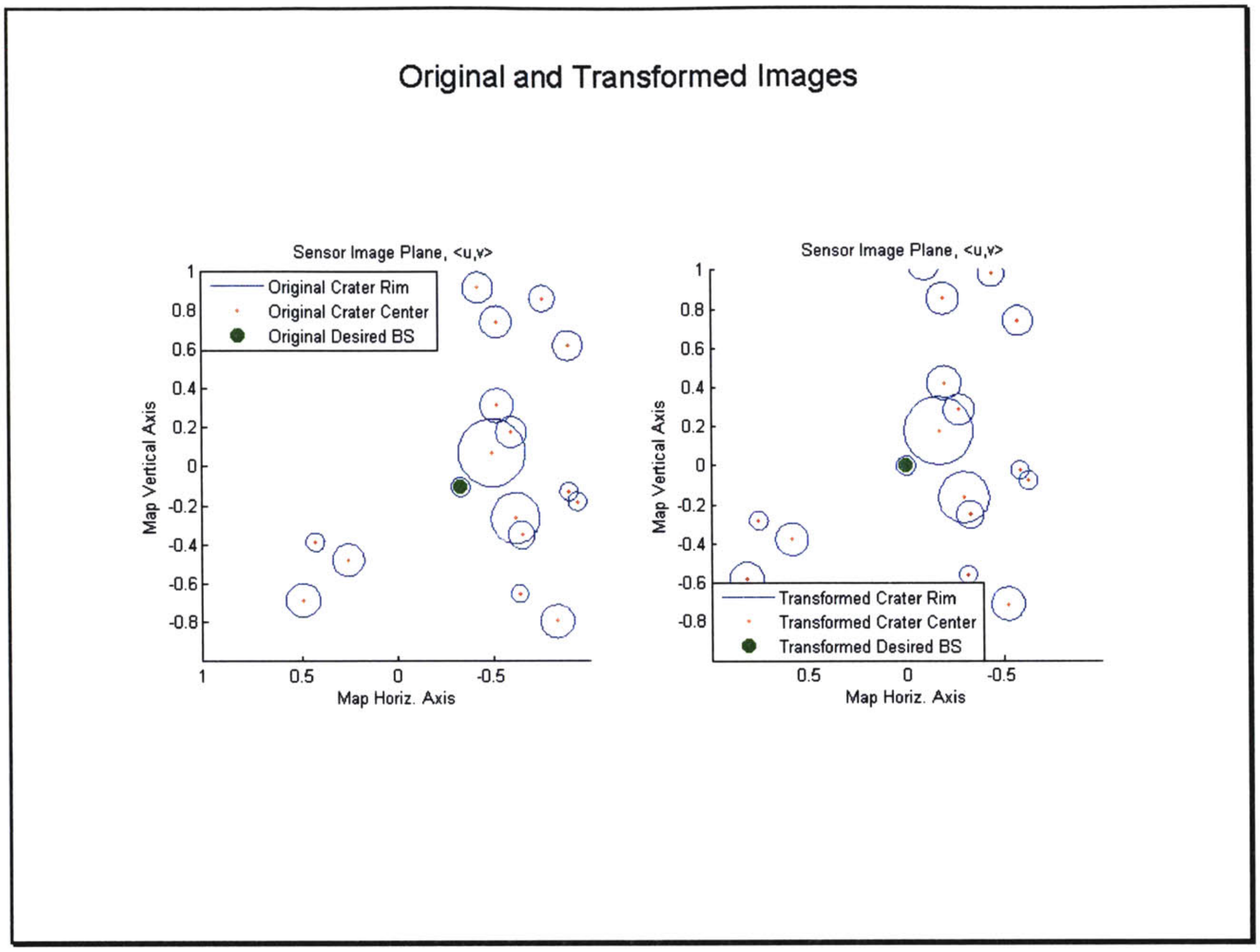


Figure 5.18. Perspective Transformation of the Example Data Points.

The offset of the transformed position of the desired boresight crater and the principle point of the image is presented in $\{m' \quad n'\}$ pixels. See Table 5-1.

Table 5-1. Comparison of Euclidean Translation and Perspective Transform Error.

	Euclidean Translation Error		**Perspective Transform Error**	
	m'	**n'**	**m'**	**n'**
BS #1	0	0	1.42×10^{-13}	4.55×10^{-13}
BS #2	0	0	-2.27×10^{-13}	1.25×10^{-12}
BS #3	0	0	8.53×10^{-13}	-5.12×10^{-13}
BS #4	0	0	-5.68×10^{-14}	5.68×10^{-14}
Average	**0**	**0**	$\mathbf{1.78 \times 10^{-13}}$	$\mathbf{3.13 \times 10^{-13}}$

It was verified that the offset of the translation of the desired boresight crater image and the principle point is effectively zero, as expected. This verifies that the offset error of the transformed case is the result of the computer processor, and not the perspective transform process.

5.7.3 Image Translation vs. Image Transformation

In addition to centering the desired boresight upon the principle point in the transformed image, the perspective transform will also cause slight differences in the relative positions of the other crater centers. This is because the camera's view of the craters in relation to one another will change over the transform because the lunar surface is curved. This effect is somewhat more difficult to test than the centering of the boresight crater. However, a comparison between a Euclidean translation of the image and the perspective transform will highlight the differences in relative position of the crater centers over the re-centering. Due to the constancy of the relative position of the craters in the Euclidean image translation, this comparison will effectively show the effects of perspective introduced by the perspective transform algorithm. For Figure 5.19, Figure 5.20, Figure 5.21, and Figure 5.22, note the offset between the crater centers and rims centered by the Euclidean translation and those repositioned by the perspective transform.

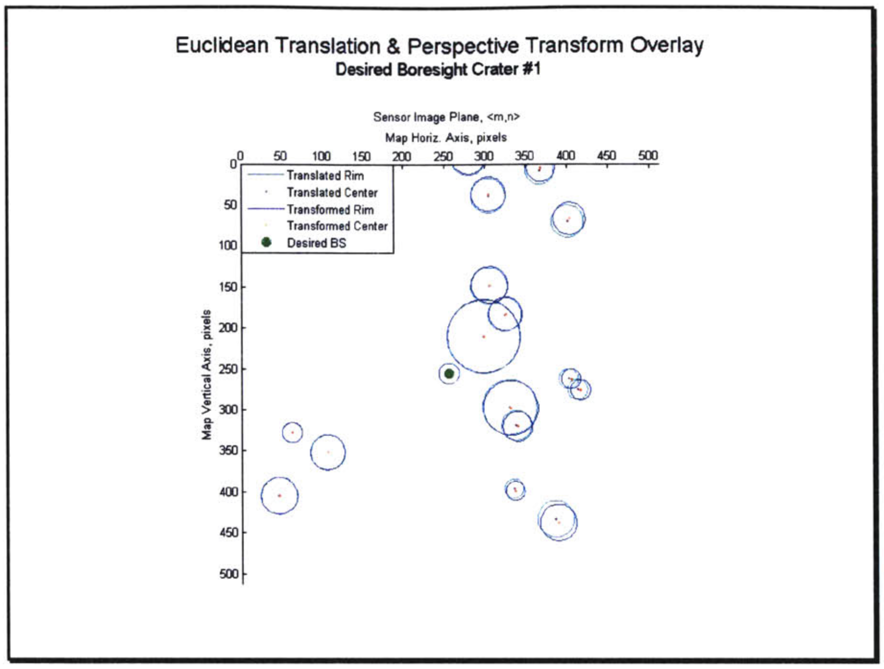


Figure 5.19[i]. Reprint Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points.

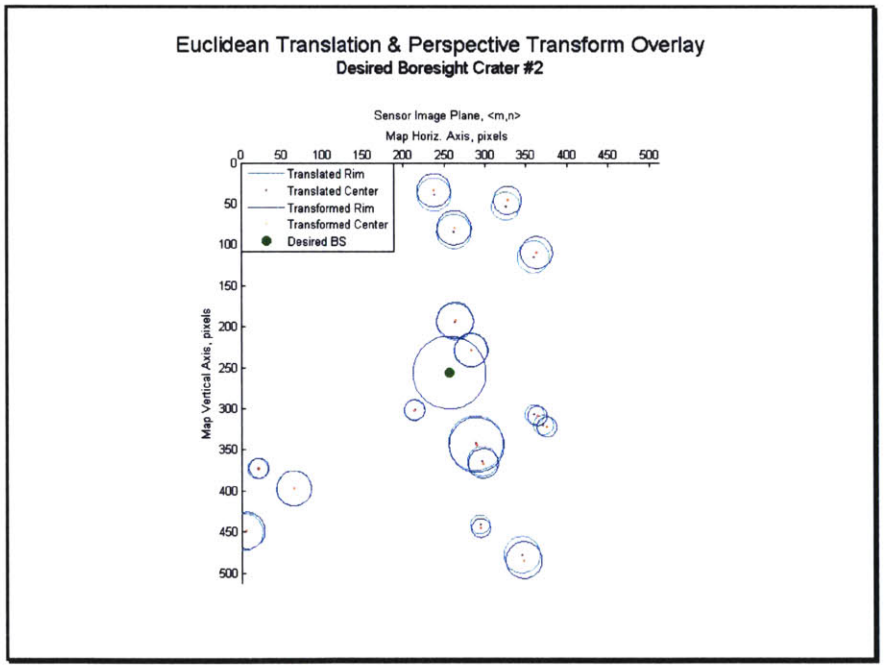


Figure 5.20. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points for Boresight Crater # 2.

[i] Also Figure 4.9

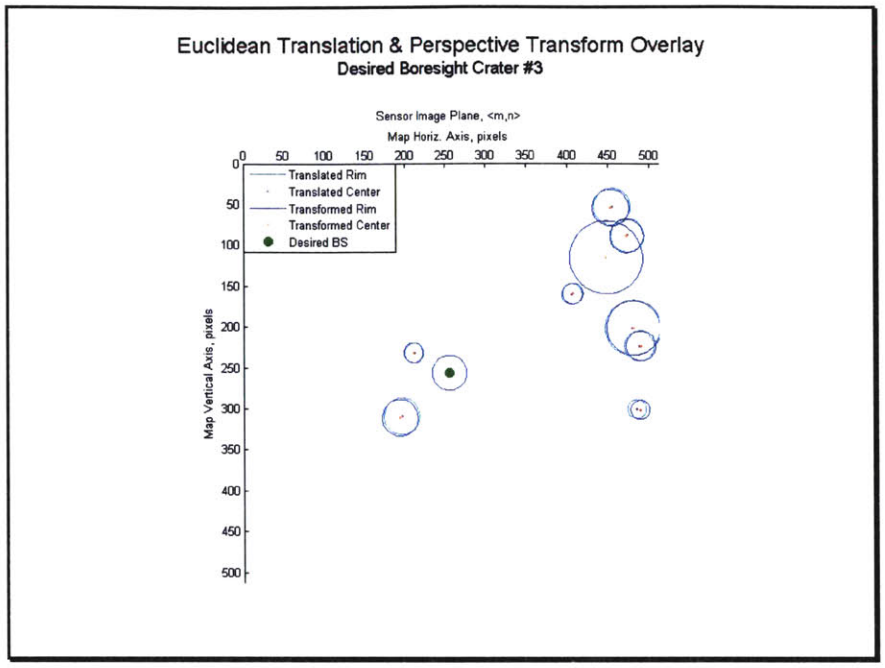


Figure 5.21. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points for Boresight Crater # 3.

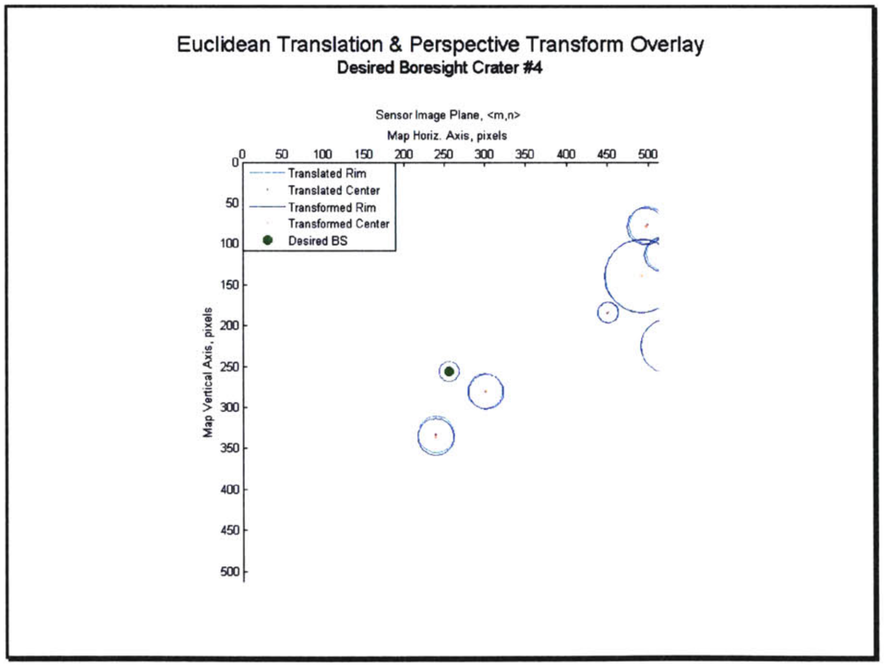


Figure 5.22. Overlay of the Euclidean Translation and the Perspective Transform of the Example Data Points for Boresight Crater # 4.

Figure 5.19, Figure 5.20, Figure 5.21, and Figure 5.22 present two overlaid images. The first of the overlaid images in each figure is that of the Euclidean image translation, where the original sensor image was translated to align the desired boresight crater and the principle point. The second of the overlaid images is that of the perspective transform of the original sensor image. There is a slight, but definite, difference between these two images in each of the four cases present in these figures.

To better characterize this difference between the two images, the offset between the corresponding transformed and translated craters was calculated. For the given original sensor image, the minimum, maximum, and mean distance between the translated and transformed crater centers was calculated, along with the standard deviation of the distances. These calculations were repeated for all four potential boresight craters selected from the original sensor image. Table 5-2 presents the results of this analysis.

Table 5-2. Distance between the Translated and Transformed Crater Centers.

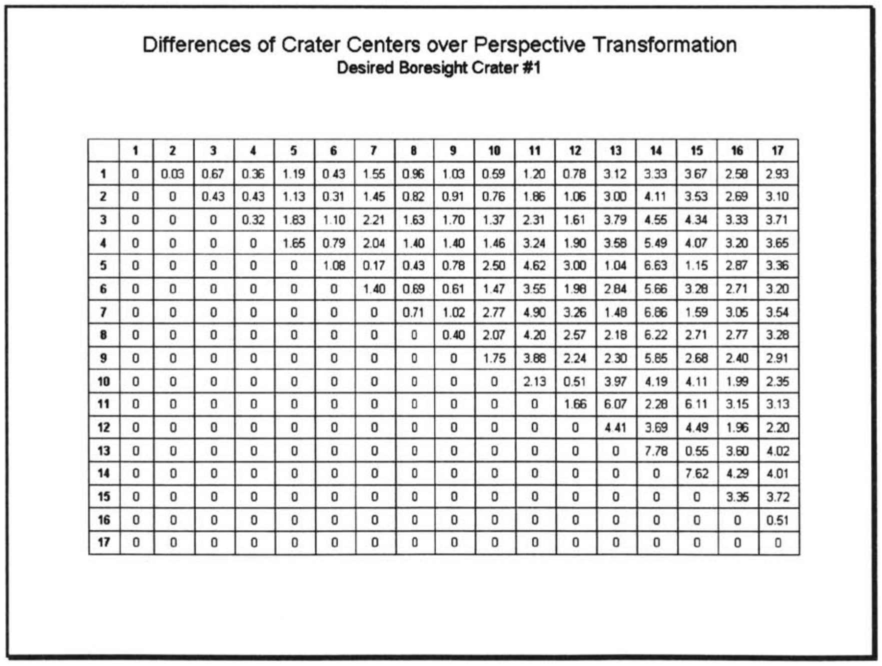
	Mean Dist. (pixels)	Minimum Dist. (pixels)	Maximum Dist. (pixels)	Std. Dev. (pixels)
BS #1	2.06	4.76×10^{-13}	5.49	1.57
BS #2	3.38	1.27×10^{-12}	7.36	2.33
BS #3	3.47	9.94×10^{-13}	11.09	3.52
BS #4	3.68	8.04×10^{-14}	12.11	3.65
Average	**3.15**	$\mathbf{7.06 \times 10^{-13}}$	**9.01**	**2.78**

5.7.4 Change in the Normalized Distances between Crater Centers

Section 5.7.3 demonstrated that the relative position of the craters will change due to the perspective transform and calculated the variation using the Euclidean translation craters as a point of reference. Another way to verify this relative position change is to calculate the distance between craters before and after the transform. Rather than solely providing the difference between the translated reference and the transformed craters, this calculation of the distance between craters will highlight the changes due to the transform in relation to all the craters. As a result, this calculation will provide greater detail as to the effect of the transform on the image as a whole rather than looking at individual craters.

The author demonstrates that the normalized distance between two given points in the original sensor image should vary over the perspective transform in Appendix B. Therefore, this analysis serves as an additional database of numerical results to solidify the conclusions of the counter-example given in Appendix B.

This analysis will first calculate the distance between every crater center in the original image, and compare these distances to the distances calculated between craters in the transformed image. The raw calculated differences in distances between the same craters in the original and transformed images are presented in Table 5-3, Table 5-4, Table 5-5, and Table 5-6.

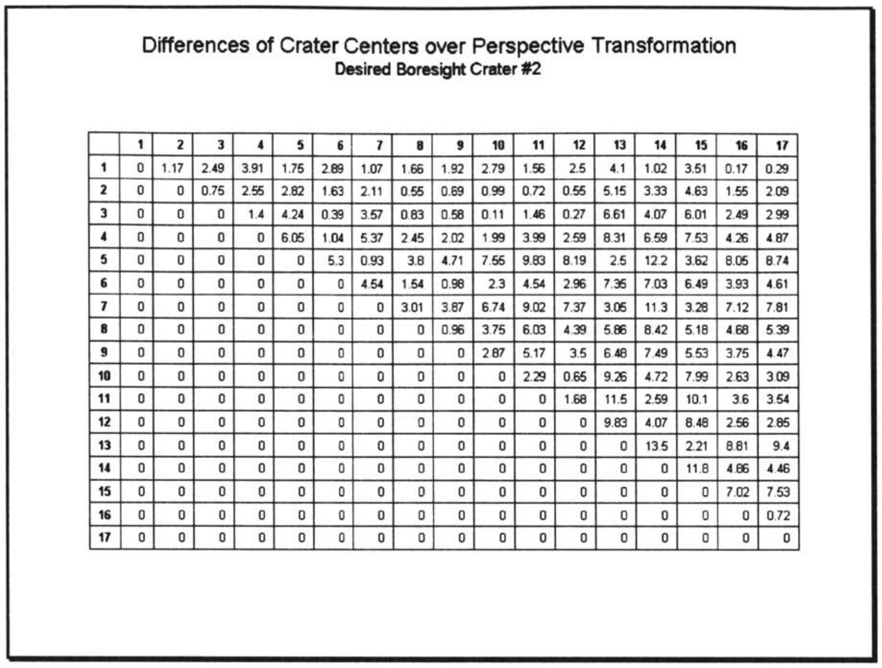


Differences of Crater Centers over Perspective Transformation

Desired Boresight Crater #1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	0.03	0.67	0.36	1.19	0.43	1.55	0.96	1.03	0.59	1.20	0.78	3.12	3.33	3.67	2.58	2.93
2	0	0	0.43	0.43	1.13	0.31	1.45	0.82	0.91	0.76	1.86	1.06	3.00	4.11	3.53	2.69	3.10
3	0	0	0	0.32	1.83	1.10	2.21	1.63	1.70	1.37	2.31	1.61	3.79	4.55	4.34	3.33	3.71
4	0	0	0	0	1.65	0.79	2.04	1.40	1.40	1.46	3.24	1.90	3.58	5.49	4.07	3.20	3.65
5	0	0	0	0	0	1.08	0.17	0.43	0.78	2.50	4.62	3.00	1.04	6.63	1.15	2.87	3.36
6	0	0	0	0	0	0	1.40	0.69	0.61	1.47	3.55	1.98	2.84	5.66	3.28	2.71	3.20
7	0	0	0	0	0	0	0	0.71	1.02	2.77	4.90	3.26	1.48	6.86	1.59	3.05	3.54
8	0	0	0	0	0	0	0	0	0.40	2.07	4.20	2.57	2.18	6.22	2.71	2.77	3.28
9	0	0	0	0	0	0	0	0	0	1.75	3.88	2.24	2.30	5.85	2.68	2.40	2.91
10	0	0	0	0	0	0	0	0	0	0	2.13	0.51	3.97	4.19	4.11	1.99	2.35
11	0	0	0	0	0	0	0	0	0	0	0	1.66	6.07	2.28	6.11	3.15	3.13
12	0	0	0	0	0	0	0	0	0	0	0	0	4.41	3.69	4.49	1.96	2.20
13	0	0	0	0	0	0	0	0	0	0	0	0	0	7.78	0.55	3.60	4.02
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7.62	4.29	4.01
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.35	3.72
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.51
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-3. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #1.

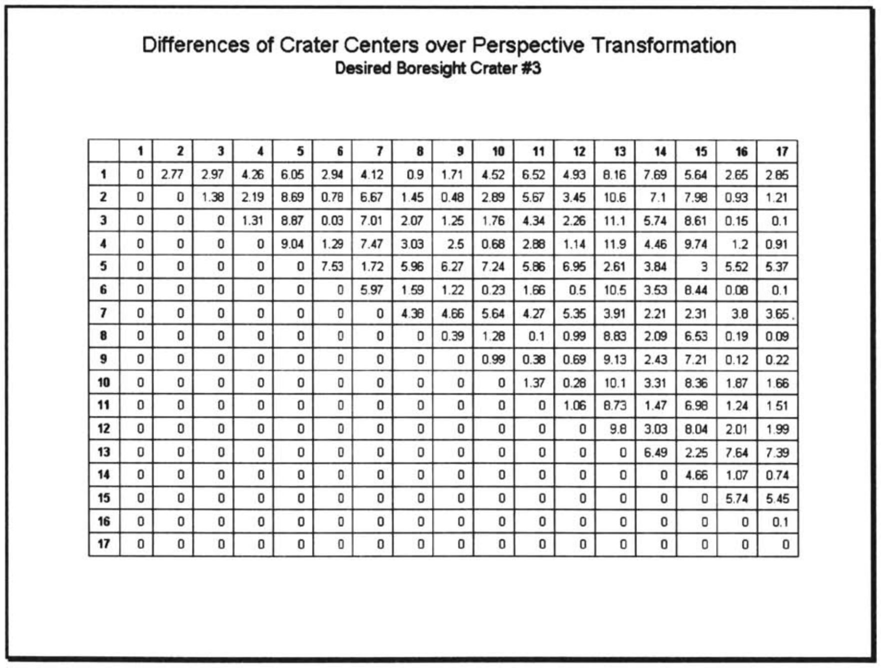


Differences of Crater Centers over Perspective Transformation

Desired Boresight Crater #2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	1.17	2.49	3.91	1.75	2.89	1.07	1.66	1.92	2.79	1.56	2.5	4.1	1.02	3.51	0.17	0.29
2	0	0	0.75	2.55	2.82	1.63	2.11	0.55	0.69	0.99	0.72	0.55	5.15	3.33	4.63	1.55	2.09
3	0	0	0	1.4	4.24	0.39	3.57	0.83	0.58	0.11	1.46	0.27	6.61	4.07	6.01	2.49	2.99
4	0	0	0	0	6.05	1.04	5.37	2.45	2.02	1.99	3.99	2.59	8.31	6.59	7.53	4.26	4.87
5	0	0	0	0	0	5.3	0.93	3.8	4.71	7.55	9.83	8.19	2.5	12.2	3.62	8.05	8.74
6	0	0	0	0	0	0	4.54	1.54	0.98	2.3	4.54	2.96	7.35	7.03	6.49	3.93	4.61
7	0	0	0	0	0	0	0	3.01	3.87	6.74	9.02	7.37	3.05	11.3	3.28	7.12	7.81
8	0	0	0	0	0	0	0	0	0.96	3.75	6.03	4.39	5.86	8.42	5.18	4.68	5.39
9	0	0	0	0	0	0	0	0	0	2.87	5.17	3.5	6.48	7.49	5.53	3.75	4.47
10	0	0	0	0	0	0	0	0	0	0	2.29	0.65	9.26	4.72	7.99	2.63	3.09
11	0	0	0	0	0	0	0	0	0	0	0	1.68	11.5	2.59	10.1	3.6	3.54
12	0	0	0	0	0	0	0	0	0	0	0	0	9.83	4.07	8.48	2.56	2.85
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13.5	2.21	8.81	9.4
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11.8	4.86	4.46
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7.02	7.53
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.72
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-4. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #2.

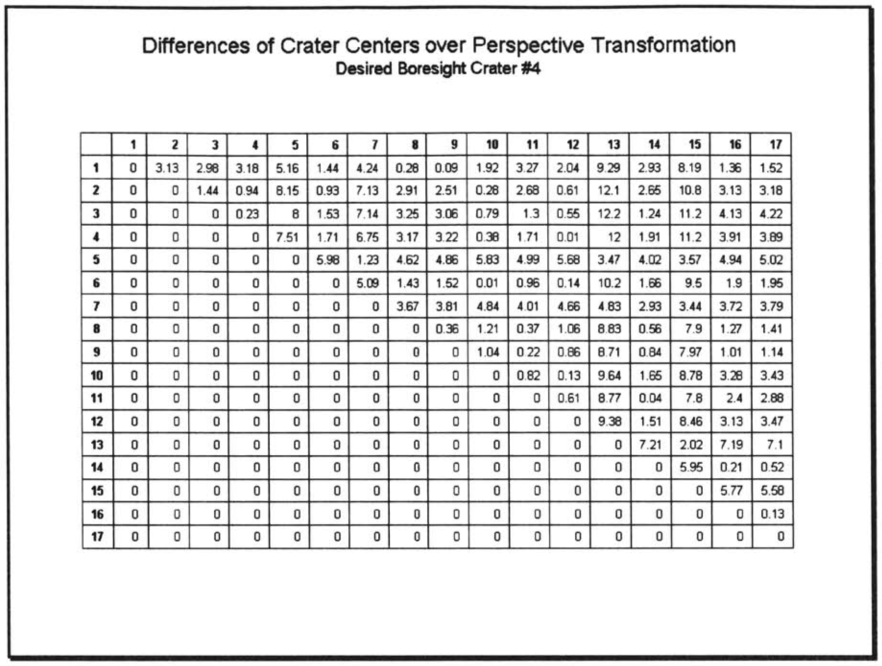


Differences of Crater Centers over Perspective Transformation

Desired Boresight Crater #3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	2.77	2.97	4.26	6.05	2.94	4.12	0.9	1.71	4.52	6.52	4.93	8.16	7.69	5.64	2.65	2.85
2	0	0	1.38	2.19	8.69	0.78	6.67	1.45	0.48	2.89	5.67	3.45	10.6	7.1	7.98	0.93	1.21
3	0	0	0	1.31	8.87	0.03	7.01	2.07	1.25	1.76	4.34	2.26	11.1	5.74	8.61	0.15	0.1
4	0	0	0	0	9.04	1.29	7.47	3.03	2.5	0.68	2.88	1.14	11.9	4.46	9.74	1.2	0.91
5	0	0	0	0	0	7.53	1.72	5.96	6.27	7.24	5.86	6.95	2.61	3.84	3	5.52	5.37
6	0	0	0	0	0	0	5.97	1.59	1.22	0.23	1.66	0.5	10.5	3.53	8.44	0.08	0.1
7	0	0	0	0	0	0	0	4.38	4.66	5.64	4.27	5.35	3.91	2.21	2.31	3.8	3.65
8	0	0	0	0	0	0	0	0	0.39	1.28	0.1	0.99	8.83	2.09	6.53	0.19	0.09
9	0	0	0	0	0	0	0	0	0	0.99	0.38	0.69	9.13	2.43	7.21	0.12	0.22
10	0	0	0	0	0	0	0	0	0	0	1.37	0.28	10.1	3.31	8.36	1.87	1.66
11	0	0	0	0	0	0	0	0	0	0	0	1.06	8.73	1.47	6.98	1.24	1.51
12	0	0	0	0	0	0	0	0	0	0	0	0	9.8	3.03	8.04	2.01	1.99
13	0	0	0	0	0	0	0	0	0	0	0	0	0	6.49	2.25	7.64	7.39
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.66	1.07	0.74
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.74	5.45
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-5. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #3.



Differences of Crater Centers over Perspective Transformation

Desired Boresight Crater #4

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	3.13	2.98	3.18	5.16	1.44	4.24	0.28	0.09	1.92	3.27	2.04	9.29	2.93	8.19	1.36	1.52
2	0	0	1.44	0.94	8.15	0.93	7.13	2.91	2.51	0.28	2.68	0.61	12.1	2.65	10.8	3.13	3.18
3	0	0	0	0.23	8	1.53	7.14	3.25	3.06	0.79	1.3	0.55	12.2	1.24	11.2	4.13	4.22
4	0	0	0	0	7.51	1.71	6.75	3.17	3.22	0.38	1.71	0.01	12	1.91	11.2	3.91	3.89
5	0	0	0	0	0	5.98	1.23	4.62	4.86	5.83	4.99	5.68	3.47	4.02	3.57	4.94	5.02
6	0	0	0	0	0	0	5.09	1.43	1.52	0.01	0.96	0.14	10.2	1.66	9.5	1.9	1.95
7	0	0	0	0	0	0	0	3.67	3.81	4.84	4.01	4.66	4.83	2.93	3.44	3.72	3.79
8	0	0	0	0	0	0	0	0	0.36	1.21	0.37	1.06	8.83	0.56	7.9	1.27	1.41
9	0	0	0	0	0	0	0	0	0	1.04	0.22	0.86	8.71	0.84	7.97	1.01	1.14
10	0	0	0	0	0	0	0	0	0	0	0.82	0.13	9.64	1.65	8.78	3.28	3.43
11	0	0	0	0	0	0	0	0	0	0	0	0.61	8.77	0.04	7.8	2.4	2.88
12	0	0	0	0	0	0	0	0	0	0	0	0	9.38	1.51	8.46	3.13	3.47
13	0	0	0	0	0	0	0	0	0	0	0	0	0	7.21	2.02	7.19	7.1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.95	0.21	0.52
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.77	5.58
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.13
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-6. Distances Between Crater Centers over the Perspective Transform for Boresight Crater #4.

The data is upper triangular because the distance between crater q and crater r is equal to the distance between crater r and crater q. The redundant data is eliminated from the table to prevent analysis of this data from being skewed.

In order to better visualize the information being presented in Table 5-3, Table 5-4, Table 5-5, and Table 5-6, these tables were plotted as contour plots. The rows and columns form the y and x axes, respectively, and the value in each cell of the tables (which is the difference between points over the perspective transform) is plotted as the contour level.

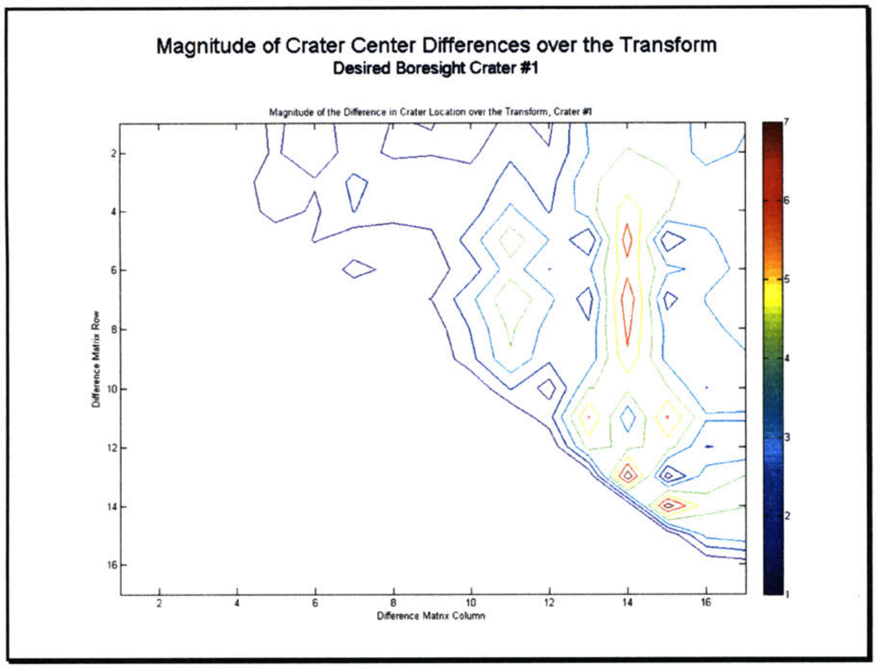


Figure 5.23. Characteristics of the Distances Between Crater Centers for Boresight Crater #1.

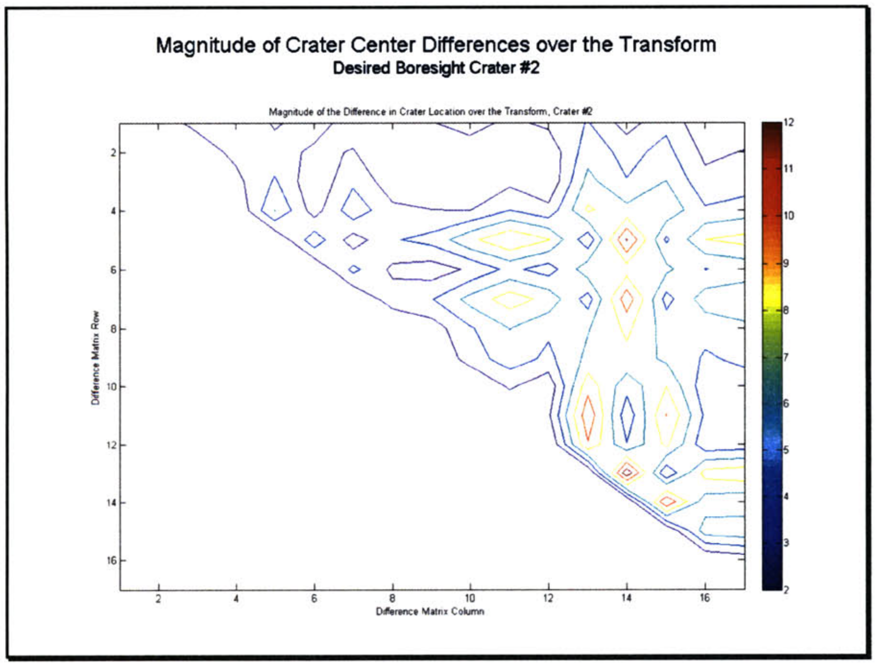


Figure 5.24. Characteristics of the Distances Between Crater Centers for Boresight Crater #2.

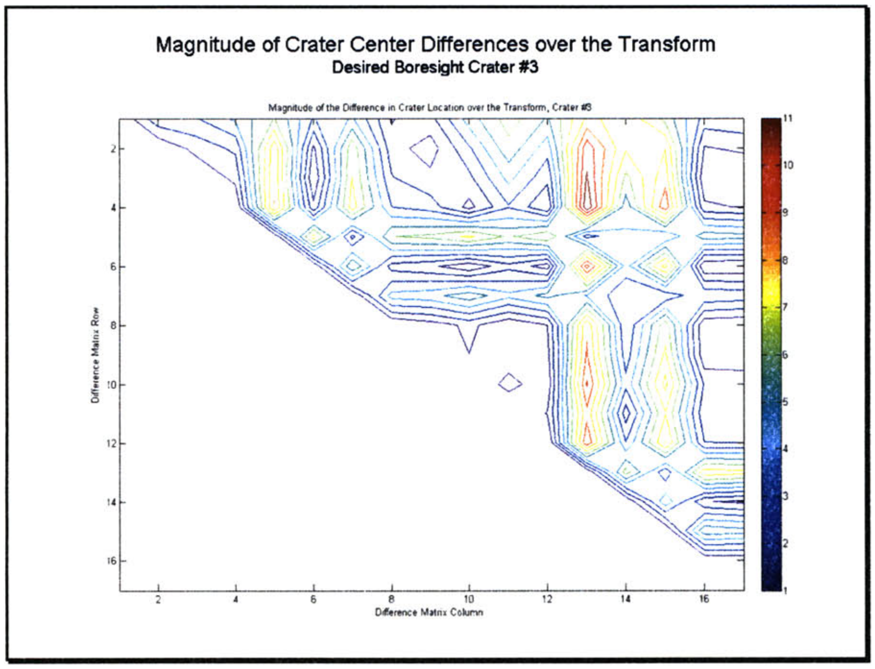


Figure 5.25. Characteristics of the Distances Between Crater Centers for Boresight Crater #3.

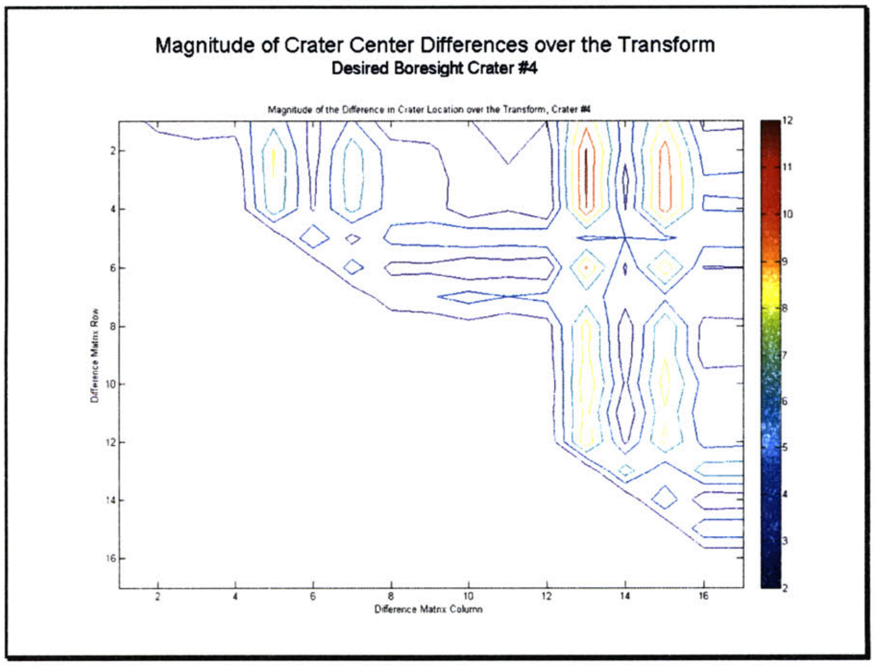


Figure 5.26. Characteristics of the Distances Between Crater Centers for Boresight Crater #4.

The contour plots highlight the similarity between the differences over the transform for the first and second, and the third and fourth potential boresight craters. (Compare Figure 5.23 and Figure 5.24, Figure 5.25 and Figure 5.26.) This is expected, as the original position of the first and second potential boresight craters is relatively near to one another. The same holds for the third and fourth potential boresight craters. (Refer to Figure 5.16). The craters positioned closest to the original principle point also have the lowest crater number. Therefore, the greatest differences in relative position over the perspective transform are expected in the upper right hand corner of the contour plot, between craters that are the farthest away from one another. These are also the highest and lowest crater numbers. The position of these craters will be influenced the most by the camera's perspective, and so they should demonstrate the greatest difference between their original position and their transformed position, in relation to all the craters around them. This general trend is noted in the contour plots above, with the greatest differences being concentrated toward the right half of the plot.

The data in Table 5-3, Table 5-4, Table 5-5, and Table 5-6 and Figure 5.23, Figure 5.24, Figure 5.25, and Figure 5.26 demonstrates that the effect of the perspective transform upon the data changes the entire relative arrangement of crater centers. Table 5-7 presents the minimum, maximum, mean, and standard deviation of the position differences over the perspective transform for all four potential boresight craters.

Table 5-7. Change in Position of Crater Centers Relative to All Other Craters Over the Transform.

	Mean Dist. (pixels)	**Minimum Dist. (pixels)**	**Maximum Dist. (pixels)**	**Std. Dev. (pixels)**
BS #1	2.29	0	7.78	1.75
BS #2	3.82	0	13.51	3.12
BS #3	3.48	0	11.86	3.16
BS #4	3.38	0	12.23	3.21
Average	**3.24**	**0**	**11.35**	**2.81**

The results presented in Table 5-7 are very similar to those presented in Table 5-2. This demonstrates that the change in the relative position of craters is essentially the same whether you are looking at the change of all craters in relation to the desired boresight or if you are looking at all craters and their change in position relative to all of the other neighboring craters.

5.7.5 Coding of the Perspective Transform Algorithm

The perspective transform is coded as a callable function. This function executes the transform upon a set of data with a given desired boresight crater. For each potential boresight crater, this function is invoked to shift the data so that the desired boresight is translated to the principle point of the transformed image. In making this shift, the transform accounts for the curvature of the lunar surface and how that will affect the appearance of the transformed image. This transform will effectively establish the required translational reference to allow for pattern vector generation which was discussed in section 4.1.

The practical inputs and outputs of the perspective transform function are similar to the inputs and outputs for the theoretical algorithm listed in section 5.2.2. The practical inputs, outputs, and constants are listed here.

Inputs:

1. Sensor image data points in the original image frame, $\{u_j \quad v_j\}$.
2. Desired i^{th} boresight crater.
3. Rotation matrix for i^{th} desired boresight, R_i.
4. Translation matrix for i^{th} desired boresight, T_i.
5. Point-wise depth for all j craters in the sensor image, d_j.
6. Spacecraft altitude, (alt).
7. Camera field-of-view (fov) and resolution (res).

Outputs:

1. Transformed image data points, $\{\tilde{m}_j \quad \tilde{n}_j\}$ for all j craters in the sensor image.

Constants:

1. Radius of the Moon, $r_m = 1737.4$ km.

The difference between the practical and theoretical inputs is that the practical perspective transform function requires the rotation matrix, the translation vector, and the point-wise depth as inputs. Although these quantities are technically part of the perspective transform, there is a significant reduction in processing speed that can be gained by calculating the rotation matrix, the translation vector, and the point-wise depth separately from the transform. By allowing these three calculations to occur separately from the transform, all the rotation matrices, translation vectors, and the point-wise depth for all potential boresight craters can be calculated at the same time. In other words, before the transform, an array of rotation matrices is formed, one for each potential boresight crater in a given sensor image. In addition, an array of translation vectors is also calculated, one for each potential boresight crater. Finally, the point-wise depth to each crater in the original sensor image is calculated.[1] These three quantities then become inputs into the transform algorithm. By calculating these quantities before the execution of the transform, the need to recalculate them for every iteration of the transform is bypassed. As a result, a significant amount of processing time can be saved in this manner.

The coding of the perspective transform algorithm followed the transform equation, Equation 5.27, very closely. After the transform is executed, the data units are converted from true image plane units $(u \quad v)$ to pixelated image plane units $(m \quad n)$. This unit conversion prepares the transformed data points for the remainder of the pattern vector generation process.

In general, the practical implementation of the perspective transform closely parallels the theoretical explanation contained in this chapter. The perspective transform was coded in Matlab version 7.1 revision 14.

[1] The point-wise depth does not depend upon the desired boresight crater at all. Therefore, the point-wise depth calculation yields a matrix rather than an array of matrices.

Chapter 6

The Main Processes of the Crater Matching Routine

The crater matching routine is a correlation method designed to match a sensor image with a crater database and determine the position of the craters in the sensor image on the lunar surface. To accomplish these goals, the crater matching routine has three distinct main processes;

1. The sensor image generation process.

   This process generates a 'fingerprint' for a given senor image based upon the pattern of craters in the original sensor image. This image signature will allow the sensor image to be quantified so that it can be matched to the database.

2. The process of crater database population.

   This process will create a database of lunar craters based upon the use of pattern vectors, effectively forming a 'reference map'. This database will be stored as a look-up table onboard the spacecraft to reduce the memory requirements.

3. The correlation process.

   The correlation process matches the image signature to the crater database and generates a position match. This position match will identify the location of each crater within the sensor image.

Each of these processes will be presented sequentially is sections 6.1, 6.2, and 6.3.

6.1 The Sensor Image Signature

Stated simply, the crater matching problem is to correlate an image of the lunar surface with a database of lunar craters. This matching will be accomplished by comparing the pattern of craters in the image with pre-determined patterns for lunar craters, which are stored in the database. Chapter 4 detailed a method to determine and record the pattern of nearby craters for a given boresight crater in the sensor image. The pattern vector that is determined by the process detailed in Chapter 4 contains all the information needed to appropriately correlate the boresight crater (for that pattern vector) to the database crater.

However, for most sensor images, there are several craters in the image that might be correlated to the lunar database. In other words, the pattern vector only correlates a single crater from the image to the database. If there are other craters in the image, the pattern of nearby craters can be developed for each of them, and these patterns can then be matched to the database independently. The result will be correlation between multiple craters from the original image and the lunar database. If enough of these craters correlate to a single area on the lunar surface, the location of which the original sensor image was taken can be determined.

The several pattern vectors generated for a single image will be combined into a single variable – the sensor image signature. The generation of the image signature corresponds to step 3 from the crater matching process as detailed in section 3.2.1. Each pattern vector contained in the signature correlates to a database crater, and the signature itself correlates to a location of the original image.

This section will analyze the process of determining the signature for a sensor image and will also continue the example problem from Chapter 4 to clarify the signature generation process.

6.1.1 Signature Generation Process

The process of generating an image signature is fairly simple, but relies heavily upon the pattern vector generation detailed in section Chapter 4. Figure 6.1 gives an overview of pattern vector generation. The reader is encouraged to review the overall process detailed in Chapter 4 if the process in the figure is unclear.

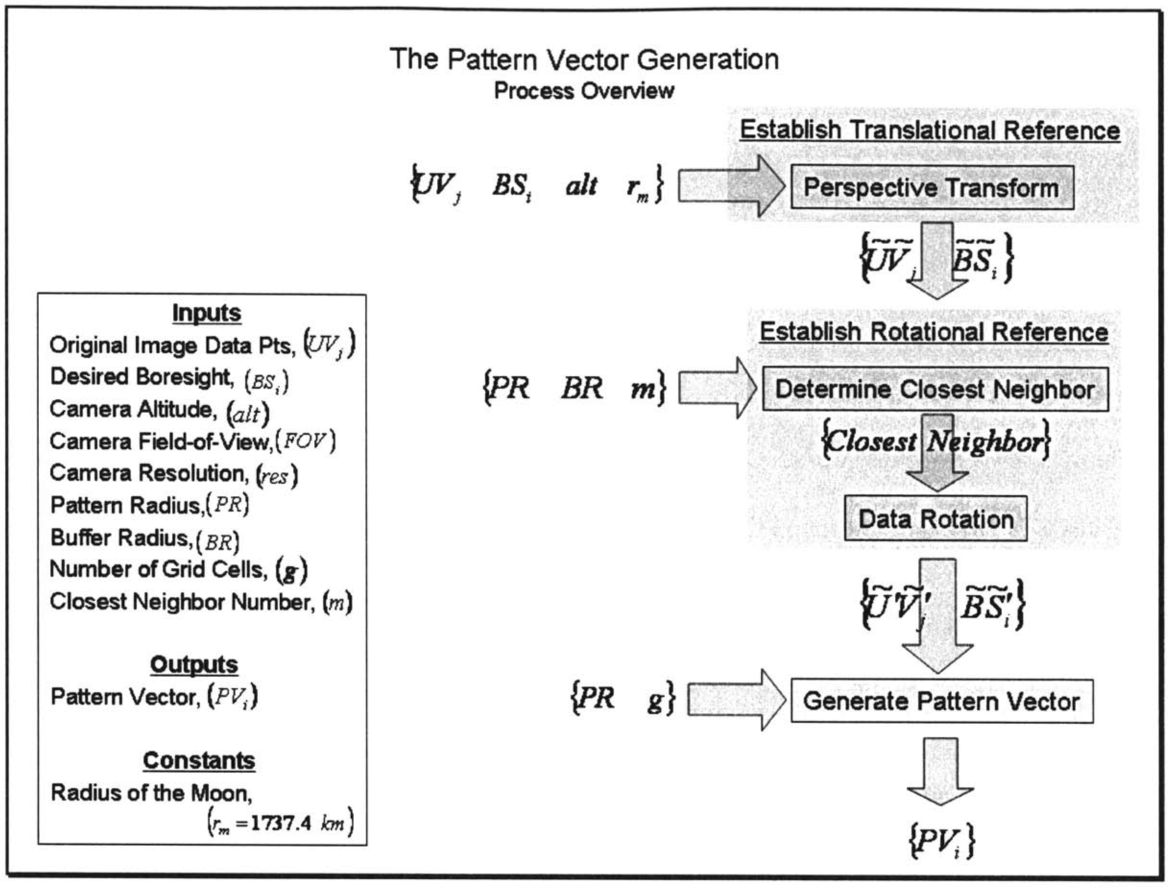

The Pattern Vector Generation
Process Overview
Establish Translational Reference
$\{UV_j \quad BS_i \quad alt \quad r_m\}$
Perspective Transform
$\{\widetilde{UV}_j \quad \widetilde{BS}_i\}$
Inputs
Original Image Data Pts, (UV_j)
Desired Boresight, (BS_i)
Camera Altitude, (alt)
Camera Field-of-View, (FOV)
Camera Resolution, (res)
Pattern Radius, (PR)
Buffer Radius, (BR)
Number of Grid Cells, (g)
Closest Neighbor Number, (m)
Outputs
Pattern Vector, (PV_i)
Constants
Radius of the Moon, $(r_m = 1737.4\ km)$
Establish Rotational Reference
$\{PR \quad BR \quad m\}$
Determine Closest Neighbor
$\{Closest\ Neighbor\}$
Data Rotation
$\{\widetilde{UV}_j' \quad \widetilde{BS}_i'\}$
$\{PR \quad g\}$
Generate Pattern Vector
$\{PV_i\}$

Figure 6.1[i]. Reprint Overview of the Pattern Vector Generation Process

The signature generation process can be broken down into three steps:

1. Pre-processing:
    a. Identify all potential boresight craters from the given sensor image.
    b. Calculate inputs for the perspective transformation.
2. Generate a pattern vector for each potential boresight.
3. Post-processing:
    a. Format each pattern vector for signature storage.
    b. Save each pattern vector to the image signature.

[i] Also Figure 4.2

The pre-processing of the sensor image can be broken down into two sub-steps, the identification of potential boresight craters and the perspective transform initialization operations. The identification of potential boresight craters simply identifies those craters within the given sensor image that are fit for pattern generation. In order to meet this requirement, a given crater must be more than a pattern radius distance from the edge of the sensor image. This will ensure that the entire pattern of nearby craters can be determined from the sensor image. That is, if a crater center is less than a pattern radius distance from the edge of the image, there may be nearby craters that are essential to the pattern that cannot be seen in the sensor image. In addition to this requirement, there must also be at least two other craters within the pattern radius for any given crater to be considered a potential boresight. This allows each potential boresight to have a crater as a closest neighboring crater and ensures there is at least one other crater to form the pattern.

The second sub-step of the pre-processing of the sensor image is the calculation of several inputs for the perspective transform. This will include the calculation of a rotation matrix and translation vector from the principle point of the original sensor image to each crater center. In addition, the point-wise distance from the camera location to the crater location will be calculated for each crater. These calculations are explained in more detail in Chapter 5.

After these pre-processing calculations, a pattern vector can be determined for each potential boresight by iteratively executing the process detailed in Chapter 4. The result of this process is a series of pattern vectors, each condensed as demonstrated in the example given in section 4.3.3.

Finally, the post-processing step converts the series of pattern vectors into an image signature. Before this pattern vector can be stored to the signature, the vector must be formatted to tag the vector with its boresight crater from the original sensor image. This will allow the pattern vector to be identified with the boresight crater around which it was formed. The format for this 'tagging' will be:

$$[Crater\_\# \quad BS_U \quad BS_V \quad (PV\_Cell \quad Crater\_Radius) \quad (PV\_Cell \quad Crater\_Radius)\ldots]$$

Equation 6.1

where $Crater\_\#$ refers to the list number of the potential boresights for the signature. The ordered pair $\langle BS_U \quad BS_V \rangle$ refers to the location of the given boresight crater in the original sensor image, given in the true image plane units (UV). The $(PV\_Cell \quad Crater\_Radius)\ldots$ ordered pairs are the components of the condensed pattern vector. (Reference section 4.3.3 and Equation 4.14.) In this format, each pattern vector can then be appended to the signature matrix. The signature matrix will therefore have i rows, where i is the number of potential boresight craters from the original image.[1]

Figure 6.2 presents these three steps, along with the included substeps, in the signature generation process. Signature generation encompasses the process of pattern vector generation.

The following section will present an example of signature formation based upon the example data utilized throughout section Chapter 4. This example will serve to clarify the details of the signature generation process.

[1] Unless a potential boresight is dismissed during the pattern vector generation process for being ill-suited for pattern generation. For each potential boresight that is dismissed, the number of rows in the image signature matrix will decrease by one.

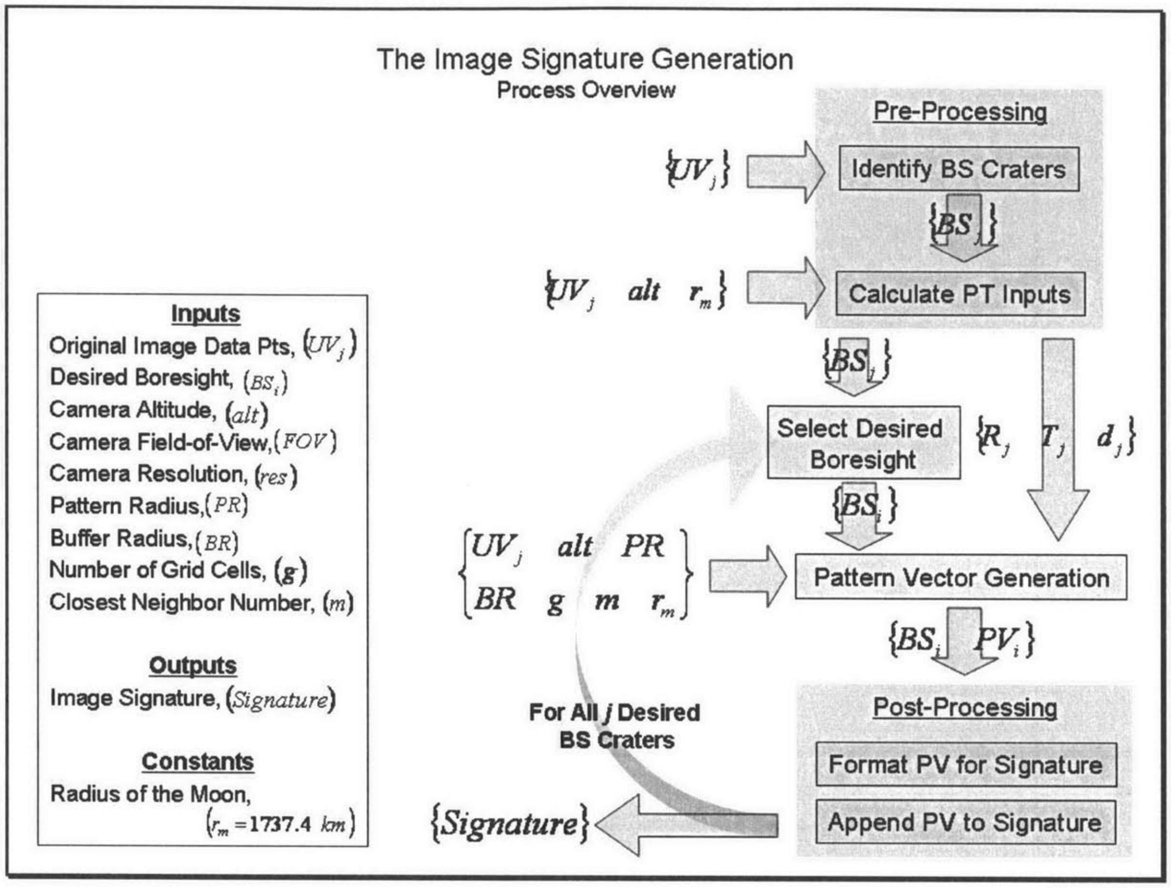

The Image Signature Generation
Process Overview
Pre-Processing
$\{UV_j\}$
Identify BS Craters
$\{BS_j\}$
$\{UV_j \quad alt \quad r_m\}$
Calculate PT Inputs
$\{BS_j\}$
Select Desired Boresight
$\{R_j \quad T_j \quad d_j\}$
$\{BS_i\}$
$\begin{Bmatrix} UV_j & alt & PR \\ BR & g & m & r_m \end{Bmatrix}$
Pattern Vector Generation
$\{BS_i \quad PV_i\}$
For All j Desired BS Craters
Post-Processing
Format PV for Signature
Append PV to Signature
$\{Signature\}$
Inputs
Original Image Data Pts, (UV_j)
Desired Boresight, (BS_i)
Camera Altitude, (alt)
Camera Field-of-View, (FOV)
Camera Resolution, (res)
Pattern Radius, (PR)
Buffer Radius, (BR)
Number of Grid Cells, (g)
Closest Neighbor Number, (m)
Outputs
Image Signature, $(Signature)$
Constants
Radius of the Moon,
$(r_m = 1737.4\ km)$

Figure 6.2. Overview of the Signature Generation Process.

6.1.2 Example Signature Formation

The example data used throughout section Chapter 4 will be continued here. The initial conditions for this example are listed in section 3.2.3.6. The example presented in section Chapter 4 detailed the pattern generation for a single boresight crater. This section will detail the generation of an image signature for the example data points which will include the generation of four separate pattern vectors.

Figure 6.3 shows the determination of the potential boresights from the original sensor image. The center of the potential boresight craters are highlighted in green. Only four of the seventeen craters in the original sensor image are suited as potential boresight craters. For this data set, this is due to the fact that most craters in the sensor image lie too close to the edge of the image. As a result, a full pattern radius will not fit within the bounds of the image, which makes these craters ill-suited as boresight craters.

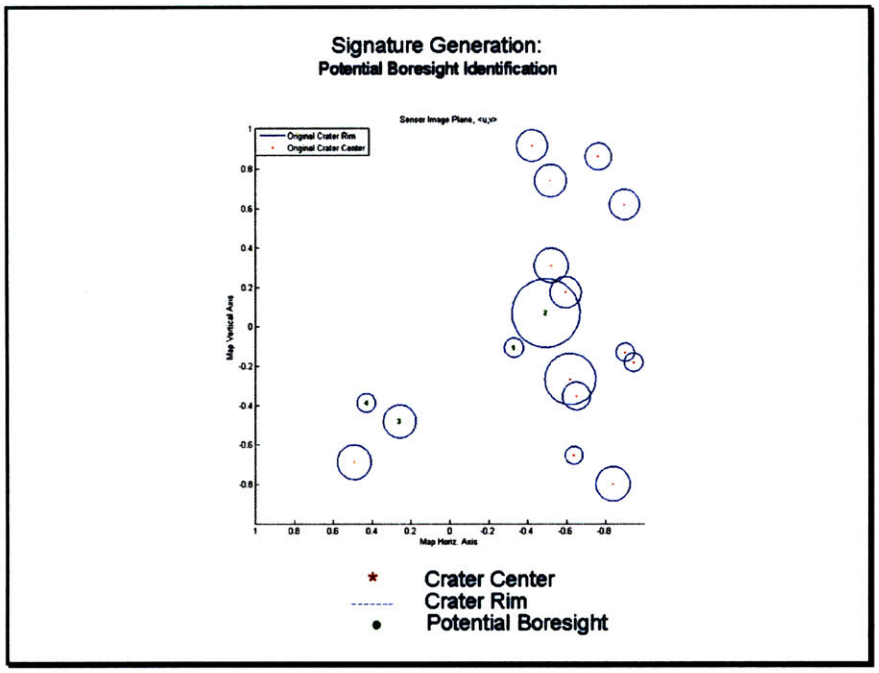


Figure 6.3[i]. Reprint Example Data Points with Potential Boresight Craters Highlighted.

Based upon these four potential boresight craters, four pattern vectors can be generated, iteratively selecting a boresight crater from the list of potential boresight craters. Figure 6.4, Figure 6.5, Figure 6.6, and Figure 6.7 demonstrate the generation of these pattern vectors. Equation 6.2, Equation 6.3, Equation 6.4, and Equation 6.5 give the pattern vectors for each of these cases, respectively. The process of generating these pattern vectors is detailed in section Chapter 4.

[i] Also Figure 5.16

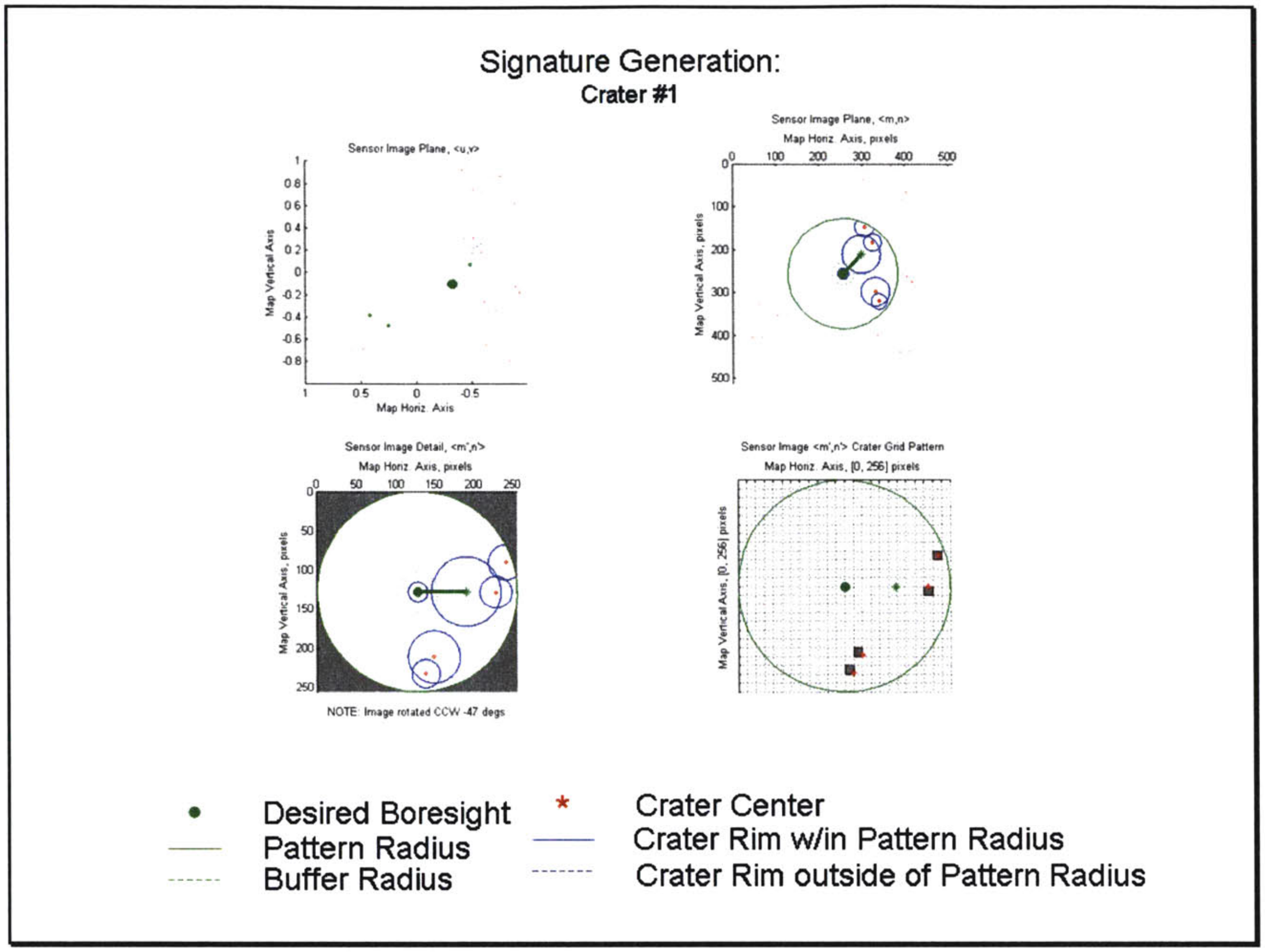


Figure 6.4. Signature Generation for Example Data Point Boresight #1.

$$Crater\ \#1:\ \langle 215 \quad 22.288 \quad 310 \quad 20.384 \quad 470 \quad 33.376 \quad 517 \quad 18.144 \rangle$$

Equation 6.2

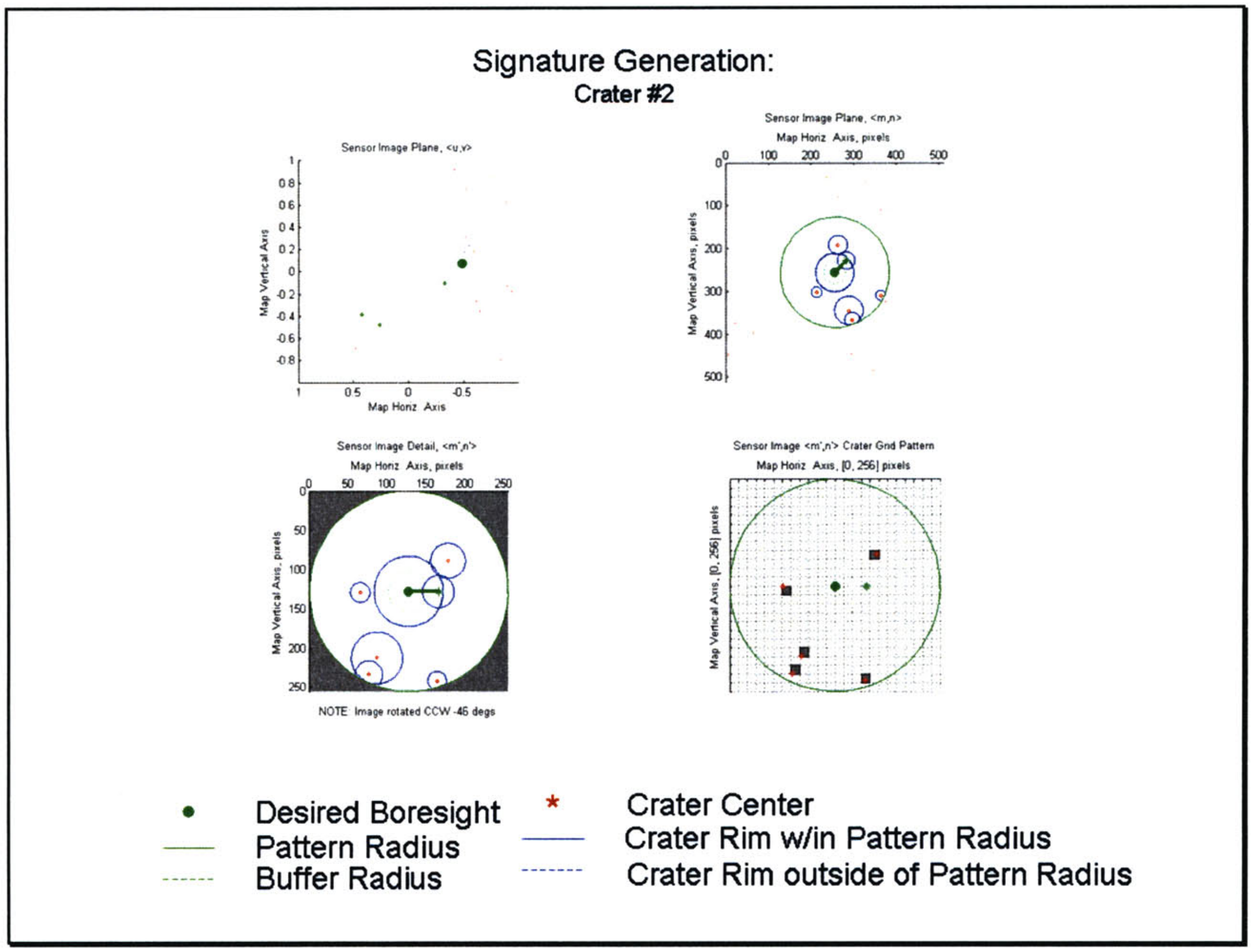


Figure 6.5. Signature Generation for Example Data Point Boresight #2.

$$Crater\ \#2:\ \langle 209 \quad 22.288 \quad 295 \quad 12.544 \quad 465 \quad 33.376 \quad 512 \quad 18.144 \quad 544 \quad 11.872 \rangle$$

Equation 6.3

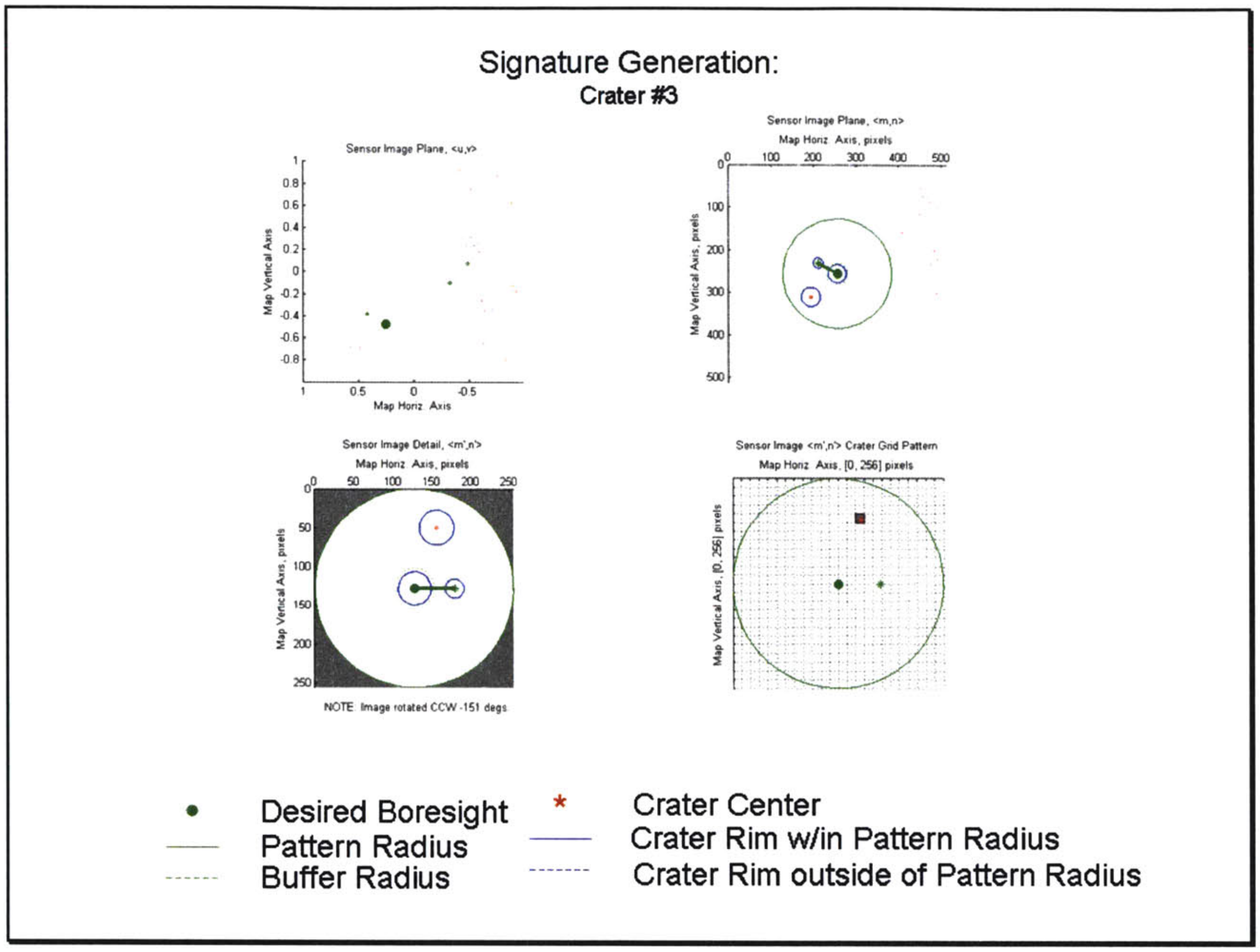


Figure 6.6. Signature Generation for Example Data Point Boresight #3.

$$Crater\ \#3:\ \langle 111 \quad 22.288 \rangle$$

Equation 6.4

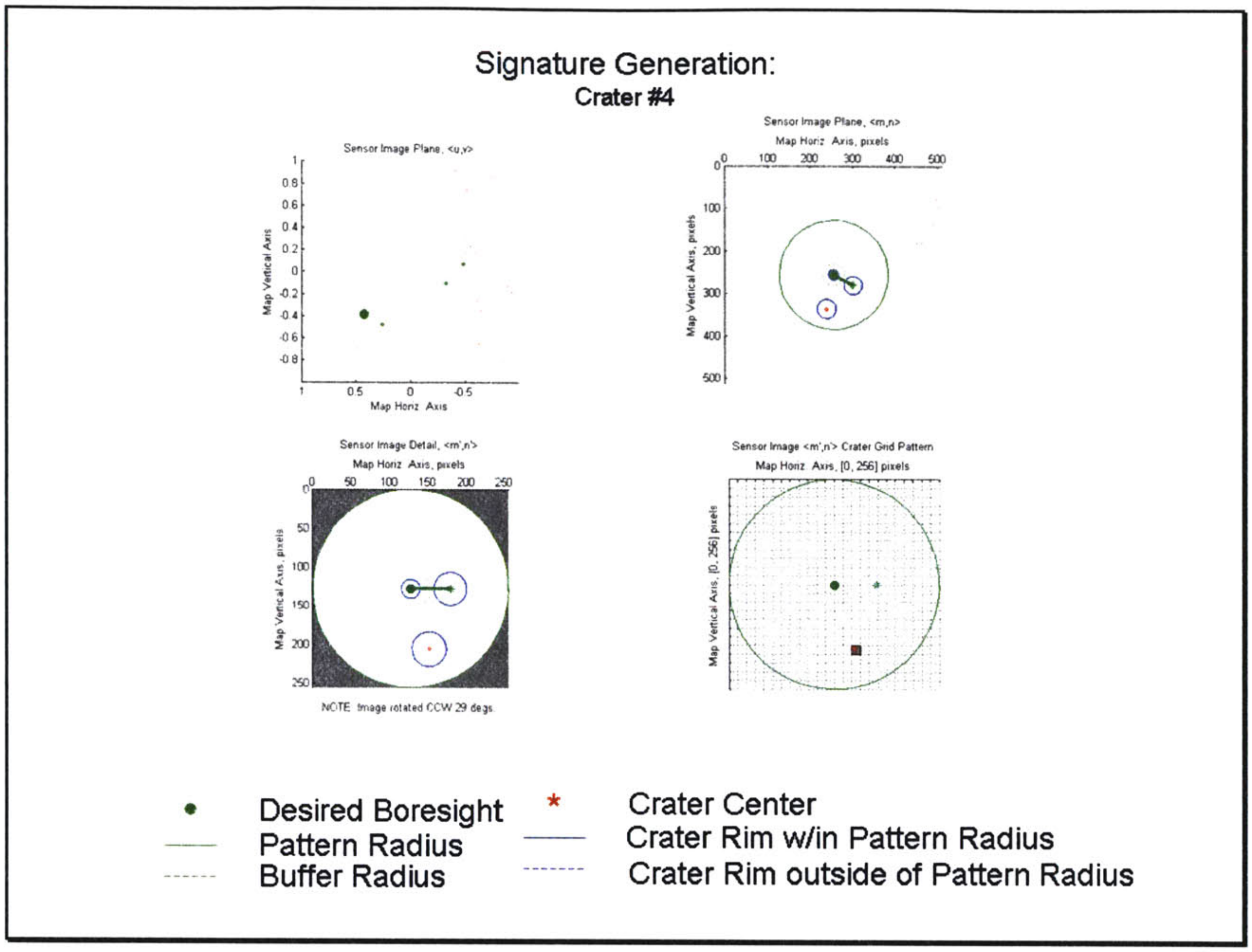


Figure 6.7. Signature Generation for Example Data Point Boresight #4.

$$\textit{Crater}\ \ \#4:\ \langle 471 \quad 22.288 \rangle$$

Equation 6.5

These four pattern vectors – in combination with each other – form the components to the image signature. However, in order to reference each pattern vector to its boresight crater, each vector must include the list number and location in the original image of the boresight crater. Then, each of these pattern vectors can be appended as a new row in the signature matrix. Equation 6.6 is the signature matrix for the image given in Figure 6.3.

$$\begin{bmatrix} 1 & -0.3281 & -0.1055 & 215 & 22.288 & 310 & 20.384 & 470 & 33.376 & 517 & 18.144 & 0 & 0 \\ 2 & -0.4922 & 0.0703 & 209 & 22.288 & 295 & 12.544 & 465 & 33.376 & 512 & 18.144 & 544 & 11.872 \\ 3 & 0.2578 & -0.4805 & 111 & 22.288 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0.4297 & -0.3867 & 471 & 22.288 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Equation 6.6

This example demonstrates the process for developing the image signature from individual pattern vectors. After being generated, the signature will be used to correlate the craters within an image to a location on the lunar surface.

6.2 The Crater Database

To this point, the discussion of the crater matching routine has focused upon the sensor image. However, the crater database is also a critical element in the correlation process. The crater database stores the pattern vectors for each crater on the lunar surface as well as the absolute position of these craters. By correlating the pattern vectors from a given sensor image to the database, the absolute location of the sensor image can be determined.

This section will examine the generation of the database and the format in which the database is stored.

6.2.1 Generation of the Database

The generation of the database is very similar to the generation of the image signature, which is described in section 0. The main difference between database generation and the signature determination lies in the method of obtaining original image. For signature determination, the image is given as an input from the sensors. However, for the database generation, the image must be developed using simulated or previously recorded lunar data.

For the author's implementation of the crater matching routine, the images necessary for the generation of a lunar database were based upon imagery of the lunar surface taken on the Deep Space Program Science Experiment[i] mission in 1994. The imagery from this mission allows a sensor image to be 'simulated' by selecting the data from the mission imagery that would be visible to a camera at a given altitude and position with a defined field-of-view.

The database generation can be accomplished in several steps.

1. Select a desired camera location.
2. Isolate the craters that lie within the field of view of the camera.
3. Generate the pattern vectors for all potential boresight craters in the image.
4. Store the pattern vectors for future correlation to sensor image craters.
5. Repeat steps 1 - 4, selecting a new camera location and storing the information to the same database.

For a complete database of craters on the lunar surface, the selected camera locations become crucial. The camera location must be specified such that a pattern vector is generated for every crater on the lunar surface. Therefore, it is important that the images overlap each other so that every crater is more than a pattern radius distance from the edge of an image for at least one camera location.

The database formed for analytic purposes does not cover the entire lunar surface. Rather, the database is limited to latitude bands. These bands are based upon where the available imagery of the lunar surface is of high enough quality to allow craters to be recognized from the image easily. Therefore, the database formation requires user-defined latitude and longitude bounds. These factors

i See the footnote for section 2.2 for further details.

define the outer extent of the database that is formed for these analyses. All the sensor images used in the analysis of the crater matching routine must fall within these bounds in order to correlate properly with the crater database. For these analyses, the latitude bands of

$$lat_b = \left\{ \begin{bmatrix} -80 & -50 \end{bmatrix}^\circ ,\ \begin{bmatrix} 50 & 80 \end{bmatrix}^\circ \right\}$$

Equation 6.7

will be used.

In addition, it becomes important to filter the pattern vectors generated for the database to ensure that if the same crater is selected as a boresight crater from two separate images, that the pattern vector is only stored once for that crater. This will ensure that the database does not include duplicate copies of the pattern vector for a single crater.

Figure 6.8 presents the process of generating the crater database. The database generation calls upon the process of determining an image signature, which is presented in section 0.

The database must be generated with the same initial conditions as those that will be used in the signature determination. Specifically, the field of view (fov), the camera altitude (alt), the pattern radius (PR), the buffer radius (BR), and the grid size (g) must be the same for both the signature and database generation processes. The mitigation of this requirement will be discussed in the author's recommendations for future work in Chapter 8.

The database pattern vectors could easily be stored in a format similar to the format of the signatures, simply as a matrix of pattern vectors. However, this format is prohibitive in terms of the final database size as well as in limiting the speed of correlation. Instead, the database will be stored as a look-up table referenced by the column number of the uncondensed pattern vector. This storage format of the database is markedly different than that of the signature. It will be presented in the following section, as well as a more detailed explanation for the utility of this format.

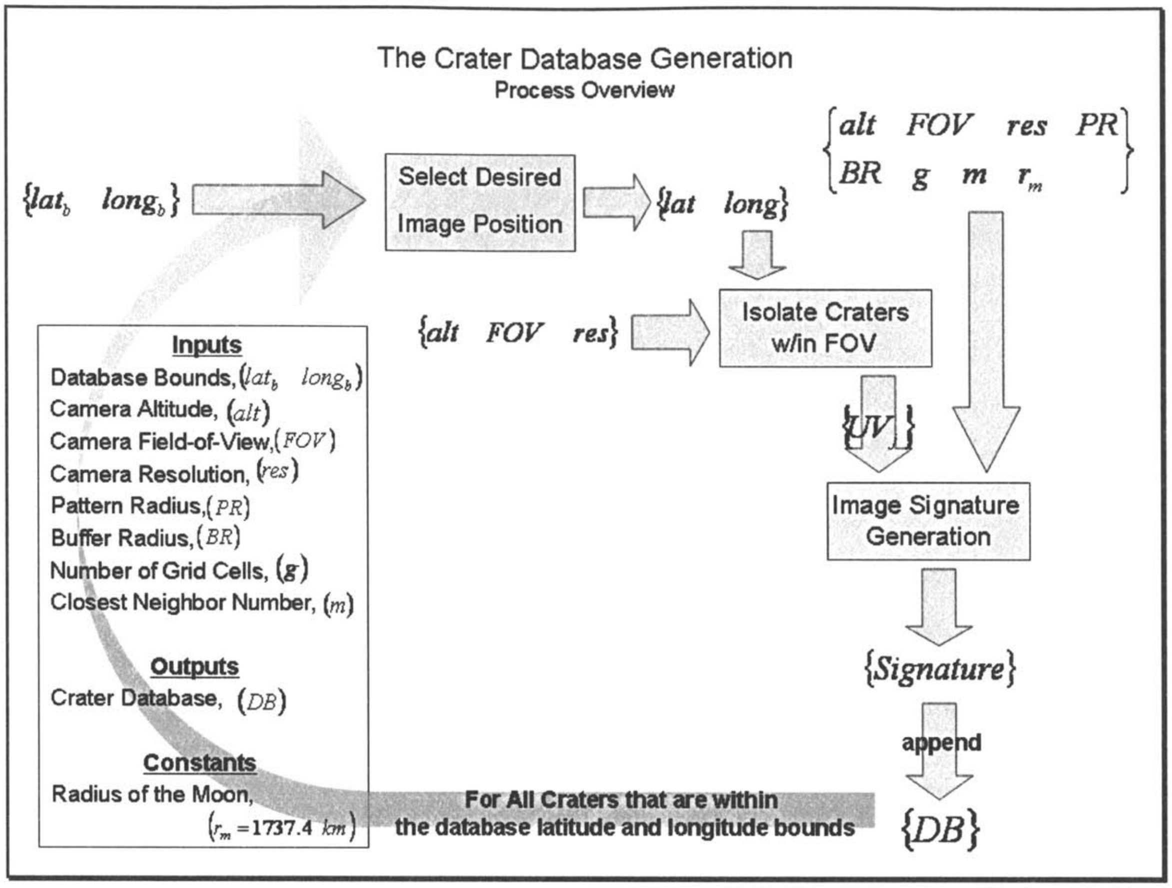

The Crater Database Generation
Process Overview
$\{lat_b \quad long_b\}$
Select Desired Image Position
$\{lat \quad long\}$
$\begin{Bmatrix} alt & FOV & res & PR \\ BR & g & m & r_m \end{Bmatrix}$
$\{alt \quad FOV \quad res\}$
Isolate Craters w/in FOV
$\{UV\}$
Image Signature Generation
$\{Signature\}$
append
$\{DB\}$
Inputs
Database Bounds, $(lat_b \quad long_b)$
Camera Altitude, (alt)
Camera Field-of-View, (FOV)
Camera Resolution, (res)
Pattern Radius, (PR)
Buffer Radius, (BR)
Number of Grid Cells, (g)
Closest Neighbor Number, (m)
Outputs
Crater Database, (DB)
Constants
Radius of the Moon, $(r_m = 1737.4\ km)$
For All Craters that are within the database latitude and longitude bounds

Figure 6.8. Overview of the Database Generation Process.

6.2.2 The Look-Up Table

The lunar database can be efficiently stored as a look-up table without sacrificing any data contained within the pattern vector. This section will examine the process for storing the data as a look-up table, the advantages and disadvantages of the look-up table, and some sparsity considerations that allow the size of the database to be reduced significantly.

6.2.2.1 Storing the Data

The process of storing the database as a look-up table begins with the generation of the first database pattern vector. This condensed pattern vector will indicate that one or more pattern vector cells are 'full'. For example, perhaps the first condensed pattern vector is

$$\langle 4 \quad 12.65 \quad 19 \quad 2.34 \quad 32 \quad 5.67 \quad 63 \quad 8.33 \rangle$$

Equation 6.8

This condensed pattern vector indicates that cells $\{4 \quad 19 \quad 32 \quad 63\}$ are 'full'. For each 'full' cell, a unique crater identifier and the crater radius will be appended to the end of the row that corresponds to the column number of the uncondensed pattern vector cell that is 'full'. Assuming that the pattern vector in Equation 6.8 is the first pattern vector to be formed in the process of database generation, the look-up table will be

$$\begin{bmatrix} 1 & \cdots & \cdots \\ \vdots & & \\ 4 & 1.01 & 12.65 \\ \vdots & & \\ 19 & 1.01 & 2.34 \\ \vdots & & \\ 32 & 1.01 & 5.67 \\ \vdots & & \\ 63 & 1.01 & 8.33 \\ \vdots & & \\ g^2 & \cdots & \cdots \end{bmatrix}$$

Equation 6.9

Equation 6.9 assumes that the unique crater identifier is 1.01 and the equation only shows the rows in the look-up table that have entries. If the next database pattern vector for this example is

$$\langle 5 \quad 3.27 \quad 32 \quad 5.09 \rangle$$

Equation 6.10

and the unique crater identifier for this pattern vector is 1.02, then the look-up table will be

$$\begin{bmatrix} 1 & \cdots & \cdots & \cdots & \cdots \\ \vdots & & & & \\ 4 & 1.01 & 12.65 & & \\ 5 & 1.02 & 3.27 & & \\ \vdots & & & & \\ 19 & 1.01 & 2.34 & & \\ \vdots & & & & \\ 32 & 1.01 & 5.67 & 1.02 & 5.09 \\ \vdots & & & & \\ 63 & 1.01 & 8.33 & & \\ \vdots & & & & \\ g^2 & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Equation 6.11

In similar fashion, as each database pattern vector is formed, the unique crater identifier and the crater radius will be appended to the end of the row indicated by the pattern vector elements. In general, the look-up table will have g^2 rows with a series of crater identifiers and crater radii in each row.

In addition to the look-up table, another reference table is necessary. This table will contain a list of the unique crater identifiers and the absolute location in lunar latitude and longitude that corresponds to each crater. This table will allow the correlation of sensor image craters with the database to determine the latitude and longitude of each crater in the sensor image.

6.2.2.2 Advantages & Disadvantages

The database will be stored as a look-up table for several reasons. The two most significant considerations – processing speed and file size – were already mentioned. The look-up table has distinct advantages in both of these areas over other storage methods. Due to the format of the look-up table, the pattern vector is not stored as an entire vector. Instead, the look-up table contains a list of all pattern vectors that have a ‘full’ grid matrix cell for the first cell, for the second cell, and so on for every grid matrix cell. By referencing the look-up table to the grid matrix cells, the number of rows in the look-up table is limited to the number of grid matrix cells. As a result of the storage of the database as a look-up table, the file size of the database is greatly reduced.

In addition, the look-up table format allows the correlation to be accomplished very quickly. By simply looking to see which craters have 'full' grid cells in all of the cells that the signature pattern vectors have 'full' cells, a correlation can be established very quickly. If the database pattern vectors were stored simply as a vector rather than a look-up table, each correlation would require a comparison between a signature pattern vector and all the database pattern vectors.

The main disadvantage of the look-up table format for storage of the database is that the pattern vectors are no longer visible as pattern vectors. In order to derive the pattern vector for a given crater from the database, each row of the database must be searched to determine if that grid cell was 'full' for the given crater. However, this disadvantage is relatively inconsequential, as the database pattern vectors are not needed as vectors by the crater matching routine.

These reasons merit the use of the look-up table as the preferred database format.

6.2.2.3 Utilizing Sparsity

Section 4.3.3 considers the sparsity of the pattern vector and uses the condensed pattern vector format to reduce the size of the image signature. Since the database is stored as a look-up table, using the condensed pattern vector format during the crater matching routine has no effect on the database size. However, there are several means of compressing the size of the database.

Consider the grid overlaid upon the example crater pattern from Plot B of Figure 4.15, which is the basis for Figure 6.9 below.

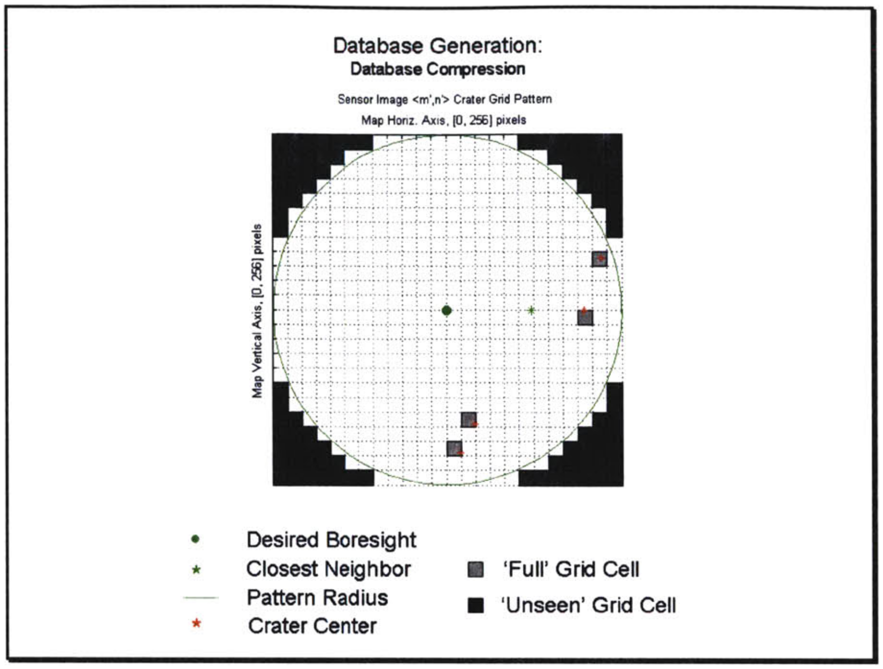


Figure 6.9. Database Compression Using the Circular Nature of the Pattern in the Grid Matrix.

The grid cells that lie outside of the pattern radius but still within the bounds of the sensor image have been shaded in Figure 6.9. For a 24×24 grid, these cells will always be outside of the pattern radius.[i] As a result, there will never be any crater centers that lie within these cells that will be included in the pattern vector. In other words, these grid cells will always be 'unseen' due to the size of the pattern radius. When the look-up table is being formed, the rows that correspond to these shaded elements will never have any entries because they are always outside of the pattern radius.

If these shaded grid cells are ignored in the formation of a pattern vector, a significant amount of memory saving may be realized in the look-up table. If the 'unseen' cells are ignored, the rows that are concatenated to form the uncondensed pattern vector by Equation 4.9 will be of varying lengths, and the rows themselves will not include any of the shaded elements from the grid matrix. In this case, the first grid matrix row to be added to the uncondensed pattern vector will consist of 10

i In general, approximately 21.5% of the area enclosed by a square of side $2 \cdot r$ is not enclosed in an inscribed circle of radius r.

elements. The second row to be added will consist of 14 elements, the third row 16 elements, and so on. (Refer to Figure 6.9.) If the shaded cells are ignored in the formation of the pattern vector from a 24×24 grid matrix, a total of 92 elements of the pattern vector will be eliminated without losing any pattern information. As a result, 92 rows of the look-up table will be eliminated as well, which is approximately a 16% reduction in the number of rows in the look-up table. This reduction in the number of rows included in the look-up table may cause a significant reduction in the file size of the database.

The pattern vector formed using this compression technique will be referred to as a 'circular condensed pattern vector'. For the remainder of this paper, a reference to a pattern vector will assume that the pattern vector is a circular condensed pattern vector.

In forming a pattern vector, the closest neighboring crater has been excluded from the pattern all along. Although the closest neighbor could be of benefit in the pattern formed for each boresight crater, the closest neighboring crater always is rotated to the positive x-axis to the right of the boresight (see section 4.2). As a result, if the closest neighbor is included in the pattern vector, the look-up columns that correspond to the grid cells to the right of the principle point will artificially have significantly more entries than other rows in the look-up table. This will increase overall file size of the look-up table by increasing the number of columns in the look-up table. Therefore, the author has chosen to remove to the closest neighboring crater (and the boresight crater) from pattern vector formation.

6.2.3 Database Characteristics

In the generation of the database, the author noted that the grid cells immediately surrounding the center point of the grid have a much higher frequency of being 'full' than other grid cells. As a result, the rows of the look-up table that correspond to these four grid cells are much longer than the rest of the rows in the look-up table. For a 24×24 grid matrix, the four grid cells that would have a higher occurrence of being 'full' would be the $(row \quad column)$ pairs $(12 \quad 12)$, $(12 \quad 13)$, $(13 \quad 12)$, and $(13 \quad 13)$. For the database generated by the author, the average row in these four locations was more than 70% greater in length than the average size of any of the other rows in the look-up table.[i]

This increase in length is most likely due to the fact that craters are often found clustered together on the lunar surface. As a result, the selection of one of these clustered craters as a boresight heightens the likelihood of craters being 'seen' in the cells surrounding the boresight.

Figure 6.10 shows the length of each row of the look-up table. For this plot, each row of the look-up table was correlated to the original grid matrix cell to which it corresponds, and the length of the row was plotted on the vertical axis.

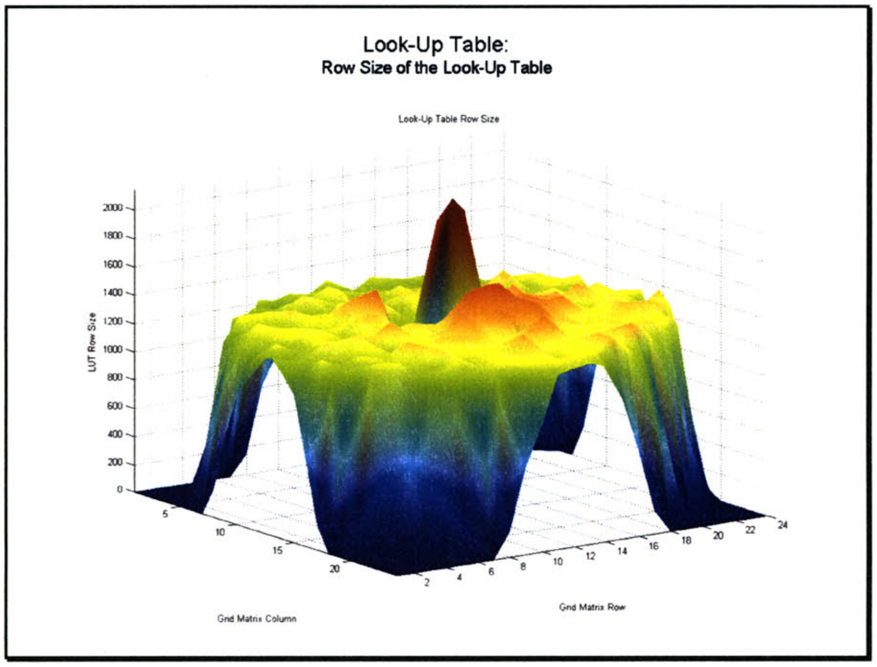


Figure 6.10. Row Size Analysis of the Look-Up Table.

[i] The average size of these four grid cells was 2094.5 elements, while the average size of the other rows was 1225.8 elements.

The center grid cells are drastically higher than the other cells. This effect may also add significantly to the size of the look-up table. There are many possible ways of counteracting this effect – for example, the four central grid cells could easily be ignored (although this would result in ignoring valuable pattern data). However, the author has chosen not to counteract this effect, as initial results indicate that removing these longer rows does not have enough effect on the size of the look-up table to warrant the loss of pattern data.

6.3 The Correlation Algorithm

This chapter has covered several topics in the development of the crater matching routine, including the determination of a pattern vector, the concept of an image signature, and the generation of a database of lunar crater patterns. The final essential topic in the crater matching routine is the algorithm which allows correlation between an image signature and the crater database.

The task of the correlation algorithm is to compare the pattern vectors in an image signature from the sensors with the database of lunar crater patterns, and to determine what area of the lunar surface appears in the original sensor image. Although the objective of the correlation algorithm seems rather mundane, the actual implementation of the correlation is complicated by the size of the lunar crater database.

The correlation technique implemented for the purposes of the crater matching routine is relatively simple. However, it does allow for successful correlation between the image signature and the crater database. This process is shown in Figure 6.11.

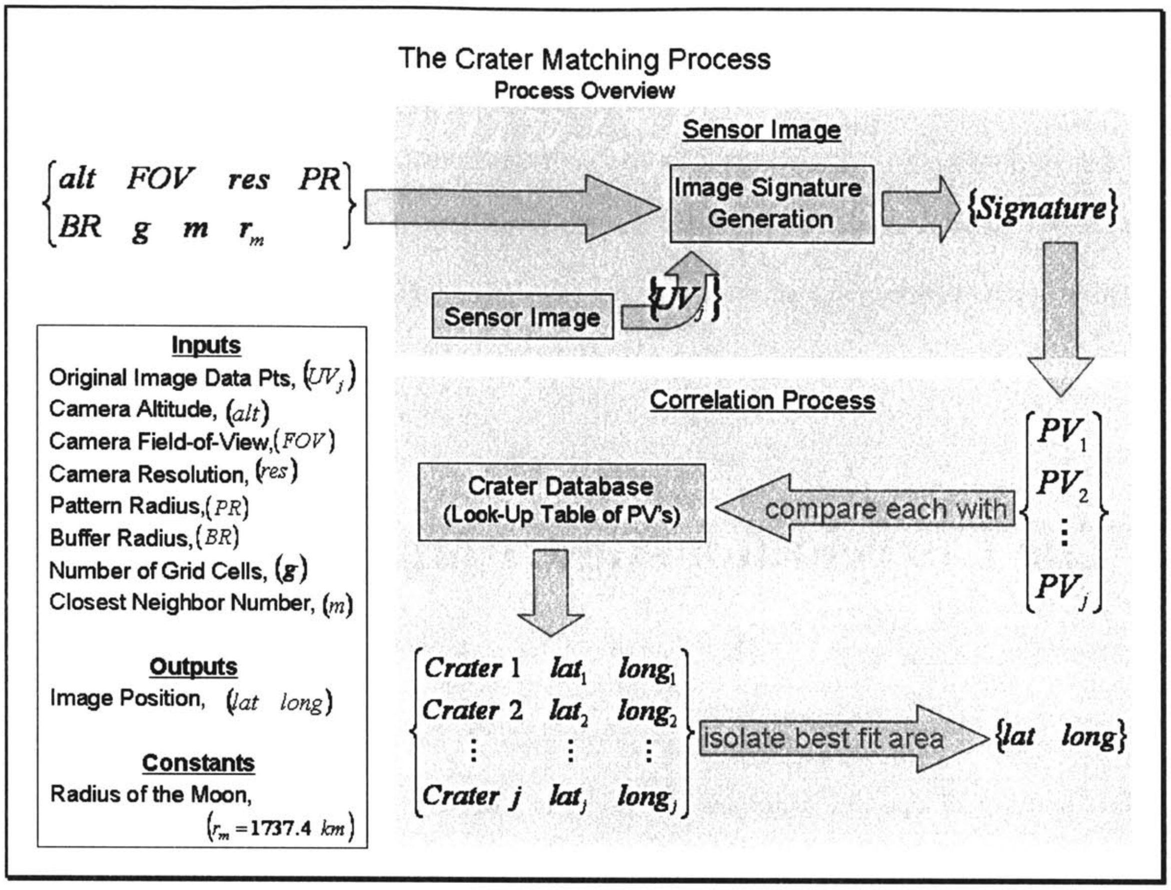


Figure 6.11. Overview of the Crater Matching Process.

Figure 6.11 includes the generation of an image signature for a given sensor image, which precedes any correlation efforts. The correlation task is relatively simple. The algorithm presented here is functional; however, the author realizes that significant benefit might be realized with a more intricate correlation process. Several recommendations for improvements to this correlation algorithm – or the implementation of an entirely different correlation algorithm – will be made in the recommendation for future work, section 8.3.

The correlation algorithm assumes the input of an image signature and the lunar crater database. For a given image signature, assume that the k^{th} pattern vector from the signature indicates that the l^{th} circular condensed grid cell is 'full'. The correlation algorithm will compare the radius associated with the l^{th} circular condensed grid cell to all the radii in the l^{th} row of the look-up table[1]. If any of the radii in the l^{th} row of the look-up table match the radius of the l^{th} circular condensed grid cell within a user-specified error margin (ε_1), the crater identifier for the crater with that radius is recorded from the look-up table as a potential match to the k^{th} pattern vector. This process is repeated for all the 'full' grid cells of the k^{th} pattern vector.

Once all the 'full' grid cells for the k^{th} pattern vector have been compared to the crater database in this way, there will usually be many crater identifiers that were recorded as potential matches. The crater identifiers for these potential matches are then sorted to determine the potential match that appears with the highest frequency. Consider only the most frequent potential match. This potential match must meet two requirements to be considered a match between the crater and the pattern vector. First, the frequency associated with this potential match must be above a second user-defined threshold (ε_2) relative to the total number of pattern vectors in the signature. Secondly, the ratio of the frequency associated with this potential match to the frequency of the second most frequent potential match must be greater than a third user-defined threshold (ε_3). If these two conditions are met, the potential match is recorded as the crater that correlates to the k^{th} pattern vector. In this manner, a pattern vector is correlated to a single crater from the lunar crater database.

This process for matching a pattern vector to a crater from the lunar database is repeated for all m pattern vectors contained in the image signature. The result will be m craters that correlate to the pattern vectors in the image signature. Ideally, all m of these craters will lie in a tightly confined area of the lunar surface, which will be the same as the area that appears in the original sensor image. In practice, however, the correlation process is not perfect. Therefore, a third user-defined threshold (ε_4) defines the percentage of the m craters that must lie within an area that is the size of the field of

[1] Which corresponds to the l^{th} circular condensed grid cell

view of the camera for the correlation to be successful. If the correlation is successful, the latitude and longitude of the craters that lie within this area are recorded as the actual location of these craters in the original image.

In this correlation process, there are three user-defined error bounds. These are

1. ε_1 - the amount of error acceptable in the radius match between the signature pattern vector cell and the look-up table.
2. ε_2 - the ratio that defines how often a given crater identifier must appear in the overall matching of the signature to the look-up table relative to the total number of pattern vectors. This threshold requires that the correlation between a signature and the database is strong enough to base a position match upon. The strength of correlation is measured by the number of component pattern vectors that match the database and determine the same image location.
3. ε_3 - the minimum ratio between the most frequent and the second most frequent craters for a given pattern vector. This threshold ensures that the match between the signature pattern vector and the database is clearly a definitive match in comparison to the second best match.
4. ε_4 - the ratio that defines how many correlated craters must lie within an area the size of the camera's field of view in order to consider the correlation of the signature to the lunar surface a success. This threshold essentially requires the correlation between pattern vectors in the image signature and those in the database to be grouped closely together on the lunar surface, or else the position match is inconclusive.

These error bounds are set by the user. The higher these bounds are set, the more difficult it is to generate a successful match between a given image and the lunar database. On the other hand, if the error bounds are set higher, the probability of an erroneous match is decreased.

These bounds are intended to prevent any significant number of false positions matches to occur by requiring the correlation to surpass these measures of the 'fit' of the correlation. Depending on how these correlation factors are set, any level of accuracy can be required of the crater matching routine before a position fix is generated. It is always preferable to have the crater matching routine fail to find a position match rather than to generate a false position match.[i]

After a successful correlation, the latitude and longitude of the craters within the original sensor image can be outputted to an algorithm that can calculate the pose of the spacecraft. These calculated values for the spacecraft pose can then be used to update the inertial guidance system of the spacecraft.[ii]

[i] This is because false matches can have a significant impact on generating navigational updates for the spacecrafts pose. Therefore, it is better to fail to match than to generate a false match.

[ii] The author developed the crater matching routine for this purpose at the Charles Stark Draper Laboratory in Cambridge MA. Draper Labs use the crater matching routine as just one component in a larger lunar landing simulation, which is comprised of a simulated lunar environment, an algorithm that identifies the craters from a sensor image, the crater matching routine, the pose estimation algorithm, and a Kalman filter to incorporate the pose updates.

Chapter 7

Implementation and Analysis of the Crater Matching Routine

In order to characterize the performance of the crater matching routine, it was coded and several error analyses performed. In this section, the coding of the crater matching routine will first be discussed. Then the results of several of the analyses performed will be presented below, as well as the inputs to initialize these analyses.

7.1 Coding of the Crater Matching Routine

Although the mathematics of the crater matching routine is rather rudimentary, the procedural nature of the routine is quite involved. The coding of the crater matching routine naturally inherits this procedural nature, which requires the execution of steps and substeps in a very specific order. In addition, since the signature formation components of the routine are required to be compatible with both the crater matching and the database generation processes. As a result, the coding of the matching routine is rather complex and merits attention here.

The coding of the crater matching routine can be broken into three major subroutines: the signature generation process, the formation of a crater database, and the crater match / correlation process. These subroutines are not independent of each other. As indicated in Figure 7.1, there is interaction between all of these processes. The interaction is indicated by the arrow s in the figure, which show the flow of the crater matching routine.

Sequentially, the first process to occur is the formation of the crater database.[1] This database formation occurs on the ground, before the spacecraft launches. The database(s) that are formed as a result of this process are then stored onboard the spacecraft for reference during the correlation process. The database formation iteratively calls the signature generation process, once for each simulated image of the lunar surface.

[1] In the form of look-up tables.

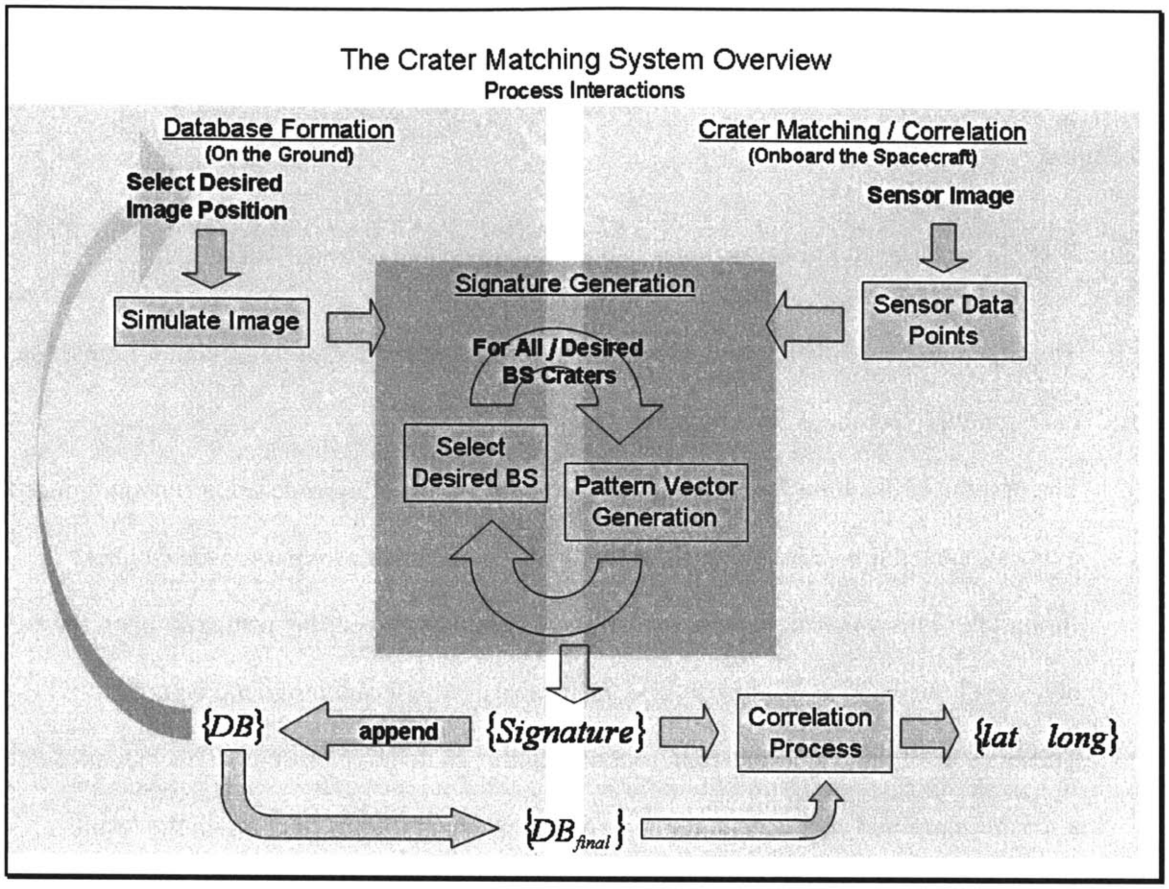


Figure 7.1[i]. Reprint Overview of the Interactions Between the Crater Matching Routine Processes.

The crater matching / correlation process occurs onboard the spacecraft in real-time. The crater matching / correlation subroutine calls the signature generation process as well, but just a single time. The signature that is passed back to the subroutine allows the correlation process to match the signature with the crater database that is stored in the onboard computer's memory.

The final significant subroutine is the signature generation process itself. The signature generation process entails two other important components; the pattern vector determination and the perspective transform. However, both of these components always fall within the context of signature generation, so they will not be considered on-par with the three major subroutines of the crater matching routine. The generation of the image signature is iterative as well, requiring the determination of a pattern

[i] Also Figure 3.2

vector for each potential boresight crater from the original image. The output of the signature generation process is the image signature, containing the pattern vector for every potential boresight in the image.

To clarify, there are three iterative processes in the crater matching routine. These are:

1. The signature generation process iteratively calls the pattern vector determination function, once for every potential boresight crater in a given image.
2. The process of database formation iteratively calls the signature generation function, once for every simulated image of the lunar surface that is determined as input for the database formation. This loop can include thousands of iterations, depending primarily upon the field-of-view of the camera, its altitude, and the database latitude and longitude bounds.
3. The analysis of the crater matching routine requires an iterative process. This process selects a sensor image and conducts an iteration of the matching routine to compile the results. These analyses are not technically a part of the crater matching routine; however, they are included as an iterative process because the analysis functionality is programmed into the routine itself. This iterative process is not indicated in Figure 7.1.

The author coded the crater matching routine as a series of callable functions. This allows the database formation subroutine and the crater matching / correlation subroutine to both invoke the same callable functions to perform common tasks when possible. For example, the signature generation subroutine is coded as a callable function. This allows both the database and crater matching subroutines to call the signature generation function, providing an image and obtaining an image signature from the function. The final crater matching routine consists of almost forty callable functions.

In addition, the formation of the crater database requires the simulation of an image of the lunar surface. This process was addressed briefly in section 6.2.1. The simulation of this image is accomplished by using data from the Deep Space Program Science Experiment to map the lunar surface. This data includes digital imagery of the surface. However, due to the duration of the mapping mission, the imagery was taken with a variety of sun angles relative to the lunar surface and the camera. Given a desired lunar surface area, a routine developed by the Charles Stark Draper Laboratory[i] in 2006-2007 allows a composite image to be formed and the craters within this image to be identified as craters. Further discussion of this routine is outside the scope of this paper. Suffice it to say that this routine effectively simulates the images required for database formation.

With these considerations taken into account, the remainder of the coding of the crater matching routine occurred in a fairly standard fashion. The coding of the routine used for the analyses of sections 7.2 and 7.3 does have room for improvement. Rather than presenting results based upon a final version of the code, the results presented in these sections are intended to demonstrate the functionality of the crater matching routine and to encourage further examination of and improvement upon the crater matching routine.

The crater matching routine was coded in Matlab version 7.1 revision 14.[ii]

[i] Draper Labs, 555 Technology Square, Cambridge MA 02139-3563

[ii] Matlab is an engineering language that facilitates simple matrix math and data processing. The language has a significant number of callable functions to perform standard operations.

7.2 Crater Matching Routine Analysis Initialization

The inputs, outputs, and constants for the analysis of the crater matching routine are listed below.

Initial Conditions:

1. Spacecraft altitude:	$alt = 100\ km$
2. Camera field-of-view:	$fov = 90^\circ$
3. Camera resolution:	$res = 512\ pixels$
4. Camera attitude:	$att = [0 \quad 0 \quad 0]^\circ$
5. Pattern Radius:	$PR = 65\ km$
6. Buffer Radius:	$BR = 6.5\ km$
7. Grid Number:	$g = 24$
8. Closest Neighbor Number:	$m = 1$
9. Latitude Bounds:	$lat_b = \left\{\ [-80 \quad -50]^\circ,\ \ [50 \quad 80]^\circ \right\}$
10. Longitude Bounds:	$long_b = \left\{\ [0 \quad 360]^\circ \right\}$

Inputs:

1. Sensor image data points in the original image frame.
2. Database of stored pattern vectors for lunar craters.

Algorithm Constants:

1. Radius of the Moon: $r_m = 1737.4\ km$

User-Defined Constants:

1. Correlation Factor 1: $\varepsilon_1 = 2\ pixels$
2. Correlation Factor 2: $\varepsilon_2 = 0.75$
3. Correlation Factor 4: $\varepsilon_3 = 0.75$
4. Correlation Factor 3: $\varepsilon_4 = 0.80$
5. Minimum Crater Radius Limit: $RadLim = 0.8\ km$

Outputs:

1. Latitude and longitude of the position match between the sensor image and the crater database.

The pattern radius and buffer radius in these initializations differ from the pattern and buffer radii in the example problem presented throughout this chapter thus far. These factors are defined by the user and indicate the amount of data to be included in the pattern vectors. The author chose to increase the pattern radius and slightly decrease the buffer radius in comparison to the example problem. The increased pattern radius will result in more data being included within each pattern vector, but fewer potential boresights for any given image.[1] The decreased buffer radius will result in a higher potential for rotational error resulting from the closest neighbor's position but might also allow pattern vectors to be formed for craters that have very few craters within the pattern radius.

[1] This results from the stipulation that any potential boresight must lie a distance of PR or greater from any edge of the image. A larger pattern radius indicates that fewer craters will meet this requirement.

Several other user-defined constants are included as well. The four correlation factors introduced and detailed in section 6.3 are assigned a value here. These factors define how precise the correlation between the sensor image and the crater database must be in order to consider the correlation an actual match. In addition, the minimum crater radius limit is defined. This initialization will allow the crater matching routine to disregard craters that have a very small radius.

Analyses are conducted for a spacecraft with a camera that is orbiting the Moon at an altitude of $100\ km$ with perfect attitude. There is a camera on this spacecraft that has a 90° field-of view and a resolution of $512\ pixels$. The Moon is considered a spherical body with a radius of $1734.7\ km$, which is a generally accepted value for the lunar radius.[24] This spacecraft will take a variety of pictures of the lunar surface in the latitude bands of $[-80 \quad -50]^\circ$ and $[50 \quad 80]^\circ$. Individually, the crater matching routine will attempt to correlate these pictures with the crater database that was formed of craters in the same area.[1] The correlation attempts will use a pattern radius of $65\ km$ and a buffer radius of $6.5\ km$. The grid overlay will have 24 rows and the same number of columns. Finally, the correlation factors are

$$\{(\varepsilon_1 = 2\ \ pixels) \quad (\varepsilon_2 = 0.75) \quad (\varepsilon_3 = 0.75) \quad (\varepsilon_4 = 0.8)\}$$

Equation 7.1

and the minimum radius that a crater must have to be considered in the correlation algorithm is $0.8\ km$.

These initial conditions represent a plausible situation encountered by a spacecraft attempting to use the lunar surface for a terrain relative position update of the spacecraft's inertial guidance systems.

[1] This spacecraft does not fly on a trajectory. Rather, the 'simulated spacecraft' parameters were used to generate representative sensor images that are randomly distributed between the latitude and longitude bounds.

Noise added to the sensor images to determine the crater matching routine's response to error will be discussed with each analysis below.

7.3 Crater Matching Routine Analysis Results

The analysis of the crater matching routine is conducted in four parts; an examination of the causes for the routine to not generate a positive position match, an analysis of the generation of positive position matches that are incorrect, a presentation of the processing speed of the routine, and an analysis of the crater matching routine's response to error in the sensor image.

Each of these analyses will use the initial conditions presented in section 7.2. Additional considerations will be present as necessary for each analysis.

For all of these cases, a successful position correlation occurs when the true latitude and longitude of the sensor image is matched to the latitude and longitude of the sensor image as determined by the crater matching process. The true latitude and longitude are known because the sensor input is simulated based upon a desired camera position, in latitude, longitude, and altitude.

7.3.1 Inconclusive Algorithm Results: Causes for No Match Possible

As a component of a system intended to update the inertial guidance system of a spacecraft, the fidelity of the crater matching routine must be very high. In other words, the position correlation produced by the crater matching routine must be generated with high confidence in the veracity of the match. To ensure a minimal frequency of incorrect position correlation, the correlation factors of section 7.2 were set to rather stringent levels for the analyses. These factors serve to ensure that the match between an image signature and the database is a very good match before allowing a position correlation is generated from the match.

Since these correlation factors effectively weed out weak matches between an image signature and the crater database, it is informative to consider the matches that are weeded out. For this analysis, a crater matching attempt will be considered an attempt to match a single sensor image with the crater database. There are several reasons that a crater matching attempt might fail. These are:

1. Inability to generate a signature from the sensor image.
    a. Because there were no craters within the pattern radius for all / some pattern vectors.
    b. Because there were too few craters within the pattern radius for all / some pattern vectors.
    c. Because there were no craters outside the buffer radius for all / some pattern vectors.
2. The attempt to match the signature to the database was inconclusive
    a. Because the ε_2 correlation threshold was not met for all / some pattern vectors. The ε_2 threshold requires that the correlation between a signature and the database is strong enough to base a position match upon.
    b. Because the ε_3 correlation threshold was not met for all / some pattern vectors. This threshold ensures that the match between the signature pattern vector and the database is clearly a definitive match in comparison to the second best match.
    c. Because the ε_4 correlation threshold was not met for the signature.
    d. Because no database entries even closely matched a pattern vector from the signature. This threshold essentially requires the correlation between pattern vectors in the image signature and those in the database to be grouped closely together on the lunar surface, or else the position match is inconclusive.

Although an image signature may have been generated, there is still the possibility that attempts to generate pattern vectors for potential boresight craters within the image were not successful. Reasons that the signature generation might have succeeded but a pattern vector generation failed are:

1. Pattern vector generation failed because there were no craters within the pattern radius.
2. Pattern vector generation failed because there were too few craters within the pattern radius.
3. Pattern vector generation failed because there were no craters outside the buffer radius.

Even though the matching attempt may have succeeded, there is the possibility that individual pattern vectors could not be matched to the crater database. The reasons that this might have occurred are fewer:

1. The pattern vector failed to correlate because the ε_2 correlation threshold was not met. The ε_2 threshold requires that the correlation between a signature and the database is strong enough to base a position match upon.
2. The pattern vector failed to correlate because the ε_3 correlation threshold was not met. The ε_3 threshold ensures that the match between the signature pattern vector and the database is clearly a definitive match in comparison to the second best match.

All of these possibilities result in an inconclusive crater matching attempt. However, all of these cases are recognizable to the routine, and the routine does not output a position correlation. Therefore, any of these cases will not result in a navigational update. Although the goal of the crater matching routine is to generate navigational updates, it is far better to have inconclusive crater matching routine iterations than to have position matches that are false. As a result, the correlation factors were set high enough to prevent as many false position matches as possible without generating too many inconclusive matches.

The data presented here is based upon 1000 iterations of the crater matching routine.[i] For every iteration, the routine attempted to correlate a sensor image to the crater database, and the results were recorded. No noise is added to the sensor image data points for this analysis.

Abbreviations that will be used in the tables and figures throughout the rest of the analysis section are listed in Table B-2 in Appendix B.

Table 7-1 presents several general statistics on the iterations for this analysis.

Table 7-1. General Success / Failure Statistics from the Analysis of the Routine.

# of iterations:	**1000**	
# of unsuitable images:	**151**	**15.1%**
# of correlation attempts:	**849**	**84.9%**
# of successful signature gen.:	825	97.17%[ii]
# of signature gen. failure:	24	2.83%[i]
# of correct position match:	742	87.4%[i]
# of incorrect position match:	0	0%
# of inconclusive position match:	83	9.78%[i]

From Table 7-1, the number of correlation attempts was less than the 1000 iterations for which the data analysis was initialized. For some of these 1000 iterations, there were either no craters within the image or no suitable boresight craters within the image. Therefore, the correlation attempt was terminated at that point. As a result, the number of correlation attempts is less than 1000. For all the following percentages in bold type for the remainder of this section, the percentage will be in reference to the number of correlation attempts (849).

The indentation of the left column in Table 7-1 is intended to indicate the hierarchy of the frequencies. Frequencies listed for the correct / incorrect / inconclusive position matches are the

[i] The data presented in sections 7.3.1, 7.3.2, and 7.3.3 are the result of the same set of 1000 iterations.
[ii] Percent relative to the total number of correlation attempts.

percentages relative to the number of correlation attempts (849). The same is true of the signature generation statistics from the same table. Although this hierarchy scheme does not affect the data in Table 7-1 significantly, it is pertinent to Table 7-2, Table 7-3, Table 7-4, and Table 7-5. For these same tables, all the numeric entries indicate if they are counting the number or signatures or pattern vectors for a given case. For those entries that are counting pattern vectors, the percentages were determined by tallying all counts at the specified hierarchy and determining the percentages relative to that total. For example, rows 2 thru 4 in Table 7-2 are all at the same hierarchy and are counting pattern vectors. Therefore, the percentages in these rows are determined by tallying column 2 for these rows (which yields 31 pattern vectors) and determining the percentages relative to this total $\left(\frac{3}{31} = 0.0968 \quad \frac{28}{31} = 0.9032 \quad \frac{0}{31} = 0.0\right)$.

Table 7-2 shows the list of reasons that the crater matching routine might fail to correlate to a position, and the frequency with which that scenario occurred.

Table 7-2. Reasons and Frequencies the Routine Failed to Generate a Position Match.

Inability to generate a signature from the sensor image:	**24 sigs.**	**2.83%**[1]
Because there were no craters w/in the PR for all / some PVs:	3 PVs	9.68%
Because there were too few craters w/in the PR for all / some PVs:	28 PVs	90.32%
Because there were no craters outside the BR for all / some PVs:	0 PVs	0%
The attempt to match the signature to the DB was inconclusive:	**83 sigs.**	**9.78%**[1]
Because the ε_2 or ε_3 correlation thresholds were not met for the signature:	83 sigs.	100%
Because the ε_2 correlation threshold was not met for all / some PVs:	0 PVs	0%
Because the ε_3 correlation threshold was not met for all / some PVs:	371 PVs	100%
Because the ε_4 correlation threshold was not met for the signature:	0 sigs.	0%
Because no DB entries even closely matched any PV from the signature:	0 sigs.	0%

Table 7-3 shows the reasons why a signature generation might succeed while one or more component pattern generation attempts failed, and the frequency with which each occurred.

[1] Percent relative to the total number of correlation attempts.

Table 7-3. Reasons and Frequencies the Routine Generated a Signature but a Component Patter Vector Generation Failed.

Signature gen. succeeded but at least one component PV gen. failed:	**8 PVs**	**0.94%** [i]
Pattern vector gen. failed because there were no craters within the PR:	0 PVs	0%
Pattern vector gen. failed because there were too few craters w/in the PR:	9 PVs	100%
Pattern vector gen. failed because there were no craters outside the BR:	0 PVs	0%

Table 7-4 shows the reasons why a matching attempt might succeed while at least one component pattern vector matching attempt failed, and the frequency with which each occurred.

Table 7-4. Reasons and Frequencies the Routine Successfully Determined a Position Match but a Component Patter Vector Matched Incorrectly.

Matching attempt success but at least one component PV matching failure:	**87 PVs**	**10.25%** [i]
The PV failed to correlate because the ε_2 correlation threshold was not met:	0 PVs	0%
The PV failed to correlate because the ε_3 correlation threshold was not met:	127 PVs	100%

From Table 7-1, 87.4% of all crater matching attempts successfully identified the correct position of the image. Given the stringency of the correlation factors – this is a definite success. Perhaps more promising is that there were no position matches that falsely identified the position of the sensor image. Given that there was no noise added for this analysis, this is not wholly unexpected. However, it does indicate that the pattern vectors are unique from one another for the given database.

In addition there are several other trends that merit attention. Table 7-2 lists that 100% of the failures to generate signatures result from images with too few craters within the pattern radius. This indicates that it may be appropriate to screen the sensor images for a minimum number of total craters before passing the images to the crater matching routine. These screening operations would save processing time for spacecraft computers and identify images that are likely to have very sparse pattern vectors anyway – which are ill-suited for correlation.

[i] Percent relative to the total number of correlation attempts.

The second noteworthy trend is that 100% of the failed correlation attempts were due to the ε_3 threshold not being met. Section 6.3 explains that ε_3 is the minimum ratio between the most frequent and the second most frequent craters for a given pattern vector. This indicates that the correlation attempts fail most frequently – by a huge percentage – because there are other pattern vectors in the crater database that match with the pattern vectors for the sensor image almost as well as the selected match.

Finally, there are also signatures that were successfully generated that had one or more component pattern vectors fail to generate. Likewise, there are successful correlation attempts that had one or more component pattern vector matching attempts that failed. The reasons for the component failures in both cases were too few craters within the pattern radius and failure to meet the ε_3 correlation threshold, respectively. Despite the failure of these individual components, the overall crater matching routine succeeded in generating a position correlation in each of these cases.

7.3.2 Failure of the Algorithm: Incorrect Position Correlation

Although the correlation factors can be modified to minimize false position matches, there is always the possibility that an image signature will be matched to the crater database and result in an incorrect position correlation. However, the possibility of this occurrence is mitigated by many factors.

1. The correlation factors are set to require a very good match between the database and the signature.
2. The signature usually contains multiple pattern vectors, which decreases the impact of a single pattern vector that is matched incorrectly.
3. Each pattern vector usually contains multiple grid cells which are 'filled' by the presence of other crater centers. As a result, pattern vectors for individual boresight craters are rarely similar to one another.

This list also draws attention to two tendencies that will increase the frequency of an incorrect position correlation.

1. If the signature contains only one – or a very few – pattern vectors, the likelihood of a false match increases.
2. If the pattern vector only contains one – or a very few – 'full' grid cells, it is much more likely that the pattern vector may be matched incorrectly to the database.

If the signature contains only one pattern vector, a false match between that vector and the database will result in a false position correlation. With fewer position vectors, it becomes easier for a false match between one vector and the database to sway the overall success of the position correlation. Likewise, if a position vector only has one 'full' grid cell, even a slight amount of error can cause a 100% correlation to the incorrect database entry. This is due to the fact that the error will only need to affect a single grid cell – and can thereby completely change the pattern vector.

This analysis uses a grid size of $g = 24$. Therefore, there are a total of $24^2 = 576$ grid cells and $2^{576} \approx 2.47 \cdot 10^{173}$ possible pattern vectors. In other words, there are almost a trillion raised to the quadrillion $\left(10^{12}\right)^{10^{15}}$ possible pattern vectors. That is far more than a googol of possibilities! However, with only one grid cell 'full' in a pattern vector – the number of possibilities is severely limited. The number of possibilities becomes the combination $C_1^{576} = 576$.[1] With two 'full grid cells, $C_2^{576} = 1.66 \cdot 10^5$. Due to the large number of pattern vector possibilities – even with only two 'full' grid cells, it is highly unlikely that image noise will change a pattern vector from that image into a pattern vector of another image. That is, with one exception. When there is only one 'full' grid

[1] $C_k^n = \dfrac{n!}{k!(n-k)!}$. It represents the number of groups of k components that can be selected from a data set of size n without the same element of n being able to be selected more than once for a given group.

cell, it is conceivable that image noise might affect this single 'full' cell enough to make the noisy pattern vector exactly – or closely – identical to some other pattern vector.

Both of the cases which increase the potential for false position correlation can be warded off by requiring more than one pattern vector in a given signature, and more than one 'full' grid cell in a given pattern vector. However, the author has chosen not to implement these filters to allow the effects of noise in the sensor image to be more apparent. These filters should be in place for an operational version of the crater matching routine.

The data presented in Table 7-5 is based upon 1000 iterations of the crater matching routine.[1] For every iteration, the routine attempted to correlate a sensor image to the crater database, and the results were recorded. No noise is added to the sensor image data points for this analysis.

Table 7-5. Correct and Incorrect Position Matches.

# of correlation attempts:	**849** sigs.	
# of incorrect position matches:	0 sigs.	0%
# of correct position matches with at least one false component PV match:	87 PVs	10.25%

Although it makes analysis somewhat duller, the author is pleased that there were no incorrect position matches for this analysis. As mentioned in the previous section, there were several times when the correlation of individual pattern vectors failed. These failures were exclusively the result of the ε_3 threshold not being met.

7.3.3 Processing Speed Analysis

The crater matching routine is designed to work in conjunction with several other software components to process sensor imagery and to generate a navigational update for the inertial guidance system of the spacecraft. Due to the relatively short amount of time that entry, descent, and landing requires, the process of modifying the spacecraft's trajectory to account for the navigational updates

[1] The data presented in sections 7.3.1, 7.3.2, and 7.3.3 is the result of the same set of 1000 iterations.

must occur close to real-time. Consequently, the crater matching routine must be able to generate results in several seconds in order for the results to be useful. The requirements listed in section 3.2.1.1 state that the routine must be able to generate a position match in three seconds.[i] This will allow enough time for ancillary program operations and still yield a navigational update in a reasonable amount of time.

The analyses detailed below are performed using the crater matching routine programmed in Matlab version 7.1 revision 14. The code was not optimized for processing speed and the computer language is not intended for practical implementation. In addition, the code has many additional analysis and graphing tools incorporated, which will not be present in the flight code. These additional features of the code are necessary for the initial analysis of the code, but they also slow the overall processing time of the routine. Despite these considerations, the analyses presented below still give some indication of the order of magnitude for the speed of the routine.

The processing speed data presented here was generated from 1000 iterations of the crater matching routine.[ii] For every iteration, the routine attempted to correlate a sensor image to the crater database, and the results were recorded. No noise is added to the sensor image data points for this analysis.

Figure 7.2 presents the processing time of the routine as a function of the number of potential boresight craters in the sensor image. The data distribution will also be fitted with a polynomial curve to more precisely characterize the relationship between processing time and the number of potential boresight craters in the sensor image.

[i] Time states as a benchmark performance time.

[ii] The data presented in sections 7.3.1, 7.3.2, and 7.3.3 is the result of the same set of 1000 iterations.

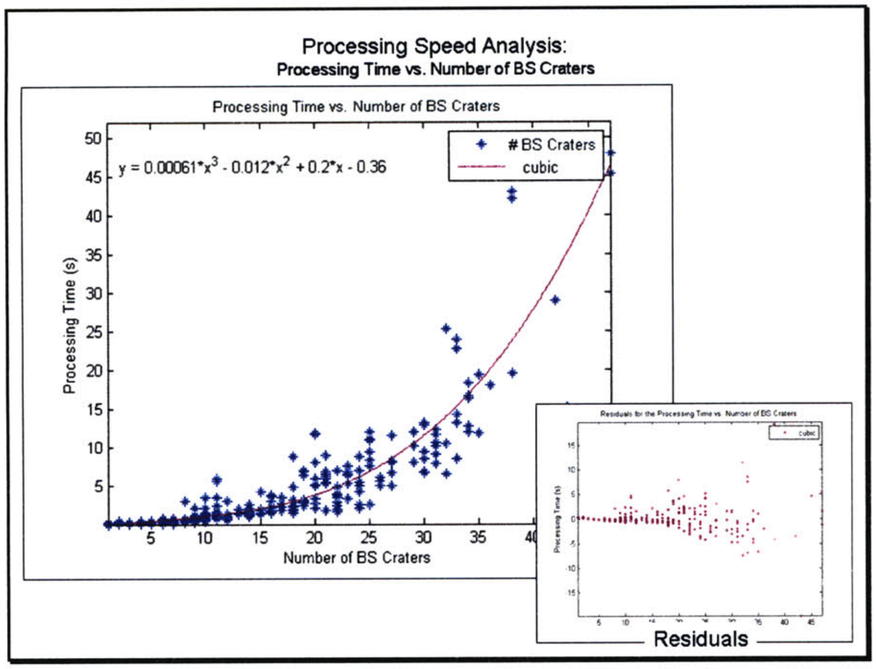


Figure 7.2. Processing Time vs. the Number of Potential Boresight Craters in the Sensor Image.

Figure 7.2 demonstrates that the processing time increases as a cubic in relation to the number of potential boresight craters in the sensor image. The best-fit cubic to match the data distribution is

$$y = 0.00061 * x^3 - 0.012 * x^2 + 0.2 * x - 0.36$$

Equation 7.2

For each desired boresight crater, a pattern vector must be generated and eventually correlated to the crater database. Therefore, it is expected that the processing time increases as the number of boresights increases. However, the processing time is not linear in relation to the number of boresights. In considering possible reasons for the cubic relationship between these quantities, the author returned to the code. The code for the routine includes many loop structures that are required to iterate over the sensor data. As a result, the author believes that the additional number of potential boresights causes a greater number of loop iterations for each boresight – thereby increasing the overall processing time at a greater-than-linear rate.

For the results presented above, the approximate processing time is presented for several representative numbers of potential boresight craters. This processing time was estimated by the cubic polynomial in Equation 7.2

Table 7-6. Processing Time vs. the Number of Potential Boresight Craters in the Sensor Image.

Number of Desired BS Craters	Cubic Fit to the Processing Time (s)
5	0.42
10	1.05
15	2.00
18	2.91
20	3.72
25	6.67
30	11.31
35	18.09
40	27.48

Based upon Equation 7.2, the crater matching routine can process a maximum of 18 potential boresight craters and still remain within the 3 second processing time window.

The processing time is also dependent upon the total number of craters in the sensor image. Figure 7.3 presents the processing time as a function of the total number of craters in the original sensor image. A polynomial fit more precisely characterizes the relationship between the processing time and the total number of craters in the sensor image.

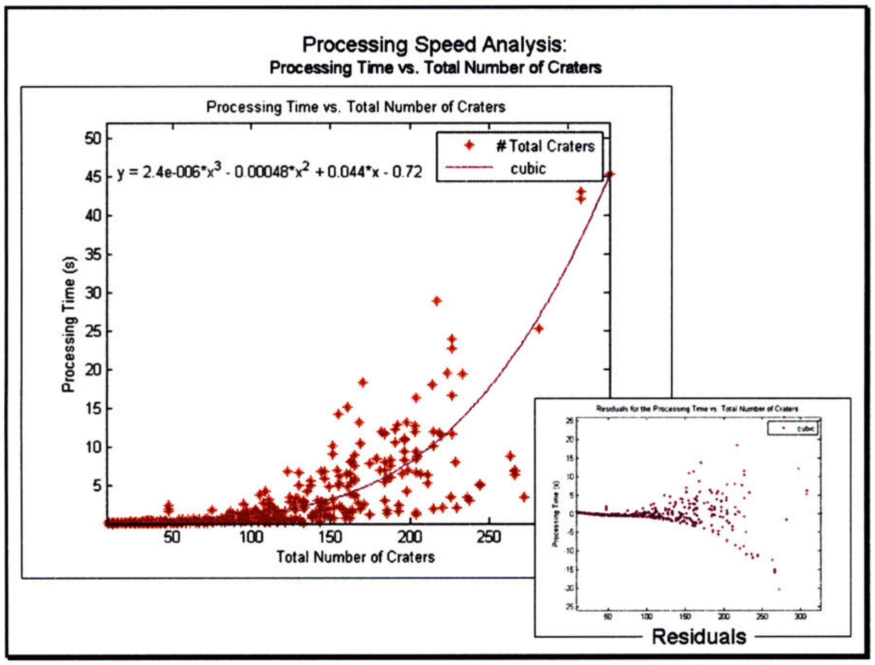


Figure 7.3. Processing Time vs. the Number of Total Craters in the Sensor Image.

Figure 7.3 demonstrates that the processing time is a cubic function of the total number of craters. The best-fit[i] cubic polynomial to the data set of Figure 7.3 is

$$y = 0.0000024 * x^3 - 0.00048 * x^2 + 0.044 * x - 0.72$$

Equation 7.3

Intuitively, there is a relationship between the number of potential boresight craters and the total number of craters in a sensor image. As the number of total craters in a sensor image increases, there will be more craters that fall near enough to the center of the image to be considered a potential boresight crater. Therefore, the relationship between the processing time and the total number of craters should be similar to the relationship between the processing time and the number of potential boresight craters. The processing time is a cubic function of the number of potential boresight

[i] The best-fit polynomial will be determined by a least-squares approximation.

craters, as noted previously. Therefore, the cubic relationship between the processing time and the total number of craters makes sense.

Notice as well the residuals in Figure 7.3 and those in Figure 7.2. The residuals from the total number of craters are very small initially, but then grow rather quickly. The residuals from the number of potential boresight craters is slightly more diffuse initially, but tends to grow less quickly. This indicates that the processing time has a greater dependency upon the number of potential boresight craters than upon the total number of craters as the number of both increases. However, it is difficult to determine which has a greater effect with lower number of potential boresight and total craters.

For the results presented in Figure 7.3, the approximate processing time is presented for several representative total numbers of craters. This processing time was estimated by the cubic polynomial in Equation 7.3

Table 7-7. Processing Time vs. the Number of Total Craters in the Sensor Image.

Number of Total Craters	Cubic Fit to the Processing Time (s)
25	0.12
50	0.58
75	0.89
100	1.28
125	1.97
147	3.00
150	3.18
175	5.14
200	8.08
250	17.78
300	34.08

Based upon Table 7-7, the crater matching routine can process a maximum of 147 craters in a sensor image and still remain within the 3 second processing time window.

As mentioned above, there is a definite relationship between the number of potential boresight craters and the total number of craters in an image. Figure 7.4 presents the total number of craters as a function of the number of potential boresight craters. In addition, the best-fit polynomial to the data set is included in the plot as well.

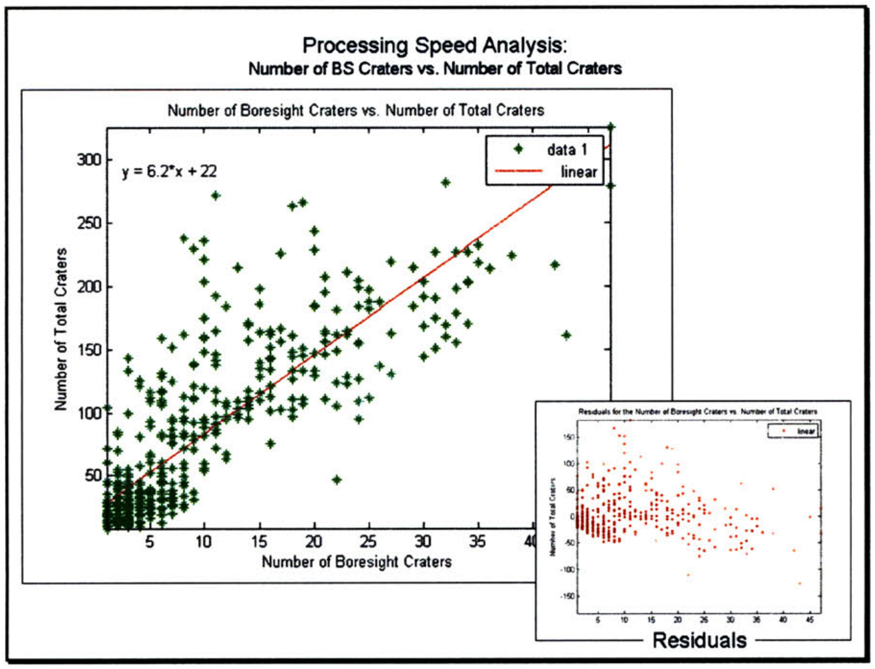


Figure 7.4. Number of Total Craters vs. Number of Potential Boresight Craters.

Figure 7.4 demonstrates a scattered linear relationship between the total number of craters and the number of potential boresight craters.[i] This linear best-fit relationship is

$$y = 6.2 * x + 22$$

Equation 7.4

[i] The author fit a quadratic to the data set, but the polynomial fit showed only very slight improvement over the linear fit. Therefore, the author decided to utilize the linear fit as the best-fit option.

For several representative values for the number of potential boresight craters, the estimated number of total craters is presented. This estimate was obtained from Equation 7.4.

Table 7-8. Number of Total Craters vs. Number of Potential Boresight Craters.

Number of Desired BS Craters	Number of Total Craters
5	53
10	84
15	115
18	134
20	146
25	177
30	208
35	239
40	270

The residuals for this fit are large, indicating the relatively weak relationship between these two quantities. In addition, the entry in Table 7-8 for 18 potential boresight craters yields 134 total craters. This quantity is relatively close to the limit of 147 total craters determined from Table 7-7. As a result, this validates that approximately 18 potential boresight craters and 140 total craters is the maximum that the routine – as programmed – can process in 3 seconds or less.

These results indicate that the crater matching routine is able to process images with a large number of total craters and many potential boresight craters within a few seconds. This processing speed meets the desired bounds – and demonstrates that with proper coding, this routine might run extremely quickly. These analyses also suggest that limits on the number of total craters to be considered in any one image, along with the number of potential boresight craters may be of significant benefit to the routine's processing speed.

7.3.4 Error Analysis

One aspect of the crater matching routine that has not yet been the topic of any serious discussion is the routine's response to error. There are several ways in which error might enter the routine, including:

1. Error in the image data points (i.e. the image does not match up with the database). This could be either error in the location of the crater centers or in the crater radii, or both.
2. Error in the altitude knowledge of the spacecraft.
3. Error in the attitude knowledge of the spacecraft / camera.
4. Error in the knowledge of the field-of-view of the camera.

The author will not analyze the effects of error in the altitude, attitude, and field-of-view knowledge. However, a brief analysis on the effects of image noise will be conducted. This analysis will consider noise in the position of the crater centers.[1] The noise in the image might result from any of a number of sources – lens aberration, inaccuracies in the database due to the imagery from which it was formed, assumptions in the routine such as the spherical Moon assumption, etc.

This analysis is not based upon 1000 iterations of the routine, as the past three analyses have been. Rather, this analysis added varying levels of white, Gaussian noise to 250 sets of image data points. For each of 43 levels, white, Gaussian noise was added to 250 sensor images and the images were each processed by the crater matching routine. The noise levels were selected to be bounded by $[500 \quad 2]$ signal-to-noise ratio (SNR). Each level was determined by raising the previous SNR to 0.95, resulting in a series of 43 SNRs. These bounds and this method of determining the next level were chosen to give a desired distribution of analysis points.

[1] The effect of noise in the crater radii can be mitigated by the correlation factor ε_1. This correlation factor specifies the acceptable error range in the cells of a match between pattern vectors.

First, a characterization of the noise levels will be introduced in Figure 7.5, Figure 7.6, and Figure 7.7. Next, Figure 7.8, Figure 7.9, and Figure 7.10 analyze the overall performance of the routine, primarily considering the success or failure rates of the routine and its components. Figure 7.11 and Figure 7.12 present the causes for signature generation failure. Figure 7.13, Figure 7.14, and Figure 7.15 address the correlation results – specifically looking at inconclusive position matches and component failures.

The author also chose to randomly select ordered pairs, with each element randomly selected from the range $[0 \quad 512]$.[i] These ordered pairs represent the pixel location of random points within a sensor image with a resolution of 512 pixels.[ii] Then noise was added to these simulated data points at varying SNRs along the range of $[500 \quad 2]$. The results are plotted in Figure 7.5, Figure 7.6, and Figure 7.7. To better characterize the amount of noise that is actually being added – the maximum noise level, as well as the standard deviation, are presented here versus the image error in pixels.

[i] Random pixel locations are selected to enable a greater sample size for the analysis.
[ii] These ordered pairs will be considered the signal.

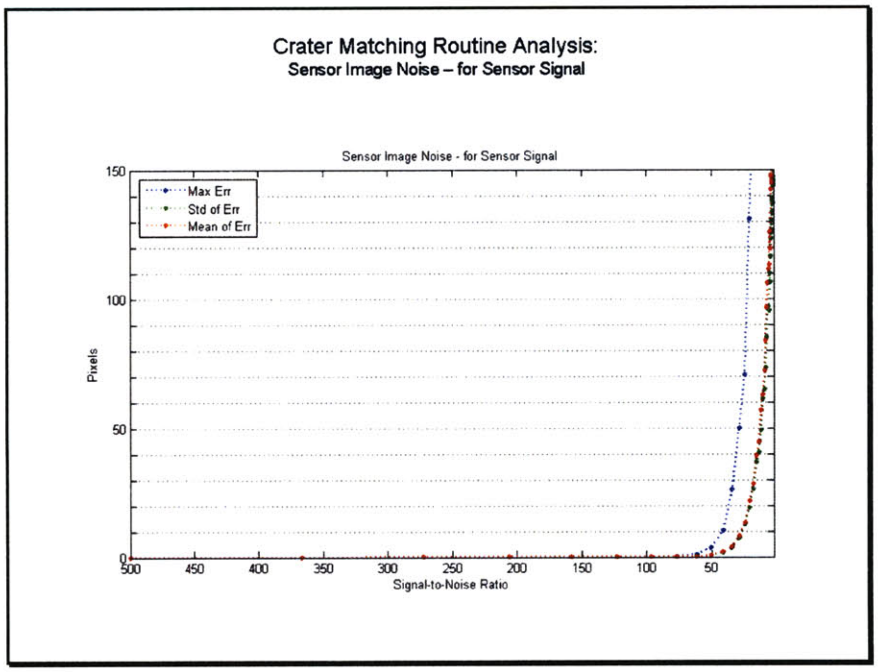


Figure 7.5. Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels.

Figure 7.6 presents a detail focusing on SNRs under 75.

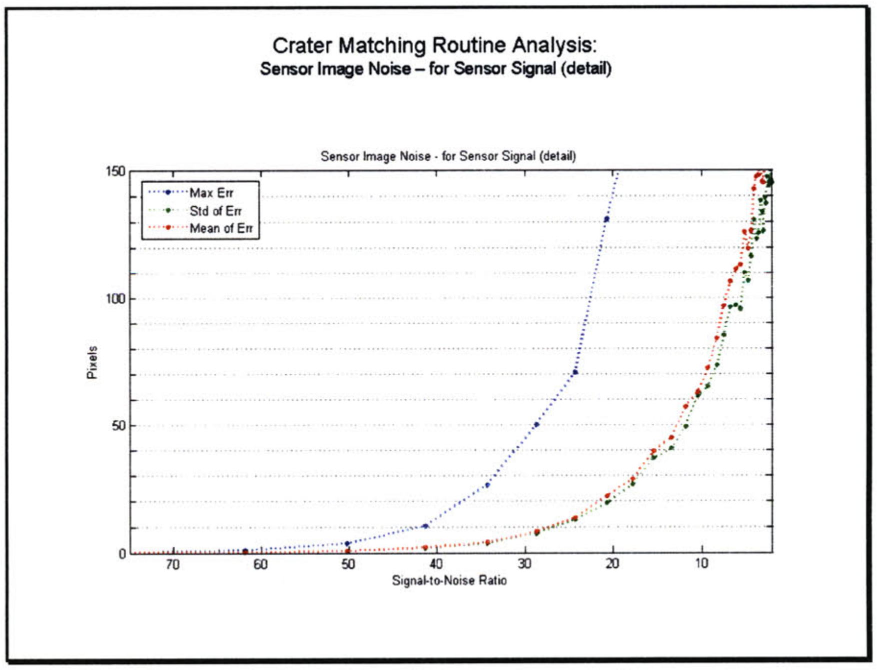


Figure 7.6. Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels, Detail #1.

Figure 7.7 looks exclusively at the SNR range of $[65 \quad 40]$. This range is important as the maximum noise being added crosses the 1 pixel threshold in this range.

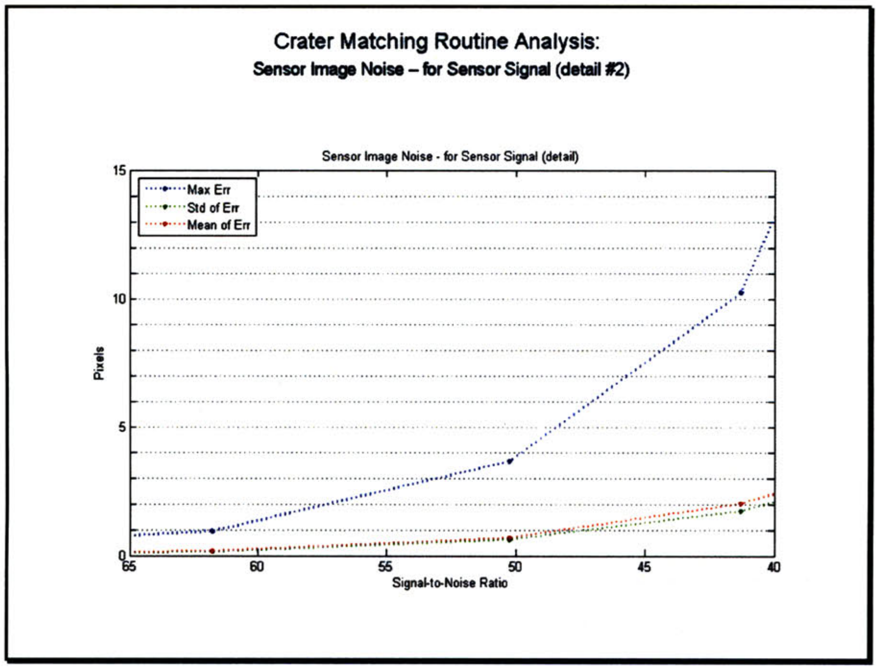


Figure 7.7. Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels, Detail #2.

At an SNR of 50, the mean added noise in pixels is right below 1 pixel. The standard deviation of the added noise is also right below 1 pixel. The maximum added noise is almost 4 pixels. The error added was white Gaussian noise. Because this is a normal distribution – the standard deviation indicates that approximately 68% of the data points lie within one standard deviation of the mean. In other words – at an SNR of 50 - 68% of the magnitude of the noise added will lie within the range of $[0 \quad 3]$ pixels, approximately. That also means that 32% of the noise added was greater than 3 pixels in magnitude, which is a significant amount of noise. The effects of this added noise will be seen in the following analyses.

Figure 7.8 is a plot of the general response of the routine to noise levels. This plot records the overall success and failures of the routine. Figure 7.9 is a detail of Figure 7.8 that focuses upon the SNR range of $\begin{bmatrix}40 & 2\end{bmatrix}$.

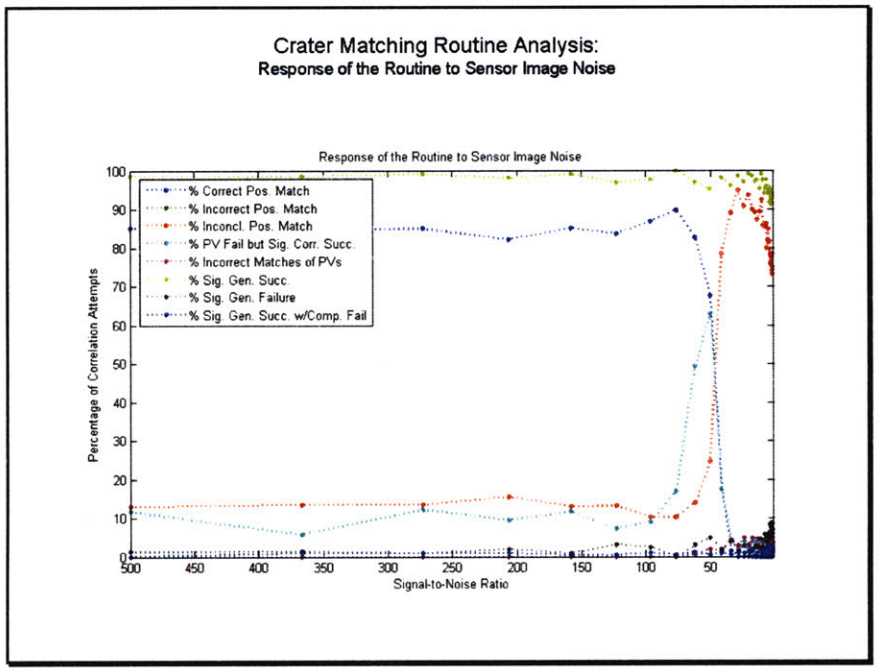


Figure 7.8. Analysis of the Routine: Response of the Routine to Sensor Image Noise.

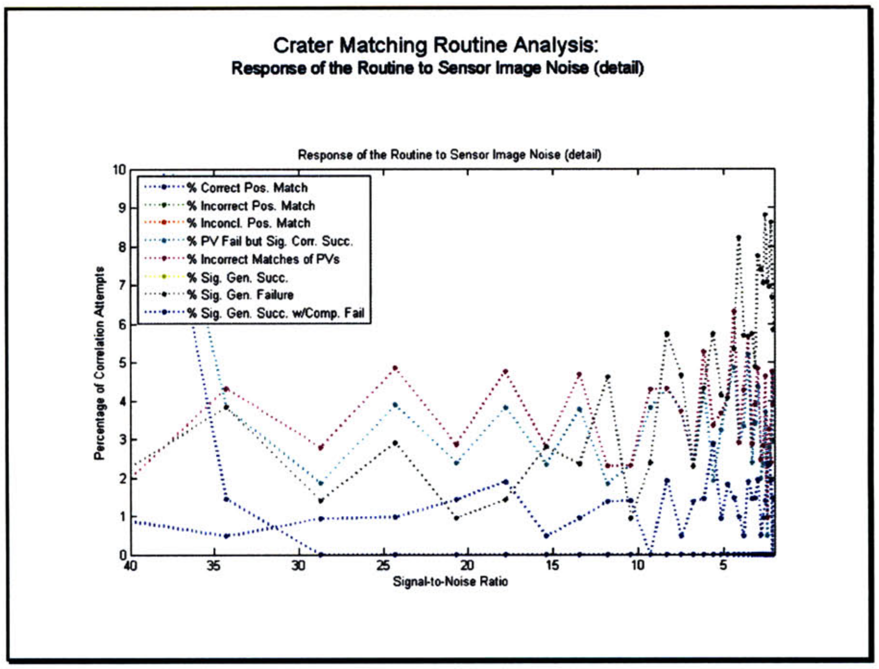


Figure 7.9. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Detail.

Based upon Figure 7.8 and Figure 7.9, it appears that the routine is fairly stable for SNRs of (approximately) $[500 \quad 75]$. For this range, the routine demonstrates a $80-90\%$ successful correlation rate, a $10-20\%$ inconclusive position match rate, and 0% occurrence of incorrect position matches.

As the SNR continues to drop, these stable characteristics change dramatically. Between the SNR range of $[75 \quad 25]$, the percentage of correct matches plummets, reaching 0% correct matches when the SNR falls just below 30. However, the occurrence of inconclusive matches increases just a quickly, reaching a rate of $80-90\%$ inconclusive matches. At the same time, the number of incorrect matches also increases from 0% to $3-6\%$ as the SNR drops.

The data for 'Incorrect Position Match' and 'Incorrect Matches of PVs' lie directly atop one another in Figure 7.9. This implies that *every* false match results from the incorrect match of one pattern vector to the crater database. In other words, all of the incorrect matches result from high noise levels and image signatures that are comprised of only one pattern vector – and this pattern vector only has one 'full' grid cell. Refer to section 7.3.2 for a more in depth discussion of the effects of a pattern vector with a single 'full' grid cell.

Figure 7.10 concentrates on the correct, incorrect, and inconclusive position matches, and plots the percentage of each of these quantities as a stacked area graph.

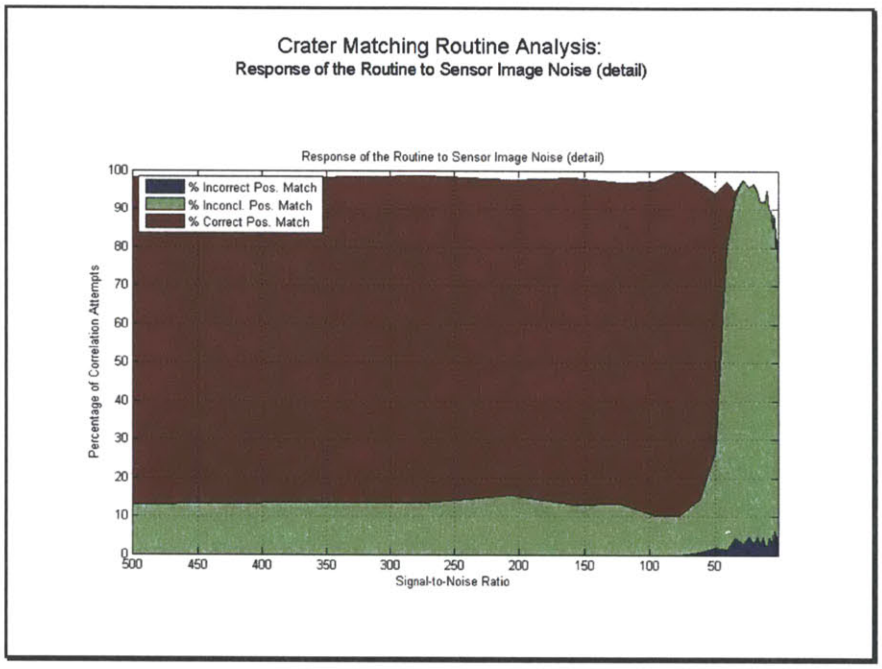


Figure 7.10. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Position Match Success and Failure.

Figure 7.10 highlights the relative proportions of each quantity in relation to the total number of correlation attempts. The sum of the percentages of correct, incorrect, and inconclusive matches does not (typically) equal 100%. This results from the failure of the signature generation for a portion of the correlation attempts, which are not included in this plot. In addition, there are no false position fixes until the SNR drops to about 75. This meets the requirement stated in section 3.2.1.1 that there are less than 0.1% false position matches. The percent of correct matches remains well above 75% until the SNR drops to approximately 50, meeting the success criterion of section 3.2.1.1.

Figure 7.11 presents the reasons that the generation of a signature failed as a function of the noise added to the sensor image.

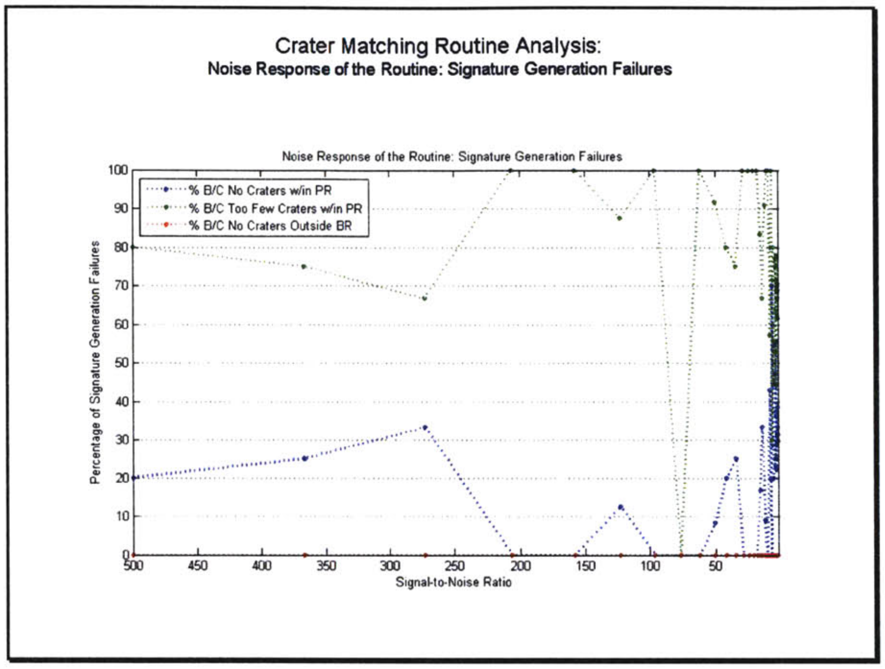


Figure 7.11. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Signature Generation Failure.

For all noise characteristics, all of the signature generation failures that were experienced resulted from too few – or no – craters within the pattern radius. Without any craters within the pattern radius, the grid matrix is completely empty and a pattern vector cannot be formed, and therefore the signature generation fails as well. The total number of signature generations does not include images that were initially disqualified for having no suitable boresight craters.

There are typically multiple pattern vectors in one image signature. As a result, it is possible to have one – or even several – pattern vectors fail to generate, but still have a successful signature formation. It is interesting to consider the scenario when the signature generation succeeds, but one or more component pattern vector formation attempt fails. Figure 7.12 analyzes the causes for component pattern vectors to fail to generate properly – but yet have a successful signature generation.

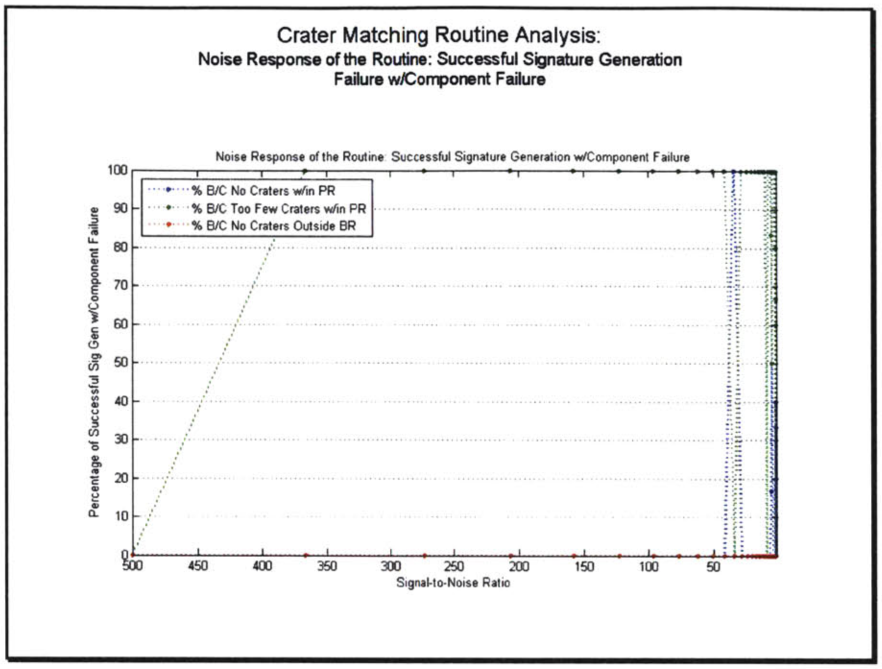


Figure 7.12. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Signature Generation Success with Component Failures.

From Figure 7.12, as in Figure 7.11, the exclusive cause of these component failures is that there are too few craters within the pattern radius.

Figure 7.13 begins to examine the correlation process rather than signature generation. This figure presents an analysis of the reasons that the correlation routine returns an inconclusive position match.

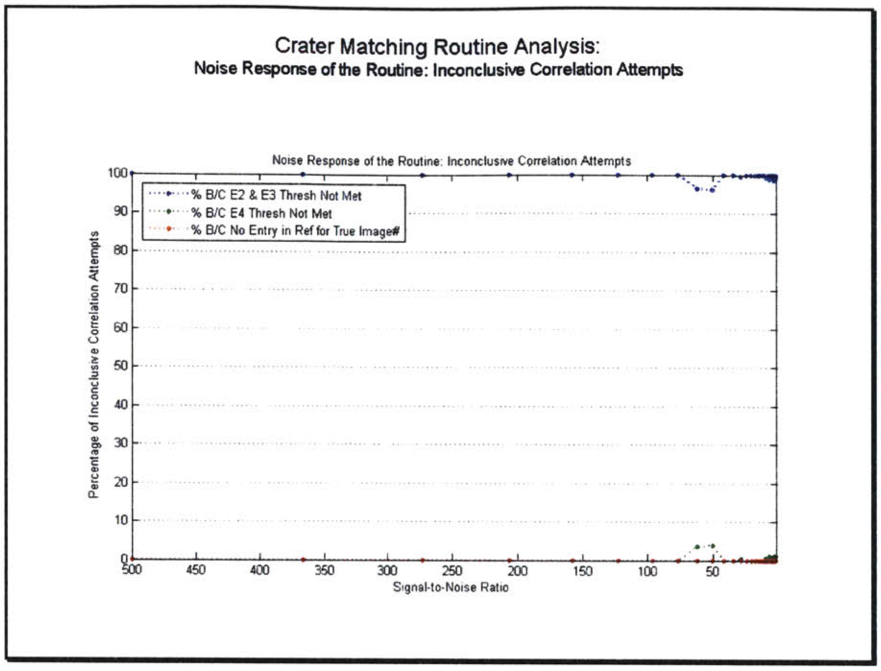


Figure 7.13. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Inconclusive Correlation Attempts.

Nearly all of the inconclusive correlation attempts were caused by the ε_2 or ε_3 correlation thresholds not being met. For this analysis, $\varepsilon_2 = 0.75$ and $\varepsilon_3 = 0.75$. Figure 7.14 will dissect the ε_2 and ε_3 correlation threshold error further, examining each threshold individually to further characterize the cause of the inconclusive correlation attempts.

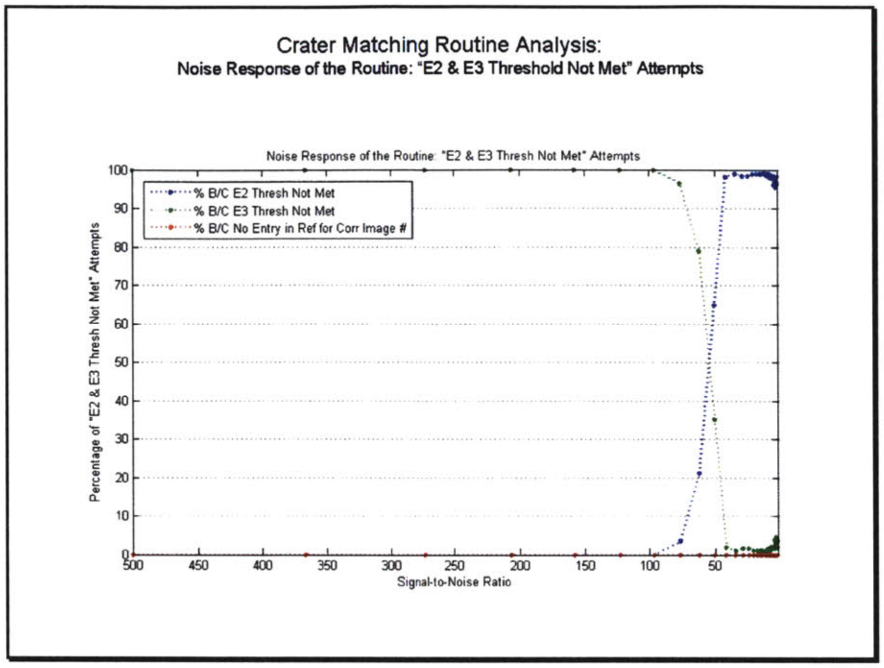


Figure 7.14. Analysis of the Routine: Response of the Routine to Sensor Image Noise, "E2 & E3 Threshold Not Met" Attempts.

From Figure 7.14, it is apparent that at lower noise levels, all of the inconclusive correlation attempts are caused when correlation attempts fall short of the ε_3 correlation threshold. The ε_3 correlation threshold defines the minimum ratio between the frequency of a positive position match and the second most frequent crater from the list of possible crater matches. As noise increases, however, a greater number of correlation attempts fall short of the ε_2 correlation threshold. The ε_2 threshold is a ratio that describes the percentage of pattern vectors that must identify the same image location for a successful position match. For example, if a signature consists of 10 pattern vectors, an ε_2 threshold of $\varepsilon_2 = 0.8$ would require that at least 8 pattern vectors identify the same image location before a positive position match for that image could be declared.

Figure 7.15 is an analysis of the situation where one or more pattern vectors fails to generate, but the signature for the given sensor image is still successfully generated. This figure analyses the reasons that these pattern vectors failed to generate.

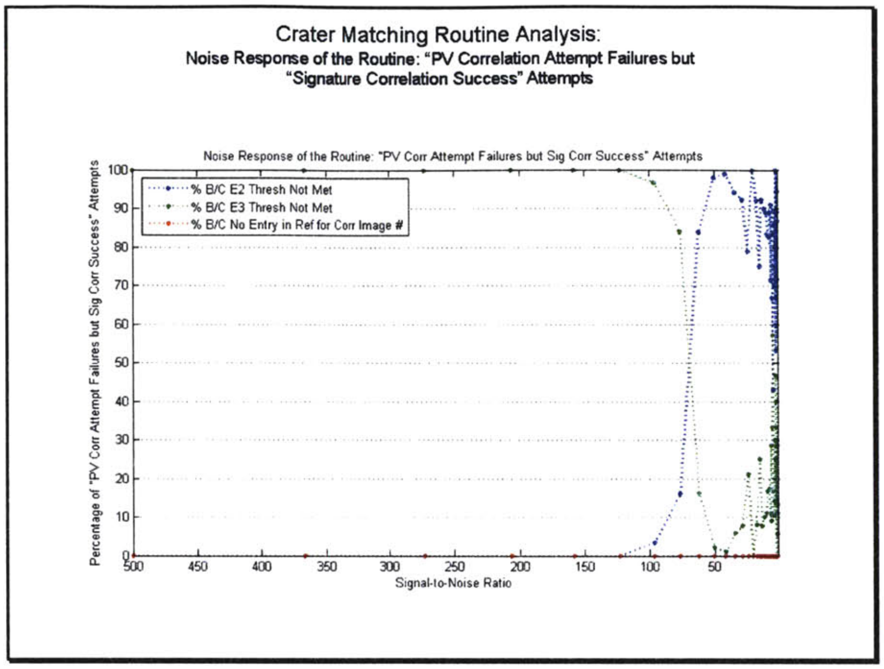


Figure 7.15. Analysis of the Routine: Response of the Routine to Sensor Image Noise, Signature Matching Success with Component PV Matching Failures.

The reasons evident in Figure 7.15 for pattern vector generation failure closely parallel the patterns evident in Figure 7.14.

These error analyses indicate that the crater matching routine, as initialized, performs well at a SNR above 50. Although this SNR may seem rather high – it correlates to a mean error of 1 pixel with a standard deviation of 1 pixel. In other words, 68% of the data points have 2 pixels or less noise – with the remainder having a greater amount of noise added. For images with a SNR in the range of $[50 \quad 2]$, the frequency of incorrect matches remains rather low, at $3-6\%$. These analyses also indicate that signature generation fails almost exclusively because there are too few craters within the pattern radius. The failure of correlation attempts is due to both the ε_2 and ε_3 correlation thresholds most often. At high levels of noise, the ε_2 threshold causes the majority of the correlation failures, while at lower noise levels, the ε_3 threshold causes most of the correlation failures.

As varying levels of noise can be expected in the sensor images of the lunar surface, it is crucial for the crater matching routine to be robust to these conditions. The four correlation factors provide a great deal of user control in encountering a variety of noise conditions. In addition, there are other factors that can greatly influence the amount of noise that the routine can handle. Chief among these is the grid size, g. Each grid cell covers multiple pixels from the image. Therefore, if the grid size is increased, the routine can naturally handle a greater amount of image noise and still generate a positive position match.

The correlation factors also allow the routine to process images that have either 'extra' craters or craters that 'disappeared.' In other words, if an image has an additional crater in it that did not appear in the imagery used to form the crater database, the routine will still be able to generate a position match. The same can be said if a crater is missing from the sensor image that was there in the crater database. The routine is able to handle these cases with ease due to two aspects of the crater matching routine. First, the routine assumes that only a percentage of the pattern vectors from the signature are going to correlate to the true image location. Secondly, the routine also assumes that only a percentage of the 'full' grid cells for each pattern vector in the signature are going to correlate to the pattern vectors in the database. The addition or subtraction of 'full' grid cells will change the percentage of 'full' grid cells that match the pattern vectors from the database, and may impact the percentage of signature pattern vectors that correlate to the true image location. However, because there are two thresholds that limit the impact of the addition or subtraction of 'full' grid cells, the routine should handle the 'extra' or 'disappearing' craters easily. This assumes that the appearance of 'extra' craters or having craters 'disappear' is a relatively infrequent occurrence.

7.4 Crater Matching Routine Conclusions

Although conceptually the crater matching routine is rather simple, the implementation of the routine is quite process intensive. That is, the steps required to execute the routine are numerous, and they must be completed in a very specific order. The main steps to the crater matching routine that were first presented in section 3.2.1, which can be listed simply as:

1. Generate a database of patterns of nearby craters for any given crater on the lunar surface. This database will be stored and queried during the correlation of the image to the database.
2. Determine the pattern of nearby craters for a given crater in the sensor image about which the pattern is generated.
3. Generate a 'signature' for a given sensor image by repeating step 2 for every possible crater in an image.
4. Correlate the patterns in the image signature with the database of patterns. Successful correlation will allow the absolute location of the craters within the sensor image to be determined.

These four steps – database generation, pattern vector determination, signature development, and correlation – are the crux of the crater matching routine. Each of these steps has been individually and extensively detailed in the preceding chapter.

The overall implementation of the crater matching routine has been very successful. The results displayed above in the analysis section indicate that the grid-based pattern vector is an effective means of 'tagging' a crater for future identification. In addition, the database size was not prohibitive, the processing time was reasonable, and the achieved accuracy promising.

More specifically, the author was very pleased with the processing speed of the algorithm. As mentioned in section 7.3.3, the routine is programmed in an engineering analysis computer language – which is not designed to match the speed of operational code. In addition, there are many extra features in the code that will be removed for operational code. Therefore, the ability to process sensor images with 18 potential boresight craters and about 140 total craters in less than 3 seconds is a significant accomplishment.

The correlation between the current program size and the (eventual) operation program size is difficult to gauge. Currently, the crater matching program files are approximately 2400 lines of code and occupy $100KB$ on the computer's hard drive. The graphing files are approximately 550 lines of code and occupy $22.5KB$ of memory. In total, the crater matching routine, as currently coded, is almost 3000 lines of code and requires about $125KB$ of memory for storage. The database is saved as a look-up table and is $1.66MB$ in size.[1] Considering the magnitude of the crater matching task, the size of the crater matching routine is within acceptable bounds.

The routine, with the initializations presented in section 7.2, proved able to generate correct position matches for images with signal-to-noise ratios as low as 50. While the frequency of correct matches declines quickly at signal-to-noise ratios below 50, the frequency of incorrect matches remains relatively low at $3-6\%$ for these low signal-to-noise ratios. In addition, the modification of the user-defined correlation factors will allow the crater matching routine to be prepared for situations with either a higher or a lower level of noise. These same correlation factors also enable the crater

[1] The database being referenced was made by acquiring a series of 1000 images of the lunar surface within the latitude and longitude bounds of $[-80 \quad -50]^{\circ}$ and $[50 \quad 80]^{\circ}$. All of the potential boresight craters were identified from the images and a pattern vector formed for each potential boresight crater. Then each pattern vector was saved to the database in the form of a look-up table. Note that the database does not necessarily cover the entire lunar surface between these bounds. In addition, this database requires that the sensor images used for the crater matching routine (pre-noise) must be the same as the images used for the database generation. This assumption does not negate the validity of the analysis accomplished herein; rather, it is merely a simulation requirement that will not be encountered once a database covering the entire lunar surface can be generated.

matching routine to be proficient at generating a positive position match when there are craters in the image that were not in the database.[1] The initial analysis of the crater matching routine's response to error in the sensor image demonstrates that the routine is robust to significant levels of noise, and still producing accurate results.

The author was unable to find any published results from another crater matching algorithm. Attempts to compare the results of this crater matching routine with analogous results from published star tracker algorithms, such as those presented by Padgett, Kreutz-Delgado, and Udomkesmalee[25], proved unfruitful. Therefore, the author will suffice to present the results of this crater matching routine.

There were several difficulties encountered in the implementation of the routine. As noted in section 3.2.2.3, the crater matching routine and grid-based pattern vector determination is based upon a process developed as a 'lost in space' algorithm for star trackers. The star tracker problem is very similar to the crater matching problem. Both require that an image from a sensor be paired with a database of image features so that the position and/or orientation of the spacecraft can be determined more accurately. Both require speed and reliability, while limiting the available disk space for operations. In both cases, imagery that correlates to possible future sensor images is available, which allows the generation of a database before the spacecraft launches. However, there are also inherent differences between the star tracker problem and the crater matching problem. Chief among these is the problem of frame of reference. In the case of star trackers, the imagery is taken of a portion of the celestial sphere. In comparison to a typical spacecraft, the celestial sphere can be considered an inertial frame. In the case of the crater matching routine, the images are not taken of an inertially static object. In other words, the position of the spacecraft will affect the relative appearance of the features in the image.

[1] This situation might be encountered due to the frequent meteor strikes on the lunar surface. Since the database is based off of lunar imagery, the time between the images collected for database formation and those taken by the spacecraft's sensor may demonstrate some of the effects of these meteor strikes in the form of 'extra' craters.

This difficulty with the frame of reference was the author's most significant modification to the grid-based star tracker algorithm. Stated more simply, the problem is that as the spacecraft moves in three-space in relation to the surface of the Moon, the appearance of any specific location on the lunar surface changes from the camera's point of view. These changes in appearance are due to the curvature of the lunar surface in relation to the planar nature of the image plane, as well as due to the changing altitude of the spacecraft.

In fact, the problem was only partially solved. The author decided to separate the problem of the spacecraft's changing altitude (in relation to the lunar surface) from the problem of the curved lunar surface being projected onto a two-dimensional image plane. The author determined a solution to the later problem, but did not address the situation of a spacecraft's changing altitude. The author will suggest several solution methods in the recommendations for continued work in section 8.3.

The difficulty with the spherical lunar surface and a two-dimensional image plane is described more completely in section 4.1. The solution method is the application of the perspective transform. This transform allows a given sensor image to be 'morphed' so that it appears as if the spacecraft was at a different location relative to the lunar surface when the image was taken. This eradicates the need for the database to include multiple feature positions (which would have been otherwise necessary to account for various spacecraft positions relative to the image location on the lunar surface).

The analyses presented herein demonstrate that the crater matching routine is a functional technique for generating a position correlation from an image of the lunar surface. This method merits further development and analysis to determine if it is both suitable and better than other crater matching methods. If it suitable and better than other methods, this crater matching method may be appropriate to implement on hardware and send to the Moon. If not, the crater matching routine has still served to increase understanding of relative navigation techniques.

Chapter 8

Conclusions

This paper investigated two alternative correlation schemes for lunar terrain relative localization (TRL) in an attempt to determine the applicability of each of these schemes to lunar entry, descent, and landing (EDL) missions. The theoretical and functional basis for each of these algorithms is detailed in the preceding chapters. In addition, the author has presented analyses to characterize the performance of each algorithm. These analyses invoke initial conditions that were chosen to mimic conditions that are expected in the lunar terrain relative navigation (TRN) task. These analyses are presented in full detail in the body of this paper. This section contains a brief summary of results, the author's comments on the meaning of these results, and several recommendations for future work are presented in this chapter.

The author also performed functional analyses of the perspective transform, which is a component of the crater matching routine. However, these analyses were merely intended to ensure the proper functionality of the transform rather than to characterize its performance. These analyses do not directly provide insight into the main thesis of the topic. That is, the results to the perspective transform analyses do not answer the question as to whether or not the crater matching routine is suitable as a TRN component to the lunar EDL mission. Therefore, the results to the perspective transform analyses and conclusions based upon them will not be presented here. Suffice it to say that the perspective transform analyses demonstrated that the transform functions as expected.

8.1 Summary of Results

The results of the implementation and analysis of the terrain contour matching (TERCOM) algorithm and the crater matching routine will be presented below. These results are presented in greater detail in Chapter 2 and Chapter 7.

8.1.1 TERCOM

The analysis of TERCOM focused upon the ability of TERCOM to generate a correct position fix given varying levels of image noise. The initial conditions of the analysis attempted to mimic actual conditions that might be encountered in lunar TRN.

For the analysis, the CraterMap utility is used to generate 100 meter resolution TERCOM reference maps. The Lunar Reconnaissance Orbiter mission will also provide lunar maps with 100 meter resolution. Therefore, these maps mimic the quality of the data expected to be available for lunar navigation. To simulate the sensor measurements which will be processed to yield terrain profiles, the maximum sample rate is assumed to be $100\ Hz$ and the maximum vehicle ground velocity is assumed to be 1700 meters per second. This correlates to an altimeter sample every 17 meters. Several of these samples are averaged to obtain a cell in the terrain profile.

Only two sources of error were introduced to the execution of the TERCOM algorithm, map error and sensor error. Both error sources are modeled with a zero mean Gaussian distribution that has an adjustable standard deviation value. The map error represents the error between the map and the true terrain. The sensor error is the error in the measured terrain profile cells and has multiple potential sources. These sources include sensor inaccuracies, imprecise attitude knowledge of the vehicle, and error caused by under-sampling the altitudes for a given map cell. For simplification, all sensor error sources are modeled as a single error term.

The TERCOM matching algorithm performed poorly under various surface conditions. Figure 8.1 and Figure 8.2 below are TERCOM's best and worst responses to image noise, respectively.[1] The sigSensor term represents the error added to the sensor measurements in meters and the sigMap term represents the amount of error added to the reference map in meters. The quantity σ_T referenced in these figures is a measure of how rough the terrain is, with a greater σ_T value corresponding to a rougher terrain.

[1] In these figures, LSSM and VAR stand for the Long Sample Short Matrix and the Variance algorithm, two specifications of the type of TERCOM algorithm implemented.

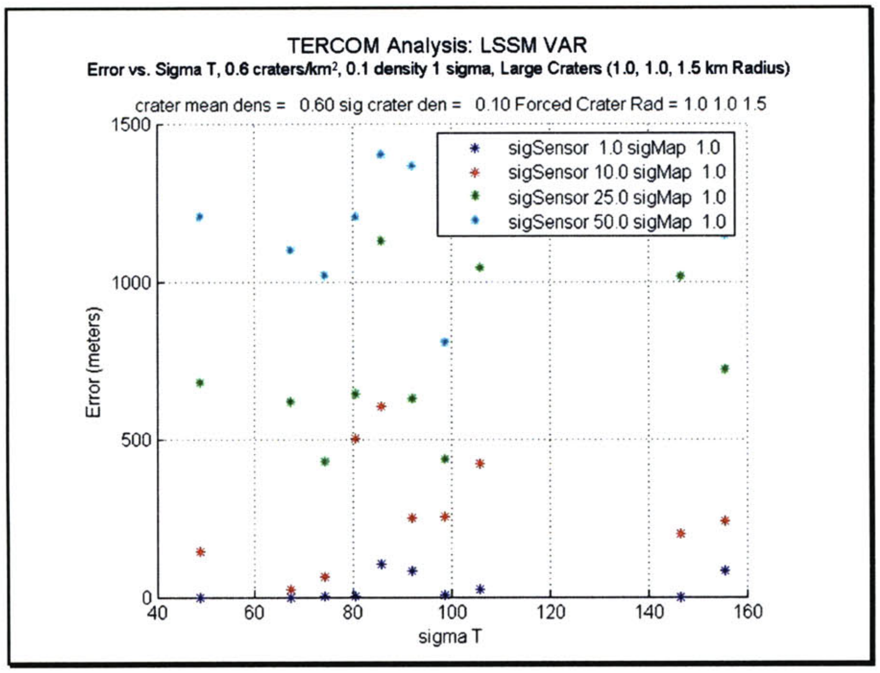


Figure 8.1[i]. Reprint TERCOM LSSM VAR Analysis: 0.6 craters/km^2 with standard dev. of 0.1 craters/km^2, Forced Large Craters of [1.0 1.0 1.5] km Radius.

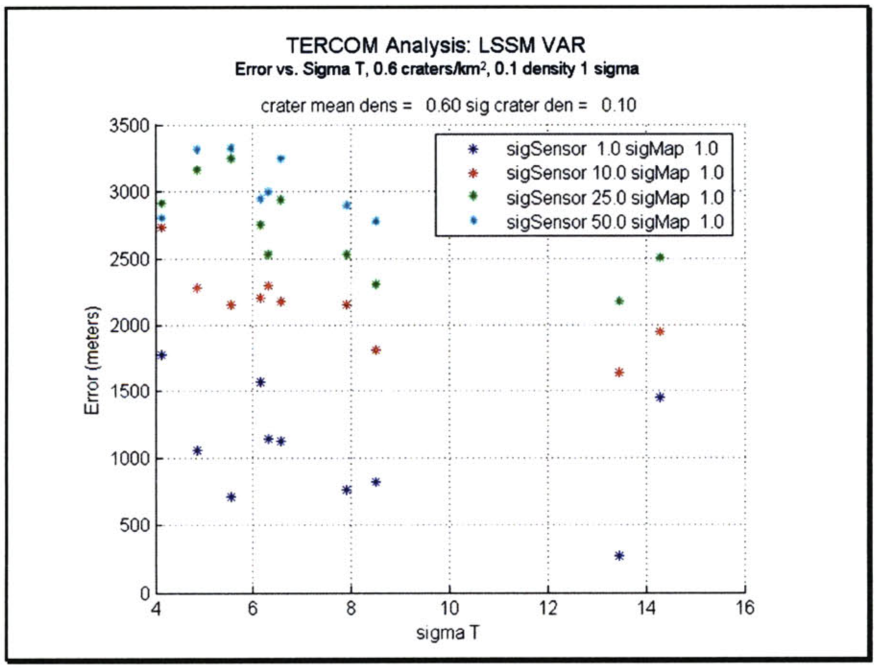


Figure 8.2[i]. Reprint TERCOM LSSM VAR Analysis: 0.6 craters/km2 with standard dev. of 0.1 craters/km2.

[i] Also Figure 2.17

In general, the error levels in the TERCOM position fix are unacceptably high. Even for the best case scenario, with a high density of craters and many large craters, any sensor noise above 10 meters – at any altitude – usually results in a position fix error of more than $500\ m$. The worst case results, for a scenario with a high crater density but no large craters, results in position fix errors well above $500\ m$ for all σ_T values. These results demonstrate that TERCOM is not able to generate position fixes for the terrain likely to be encountered in a lunar EDL mission.[ii]

8.1.2 The Crater Matching Routine

The initial conditions used for the analysis of the crater matching routine represent a plausible situation encountered by a spacecraft attempting to use the lunar surface for a terrain relative position update of the spacecraft's inertial guidance systems. The analyses are conducted for a spacecraft with a camera that orbits the Moon at an altitude of $100\ km$ with perfect attitude. There is a camera on this spacecraft that has a 90° field-of view and a resolution of $512\ pixels$. The Moon is considered a spherical body with a radius of $1734.7\ km$, which is a generally accepted value for the lunar radius.[26] This spacecraft takes a variety of pictures of the lunar surface in the latitude bands of $[-80\quad -50]^\circ$ and $[50\quad 80]^\circ$. Next, the crater matching routine attempts to correlate these pictures with the crater database that was formed of craters in the same area.[iii] The correlation attempts used a pattern radius of $65\ km$ and a buffer radius of $6.5\ km$. The grid overlay had 24 rows and the same number of columns. The correlation factors, which define 'how close' a correlation between an image and the database must be for a position match, are set as

[i] Also Figure 2.15

[ii] However, TERCOM has been used extensively for cruise missile guidance. It is expected that the increase in position fix error that was commented upon is caused by the altitude difference between cruise missile application of TERCOM and the use of the same in spacecraft navigation.

[iii] In other words, this 'simulated spacecraft' does not fly on a simulated trajectory. Rather, the 'simulated spacecraft' parameters were used to generate representative sensor images that are randomly distributed between the latitude and longitude bounds.

$$\{(\varepsilon_1 = 2 \ \ pixels) \ \ (\varepsilon_2 = 0.75) \ \ (\varepsilon_3 = 0.75) \ \ (\varepsilon_4 = 0.8)\}$$

Equation 8.1

and the minimum radius that a crater has to have to be considered in the correlation algorithm was $0.8 \ km$.

Correlation is considered to be successful when the true latitude and longitude of the sensor image match the latitude and longitude of the sensor image as determined by the crater matching process. The true latitude and longitude are known because the sensor input is simulated based upon a desired camera position, in latitude, longitude, and altitude.

Using these initial conditions, the analysis of the crater matching routine is initiated. The results of this analysis focus upon several areas; the success/failure of the routine to generate a position fix, the processing speed of the routine, and the routine's response to image noise.

The overall success and failure of the routine is summarized in Table 8-1.

Table 8-1[i]. Reprint General Success / Failure Statistics from the Analysis of the Routine.

# of iterations:	**1000**	
# of unsuitable images:	**151**	**15.1%**
# of correlation attempts:	**849**	**84.9%**
# of successful signature gen.:	825	97.17%[ii]
# of signature gen. failure:	24	2.83%[ii]
# of correct position match:	742	87.4%[ii]
# of incorrect position match:	0	0%
# of inconclusive position match:	83	9.78%[ii]

From Table 8-1, 87.4% of all crater matching attempts successfully identified the correct position of the image. Given the stringency of the correlation factors – this is a definite success. *Perhaps more promising is that there were no position matches that were generated that falsely identified the*

[i] Also Table 7-1
[ii] Percent relative to the total number of correlation attempts.

position of the sensor image! Given that there was no noise added for this analysis, this is not wholly unexpected. However, it does indicate that the pattern vectors are unique from one another for the given database, and the crater matching routine can successfully generate a position fix for a given image.

In addition, there are several other trends that merit attention. Analysis indicates that 100% of the failures to generate a signature resulted from multiple potential boresight craters with too few craters within the pattern radius. This indicates that it may be appropriate to screen the sensor images for a minimum number of total craters before passing the images to the crater matching routine. These screening operations would save processing time for spacecraft computers and identify images that are likely to have very sparse pattern vectors anyway – which are ill-suited for correlation.

The second noteworthy trend is that 100% of the failed correlation attempts were due to the failure of the correlation process to meet the ε_3 threshold. This threshold defines a minimum ratio between the certainty associated with the best and second best matches between a pattern vector and the database. This trend indicates that the correlation attempts fail most frequently – always, for this analysis – because there are more than one pattern vectors in the crater database that match a given pattern vector for the sensor image very well. This can be mitigated by requiring each image signature to include several pattern vectors before correlation is attempted. In addition, requiring multiple 'full' grid cells per pattern vector will reduce the number of failed correlation attempts due to the ε_3 threshold.

The analysis also demonstrated that occasionally signatures were successfully generated while one or more component pattern vectors of the signature failed to generate. This was due to a very limited number of craters within the pattern radius for these cases. Likewise, there were successful correlation attempts that had one or more component pattern vector matching attempts that failed.

This was due to a failure of the correlation process to meet the ε_3 correlation threshold for the given crater. Despite the failure of these individual components, the overall crater matching routine succeeded in generating a position correlation in each of these cases.

Processing time is another important performance characteristic for the crater matching routine. The analysis of the routine's processing time focused upon the relationship between the number of potential boresight craters, the number of total craters in the sensor image, and the processing speed of the routine. Table 8-2 and

Table 8-3 present the numeric results of this analysis, obtained by fitting a cubic polynomial curve to the processing time data for all of the analysis iterations.

Table 8-2[1]. Reprint Processing Time vs. the Number of Potential Boresight Craters in the Sensor Image.

Number of Desired BS Craters	Cubic Fit to the Processing Time (s)
5	0.42
10	1.05
15	2.00
18	2.91
20	3.72
25	6.67
30	11.31
35	18.09
40	27.48

[1] Also Table 7-6

Table 8-3[1]. Reprint Processing Time vs. the Number of Total Craters in the Sensor Image.

Number of Total Craters	Cubic Fit to the Processing Time (s)
25	0.12
50	0.58
75	0.89
100	1.28
125	1.97
147	3.00
150	3.18
175	5.14
200	8.08
250	17.78
300	34.08

These results indicate that the crater matching routine is able to process images with a large number of total craters and many potential boresight craters within a few seconds. Specifically, an image with 18 potential boresight craters and 140 total craters can be processed by this routine in under 3 seconds. This processing speed meets the desired bounds – and suggests that with proper coding, this routine might run very quickly. These analyses also support limiting the number of total craters in any one image, along with the number of potential boresight craters, to provide significant benefit to the routine's processing speed.

For the final analysis, varying amounts of noise were added to the center position of the craters in the original sensor image to analyze the response of the crater matching routine to image noise. For each case, the amount of noise added and the error of the position fix generated were recorded.

A detail of the routine's tendency to generate a correct, incorrect, or inconclusive position match is highlighted in Figure 8.3.

[1] Also Table 7-7

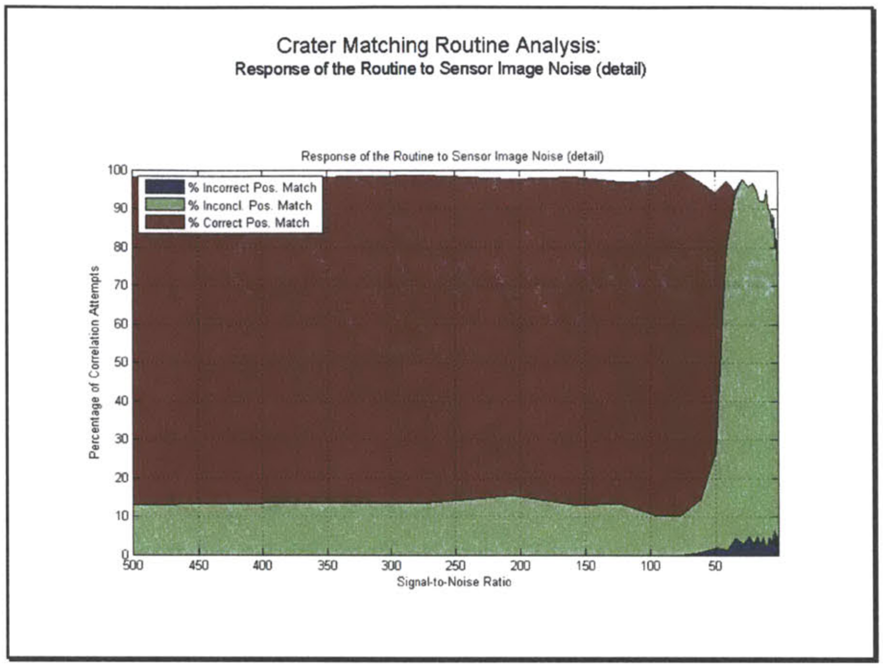


Figure 8.3[i]. Reprint Analysis of the Routine: Response of the Routine to Sensor Image Noise, Position Match Success and Failure.

Figure 8.4 shows the relationship between the signal-to-noise ratio of the noise added to the image and the resulting error in the crater center position in pixels.

[i] Also Figure 7.10

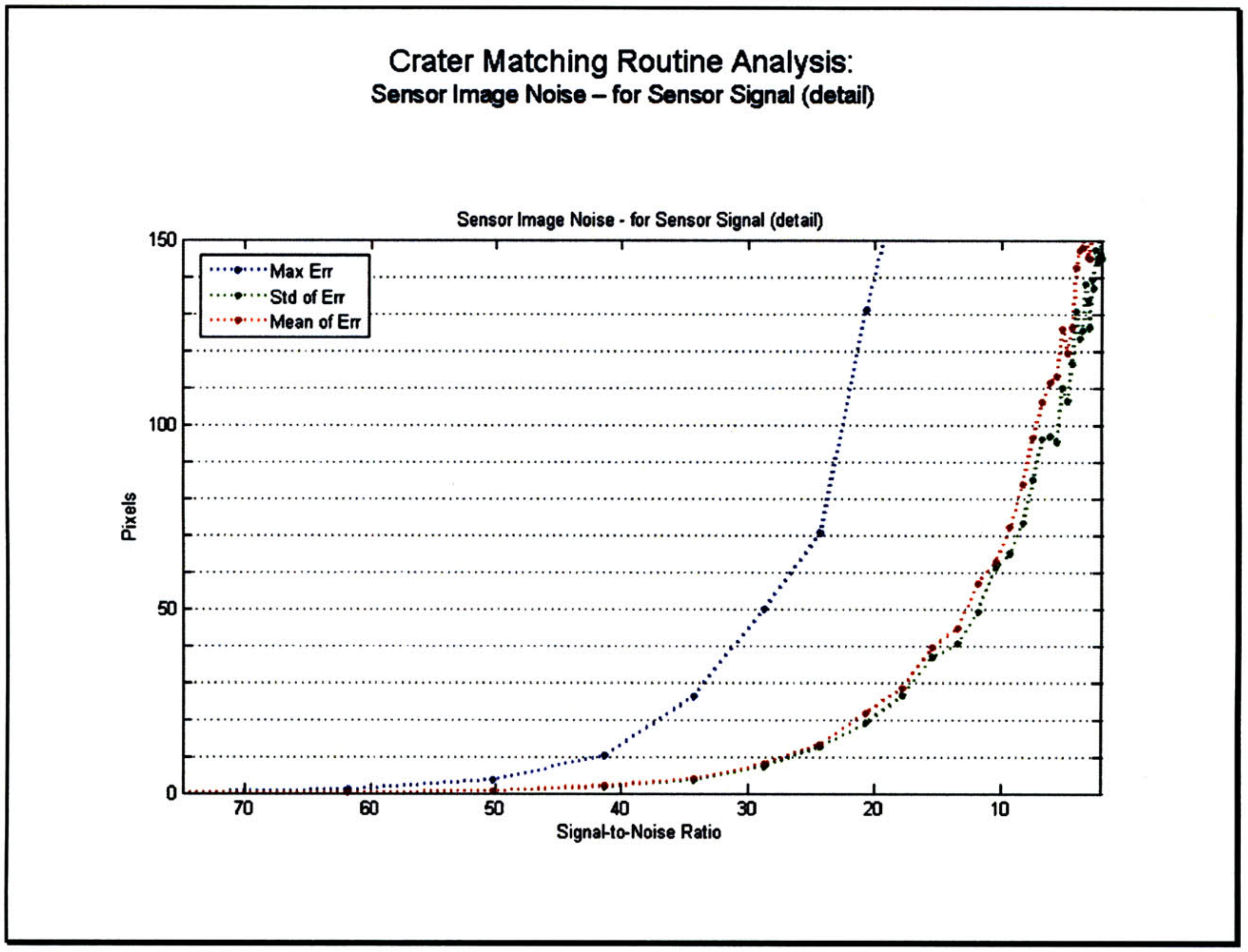


Figure 8.4[i]. Reprint Analysis of the Routine: Signal-to-Noise Ration and Error in Image Pixels (detail).

Right around a signal-to-noise ratio of 50 the routine's general tendency to generate correct position matches turns into a general tendency to generate inconclusive position matches. This signal-to-noise ration correlates to a 1 pixel mean, 1 pixel standard deviation in the position of the crater center. These results demonstrate that the routine can determine a position fix even in the presence of significant amounts of noise.

[i] Also Figure 7.5

8.2 Observations

The author has made several observations based upon the analysis of the TERCOM algorithm and the crater matching routine. Some of these comments will be presented below. The author's observations are presented in greater detail in Chapter 2 and Chapter 7.

8.2.1 TERCOM

It appears that TERCOM can produce correct position fixes – but only on a very limited number of cratered surfaces. The algorithm tends to generate lower overall errors as the 'roughness' of a map increases. However, the roughness of the terrain does not necessarily make that terrain unique, which is required for the generation of an effective position fix. The fix error also decreases as the density of craters in a given position fix area increases. In addition, initial results indicate that the TERCOM algorithm does not work well in the absence of large craters. Even with low noise levels, a lack of large, deep craters causes an unacceptably high level of position fix error.

The initial performance of the TERCOM algorithm is disappointing. Any position fix generated by the TERCOM algorithm for lunar TRN must have a very low probability of a false position fix. For all tested cases of the algorithm, false position fixes were generated, many with significant amounts of error. Even with very low amounts of sensor and map noise, false matches were generated. In addition, the maximum achievable accuracy for TERCOM is the dimension of a terrain profile cell, which was $100\ m$ for these analyses. In order to achieve precision landing, the position of the spacecraft must be known much more accurately than $100\ m$. Greater accuracy will require that the reference matrix cells each cover a smaller surface area of the Moon.[1] However, the high frequency

[1] Sub-pixel interpolation of the match might allow for greater-than-cell-size accuracy. However, the decision to investigate other correlation algorithms is rooted in the high frequency of false position fix for TERCOM, not in the achievable accuracy.

of false fix greatly outweighed the limits on the achievable accuracy in the analysis of TERCOM. TERCOM's position fix error is unacceptable for the purposes of lunar navigation – and the magnitude of the error implies that it is not due to the reference map, but primarily the result of false correlation.

The TERCOM algorithm did yield several positive results. Chief among these was the successful implementation of the VAR algorithm. Although both the MAD and VAR[1] algorithms perform poorly under lunar TRN conditions, the VAR algorithm does allow TERCOM to function without a barometric altitude measurement. On most given sets of data, the results of the VAR algorithm nearly match the results from the MAD algorithm. This indicates that the VAR algorithm functions properly as an alternative to the MAD algorithm for the lunar case – and potentially for other uses.

Analysis demonstrates that TERCOM is not robust enough to be able to generate position matches in the variety of conditions that will be encountered on the lunar surface. Combined with the high frequency of false position fixes for the initial TERCOM correlation attempts, the author decided that other TRN methods are more appropriate for lunar EDL purposes.

8.2.2 The Crater Matching Routine

The overall implementation of the crater matching routine was very successful. The results indicate that the grid-based pattern vector is an effective means of 'tagging' a crater for future identification. In addition, the database size was not prohibitive, the processing time was reasonable, and the achieved reliability promising.

More specifically, the author was very pleased with the processing speed of the algorithm. The routine is programmed in an engineering analysis computer language, which is not designed to execute as quickly as operational code. There are also many extra features coded into the current

[1] The MAD algorithm is the Mean Absolute Difference algorithm. The VAR algorithm is the Variance Algorithm. Both are algorithms that are used by the TERCOM process.

routine that will be removed for an operational implementation. Therefore, the ability to process sensor images with 18 potential boresight craters and about 140 total craters in less than 3 seconds is a significant accomplishment.

The correlation between the current program size and the (eventual) operational program size is difficult to gauge. Currently, the crater matching program files are approximately 2400 lines of code and occupy $100KB$ on the computer's hard drive. The graphing files are approximately 550 lines of code and occupy $22.5KB$ of memory. In total, the crater matching routine as currently coded is almost 3000 lines of code and requires about $125KB$ of memory for storage. The database is saved as a look-up table and is $1.66MB$ in size.[1] Considering the magnitude of the crater matching task, the size of the crater matching routine is within acceptable bounds.

The routine proved able to generate correct position matches for images with signal-to-noise ratios (SNR) as low as 50. For SNRs higher than 75, there were no false position matches, meeting the requirement for less than 0.1% false matches. For SNRs higher than 50, there were around 85% correct position matches, meeting the requirement for at least 75% correct position matches. While the frequency of correct matches declines quickly at SNRs below 50, the frequency of incorrect matches remains relatively low at $3-6\%$ for these low SNRs. The remaining position match attempts were inconclusive. The fact that false position matches are rarely generated is crucial to the possibility of using the crater matching routine for lunar TRN. In addition, the modification of the user-defined correlation factors will allow the crater matching routine to be prepared for situations with either a higher or a lower level of noise. These same correlation factors also enable the crater matching routine to be proficient at generating a positive position match when there are craters in the

[1] This simulated database stores 1000 images of the lunar surface. This corresponds to over 30 million square kilometers of the lunar surface with the given initial conditions.

image that were not in the database.[1] The initial analysis of the crater matching routine's response to error in the sensor image demonstrates that the routine is robust to significant levels of noise, and still produces accurate results.

The author was unable to find published results from any similar crater matching algorithm. Attempts to compare this crater matching routine's results with analogous results from published star tracker results, such as those presented by Padgett, Kreutz-Delgado, and Udomkesmalee[27], proved not to be useful.

The analyses presented herein demonstrate that the crater matching routine is a functional technique for generating a position correlation from an image of the lunar surface. This method merits further development and analysis to determine if it is both suitable and better than other crater matching methods. If it is suitable and better than other methods, this crater matching method may be appropriate to implement on hardware for further lunar TRN application testing.

8.3 Recommendations for Future Work

The author's recommendations for future work will be presented below. These recommendations are intended to suggest possible areas for improvement upon and refinement of the ideas detailed in this thesis. They do not encompass all possibilities for improvement; rather, they are intended to suggest several topics which the author believes might be of benefit in understanding and optimizing the correlation algorithms.

[1] This situation might be encountered due to the frequent meteor strikes on the lunar surface. Since the database is based off of lunar imagery, the time between the images collected for database formation and those taken by the landing craft's sensor may demonstrate some of the effects of these meteor strikes in the form of 'extra' craters in the images.

8.3.1 TERCOM

Although the current level of analysis shows that TERCOM is not a very robust or accurate approach to lunar navigation, more analysis is recommended to validate these conclusions. These conclusions are based upon a relatively small amount of analysis. Additional analyses are likely to confirm the author's decision to look to other methods for determining a position update for the spacecraft's inertial navigation system. However, although the author thinks it unlikely, it is possible that TERCOM may be proven more robust than it initially appears. If the robustness and the reliability of TERCOM are improved, the algorithm may be suitable as a lunar navigation technique.

To continue the analysis of TERCOM, several of the simplifying assumptions in this analysis should be addressed in future work. These include:

1. Constant Altitude – the assumption that the spacecraft will maintain constant altitude over the reference matrix may not be true, depending upon the stage in the EDL process.
2. Cross-Track Velocity Knowledge – the algorithm assumes that the spacecraft has perfect knowledge of its cross-track velocity. However, if there is even a slight skewing caused by unaccounted or unknown cross-track velocity, the entire matching may fail because the terrain profile includes data from two or more reference matrix columns.
3. Down-Track Velocity Knowledge – the algorithm also assumes perfect down-track velocity knowledge. If this knowledge is not perfect, the size of the terrain profile cells will not properly match with those of the reference matrix, and correlation errors may result.
4. Attitude Knowledge – errors in the spacecraft's attitude knowledge may cause errors in recording the terrain profile properly.

These simplifying assumptions should be considered and addresses appropriately. At the least, analysis to determine the probable effects of these assumptions is warranted.

The error sources that contribute to the sensor error should be explicitly identified and simulated as well. These error sources include:

5. Horizontal velocity errors (down range & cross track)
6. Vertical velocity errors
7. Attitude errors
8. Errors due to insufficient cell sampling
9. Relative vertical measurement error
10. Position knowledge error

Modeling these sensor error factors will enable the TERCOM simulation to more accurately represent the conditions that are likely to be encountered in lunar TRN.

To more accurately simulate the lunar surface environment, terrains that include underlying features such as sloped surfaces and hills should also be included in the analysis. It is expected that the overall TERCOM error will decrease when such underlying surface features are included. This expectation should be confirmed with analytic results.

The simulation should also be expanded to include the landing vehicle dynamics in the following manners:

11. Simulated trajectories of the landing vehicle over a fix area or set of fix areas.
12. Closed loop simulation of the landing vehicle over a fix area or set of fix areas.

These and other modifications to the TERCOM correlation algorithms may make the algorithm more robust to the lunar TRN environment. If modification does allow the TERCOM algorithm to perform well under conditions representative of those expected in lunar EDL, it may be appropriate to reconsider the algorithm for lunar landing. Otherwise, TERCOM is not the best choice for lunar TRN.

8.3.2 The Crater Matching Routine

The implementation and analysis of the crater matching routine suggest that the routine may be appropriate for further development and analysis. The topics presented below are the author's recommendations for where future efforts may be of the greatest import in regards to the routine.

8.3.2.1 Required Improvements for Operability

In order for the routine to operate with full functionality, these conditions must be considered and resolved:

13. Two Craters in Same Grid Cell.

    The current implementation of the crater matching routine does not address what occurs when two crater centers are located in the same grid cell. In order for the pattern vector to incorporate all available data, all craters within the same grid cells should be included in the pattern vector. A method for accommodating this occurrence should be developed and implemented.

14. Closest Neighboring Crater Misidentification

    The determination of a pattern vector is highly dependent upon the identification of the closest neighboring crater. If the wrong crater is selected as the closest neighbor (for whatever reason), or if the closest neighboring crater is significantly impacted by sensor noise, the rotation of the image will be incorrect and all of the craters in the image will be rotated to the wrong position for pattern vector determination. If the offset of the closest neighbor is great enough, the resultant pattern vector could be entirely incorrect.

    The author sees several potential methods to correct this situation:

a. Use Polar Coordinates

If the entire pattern vector generation process is accomplished in polar coordinates, rotational error is much simpler to counteract. However, this would require the entire pattern vector determination to be reconfigured.

b. Generate Pattern Vectors for Second Closest Neighbor, etc.

If the closest neighbor is overlooked, the pattern vector determination will select the second closest neighbor and determine a pattern based upon that crater. To counteract the misidentification of the closest neighbor, the database could include pattern vectors for the second closest neighboring crater for all craters. If desired, the third closest neighboring crater pattern vector might be included as well, etc. This process will greatly increase the size of the database. In addition, this process will not account for the situation where sensor noise significantly affects the closest neighboring crater and causes improper rotation. The routine is already programmed to allow the second (or third, etc.) closest neighboring crater to be selected. The closest neighbor number m allows the user to specify which neighboring crater to use as the rotational reference point.

c. Reference Boresight Crater – Closest Neighbor Pairs.

The correlation technique might impose the requirement that the radius of the boresight crater and the radius of the closest neighboring crater must match those of the pattern vector to which they match. This would effectively set up another filter that discards any pattern vector match between the image and the database where the boresight and closest neighboring crater radii of the image do not match those of the same craters from the database. This method would account for the selection of the wrong closest neighboring crater, but would fail to account for the situation where sensor noise significantly impacts the position of the closest neighboring crater.

15. Crater Visibility at Altitude

    Depending upon the altitude of the spacecraft, different numbers of craters are visible to the camera. A method to determine which craters from the database will be 'seen', depending upon the altitude of the spacecraft, must be developed and implemented.

16. Altitude Independency

    The current implementation of the crater matching routine does not account for any spacecraft altitude variations. The crater database was specifically generated for spacecraft at 100 km and all the analyses conducted were based upon this altitude. However, the lunar EDL environment will require the routine to function at all altitudes that might be encountered during EDL. Therefore, a means of making the matching routine independent of altitude is necessary.

    The author sees several methods that might allow this functionality for the algorithm

    d. Generate Multiple Databases

       If databases are generated and are applicable only for a range of spacecraft altitudes, a series of databases could be generated to cover the range of all altitudes that are expected to be encountered. This would also allow a simple method to vary the number of craters expected to be visible at different altitudes. However, it would greatly increase the size of the database storage requirement.

e. Scale the Data to Selected Closest Neighbor Distance

The data might be scaled so that the closest neighboring crater center is always a certain distance from the desired boresight crater. This would force all data sets into the same relative scale and allow the crater matching to function for all altitudes.

f. Use Radius Scaling with Polar Coordinates

If polar coordinates are used in pattern vector generation, the radius associated with each crater position might be scaled based upon the altitude of the spacecraft, which is known via sensor measurement. This scaling would allow the matching process to function over all altitudes.

17. Attitude Independence

As currently implemented, the crater matching routine assumes that the spacecraft has perfect attitude. An operational version of the routine must account for the possibility of attitude variations of the spacecraft. These attitude variations are most likely able to be accounted for by modifying the perspective transform to include attitude terms.

8.3.2.2 Translation of the Routine into Operational Code[i]

There are several tasks that will be required to translate the routine into operation code. These include:

18. Optimize the Routine

The crater matching routine is not currently programmed to run efficiently. Rather, the code is analytic, intended to allow the author to analyze the routine's applicability to lunar TRN.

[i] The author uses the term 'operational code' to imply code that is neither analytic code nor flight code. Rather, 'operational code' is code that is programmed for testing and is intended to be very close to a flight version of the code, allowing a better understanding of the code to be gained during testing.

To more fully characterize the routine, the code must be stripped of much of the analytic functionality. In addition the user defined constants must be analyzed and optimized for the lunar environment.

19. Program Routine into Operational Code

    The routine must be programmed in operational code that will allow the performance of the routine to be more precisely understood.

20. Incorporate Routine into an End-to-End Simulation

    The operational code can be integrated into and end-to-end simulation to allow overall performance analysis of the navigation update process. This simulation will demonstrated the effectiveness of the navigation update process in allowing the inertial guidance system of the spacecraft to help keep the spacecraft on its desired trajectory.

8.3.2.3 Possible Modifications to the Routine

There are many possible modifications to the routine that might improve its performance. Some of these include:

21. Program a Baseline

    It may be appropriate to program a triangle-based star tracker algorithm as a crater matching method. This algorithm would be a potentially useful baseline for performance comparison. In general, triangle-based approaches are the standard for the 'lost-in-space' algorithm of star trackers. Reference section 3.2.2.2.1.

22. Probabilistic Modeling

The incorporation of some type of probabilistic modeling into the matching process might prove useful in the correlation results. As currently programmed, the pattern vector indicates if a grid cell is 'full' or 'empty'. Rather than each cell containing merely a 0 or a 1 (effectively) indicating the presence of a crater, crater centers might be modeled as some type of mean distribution. This would allow the cells that are very near a 'full' cell to indicate such. Therefore, if image noise causes a crater center to appear shifted by a cell or so, the probabilistic modeling might account for that shift.

23. Confidence Measures

A measure of confidence attached to each position match would be helpful. This confidence measure would need to incorporate all factors that indicate how closely a given image matches the correlated area of the database.

24. Investigate Other Matching Possibilities

There are many other possibilities for the crater matching task. Some of these may perform better than this crater matching task. These other possibilities should be considered and investigated as appropriate.

Some possibilities that were of interest to the author were a genetic algorithm and a neural network approach to solving the star tracker problem.[28] In addition, a pyramid scheme based upon the standard triangle-based star tracker algorithm is another method that might deserve more attention.[29]

25. Complete Analysis

The analysis of the routine was by no means exhaustive. Many of the areas that would benefit from more in-depth analysis were mentioned in Chapter 7. Most importantly, the response of the routine to various types and levels of noise should be more fully characterized.

26. Matching Techniques

The current matching technique is very basic. A more complex matching scheme may allow the correlation process to account for 'near matches' between grid cells of the pattern vector and the database, which does not currently occur.

27. Database Searching Techniques

The database is stored as a look up table, and is searched by referencing the cell number of 'full' cells. There may be other methods that are more suited to searching the database. One suggestion might be a modification of the k-vector approach for accessing the database as described by Martari, Junkins, and Samaan.[30]

Appendix A: Perspective Transform Proofs

During the course of the derivation and implementation of the perspective transform, the author found it useful to prove various traits of the algorithm. Two of these proofs lend particular insight to the nature of the transform and will be included here.

Proof # 1: Proof that the Normalized Distance Between the Original and Desired Principle Point does not Vary Due to the Perspective Transform.

This proof demonstrates that the distance between the principle point and the desired principle point in the original image will be equal to the distance between the same points in the transformed image. Pictorially, this is represented in Figure A-1. The proof itself can be broken into two steps:

28. Prove that the angle between the original and desired principle points, measured from the origin of the camera frame, is equal across the transform. That is, prove that $\alpha_1 = \alpha_2$.
29. Given $\alpha_1 = \alpha_2$, prove that the distance between the original and desired principle points is equal across the transform. In other words, prove that $d_1 = d_2$.

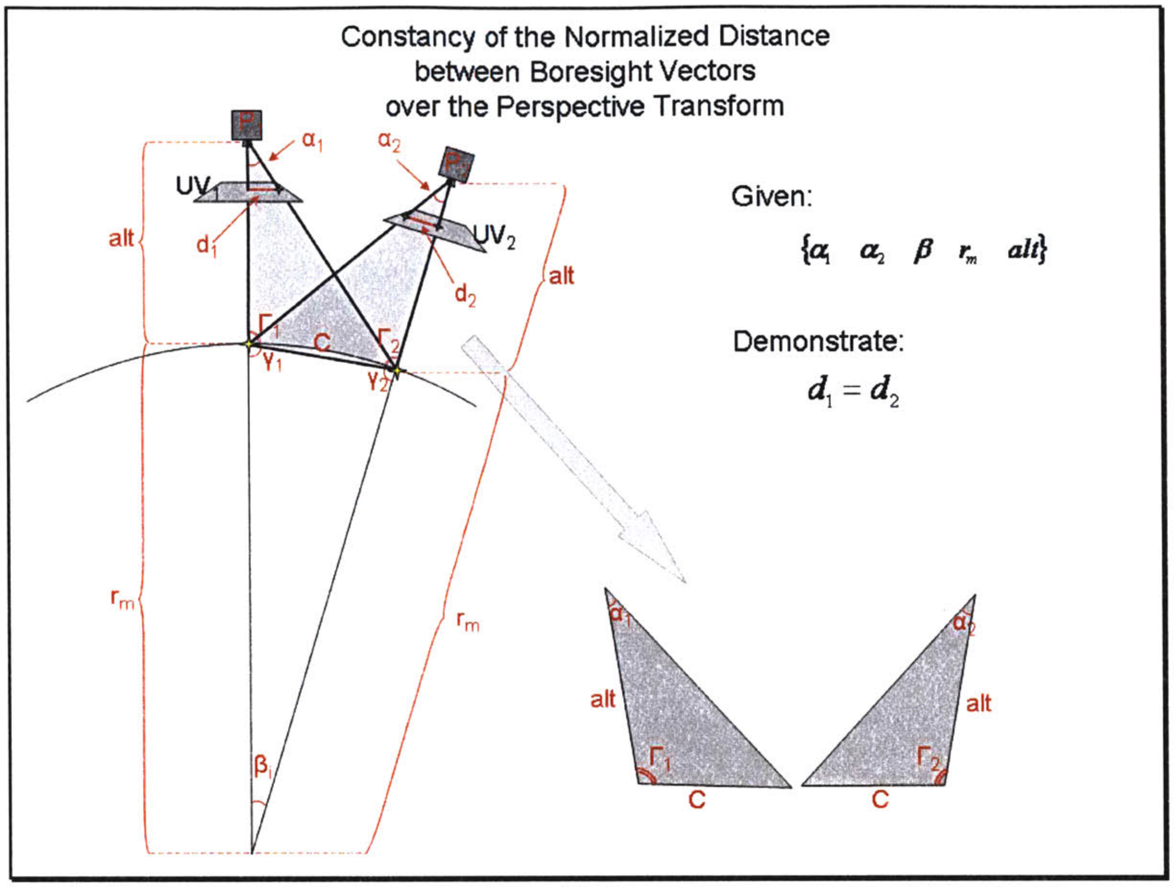


Figure A-1. Proof that the Normalized Distance Between Two Boresight Vectors is Constant Over the Transform.

Proof #1, Step A: Prove that $\alpha_1 = \alpha_2$.

30. By isosceles triangle with the included angle as β_1, it is evident that $\gamma_1 = \gamma_2 = \gamma$.

31. By adjacent exterior angles of this isosceles triangle, $\Gamma_1 = \Gamma_2 = \Gamma = \pi - \gamma$.

32. By side-angle-side, the highlighted triangles are congruent (and equal).

$SAS \Rightarrow \{alt \quad \Gamma \quad C\}$. See Figure A-1.

33. Since the highlighted triangles are congruent, all angles must be equal. Therefore, $\alpha_1 = \alpha_2$.

Proof #1, Step B: Demonstrate that if $\alpha_1 = \alpha_2$ then the distance between the current principle point and the desired principle point in the original image plane is equal to the same distance in the transformed image plane.

Since the image planes UV_1 and UV_2 are both defined as unit distance from the camera, the normalized distance between any two points on the image plane can be defined by Equation 5.2 as:

$$\sqrt{u_l^2 + v_l^2} = \tan(\alpha_l)$$

Equation 8.2

Reference Figure 5.6. In the first step of this proof, it was shown that $\alpha_1 = \alpha_2$. Therefore, it follows that $\tan(a_1) = \tan(a_2)$ and $\sqrt{u_1^2 + v_1^2} = \sqrt{u_2^2 + v_2^2}$, which implies that the normalized distance between the original principle point and the desired principle point in the image plane will not vary due to the transform.

Proof #2: Demonstration that the Norm of an Arbitrary Vector May Vary Over the Perspective Transform.

This proof is somewhat more difficult than the previous proof, as it attempts to prove that any arbitrary vector in the original image plane does not *necessarily* stay constant over the perspective transform. This demonstration will not show that the norm of an arbitrary vector must change over the transform; rather, it shows that the norm of an arbitrary vector will change over the transform for some case(s).[i]

Thus, a single example demonstrates that the statement is indeed true.

i Refer to section 5.7.4 for an example of the norm of an arbitrary vector remaining constant.

The counter-example presented below follows this logic. This method of counter-example is not considered a proof. Nevertheless, it does demonstrate that the statement is true.

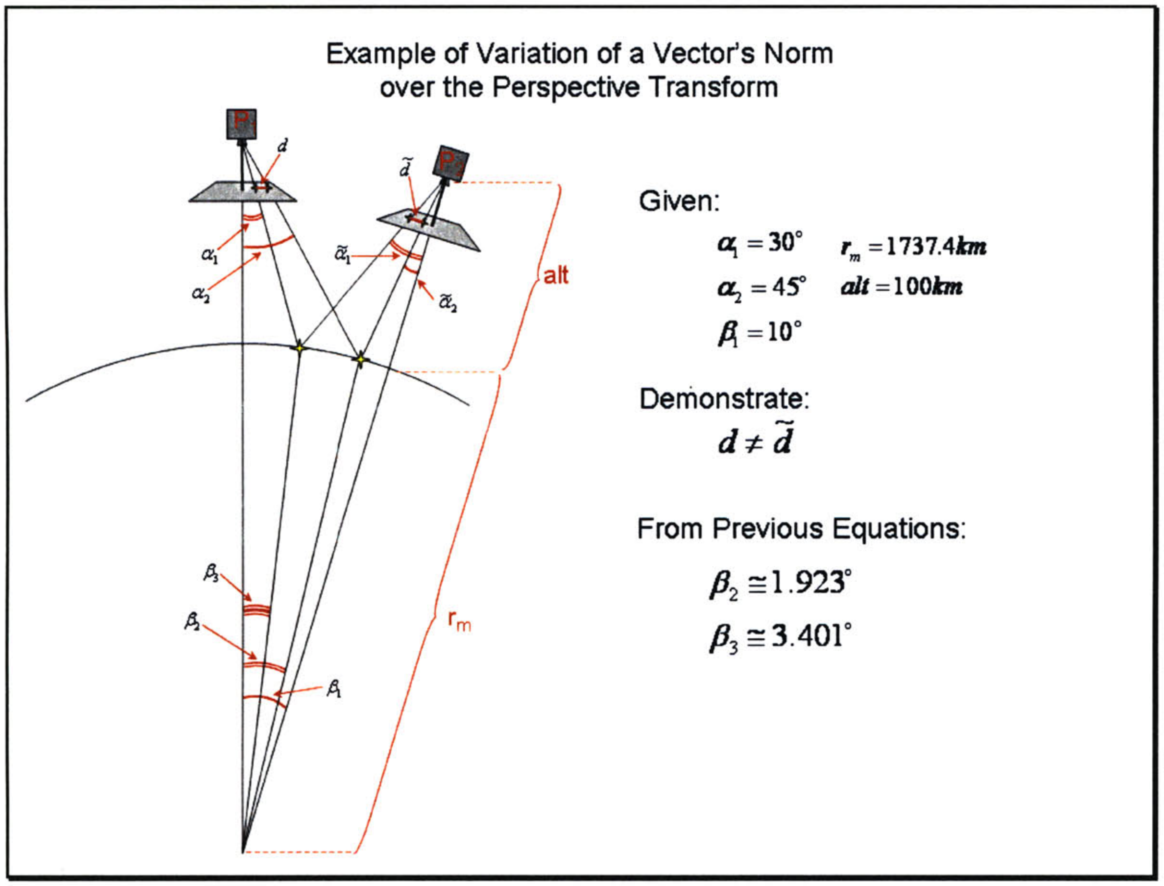

Example of Variation of a Vector's Norm over the Perspective Transform
P1
P2
d
$\tilde{d}$
α_1
α_2
$\tilde{\alpha}_1$
$\tilde{\alpha}_2$
alt
β_3
β_2
β_1
r_m
Given:
$\alpha_1 = 30^\circ$ $r_m = 1737.4km$
$\alpha_2 = 45^\circ$ $alt = 100km$
$\beta_1 = 10^\circ$
Demonstrate:
$d \neq \tilde{d}$
From Previous Equations:
$\beta_2 \cong 1.923^\circ$
$\beta_3 \cong 3.401^\circ$

Figure A-2. Demonstration that the Normalized Distance Between Any Two Points in the Image Plane is not Constant Over the Transform.

Proof #2: Example Problem Setup

Given:

$$\begin{array}{ll} \alpha_1 = 30^\circ & \\ \alpha_2 = 45^\circ & r_m = 1737.4km \\ & alt = 100km \\ \beta_1 = 10^\circ & \end{array}$$

Equation 8.3

Show that:

$$d \neq \tilde{d}$$

Equation 8.4

Proof #2: Example Problem Calculations

34. From **Equation 5.2**, **Equation 5.3**, and **Equation 5.4**.

$$\beta_2 \cong 1.923^\circ$$
$$\beta_3 \cong 3.401^\circ$$

Equation 8.5

35. From manipulation of **Equation 5.2**

$$d = \tan(\alpha_2) - \tan(\alpha_1)$$
$$\tilde{d} = \tan(\tilde{\alpha}_2) - \tan(\tilde{\alpha}_1)$$

Equation 8.6

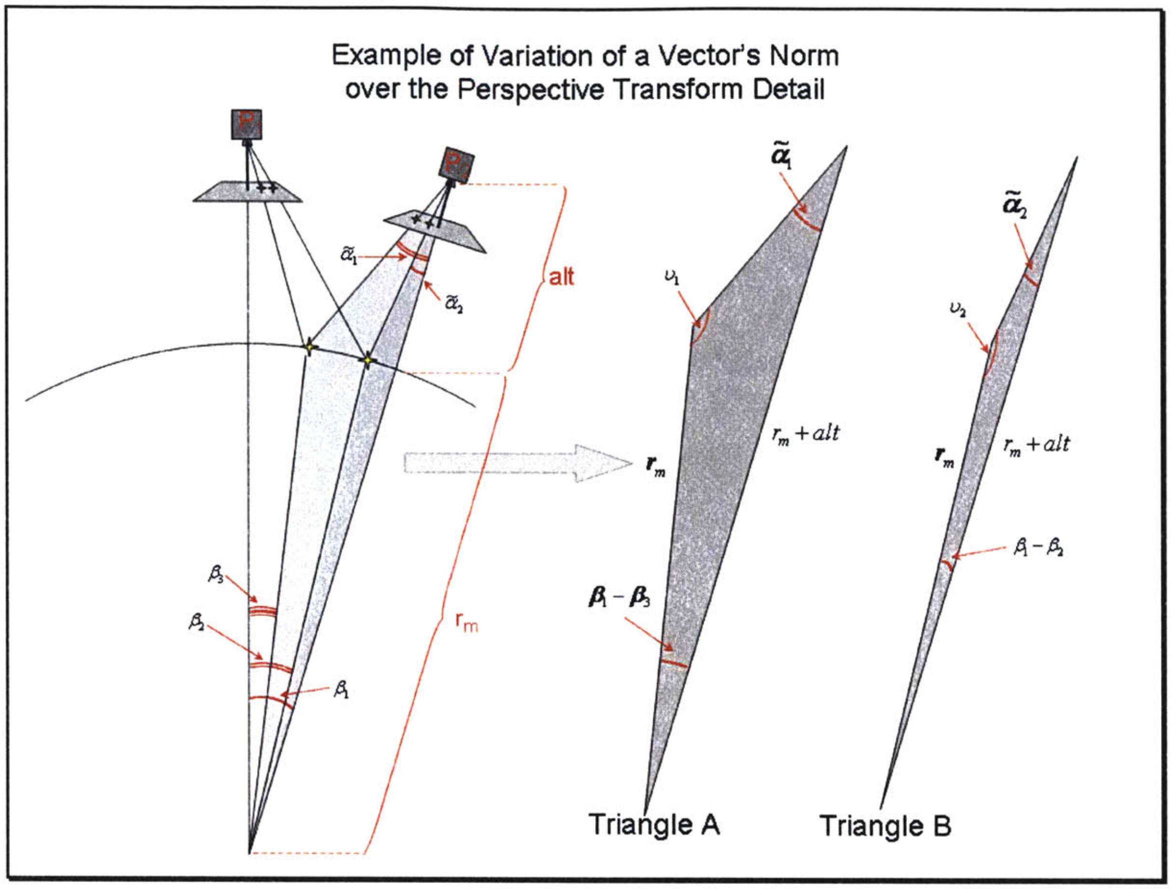


Figure A-3. Demonstration that the Normalized Distance Between Any Two Points in the Image Plane is not Constant Over the Transform, Detail.

36. Triangle A: Use the rules for interior angles of a triangle and the Law of Sines to solve for $\tilde{\alpha}_1$ in terms of known quantities. Refer to Figure A-3.

$$\upsilon_1 = \pi - (\beta_1 - \beta_3) - \tilde{\alpha}_1$$

Equation 8.7

$$\frac{\sin(\tilde{\alpha}_1)}{r_m} = \frac{\sin(\upsilon_1)}{r_m + alt}$$

Equation 8.8

$$\frac{\sin(\tilde{\alpha}_1)}{r_m} = \frac{\sin(\pi - (\beta_1 - \beta_3) - \tilde{\alpha}_1)}{r_m + alt} = \frac{\sin(\beta_1 + \beta_3 + \tilde{\alpha}_1)}{r_m + alt}$$

$$= \frac{\sin(\beta_1 + \beta_3)\cos(\tilde{\alpha}_1) + \cos(\beta_1 + \beta_3)\sin(\tilde{\alpha}_1)}{r_m + alt}$$

$$\frac{r_m + alt}{r_m} = \frac{\sin(\beta_1 + \beta_3)\cos(\tilde{\alpha}_1) + \cos(\beta_1 + \beta_3)\sin(\tilde{\alpha}_1)}{\sin(\tilde{\alpha}_1)}$$

$$= \frac{\sin(\beta_1 + \beta_3)}{\tan(\tilde{\alpha}_1)} + \cos(\beta_1 + \beta_3)$$

Equation 8.9

$$\tan(\tilde{\alpha}_1) = \frac{\sin(\beta_1 + \beta_3)}{\frac{r_m + alt}{r_m} - \cos(\beta_1 + \beta_3)}, \quad \left\{\tilde{\alpha}_1 : \tilde{\alpha}_1 \in \left(0 \quad \frac{\pi}{2}\right)\right\}$$

Equation 8.10

Substituting known values into Equation 8.10

$$\tilde{\alpha}_1 \cong 69.042^\circ$$

Equation 8.11

37. Triangle B: Use the same process to solve for $\tilde{\alpha}_2$ in terms of known quantities.

$$\upsilon_2 = \pi - (\beta_1 - \beta_2) - \tilde{\alpha}_2$$

Equation 8.12

$$\tan(\tilde{\alpha}_2) = \frac{\sin(\beta_1 + \beta_2)}{\frac{r_m + alt}{r_m} - \cos(\beta_1 + \beta_2)}, \quad \left\{\tilde{\alpha}_2 : \tilde{\alpha}_2 \in \left(0 \quad \frac{\pi}{2}\right)\right\}$$

Equation 8.13

$$\tilde{\alpha}_2 \cong 69.906^\circ$$

Equation 8.14

38. Use Equation 8.6 to determine d and $\tilde{d}$.

$$d = 0.423$$
$$\tilde{d} = 0.123$$

Equation 8.15

This demonstrates that d is not necessarily equal to $\tilde{d}$ when d is an arbitrary vector in the image plane. Therefore, this example shows that in general, the norm of a given vector will not necessarily remain constant over the perspective transform as defined in this paper.

Appendix B: Abbreviations and Acronyms

Table B-1. Acronyms.

Acronym	Meaning
AGL	Air-to-ground Level
CEP	Circular Error Probability
DSPSE	Deep Space Program Science Experiment
EDL	Entry, Descent, and Landing
GPS	Global Positioning System
INS	Inertial Navigation System
LIDAR	Light Detection and Ranging
LOLA	Lunar Orbiter Laser Altimeter
LRO	Lunar Reconnaissance Orbiter
LSSM	Long Sample Short Matrix
MAD	Mean Absolute Difference
MSL	Mean Sea Level
NASA	National Aeronautics and Space Administration
PDF	Probability Distribution Function
PTAN	Precision Terrain Aided Navigation
SSLM	Short Sample Long Matrix
ST9	Space Technology 9
TAINS	TERCOM Aided Inertial Navigation System
TERCOM	Terrain Contour Matching
TRL	Terrain Relative Localization
TRN	Terrain Relative Navigation
VAR	Variance
VBN	Vision Based Navigation
VSE	Vision for Space Exploration

Table B-2. Abbreviations.

Abbreviation	Meaning
B/c	Because
BR	Buffer Radius
BS	Boresight
Corr.	Correlation
DB	Database
Err.	Error
FOV	Field of View
Gen.	Generation
Inconcl.	Inconclusive
Max.	Maximum
OFP	Over Flight Path
Pos.	Position
PR	Pattern Radius
Pts.	Points
PV	Pattern Vector
Ref.	Reference
Sig.	Signature
SNR	Signal-to-Noise Ratio
Std.	Standard Deviation
Succ.	Success
Thresh.	Threshold
TP	Terrain Profile
w/*	With

Appendix C: Matlab Code for the Crater Matching Routine

The code presented in this appendix is the code used to implement all theory presented in this thesis, as well as used to generate all results presented as the author's own.

The code was written in MatLab v7.1 (R14) licensed to The Charles Stark Draper Laboratory. MatLab is a high-level computer language with an extensive array of built-in functions, and a wide variety of downloadable toolboxes with additional functions. The language is widely used in the astronautical engineering field (and other similar engineering fields) for initial implementation and testing of code. A wide variety of graphical and analytic tools are accessible in MatLab to simplify this process. However, MatLab code is not suitable for real-time purposes.

The author's code is highly function-based. To simplify the presentation of the code, an outline of the overall structure of the code is first presented, arranged by the order in which functions are called. Following this, the code for all functions is presented alphabetically.

Note that the top-level code files for the crater matching routine and the crater database generation are CM_Main.m and DB_Main.m, respectively.

Called Function Outline

<u>CM_Main.m</u>

1. CM_LoadParam.m
    1. Gen_DefaultAsk.m
        39. Gen_DefaultSet.m
2. Gen_InitialOps.m
    1. PT_RMat.m
3. Gen_ProcessImage.m
4. CM_Signature.m
    1. Gen_BSList.m
        40. PlotCraters.m
    2. PT_RTDCalc.m
        41. PT_DCalc.m
        42. PT_RTCalc.m
            g. PT_RMat.m
            h. PT_TVec.m
                1. PlotTrans.m
    3. PV_Main.m
        43. PT_Main.m
            a. PlotOrigCraters.m
                1. PlotCraters.m
            b. PT_Analysis.m
                1. PV_EuclideanTrans.m
                2. PT_PtsNorm.m
                3. PlotCraters.m
        44. PV_SensorView.m
            c. PlotCraters.m
            d. PlotOverlay.m
            e. PlotSensorView.m
        45. PV_CloseNeighbor.m
            f. PlotOverlay.m
        46. PV_PatternGen.m
            g. PlotPatternGen.m
                4. PlotCraters.m
                5. PlotOverlay.m
                6. PlotSensorView.m
                7. PlotGrid.m
                    i. PlotOverlay.m
        47. PV_Save.m
            h. Gen_CircleSquare.m
                8. Gen_CompressPV.m
    4. CM_Save.m
5. CM_Matching.m
6. CM_Analysis.m
7. CM_Analysis_Proc.m

<u>DB_Main.m</u>

1. DB_LoadParam.m
    1. Gen_DefaultAsk.m
        48. Gen_DefaultSet.m
2. Gen_InitialOps.m
    1. PT_RMat.m
3. Gen_CircleSquare.m
4. Gen_CompressPV.m
5. DB_CRMain.m[1]
6. Gen_ProcessImage.m
7. CM_Signature.m
    1. Gen_BSList.m
        49. PlotCraters.m
    2. PT_RTDCalc.m
        50. PT_DCalc.m
        51. PT_RTCalc.m
            i. PT_RMat.m
            j. PT_TVec.m
                1. PlotTrans.m
    3. PV_Main.m
        52. PT_Main.m
            k. PlotOrigCraters.m
                1. PlotCraters.m
            l. PT_Analysis.m
                1. PV_EuclideanTrans.m
                2. PT_PtsNorm.m
                3. PlotCraters.m
        53. PV_SensorView.m
            m. PlotCraters.m
            n. PlotOverlay.m
            o. PlotSensorView.m
        54. PV_CloseNeighbor.m
            p. PlotOverlay.m
        55. PV_PatternGen.m
            q. PlotPatternGen.m
                4. PlotCraters.m
                5. PlotOverlay.m
                6. PlotSensorView.m
                7. PlotGrid.m
                    i. PlotOverlay.m
        56. PV_Save.m
            r. Gen_CircleSquare.m
            s. Gen_CompressPV.m
    4. CM_Save.m
8. DB_Save.m

BSTranslator.m (obsolete, replaced by PT_Main.m)

[1] DB_CRMain.m is a program used by Draper Labs to simulate sensor images of the lunar surface given a camera orientation and position. In addition, this program identifies the crater centers and radii from the given image. This program will not be presented in detail herein.

Called Functions, Alphabetical

9. CM_Analysis.m

   This function contains functionality for automated analysis of the crater matching routine. It parses the success / failure / inconclusive and error readings for analysis.

10. CM_Analysis_Proc.m

    This function processes the results from CM_Analysis.m and displays and graphs the results automatically.

11. CM_LoadParam.m

    This function loads the initial parameters necessary for execution of the crater matching routine. It also will load default parameters if initialized to do so.

12. CM_Main.m

    The main function for the crater matching routine.

13. CM_Matching.m

    The correlation subroutine of the crater matching process, which attempts to match the signature to the database.

14. CM_Save.m

    This function saves the data from signature generation.

15. CM_Signature.m

    This function generates a signature for a given sensor image.

16. DB_LoadParam.m

    This function loads the necessary initialization parameters that are necessary for the database population.

17. DB_Main.m

    The main function for the database population

18. DB_Save.m

    This function saves pattern vector data to the database.

19. Gen_BSList.m

    This function generates a list of potential boresight craters given a sensor image.

20. Gen_CircleSquare.m

    This function is involved in numbering only those grid cells that have part of the cell inside of the pattern radius.

21. Gen_CompressPV.m

    This function is involved in numbering only those grid cells that have part of the cell inside of the pattern radius, and can either compress the pattern row vector or decompress the compressed row vector.

22. Gen_DefaultAsk.m

    This function asks the user if default initialization parameters should be used.

23. Gen_DefaultSet.m

    This function sets initialization parameters to default values, if requested by the user.

24. Gen_InitialOps.m

    This function performs initialization operations, such as defining constants and defining empty variables that must be predefined.

25. Gen_ProcessImage.m

    This function filters the original sensor image and converts it to a usable format. It also can add noise to the image if set to do so.

26. Other_LUTRowLengthGraph.m

    This function generates a plot of the row length of the look up table.

27. PlotCraters.m

    This function plots the craters from a given set of data points.

28. PlotGrid.m

    This function plots the full and empty grid cells, along with the grid lines themselves.

29. PlotOrigCraters.m

    This function plots the original sensor image craters.

30. PlotOverlay.m

    This function can plot the desired boresight crater, the potential boresight crater, the closest neighboring crater, the buffer radius, and the pattern radius. It can do so for images in different units as well.

31. PlotPatternGen.m

    This function plots the generation of the pattern from an image and the desired boresight.

32. PlotSensorView.m

    This function plots only the craters and portions of crater rims that fall within the pattern radius.

33. PlotTrans.m

    This function plots the translation vector calculation.

34. PT_Analysis.m

    This function is an analytic tool to measure the performance of the perspective transform.

35. PT_DCalc.m

    This function calculates the pointwise distance for the perspective transform.

36. PT_Main.m

    This is the main function for the perspective transform.

37. PT_PtsNorm.m

    This function calculates the normalized distance between every possible set of points in a matrix.

38. PT_RMat.m

    This function calculates the rotation matrix for the perspective transform.

39. PT_RTCalc.m

    This function serves as a header function for the calculation of the rotation matrix and translation matrix for the perspective transform.

40. PT_RTDCalc.m

    This function serves as a header function for the calculation of the rotation matrix, translation matrix, and the pointwise distance for the perspective transform.

41. PT_TVec.m

    This function calculates the translation vector for the perspective transform.

42. PV_CloseNeighbor.m

    This function determines the closest neighbor for a given image and desired boresight crater.

43. PV_EuclideanTrans.m

    This function performs the Eulcidean translation of data points to center a desired boresight.

44. PV_Main.m

    This is the main function for the patter vector determination.

45. PV_PatternGen.m

    This function rotates the image data and generates a pattern vector from the data points within the pattern radius.

46. PV_Save.m

    This function saves the pattern vector.

47. PV_SensorView.m

    This function determines which craters are within the pattern radius.

REFERENCES

[1] Apollo 11 Multimedia, Apollo 11 Image Library. http://www.hq.nasa.gov/alsj/a11/images11.html Updated 17 Sept 2006. Referenced 19 May 2007.

[2] Apollo 12 Multimedia, Apollo 12 Image Library. http://history.nasa.gov/alsj/a12/images12.html Updated 23 Nov 2006. Referenced 19 May 2007.

[3] "Mars Science Laboratory: The Next Wheels on Mars." http://www.space.com/businesstechnology/technology/mars_science_lab_050105.html Space.com. Updated 05 Jan 2005. Referenced 19 May 2007.

[4] Bjorn Jalving, Magne Mandt, and OveKent Hagen. "Terrain-Referenced Navigation of AUVs and Submarines Using Multibeam Echo Sounders." http://www.navlab.net/Publications/Terrain_Referenced_Navigation_of_AUVs_and_Submarines_Using_Multibeam_Echo_Sounders.pdf

[5] T. Baker and R.W. Clem. "Terrain Contour Matching (TERCOM) Primer", ASD-TR-77-61. Aeronautical Systems Division, Wright-Patterson AFB, Ohio, August 1977 (AD B021328).

[6] J.P. Golden. "Terrain Contour Matching (TERCOM) – A Cruise Missile Guidance Aid." SPIE Image Processing for Missile Guidance, vol 238, 1980. pp 10-18.

[7] Golden.

[8] Baker and Clem.

[9] G. Neukum, B.A. Ivanov, and W.K. Hatmann. "Cratering Records in the Inner Solar System in Relation to the Lunar Reference System", Chronology and Evolution of Mars, Kluwer Academic Publishers, Norwell, Mass, 2001. pp 53-86.

[10] Baker and Clem.

[11] Bruno Sinopoli, Mario Micheli, Gianluca Donato, and T. John Koo. "Vision Based Navigation for an Unmanned Aerial Vehicle," Proceedings of IEEE International Conference on Robotics and Automation, 2001, pp. 1757--1765. http://citeseer.ist.psu.edu/sinopoli01vision.html

[12] National Aeronautics and Space Administration. Mars Pathfinder Mission. http://www.nasa.gov/mission_pages/mars-pathfinder/index.html

[13] National Aeronautics and Space Administration. Mars Exploration Rovers. http://www.nasa.gov/mission_pages/mer/index.html

[14] National Aeronautics and Space Administration. New Millenium Program. http://nmp.jpl.nasa.gov/index.html

[15] "Smooth Landings." Jet Propulsion Laboratory, Robotics Division. California Institute of Technology. http://www-robotics.jpl.nasa.gov/news/newsStory.cfm?NewsID=66

[16] Padgett, Kreutz-Delgado, and Udomkesmalee. "Evaluation of Star Identification Techniques." AIAA Journal of Guidance, Control, and Dynamics. vol. 20, no. 2, Mar-Apr 1997.

[17] D. Mortari , J. Junkins, and M. Samaan. "Lost-in-Space Algorithm for Robust Star Pattern Recognition." 24th Annual AAS Guidance and Control Conference, Breckenridge CO. 31 Jan – 04 Feb 2001.

[18] D. Mortari and B. Neta. "*k*--vector Range Searching Techniques." Paper AAS 00-128 of the 10th Annual AIAA/AAS Space Flight Mechanics Meeting, Clearwaters, FL, Jan 23-26, 2000.

[19] Ibid

[20] Padgett and Kreutz-Delgado.

[21] David A. Vallado. Fundamentals of Astrodynamics and Applications. 2nd Ed. El Segunda CA, Microcosm Press, 2001. pp 906.

[22] Mark A. Wiczorek. http://en.wikipedia.org/wiki/Image:MoonTopoGeoidUSGS.jpg 16 Nov 2006. Licensed for free use under Creative Commons Attribution 2.5. http://creativecommons.org/licenses/by/2.5/

[23] Hartley and Zisserman.

[24] Jerry Jon Sellers. Understanding Space: An Introduction to Astronautics. 2nd Ed. McGrw-Hill Co Inc., 2000. pg 735.

[25] Padgett, Kreutz-Delgado, and Udomkesmalee.

[26] Sellers.

[27] Padgett, Kreutz-Delgado, and Udomkesmalee.

[28] L. Paladugu, M. Schoen, and B. Williams. "Intelligent Techniques for Star-Pattern Recognition." IMECE 2003. November 16-21 2003, Washington D.C.

[29] Mortari and Neta.

[30] Mortari, Junkins, and Samaan.