Autonomous Robotic Exploration of Unknown Environments

R. Gartshore

Autonomous Robotic Exploration of Unknown Environments

R. Gartshore

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey, GU2 7XH
United Kingdom

September 2005

Summary

This thesis presents a unique exploration strategy for a mobile robot moving in a two-dimensional plane. The task of the exploration algorithm is to move around autonomously in an unknown room environment of a limited size whilst building and using a 2D map to avoid obstacles and plan its next move. The strategy is not target-driven, rather seeking new unseen areas to view and explore.

The novelty of the presented strategy is the use of a view-improvement technique along with an optimal viewpoint planning method for the calculation and selection of the next-best-viewpoint. Results for various environment situations are discussed, for a system with a limited field-of-view, with an application to a mobile robot with a single camera.

Key words: Autonomous Exploration, Viewpoint Planning, Mobile Robotics.

Email: R.Gartshore@surrey.ac.uk

WWW: http://www.surrey.ac.uk/

Acknowledgements

Thanks to to my supervisors, in order of appearance: Phil McLauchlan, Alberto Aguado, John Illingworth and especially Phil Palmer for the initial idea concept and the final suggestions to finish off the work.

Thanks also go to friends and colleagues at CVSSP and my family for their continued support throughout the duration. Special thanks to Josef Kittler, Phil Jackson, Charles Galambos, and Matt Deighton, for pushing me through to the end, but not over the edge.

Contents

List of Figures

Chapter 1

Introduction

The science-fiction film industry has, through the years, depicted robots as humanoid machines capable of carrying out a multitude of tasks, autonomously and without guidance from any human operator. The vast following of television series, feature films and books has inspired human imagination to idealise how a robot should be able to perform tasks which may aid us in our survival, help us with daily tasks like cleaning, reduce labour intensive jobs in the manufacturing industry, and even reduce the life-threatening jobs in times of war.

Tasks we would prefer robots to perform, be it to improve our lifestyle or to develop our understanding and knowledge of our ever expanding world, have some things in common:

- the ability to sense the world around
- to interpret the data received by the senses
- to act upon interpreted knowledge to perform the task at hand
- and probably at some stage, to communicate the acquired knowledge or outcome of the task to humans

In safe environments on Earth, tasks such as explore and collect information are relatively simple for humans to perform. However for more dangerous parts of the Earth such as the Chernobyl disaster site, it is necessary for robots to be employed to take on the role of a human. It is costly to send a man to Mars, but the same task is still needed. In these environments, it becomes impractical for human operators to define each and every move a mobile robot is to take in order to manoeuvre itself within the environment. The interpretation of the immediate environment by a human operator also delays the time taken for the robot to perform its task, and to move itself around the environment.

1.1 Background and Issues

The expanding area of mobile robotics encompasses a vast range of research areas for environments from indoor and outdoor terrestrial to the more extreme planetary, underwater, arctic and nuclear

fall-out etc. For such a diverse range of environments, emerging new hardware technologies to allow mobile robots to survive and carry out their tasks in difficult terrains and harsh environments. This incorporates research into the materials able to cope in extreme conditions of temperature and pressure variations as well as the basic requirement for the ability of the mobile robot to move around the environment. See for example [Patel et al., 2002] who compare the ability for wheeled and tracked vehicles to traverse the Mars landscape. To find solutions to some of these problem environments, recent research has focused on biologically inspired systems ([Trullier et al., 1997] presents a review of such systems) to achieve specific tasks which in turn has expanded the research area further into emergent behaviour from multiple agents ([Polesel et al., 2000] consider robot football) as well as cooperative robots ([Chaimowicz et al., 2003] consider hybrid systems in a collaborative robot system). This work will focus on single robot systems, although the algorithms described could be considered for multiple agent systems.

The ability for a robot to autonomously explore an environment is an area of research that has recently not received a lot of attention. Several algorithms exist for the interim stages of exploration: environment data acquisition, storage and interpretation, and point-to-point navigation. Several algorithms exist that can map the environment from the directions of a human operator, e.g. [Huang and Beevers, 2005]. However, none exist which start with no explicit knowledge of the number, position, or size of the obstacles in its environment, or the size of the environment itself, and enable a mobile agent to safely and more intelligently explore the environment without any human operator input during its execution. This problem is addressed in this thesis.

Mobile robotic exploration research began with Victor Klee's "Art Gallery Question" in 1973. Historically, Václav Chvátal responded with his "Art Gallery Theorem" in 1975, and later Stephen Fisk published a short proof of Chvátal's Theorem in 1978 (a full discussion can be found in [O'Rourke, 1994]). This geometric problem was extended to encompass the problem of moving a point-sized object around a two-dimensional environment. We take for granted the processing speeds available to us today, but this was a severe limitation on what could be achieved at that time. The problem of moving an object around an environment originally split into two main areas of research: for *a priori* known, and *a priori* unknown environments. For unknown environments, exhaustive exploration algorithms were developed where each portion of the environment was necessarily visited in order to build up information of what obstacles existed. The sensors used on mobile robots at that time were mainly tactile sensors, which only allow an obstacle to be detected when the robot collides with it. Extensive studies have been performed about the problem of moving a robot through environments that are initially fully known. Such optimal methods to move a robot from a starting position to an identified target position using say shortest route, or minimum number of steps are discussed in section 1.1.1. A basic extension to these problems take the size and shape of the robot into account, and solutions can still be found geometrically.

As mobile robots became more affordable and widely available, the research area has developed for a mobile platform capable of interacting with a real-world environment. This opened up a new area of research into the analysis of the errors encountered when dealing with real world situations. Errors

in the positions of obstacles, and in the position of the robot as it moves, were found to be related, [Ayache and Faugeras, 1989], and could be constructively used to reduce the error in the information of not only the positions of the obstacles in the scene but the position of the robot in relation to them.

Before the processing power of computers on-board a robot platform became more powerful in terms of processing speed, the sensor data captured by the robot's sensors would be stored and processed after the end of the acquisition stage. Nowadays the processing of the data and its interpretation can be done during the acquisition stages, and algorithms to simultaneously map the environment and correct for the position of the robot (simultaneous localisation and mapping, SLAM) have become a large area for research to this day (a survey of current approaches is presented by [Thrun, 2002]). The ability for a robot to incrementally build a map of its environment also allows decisions to be made about where the robot should move to next, based on current world information. This is an essential part of a system for autonomous exploration in unknown environments.

[Brooks, 1991] questioned the need for a representation of the information acquired from the scene, and proposed that by separating the goal of a system from the lower level procedures, such as obstacle avoidance, a simple yet intelligent system can be acheived. Brooks discussed the use of implicit versus explicit knowledge (where explicit means to represent the information in a specific form and implicit is to represent information in an abstract form, where the information is learnt without a known structure).

The state of the art techniques used for the interim stages of robot exploration are discussed below. The early work of navigation in *a priori* known and unknown terrains is considered first, followed by sensor placement strategies and behaviour based approaches to systems that have the ability to explore environments. The problem of SLAM is discussed followed by currently used exploration techniques. The problem addressed in this thesis will then be considered followed by the research approach taken to solve this problem; followed by an outline of the thesis layout along with terms that are frequently used throughout this work.

1.1.1 Navigation in known terrains

In an environment where the positions of obstacles in relation to the position of the robot are known, several methods to find a safe path for a robot to safely traverse, or navigate, have been proposed. One method is a Voronoi Graph, a connected line graph where each line is defined by the equal distances to the nearest obstacles, as used by [Kirkpatrick, 1979], [O'Dunlaing and Yap, 1985] and [Nagatini and Choset, 1999]. The safe path for the robot to take between obstacles is chosen from the Voronoi graph and the derived path is piecewise linear. As a complement to the Voronoi graph, [Guibas and Stolfi, 1985] use a Delaunay triangulation of the obstacle map and restrict the robot motion to the direct paths between triangle centres. This has a similar effect to the Voronoi graph by keeping the robot as far away from the obstacles as possible. [Briggs and Donald, 2000],

[Oommen et al., 1987] and [Rao et al., 1991] calculate the visibility of regions within the scene to construct a visibility graph (where respective graph points are visible from each other). These methods all reduce the calculation of the "best" path to a search algorithm such as depth-first, best first, A* (see [Russell and Norvig, 1995] for a description of these search algorithms), and more recently a dynamic-A* algorithm, D* [Stentz, 1995]. The above provide piecewise linear paths for navigation. To increase the overall speed of navigating through an environment, [Connolly et al., 1990] introduce the use of harmonic functions[1] to plan smooth paths, whereas [Khatib, 1986] discuss a local approach to using potential fields. Although it is possible to plan smooth paths through known obstacles, with the overall effect to increase the speed and improve the fluidity of movement through the scene, the more simple piecewise approach is adopted in the work presented in this thesis. This basic approach allows the evaluation of the environment to occur at the end of each piecewise motion, without consideration for the overall direction of the navigation.

Where Voronoi graphs are used for path-planning, the robot is kept as far away from obstacles as possible, therefore the risk of hitting an obstacle is minimised in the presence of errors in the obstacle position. Adding an element of uncertainty into the environment develops the path-planning problem from an off-line search problem to a problem that must be solved during the path-planning operation, since it cannot be assumed that the obstacles are in exactly the same position.

A basic extension to the point-sized navigation problems described above take the size and shape of the robot into account. The solution to these problems can still be found geometrically, and was studied by [Lozano-Pérez, 1983] who, in order to find a safe route for a given shape in a given environment, create an N-dimensional grid, the configuration space, for the N degrees of freedom, to navigate from the start to a given target position. [Brooks, 1983] considers the alternative approach by considering the space between the obstacles in a free-space approach.

The above algorithms, although useful for navigation through a terrain, do not answer the question of how to explore an unknown scene. The voronoi diagram approach and its partner, the triangular mesh, and its use for coding information gathered from the scene is discussed in Chapter 4.

1.1.2 Navigation in unknown terrains

With no *a priori* information of the environment, several researchers have considered the simpler problem of a terrain with only polygonal obstacles that can be fully circumnavigated. Two basic algorithms are presented by [Lumelsky and Stepanov, 1987] for a robot with a tactile sensor, to move from a start position towards a target position. In the event of hitting an obstacle, the robot circumnavigates it and then once past it, continues to seek its target.

This algorithm however, requires the robot to hit an obstacle in order to detect it. This is not always desirable. Other methods that do not require the robot to hit anything have been proposed. [Rao and Iyengar, 1990] propose a method which uses a visibility graph in two and three dimensions

[1] Solutions to Laplace's equations are harmonic functions.

to learn the environment in order to use sensorless navigation, as in *a priori* known environments. [Oommen et al., 1987] incrementally learns the visibility graph by choosing an arbitrary target position on successive moves. On reaching the edge of an obstacle, the robot makes an exploratory trip to either end of the edge and at each vertex gathers information of other obstacles. The robot then circumnavigates the obstacles until it can leave it whilst heading towards the destination. This solution is not optimal since the exploratory trips may not yield any new information, however the idea does move away from the traditional target-seeking navigation.

To increase the amount of new information learnt by exploratory motion, partial maps can be incrementally learnt, as shown by [Foux et al., 1993] using a 360° laser range finder to provide the distance information to the nearest obstacle. [Nagatini and Choset, 1999] use 16 sonar range sensors to construct a Voronoi graph of the local area and then use this graph to sequentially explore all nodes in the graph. Rather than using retracted graphs e.g. Voronoi graph, [Lorigo et al., 1997] use the original free space with the 'ground plane constraint' and uses 'repellent-functions' to move away from all obstacles in the current field of view, thus reducing the likelihood of hitting an obstacle.

Although these navigation algorithms do solve the problem of navigating a robot through an unknown environment, they are all oriented towards exhaustive or target-oriented exploration. None are optimal for applications such as reconstruction and recognition of objects or exploration of an environment. Since a 'best' path through the terrain may not acquire the 'best' images for successful object reconstruction, these algorithms are not suited to navigation for global map building, although they could be useful for local navigation. When navigating through an environment is not the only purpose of the system, but also required is to expand the amount of information of the surrounding obstacles, methods to calculate a position for the next sensor position are useful.

1.1.3 Sensor placement

Sensor placement algorithms for various tasks such as object identification and reconstruction were developed during the 1990s. [Arbel and Ferrie, 1999] discuss an off-line approach to identify objects, where the placement of the sensor is calculated to minimise the view ambiguity using entropy maps (which encodes the discriminating properties of an object as a function of the viewing position). This method is computationally expensive and could not be run in real-time. [Lacroix et al., 1992] incorporate visibility constraints and a time cost into their 'best' orientation calculation of the object to minimise the ambiguity from an object's aspect table, in which views of the object from 236 directions at 20 distances are stored. This aspect table is used to determine the next-best-view, however this method suffered from a large set-up time. [Cameron and Durrant-Whyte, 1990] also minimise the ambiguity of the identification of an object by generating probabilistic membership functions from acquired data of objects through sensor locations. The greatest difference between object membership is then represented as a utility function which is maximised (with constraints such as field of view) to provide the 'optimal' sensor location. These algorithms would be useful if the characteristics of obstacles in the environments are known, but is not so useful when the appearance

of obstacles in the environment are not known prior to exploration, as is the case considered in this thesis.

For the purpose of exploration, a good sensor placement would be one which allows a large amount of new data to be acquired. Without knowing where this data will be this is difficult to calculate. Rather than just constraining the sensor placement e.g. with field of view, more than one set of acquired data could be fused to reduce the uncertainty of a choice, or to weight the decision of a direction to move. Several proposals for 'optimal' sensor placement use more than one set of acquired data and the following describes how they can be fused together to provide a single output decision.

[Briggs and Donald, 2000], and [Lorigo et al., 1997] have both used more than one set of data to make decisions. An experimental study of voting schemes by [Pirjanian et al., 1997] with various inputs demonstrate how with several modules all working towards the same goal, e.g. to track a moving object, with the same acquired data, modules that fail can be compensated for by other modules. This allows the failed module another opportunity to e.g. find the object in the scene. Later, [Pirjanian and Christensen, 1997] show how simple functions (fast forward, move to target, avoid obstacle) can be combined by weighting or lexicographic[2] methods, to allow the robot to make a decision of where to move next. A behaviour based outdoor navigation example is given by [Langer et al., 1994] who use a local map of the robot to explore a terrain. The behaviours of the system (obstacle avoidance, target seeking, drive straight, maintain turn) are fused by an arbiter that weights votes (a set of pre-defined arc directions for motion) from each behaviour. [Brooks, 1986] describes a layered behavioural approach, from basic object avoidance to reasoning about the behaviour of objects using a priority measure to influence the overall system behaviour.

In order to review the knowledge of the system for navigation, sensor placement, etc. the positions of obstacle features in relation to the position of the robot need to be stored. Errors in the position of the sensors at the obstacle detection stage lead to errors in the global position of scene features. By simultaneously updating the position of the robot and adding information of obstacle features in the a global map these errors can be reduced.

1.1.4 Simultaneous localisation and mapping, SLAM

Also known as concurrent mapping and localisation, CML, research in this area demonstrates the requirement for positions of features in the scene to be mapped at the same time as calculating the position of the robot in relation to these features. In the earlier stages of mapping algorithms using sonar and vision sensors on large experimental areas, it was noticed that long straight walls were not being stored as such. As a mobile robot moves, errors in the odometry information arising from wheel slippage, non-uniform floor surface, and poorly calibrated tick-information causes the position information provided by the odometry to increasingly deviate from its true position. Features

[2]Lexicographic fusion ranks the importance of the objective and eliminates entries until a unique solution exists or all problems are solved

detected from these positions would be built into the map relative to the position of the robot, hence the positions of features would also drift away from their true position. Algorithms have been developed to correct for this motion drift such as [Davison and Murray, 2002] and [Newman et al., 2003] who store correlations between each feature and robot position.

The above algorithms map features in the scene that are useful to the localisation calculation, e.g. recognisable from different sensor placements, distinguishable from other close-by obstacle features. When all obstacles around the robot need to be mapped, those features which are useful for localisation should be labelled as such, so as not to reduce the impact of the error reduction by using features that are mainly useful to the robot for obstacle avoidance.

1.1.5 Exploration

The incremental process of building maps and reacting to changes in the environment enables the possibility for the robot to explore its environment on-board in real-time. Algorithms to traverse an environment which is incrementally built as the robot moves have been developed from simple methodical and thorough methods, to those which are slightly more optimal. Examples of the methodical methods include the popular Lawnmower algorithm discussed by [Arkin et al., 1993]. Other similar algorithms, discussed by [Sim and Dudek, 2003], include the seed-spreader and random walks. An example of a more optimal approach is the octree method described by [Connolly, 1985]. This method hierarchically splits the area around an object into regions in order to ensure all of the object has been viewed.

A non-exhaustive exploration algorithm was developed by [Grabowski et al., 2003] for which an Occupancy grid of the seen areas are stored, where the boundary between the known and unknown areas forms a frontier for the exploration algorithm to key into. [Yamauchi, 1997] also considers these frontiers as a basis for exploration which does offer some solution of how to expand the knowledge of the environment but does not cover the problem of which frontier to choose when more than one option is available (however [Youngblood et al., 2000] selects frontiers by moving the robot in increasing sized concentric circles until all frontiers are removed). An occlusion boundary is used by [Kutulakos et al., 1994a] to incrementally build models by purposively rotating the object (on a turntable) to allow previously unseen areas to come into view, or by arbitrarily adjusting the viewpoint by a small amount to recover the shape information, as in [Kutulakos and Dyer, 1994]. [Kutulakos et al., 1994b] also showed how to incrementally explore an unknown object with either a laser or a camera. The point on the object to view was either the occlusion boundary or the visible rim for the range sensor and camera respectively. With larger scenes where the sensor must move between objects, or when the scene is too large to be placed on a turntable, these algorithms cannot be applied directly to larger scenes.

Other methods to expand the knowledge of the environment include wall-following, for example [Lee and Recce, 1994], where sonar sensor readings are used to follow the wall of an indoor environment whilst building a map. [Althaus and Christensen, 2003] describes a method for domestic

robots, where the robot follows a human operator around the environment whilst building a map. Another strategy is for the operator to specify a set of way-markers or landmarks for the robot to navigate between, e.g. [Deng et al., 1996], [Lazanas and Latombe, 1992]. [Cartwright and Collett, 1983] discussed the navigating behaviours of bees. In their seminal work, they show how bees learn landmarks by storing an unprocessed two-dimensional snapshot of the current location which is matched to stored snapshot landmarks. The difference between the current snapshot and stored snapshots drive the bee toward the right location.

Where the size or accessibility of the scene is not known, and information of only the local environment has been gathered, the question for an exploration algorithm is not only "what should I do next?", but "what then?". When the answer to the second question depends so greatly upon the outcome of the first, no simple exhaustive exploration or information–decision algorithm can suffice. In addition to this, when faced with many options of next potential moves, how can it be determined which move may yield the highest reward: to discover new unseen areas yet to be explored.

1.2 Problem Statement

Several SLAM algorithms exist that can build up a map of the environment which the robot can then use to correct its position within the map. However, the map built only contains key world feature elements that the robot will find useful to correct its position, and not necessarily the positions of obstacles in the scene. Several algorithms exist that can build up a model of the environment from controlled motion of a camera, e.g. on a mobile robot. Motion models for traversing known or unknown environments are usually destination driven, where a specific target location exists. The localisation problem is one which must be dealt with in order for the map not to drift and bend. Map building is necessary so that the robot knows where it is, where it has been, and what it has seen, in order to anticipate obstacles, should it need to traverse the area again.

Although algorithms exist to map *a priori* unknown scenes, none are capable of environment mapping without either the aid of a human operator at some point during the execution phase, or those that randomly visit different or already visited areas of the scene. Some exploration algorithms exist to methodically cover the entire area. The problem of exploring an area in an optimal way and building an accurate map has still not been solved. To operate autonomously, an algorithm needs to decide where the sensor can and should move next, for which sensor position planning can be used. Techniques have been developed for known objects, or unknown objects of a limited size. Where the scene is not *a priori* known, or the sensor must move between the objects to view the area entirely these techniques can not be applied directly.

1.3 Aim

The aim of the research presented in this thesis is to develop an exploration algorithm to build up information about an environment which will enable a mobile agent to autonomously move around an *a priori* unknown environment using a more intelligent approach.

Whilst exploring an environment, some method of keeping track of what has been detected by the robot's input system will be required. Several methods exist to provide information about the world around us from the perspective of a camera. Many of these techniques are run off-line, after the camera has left the filming arena. However, for a robot to explore an environment without unnecessarily repeating the areas it visits, some form of map must be built as the exploration continues, in real-time and at run-time.

As the map is incrementally built, the robot must be able to make a decision on its next motion command. By analysing the current knowledge of the world, for example which obstacles and areas have been seen and which obstacles and area should be viewed next, an exploration algorithm capable of visiting the majority of the scene area and viewing the majority of obstacles in the scene will be developed. A sensor placement strategy which allows an optimum amount of information to be gathered from a well chosen position would be advantageous to reduce the number of sensor locations to explore the scene.

In summary, the following will be developed:

1. An exploration algorithm which maximises the improvement in knowledge of the environment.
2. An optimal strategy for the next-best-viewpoint calculation.
3. A map building algorithm able to store information about the environment to provide the exploration algorithm the information it requires using vision.

1.4 Thesis Overview

To achieve the aim of developing a mobile robot system to autonomously explore its environment, a modular framework has been developed and the layout of the system is shown in Figure 1.1. The robot platform for which this framework has been developed for, is fitted with a single camera, on-board processing capabilities and basic movement software. The robot platform is discussed in more detail in Section 2.2. Figure 1.1 describes how features extracted from each image are used to build up a map of the world. This map, along with a current image, is used to re-calculate the position of the robot. The built map is evaluated to retrieve information about the scene obstacles which are used to select the next best viewpoint for a given obstacle. The choice of obstacle is based on a view-improvement calculation to enable autonomous exploration. When the robot moves to the new position, new images are grabbed and the process repeats. A more detailed explanation of the system follows.

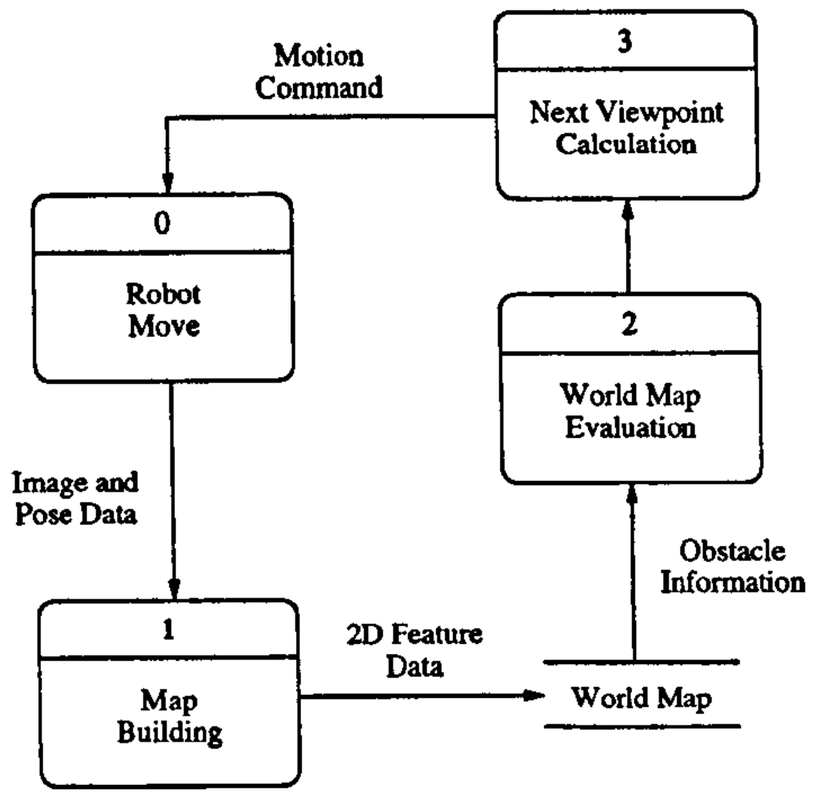


Figure 1.1: Autonomous Exploration System Diagram.

Scenes considered in this thesis consist of obstacles gathered from office-type environments, which are generally made up of box-like objects e.g. filing cabinets, bookshelves, boxes etc. A further simplification has been assumed: that the robot will only travel on a two-dimensional flat surface. Although this does reduce the direct usability of the system in other environments, by simplifying the scene feature extraction to a basic level, it allows the basic information about the location of obstacles to be attained. Other, perhaps more sophisticated, techniques to extract the location of obstacles could be used in place of this first module, whether they use vision or other sensors. The key to the remaining part of the system is that obstacle location information is acquired from the scene and stored for later interpretation and use.

Input data consists of colour Red-Green-Blue (RGB) image sequences and odometry information provided by the tick-encoders on the robot wheels as the robot travels through the scene. Features that can be easily extracted from images gathered in these scenes, which are the key element to these obstacle's representation in 2D, are the vertical edges found on the exterior of the objects. The input to the system framework is therefore based on vertical edges that define potential obstacle boundaries that are detected in each image. Being stable image features they reduce the processing of information and their position can be accurately computed from the image. The position of these edges in the image are projected from the camera centre into the robot's world, which is incrementally built as the robot moves. The intersection of two projection lines from a feature in two images is defined as the position of this feature in the world. The problem of finding corresponding features in two images is well known, and is overcome in this system by gathering evidence of feature positions in an accumulator grid that produces robust information about object boundary locations of potential obstacles. During the map building process, the world map features are used to estimate the position and orientation (pose) of the robot. An initial estimate of the position is obtained from a simple model of the robots motion using the odometry information. The discrepancy between detected image features and the projection of existing world map features onto the image is then minimised to provide a better estimate of the robot's position.

Obstacle locations that are detected many times are extracted from the accumulator grid and stored in a world map. This map is maintained in a graph where nodes represent the scene features and links between these features define barriers or obstacles. Initially, obstacles are formed on all links between the nodes in the map. Chapter 2 introduces the framework for building the map from the acquisition of each image. The feature extraction is discussed and the transformation from one-dimensional image features to a two-dimensional world frame is given. Preliminary results are given to establish the process of building the map.

From the world map, positions of obstacle boundaries are used as input to a next-best-viewpoint planning strategy. This algorithm allows the robot to plan its next best view-point given the current knowledge of the surrounding area whilst avoiding known obstacles. Using a set distance to travel, the optimal next position is calculated such that the viewing angle at the hypothesised position will be maximised. Chapter 3 introduces the next-best-viewpoint calculation which is developed from the case with one corner to view around, one opening to view through, to N openings to view. Errors in the positions of the corners and the robot are considered. An example exploration of a synthetic scene using a human operator's selection of the openings to view is discussed.

Chapter 4 addresses the question of how to choose an opening by evaluating the world map. Candidate openings are selected from the current list of obstacles. Given this set of obstacles and their corresponding next-best-viewpoints, the improvement that would be made to the world map from data gathered at hypothesised positions is used to select which candidate to view. Using successive obstacle selections, a set of algorithms to explore an unknown scene is presented. Three improvement measures are considered and example explorations, within a synthetic environment, for each of the measures are considered. The synthetic scene map used in this and the following chapter is used in a similar way to a look-up table, where the extremal boundaries of obstacles which are visible from a given viewpoint (taking occlusions by other obstacles into account) are added to the world map. This replaces the image acquisition and processing system that would provide the same information after the back-projection calculation, accumulator grid and world feature selection described above.

By merging the view-improvement measures together with an obstacle importance measure which considers areas that are new to the world map more important than the areas that have already been explored (to instill a pioneering aspect to the exploration), Chapter 5 considers the synthetic scene exploration of three environments. Each of these environments highlights the use of the exploration algorithm and demonstrates the ability of the view-improvement approach to enable successful autonomous exploration.

1.5 Contributions to Research Field

A unique exploration algorithm has been developed which incorporates knowledge of the environment in a targeted way to develop a more intelligent approach. This algorithm has been applied to the motion of an autonomous agent discovering an initially unknown environment and building an

internal map as it traverses the area. This approach is not target driven, rather seeking new areas to view and explore.

In the development of this strategy, a novel planning algorithm which provides for optimal viewing angles and maximises the opportunity to discover uncharted areas during the exploration has been developed.

As information is collected, an internal map is constructed based upon information accumulated in an occupancy grid based upon visual data. The novelty of this approach is that features of high confidence are extracted from this grid and this greatly reduces the computational overhead when querying the map. Errors in the locations of significant features are shown to reduce over time.

1.6 Glossary of Terms and Abbreviations

Exploration To visit areas of the scene that are not well known.

Autonomous No input from a human operator during program execution.

Navigation To get from a start position to a given target position.

World Map Collection of world feature points which are connected within a triangular mesh.

World Feature Point A reconstructed scene feature, stored in the world map.

World Element A world feature point, stored with an estimate of the error on its position, a confidence of the feature existing (C_f), and a list of camera positions from which is was viewed.

Image Feature Point An extracted feature point on an image.

Obstacle A non-traversable barrier to the robot (e.g. a cardboard box); in this work, an obstacle is detected in an image by the vertical edges on the obstacles boundary (e.g. the corners of the cardboard box).

Opening A traversable virtual barrier (e.g. a doorway); a gap between obstacles.

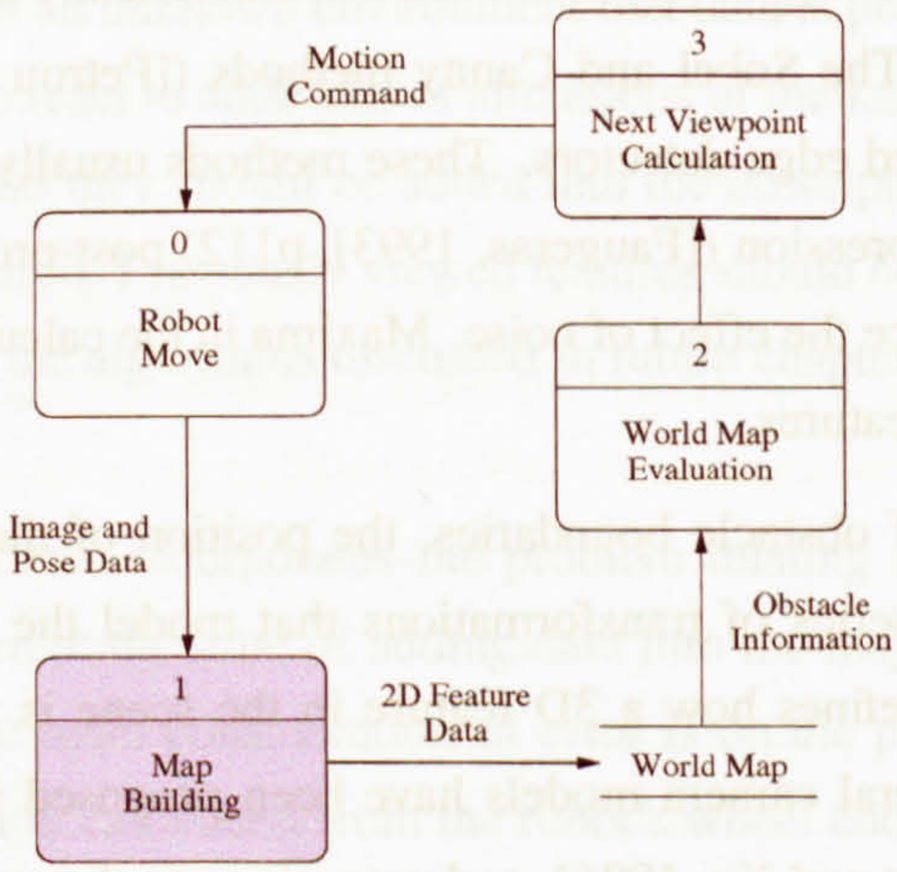
WCF - World Co-ordinate Frame.
$\mathbf{P}$ - position of camera in the WCF
σ_P - error on the camera position P
α - rotation of the camera, WCF, with respect to the x-axis

$\mathbf{p}$ - position of world element in the WCF
$\mathbf{p}'$ - image feature point in the camera's image co-ordinate frame
σ_β - error on the image feature position p'
$\mathbf{M}$ - linear camera model projection matrix
$\mathbf{R}$ - rotation matrix of camera position
$\mathbf{T}$ - translation matrix of camera position (WCF)

$\mathbf{K}$	-	camera's intrinsic paramters
C_m	-	confidence of the match between an image feature detected in two images
$\mathbf{g}$	-	image pixel gradient value (RGB)
C_f	-	confidence of the existence of a world feature
l_i	-	back-projection line from the image feature to the world feature
C	-	position of an obstacle corner feature (WCF)
θ	-	angle to rotate to move to the next-best-viewing position
σ_θ	-	error on the θ calculation
r	-	distance to move to the next-best-viewing position
R	-	distance from current robot position P to corner position C
γ	-	angle between the horizontal x-axis and the obstacle corner C (WCF)
ψ	-	angle between the current position, the corner and the next-best-viewing position
ν	-	viewing angle from the next-best-viewing position to the corner in question
J_1	-	utility function for the next-best-viewing position for one corner
J_2	-	utility function for the next-best-viewing position for one opening
C_o	-	confidence of the existence of an obstacle in the world map
$\mathbf{w}$	-	viewline vector from the camera to a position in the world map; used to specify the direction from the camera position to obstacle- or area-view histogram cells
I_o	-	the improvement in the view of an obstacle from a potential next view position compared to previous viewing positions of the obstacle
I_a	-	the improvement in the view of the area from a potential next view position compared to previous viewing positions of the area
I_p	-	similarity in a potential next view position compared to previous viewing positions
O_i	-	importance measure of an obstacle
I_J	-	joint improvement measure; a combination of I_o, I_a, I_p and O_i

Chapter 2

Incremental Map Building



This chapter introduces the method used for the construction of a two-dimensional map of scene obstacles. Position information of obstacles are inferred from camera data acquired from a mobile robot, and a robust reconstruction of the locations of scene obstacles using an accumulator grid is generated. The novelty of this method lies in the probabilistic approach to adding data into an accumulator and the extraction of confident features for storage in a world map. The main goal is to provide a representation of the obstacles in the environment to enable interpretation of the reconstructed world for exploration.

2.1 Introduction

In order for a mobile robot system to move within an environment, the positions of obstacles that restrict the motion of the robot need to be known. The construction of obstacle maps are therefore necessary when more than just the current local obstacle locations are needed, such as the case for reactive systems. Map building is approached in many ways depending on the source of data and the requirement of the maps usage once built.

Sensors to acquire data from a scene are grouped into two categories, active and passive. Passive sensors (e.g. vision or infra-red cameras) rely on the environment to provide a medium for observation, whereas active sensors (usually depth sensors e.g. laser or sonar range scanners) emits energy into the environment, and by observation or timing measurements, infer scene structure. The readings taken from time-of-flight scanners can be prone to errors because there is usually no verification of whether the received signal is the same as the signal sent. Repeating sensor readings and pulsed chirping can combat this problem. These sensors, however, give only the depth and direction information of obstacles in the scene. Vision systems can provide more information such as the position of edge data and colour. In this chapter, the data source for the map building process are colour images, captured from a single camera on-board the mobile robot described below.

Vertical edges, which are a boundary between two regions with relatively distinct colour- or gray-level values can be extracted from an image using simple edge detection techniques. A standard approach is to convolve the image with a small mask to numerically estimate the gradient of the pixel values across the image. The Sobel and Canny methods ([Petrou and Bosdogianni, 1999], p306) are examples of widely used edge detectors. These methods usually incorporate a smoothing stage and use a non-maxima suppression ([Faugeras, 1993], p112) post-processing stage after the gradient calculation in order to reduce the effect of noise. Maxima in the calculated gradient values determine the pixel position of edge features.

Given the pixel position of obstacle boundaries, the position of the corresponding world features can be estimated using a series of transformations that model the camera and its position in the world. A camera model defines how a 3D feature in the scene is mapped onto its respective 2D feature in the image. Several camera models have been proposed in the literature, from standard projective models e.g. [Wan and Xu, 1996], and extensions to the static model for zoom lenses e.g. [Willson and Shafer, 1993] and outdoor systems e.g. [Collins and Tsin, 1998]. A good summary of widely used camera models and calibration techniques can be found in [Lewin et al., 1998]. The inverse of these transformations is used to calculate the projected line that connects both the image feature and the scene feature. The intersection of two projected lines of the same scene features from different camera positions defines the location of that feature in the scene. The problem of finding the same scene feature in pairs of images is a well researched problem in the stereo-vision community. Techniques including epipolar geometry and the tri-focal tensor, [Faugeras, 1993], which enables good matches to be found, or wrong matches to be excluded from consideration, are well used. The main advantage of using the epipolar constraint is that it limits the location of corresponding edge matches.

Ideally the intersection of two projected lines provides an exact position for the obstacle location in the scene. However, errors in the estimation of the camera position, image discretisation effects and incorrect matching of features between two images mean that this exact ideal can never be achieved. The performance of vision systems can vary with changes in light conditions and with changes to the viewed object position and orientation. Repeating measurements of features in the scene can improve the accuracy of the reconstructed feature location by confirming information on the posi-

tion. The same feature being detected several times also increases the confidence that the feature match correctly represents a scene feature. This accuracy improvement and confidence enhancement can be achieved over time by accumulating the detected obstacle locations in a 2D grid where each cell represents an area in the scene. A common method used in sonar and laser systems is to store the information in a two-dimensional grid where each cell encodes the probability of the corresponding scene area being occupied by an obstacle. Such a data structure is called an Occupancy grid. This technique is used by [Borenstein and Koren, 1988], [Borenstein and Koren, 1989], [Borenstein and Koren, 1991], [Ulrich and Borenstein, 2000], and [Schiele and Crowley, 1994], to name a few. Each cell is initialised with a value of 0.5 to signify that there is an equal chance of the cell being occupied or empty. As data is added to the grid, this value increases when an obstacle is detected, or decreases if the cell is found empty. This works well for sonar and laser systems where a large amount of data from repeated readings are added into the grid.

It is essential in exploration of an unknown environment that data is processed as it is received, since the navigation must be able to react to additions or alterations of the known parts of the environment. As new features are discovered they should be added into the description of the world, therefore an incremental algorithm is required. Previously viewed features should be updated when viewed again. The map will be analysed by the algorithms discussed in future chapters to facilitate exploration and obstacle avoidance.

The robot mapping problem also incorporates the problem relating to the precision to which the position of the robot is known at the time of adding data into the map. If the position of the robot is accurately known, then the main consideration of error is on the positions of the features being mapped. Odometry information calculated from the robot's wheel encoders provides an estimate of how the robot has moved, which can be used to estimate the global pose of the robot. However the odometry information has an error associated with it, proportional to the distance that the robot moves and rotates. The error on the estimated position will therefore increase over time, corresponding to a drift in the robot's position. This drift in position in turn means the positions of features added to the map will increasingly shift from their true position.

In order to correct for the drift in the robot's position, an area of research referred to as simultaneous localisation and mapping, SLAM, or continuous mapping and localisation, CML, has received much recent attention. In the literature, the most popular approach undertaken to solve this SLAM problem is a method which enables the calculation of correlations between each feature in the map and the current robot position. As the robot moves and gathers more data, the state of the world is updated along with the correlation between the feature positions. This is implemented in a Kalman Filter, a recursive process of prediction–correction (see [Bar-Shalom and Li, 1993], Chapter 5) using a model of the camera motion to predict its next position, and the measurements from the world feature locations to update and correct this prediction. A comparison of the various probabilistic techniques of robot mapping, covering the Kalman Filter and Occupancy grids, can be found in [Thrun, 2002]. [Dissanayake et al., 2001] use a Kalman Filter with a robot and all feature positions in the state vector. A motion model of their long wheel-base van is used to predict the van position, and shows that

the correlation between features in the environment is of paramount importance in solving the SLAM problem. [Davison and Murray, 2002], [Folkesson and Christensen, 2003], [Laubach et al., 1999], [Newman et al., 2003], to name but a few, focus on the SLAM/CML problem in this way. The SLAM problem is not yet fully solved and is still an ongoing research area.

Other optimization tools can be used to match the robot's current findings to that which best match the position of the robot in the known world. This is used by [Yamauchi et al., 1998] with an Occupancy grid approach. The position of the robot is calculated from the global world without an estimate of the current position. Their approach can work well in environments where different robot positions can be completely distinguishable from one another, but does not perform well in environments where many obstacles appear similar or patterns in obstacle configurations are detected. This global method is useful if the robot moves a large distance between position correction calculations. However, it may be computationally quicker if there already exists an estimate of the position which only needs to be improved, since just a local (rather than global) search could be considered. [Duckett and Nehmzow, 1997] also use an Occupancy grid for localisation, where initial guesses of locations are calculated followed by arbitrary moves in order to limit the positions that are valid.

As the robot moves and position errors are accumulated, the matching of image features to world features is key to the position estimation method presented in this chapter. Subsequent chapters discuss the need to view new areas of the environment in order to explore the uncharted areas and expand the information of the scene obstacle locations. It is of course necessary to maintain visible confirmation of known features for the position estimation to calculate a good position estimate. The trade-off between maintaining an accurate estimate of the current position and viewing new areas is not established in this work, but must be considered before the system can be used in real-world situations.

In this chapter, the problem of incrementally building a map of obstacles in a scene is addressed. Initially a tele-operated robot is considered, but this is not central to the method, and all the techniques are equally applicable to an autonomous vehicle. The camera model is considered first, and it is shown how the inverse of the model is used along with a probabilistic error model to insert the location of obstacles into an accumulator. To validate the map building process, results of an incrementally built map are shown. These results demonstrate the possibilities of building human- and machine-interpretable static obstacle maps. The results show a reduction in errors on feature positions over time. The navigation algorithms that *completely* replace the operator are presented in subsequent chapters. A calculation for the position of the camera is then presented. This method triangulates the camera position from positions of three image features on the current image to their corresponding world map feature positions.

2.2 Mobile Robot Testbed

The algorithms described in this chapter are tested with a Pioneer P2-DX robot from ActivMedia [http://www.activmedia.com, 2000]. The robot is circular with 50 cm radius and is 40 cm high. Steering is by a two wheel differential motor, with an additional caster wheel to stabilise the base. The robot is fully equipped with on-board processing, power supply, and a single colour RGB camera capable of pan-tilt-zoom (only the pan is used in this work).

Software libraries supplied with the robot enable the global position and orientation of the robot base to be inferred from the wheel tick encoders. Using the supplied software, the distance actually travelled per wheel tick is initially calibrated for the floor surface on which the robot is to be used. This reduces the error on this odometry information, however position estimation software is also needed to further improve the accuracy of the position; errors on the angle the base rotates has a greater effect on the global position error than translational errors, however the position error encountered on the trials carried out in this work did not dramatically effect the results.

The robot base is equipped with on-board power supply, capable of powering the base, camera and on-board processing for more than an hour without needing to recharge. When the power is running low an alarm sounds, allowing the operator to pause the experiments and hot-swap batteries.

2.3 Obstacle Detection

An image is a rich source of data but its interpretation in terms of scene elements is complicated and difficult. The general scene interpretation problem has proven intractable despite nearly fifty years of intensive research. Therefore in order to progress exploration a restricted, but fairly common, sub-class of scenes which are representative of indoor, man-made office or laboratory environments is assumed. Typically these involve floors, which are level horizontal planes, and solid obstacles which are walls, doors, or office furniture. These objects are delineated by a plethora of straight horizontal or straight vertical edge structures. By restricting attention to these scenes enables the use of relatively simple feature extractors which can be implemented to achieve near real-time performance.

Vertical edges in the local vicinity of the robot define the location of obstacles that will immediately influence the path of the exploration. Other more distant vertical edges in an image will form obstacles in the global scene but do not affect the immediate path of the robot. In addition, errors on the location of reconstructed features increase as the feature's distance from the camera increases (discussed further in Section 2.5.1) and hence local edges provide the most accurate information. To reduce the computational overhead of finding all vertical edges in each image, only edges that intersect the central horizontal scan line are considered. Figure 2.1 is a typical office environment and Figure 2.2 is an example image acquired by the mobile robot's vision system.

Vertical edges which are a boundary between two regions with relatively distinct colour- or gray-level values can be extracted from an image using simple edge detection techniques. A standard

Figure 2.1: Office scene set-up with mobile robot

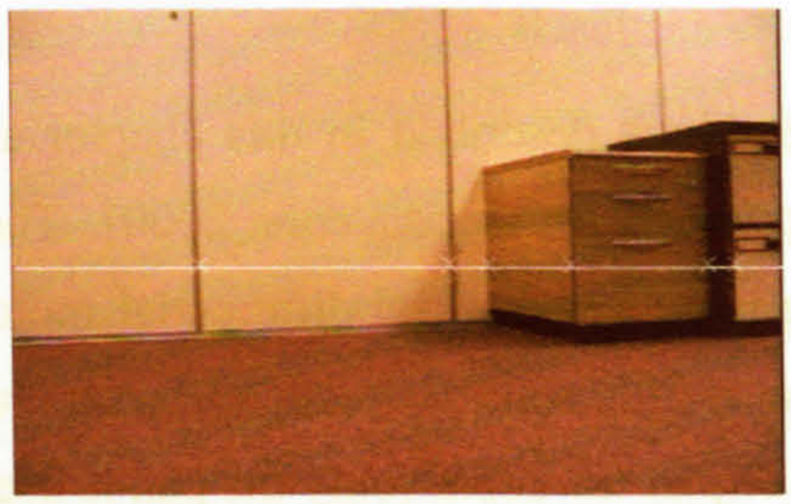


Figure 2.2: Example image with horizontal scan line and detected vertical edge features highlighted.

approach is to convolve the image with a small mask to numerically estimate the gradient of the pixel values across the image. The mask of Figure 2.3 is an extension of the Sobel mask, see [Petrou and Bosdogianni, 1999], p296, for a derivation of the numbers in the mask.

This vertical gradient mask is convolved with the central scan lines of each image. This provides a first-order edge-gradient value $\mathbf{g}$ at each pixel location, where $\mathbf{g}$ is the red-green-blue vector of the colour level derivative. Maxima in these calculated edge-gradients define candidate vertical obstacle boundaries. Using nonmaxima-suppression ([Faugeras, 1993], p112) any sub-maximal edges are ignored, providing sparse image data. The pixel position of obstacle boundaries in an image provides the direction information for each feature from the camera into the scene via a model of the camera.

1	1	0	−1	−1
2	2	0	−2	−2
2	2	0	−2	−2
2	2	0	−2	−2
1	1	0	−1	−1

Figure 2.3: Vertical gradient mask which is convolved with the central scan lines of each image to detect vertical edge features.

The central horizontal scan line and the detected vertical edges are highlighted in Figure 2.2. In the majority of images acquired by the robot's camera, immediate obstacles to the robot can be found at this height in each image. In this work, only one of the interlaced fields and five scan lines are considered.

2.4 Linear Camera Model

A camera model defines the mapping of a scene feature onto its respective image feature. The vertical edges of obstacles in the scene are defined by their position in the horizontal scene plane (2D). Image

features are defined by their position in the horizontal image plane (1D) across the horizontal scan line. The camera model is therefore simplified from the standard three-dimensional scene frame with image feature positions in two dimensions, to a 2D scene frame with 1D image features. This is adequate for the system being presented here.

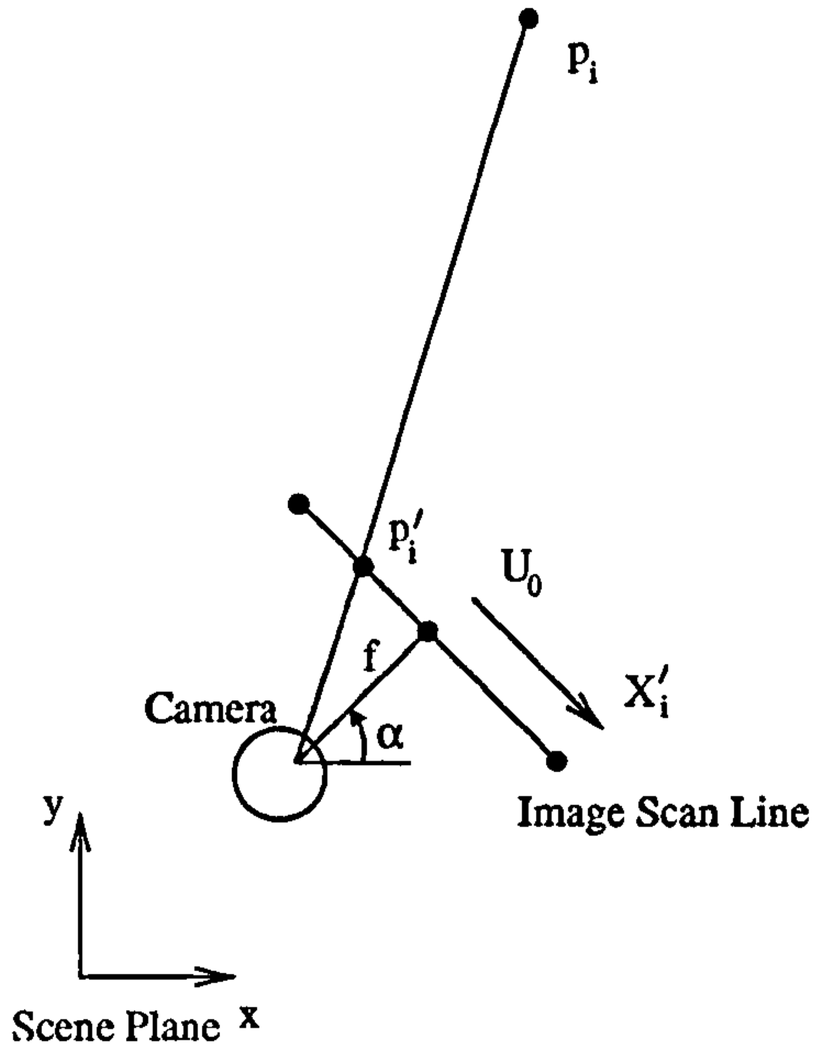


Figure 2.4: Plan view of the scene. Camera model to define the transformation of scene feature $\mathbf{p_i}$ in (x,y) scene co-ordinate frame to the feature $\mathbf{p_i}'$ on the Image Scan Line. The image feature $\mathbf{p_i}'$ is defined as a number of pixels, X_i', from the image's principal point, U_0. f defines the focal length of the camera lens. The green circle in the Scene Plane depicts the robot position with offset camera; α is the camera angle.

Figure 2.4 shows a two-dimensional plan view of a three-dimensional scene. The scene point is defined in homogeneous coordinates ([Gonzalez and Woods, 1992], p57) by $\mathbf{p_i} = [x_i, y_i, 1]^T$ and is mapped onto the image point $\mathbf{p_i}' = [X_i', 1]^T$ by a projection $\mathbf{M}$ of the form

$$\mathbf{p_i}' \propto \mathbf{M}\mathbf{p_i} \tag{2.1}$$

Since any scene point along the line $\mathbf{p_i}'\mathbf{p_i}$ can be mapped onto the image point $\mathbf{p_i}'$ by the projection $\mathbf{M}$, $\mathbf{M}\mathbf{p_i}'$ is only proportional to $\mathbf{p_i}$.

Motion of the camera is defined by a rotation matrix $\mathbf{R}$ and a translation $\mathbf{T}$ with respect to the scene (extrinsic parameters), defined in a right-handed co-ordinate frame where

$$\mathbf{R} = \begin{bmatrix} \sin\alpha & -\cos\alpha \\ \cos\alpha & \sin\alpha \end{bmatrix} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{2.2}$$

where α is measured in the anti-clockwise direction from the x-axis, as shown in Figure 2.4.

The projection $\mathbf{M}$ can be related to a model of the camera's intrinsic parameters $\mathbf{K}$ and the camera position and orientation by a series of linear transformations which, assuming no lens distortion,

define the camera model ([Gonzalez and Woods, 1992], pp56-68; [Faugeras, 1993], Chapter 3)

$$\mathbf{M} = \mathbf{K}[\mathbf{R}|\mathbf{T}] \tag{2.3}$$

As well as external transformations, the camera model includes intrinsic parameters of the camera: the focal length f, the image coordinate of the principal point U_0, and the pixel size K_u. These intrinsic parameters are specific for a given camera, and change according to the camera lens and CCD (Charge Couple Device) array. These parameters are represented by the matrix K, where

$$\mathbf{K} = \begin{bmatrix} K_u & U_0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 \\ 0 & 1 \end{bmatrix} \tag{2.4}$$

which is simplified from three to just two unknown parameters, $K_u f$ and U_0. Estimates of these two parameters can be calculated by performing a calibration using known scene features and their corresponding image feature locations.

2.4.1 Camera calibration

A dual-plane calibration chart is used for the calibration to provide several 2D known scene features in the scene which can easily be detected and matched to their respective 1D features in the image.

To calculate the intrinsic parameters $K_u f$ and U_0, the following equation needs to be solved.

$$\mathrm{p_i}' = \begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} K_u f & U_0 \\ 0 & 1 \end{bmatrix} \left[\begin{array}{cc|c} \sin\alpha & -\cos\alpha & t_x \\ \cos\alpha & \sin\alpha & t_y \end{array}\right] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.5}$$

Along the image scan line, each pixel feature position X', is calculated by

$$X' = \frac{u}{w} = U_o + K_u f \frac{A_1}{A_2} \tag{2.6}$$

where $A_{1i} = X_i \sin\alpha - Y_i \cos\alpha + t_x$ and $A_{2i} = X_i \cos\alpha + Y_i \sin\alpha + t_y$ and $A_i = A_{1i}/A_{2i}$
For a fixed camera position, each image feature to scene feature match defines a row in a system of equations in the form

$$\begin{bmatrix} 1 & A_i \\ \vdots & \vdots \\ 1 & A_n \end{bmatrix} \begin{bmatrix} {U_0}^* \\ K^* \end{bmatrix} = \begin{bmatrix} {X_i}' \\ \vdots \\ {X_n}' \end{bmatrix} \tag{2.7}$$

for n image-world feature matches, where ${U_0}^*$ and K^* are the estimated values for U_0 and $K_u f$ respectively.

Errors such as incorrect image to scene feature matches and inaccuracy in known scene feature locations (human operator using a tape measure) can lead to calculations of the two parameters K^* and ${U_0}^*$ which are far from correct. These estimates are known as outliers. Such outliers affect the

mean value if they are considered in its calculation. A better estimate of the mean can be calculated by recognising these values as outliers and not including them in the mean value calculation.

A fast and efficient tool, the mean-shift algorithm, can be used to calculate the robust mean of an N-dimensional problem. This algorithm is presented by [Comaniciu and Meer, 1999]. To simplify this algorithm, the values of K_i^* and U_{0i}^* are normalised to the same range of values so that they can be used as input for a two-dimensional mean-shift algorithm. The algorithm begins with a solution for $K_u f$ and U_0 being selected at random from the set of values calculated from equation [2.7]. The statistical mean is calculated for these selected values and all the other values for $K_u f$ and U_0 within a set distance from the selected point. By normalising the two values to the same range, the same maximum distance from the mean can be used for both values. This set distance is selected by the operator as a nominal value, and depends upon the required accuracy of the final mean value as well as the distribution of the points. A new hypothesised mean is "shifted" to that solution. The points that fall within the set distance are then used to calculate the next mean hypothesis. This is repeated until the mean value converges, as shown by [Comaniciu and Meer, 1997].

The intrinsic parameters can fluctuate as the camera lens temperature changes, which happens when the camera is first turned on. Once the lens of the camera has stabilised at room temperature the intrinsic parameters can be assumed constant. Here, it has been assumed that no lens distortion affects the linear camera model. This can hold for good quality camera lenses, but for cheaper lenses it will not be the case. The lens distortion affects the outer edges of the image more than the centre. In this work, the outer eight pixels (on each side of the image) are ignored.

2.4.2 Back projection

Once the intrinsic camera parameters are known the camera model is complete. Each image feature constrains the location of a scene feature to a 1D line in the 2D world plane from the camera centre through the world feature and beyond. The position of detected image features can therefore be projected from the camera centre to estimate the location of world features.

This back-projection can be achieved by inverting the equations of image formation. According to equation [2.1], features in the 2D world can therefore be determined from image features by

$$\mathbf{p_i} \propto \mathbf{M^{-1}} \mathbf{p_i}' \tag{2.8}$$

In order to obtain a practical realisation of this equation, the back-projection of pixels in world coordinates is considered.

$$\mathbf{M^{-1}} = [\mathbf{R^T}| - \mathbf{T}]\mathbf{K^{-1}} \tag{2.9}$$

This equation is simplified such that the world coordinates of $\mathbf{p_i}'$ are given by $\left[\frac{X'-U_0}{K_u f}, 1\right]^T$ and the projection matrix only contains extrinsic parameters. Using $\mathbf{p_i}'$ in a homogeneous form $[X_i', 1]^T$

$$\mathbf{p_i} \propto [\mathbf{R^T}| - \mathbf{T}] \begin{bmatrix} 1/K_u f \\ 1 \end{bmatrix} \left(\mathbf{p_i}' - \begin{bmatrix} U_0 \\ 1 \end{bmatrix} \right) \tag{2.10}$$

The proportionality is removed by including a scale factor c. This equation can be interpreted as a straight line passing through the camera centre and the position of the feature in the image and in the world. In practise the world coordinates of $\mathbf{p_i}'$ cannot be calculated, but a point on the line can be determined. If $c = 3$, for example, then

$$\mathbf{p_i} = 3\,[\mathbf{R^T}|-\mathbf{T}] \begin{bmatrix} 1/K_u f \\ 1 \end{bmatrix} \left(\mathbf{p_i}' - \begin{bmatrix} U_0 \\ 1 \end{bmatrix} \right) \tag{2.11}$$

This equation has the same geometric interpretation as equation [2.10] in that both define a back-projected line from the camera position through the point $\mathbf{p_i}$.

This model is only applicable to the restricted motion of a camera moving along the horizontal ground plane, and considering features across only the central scan lines of each image. If this limitation was removed so that either the camera was able to move over non-level floors, or that the features can be detected across the entire image, more than just one dimension would need to be considered, i.e. the extrinsic camera parameters would have to be represented in three-dimensions, and the intrinsic camera parameters would have to be represented in two-dimensions. The simplification assumed here, however, is adequate for the environments considered in this thesis.

2.5 Evidence Gathering and World Feature Detection

The back-projected line equations provide an exact algebraic solution to the direction of the feature position in the world. This ideal solution is not accurate because of two main error sources: the position of the cameras and the error in locating features in the image.

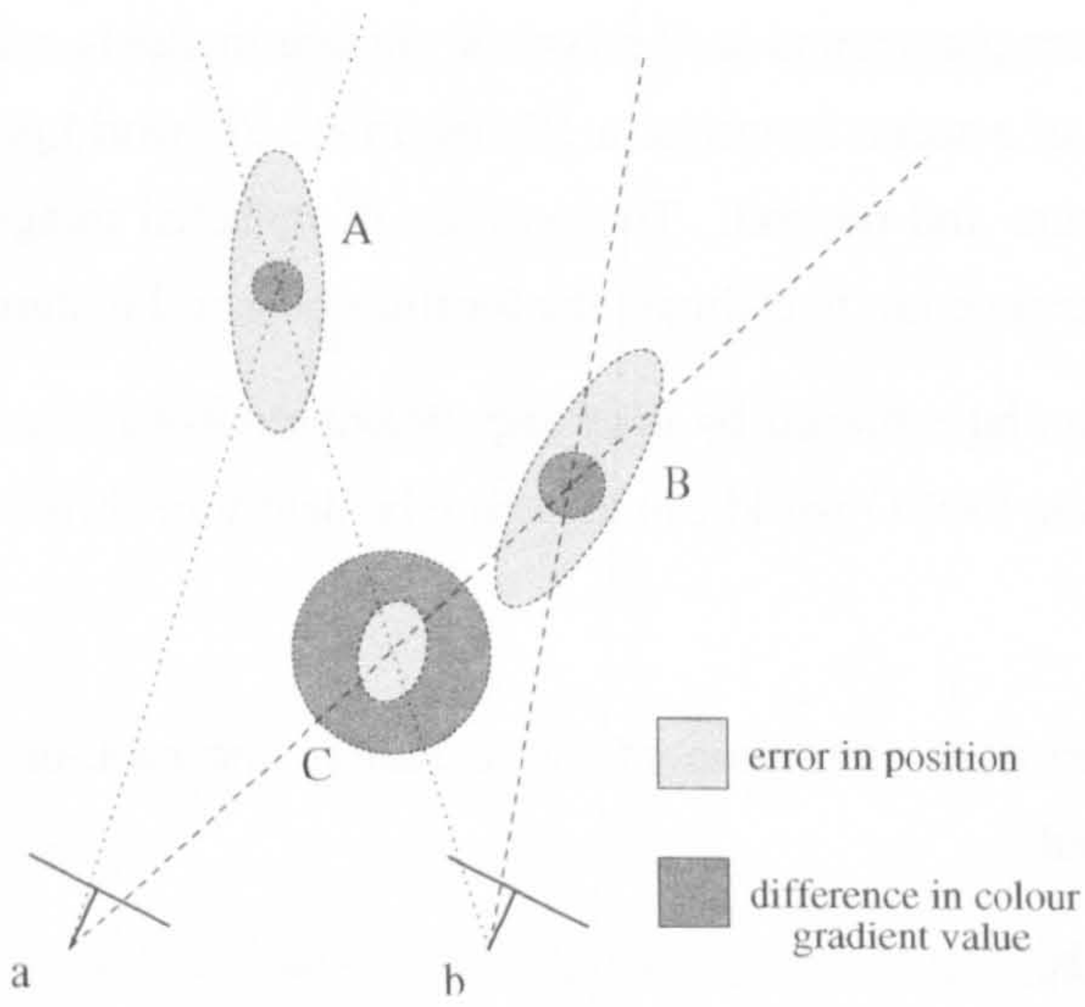


Figure 2.5: Two scene features A and B are detected at two camera locations a and b; intersections of the back-projected lines that correctly define the feature locations A and B, and an incorrectly matched feature at C.

Consider the situation of Figure 2.5 in which two scene features A and B are detected in the images acquired at two camera positions a and b. The two back-projected lines from each camera intersect at four locations: at the correct locations of A and B (lines aA–bA and aB–bB respectively), and at two incorrect locations (lines aB-bA at C, and aA-bB which is out of the range of the figure).

Each of these four intersections define potential obstacle locations in the world. The lighter shaded error ellipses in Figure 2.5 at A, B, and C indicate the size and shape of the error covariance estimate of these world feature positions. Since the magnitude of the position error may not be indicative of the correctness of the match, some other measure must be used to indicate the confidence of the match. In Section 2.3 the gradient of a pixel, g, was used to identify a vertical edge in an image. This information can be used here to measure the similarity of the detected edge features by calculating the magnitude of the distance between the two edge gradient vectors. The darker shaded circles in Figure 2.5 indicate the value of $|\mathbf{g_i} - \mathbf{g_{i-1}}|$ for the hypothesised feature locations (A, B, and C). To measure the confidence of the edge match between images i and $i-1$, the following is used.

$$C_m = \frac{1}{1 + |\mathbf{g_i} - \mathbf{g_{i-1}}|} \tag{2.12}$$

As the RGB values of the two vertical edge gradients converge, the confidence of the feature match tends towards 1.0. This confidence measure reduces the impact of the feature location confidence calculated using the error model discussed below. A drawback of this approach however is that a low confidence weighting is given to the same obstacle boundary seen with very different backgrounds (since their edge gradient values would be very different). Another consideration is that lighting conditions may vary whilst image data is collected from the environment. Although this may change the effectiveness of a particular threshold for global feature extraction, the comparative measure of equation [2.12] is calculated with a short time difference and will therefore not be affected by changes in lighting conditions which vary over a large amount of time.

As well as the locations of features found from incorrect feature matches, some vertical edge features, such as shadows, may be added into the world map alongside real obstacle features. To allow such transient artefacts to be distinguishable from stable real obstacles, points in the accumulator are labelled with a confidence value, C_f, that measures their certainty of being real scene obstacle features; C_f is calculated from the number of votes V accumulated for the scene feature and the number of times S that the relevant scene area is viewed. The values of S and V are accumulated throughout the robot's motion from each viewpoint.

$$C_f = 1 - \frac{S}{V^2} \tag{2.13}$$

Figure 2.6 shows a plot of C_f against V and S where $S \geq V$. As the number of votes accumulated for a feature increases and $S = V$, which is true if the feature is detected at every opportunity, the confidence of C_f increases (as shown by the darkening colour of the figure along the line of $S = V$). This is important to note since a feature detected three times from three opportunities should not be considered as confident as a feature detected ten times from ten opportunities. V/S is the fraction of opportunities that the feature has been seen, but does not have the required properties of equation

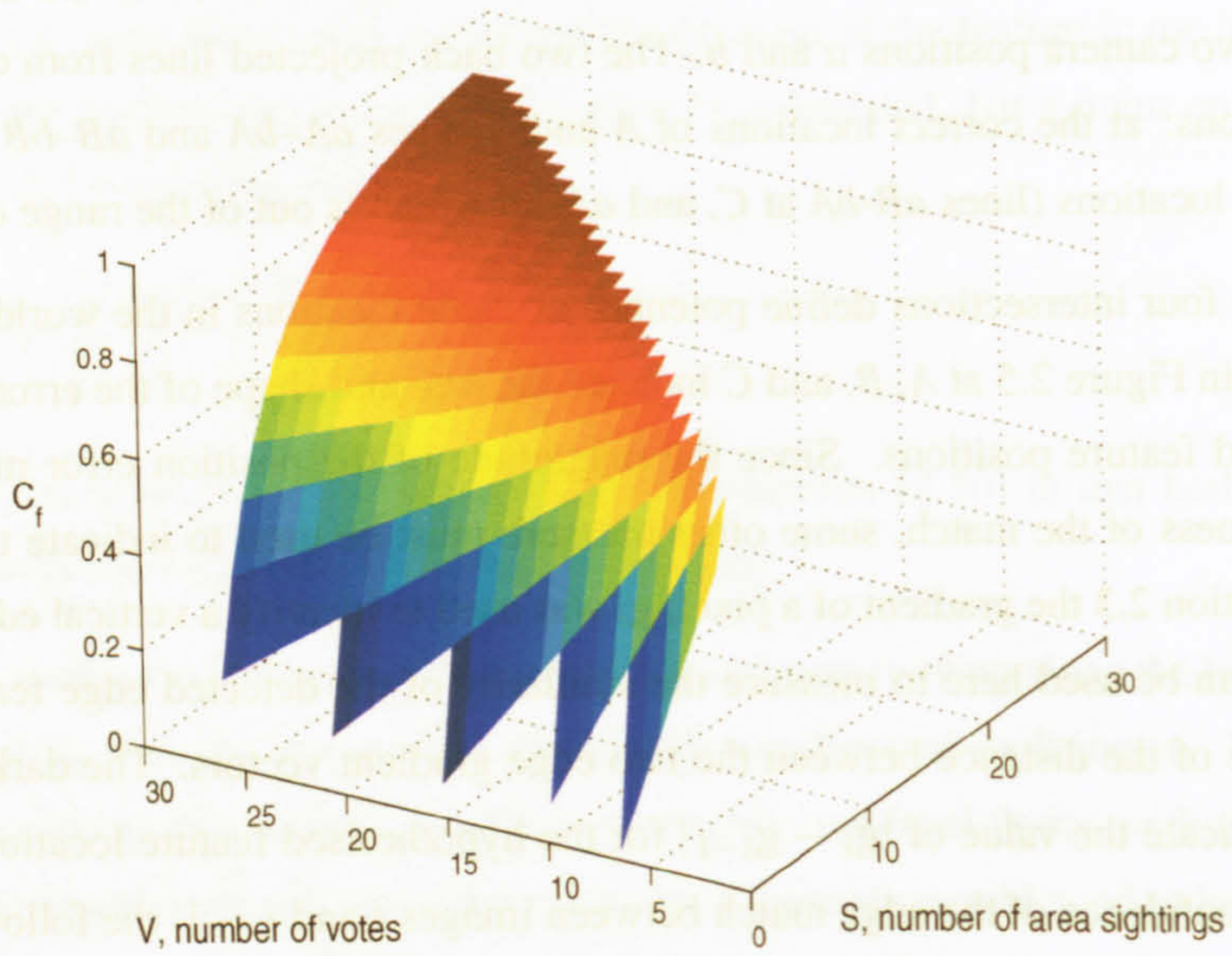


Figure 2.6: Confidence measure of a feature existing in the scene, C_f, from the number of votes V of the scene feature and the number of times S that the scene area is seen, where $S \geq V$.

[2.13]. When $V \ll S$, the confidence value is negative (not plotted in the figure). A threshold of $C_f > 0$ can be used to select features that should be inserted into the world map. When $V = S = 1$, $C_f = 0$ so the same threshold applies, since features detected once during a single view of the area should be ignored.

2.5.1 Back-projected line error model

The intersections of back-projection lines, as shown previously in Figure 2.5, indicate the approximate position of the feature in the world. By taking into account the errors on the position of the cameras and on the image feature locations, this approximate position can be better represented. The intersections of back-projection lines that estimate the position of features in the scene are accumulated in a two-dimensional grid to build up information of the feature locations over time. This two-dimensional grid represents the entire 2D area of the scene, such that each cell covers a specific region of the scene area.

Figure 2.7 shows the situation of the two camera positions and one of the back-projection line intersections of Figure 2.5. An indication of the camera position error, σ_P is shown in pink, running parallel to the back-projection line of camera b. An indication of the image feature position error, σ_β is shown in red.

The position error ellipse shown in Figure 2.7 shows an approximate area in which the position of the back-projected feature may exist. For clarity, errors from camera position a are not shown. Using

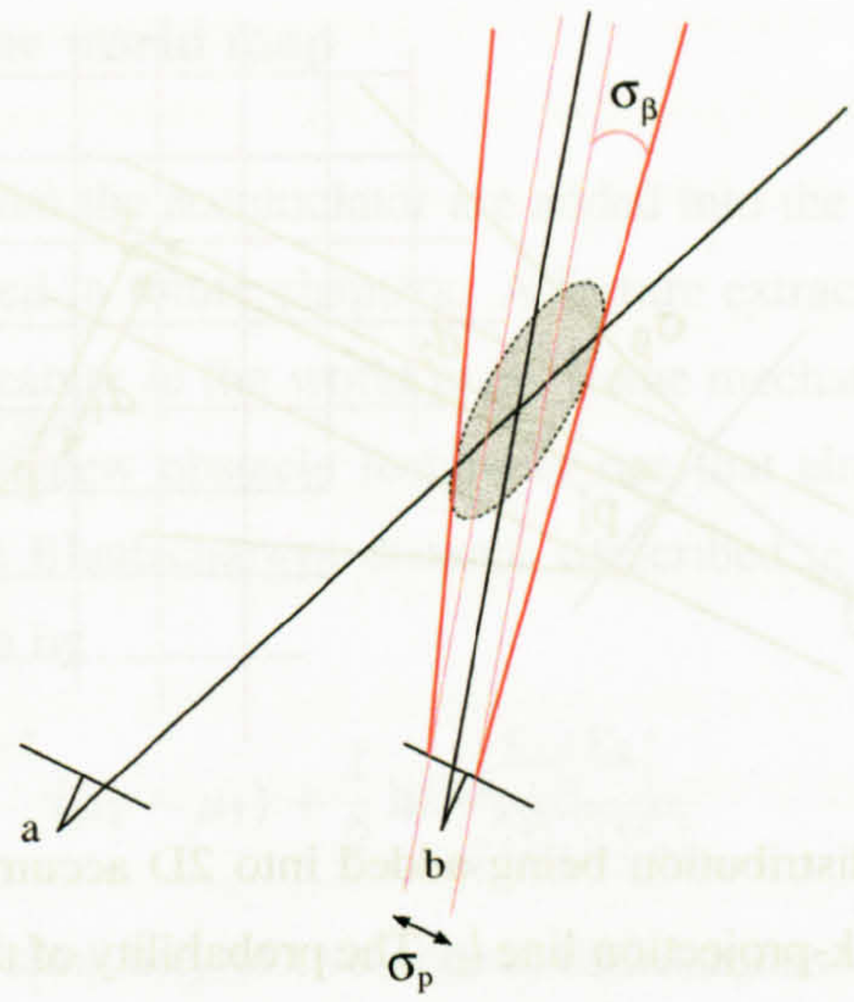


Figure 2.7: An indication of the error on the position of a back-projected feature that stem from errors on the camera positions and image feature positions; the position error of the robot σ_P and angular error on the image feature position σ_β indicate the area in which the obstacle feature may exist.

the information of both errors from both cameras, a distribution at the location of the reconstructed feature is added to the accumulator. This has the effect of spreading out the impact of a single cell being updated, and increasing the chance of scene feature position calculations from multiple views being accumulated in the same grid cell.

The probabilistic evidence of a world feature existing at any location within the accumulator is calculated from the error distribution on the position and orientation of the back-projected lines that intersect at the location of the hypothesised world feature. Figure 2.8 describes the error distribution adopted for an image feature $\mathbf{p_i}'$ from its corresponding back-projected line l_i. Within the region of the back-projection lines intersection shown in Figure 2.7, an estimate of the probability of the feature point existing at each accumulator cell centre p_i is calculated. This calculation uses the point x (the foot of the normal vector between the cell centre p_i, and the projected line l_i) and lengths d_1 and d_2, which are all calculated geometrically. Figure 2.8 shows one such example, where a possible p_i is at the centre point of the shaded accumulator cell.

σ_P is the uncertainty in the position of the camera which can been estimated from a series of motion tests against ground-truth data, but should be taken from the estimate provided by the position estimation module discussed later in Section 2.7. σ_β is the angular uncertainty of the detected feature in the image and, because the edge detection is not to sub-pixel accuracy, is taken to be the width of one pixel. The error on the camera angle will also effect the direction of the back-projection lines. This error should be added to the pixel position error σ_β.

In Figure 2.8, the line $[x, p_i]$ is perpendicular to the back-projected line l_i. The error distribution σ_i

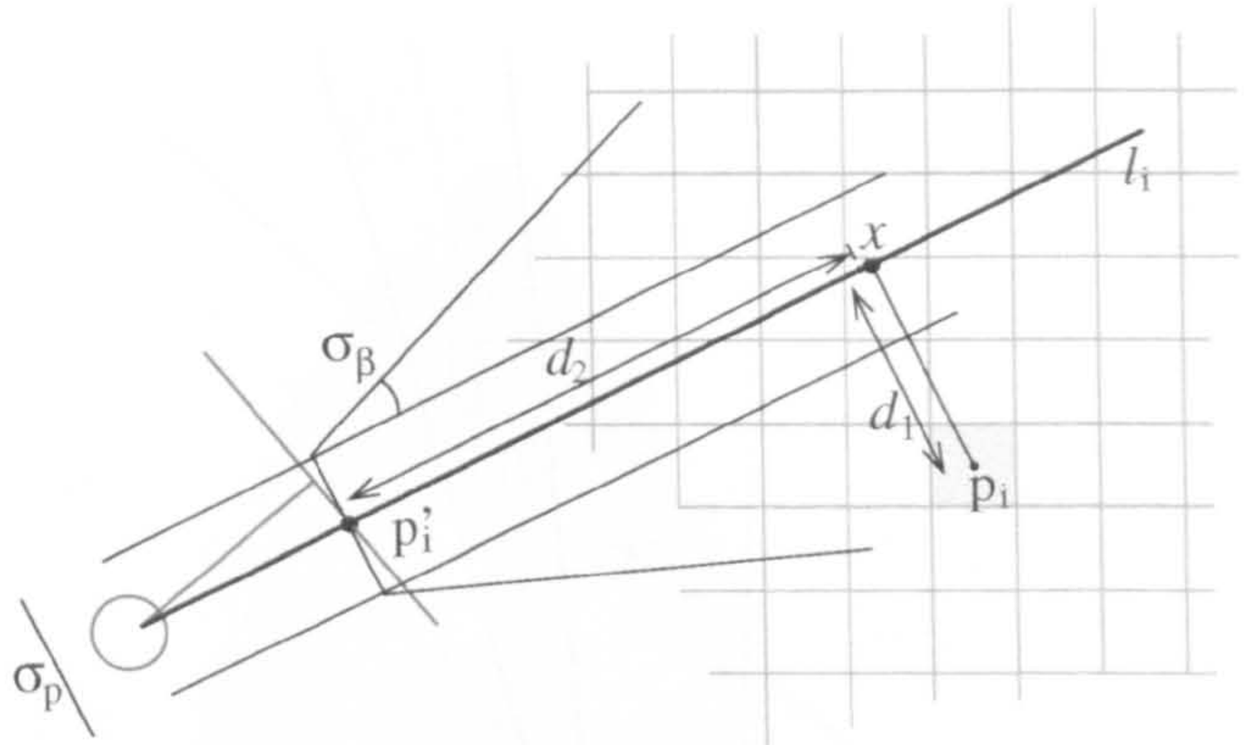


Figure 2.8: Position error distribution being added into 2D accumulator for image point $p_i{}'$ from camera position P along back-projection line l_i. The probability of the obstacle feature point existing at accumulator cell centre e.g. p_i, is calculated using the lengths d_1 and d_2 (defined from x) for each accumulator cell centre; the position error of the robot σ_P and angular error on the image feature position σ_β define the area of accumulator cells in which the obstacle feature, p_i, may exist.

of the line $[x, p_i]$ can therefore be written as

$$\sigma_i^2 = (d_2 \tan(\sigma_\beta))^2 + \sigma_P^2 \tag{2.14}$$

Assuming a Gaussian probability profile perpendicular to l_i, the probability $P(p_i)$ that the world feature exists at location p_i, is given by

$$P(p_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp(-\frac{{d_1}^2}{2{\sigma_i}^2}) \tag{2.15}$$

As the distance from the camera to the feature point increases, the width of the error distribution on the line $[x, p_i]$ increases. At the intersection of each pair of back-projected lines from two camera positions, as described in Figure 2.5, and to an area of $2.5\sigma_i$ around the intersection, the probability $P(p_i)$ is calculated from the first back-projected line. This is multiplied with the probability $P(p_i)$ for the 'matching' back-projected line from the second camera position and added to the accumulator at each p_i, weighted by the edge-gradient match confidence C_m from equation [2.12].

Each pair of back-projected line distributions added to the accumulator potentially define the location of a world feature. Over time, the accumulated distributions of detected feature locations form peaks in the accumulator grid at the best position for the feature. A threshold is applied to the accumulator to extract these locations of peak values in the accumulated distributions after each set of information from a pair of images has been added. The peaks are extracted using a non-maxima suppression method (the same technique as was used to extract peaks from the edge gradient values in Section 2.3). The error covariance is estimated from the accumulated distributions.

2.5.2 Adding points to the world map

Obstacle features extracted from the accumulator are added into the world map for analysis by the exploration algorithm described in future chapters. A feature extracted from the accumulator may already be represented by a feature in the world map. Some mechanism is needed to test whether an extracted peak represents a new obstacle feature or one that already exists in the world map. For a normal distribution, the Bhattacharyya distance (described in [Fukunaga, 1990], p99), from [Bhattacharyya, 1948] is given by

$$\frac{1}{8}(\mu_2 - \mu_1)^T \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \ln \frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \tag{2.16}$$

This distance is a measure of similarity of two normal distributions with mean positions μ_1 and μ_2 and error covariances Σ_1 and Σ_2 respectively. The first term gives the class separability of the mean difference, while the second term gives the class separability due to the covariance difference.

The maximum Bhattacharyya distance between a current peak and previously stored obstacle features is calculated, and if greater than a set threshold, the stored feature position and error covariance is updated (by multiplying the two probability density functions). This threshold is calculated as the Bhattacharyya distance of a typical world feature match for hand-matched features, and set after a series of test runs.

2.6 Results

Using the methods described in this chapter, an example image sequence acquired from the mobile robot described in Section 2.2 is considered.

Prior to the experiment, the robot is given no information about the position, size or quantity of obstacles in the environment. In this experiment the accumulator cell size is set to 25mm × 25mm, whereas [Borenstein and Koren, 1989] use a 100mm × 100mm cell size for their Occupancy grid. Borenstein uses sonar for the input data for map building which is less accurate than vision senors and since only one cell is added to for each positive obstacle detection rather than a distribution of values across cells as in this work, a larger cell area is appropriate. A distance of 100mm is travelled between each image.

Figure 2.9 shows 'snapshots' of the obstacle position information after a number of frames. The left column shows the images acquired by the robot's on-board vision system. Highlighted on the images are the edges found by the vertical edge detector.

Using the methods described earlier, each vertical edge is back-projected and line intersections incorporating the error-model of Section 2.5.1 are added to the accumulator (shown in the centre column). The image edge features which have been accumulated and exceed the accumulator threshold and are therefore ready to be added into the world map are highlighted on the accumulator at each stage. The

world map of extracted scene features is shown in the right column. Extracted features are shown by a centre point and their computed covariance. The right column also shows in solid lines the ground truth map of the room's layout and the straight line motions that the robot makes.

Subsequent accumulator states are shown along with the maps created after the peaks found in the grid are added into the map. The world feature positions are displayed by their error covariances (drawn to 2.5σ to make them easily visible). The grey-level of these ellipses are graded according to C_f, the confidence that the feature is in the world. Darker ellipses indicate a higher confidence.

At the start of the sequence, error distributions on the world feature positions are added to the accumulator. A threshold is set to ensure that until enough evidence of a feature's presence is accumulated, then that feature's information is not added into the world map. This means that at the beginning of the map building process no information is transferred from the accumulator and added into the world map. This is a major strength of the Occupancy grid mechanism. It can be seen in each of the the grids of Figure 2.9 that spurious data is accumulated in the grid due to incorrect random feature associations. This is to be expected since all possible back-projected trajectory matches are added into the accumulator. However, it is not necessary to use computationally expensive matching procedures, since the accumulator grid only builds up strong peaks where features are consistently found.

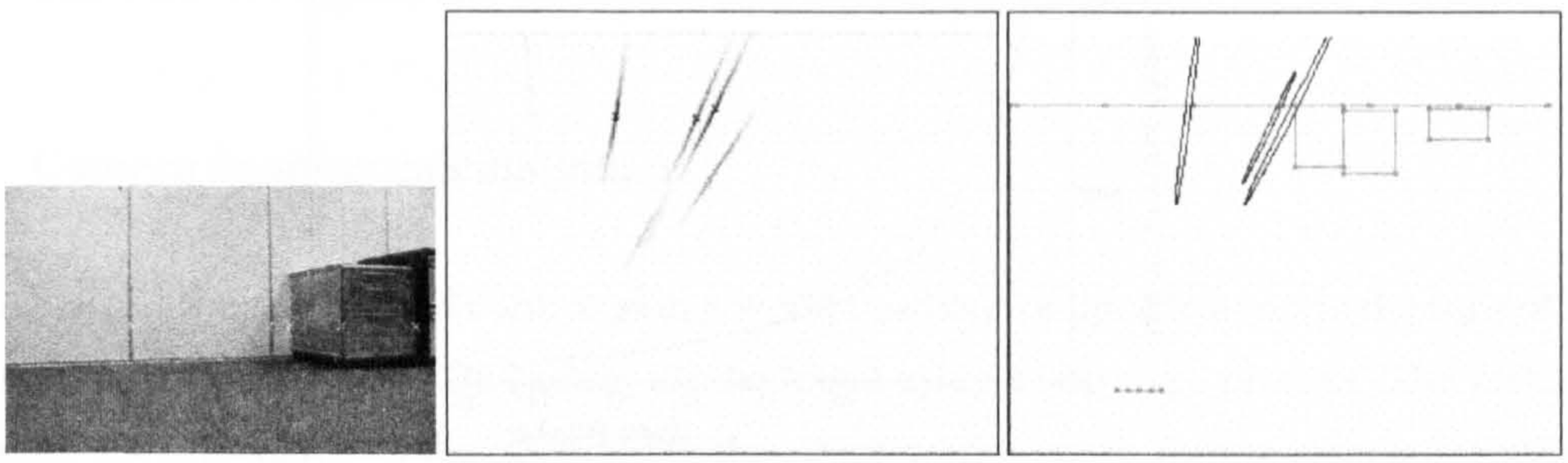
Figure 2.9(a) shows four correctly accumulated distributions, of which three have peaks highlighted. These three peaks are added into the world map. They are all new peaks since they are the first to be added into the world map. The fourth accumulated distribution, the near corner of the drawers, does not have a peak high enough to pass the threshold so is not added into the world map at this stage. In 2.9(b) the near drawer corner feature point has been viewed again and is added into the world map. Until the camera re-views a region of the scene, the distributions at that part of the accumulator remain in the accumulator but are not transferred into the world map.

As the robot moves across the scene, a peak in the accumulator becomes strongest where the rotating distributions intersect, resulting in a much localised peak. This improves the accuracy of localisation in the scene. As the sequence continues, it can clearly be seen how the robot builds up its world map from image features using the accumulator, as shown in Figure 2.9(d).

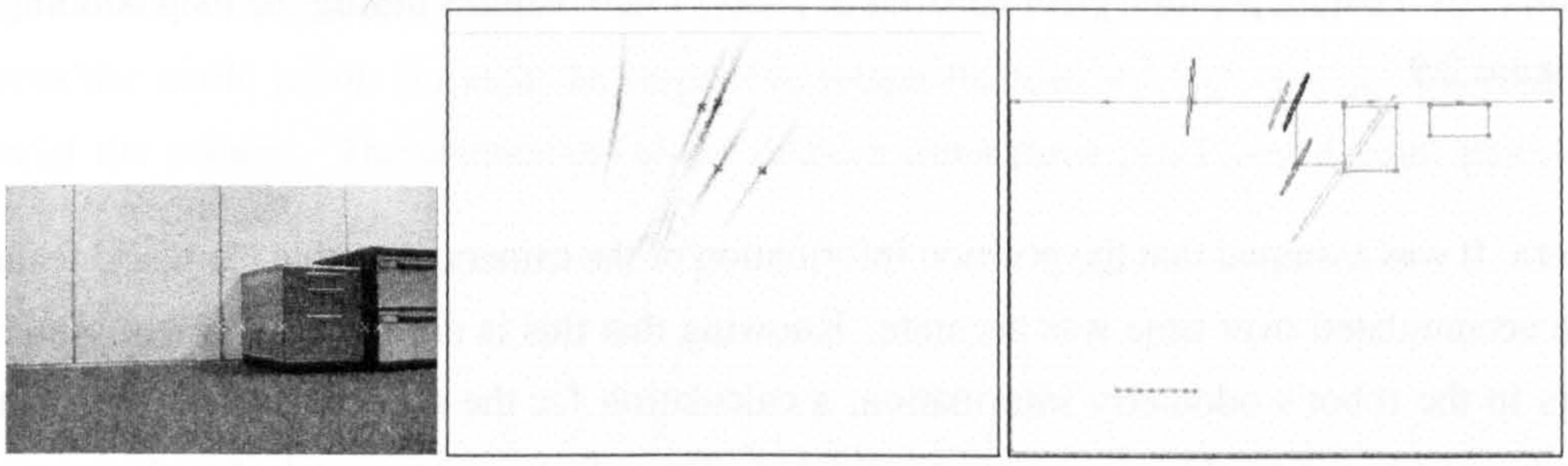
The two most frequently seen features have been traced by hand through the map building process. Figure 2.10 shows the area of the error covariance to 2.5σ of these two most frequently seen features. It can be seen that these two error covariance estimates substantially decreases over time. This relates to the sharpness of the peak in the accumulator.

For the purpose of these results, no correction of the position of the robot is done; odometry readings alone are used to back-project the world features. Due to the errors associated with odometry readings, there is a shift in the positions of features from their true position, which increases as the robot moves further from its starting position.

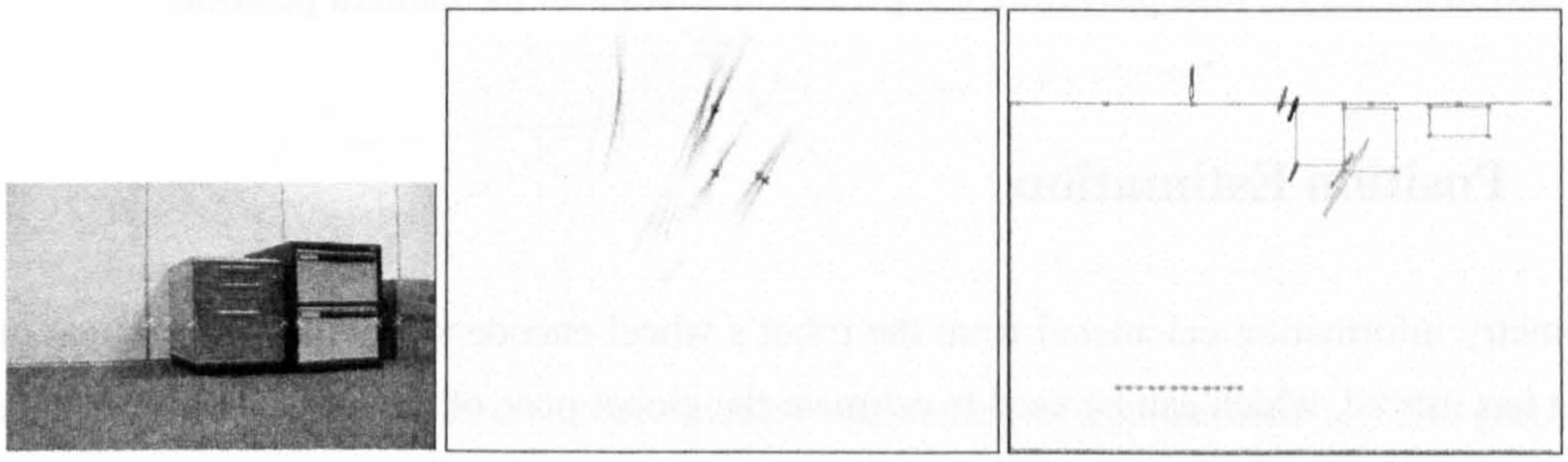
This chapter has discussed the incremental map building algorithm using a mobile robot with a single



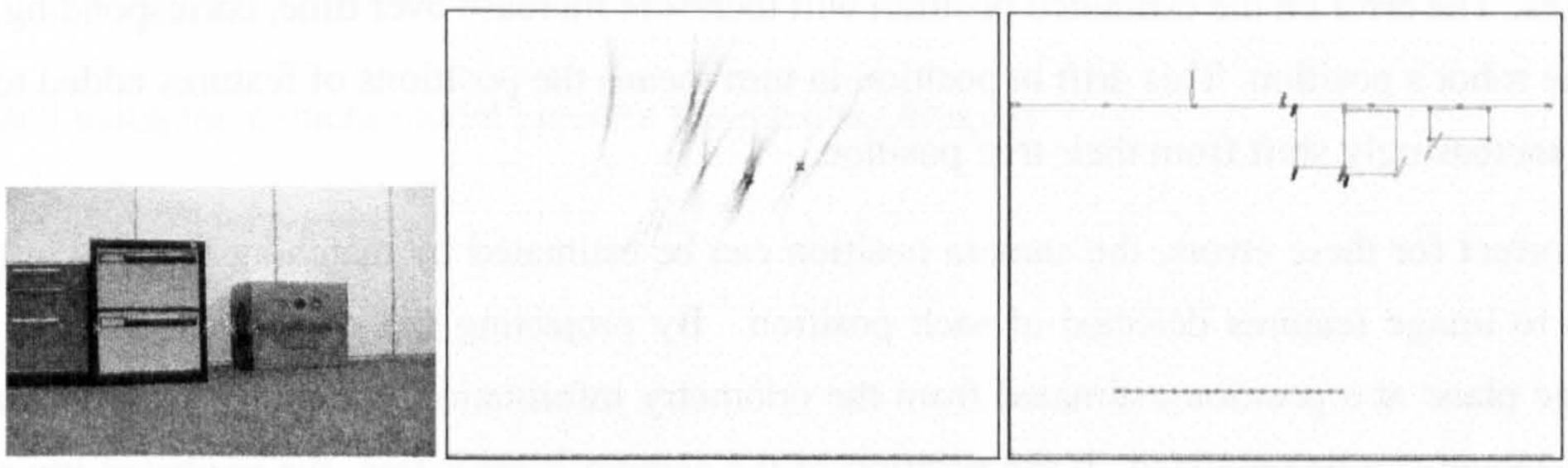
(a) System Snapshot at Frame 6



(b) System Snapshot at Frame 9



(c) System Snapshot at Frame 13



(d) System Snapshot at Frame 22

Figure 2.9: Example Robot Sequence. 1st column: current image with vertical edges detected. 2nd column: 2D accumulator with current peaks shown. 3rd column: ground truth map, overlaid with feature estimates and robot position (moving horizontally).

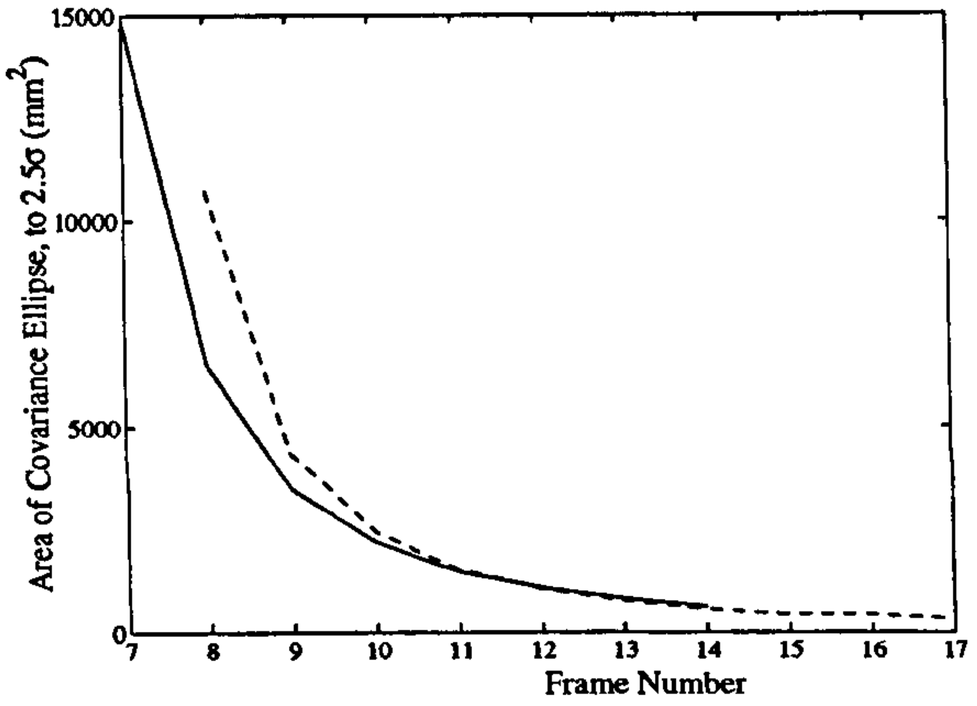


Figure 2.10: Feature position error covariance areas of two features during the map building example of Figure 2.9.

camera. It was assumed that the position information of the camera to enable the world feature points to be accumulated over time was accurate. Knowing that this is not the case in real systems due to errors in the robot's odometry information, a calculation for the robot position using features that already exist in the world map is now considered. These features are matched to the image features detected in an image, thus providing a separate calculation of the camera position.

2.7 Position Estimation

Odometry information calculated from the robot's wheel encoders provides an estimate of how the robot has moved, which can be used to estimate the global pose of the robot. However the odometry information has an error associated with it, proportional to the distance that the robot moves and rotates. The error on the estimated position will therefore increase over time, corresponding to a drift in the robot's position. This drift in position in turn means the positions of features added to the map will increasingly shift from their true position.

To correct for these errors, the camera position can be estimated by matching features in the world map to image features detected at each position. By projecting the world map features onto the image plane at a position estimated from the odometry information, the image features that should be detected can be predicted. If the position of the camera is error-free, the predicted image feature positions would exactly match those which are actually detected. However, if the camera position is not exactly known, the predicted and detected features will not exactly match. If the predicted and detected image features can be matched using their edge gradient values, the position and orientation of the camera can be computed using three pairs of world–image feature matches.

An algorithm to calculate the robot's pose using the vision system is presented. The odometry information is used to make an initial estimate of the robot's position. Features in the world are projected onto the image plane at this estimated position. Three image to world feature matches are required in

order to calculate the pose of the robot, minimising the discrepancy between detected image features and projected world features on the image plane. This pose calculation is done in two stages: the first calculates the orientation, and the second uses this orientation to calculate the position.

2.7.1 Camera heading calculation, α

Given the pixel co-ordinates and corresponding world positions of three features in the current image, the position and orientation of the camera can be found using a two stage process. The first stage is to calculate the orientation of the camera. For this, three world–image feature pairs are required.

Figure 2.11 shows three world feature points stored in the world map, p_1, p_2, and p_3. The projection lines from the world points through the respective image features p_1', p_2', and p_3' intersect at the position of the camera. The orientation of the camera using three pixel–world point pairs can be

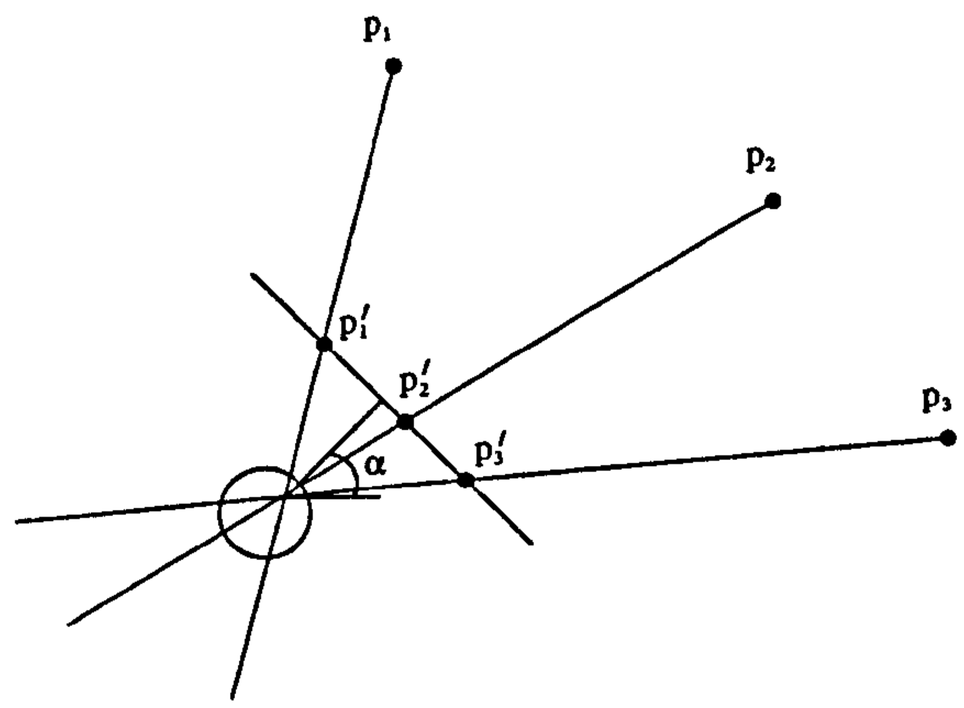


Figure 2.11: Three world feature points p_i, with corresponding image features p_i'. The green circle depicts the robot position, with offset camera at position $[t_x t_y]^T$ and orientation α.

calculated using the camera model given in the previous chapter

$$\mathbf{p_1}' = \mathbf{K[R|T]p_1} \tag{2.17}$$

$$\mathbf{p_2}' = \mathbf{K[R|T]p_2} \tag{2.18}$$

$$\mathbf{p_3}' = \mathbf{K[R|T]p_3} \tag{2.19}$$

Subtracting $\mathbf{p_1}'$ from the other two points gives the following two equations, which eliminates the translation component $\mathbf{T}$.

$$\begin{bmatrix} u_2 \\ w_2 \end{bmatrix} - \begin{bmatrix} u_1 \\ w_1 \end{bmatrix} = \mathbf{KR[p_2 - p_1]} \tag{2.20}$$

$$\begin{bmatrix} u_3 \\ w_3 \end{bmatrix} - \begin{bmatrix} u_1 \\ w_1 \end{bmatrix} = \mathbf{KR[p_3 - p_1]} \tag{2.21}$$

where the pixel $\mathbf{p_i}' = [u_i\ w_i]^T$ is a normalised homogeneous co-ordinate of the pixel co-ordinate along the horizontal scan line, $X_i' = u_i/w_i$. $p_i = [x_i\ y_i]^T$ is the world point in two dimensions. these equations can be re-written such that

$$w_2 \begin{bmatrix} X_2' \\ 1 \end{bmatrix} = w_1 \begin{bmatrix} X_1' \\ 1 \end{bmatrix} + \mathbf{KR} \begin{bmatrix} \triangle x_2 \\ \triangle y_2 \end{bmatrix} \tag{2.22}$$

$$w_2 \begin{bmatrix} X_3' \\ 1 \end{bmatrix} = w_1 \begin{bmatrix} X_1' \\ 1 \end{bmatrix} + \mathbf{KR} \begin{bmatrix} \triangle x_3 \\ \triangle y_3 \end{bmatrix} \tag{2.23}$$

where

$$\triangle x_2 = x_2 - x_1 \quad , \quad \triangle y_2 = y_2 - y_1 \tag{2.24}$$

$$\triangle x_3 = x_3 - x_1 \quad , \quad \triangle y_3 = y_3 - y_1 \tag{2.25}$$

By simplifying

$$\mathbf{K^{-1}} \begin{bmatrix} X_i' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{K_u f} & -\frac{U_0}{K_u f} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_i' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{X_i' - U_0}{K_u f} \\ 1 \end{bmatrix} = \begin{bmatrix} k_i \\ 1 \end{bmatrix} \tag{2.26}$$

Using k_i, the two equations can be re-written such that

$$w_1 \begin{bmatrix} k_1 \\ 1 \end{bmatrix} = w_2 \begin{bmatrix} k_2 \\ 1 \end{bmatrix} - \mathbf{R} \begin{bmatrix} \triangle x_2 \\ \triangle y_2 \end{bmatrix} \tag{2.27}$$

$$w_1 \begin{bmatrix} k_1 \\ 1 \end{bmatrix} = w_3 \begin{bmatrix} k_3 \\ 1 \end{bmatrix} - \mathbf{R} \begin{bmatrix} \triangle x_3 \\ \triangle y_3 \end{bmatrix} \tag{2.28}$$

With $\mathbf{R} = \begin{bmatrix} \sin\alpha & -\cos\alpha \\ \cos\alpha & \sin\alpha \end{bmatrix}$, equations [2.27] and [2.28] are expanded and simplified to

$$w_1 k_1 = k_2 w_1 + (k_2 \triangle y_2 - \triangle x_2)\sin\alpha + (k_2 \triangle x_2 + \triangle y_2)\cos\alpha \tag{2.29}$$

$$w_1 k_1 = k_3 w_1 + (k_3 \triangle y_3 - \triangle x_3)\sin\alpha + (k_3 \triangle x_3 + \triangle y_3)\cos\alpha \tag{2.30}$$

This can be expressed in matrix form such that

$$w_1 \begin{bmatrix} k_1 - k_2 \\ k_1 - k_3 \end{bmatrix} = \begin{bmatrix} k_2 \triangle y_2 - \triangle x_2 & k_2 \triangle x_2 + \triangle y_2 \\ k_3 \triangle y_3 - \triangle x_3 & k_3 \triangle x_3 + \triangle y_3 \end{bmatrix} \begin{bmatrix} \sin\alpha \\ \cos\alpha \end{bmatrix} \tag{2.31}$$

$$w_1 \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} \sin\alpha \\ \cos\alpha \end{bmatrix} \tag{2.32}$$

Defining $H = S_1/S_2$, the solution for α is such that

$$\alpha = \tan^{-1}\left(\frac{Q_{12} - HQ_{22}}{HQ_{21} - Q_{11}}\right) \tag{2.33}$$

2.7.2 Camera position calculation, $\begin{bmatrix} t_x & t_y \end{bmatrix}^T$

With the orientation of the camera known, the location of the camera can be calculated from two world points and their corresponding pixel information

$$\mathbf{p_1}' = \mathbf{K[R|T]p_1} \tag{2.34}$$

$$\mathbf{p_2}' = \mathbf{K[R|T]p_2} \tag{2.35}$$

Again using the equation [2.26] for k_i, these calculations can be rearranged and expanded such that for two world-pixel point pairs

$$k_1 \cos\alpha(x_1 - t_x) + k_1 \sin\alpha(y_1 - t_y) = (x_1 - t_x)\sin\alpha - (y_1 - t_y)\cos\alpha \tag{2.36}$$

$$k_2 \cos\alpha(x_2 - t_x) + k_2 \sin\alpha(y_2 - t_y) = (x_2 - t_x)\sin\alpha - (y_2 - t_y)\cos\alpha \tag{2.37}$$

Which can be expressed in matrix form

$$\begin{bmatrix} k_1(x_1\cos\alpha + y_1\sin\alpha) - x_1\sin\alpha + y_1\cos\alpha \\ k_2(x_2\cos\alpha + y_2\sin\alpha) - x_2\sin\alpha + y_2\cos\alpha \end{bmatrix} = \begin{bmatrix} k_1\cos\alpha - \sin\alpha & k_1\sin\alpha + \cos\alpha \\ k_2\cos\alpha - \sin\alpha & k_2\sin\alpha + \cos\alpha \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{2.38}$$

And this can be re-written such that

$$\begin{bmatrix} -x_1 + k_1 y_1 & k_1 x_1 + y_1 \\ -x_2 + k_2 y_2 & k_2 x_2 + y_2 \end{bmatrix} \begin{bmatrix} \sin\alpha \\ \cos\alpha \end{bmatrix} = \begin{bmatrix} k_1 & 1 \\ k_2 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{2.39}$$

and this is rearranged to find a solution for the position of the camera

$$\begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}^{-1} \begin{bmatrix} k_1 & 1 \\ k_2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -x_1 + k_1 y_1 & k_1 x_1 + y_1 \\ -x_2 + k_2 y_2 & k_2 x_2 + y_2 \end{bmatrix} \begin{bmatrix} \sin\alpha \\ \cos\alpha \end{bmatrix} \tag{2.40}$$

With $N \geq 3$ world point and corresponding image pixel pairs, it is possible to calculate ${}_N C_3$ position and orientations for the camera. These solutions are not independent. The mean shift algorithm described in the calibration section can be used to get the best estimate of the position and orientations.

Using the odometry information as a first estimate of the current position of the robot, the above calculations can be used to update the position of the robot. A position estimation calculation can therefore be implemented without the use of a Kalman Filter.

2.8 Conclusions

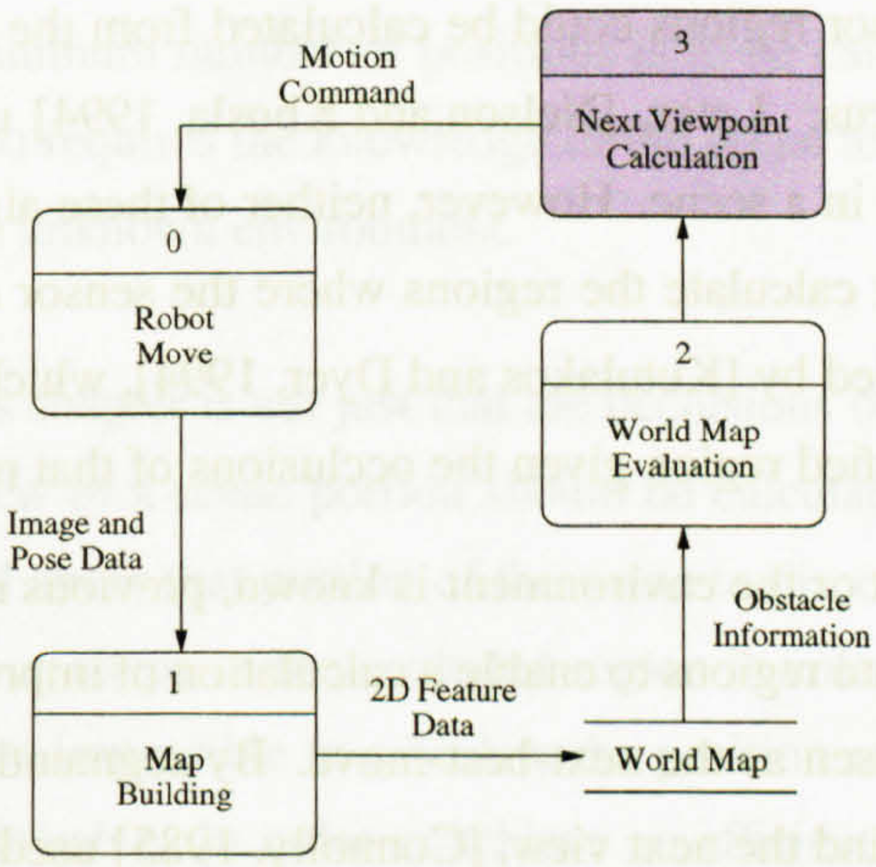
In this chapter, a map building framework for a mobile robot with a single camera has been presented. The novelty of the method lies in a probabilistic approach to the accumulator and better localisation

of feature points. Vision information is used to gather evidence of the existence of features in the environment without requiring the positive matching of features. Located features are accurate and robust to errors. The accumulator effectively reduces the covariance of the estimates by integrating vision data over time.

In the next chapter an algorithm to calculate the next-best-viewpoint given an obstacle in the world map is introduced. This is the first step towards an autonomous exploration algorithm: evaluating the incrementally built world map for exploration purposes.

Chapter 3

Viewpoint Planning Strategy



In the previous chapter a map building framework for a moving camera in a static scene was introduced. The obstacle location map was incrementally built as the robot moved through the scene. Traversable openings are formed between obstacle feature pairs. This chapter considers the calculation for the next-best-viewpoint for a camera through an opening to effect both exploration and environment model construction. A utility function is developed for the best-view around one corner of an obstacle and extended for the best-view through an opening formed by two such corners. An example exploration of a two-dimensional map is shown, and a measure of the stability of the next-best-viewpoint calculation is discussed.

3.1 Introduction

Exploration algorithms have been developed to explore an unknown environment using methods such as the lawnmower, [Arkin et al., 1993], and wall-following, [Duckett and Nehmzow, 1997], algorithms (discussed in more detail in Chapter 1). Such exhaustive methods are not efficient. Where the location of obstacles in the environment are not fully known and for the purposes of exploration,

the best point from which to view the unknown parts of the scene needs to be calculated. This implies, within a certain range of movement, an algorithm to guide the robot to an optimal viewpoint is required.

It is unlikely for a single viewing position to allow the entire environment to be viewed all at once, but a series of optimal viewpoints each providing new information could lead to an exploration of a complex scene. Generally the goal of a next-best-view system is for model reconstruction, such that each viewpoint improves the knowledge of the object or the scene. For example the repeated readings of previously modelled features, or acquiring data of previously unseen features.

Considering an object being viewed by a sensor (camera, LRF or sonar), there are an infinite number of sensor positions from which the object can be viewed. An important question is "Where is the 'best' or 'optimal' position to view the object from?" The answer to this question depends upon the purpose of the system. To restrict the regions for the sensor to be located, [Cowan and Kovesi, 1988] showed how allowable sensor regions could be calculated from the sensor constraints such as field-of-view, resolution and focus. Later, [Nelson and Khosla, 1994] used such constraints to track a moving object dynamically in a scene. However, neither of these algorithms calculate the *next* view for the sensor. Instead they calculate the regions where the sensor *could* be placed. The viewpoint adjustment problem is tackled by [Kutulakos and Dyer, 1994], which requires a camera to be moved to provide a view of a specified region given the occlusions of that region, but is not optimal.

Where the size of the object or the environment is known, previous research has focused on splitting the object or environment into regions to enable a calculation of improvement for that sensor location, and the best location is chosen as the next-best-move. By segmenting the object into equal regions and using these regions to find the next view, [Connolly, 1985] used an oct-tree to describe the environment surrounding an object, rather than the object itself, where the depth of the tree showed the least viewed area. [Maver and Bajcsy, 1993] however realised that the occlusion boundary could be used to define the next-viewpoint. They used a histogram method to represent the number of polygons visible from each viewpoint, whereas [Whaite and Ferrie, 1991] used a local improvement map to define the next sensor placement. Although these algorithms do successfully improve the knowledge of the scene, they do not guarantee a completely optimal viewing position. Chapter 4 uses the idea of splitting the environment into regions to measure the improvement of the viewing position calculated in this chapter. The calculation of the next-best-viewpoint is considered separately from the anticipated improvement. Instead, only the viewing angle of the opening is optimised.

The corners of obstacles which form occlusions to regions of interest in the scene have previously been considered for use with navigation algorithms to direct the motion of a mobile robot such as [Murray et al., 1996]. Murray uses the corners of obstacles to steer a mobile robot such that a single corner with a set distance constraint is used to provide a safe navigation as shown in Figure 3.1. The work described in this chapter is intended to find an optimal viewing solution using corners rather than just moving around them.

Another problem requiring the calculation of sensor placements is the well known Art Gallery Prob-

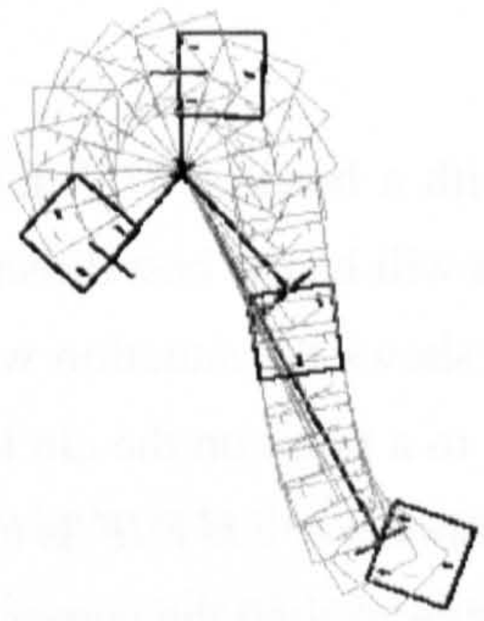


Figure 3.1: Navigation using corners, [Murray et al., 1996], Figure 3. To navigate, the robot maintains a fixed distance away from the corner.

lem, [Chvátal, 1975]. The minimum number of positions is to be calculated from which the entire scene will be visible. This also requires the knowledge of the scene to be very precise, and does not account for the situation of an unknown environment.

The problem addressed in this chapter is not just that the occlusions of a shape or a scene should be removed, but that the next view of a scene portion should be calculated such that the *most* amount of information can be attained about that portion of the scene to discover whether the area warrants further exploration. The size or shape of the occluded region is unknown, in fact there may be no occluded region. It is therefore impossible to calculate the position of the camera that would allow for the unseen region to be viewable. In order to achieve an efficient exploration, new information about the scene should be used as it is acquired.

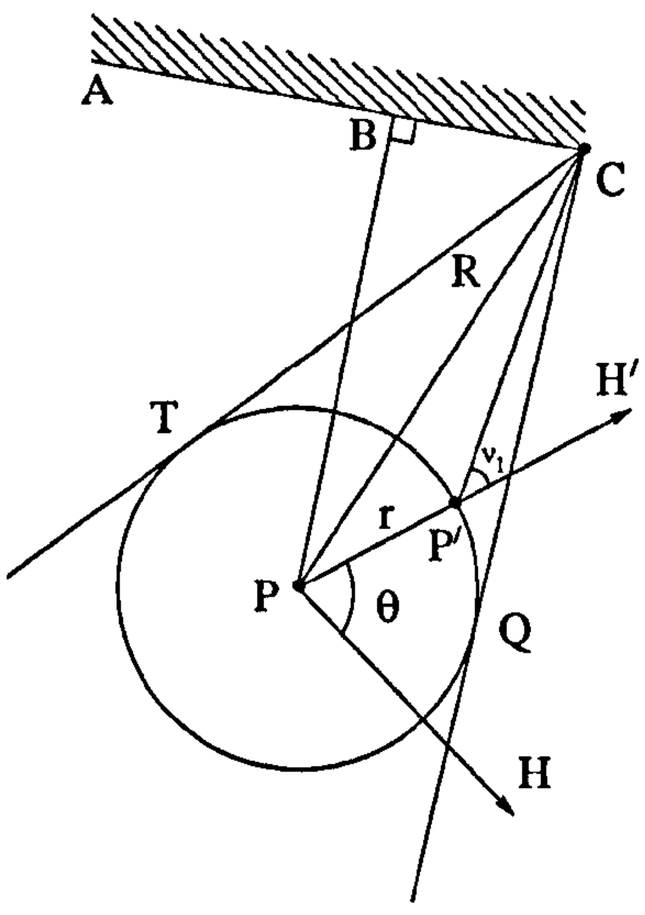
The obstacle map constructed using the algorithms described in Chapter 2 provides the location of obstacle boundaries in the scene. The camera on the robot should be moved to a position that will improve the view through openings, with the intention of gaining the greatest amount of new information through the opening. By fixing the distance to travel, this problem can be quantified by a utility function for a given opening and maximised providing the best view through the opening.

In section 3.2 the simplest problem of one corner being visible is considered. A utility function is proposed to indicate the next-best-viewing position for the camera. This is then extended to the situation with an opening formed by two corners in section 3.3, and finally generalised to any number of openings. Section 3.5 demonstrates the proposed algorithm with an example exploration of an initially unknown maze, and section 3.6 presents an error analysis on the maximised utility functions.

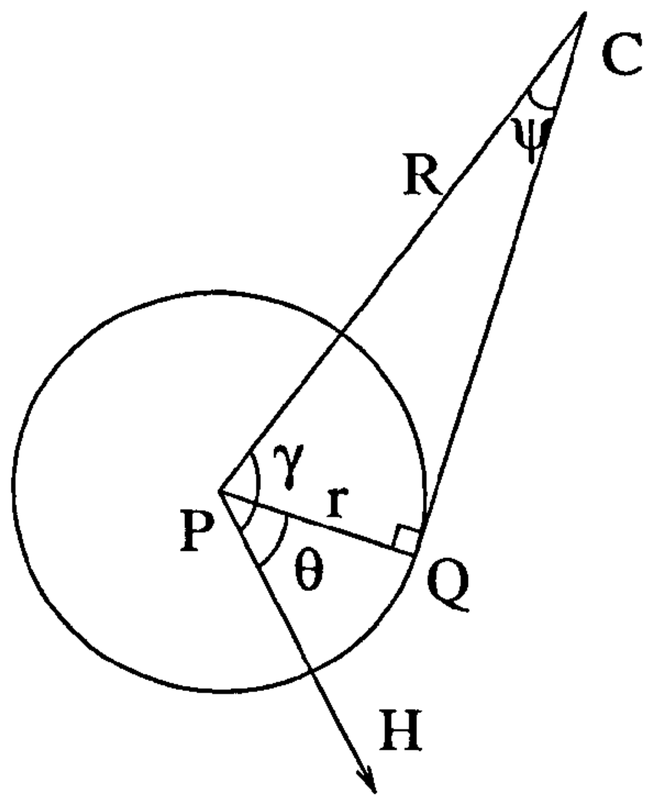
For the next-best-view calculations that follow, $r < R$ i.e. the camera can not move past the corner or through the opening being considered. The next-best-view is to calculate the best view through an opening or around a corner for the purposes of exploration.

3.2 The One Corner Case

Consider a robot at a position P with a heading PH and a wall ABC in its vicinity. If a linear move of distance r is executed, what will be the best direction to move in so as to give the best view around the corner C? Figure 3.2(a) shows the situation with a corner C in the robots vicinity. The robot moves from P by a distance r to a point on the circle $TP'QH$ where P' is the position of the next-best-view. The rotation of the robot $\theta = \angle HPH'$ is to be calculated and the robot to move by a distance r in order to maximise the view around the corner C. Consider the corner C and the general



(a) One corner case.



(b) Optimal solution for one corner case.

Figure 3.2: Viewpoint planning for one corner. An obstacle ABC forms one corner at C in the vicinity of the robot at P. The direction to move is defined by θ, using a fixed distance r. P' is the position at which the viewing angle ν_1 is optimised.

point P' on the circle $TP'Q$ in Figure 3.2(a). As the point P' moves around this circle, the line CP' will oscillate between CT and CQ, which are both tangential to the motion circle. The point Q is clearly the optimal location to give the best view around the corner C, shown in Figure 3.2(b).

With reference to the optimal solution of Figure 3.2(b), let

$$\angle HPC = \gamma \quad \text{and} \quad \angle PCQ = \psi$$

Since $\angle PQC$ is a right angle,

$$\begin{aligned} \angle QPC &= \pi/2 - \psi \\ \gamma &= \theta + \pi/2 - \psi \\ \Longrightarrow \gamma - \theta &= \pi/2 - \psi \end{aligned}$$

Taking the cosine of each side,

$$\begin{aligned} \cos(\gamma-\theta) &= \cos(\pi/2-\psi) \\ &= \sin\psi \\ \Longrightarrow \cos(\gamma-\theta) &= \frac{r}{R} \end{aligned} \tag{3.1}$$

This is therefore the optimal solution, where R is the distance from the robot to the corner and r is the move distance. The value of R can be calculated from the Euclidean distance from P to C.

As $r \rightarrow R$, $\theta \rightarrow \gamma$ and the robot moves directly towards the corner. However, as $r \rightarrow |BC|$, then $\sin\psi = \frac{|BC|}{R} = \sin(\angle BPC)$. Hence the robot moves parallel to the wall with CQ perpendicular to the wall.

Having determined the optimal solution, a utility function will now be defined that this solution maximises.

3.2.1 The utility function for the one corner case

Suppose the robot moves from P to P' with the new heading H'. The viewing angle ν_1 is given by $\angle H'P'C$, where the original viewing angle was $\angle HPC$.

$$\nu_1 = \angle H'P'C = \psi + \angle P'PC$$

However

$$\angle HPC = \gamma \Longrightarrow \angle P'PC = \gamma - \theta$$

Therefore the viewing angle ν_1 can be defined as

$$\nu_1 = \psi + \gamma - \theta \tag{3.2}$$

γ is a constant value for a given corner and robot position, whereas ψ is a function of θ. The angle ψ must therefore be maximised by the choice of θ to maximise the viewing angle ν_1.

From the triangle $PP'C$ in Figure 3.3, where $P'a$ is perpendicular to PC, ψ can be defined as

$$\tan\psi = \frac{P'a}{aC}$$

where

$$P'a = r\sin(\gamma-\theta)$$

and

$$\begin{aligned} aC &= R - Pa \\ &= R - r\cos(\gamma-\theta) \end{aligned}$$

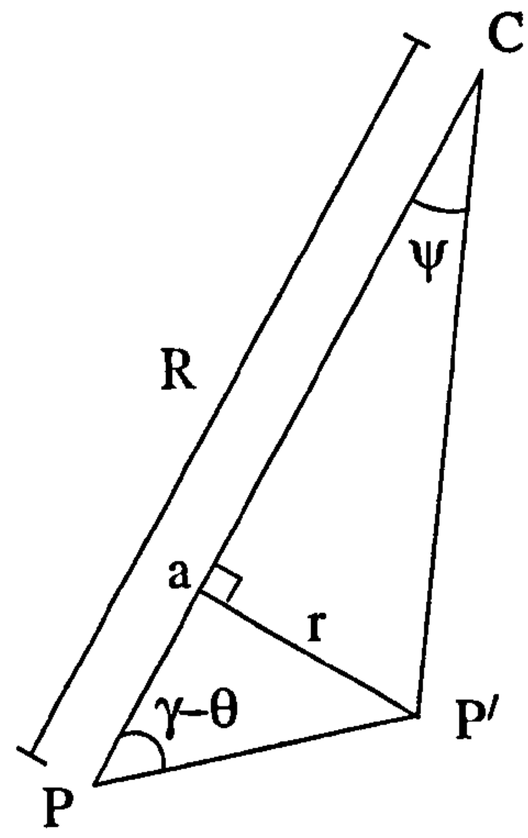


Figure 3.3: Triangle $PP'C$: generating the utility function for one corner.

the angle ψ can therefore be defined by

$$\begin{aligned} J_1 = \tan\psi &= \frac{r\sin(\gamma-\theta)}{R - r\cos(\gamma-\theta)} \\ &= \frac{r\sin\phi}{R - r\cos\phi} \quad \text{where} \quad \phi = \gamma - \theta \end{aligned} \tag{3.3}$$

This provides an equation for ψ as a function of θ which needs to be maximised to ensure that this is a solution for the maximum value of ν_1.

3.2.2 The optimal solution of the utility function

To find a solution to the maximum value of the utility function of equation [3.3], it is differentiated with respect to ϕ.

$$\begin{aligned} \sec^2\psi\frac{\mathrm{d}\psi}{\mathrm{d}\phi} &= \frac{(R - r\cos\phi)r\cos\phi - r\sin\phi\, r\sin\phi}{(R - r\cos\phi)^2} \\ &= \frac{Rr\cos\phi - r^2(\cos^2\phi + \sin^2\phi)}{(R - r\cos\phi)^2} \\ \sec^2\psi\frac{\mathrm{d}\psi}{\mathrm{d}\phi} &= \frac{Rr\cos\phi - r^2}{(R - r\cos\phi)^2} \end{aligned} \tag{3.4}$$

To devise an alternative equation for $\sec^2\psi$, the standard equation [$\sec^2 x = 1 + \tan^2 x$] and the solution of equation [3.3] is used to show

$$\begin{aligned}
\sec^2\psi &= 1 + \frac{r^2\sin^2\phi}{(R - r\cos\phi)^2} \\
&= \frac{(R - r\cos\phi)^2 + r^2\sin^2\phi}{(R - r\cos\phi)^2} \\
&\quad R^2 - 2Rr\cos\phi + r^2\cos^2\phi + r^2\sin^2\phi \\
\sec^2\psi &= \frac{R^2 - 2Rr\cos\phi + r^2}{(R - r\cos\phi)^2} \qquad (3.5)
\end{aligned}$$

Equation [3.5] can then be substituted into the differential equation [3.4] to find a solution to the derivative of the utility function.

$$\begin{aligned}
\frac{R^2 - 2Rr\cos\phi + r^2}{(R - r\cos\phi)^2}\frac{\mathrm{d}\psi}{\mathrm{d}\phi} &= \frac{Rr\cos\phi - r^2}{(R - r\cos\phi)^2} \\
\frac{\mathrm{d}\psi}{\mathrm{d}\phi} &= \frac{Rr\cos\phi - r^2}{R^2 + r^2 - 2Rr\cos\phi} = 0 \quad \text{for a maximum} \qquad (3.6)
\end{aligned}$$

Setting the final differential equation [3.6] to zero provides the optimal solution,

$$\begin{aligned}
Rr\cos\phi - r^2 &= 0 \\
\cos\phi &= \frac{r}{R} \qquad (3.7)
\end{aligned}$$

This is the same optimal solution as shown previously in equation [3.1].

If this maximum solution [3.7] is substituted into the utility function of equation [3.3] an alternative equation for $\tan\psi$ can be found.

$$\begin{aligned}
\tan\psi &= \frac{r\sin\phi}{R - r\cos\phi} \\
&= \frac{r\sqrt{1 - \frac{r^2}{R^2}}}{R - \frac{r^2}{R}} \\
\tan\psi &= \frac{r}{\sqrt{R^2 - r^2}} \qquad (3.8)
\end{aligned}$$

This equation can be expressed as a triangle of Figure 3.4 with the third side calculated from Pythagoras' theorem.

By considering the viewing angle ν_1 derived in equation [3.2] as $\nu_1 = \psi + \phi$, and combined with

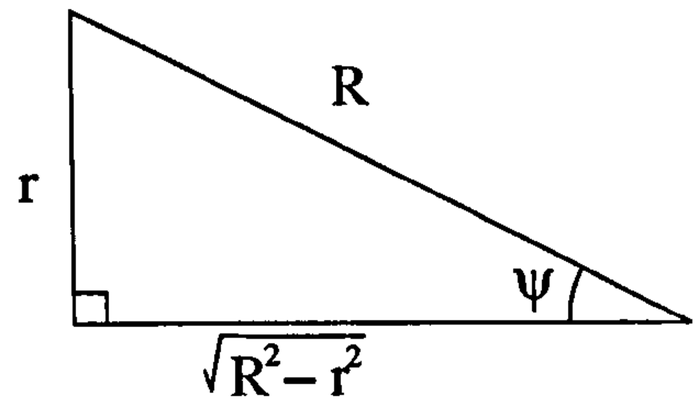


Figure 3.4: Triangle of optimal solution for the one corner case, equation [3.8].

this optimal solution,

$$\begin{aligned}
\cos(\nu_1) &= \cos(\psi + \phi) \\
&= \cos\psi\cos\phi - \sin\psi\sin\phi \\
&= \frac{\sqrt{R^2 - r^2}}{R}\frac{r}{R} - \frac{r}{R}\sqrt{1 - \frac{r^2}{R^2}} \\
&= \frac{r\sqrt{R^2 - r^2}}{R} - \frac{r\sqrt{R^2 - r^2}}{R} \\
&= 0
\end{aligned}$$

This implies that $\gamma = \pi/2$, which confirms that the optimal location for the next-best-viewpoint for the camera is at the tangent of the motion circle with the corner, and was found by using the utility function given in equation [3.3].

3.3 The Two Corner Case

Consider a robot located at point P with heading PH and two walls in the vicinity forming an opening between them. The point P' on the motion circle of radius r is to be determined, which maximises the viewing angle ν_2 through the aperture.

Figure 3.5 shows the situation of the robot at P with the two corners C_1 and C_2. The robot moves from P by a distance r to a point P' so that the viewing angle through the opening is a maximum. The rotation of the robot θ is to be calculated and the robot to move by a distance r to provide the next-best-view position.

3.3.1 The utility function of the two corner case

From Figure 3.5 an expression for the viewing angle at P' can be determined. From triangle C_1PP',

$$\begin{aligned}
\angle P'PC_1 &= \gamma_1 - \theta \\
\angle PP'C_1 &= \pi - \psi_1 - (\gamma_1 - \theta)
\end{aligned} \tag{3.9}$$

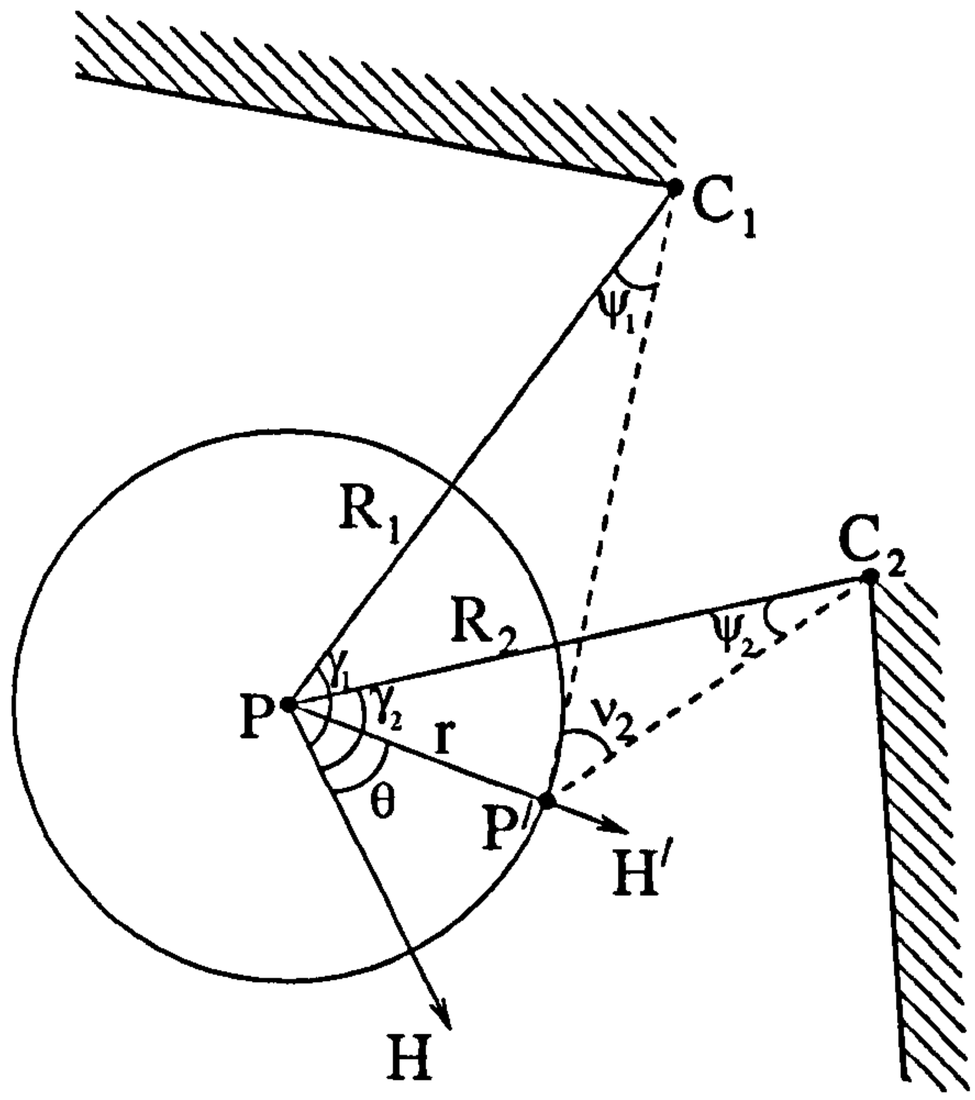


Figure 3.5: Viewpoint planning for two corners that form an opening between them. Two obstacles that form corners C_1 and C_2 in the vicinity of the robot at P. The direction to move is defined by θ, using a fixed distance r. P' is the position at which the viewing angle ν_2 is optimised.

From triangle C_2PP',

$$\begin{aligned} \angle P'PC_2 &= \gamma_2 - \theta \\ \angle PP'C_2 &= \pi - \psi_2 - (\gamma_2 - \theta) \end{aligned} \tag{3.10}$$

The difference between the two angles of [3.9] and [3.10] is the new viewing angle through the aperture, $C_1P'C_2$. The corners are numbered in a clockwise direction.

$$\begin{aligned} \nu_2 &= \angle PP'C_2 - \angle PP'C1 \\ &= [\pi - \psi_2 - (\gamma_2 - \theta)] - [\pi - \psi_1 - (\gamma_1 - \theta)] \\ &= (\psi_1 - \psi_2) + (\gamma_1 - \gamma_2) \end{aligned} \tag{3.11}$$

$(\gamma_1 - \gamma_2)$ was the viewing angle at the robot position P and is fixed for the situation. Therefore if $(\psi_1 - \psi_2)$ is maximised, the maximum view through the aperture at P' can be found. Following from the one corner case, the utility function for an opening is taken to be

$$J_2 = \tan(\psi_1 - \psi_2) \tag{3.12}$$

Again, this utility function is differentiated to find a maximum for the function.

3.3.2 The optimal solution of the utility function

To find the maximum of the utility function of equation [3.12], this is differentiated with respect to θ.

$$\frac{\mathrm{d}J_2}{\mathrm{d}\theta} = \sec^2(\psi_1 - \psi_2)\left(\frac{\mathrm{d}\psi_1}{\mathrm{d}\theta} - \frac{\mathrm{d}\psi_2}{\mathrm{d}\theta}\right) \tag{3.13}$$

$$= \sec^2(\psi_1 - \psi_2)\cos^2\psi_1\cos^2\psi_2\left(\sec^2\psi_2\sec^2\psi_1\frac{\mathrm{d}\psi_1}{\mathrm{d}\theta} - \sec^2\psi_1\sec^2\psi_2\frac{\mathrm{d}\psi_2}{\mathrm{d}\theta}\right) \tag{3.14}$$

From the one corner case at [3.4]

$$\sec^2\psi\frac{\mathrm{d}\psi}{\mathrm{d}\phi} = \frac{Rr\cos\phi - r^2}{(R - r\cos\phi)^2} = \frac{\zeta}{\xi^2} \tag{3.15}$$

such that

$$\zeta_1 = R_1 r\cos\phi_1 - r^2 \qquad \xi_1 = R_1 - r\cos\phi_1$$
$$\zeta_2 = R_2 r\cos\phi_2 - r^2 \qquad \xi_2 = R_2 - r\cos\phi_2$$

The differential equation can be re-written as

$$\frac{\mathrm{d}J_2}{\mathrm{d}\theta} = \sec^2(\psi_1 - \psi_2)\cos^2\psi_1\cos^2\psi_2\left(\frac{r\zeta_1\sec^2\psi_2}{{\xi_1}^2} - \frac{r\zeta_2\sec^2\psi_1}{{\xi_2}^2}\right) \tag{3.16}$$

$$= \frac{r\sec^2(\psi_1 - \psi_2)\cos^2\psi_1\cos^2\psi_2}{{\xi_1}^2{\xi_2}^2}\left(\zeta_1{\xi_2}^2\sec^2\psi_2 - \zeta_2{\xi_1}^2\sec^2\psi_1\right) \tag{3.17}$$

But previously from equation [3.5], it was shown that

$$\sec^2\psi = 1 + \frac{r^2\sin^2\phi}{\xi^2}$$
$$\Longrightarrow \xi^2\sec^2\psi = \xi^2 + r^2\sin^2\phi$$
$$\Longrightarrow \xi^2\sec^2\psi = R^2 + r^2 - 2Rr\cos\phi$$

The differential therefore becomes

$$\frac{\mathrm{d}J}{\mathrm{d}\theta} = \frac{r\sec^2(\psi_1 - \psi_2)\cos^2\psi_1\cos^2\psi_2}{{\xi_1}^2{\xi_2}^2}$$
$$\left(\zeta_1[{R_2}^2 + r^2 - 2R_2 r\cos\phi_2] - \zeta_2[{R_1}^2 + r^2 - 2R_1 r\cos\phi_1]\right) \tag{3.18}$$

For the maximum viewing through the aperture, $\frac{\mathrm{d}J}{\mathrm{d}\theta} = 0$. This can happen if either ψ_1 or ψ_2 becomes $\frac{\pi}{2}$. However in general this cannot happen (see the one corner case). Or that

$$\zeta_1[{R_2}^2 + r^2 - 2R_2 r\cos\phi_2] = \zeta_2[{R_1}^2 + r^2 - 2R_1 r\cos\phi_1] \tag{3.19}$$

The Case for Equidistant Corners

If $R_1 = R_2 = R$, both corners will be equidistant. The optimal solution [3.19] then becomes

$$(r - R\cos\phi_1)(R^2 + r^2 - 2Rr\cos\phi_2) = (r - R\cos\phi_2)(R^2 + r^2 - 2Rr\cos\phi_1) \tag{3.20}$$

Expanding the brackets and simplifying gives us

$$R(R^2 - r^2)(\cos\phi_1 - \cos\phi_2) = 0 \tag{3.21}$$

For $r < R$, such that the robot will not pass either of the corners, the maximum solution becomes

$$\cos\phi_1 - \cos\phi_2 = 0 \tag{3.22}$$

By substituting

$$\gamma_1 - \theta = \frac{\gamma_1 - \gamma_2}{2} + \frac{\gamma_1 + \gamma_2}{2} - \theta \tag{3.23}$$

$$\gamma_2 - \theta = -\frac{\gamma_1 - \gamma_2}{2} + \frac{\gamma_1 + \gamma_2}{2} - \theta \tag{3.24}$$

and using

$$\mu = \frac{\gamma_1 - \gamma_2}{2} \quad \text{and} \quad \omega = \frac{\gamma_1 + \gamma_2}{2} - \theta$$

the optimal solution of [3.22] can be re-written such that

$$\begin{aligned} \cos\phi_1 - \cos\phi_2 &= \cos(\mu + \omega) - \cos(-\mu + \omega) \\ &= -[\cos(\mu - \omega) - \cos(\mu + \omega)] \\ &= -2\sin(\mu)\sin(\omega) = 0 \end{aligned} \tag{3.25}$$

When equal to zero there are two solutions

$$\theta = \frac{\gamma_1 + \gamma_2}{2} \tag{3.26}$$

$$\text{or} \quad \theta = \pi + \frac{\gamma_1 + \gamma_2}{2} \tag{3.27}$$

The solution [3.26] is clearly the maximum. It corresponds to the robot moving directly towards the middle of the opening. When the corners are equidistant, this makes sense.

The Case for Non-Equidistant Corners

However when $R_1 \neq R_2$, the general case, the optimal solution of equation [3.19]

$$(R_1 r\cos\phi_1 - r^2)({R_2}^2 + r^2 - 2R_2 r\cos\phi_2) = (R_2 r\cos\phi_2 - r^2)({R_1}^2 + r^2 - 2R_1 r\cos\phi_1)$$

with the brackets expanded and simplified, becomes

$$r({R_1}^2 - {R_2}^2) - r^2(R_1\cos\phi_1 - R_2\cos\phi_2) - R_1R_2(R_1\cos\phi_2 - R_2\cos\phi_1) = 0 \qquad (3.28)$$

Taking the first set of brackets and using μ and ω from above,

$$\begin{aligned} R_1\cos(\gamma_1-\theta) - R_2\cos(\gamma_2-\theta) &= R_1\cos(\mu+\omega) - R_2\cos(-\mu+\omega) \\ &= R_1\cos(\mu+\omega) - R_2\cos(\mu-\omega) \\ &= R_1(\cos\mu\cos\omega - \sin\mu\sin\omega) \\ &\quad -R_2\cos(\cos\mu\cos\omega + \sin\mu\sin\omega) \\ &= (R_1-R_2)\cos\mu\cos\omega - (R_1+R_2)\sin\mu\sin\omega \qquad (3.29) \end{aligned}$$

Using this again for the second set of brackets,

$$R_1\cos\phi_2 - R_2\cos\phi_1 = (R_1-R_2)\cos\mu\cos\omega + (R_1+R_2)\sin\mu\sin\omega \qquad (3.30)$$

Substituting both into [3.28] and expanding and simplifying

$$r({R_1}^2 - {R_2}^2) - (R_1R_2 + r^2)(R_1-R_2)\cos\mu\cos\omega - (R_2R_2 - r^2)(R_1+R_2)\sin\mu\sin\omega = 0$$

By dividing throughout by $r({R_1}^2 - {R_2}^2)$, this equation is simplified to give

$$1 = p\cos\omega + q\sin\omega$$

Where

$$p = \frac{R_1R_2 + r^2}{r(R_1+R_2)}\cos\mu \quad \text{and} \quad q = \frac{R_1R_2 - r^2}{r(R_1-R_2)}\sin\mu \qquad (3.31)$$

This can be represented by the triangle of Figure 3.6. Dividing throughout by $\sqrt{p^2+q^2}$ gives

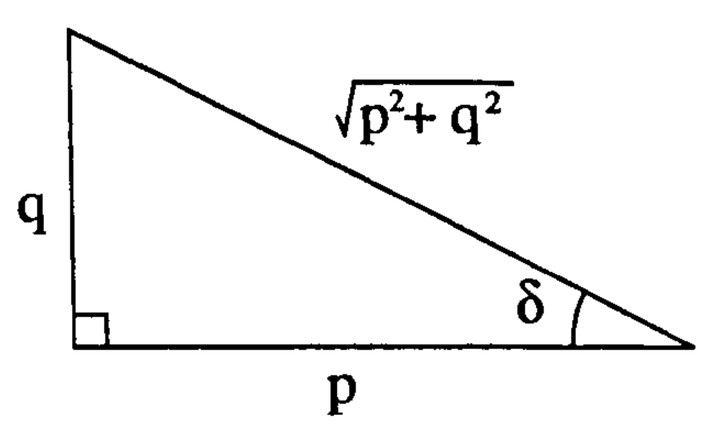


Figure 3.6: Representation of solution for utility function with one opening formed by corners equidistant from the robot position (equation [3.31]).

$$\frac{1}{\sqrt{p^2+q^2}} = \frac{p}{\sqrt{p^2+q^2}}\cos\omega + \frac{q}{\sqrt{p^2+q^2}}\sin\omega$$

From Figure 3.6 it can be seen that

$$\cos\delta = \frac{p}{\sqrt{p^2+q^2}} \quad , \quad \sin\delta = \frac{q}{\sqrt{p^2+q^2}} \quad \text{and} \quad \tan\delta = \frac{q}{p}$$

The second optimal solution can therefore be simplified to yield the utility function, as

$$\frac{1}{\sqrt{p^2+q^2}} = \cos(\omega - \delta)$$

This can be rearranged for θ such that

$$\theta = \left(\frac{\gamma_1+\gamma_2}{2}\right) - \tan^{-1}\left(\frac{q}{p}\right) - \cos^{-1}\left(\frac{1}{\sqrt{p^2+q^2}}\right) \tag{3.32}$$

In this case the robot does not head directly towards the centre of the opening, but the direction depends upon the values of R_1, R_2, γ_1, and γ_2. How these values effect the value of θ is considered in more detail in section 3.5.

3.4 The N Corner Case

The utility function described above only deals with the case of one or two corners. It is unlikely that a robot would be able to navigate through a scene without encountering a space where more than just one corner or one opening is obstructing its path. Therefore the utility function described above must be generalised to accommodate any number of openings.

With only a single corner the next-best-viewpoint is at the tangent of the motion boundary with the corner line. When faced with one opening and equidistant corners, the movement of the robot is predictably towards the centre of the opening. However with two non-equidistant corners, the direction of motion depends upon the distance to be travelled, as well as the distance to the corners of the openings.

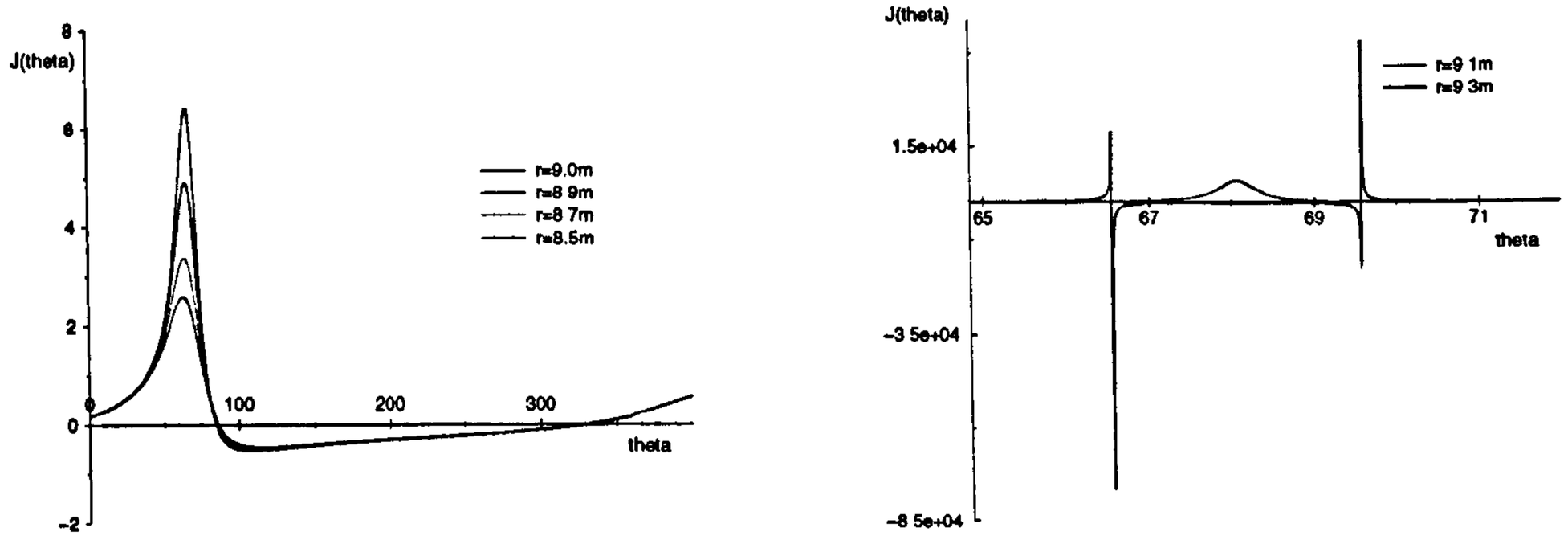
Consider a robot P with the heading PH with several walls in the vicinity forming $N/2$ openings between them. The point P' on the motion circle of radius r that maximises the viewing angle through all of the apertures is to be determined.

An N corner case can be generalised using

$$J_N = \sum_{i=1}^{N} \tan(\psi_{2i-1} - \psi_{2i}) \quad , \tag{3.33}$$

3.5 Viewpoint Planning Strategy Validation

The viewpoint planning functions summarised above are shown to be maximised by the utility functions for any number of openings and corners. Examples of the utility function for one opening of equation [3.12] are given in Figure 3.7(a).



(a) $r \ll R$ (b) $r \approx R$

Figure 3.7: Utility function of one opening for increasing values of r. $\gamma_1 = 80°$, $\gamma_2 = 50°$, R_1=10 m, and R_2=15 m.

This utility function is characterised by the curves in Figure 3.7(a), but as the value of r is increased towards the values for R_1 or R_2, the utility function breaks into two tangential spikes reflecting the utility function definition. These spikes are located towards γ_1 and γ_2 corresponding to the direction towards each corner. For such cases, the utility function for equidistant corners should be used.

3.5.1 Approach to openings

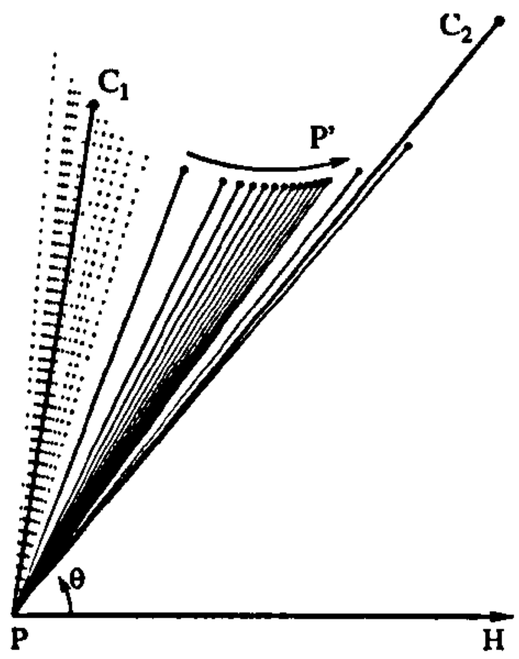
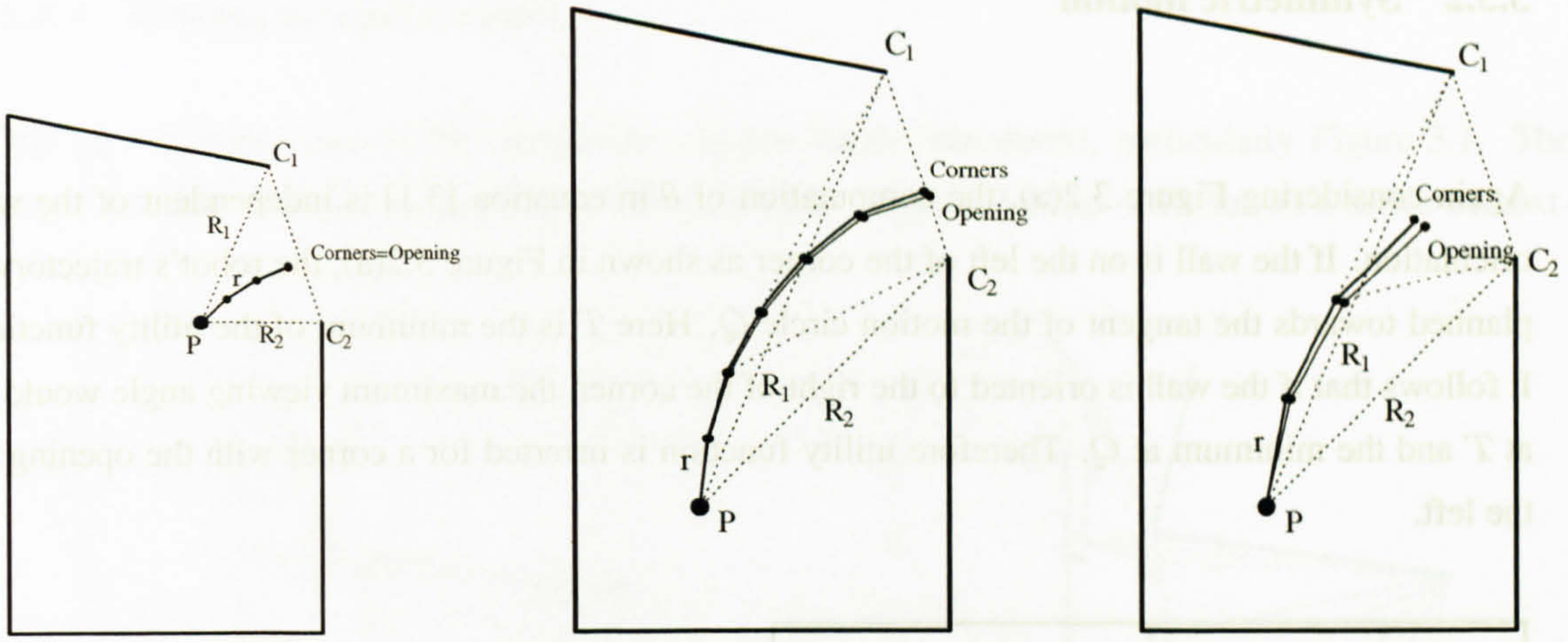


Figure 3.8: How the approach towards an opening varies with increasing r.

Further analysis of the tangential characteristic shows that the two spikes have an identical magnitude when $R_1 = R_2$. Taking the maximum of the utility function as usual, Figure 3.8 shows the motion trajectory towards an opening with increasing r. The dotted trajectories show the smaller of the two tangential spikes. As the robot is allowed to move closer to the first corner ($r \approx R_1$), the trajectory is directed towards the other, i.e. not directing the robot towards the corner it would be able to reach.

Figures 3.9(a)–3.9(c) show the robot's approach to the opening formed by the two corners C_1 and C_2 using the next-best-view calculation. It can clearly be seen that the calculated next position is



(a) r=0.75m (b) r=1.25m (c) r=2m

Figure 3.9: Moving towards opening with r fixed at 0.75m, 1.25m, and 2m respectively. Two utility calculations are considered: for two separate corners, and as one opening.

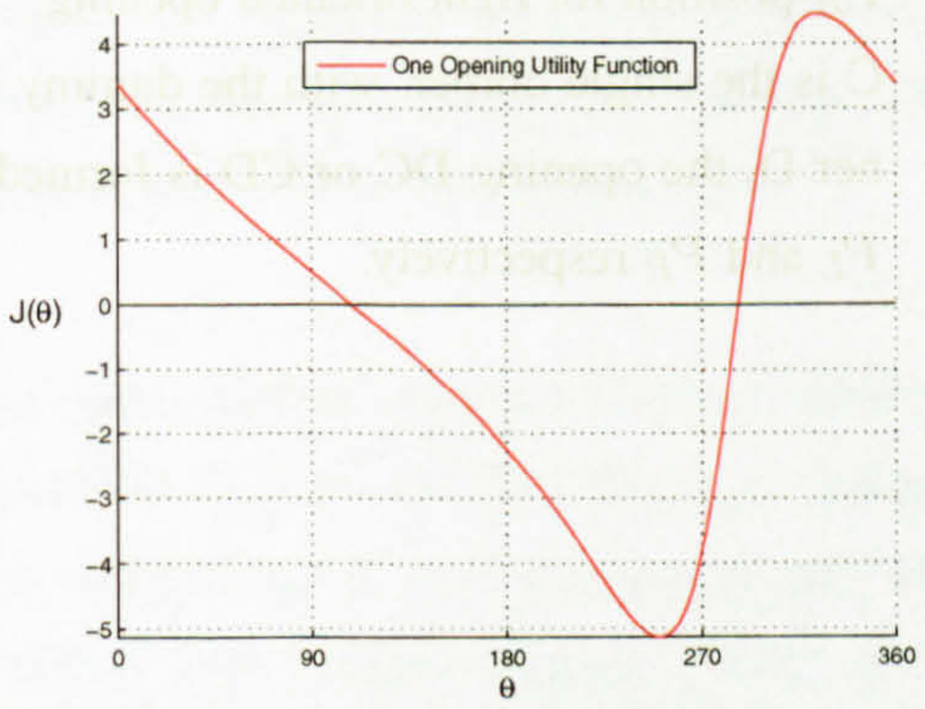


Figure 3.10: Utility curve for one opening, with corner positions identical to Figure 3.11.

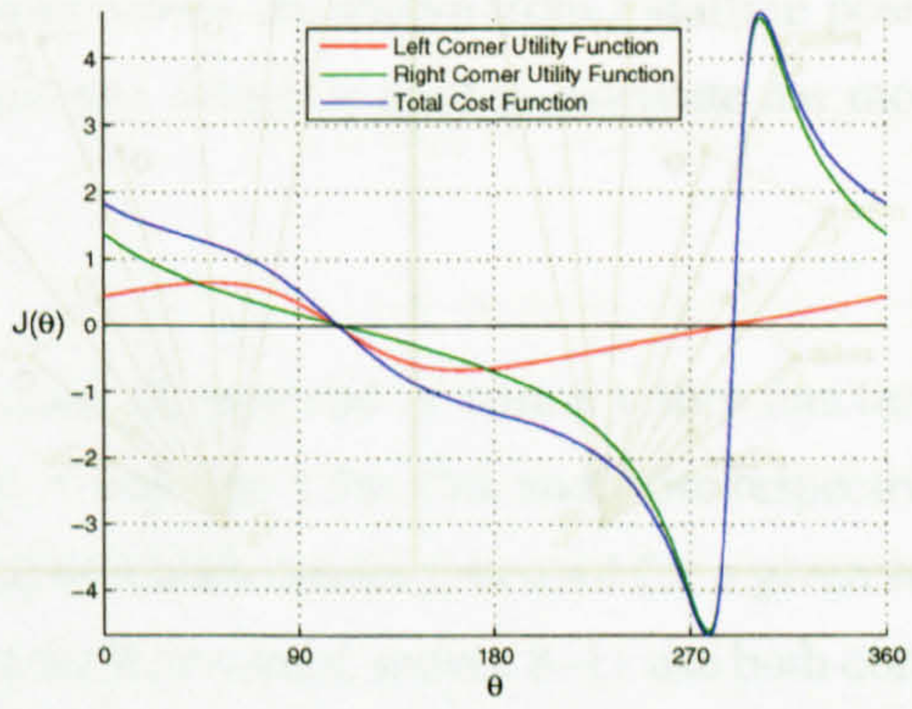


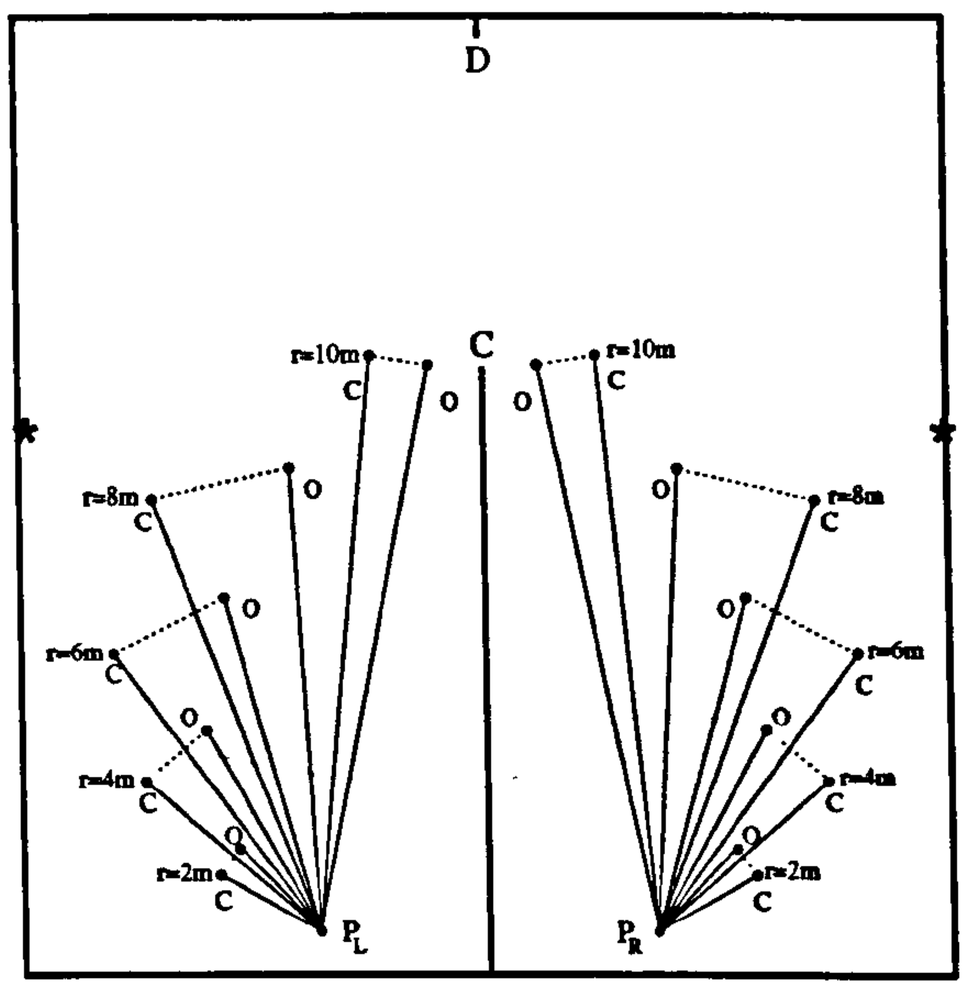
Figure 3.11: Utility curve for corners, with corner positions identical to Figure 3.10.

not towards the centre of the opening until r is similar to R_1 & R_2. As the distance to move (r) is increased, from Figure 3.9(a)–3.9(c), the trajectory towards the opening is very similar.

The trajectories for C_1 and C_2 being used in the next-best-view calculation as an opening, in comparison to the next-best-view around corner C_1 and the next-best-view around corner C_2 is also similar. Figures 3.10 and 3.11 show the utility curves for these two cases respectively. Not only are the trajectories almost identical but the utility curve for one opening and the summed curve for the two corners are very similar also.

3.5.2 Symmetric motion

Again considering Figure 3.2(a), the computation of θ in equation [3.1] is independent of the wall orientation. If the wall is on the left of the corner as shown in Figure 3.2(a), the robot's trajectory is planned towards the tangent of the motion circle Q. Here T is the minimum of the utility function. It follows that if the wall is oriented to the right of the corner, the maximum viewing angle would be at T and the minimum at Q. Therefore utility function is inverted for a corner with the opening on the left.



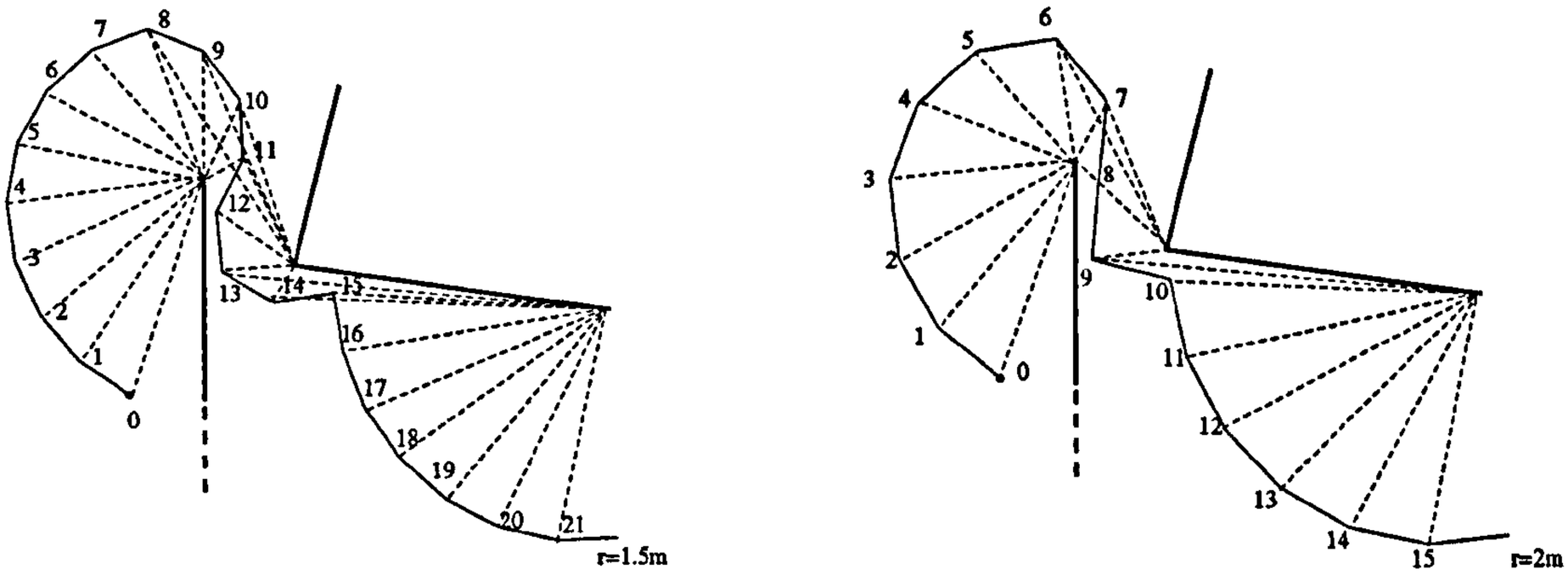
P_L-robot position for left oriented opening
P_R-position for right oriented opening
C is the single corner, with the dummy corner D, the opening DC or CD is formed for P_L and P_R respectively.

Figure 3.12: Testing the symmetry of the next-best-view calculation with differing wall orientation. r is varied to display the characteristic of the utility function used for two separate corners (C) and for one opening (O).

Figure 3.5.2 shows tests carried out for complementary corner orientations (C) as r is increased. Also shown is the trajectory when another corner D is introduced so that the corner becomes an opening, O. When considered as an opening the robot is directed more towards the opening itself, whereas with only the corner C, a wider berth is given. If the enclosing wall (marked with asterisks) was nearer to the robot, the single corner trajectory could lead the robot into the wall. However the corner trajectory does provide a greater viewing angle around the considered corner than the opening trajectory (since the second corner constrains the opening's utility function). Therefore if the next-best-view algorithm was being applied to an environment with narrow corridors and only a single corner was available for the next-best-view calculation, an extra corner could be added to constrain the motion of the robot but still acquire an optimal view.

3.5.3 Moving around corners

The idea of using corners for navigation was previously introduced, particularly Figure 3.1. The utility function can also be applied to this technique. Figures 3.13(a)–3.14 shows a series of next-



(a) r=1.5m (b) r=2m

Figure 3.13: Using a fixed distance to travel, r, sequential moves are shown from a starting position around corners and through an opening. The corner/opening which is used to calculate the motion trajectory are shown by dashed lines.

best-view choices using the corresponding one corner, one opening and N corner utility function of equations [3.7], [3.32], [3.33] for a set distance to move, r, equal to 1.5m, 2m, and 2.5m respectively. The dotted lines in each diagram identify which corner, or which opening, is used for a given robot position. For example in Figure 3.13(a), moves 0–7 use the first corner, moves 8–11 use both corners as an opening, move 11 uses the second corner, moves 13 and 14 use the second and third corners with views around both corners etc. A single corner is used for the calculation when only one corner is within visibility of the robot's position and two corners are selected for use as an opening in a similar way. Once the robot's position is past the opening, i.e. the path of the robot would intersect the threshold of the opening, the opening is no longer considered for use. This corner selection method is also applied in the exploration techniques discussed in the following chapter.

The trajectories computed here follow a similar pattern as that of [Murray et al., 1996] but for completely different reasons. Murray used corners as fixing points and the robot trajectory was planned to keep a set distance from the corner, i.e. it moves on a circle of set radius. In the presented method, the corners are used to calculate the best-view of an occluded region.

Comparisons for increasing r for a given single corner scenario are shown in Figure 3.15. As the given distance to move, r increases, the optimum next-best-view position is directed more towards the corner. Throughout this chapter, the value of r is assumed constant. As shown here, varying the value of r displays different motion characteristics, which could be varied with changes to the local environment. The value for r used during the exploration in subsequent chapters is limited to three

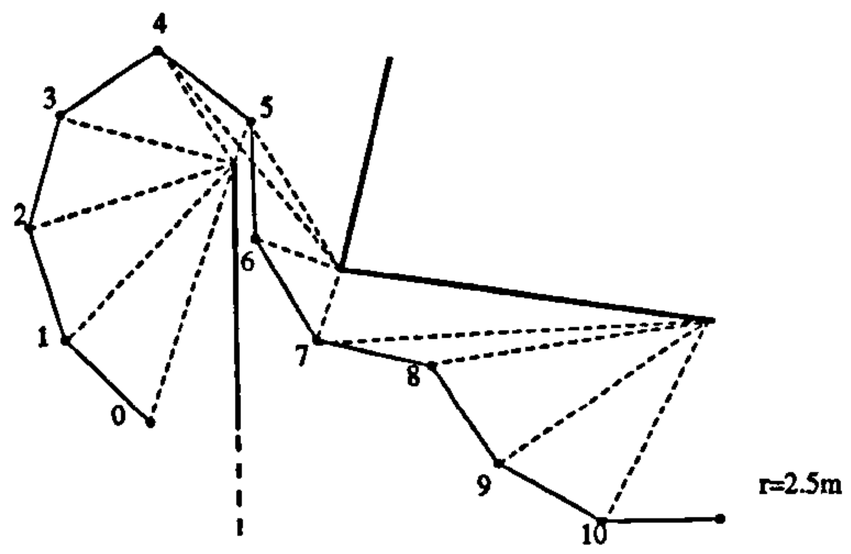


Figure 3.14: Using fixed distance r=2.5m, sequential moves are shown for the robot's motion around corners and through an opening.

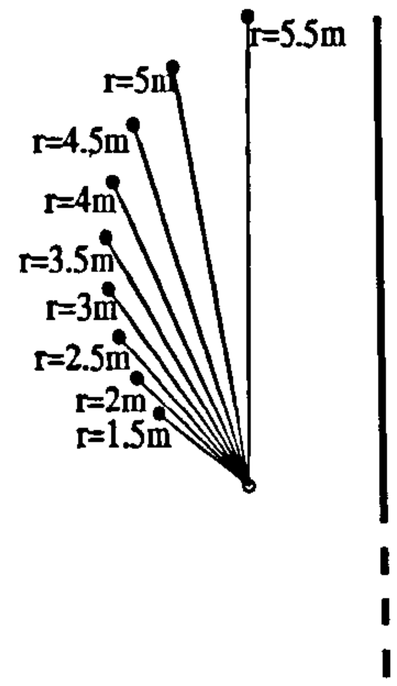


Figure 3.15: For a given corner, the value of r is varied, showing the effect of the choice of r on the robot's trajectory.

choices. This coarse selection method is adequate for the environments shown, but it may be possible to choose the value of r dynamically (depending on the local environment).

3.5.4 Example exploration

The proposed strategy has been shown to operate under several situations. It can also be used to explore an unknown environment, using only information of the positions of obstacle corners that occlude regions, and the obstacle's wall orientation for any single corners. A pre-set allowable distance to move, r, is constant throughout the exploration.

Figure 3.16 shows a sample scene where the robot starts in the bottom left corner at position 0. The bold lines are the positions of obstacles, the blue shaded regions show inaccessible areas. The corners used for the exploration are given to the exploration algorithm by the human operator. As before, the corners chosen are those which are visible to the robot and where the threshold of the opening has not been past. The viewing angle ν, is shown at each move for the chosen opening. This example shows how 99.8% of the available area has been viewed in 30 moves (the robots field-of-view constraint is ignored in this example). The small yellow area shows the area of the scene that was not visible to the robot during its exploration. Moves 0 to 10 inclusive each only using a single opening and it can be seen that the robot moves safely through the first few openings. At moves 11 & 12, two and three openings are used respectively.

At move 12, the next-best-view algorithm is provided with three openings (a, b and c) where c occludes the largest area. The openings a and b do not occlude regions but have not yet been crossed and therefore are still included as candidate opening options. The utility function for the openings a, b and c are shown in figures 3.17 to 3.18 respectively. It is obvious that when these are summed, the utility function of Figure 3.18 will clearly dominate the other two. Thus exploration proceeds through the narrow corridor straight ahead.

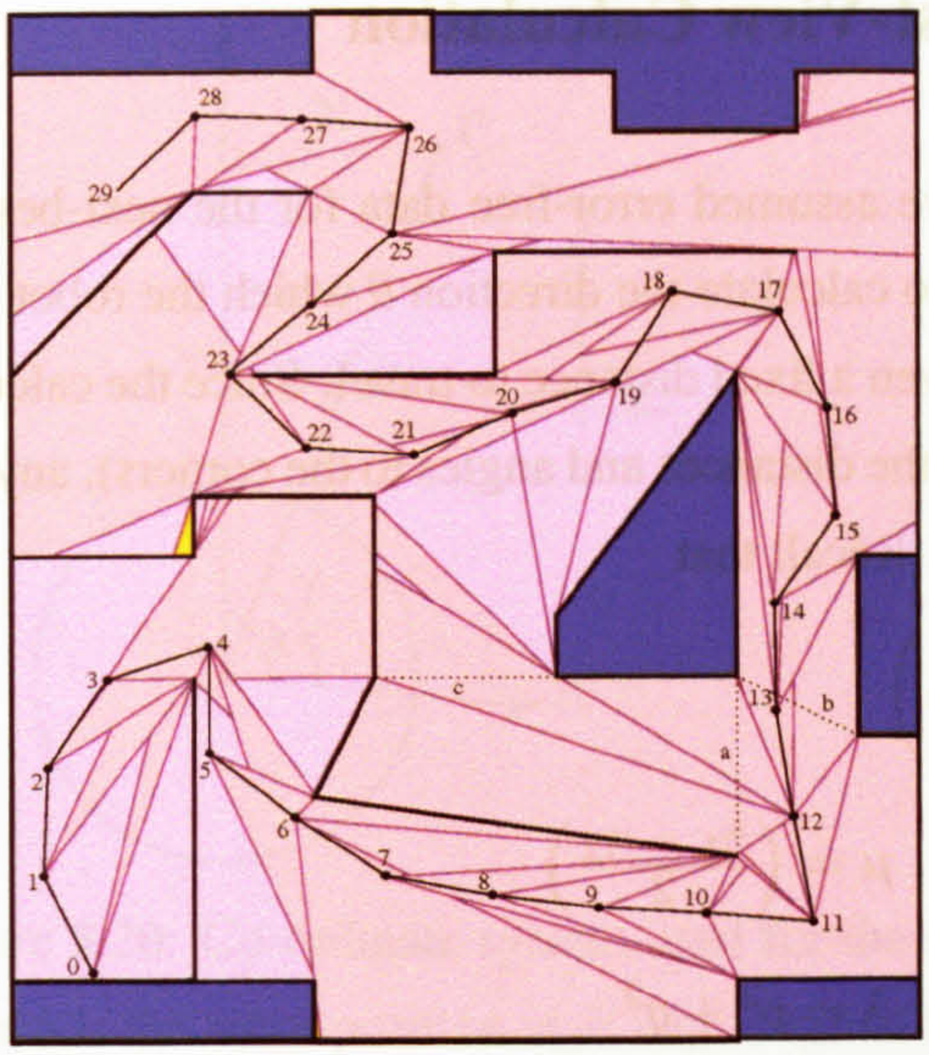


Figure 3.16: Example exploration of unknown environment with fixed distance to move, r=2m. Robot starts at position 0; the opening/corner to view through/around is selected by the operator.

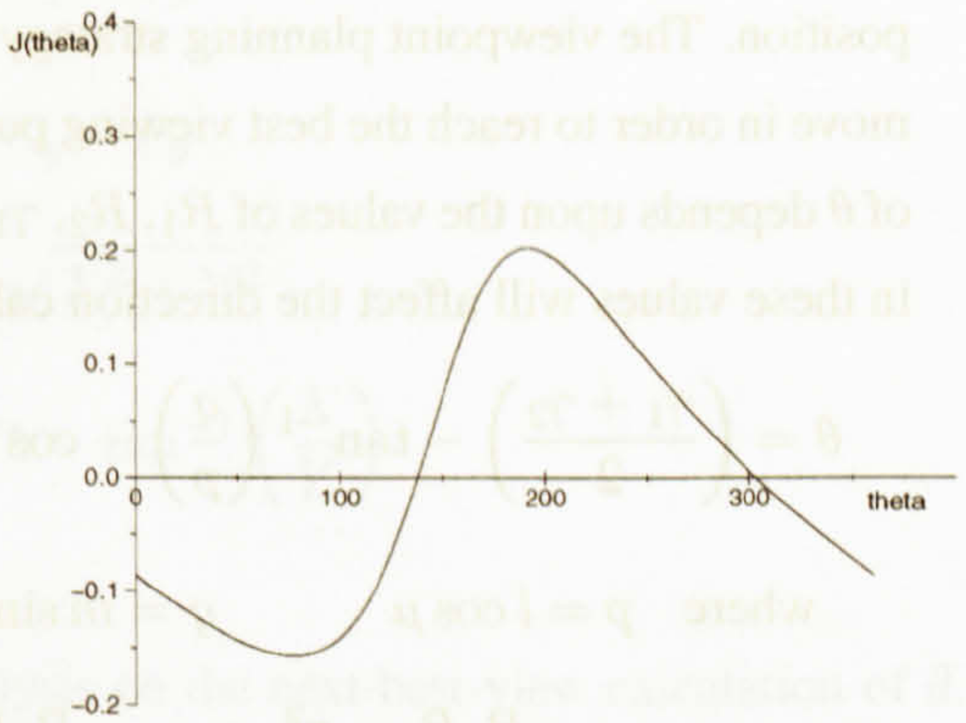


Figure 3.17: Utility curve at move 12 for highlighted opening a in Figure 3.16.

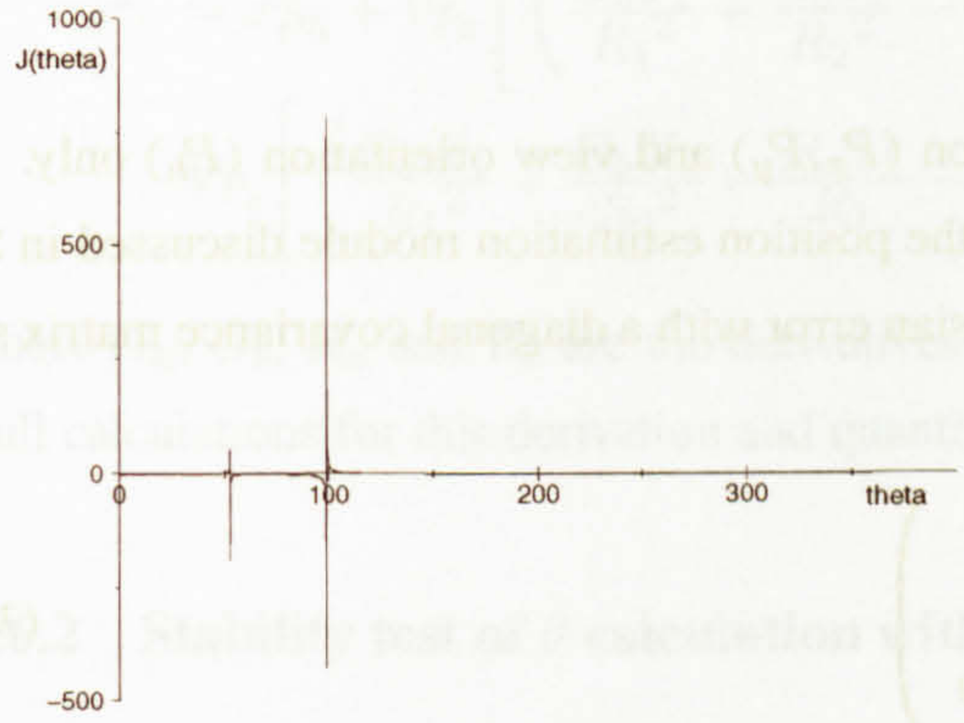


Figure 3.18: Utility curve at move 12 for highlighted opening b in Figure 3.16.

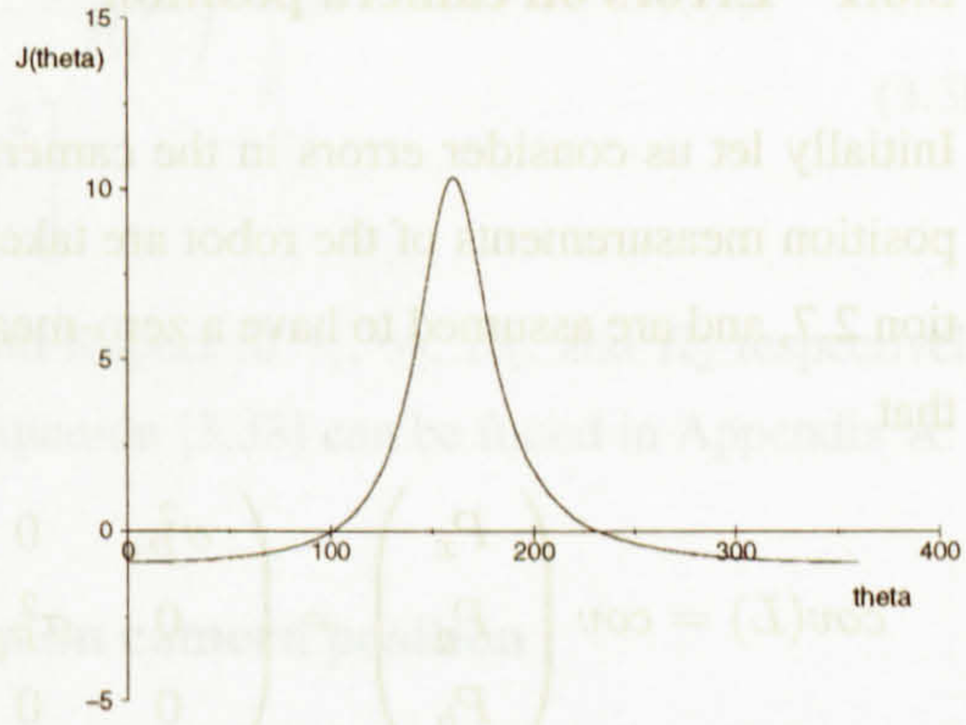


Figure 3.19: Utility curve at move 12 for highlighted opening c in Figure 3.16.

At move 19, the choice of openings, again, require a distinct direction decision to be made. By again summing the utility functions for the two visible openings, the wider opening at c would yield a greater total viewing angle. However this would mean that the robot would be viewing areas already seen. In this case, the information for opening c was ignored by the operator in order to show the full exploration of the scene, rather than the circumnavigation of an obstacle. The following chapter discusses the automatic choice of openings to be used by the next-best-view algorithm. Again at move 25, where the area to the right of the map has already been seen from moves 12 through 16, the opening for the previously seen area is ignored.

3.6 Error Analysis on the Next-Best-View Calculation

All the examples shown so far in this chapter have assumed error-free data for the next-best-view position. The viewpoint planning strategy is used to calculate the direction θ which the robot should move in order to reach the best viewing position given a fixed distance to travel. Since the calculation of θ depends upon the values of R_1, R_2, γ_1 and γ_2 (the distances and angles to the corners), any errors in these values will affect the direction calculation. Recall that

$$\theta = \left(\frac{\gamma_1 + \gamma_2}{2}\right) - \tan^{-1}\left(\frac{q}{p}\right) - \cos^{-1}\left(\frac{1}{\sqrt{\lambda}}\right)$$

$$\text{where} \quad p = l\cos\mu \qquad q = m\sin\mu \qquad \mu = \left(\frac{\gamma_1 - \gamma_2}{2}\right) \tag{3.34}$$

$$l = \frac{R_1R_2 - r^2}{r(R_1 - R_2)} \quad m = \frac{R_1R_2 + r^2}{r(R_1 + R_2)} \quad \lambda = p^2 + q^2$$

The values of R_1, R_2, γ_1 and γ_2 will be affected by small changes to the positions of the robot and of the corners. The problem with errors only on the camera position will be considered first, and then with errors on the positions of the corners.

3.6.1 Errors on camera position

Initially let us consider errors in the camera position (P_x,P_y) and view orientation (P_h) only. The position measurements of the robot are taken from the position estimation module discussed in Section 2.7, and are assumed to have a zero-mean Gaussian error with a diagonal covariance matrix such that

$$cov(L) = cov\begin{pmatrix} P_x \\ P_y \\ P_h \end{pmatrix} = \begin{pmatrix} \sigma_{Px}^2 & 0 & 0 \\ 0 & \sigma_{Py}^2 & 0 \\ 0 & 0 & {\sigma_{Ph}}^2 \end{pmatrix} \tag{3.35}$$

where σ_{Px}, σ_{Py}, and σ_{Ph} are the standard deviations of the camera's x and y position and orientation in a world co-ordinate frame.

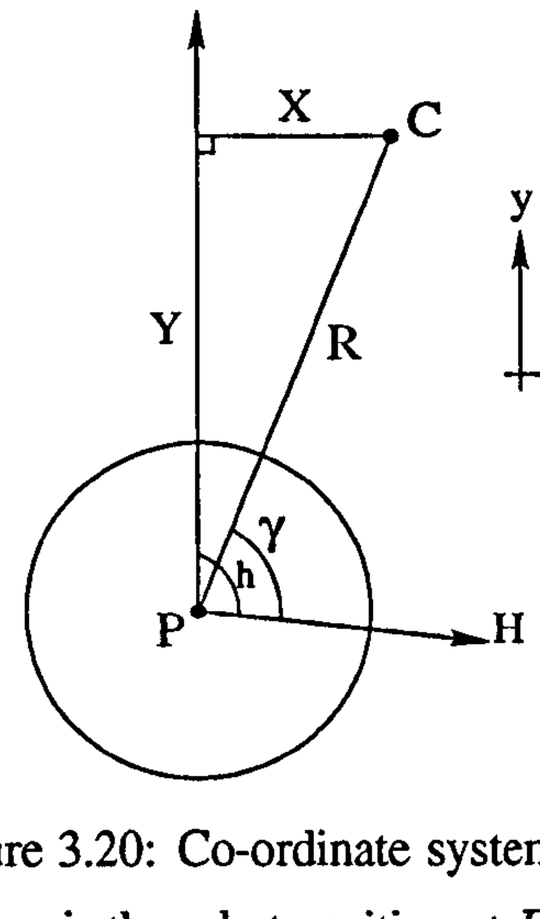
The variance of θ can be estimated from a series of Jacobian matrix calculations, where S are the parameters of the equation for θ and L is the position of the camera.

$$\sigma_\theta{}^2 = \frac{\partial\theta}{\partial S} cov(S) \frac{\partial\theta}{\partial S}^T \quad \text{where} \quad S = [\ \gamma_1 \quad \gamma_2 \quad R_1 \quad R_2\ ]^T \tag{3.36}$$

$$cov(S) = \frac{\partial S}{\partial L} cov(L) \frac{\partial S}{\partial L}^T \quad \text{where} \quad L = [\ P_x \quad P_y \quad P_h\ ]^T \tag{3.37}$$

The co-ordinate system of Figure 3.20 is defined to allow the values of R_1, R_2, γ_1, and γ_2 to be expressed in terms of the camera pose and the corner positions.

Here the errors are assumed linear and local to the point being considered i.e. the robot or corner position is within two to three standard deviations from the estimated point position (and orientation



$$X = C_x - P_x$$

$$Y = C_y - P_y$$

$$R = \sqrt{X^2 + Y^2}$$

$$\gamma = h - \tan^{-1}\left(\frac{X}{Y}\right)$$

Figure 3.20: Co-ordinate system used for the error analysis on the next-best-view calculation of θ. Shown is the robot position at P with original heading H, and one corner C in the vicinity.

in the case of the camera). The variance of θ can be estimated given errors on the position and orientation of the robot, $\sigma^2_{P_x}$, $\sigma^2_{P_y}$, $\sigma^2_{P_h}$ such that

$$\begin{aligned}\sigma_\theta{}^2 &= \sigma^2_{Ph} + V_{P_x}\left[\left(\frac{G_a Y_1}{R_1{}^2} + \frac{G_b Y_2}{R_2{}^2} - \frac{R_a X_1}{R_1} - \frac{R_b X_2}{R_2}\right)^2\right] + \\ & V_{P_y}\left[\left(\frac{G_a X_1}{R_1{}^2} + \frac{G_b X_2}{R_2{}^2} - \frac{R_a Y_1}{R_1} - \frac{R_b Y_2}{R_2}\right)^2\right]\end{aligned} \tag{3.38}$$

where G_a, G_b, R_a and R_b are the derivatives of θ with respect to γ_1, γ_2, R_1, and R_2 respectively. Full calculations for this derivation and quantities in equation [3.38] can be found in Appendix A.

3.6.2 Stability test of θ calculation with errors on camera position

For the environment situation with two corners forming an opening shown in Figure 3.21, the camera position was varied over this test area. The value of θ for positions across the test area are shown in Figure 3.22(a), where θ is measured from the horizontal axis, and it can be seen that positions to the far right of the test area yield a higher rotation value to direct the camera back towards the opening. The corresponding estimated variance of these θ calculations, using [3.38], is plotted in Figure 3.22(b) with input error values of σ_{P_x} =0.03m and σ_{P_y} =0.03m. σ_{P_h} is set to zero since this just adds to the total error (see equation [3.38]). It can be seen that the error of θ is stable over the majority of the test area for any position of P. However, as the position of the robot approaches the corners, the calculation of σ_θ becomes large. The areas within the regions of $R_1 \leq r$ and $R_2 \leq r$ are not considered since they are not defined for the calculation of θ. Figure 3.22(c) shows a portion of the area away from the corners. It is clear that the error gradually increases as the starting position of the camera is closer to the corner.

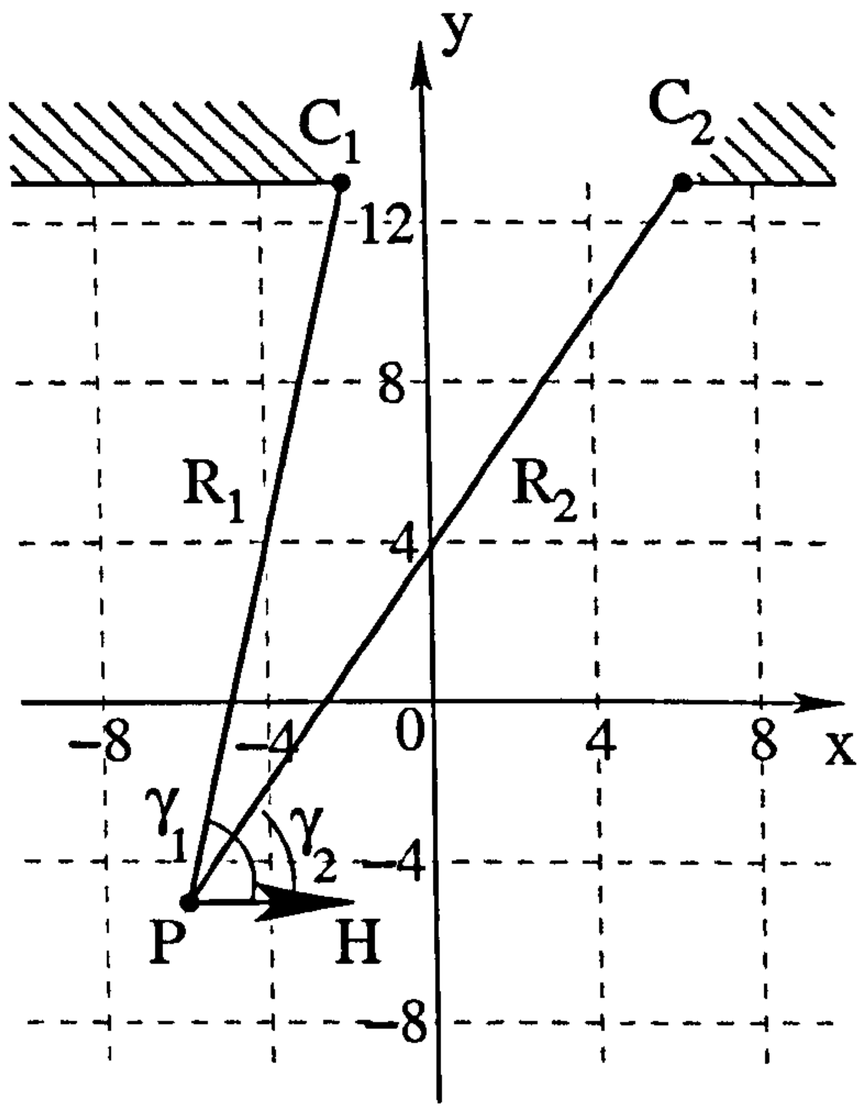
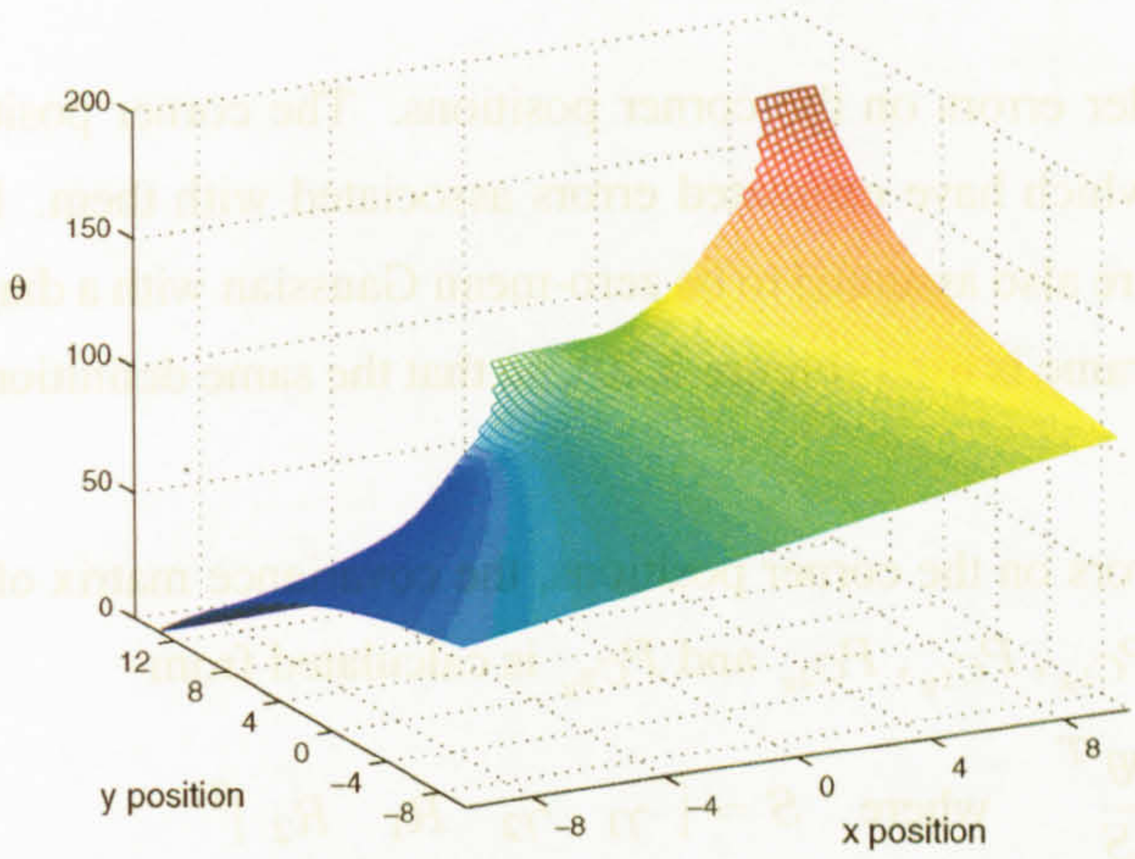
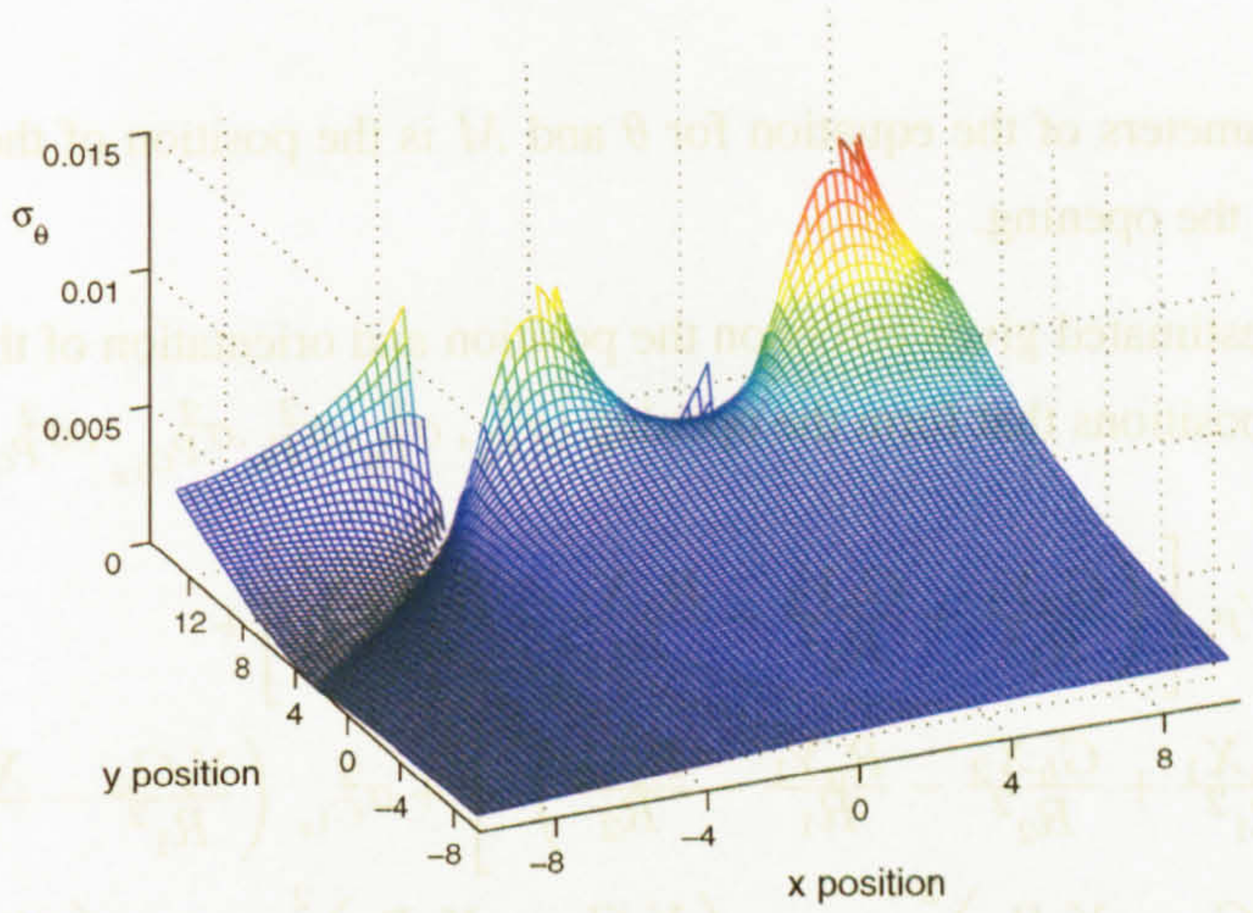


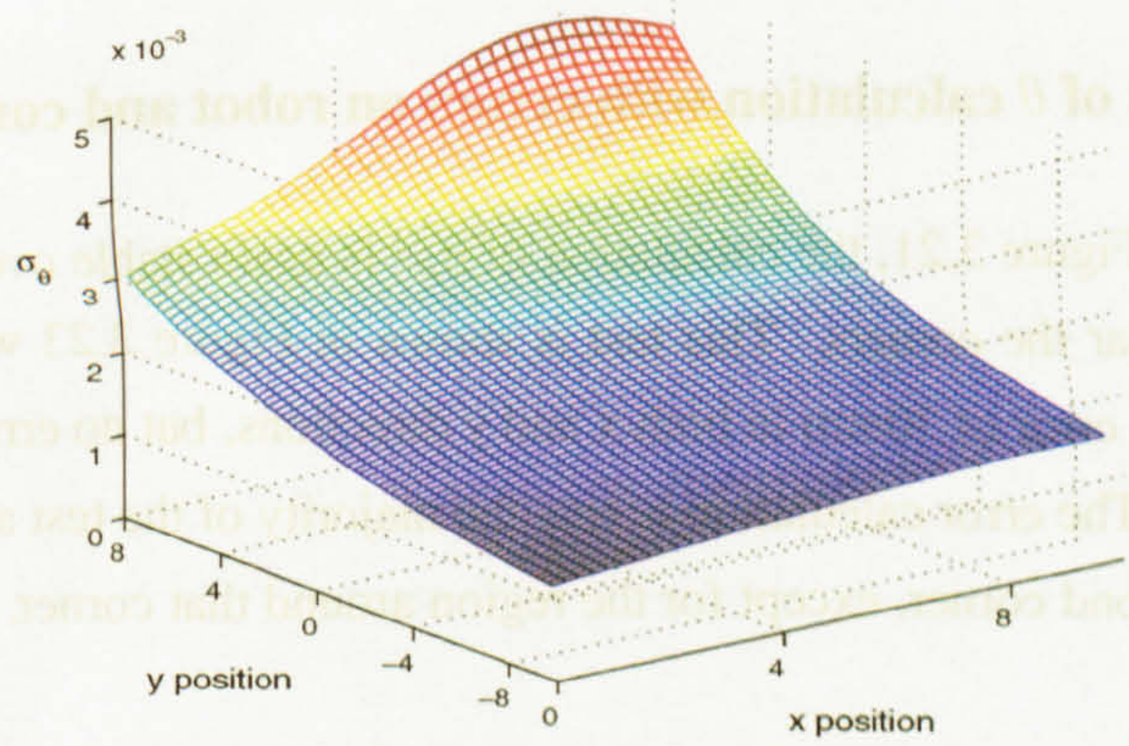
Figure 3.21: Test area for error analysis. Corner positions C_1 and C_2 are fixed at the positions shown; the robot position P is varied across the test area.



(a) Calculation of the next-best-view orientation θ.



(b) σ_θ for errors on robot position.



(c) σ_θ for errors on robot position - focus area

Figure 3.22: Values of θ and σ_θ calculated for robot positions across the test area of Figure 3.21 of one opening formed by two corners. σ_θ is calculated for position errors on the robot only.

3.6.3 Errors on corner positions

Let us now also consider errors on the corner positions. The corner positions are taken from the map building module which have estimated errors associated with them. For the purpose of these calculations the errors are also assumed to be zero-mean Gaussian with a diagonal covariance matrix. The same co-ordinate frame is used (Figure 3.20), so that the same definitions for R_1, R_2, γ_1, and γ_2 exist.

With the addition of errors on the corner positions, the covariance matrix of γ_1, γ_2, R_1 and R_2 with respect to P_x, P_y, P_h, $P_{C_{1x}}$, $P_{C_{1y}}$, $P_{C_{2x}}$ and $P_{C_{2y}}$ is calculated from

$$\sigma_\theta{}^2 = \frac{\partial\theta}{\partial S} cov(S) \frac{\partial\theta}{\partial S}^T \quad \text{where} \quad S = [\ \gamma_1 \quad \gamma_2 \quad R_1 \quad R_2\ ]^T \tag{3.39}$$

$$cov(S) = \frac{\partial S}{\partial M} cov(M) \frac{\partial S}{\partial M}^T \quad \text{where} \quad M = [\ P_x \quad P_y \quad P_h \quad P_{C_{1x}} \quad P_{C_{1y}} \quad P_{C_{2x}} \quad P_{C_{2y}}\ ]^T \tag{3.40}$$

where S are the parameters of the equation for θ and M is the position of the camera and the two corners which define the opening.

The variance of θ is estimated given errors on the position and orientation of the robot in addition to errors on the corner positions that form the opening ($\sigma^2_{P_x}$, $\sigma^2_{P_y}$, $\sigma^2_{P_h}$,$\sigma^2_{P_{C_{1x}}}$, $\sigma^2_{P_{C_{1y}}}$, $\sigma^2_{P_{C_{2x}}}$ and $\sigma^2_{P_{C_{2y}}}$.

$$\begin{aligned}\sigma_\theta{}^2 \ = & \sigma^2_{P_h} + V_{P_x}\left[\left(\frac{G_a Y_1}{R_1{}^2} + \frac{G_b Y_2}{R_2{}^2} - \frac{R_a X_1}{R_1} - \frac{R_b X_2}{R_2}\right)^2\right] + \\ & V_{P_y}\left[\left(\frac{G_a X_1}{R_1{}^2} + \frac{G_b X_2}{R_2{}^2} - \frac{R_a Y_1}{R_1} - \frac{R_b Y_2}{R_2}\right)^2\right] + \sigma^2_{C_{1x}}\left(\frac{Y_1 G_a}{R_1{}^2} - \frac{X_1 R_a}{R_1}\right)^2 + \\ & \sigma^2_{C_{1y}}\left(\frac{X_1 G_a}{R_1{}^2} + \frac{Y_1 R_a}{R_1}\right)^2 + \sigma^2_{C_{2x}}\left(\frac{Y_2 G_b}{R_2{}^2} - \frac{X_2 R_b}{R_2}\right)^2 + \sigma^2_{C_{2y}}\left(\frac{X_2 G_b}{R_2{}^2} + \frac{Y_2 R_b}{R_2}\right)^2\end{aligned} \tag{3.41}$$

3.6.4 Stability test of θ calculation with errors on robot and corner positions

For the same setup as Figure 3.21, the calculation of $\sigma_\theta{}^2$ is again stable over the majority of the test area but is unstable near the corners. This test is shown in Figure 3.23 where C_2 (the right-hand corner) is set to have an error of 30mm in both x and y directions, but no error on C_1. Instability also occurs when $R_1 \approx R_2$.The error calculation across the majority of the test area is not affected by the added errors on the second corner, except for the region around that corner.

3.7 Conclusions

This chapter has focused on a viewpoint planning strategy to calculate the position that yields a maximum view through any number of openings or corners for a specified fixed distance to travel.

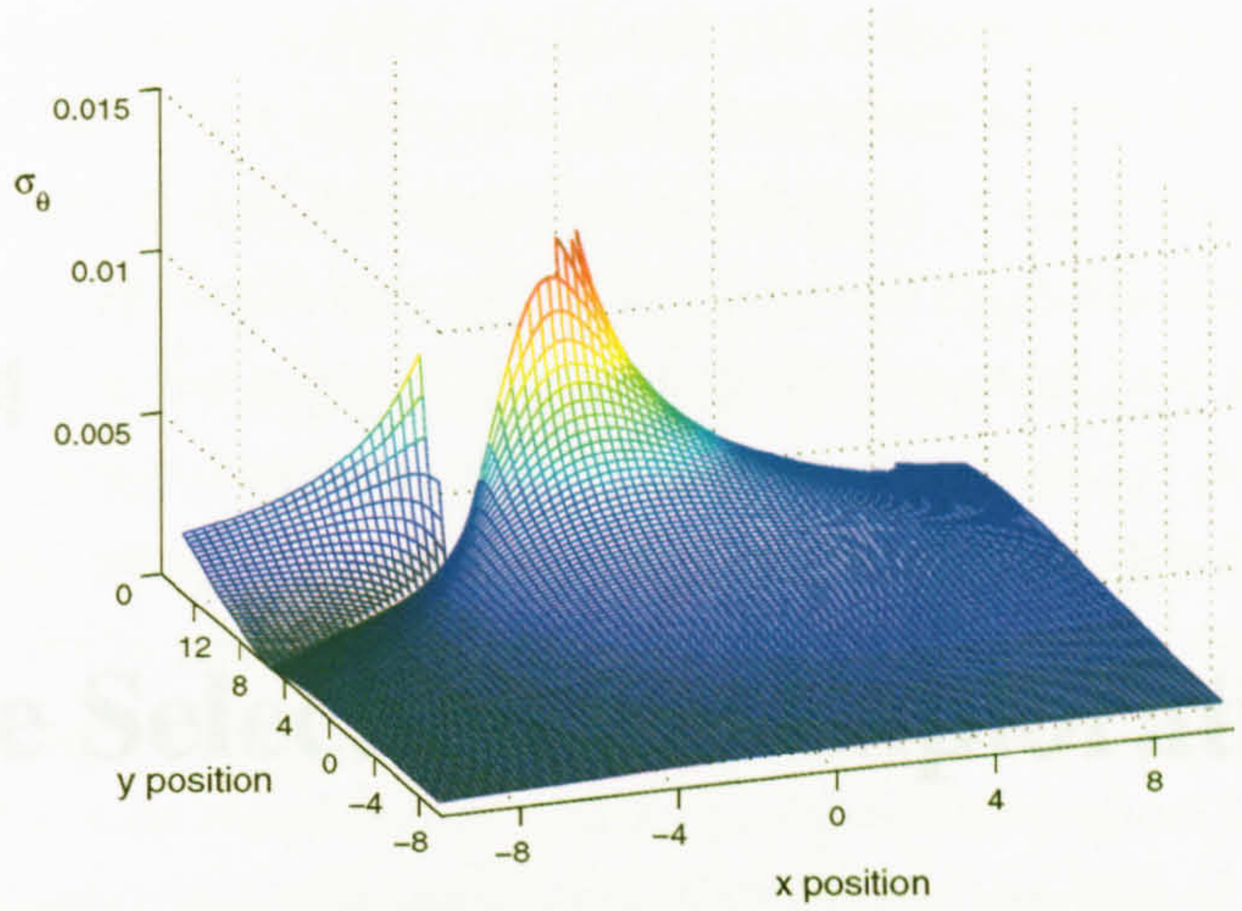
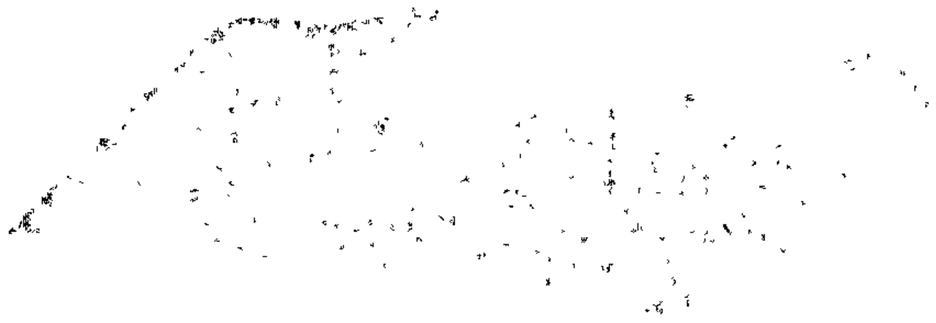


Figure 3.23: Values of θ and σ_θ calculated for robot positions across the test area of Figure 3.21 of one opening formed by two corners at (-2,13) and (6,13). σ_θ is calculated for position errors on the robot and one corner.

The algorithm for an ideal world with no errors was introduced and it was shown how the algorithm could be used in many situations from the exploration shown in section 3.5. This showed that for a given distance to move with corner information given, the algorithm could be used to explore an entire unknown area. This algorithm requires the visible corner positions to be read into the algorithm along with the correct opening orientations of the corners. Errors were introduced into the input data to show how the next-best-view calculation for one opening would be affected. A linear approximation of the errors was taken on the calculation of θ, and the algorithm was shown to be robust for the general condition of the initial camera position not being too close to the corners.

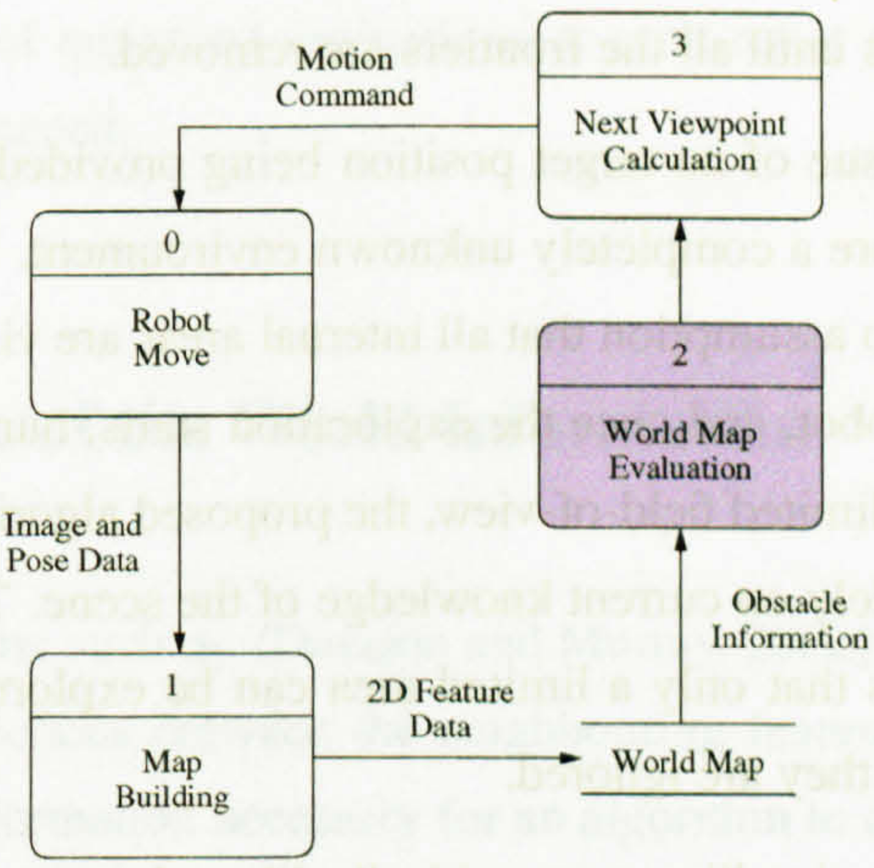
A piecewise linear motion is adopted by the strategy to reduce motion errors incurred by curvilinear or cyclical motion. This motion allows the map to be incrementally built during the motion. The world is assessed at the end of each linear piece move, thus reducing the computational requirements. The assumption made during each motion section is that unless a new feature is detected in the scene that creates an obstacle across the motion trajectory, then this information of a new obstacle being present in the world is only used at the end of the motion to calculate the following next-best-view. In summary, the robot moves in a straight line to a position where the robot will gain the largest viewing angle around a corner, or through an opening of two corners and so on.

With a technique to find the useful corners in a current field-of-view (where useful corners are those which occlude regions that have not previously been seen), the strategy developed in this chapter will enable an exploration through a two-dimensional environment *a priori* unknown to the viewpoint planning algorithm. The next chapter considers a set of methods to enable automatic selection of the obstacle corners to use as input to the next-best-view algorithm presented in this chapter.



Chapter 4

Obstacle Selection for Exploration



In the previous chapter, a next-best-view algorithm was discussed. Given the position of an opening relative to the position of the camera, the algorithm calculates the position that provides the maximum aperture view through the opening. This chapter addresses the question of how to choose an opening. Candidate openings are selected from the current obstacle list. Given a set of obstacles and their corresponding next-best-viewpoints, the improvement that would be made to the world map from data gathered at each hypothesised position is used to select which candidate to view. Using successive obstacle selections, a set of algorithms to explore an unknown scene is presented.

4.1 Introduction

To solve the problem of autonomous exploration, many researchers have concentrated on exhaustive exploration techniques such as a raster scan where every part of the environment is physically visited by the robot, e.g. the Lawnmower algorithm (discussed by [Arkin et al., 1993]). Other methods include: wall-hugging, involving the use of a side wall to guide the robot past straight obstacles ([Albers et al., 1999]), circumnavigation algorithms which utilise a wall-hugging algorithm to

move around obstacles that intersect the path of a raster scan algorithm ([Kutulakos et al., 1993], [Lorigo et al., 1997]). Human operators can also decide where a robot should visit, either using a series of way-markers (a series of positions that the robot should visit) as in [Newman et al., 2003], or by defining target positions on- or off-line to aid navigation such as [Burgard et al., 2000]. An even less intelligent algorithm is for the robot to follow the path defined by a human walking in front of the robot ([Althaus and Christensen, 2003]) and the built map is then used for navigating around the known environment. An exploratory algorithm which is not based on an exhaustive technique has been developed by [Grabowski et al., 2003]. In this method an Occupancy grid of the seen areas is stored and the boundary between the known and unknown areas forms a frontier for the exploration algorithm to key into. This frontier-based exploration offers a solution to the problem of how to expand the knowledge of the environment. However, they do not discuss how to choose a frontier when more than one option is available. [Youngblood et al., 2000] also uses a frontier-based approach to the autonomous exploration problem. Their method for frontier selection is to move the robot in increasing concentric circles until all the frontiers are removed.

This thesis addresses the issue of no target position being provided and no prior knowledge of the scene. The robot is to explore a completely unknown environment. The shape and size of the scene is not known, and there is no assumption that all internal areas are visible or accessible. There are no way-markers to guide the robot, and once the exploration starts, human intervention is not possible. Using a vision system with limited field-of-view, the proposed algorithm must select which direction to move and view, based solely on current knowledge of the scene. The only restriction placed upon the exploration algorithm is that only a limited area can be explored. If the vision system detects features outside of this area they are ignored.

The robot is to explore in an intelligent way, ideally like a human might. For example, you arrive at a hotel in a city that you've never visited before. You have no map or guide to help you and therefore no knowledge of where the interesting areas are. You decide to explore the immediate area of the hotel by setting a target of exploring the area within one square mile. On leaving the hotel you take a quick scan of the hotel's surrounding buildings and then select an area to head towards. This direction may lead to a new area to explore further, or it may lead to a dead-end.

From an initial scan of the environment at a starting position, a representation of the scene is incrementally built using the algorithms described in Chapter 2. To gain more knowledge of the scene, a calculation of the improvement of the view of detected obstacles allows the next move to be selected. The vision system cannot differentiate between obstacles in the scene and openings which can be traversed. A confidence measure of whether world obstacles represent scene obstacles is maintained using the visibility constraint described by [Manessis, 2001]. When the view of a barrier reveals new areas (within the restricted area), these are also explored. In this chapter a map building algorithm that can successfully maintain information of world feature points such as those discussed in Chapter 2 is assumed. Errors in this part of the system will be important for the effectiveness of the exploration and will need to be addressed for the exploration to be useful in a real system. In this chapter however, attention is drawn to the exploration algorithm itself.

In this chapter a strategy for exploration is presented, where each move is evaluated separately. Each robot position provides new information about the scene: either the discovery of new scene features, or an update of information of the already known scene features. This chapter develops an exploration strategy that maximises the amount of new information obtained from each new viewpoint. Initially considered is how the world map features are connected to represent obstacles in the scene. Using this obstacle representation, the quantisation of information about where an obstacle has been viewed from is reviewed. Considered next is a method to store the information of those areas of the scene that have been viewed, so that even when no features are detected in the scene, information of the viewed area is stored. Finally, as a third element, a method to push the robot away from previously visited positions (a similarity measure) is considered.

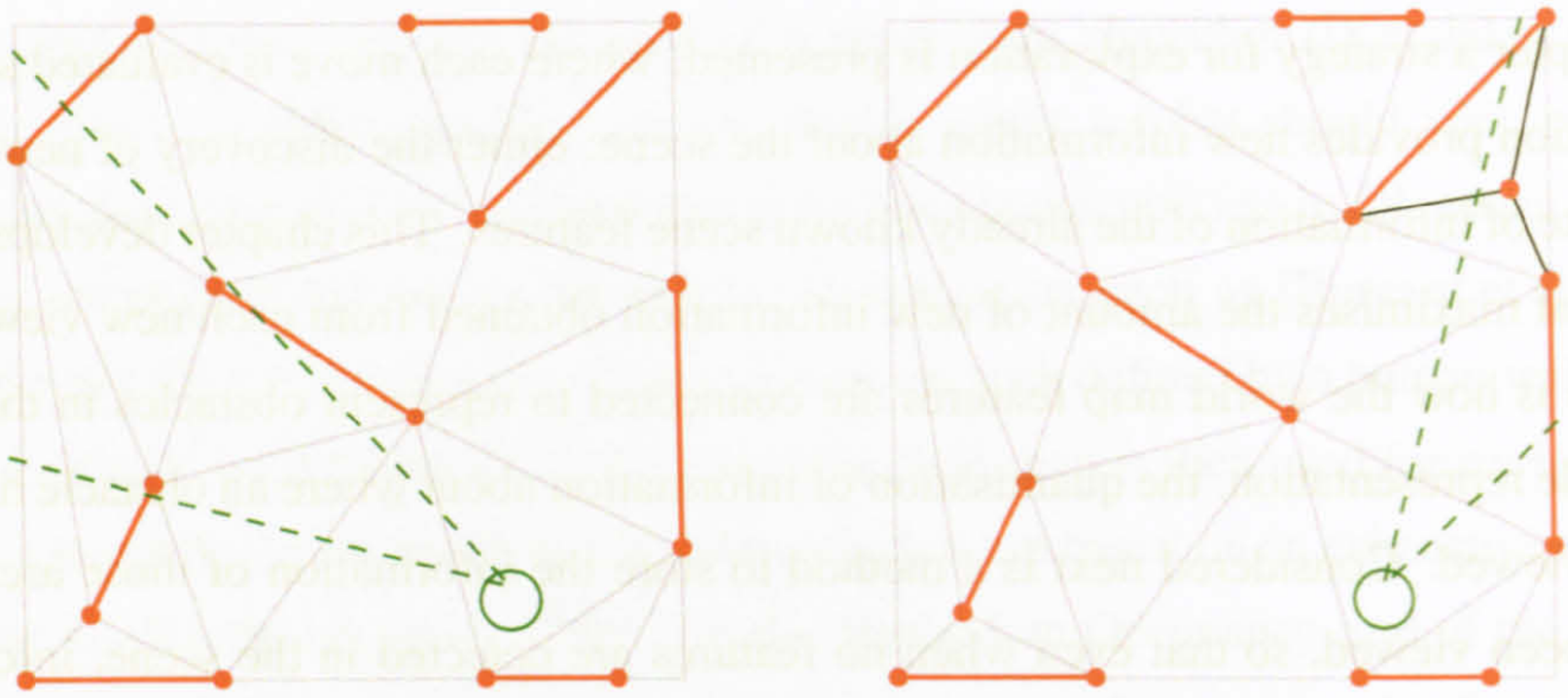
Each of these three methods provides a quantised evaluation of the view of an obstacle from its corresponding next-best-viewpoint, discussed in the previous chapter. By selecting the next viewpoint as the best from the choice of quantised evaluations at each robot position, an exploration of an unknown environment can proceed.

4.2 Internal Storage of the World Information

Many map building algorithms such as [Davison and Murray, 2002] store position information of scene features with no connections between the neighbouring features. Feature positions on their own do not provide all the information necessary for an algorithm to determine which areas are safe to navigate through, i.e. whether the area between two features creates an obstacle to the robot, or a traversable opening. A triangulation of the detected scene features in two dimensions generates a simple non-overlapping structure suitable for encoding this information.

Obstacle features detected from the scene are stored in a two-dimensional world map. When new features are added into this map, a triangular mesh is maintained such that each feature is connected to other features by non-intersecting connecting lines. Connecting lines (construction-lines) between world map features either represent real obstacles in the scene or are constructed solely to maintain the triangular structure of the mesh. When the mesh is initially constructed, all construction-lines are assumed to be obstacles to the robot. The next section discusses how these obstacles are distinguished from traversable openings.

Figure 4.1 shows an example triangular mesh construction. Figure 4.1(a) shows a triangulation with the robot at the position shown by the green circle, and the viewing region indicated by the dashed green lines. Figure 4.1(b) shows the mesh after a new feature has been inserted into the mesh. When a new feature is to be added into the mesh, the triangle which surrounds the position of the new feature is found. The new feature is connected to this triangle's features with three construction lines, shown in Figure 4.1(b) as black lines, and these new lines become part of the triangulation.



(a) Triangular mesh before additional feature

(b) Triangular mesh after additional feature

Figure 4.1: Example triangulation of world map features. Camera position shown in green; scene features in red, construction-lines in grey.

4.2.1 Confidence of world obstacles

A constraint for an environment described by [Manessis, 2001] states that a line from the camera to a scene feature cannot be intersected by a scene obstacle. Figure 4.1 shows a robot position and its current view of two world features. Lines from the camera to these two features (shown in the Figure by green dashed lines), the visibility lines, intersect three construction lines. The visibility constraint is violated for both features and the intersected construction lines therefore cannot represent environment obstacles.

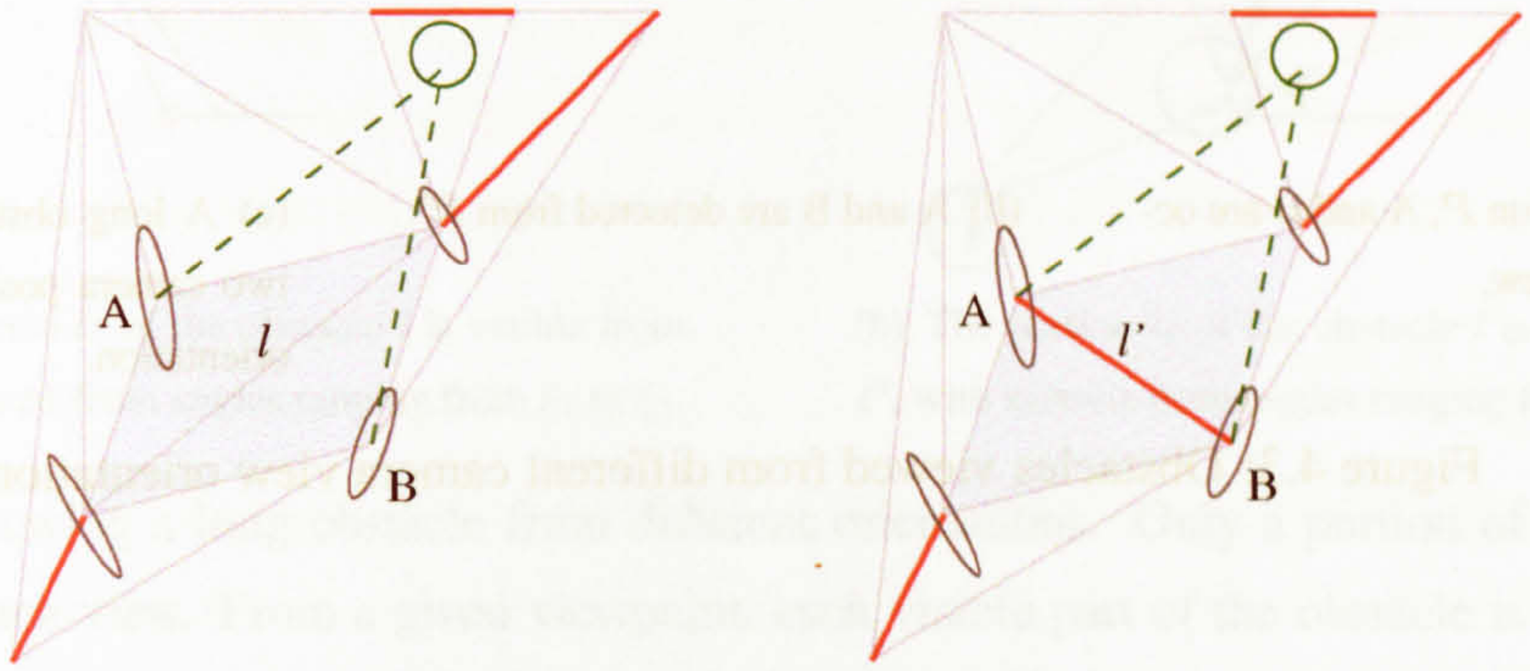
The visibility constraint described by [Manessis, 2001] considered only a binary solution, either construction lines are obstacles, or they are not. Since the model of the environment used in this work includes features which are uncertain (termed in equation [2.13] by C_f), a formulation for the visibility violations that encompasses this uncertainty is required. The more often a construction line is intersected by visibility lines and the higher the confidence of the world feature's existence that forms the visibility lines, the less likely the construction line is to be an obstacle. Using basic probability theory, and the assumption that each feature's confidence is independent, a confidence measure of a construction line being a real obstacle, C_o, can be calculated using the N visibility constraint violations and the corresponding feature confidences C_f.

$$C_o = \prod_{i=1}^{N} \left(1 - C_{f_i}\right) \tag{4.1}$$

The path of the robot is deemed feasible if it does not intersect any construction line labelled with $C_o \geq 0.7$. When first created, each construction line has a $C_o = 1$, until the obstacle violates a visibility constraint.

4.2.2 Inserting the correct obstacle

When a new feature is added into the triangulation, construction lines are created between that feature and all adjacent features. However, lines added in this way may not necessarily represent an environment obstacle i.e. two vertices detected on a scene obstacle may be added to the mesh without their connecting obstacle line, rather the line's quadrilateral pair.



(a) Incorrect line l inserted into the triangulation

(b) Correct line l' swapped into the triangulation

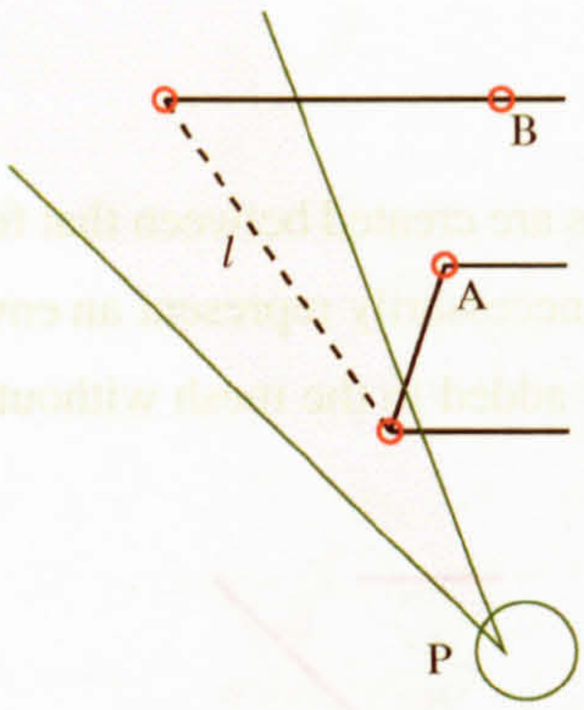
Figure 4.2: Evaluation of obstacles in the world map: testing construction-line's quadrilateral pair.

To illustrate this, Figure 4.2 shows an example of two world features A and B that have been added into the triangulation. The triangulation in Figure 4.2(a) is initially calculated. To represent the scene correctly however, A and B should be connected by a construction line and labelled as an obstacle, as in Figure 4.2(b). In Figure 4.2(a) a path between A and B is a valid path. To overcome this problem, after each set of new feature data is added to the world map, the triangulated mesh of world obstacles is evaluated. Evaluation is done by trying alternative diagonals of each quadrilateral in the world map mesh using the value of C_o, equation [4.1], to identify the line which is the more likely obstacle. This is all done before any analysis of the world map for exploration is carried out.

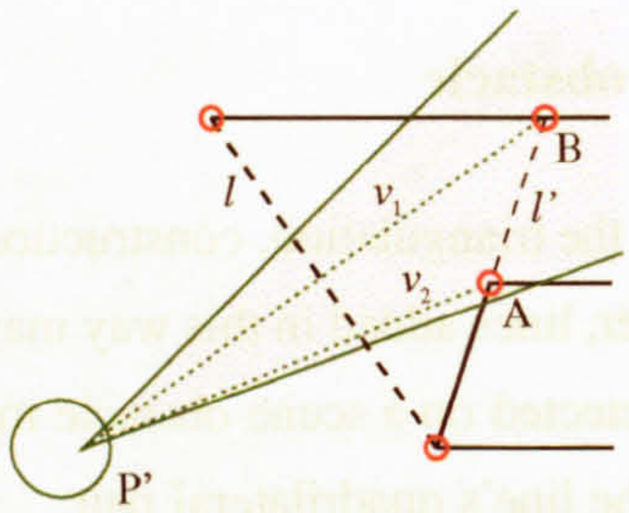
4.3 View-Improvement

Obstacles in the world map potentially occlude unseen features in the environment. Construction lines may correctly represent real obstacles, or they may be currently labelled as such but only because thus far they have only been viewed from positions where features they occlude have not yet become visible. Figure 4.3(a) shows the robot at position P with its camera's view direction and field-of-view. When first added into the triangulation the construction line l will be labelled as an obstacle since the features A and B have not yet been viewed. When the robot moves to position P' in Figure 4.3(b), A and B are visible. The visibility lines v_1 and v_2 intersect the construction line l, the confidence measure C_o is then recalculated.

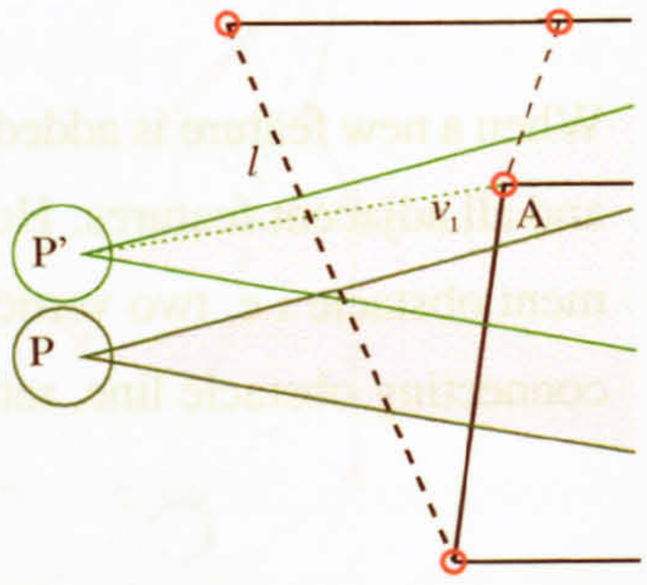
When selecting a position to move to, knowing that when viewing a construction line from different



(a) From position P, A and B are occluded from view.



(b) A and B are detected from P'.



(c) A long obstacle l, viewed from two camera positions with the same orientation.

Figure 4.3: Obstacles viewed from different camera view orientations.

positions a new scene features may become visible, is a useful piece of information. By storing information from the positions a construction line has been viewed from, a change in viewing positions and therefore an improvement of a viewing position can be calculated for a potential viewpoint.

Figure 4.3(c) shows an example for a long construction line l. The new information obtained by moving from P to P' could be achieved without moving the robot but by instead altering the orientation of the camera. The calculation of the view improvement for an obstacle therefore needs to incorporate the angle from which it was seen.

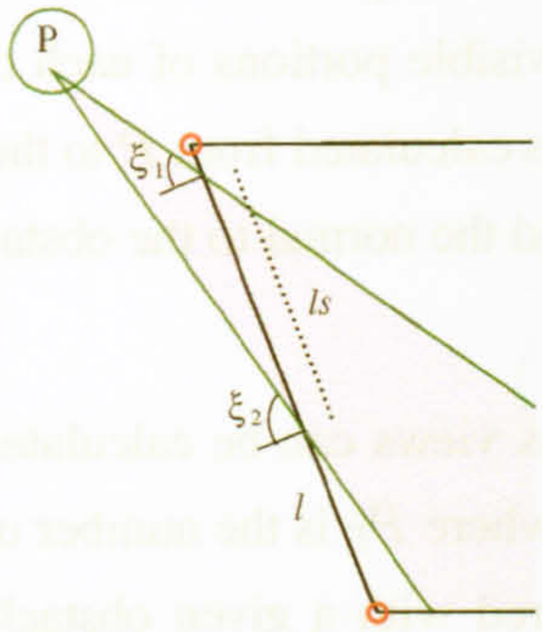
In the next three sections, three factors that can be used to quantify the likely improvement in knowledge of the environment due to a change in viewpoint will be considered.

4.3.1 Obstacle-view improvement

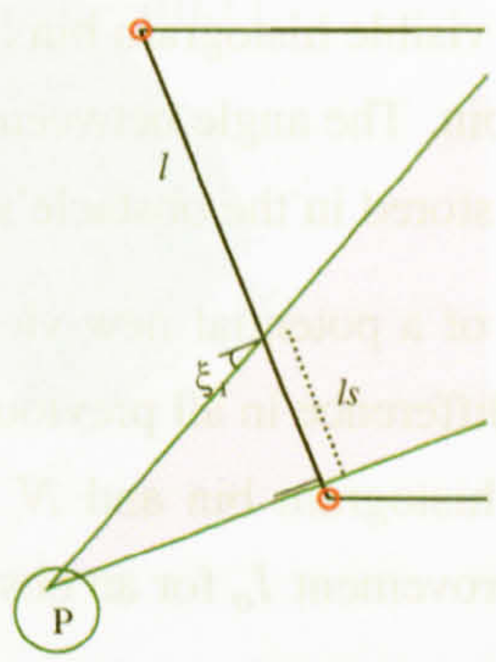
An entire obstacle may not be visible from a given viewpoint. Figure 4.4 shows two example views of an obstacle l; in each case, only the portion ls is visible from the position P.

In order to calculate a difference in the view compared to all previous views it is necessary to store information about previous views. As indicated in Figure 4.4, the entire length of an obstacle may not be visible from each viewpoint. It is therefore necessary to somehow store the information of which portions of each obstacle have been viewed.

Figure 4.4(a) shows the obstacle viewed from one orientation, and Figure 4.4(b) from another. As an example, assume that the centre of the obstacle length has been viewed from both orientations. It is possible that different information could have been acquired from behind the obstacle from each of these viewpoints. It is therefore not enough to just store information that a portion of an obstacle has been viewed, but also the angle from which that portion of the obstacle was viewed from. It should be noted that as each portion of an obstacle length is considered (from a given viewpoint), the



(a) The portion ls of the obstacle l is visible from P, with viewed-from angles ranging from ξ_1 to ξ_2.



(b) The portion ls of the obstacle l is visible from P, with viewed-from angles ranging from ξ_1 to 0.

Figure 4.4: Viewing a long obstacle from different orientations. Only a portion of the obstacle is visible from each view. From a given viewpoint, each visible part of the obstacle is viewed from a slightly different angle (measured from the normal to the obstacle).

angle from which that portion of the obstacle was viewed from changes. In Figure 4.4(a), the angle from which the obstacle is viewed from increases from ξ_1 to ξ_2, with reference to the normal of the obstacle direction.

To encapsulate the information about which portions of each obstacle have been viewed and which angle they were viewed from, a 1D histogram is stored with each obstacle that represents each portion of the obstacle along it's length. Figure 4.5 shows such a histogram for an obstacle l with N equal sized bins, and normal vector $\mathbf{n}$.

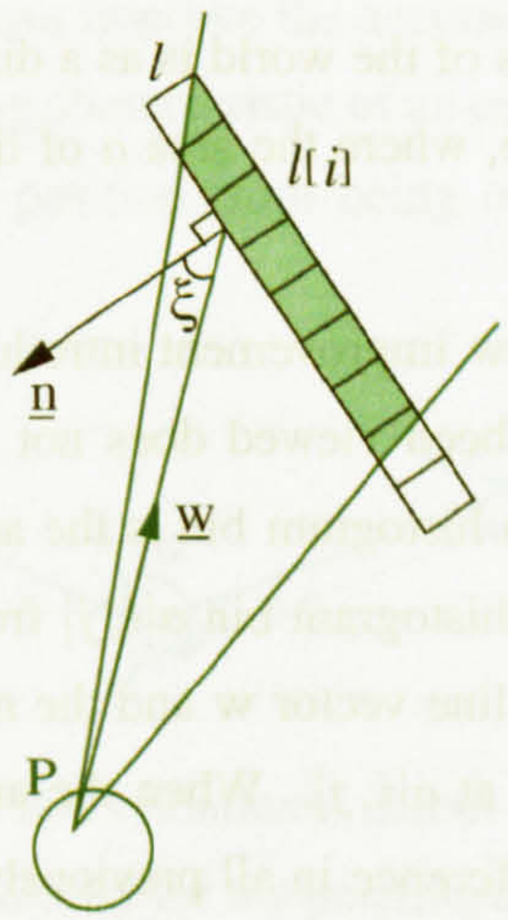


Figure 4.5: 1D histogram for obstacle l with nine histogram bins. Angles from which the obstacle is viewed are stored in each visible histogram cell. Shown is an example viewline vector $\mathbf{w}$ from P, and the storage of the angle ξ at cell $l[i]$.

From each camera viewpoint, each obstacle visible from that viewpoint is updated with the new view information. However, only the histogram bins that represent the visible portions of each obstacle are updated. For a visible histogram bin $l[i]$, the viewline vector $\mathbf{w}$ is calculated from P to the centre of that histogram bin. The angle between this viewline vector $\mathbf{w_i}$ and the normal to the obstacle $\mathbf{n}$ is calculated (ξ) and stored in the obstacle's histogram bin at $l[i]$.

The improvement of a potential new view compared to all previous views can be calculated using the accumulative difference in all previously stored views $p \leq P_T$, where P_T is the number of views stored in a given histogram bin and N is the number of bins stored with a given obstacle. The obstacle-view improvement I_o for an obstacle viewed from position X is defined as

$$I_o = \sum_{i=1}^{N} \prod_{p=1}^{P_T} (\mathbf{n}.\mathbf{w_{iX}} - \mathbf{n}.\mathbf{w_{ip}}) \tag{4.2}$$

where $\mathbf{w_{iX}}$ is the viewline vector from X to the centre of the histogram bin $l[i]$. The more histogram bins of an obstacle that are seen, and the larger the difference in the stored angles, the larger the improvement for a new view. That is to say, that an obstacle viewed from the same angle but different positions does not improve the view for the obstacle as much as viewing the obstacle from the same position with a different view orientation.

4.3.2 Area-view improvement

The detection of features and their subsequent triangulation and identification of obstacles and openings greatly informs the robot exploration algorithm. When, however, no features are detected, the information that a section of the environment was viewed and that no features were detected is also useful. The storage of this information and its use to calculate an improvment of a potential new view is considered in this section.

A simple way to store the seen parts of the world is as a discrete grid of equal-sized bins forming a 2D histogram, as seen in Figure 4.6, where the area a of the scene able to be explored is split into equal sized bins.

In the same vein to the obstacle-view improvement introduced in the previous section, the number of times a portion of the area has been viewed does not encapsulate all the available and useful information. Instead, stored in each histogram bin is the angle from which that portion of the area was viewed from. For each visible histogram bin $a[i,j]$ from a given viewing position, the viewed angle ξ is calculated using the viewline vector $\mathbf{w}$ and the normal to the centre of the histogram bin $\mathbf{n}$ and stored in the area histogram at $a[i,j]$. When the area is viewed from another position and orientation P', the accumulative difference in all previously stored views $p \leq P_T$ (where P_T is the number of views stored in a given histogram bin) can be calculated. The area-view improvement I_a for the visible area viewed from position X is defined as

$$I_a = \sum_{i,j} \prod_{p=1}^{P_T} (\mathbf{n}.\mathbf{w}'(i,j)_X - \mathbf{n}.\mathbf{w}(i,j)_p) \tag{4.3}$$

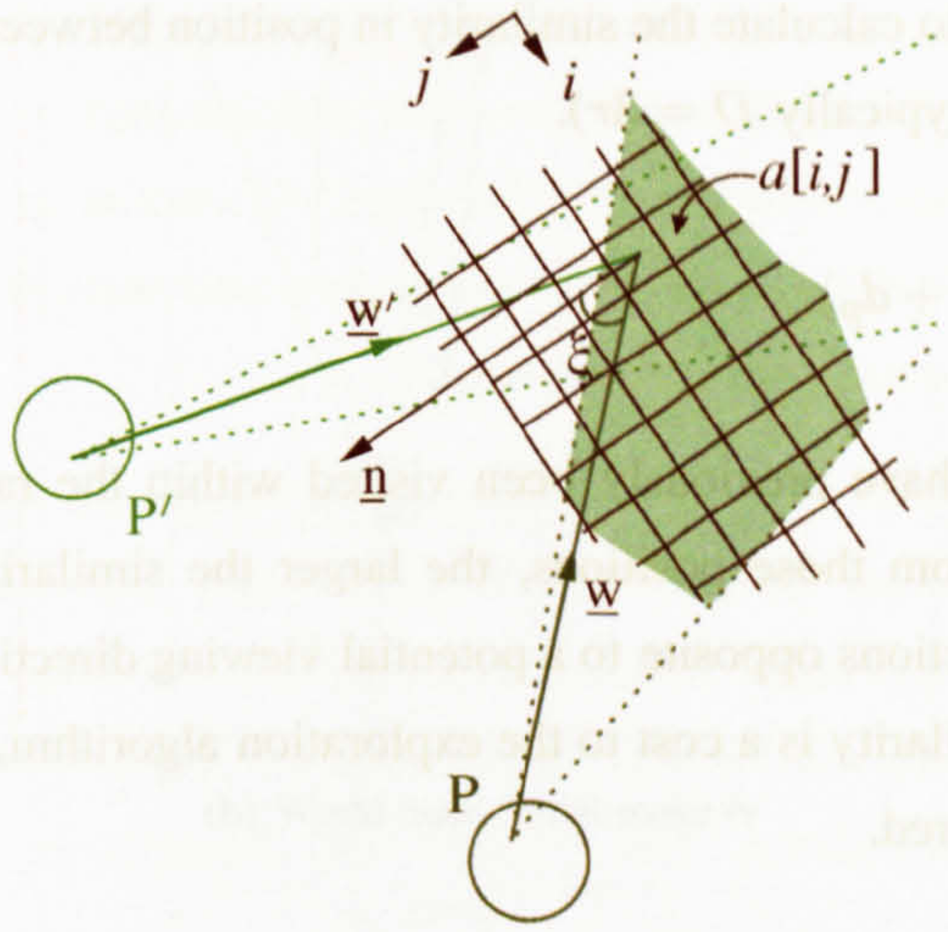


Figure 4.6: 2D histogram for scene area, a. Angles from which each area portion is viewed are stored in each visible histogram cell. Shown is an example viewline vector $\mathbf{w}$ from P, and the storage of the angle ξ at cell $a[i, j]$. A second example viewline vector $\mathbf{w}'$ from P' is also shown.

The larger the viewed area and the greater the difference in the stored angles in the area's histogram, the larger the improvement for a new view. When new features are added into the world map, the parts of the area that have been seen may change. After each set of new features has been added to the world map, the seen histogram is evaluated before the the area-view improvement is calculated.

4.3.3 Similarity in view position

Similar viewing positions provide little new information about the world map. Viewing the same features from a very similar position can improve the accuracy of the feature positions or the robot position. However, to aid the pioneering characteristic of an exploratory robot, a measure of position-similarity is used to deter a viewing position from being re-visited. Figure 4.7 shows the robot

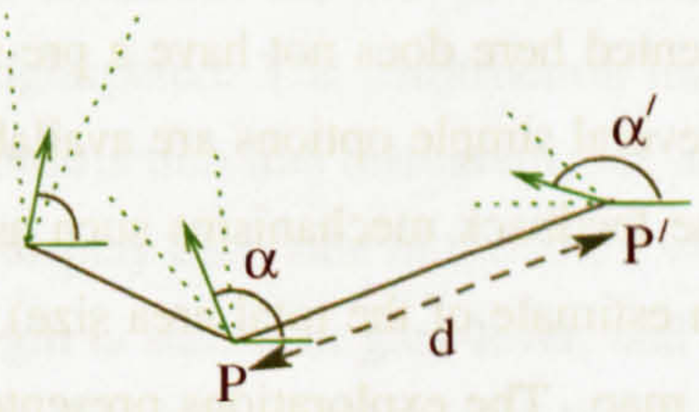


Figure 4.7: Similarity in view position and orientation. From the camera position P with orientation α (also shown is field-of-view), the similarity for the position P' and orientation α' is calculated using the distance between the two positions d and the difference in orientation angles using equation [4.4].

position P and a candidate position P' with viewing directions α and α' respectively. The distance between the camera positions, labelled d, and the view orientations with respect to the horizontal

axis, labelled α, are used to calculate the similarity in position between P' and all previous positions ($p \leq P_T$) where $d < D$ (typically $D = 3r$).

$$I_p = \sum_{p=1}^{P_T} (|\cos(\Delta\alpha_p)| + d_p) \tag{4.4}$$

The more positions that have previously been visited within the radius D and the more similar the viewing directions from those positions, the larger the similarity in position will be. Since $\cos(\pi) = 0$, viewing directions opposite to a potential viewing direction do not greatly contribute to this calculation. This similarity is a cost to the exploration algorithm, hence the minimum position-similarity should be favoured.

4.4 Exploration using View-Improvement Calculations

In this chapter, three methods to calculate the improvement of a new view of a scene have been considered: using information from the previously viewed regions of obstacles, scene area, and visited positions. In this section, using each of these improvement measures separately, the explorations of an unknown environment are shown.

The exploration algorithm starts with an initial scan of the environment, which is identical for each method of exploration. No information of the position or number of obstacles in the scene is known prior to the exploration. The robot builds up the map relative to its starting position of $(0, 0)$. An initial scan of the environment (16 views at 24° intervals) is done to build up a representation of the local area. Since the camera is offset from the centre of the robot, as the robot rotates on its $(0, 0)$ position, sideways movement of the camera is achieved during this scan, therefore mapping of features using back-projection as discussed in section 2.4.2 is possible. Neighbouring scene features are assumed to be obstacles until the construction line that defines the hypothesised obstacle boundary is violated by visibility constraints of other obstacles.

The exploration algorithm presented here does not have a pre-set criteria for when the exploration will automatically terminate. Several simple options are available e.g. time, distance travelled, battery power, etc. as well as scene feedback mechanisms such as the percentage of the area that has been viewed (which requires an estimate of the total area size), or a measure of the amount of new information being added to the map. The explorations presented in this and the next chapter were terminated by the operator when enough of the algorithm's behaviour had been displayed.

Figure 4.8 shows the map being built from the test scene of Figure 4.8(a). There are five internal obstacles, the length and position of which are randomly added to the scene, of area 9m×10m. Sixteen equal length obstacles are placed around the exterior of the environment. As new scene features are detected they are added into the world map representation, and the confidence of each construction line being an obstacle, C_o, is recalculated using the visibility lines from the new view.

(a) Scene obstacle map (b) World map: initial move iv (c) World map: initial move x

(d) World map: initial move xvi (e) Seen obstacles histograms (f) Seen area histogram

Figure 4.8: Initial moves in the basic map scene: Features visible from each camera position (shown in green) are added into the world map. Features detected from the most recent camera position are shown in pink. Correctly detected obstacles highlighted in red. All construction lines displayed to grey-level of C_o, with $C_o = 1$ drawn in black (for clockwise line orientation); counter-clockwise orientated lines drawn to blue-level.

Previous robot view position and orientations are shown. The current view position has the field-of-view region and visible features highlighted. The construction lines are drawn to a grey-level value to represent their C_o value. The construction line histograms are actually drawn as two lines slightly offset from each other in order to display each side of the line's view histogram. The clockwise line orientation with respect to the origin is shown in grey-level, and the counter-clockwise orientation line is shown in varying blue-level. When the construction line has been seen from both sides, both of these line histograms are shown. If the construction line has a very high C_o value and has been at least partially seen, it is highlighted in red (the algorithm does not receive this information).

For example, Figure 4.8(b) shows the view from the camera position at the fourth step of initial the scan. Three scene features are detected from this view and are added into the world map. After the features are added to the world map, the obstacle confidence, C_o, of each construction line within the triangular mesh is evaluated. The values of these obstacle confidences is displayed in grey-level,

with $C_o = 1$ drawn in black and $C_o = 0$ drawn in white. Figure 4.8(b) shows one correctly detected obstacle (highlighted in red), and three (incorrectly) hypothesised obstacles (drawn in black), These three hypothesised obstacles are constructed between detected scene features, but do not correctly represent the scene obstacle locations. As the exploration continues, these obstacles should disappear when other scene features are viewed through them.

After the initial scan, three of the five internal and five of the sixteen external obstacles have been found. The fourteen obstacles with $C_o \geq 0.7$ (eight which are correctly labelled and the six potentially visible others) are initially considered for further viewing. Positions providing the next-best-viewpoint are calculated and, if still visible from the hypothesised position, the improvement measure (I_o,I_a, or I_p) is calculated. The obstacle with the maximum improvement determines the next move.

The exploration of this test scene using each of the three improvement measures in turn, will now be considered. For the exploration descriptions that follow, it should be noted that three options of distance to move (r in Figure 3.5) are used to calculate three next viewpoints for a given obstacle: 0mm (rotation only), 400mm and 1200mm. This allows the robot to not traverse if it has found a good position to view several obstacles from, or to quickly move to a new area if the current position does not allow for good viewpoint possibilities. For each viewpoint calculated, if the obstacle is viewable from the new position by a number of orientations, options for view orientation are also calculated.

4.4.1 Exploration of a basic map using the obstacle-view improvement

Figure 4.9 shows the first few moves of the exploration algorithm using the maximum improvement of the view of obstacles in the world map (following the set of viewing positions shown in Figure 4.8). Each row shows one snapshot of the exploration. The left column shows the world map with the construction line histograms (only the number of entries in the line's histogram are shown). The next viewpoint selected is highlighted by the field-of-view lines of the camera and the obstacle used to calculate that viewpoint is highlighted by circles at the ends of the obstacle. The second column shows the updated obstacle map after the move, after all update calculations of the obstacles have been made. Previous robot position and view orientations are also shown. The third column, for reference, shows the new position of the robot within the scene.

The initial moves using the obstacle-view improvement select the longer construction lines at new view orientations of different obstacles in turn. The second move finds two new obstacles, one correct obstacle on the right external wall and one construction line. After each move, all obstacles have their confidence measures, C_o, recalculated. Since the obstacle chosen for move two now intersects the visibility lines from the new position, it will no longer be labelled as an obstacle. It therefore will not be considered for future choices of view improvement. Since two new obstacles are added into the world map, the number of obstacles to consider for view improvement does not decrease.

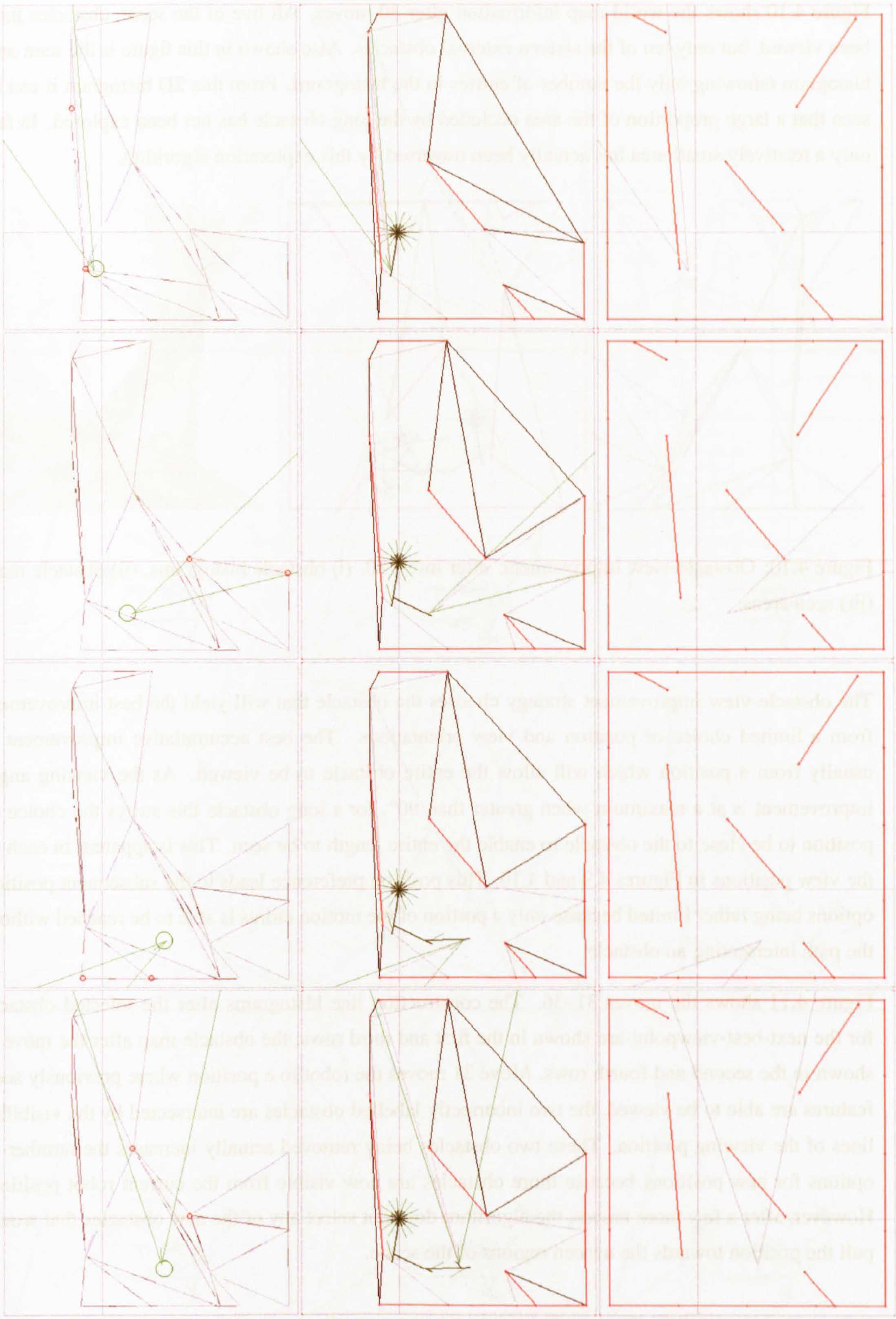


Figure 4.9: Obstacle-view improvement: moves 1–4. (i) next move on obstacle histogram map, selected obstacle highlighted with corner circles, (ii) obstacle map, (iii) position in scene.

Figure 4.10 shows the world map information after 30 moves. All five of the scene obstacles have been viewed, but only ten of the sixteen external obstacles. Also shown in this figure is the seen area histogram (showing only the number of entries in the histogram). From this 2D histogram it can be seen that a large proportion of the area occluded by the long obstacle has not been explored. In fact only a relatively small area has actually been traversed by this exploration algorithm.

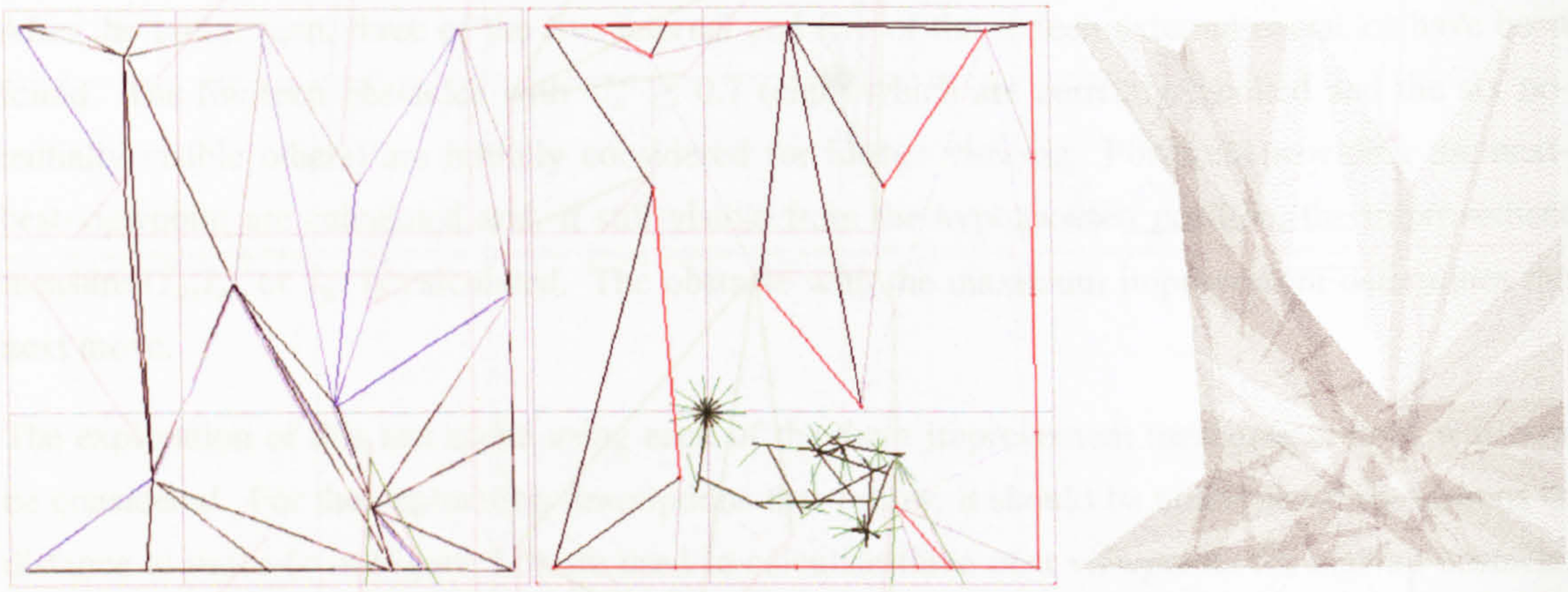


Figure 4.10: Obstacle-view improvement: after move 30. (i) obstacle histograms, (ii) obstacle map, (iii) seen areas.

The obstacle-view improvement strategy chooses the obstacle that will yield the best improvement from a limited choice of position and view orientations. The best accumulative improvement is usually from a position which will allow the entire obstacle to be viewed. As the viewing angle improvement is at a maximum when greater than 90°, for a long obstacle this sways the choice of position to be close to the obstacle to enable the entire length to be seen. This is apparent in each of the view positions in Figures 4.9 and 4.10. This position preference leads to the subsequent position options being rather limited because only a portion of the motion radius is able to be reached without the path intersecting an obstacle.

Figure 4.11 shows the moves 31–36. The construction line histograms after the selected obstacle for the next-best-viewpoint are shown in the first and third rows; the obstacle map after the move is shown in the second and fourth rows. Move 31 moves the robot to a position where previously seen features are able to be viewed, the two incorrectly labelled obstacles are intersected by the visibility lines of the viewing position. These two obstacles being removed actually increases the number of options for new positions because more obstacles are now visible from the current robot position. However, after a few more moves, the algorithm does not select any of the new obstacles that would pull the position towards the unseen regions of the scene.

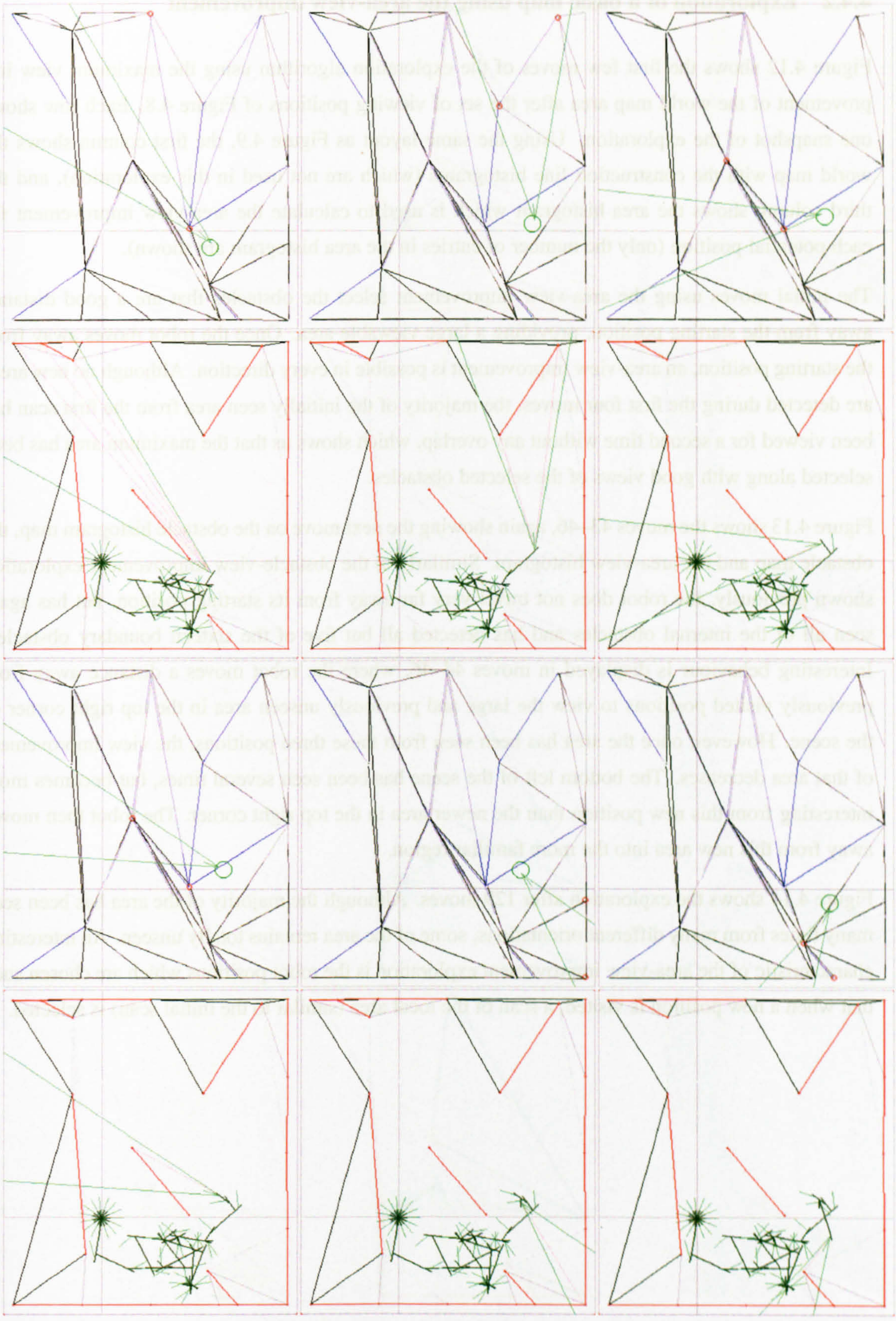


Figure 4.11: Obstacle-view improvement: moves 31–36. (i) obstacle histograms, (ii) obstacle map.

4.4.2 Exploration of a basic map using the area-view improvement

Figure 4.12 shows the first few moves of the exploration algorithm using the maximum view improvement of the world map area after the set of viewing positions of Figure 4.8. Each row shows one snapshot of the exploration. Using the same layout as Figure 4.9, the first column shows the world map with the construction line histograms (which are not used in this exploration), and the third column shows the area histogram which is used to calculate the area-view improvement for each potential position (only the number of entries in the area histogram are shown).

The initial moves using the area-view improvement select the obstacles that are a good distance away from the starting position, providing a large viewable area. Once the robot moves away from the starting position, an area-view improvement is possible in every direction. Although no new areas are detected during the first four moves, the majority of the initially seen area from the first scan has been viewed for a second time without any overlap, which shows us that the maximum area has been selected along with good views of the selected obstacles.

Figure 4.13 shows the moves 43–46, again showing the next move on the obstacle histogram map, the obstacle map and the area-view histogram. Similarly to the obstacle-view improvement exploration shown previously, the robot does not travel very far away from its starting position, but has again seen all of the internal obstacles and has detected all but five of the sixteen boundary obstacles. Interesting behaviour is displayed in moves 44–46, where the robot moves a distance away from previously visited positions to view the large and previously unseen area in the top right corner of the scene. However, once the area has been seen from these three positions, the view improvement of that area decreases. The bottom left of the scene has been seen several times, but becomes more interesting from this new position than the newer area in the top right corner. The robot then moves away from this new area into the more familiar region.

Figure 4.14 shows the exploration after 129 moves. Although the majority of the area has been seen many times from many different orientations, some of the area remains totally unseen. An interesting characteristic of the area-view improvement exploration is the robot positions which are chosen such that when a new position is visited, a scan of the local area (similar to the initial scan) is selected.

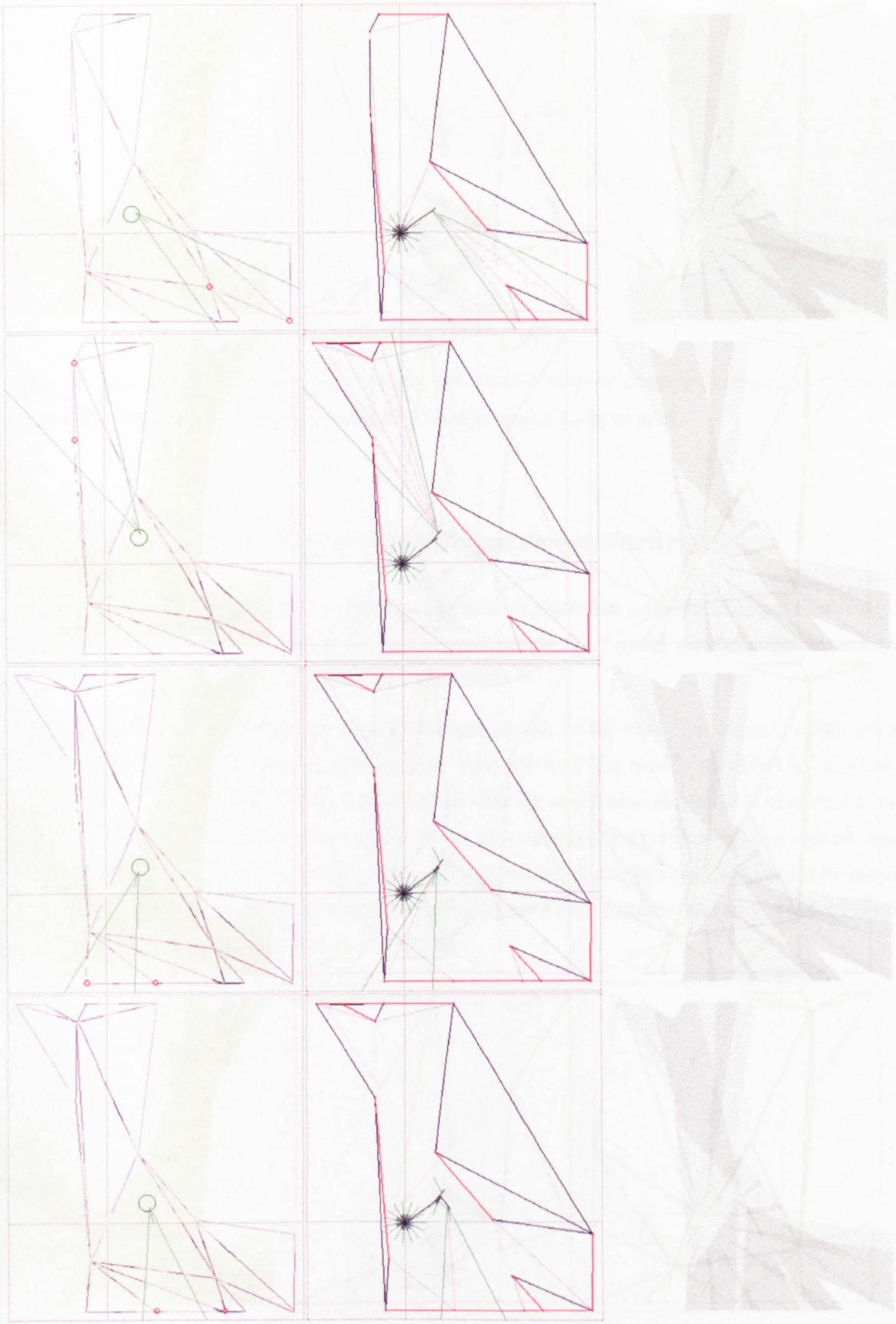


Figure 4.12: Area-view improvement: moves 1–4. (i) next move on obstacle histogram map, selected obstacle highlighted with corner circles, (ii) obstacle map, (iii) areas seen.

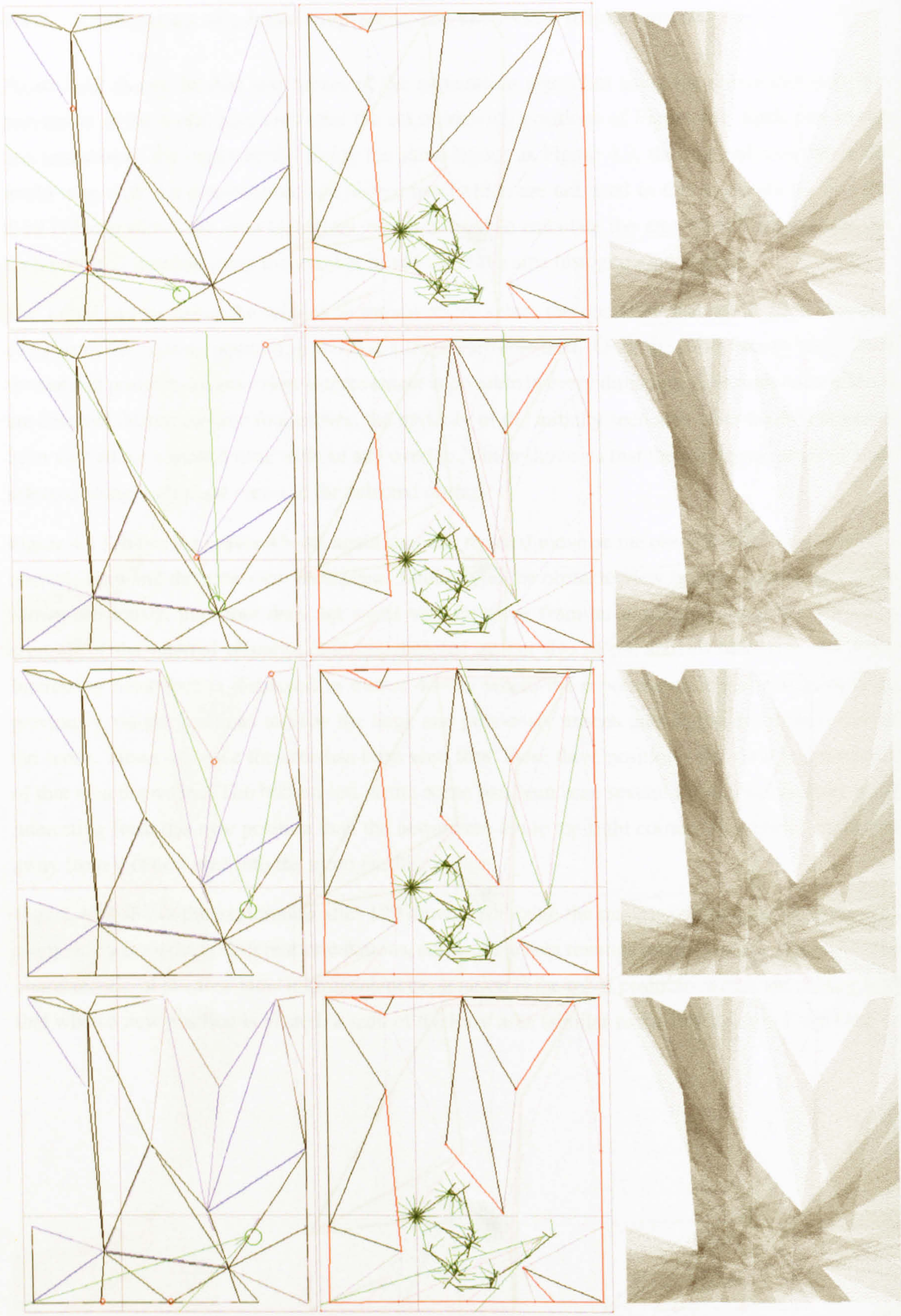


Figure 4.13: Area-view improvement: moves 43–46. (i) next move on obstacle histogram map, selected obstacle highlighted with corner circles, (ii) obstacle map, (iii) areas seen.

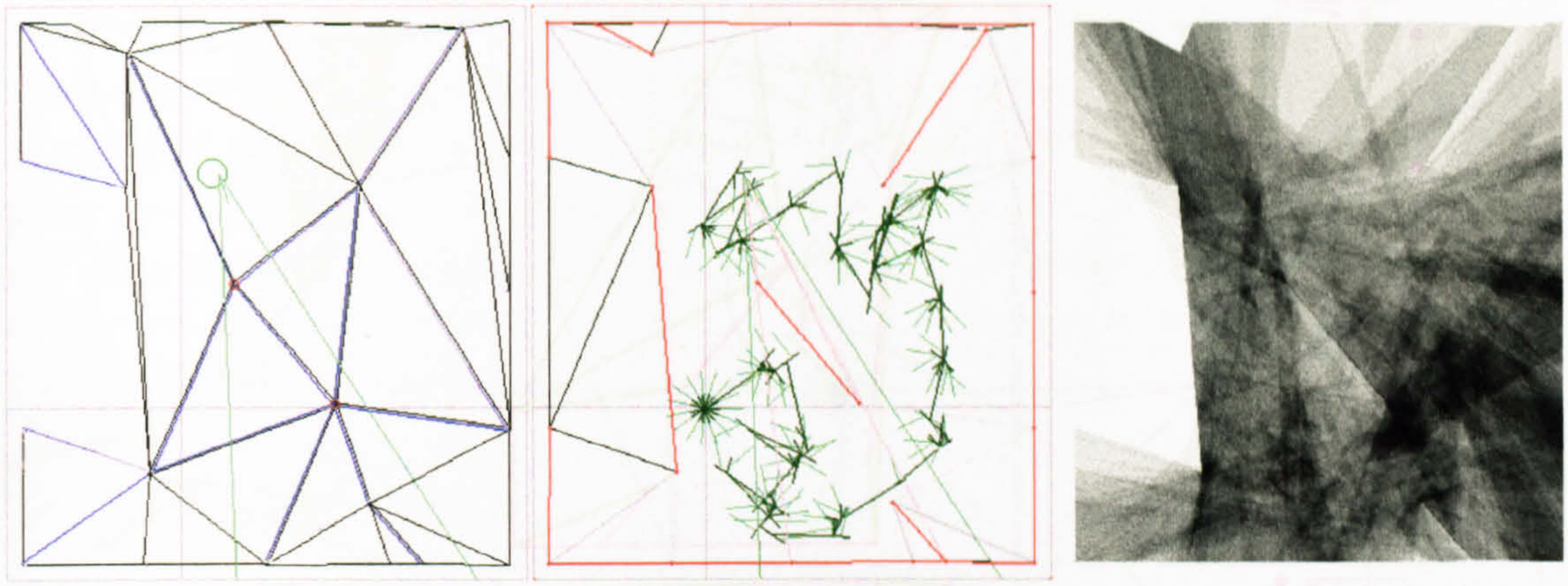


Figure 4.14: Area-view improvement: moves 129. (i) next move on obstacle histogram map, selected obstacle highlighted with corner circles, (ii) obstacle map, (iii) areas seen.

4.4.3 Exploration of a basic map using the position-similarity

Figure 4.15 shows the first few moves of the exploration algorithm using the minimum similarity of the previously visited viewpoints in the world map (after the set of initial viewing positions of Figure 4.8). Each row shows one snapshot of the exploration.

This exploration technique uses no measure of improvement for the view of an obstacle or the amount of area that will be seen from the new position. After the first four moves, the robot has travelled a good distance into the region of the scene viewed from the initial scan moves. It is important to note that the robot will never move into a region of the scene that has not previously been viewed, this is explicit in the algorithm because the unseen areas lay behind obstacles and a path cannot be planned through any obstacle. All five of the internal obstacles have been detected, but only six of the sixteen external obstacles have been viewed.

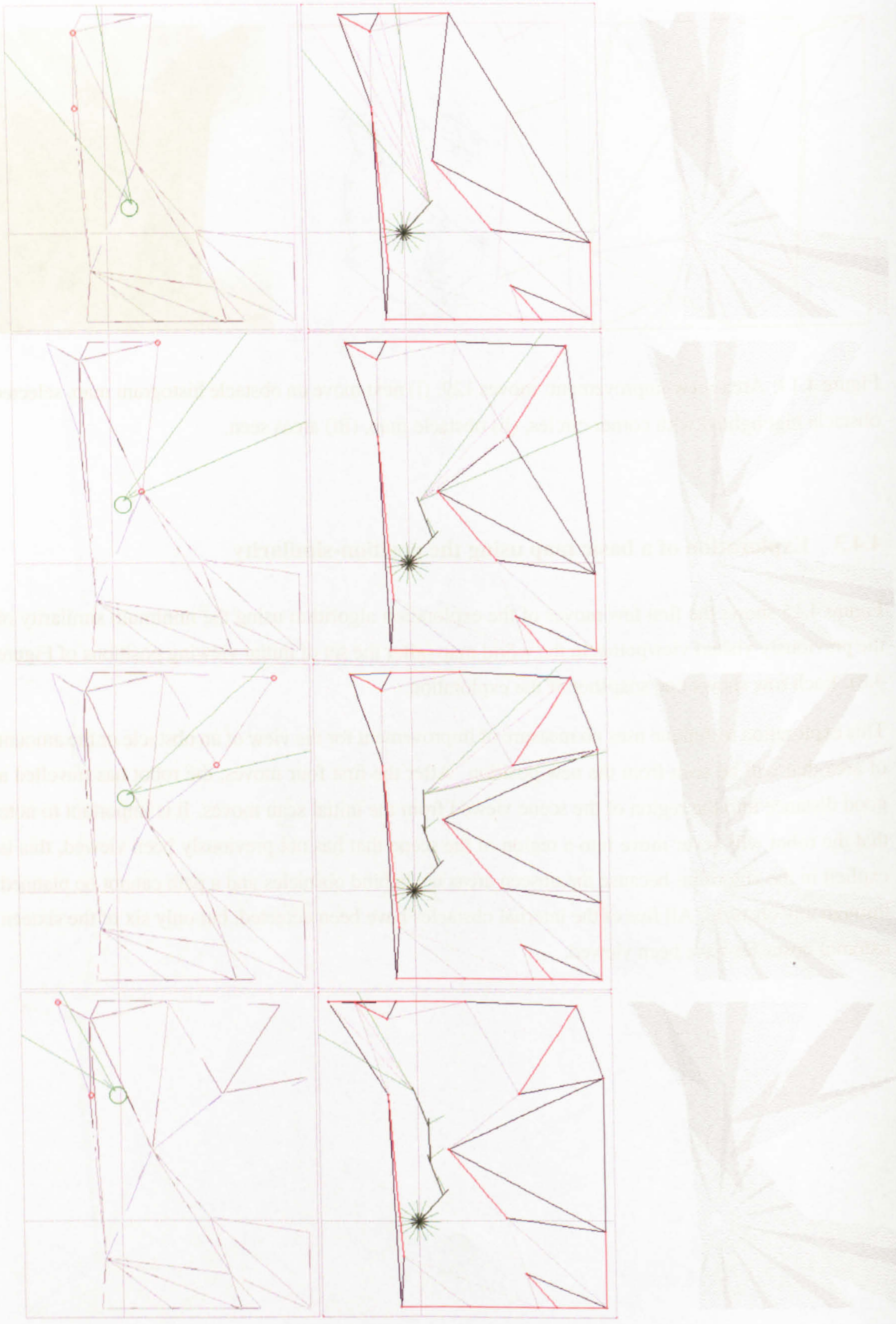


Figure 4.15: Minimum position-similarity: moves 1–4. (i) next move on obstacle histogram map, selected obstacle highlighted with corner circles, (ii) obstacle map, (iii) areas seen.

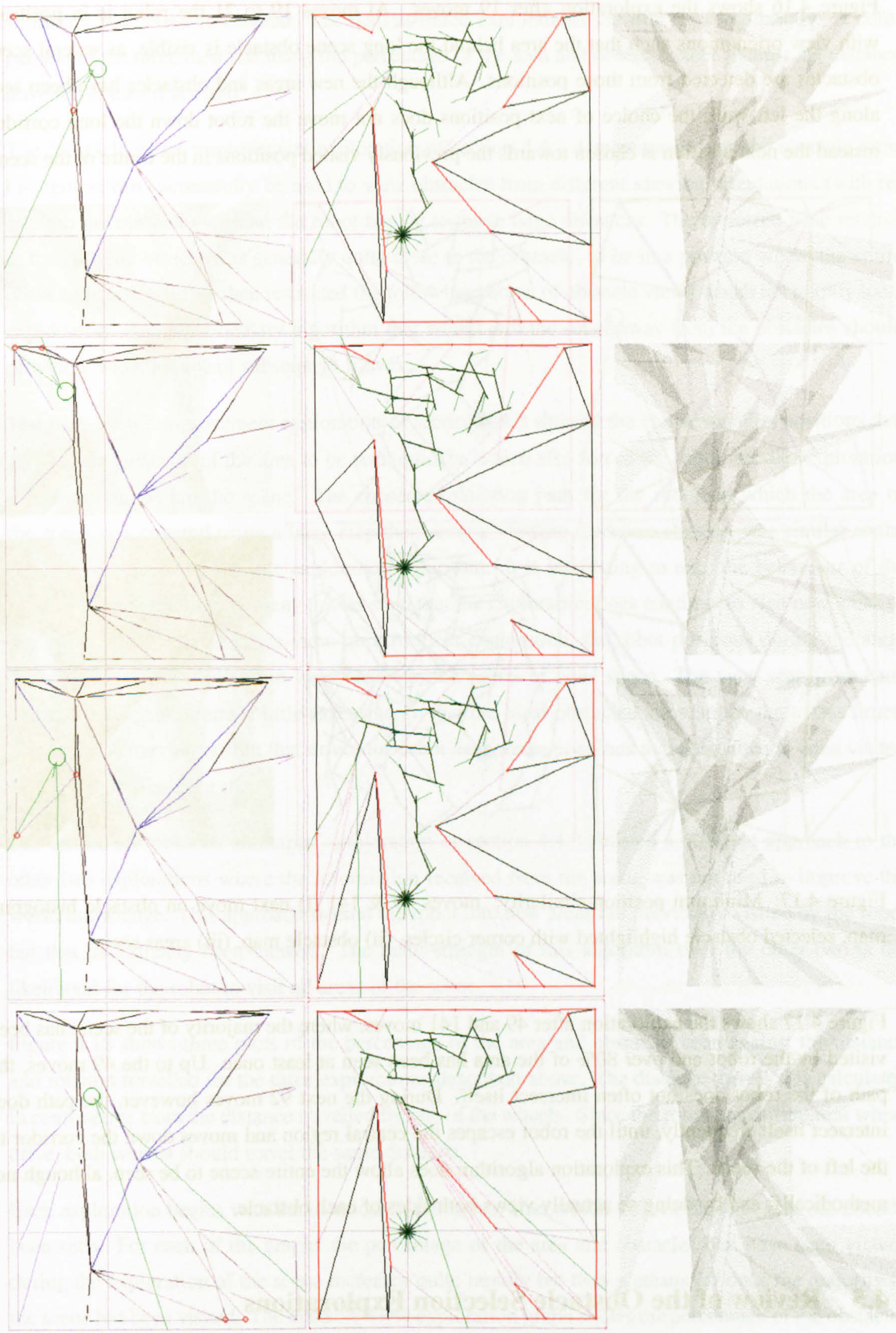


Figure 4.16: Minimum position-similarity: moves 19–21,23. (i) next move on obstacle histogram map, selected obstacle highlighted with corner circles, (ii) obstacle map, (iii) areas seen.

Figure 4.16 shows the exploration after 19 moves. At moves 19 to 21 the robot is in positions with view orientations such that the area behind the long scene obstacle is visible, as several scene obstacles are detected from those positions. Although the new areas and obstacles have been seen along the left wall, the choice of next positions does not move the robot down the long corridor, instead the next position is chosen towards the previously visited positions in the centre of the scene.

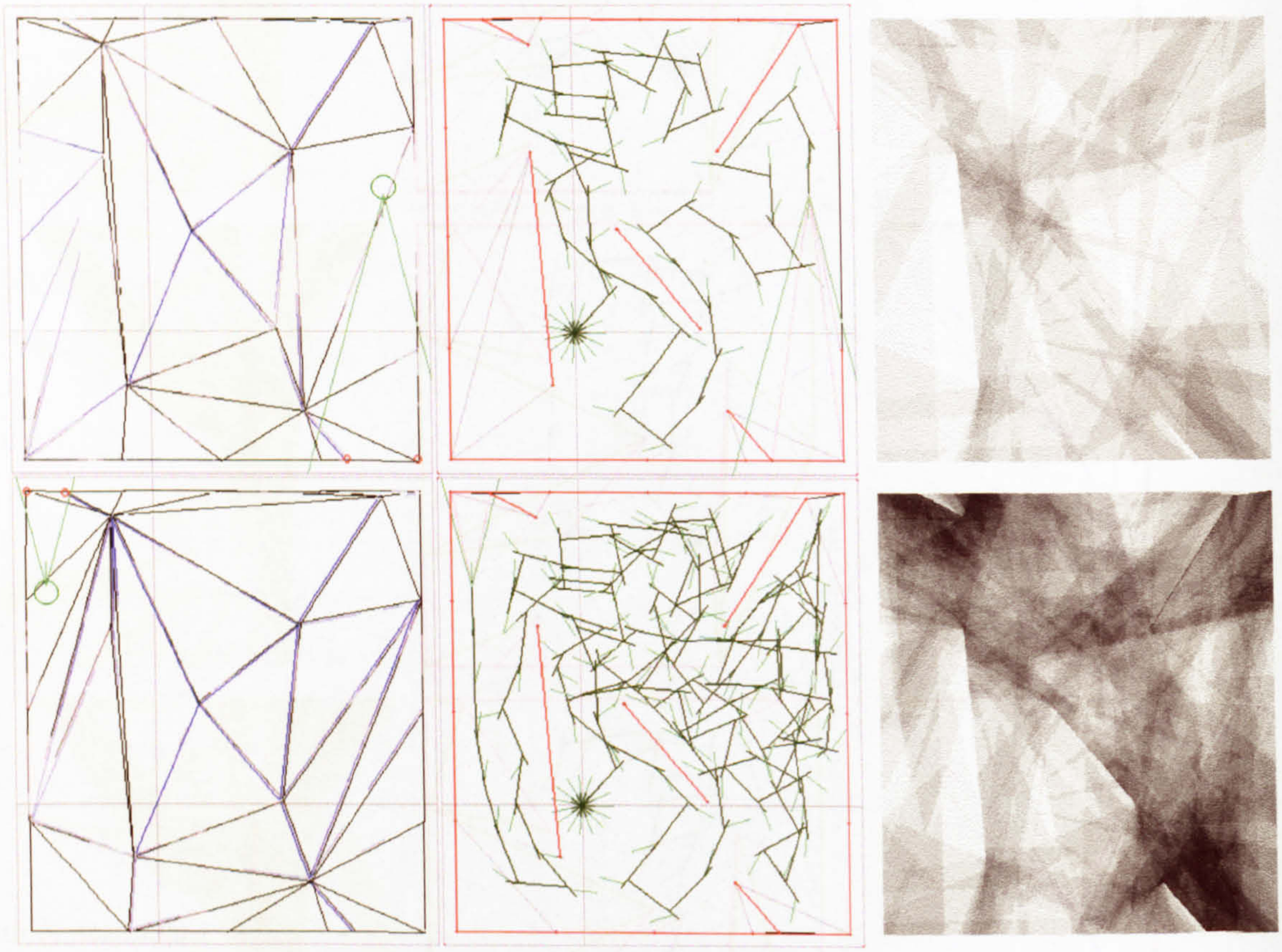


Figure 4.17: Minimum position-similarity: moves 49 & 141 (i) next move on obstacle histogram map, selected obstacle highlighted with corner circles, (ii) obstacle map, (iii) areas seen.

Figure 4.17 shows the exploration after 49 and 141 moves, where the majority of the scene has been visited by the robot and over 80% of the area has been seen at least once. Up to the 49 moves, the path of the robot does not often intersect itself. During the next 92 moves however, the path does intersect itself frequently, until the robot escapes the central region and moves down the corridor to the left of the scene. This exploration algorithm does allow the entire scene to be seen, although not methodically, and by doing so actually views both sides of each obstacle.

4.5 Review of the Obstacle Selection Explorations

The exploration of a basic scene with five interior and sixteen exterior obstacles using three methods: maximum obstacle view improvement, maximum area-view improvement and minimum position-

similarity have been considered. The demonstrated explorations are first reviewed briefly. A comparison of the three methods using the percentage of the area and obstacles seen against the distance travelled is then discussed.

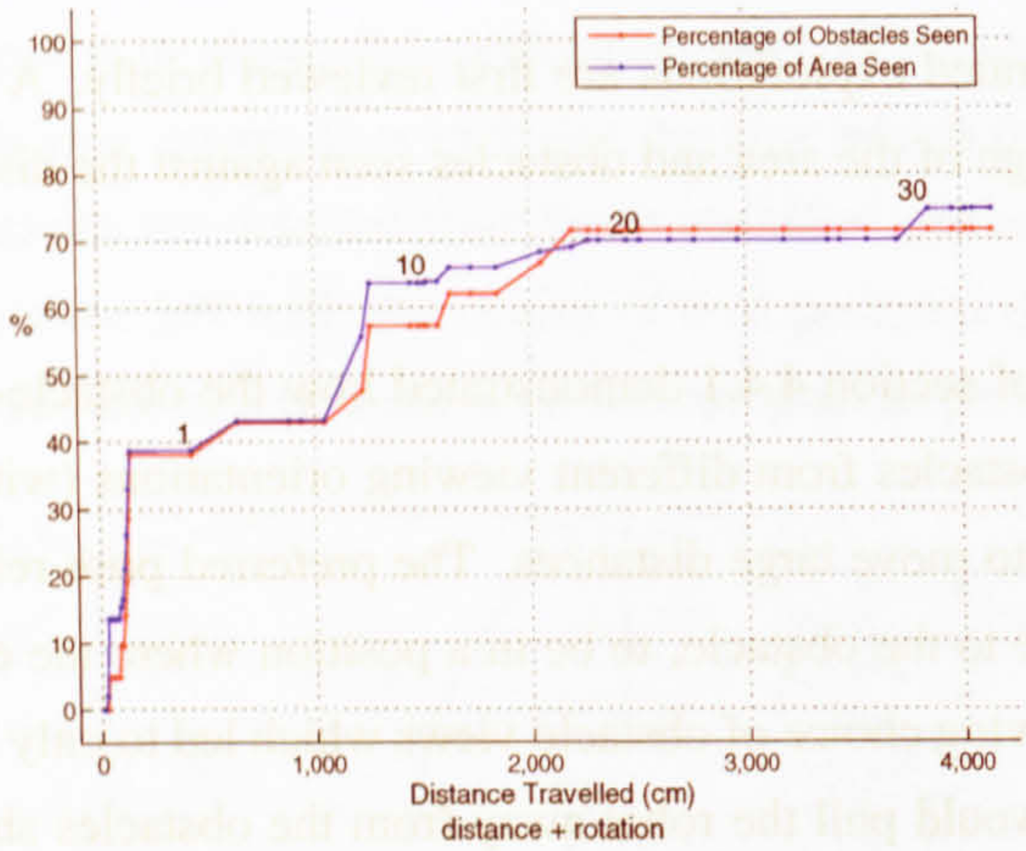
The 'obstacle-view' improvement exploration of section 4.4.1 demonstrated how the obstacle-view histograms can successfully be used to view obstacles from different viewing orientations (with respect to the obstacle) without the robot having to move large distances. The preferred pose relative to the chosen obstacle was generally quite close to the obstacle, to be in a position where the entire obstacle is visible. This then restricted the following choice of obstacle views which led to only local exploration. A complementary algorithm that would pull the robot away from the obstacles should provide a wider choice of subsequent moves.

The 'area-view' improvement exploration of section 4.4.2 showed the choice of robot positions that allowed the majority of the area to be seen quickly, which also forces the robot into the exploration of most areas within the scene. The chosen exploration path for the robot, in which the area of the scene was covered using a large step then several viewing direction choices, was similar to the original scan used prior to the exploration algorithm. It is interesting to note the behaviour of the area-view exploration over many moves such that the exploration does continue to visit new areas of the scene. Unlike the obstacle-view improvement exploration, the robot positions during the area-view improvement exploration were nearer to the centre of open space. The view scan from each new robot position seems a little excessive, in that the local obstacles are scanned numerous times. A complementary algorithm that drives the robot away from positions that have already been visited should counter-balance this.

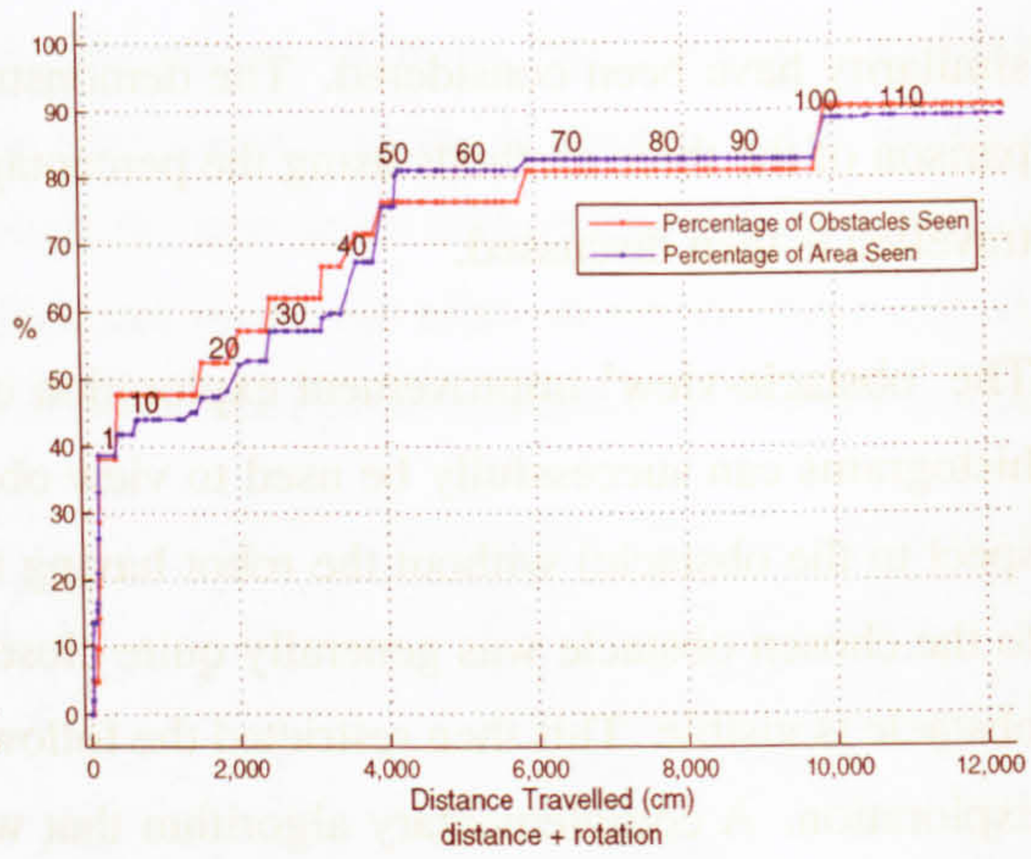
The minimum 'position-similarity' exploration of section 4.4.3 showed a different approach to the other two explorations where the information received from the scene was not used to improve the world knowledge. This approach steered the robot into new areas not previously visited by the robot but that had already been viewed. The main strength of this algorithm over the other two is the likelihood for the robot to visit all areas of the scene.

Figure 4.18 shows three plots of the percentage of the area and obstacles seen against the distance and rotation travelled for the three explorations described above. The distance travelled is calculated in centimetres from the distance travelled by one of the wheels. Since the robot has differential wheel drive, both wheels should travel the same distance.

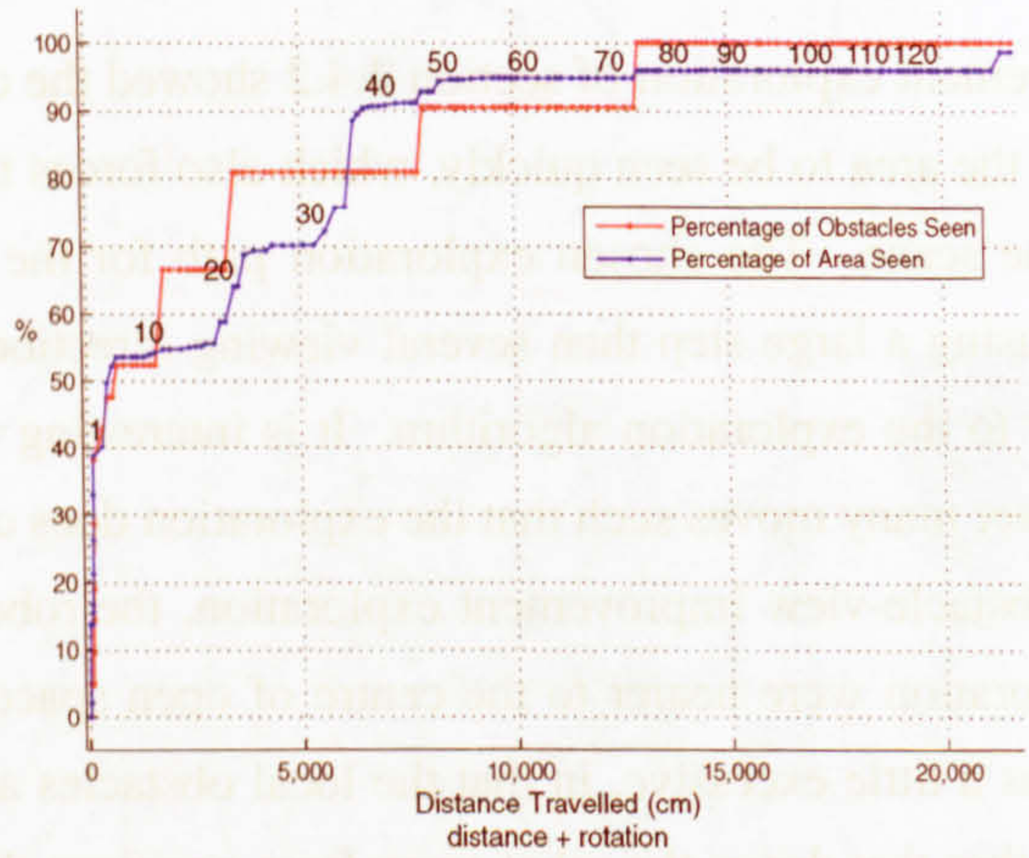
Each exploration begins with the identical initial move, after which 40% of the total area has already been seen. For each of the graphs the percentage of the area and obstacles that have been viewed during the exploration of the scene increases quite rapidly but then plateaus off once the majority of the scene has been visited. The obstacle-view exploration graph shows the percentage of the obstacles and the area seen grow at a similar rate, perhaps with the amount of area seen being slightly higher than the number of obstacles detected. The other two graphs however show how the amount of the area that is seen lags behind the number of obstacles detected.



(a) Obstacle-view histogram improvement



(b) Area-view histogram improvement



(c) Minimum position-similarity

Figure 4.18: Analysis of basic map exploration using area- and obstacle-view improvement and position-similarity measures: percentage of obstacles and areas seen versus total distance travelled.

The obstacle-view exploration of the basic scene does not successfully view all areas or all obstacles in the scene, however after 30 moves, 4000cm have been travelled with over 70% of the obstacles and area seen. After 4000cm during the area-view exploration have been travelled, 75% of the area and obstacles have been viewed, compared to 70% and 80% of the area and obstacles respectively during the position-similarity exploration. 90% of the area and the obstacles have been detected after 10,000cm during the area-view exploration, and after 8000cm during the position-similarity exploration. Since both of these explorations demonstrated the characteristic of choosing to move the larger of the distance choices, this makes sense.

4.6 Conclusions

In this chapter an exploration strategy for a vision system with limited field-of-view using three methods: maximum obstacle-view improvement, maximum area-view improvement and minimum

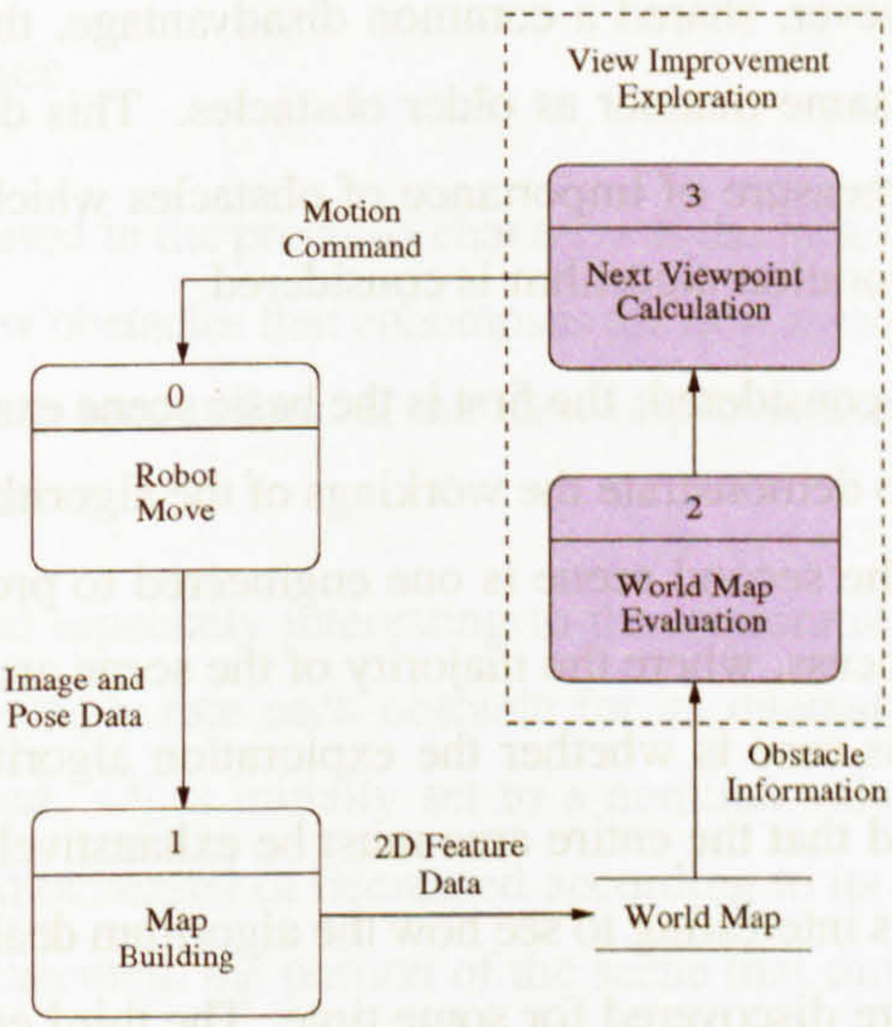
position-similarity has been presented.

Each of the methods use the same algorithm to provide a selection of viewing positions calculated using the next-best-view position of each obstacle, from Chapter 3. From the choice of viewing positions, the obstacle-view improvement, area-view improvement or position-similarity is calculated and the maximum improvement (or minimum similarity) is chosen for the next move. The example explorations shown have highlighted the advantages of each method and how they complement each other. Each of the three methods are missing one common factor, that each obstacle in the world map is considered equal i.e. there is no obstacle priority. This means that one obstacle may be viewed repeatedly whereas, should another obstacle be selected, new areas may be explored. Figure 4.16 and Figure 4.13 showed the situations where new features were detected but the new area was not explored further.

The next chapter uses all three methods described here, together with an obstacle priority measure in a joint improvement cost function to explore an unknown environment.

Chapter 5

Autonomous Exploration



In the previous chapter, three methods for exploration driven by: maximising the view of an obstacle using a measure of obstacle-view and area-view, and minimising the position-similarity were discussed. Each technique was considered separately in an example of a basic scene to highlight the functionality and use of each algorithm. The advantages and disadvantages of each were discussed. This chapter shows a joint-improvement technique for exploration by using an amalgamation of these three techniques. A variety of environments are shown in order to demonstrate the autonomous exploration algorithm and its ability to successfully explore and build up knowledge of an unknown scene, without the use of pre-planned actions for specific scenarios.

5.1 Introduction

An algorithm designed to autonomously explore an unknown environment which has no restriction to the configuration of obstacles within the environment must be able to function in the wide variety of situations that it may encounter during its exploration. Such situations may include: wide

open spaces with little restriction on movement, small spaces with numerous features and limited movement ability, narrow corridors, dead ends, and many others. Some researchers have focused on "hard-coding" an exploration algorithm with a limited set of options about how to handle a given set of situations, e.g. wall-hugging in narrow corridors, [Nehmzow and Owen, 2000]. Such techniques are limited by design since they do not allow a robot to handle any non-programmed events, and are also susceptible to errors in the classification of the particular section of the environment. The aim of this chapter is to use together the tools developed in the previous chapter to allow efficient exploration of an unknown environment without hard-coding special situations.

Considered in this chapter is the joining of the information from the three separate algorithms discussed in the previous chapter: an area-view improvement, an obstacle-view improvement, and a position-similarity measure. With these three measures combined, their respective disadvantages should be counter-acted and their advantages together should lead to an efficient exploration. All three of the measures, however, shared a common disadvantage, that of treating newly discovered areas and obstacles in the same manner as older obstacles. This does not encourage a pioneering instinct. In this chapter, a measure of importance of obstacles which can encourage this pioneering aspect for a successful exploration algorithm is considered.

Three environments will be considered: the first is the basic scene example considered in the previous chapter and is shown here to demonstrate the workings of the algorithm in comparison to those methods discussed previously; the second scene is one engineered to provide one example of a difficult scene for an exploration process, where the majority of the scene area is known prior to exploration. The interesting point in this case is whether the exploration algorithm can find the unknown area without it being pre-planned that the entire area must be exhaustively searched, or knowing that the entire area is accessible. It is interesting to see how the algorithm deals with a large open space where no new parts of the scene are discovered for some time. The third environment is a map engineered to display another specific characteristic: a long corridor with a dead end. The exploration algorithm presented here does not hard-code these scenarios. How they are handled with the joint-improvement utility function with obstacle-importance will be discussed.

5.2 Joint-Improvement Utility Function

In the previous chapter, three methods that measure the improvement of a new position during exploration were presented. Two of these methods measured the angular view improvement of an obstacle chosen for view and the predicted area to be visible. The third was a measure of the similarity to the previously visited positions. It was noted that most of the disadvantages of each of the methods could have been counter-balanced by one of the other improvement measures. By joining all three measures together into one utility function, a simple calculation can measure the best choice of the next-best-viewpoint options.

The area-view (I_a) and obstacle-view (I_o) improvement measures increase for an improvement of

knowledge of the scene. The position-similarity measure (I_p) decreases for a preferred destination position. These three measures are normalised to values between zero and one, which allows them to be considered equally by an additive joint-improvement function. The original values are divided by their respective maximum values (chosen after a number of test runs) which in the case of the area-obstacle-view improvements depend upon the resolution of their respective bin sizes. The normalised values $\tilde{I}_o$, $\tilde{I}_a$, and $\tilde{I}_p$ are added together to provide a joint-improvement measure I_J given by

$$I_J = \max(\tilde{I}_o + \tilde{I}_a + (1 - \tilde{I}_p)) \tag{5.1}$$

From the choice of the next-best-view positions for each obstacle in the robot's vicinity, the obstacle which yields the highest joint-improvement, I_J, is selected. The move to the position which provides this improvement is passed through the system for execution.

5.2.1 Obstacle importance

One common problem discovered in the previous chapter was the lack of a pioneering aspect; when a new area is discovered, the new obstacles that encompass the new area were only as interesting as the obstacles that have already been seen before. If the view improvements are the same, no preference is shown to the newer obstacles.

An obstacle should be deemed especially interesting to the exploration algorithm if it has not been seen many times before. To simply rate each obstacle for its interest to the selection algorithm an obstacle-importance O_i is used. O_i is initially set to a nominal value for all obstacles when first added into the world map, and increased or decreased according to its importance to the exploration algorithm. Before a move is executed, the portion of the scene that should be acquired from the next viewing position can be predicted. The actual information received from the new position can be compared to that which was predicted and used to feed back the success of the data acquisition to provide successive move calculations with useful information.

A measure of the information that should be acquired from a move is the area that is anticipated to be viewable. Figure 5.1 shows an example where the predicted area is not the same as the actual viewable area. When the scene has only been seen from P and the features B and C have not yet been detected, the obstacle AD will exist. Later, when the scene is viewed from P' (for example when obstacle AD is selected for view) scene features B and C become visible and will be added to the world map. The expected area from P' would have been that up to obstacle AD, whereas the actual area, once features B and C have been added, is the quadrilateral $ABCD$ over and above that which was expected. In this example, the new obstacles that surround the new available area (AB, BC, and CD) become very important to the pioneering requirement of the exploration algorithm and should be prioritised.

A plausible heuristic is that the more times an obstacle is viewed, the less likely that that obstacle is to occlude features yet to be discovered. An obstacle that has been viewed five times is less likely to

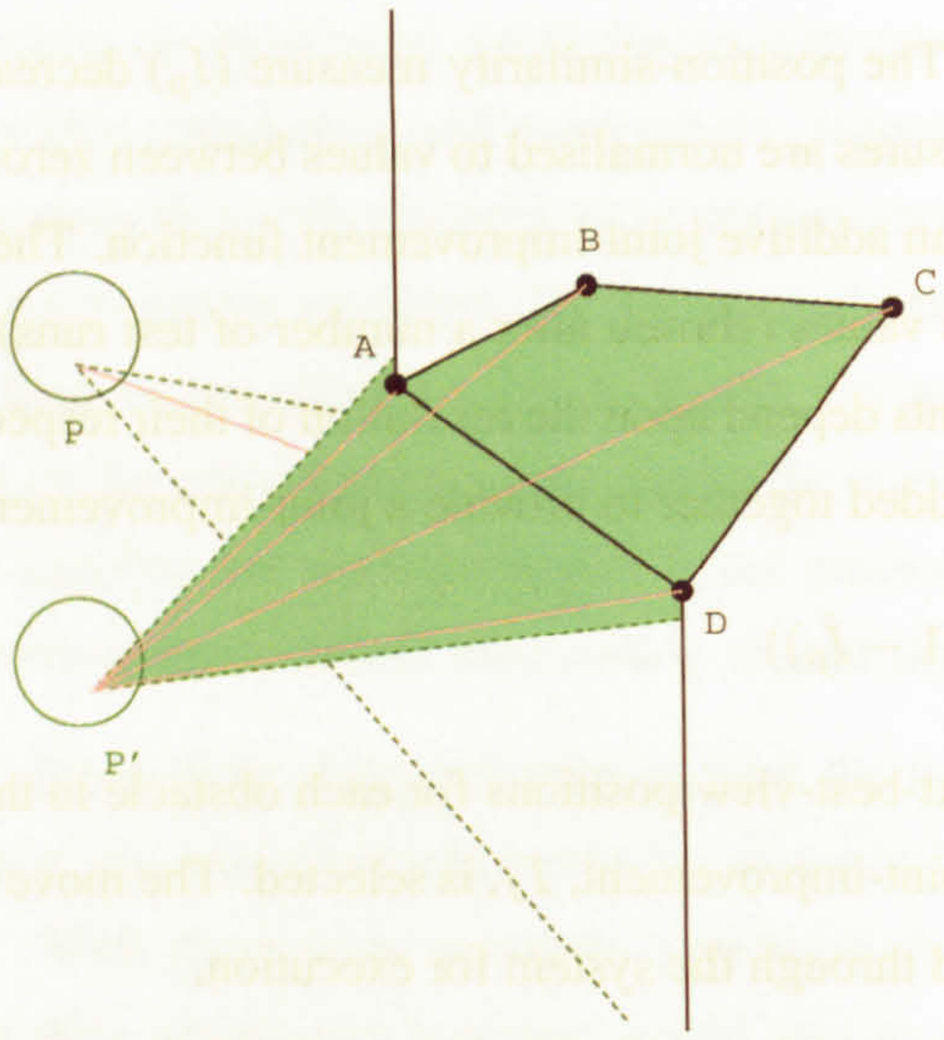


Figure 5.1: Obstacle Importance: predicted versus actual area. When only viewed from P, obstacle AD exists (since B and C are yet to be detected). The predicted area from P' is therefore $P'AD$. Instead features B and C are discovered, so the actual viewed area is $P'ABCD$.

be incorrectly labelled than one that has only been viewed once (assuming that the view angles are not identical). With this in mind, as obstacles are selected for view by the exploration algorithm they become less interesting as they are less likely to reveal unseen occluded features. The value of O_i is reduced by a factor of 10 if more than 70% of the obstacle is seen from the new position. O_i is multiplied by a factor equivalent to the difference in the area which was expected to be viewed and the area that was actually viewed, when new obstacles are discovered (in the example of Figure 5.1, this would be area $ABCD$).

Obstacle histograms are re-assessed at each evaluation step to keep the portions updated with new information that may change future motion decisions. In the same vein, the area histogram is also re-assessed at each evaluation step. This is important since it is possible for new features to be detected which contradict the information that a portion of the area has been seen. In this case, areas that were previously labelled as seen will be changed, since they have not actually been viewed. This updated histogram information is then used to specify which obstacles are more interesting for future exploration.

As the scene features of already known world features are updated, their positions are likely to change. In such cases, the area expected to be viewed will alter, not because the obstacle is new but because it has been updated. The greater the change in position of either obstacle end point, the greater will be the difference between the area expected to be updated and the area that is actually updated. This will therefore also be taken into account during obstacle selection.

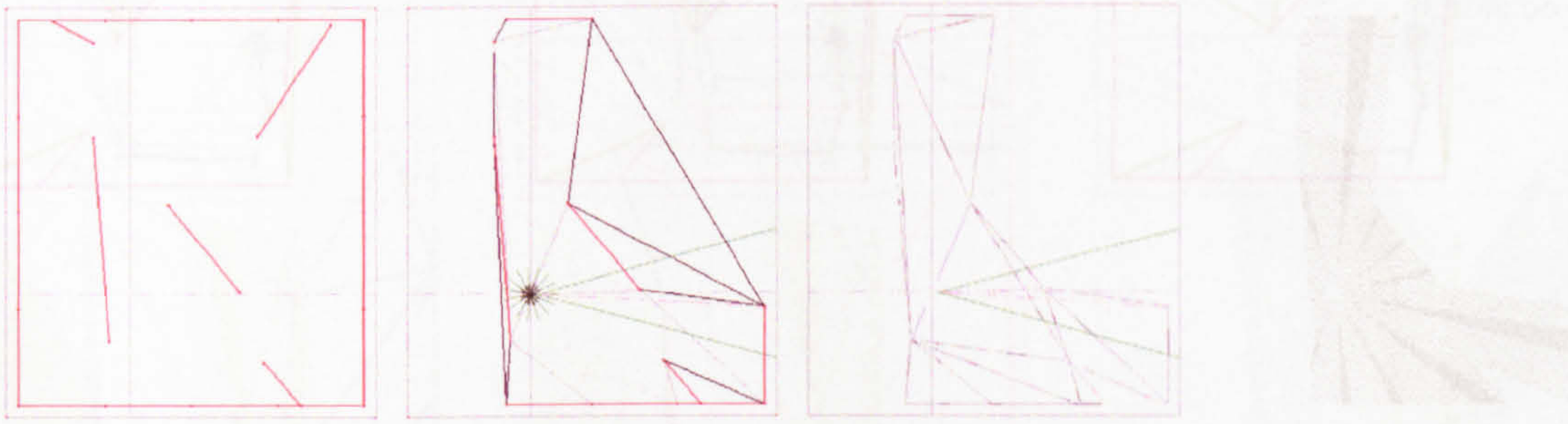
The rear-side of previously known obstacles are labelled with separate obstacle-importance. Although obstacles that have already been viewed from one side are unlikely to be re-labelled as non-

obstacles when viewed from the other side, the importance of an obstacle relates to the number of times it has been viewed and the circumstances under which it was discovered and updated. To say that since an obstacle had been viewed from one side makes it less interesting when viewed from the other side would be incorrect. For example, if both sides of an obstacle shared their importance value, an incorrectly labelled obstacle that had only been seen many times from one side (but only labelled as an obstacle because no other features were visible through it) may not be considered for subsequent view selection from its rear side, even though this may lead to the discovery that this obstacle is actually an opening.

Let us first consider the exploration of the basic scene from the previous chapter using the joint-improvement utility function with obstacle-importance to select the next obstacle to view.

5.3 Basic Scene

Figure 5.2(a) shows the scene map of the basic scene used in the previous chapter, (Figure 4.8(a)). This scene is constructed of sixteen external obstacles of similar length and five internal obstacles of random position, length, and orientation. The system begins with no knowledge of the number, size, position, or orientation of the obstacles in the scene. The exploration begins with fifteen rotations to cover 360°. Figure 5.2(b) shows the obstacle map after these rotations. The single improvement methods in Chapter 4 each started in this position with this same knowledge of the scene. All obstacles at this stage have an obstacle-importance of 100, and therefore will be selected for view based only on their joint-improvement.



(a) Basic scene (b) Initial move xvi (c) Obstacle histograms (d) Area histogram

Figure 5.2: Basic scene exploration: last initial move. Correctly detected obstacles highlighted in red, current and previous view directions shown in green.

5.3.1 Exploration

Figure 5.3 shows the first few moves selected by the exploration algorithm. Each column shows one snapshot of the exploration. It can be seen in move 4 that the obstacle selected by the exploration algorithm (shown on the line histogram map by circles at each of the hypothesised obstacle's feature

points) was occluding an unseen scene feature, C. A new hypothesised obstacle is created (L1) with this new feature. Due to its high obstacle-importance, its joint-improvement value will be considerably higher than for any other obstacle. The best joint-improvement from the position options for that obstacle is selected for view in move 5 where three new scene features are discovered. Again a newly constructed hypothesised obstacle (L2) is selected for view in move 6.

These moves are an exciting example of how the obstacle-importance measure is used alongside the improvement measures to rapidly and repeatedly discover new areas of the scene.

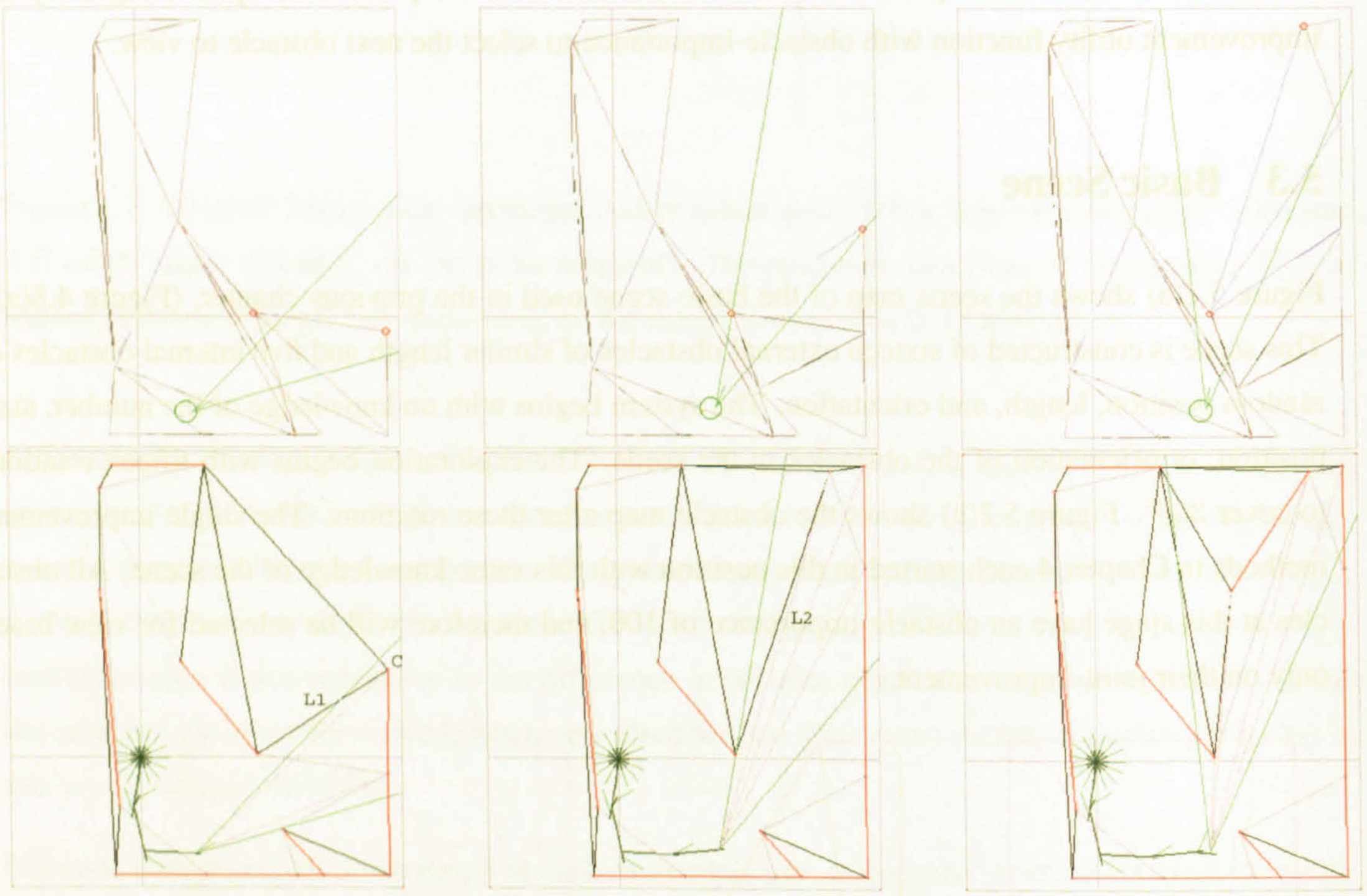


Figure 5.3: Basic scene exploration: moves 4–6. (row i) next move on obstacle histogram map, selected obstacle highlighted by corner circles, (ii) obstacle map after move, (iii) seen areas after move. As new scene features are discovered, new hypothesised obstacles are constructed and prioritised.

The exploration proceeds in Figure 5.4 as the first obstacle is navigated around revealing a huge area of unexplored territory. This section of the exploration is dominated by large obstacle-importance values which pulls the robot position around into the new area. At this stage the obstacle-view, area-view and position-similarity are all very strong. Combined with a large obstacle-importance, the exploration selects views of large areas and those which encompass the entire length of obstacles and at positions well away from previously visited positions.

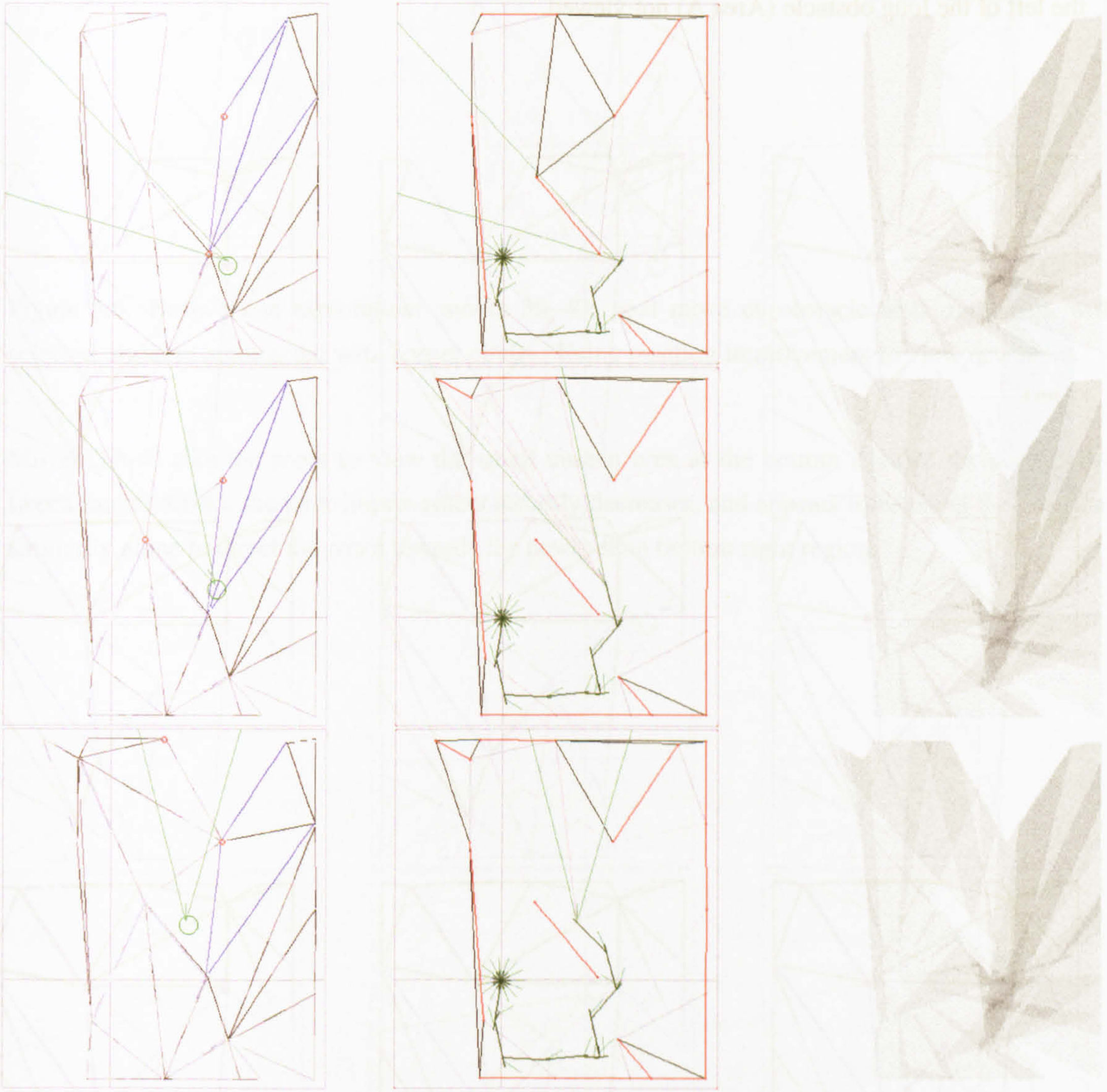


Figure 5.4: Basic scene exploration: moves 11–13: large obstacle-importance influences the choice of obstacles to view. Area-view and position-similarity also play a part in this choice; obstacle histograms with defined clockwise obstacles shown in grey-level and anti-clockwise obstacles shown in blue-level; if both sides of an obstacle are viewed, both grey and blue lines are drawn.

Figure 5.5 shows the chosen obstacle for view, for moves 24–32. The robot has reached the far side of the scene and has viewed over 75% of the area. The long obstacle which fences off the far left side of the scene has been navigated around and a portion of the area behind the obstacle has been viewed and entered into. This limits the choice for the next move to those which provide motion direction within only 20°–100°, with a larger move distance, and to the very local obstacles. Unless the far end of the long obstacle is viewed, the incorrect hypothesis of the short obstacle selected for view within moves 24 and 26 will remain. This view is not chosen (even if available from the limited move options) and the surrounding local obstacles are considered instead, leaving the unseen area to the left of the long obstacle (Area A) not viewed.

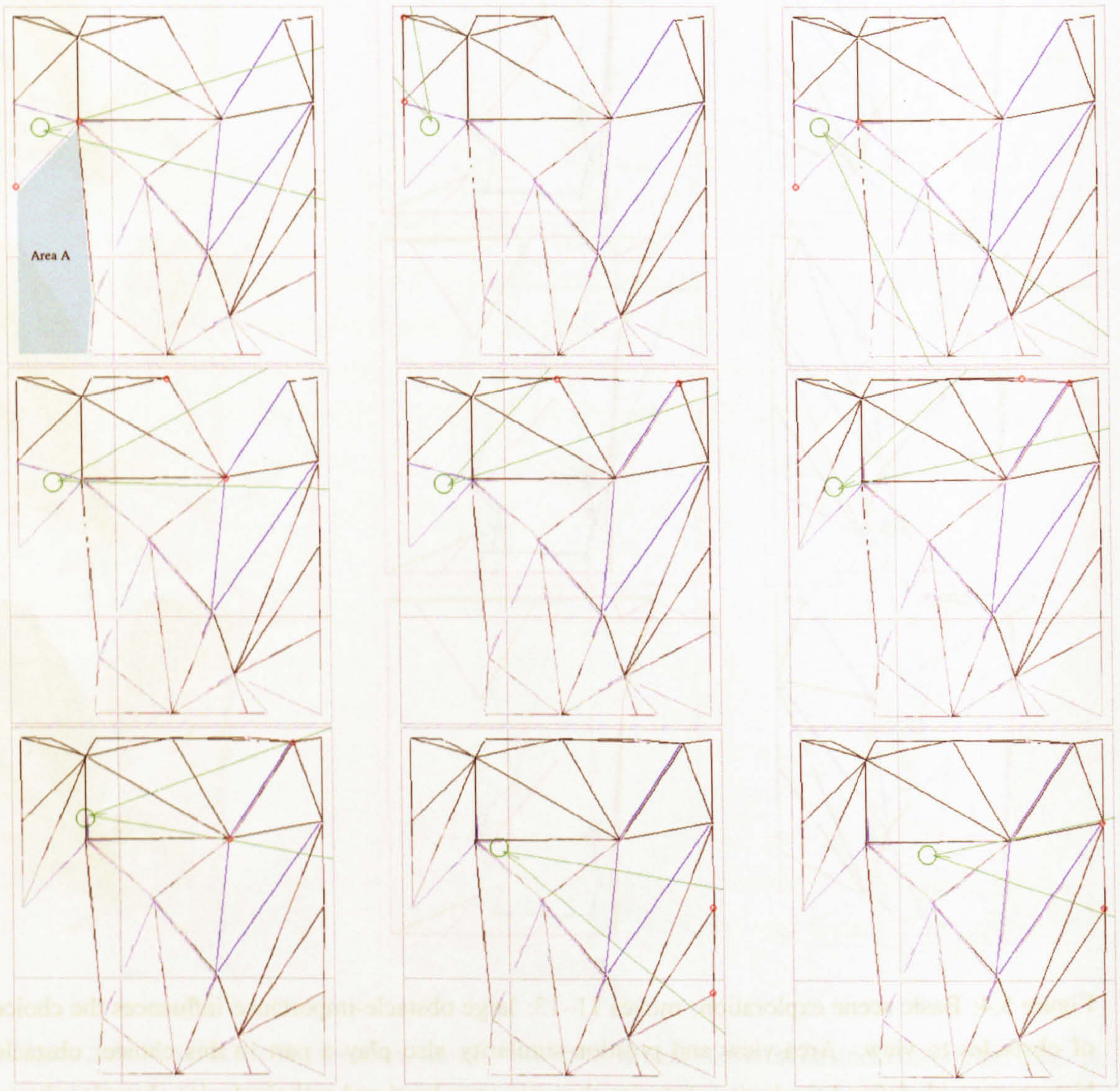


Figure 5.5: Basic scene exploration: moves 24–32: next move on obstacle histogram map. Limited choice of next positions. Area A remains unseen.

The exploration algorithm does not use information of the portions of a scene which have not been seen, as no assumptions to the size or shape of the environment are made, or that all areas of a scene are accessible. This has the advantage that a portion of the scene is not repeatedly and exhaustively viewed with no view available, but has the disadvantage that a portion of the scene may be missed, as in this case. (This portion of the scene is discovered later in the exploration.)

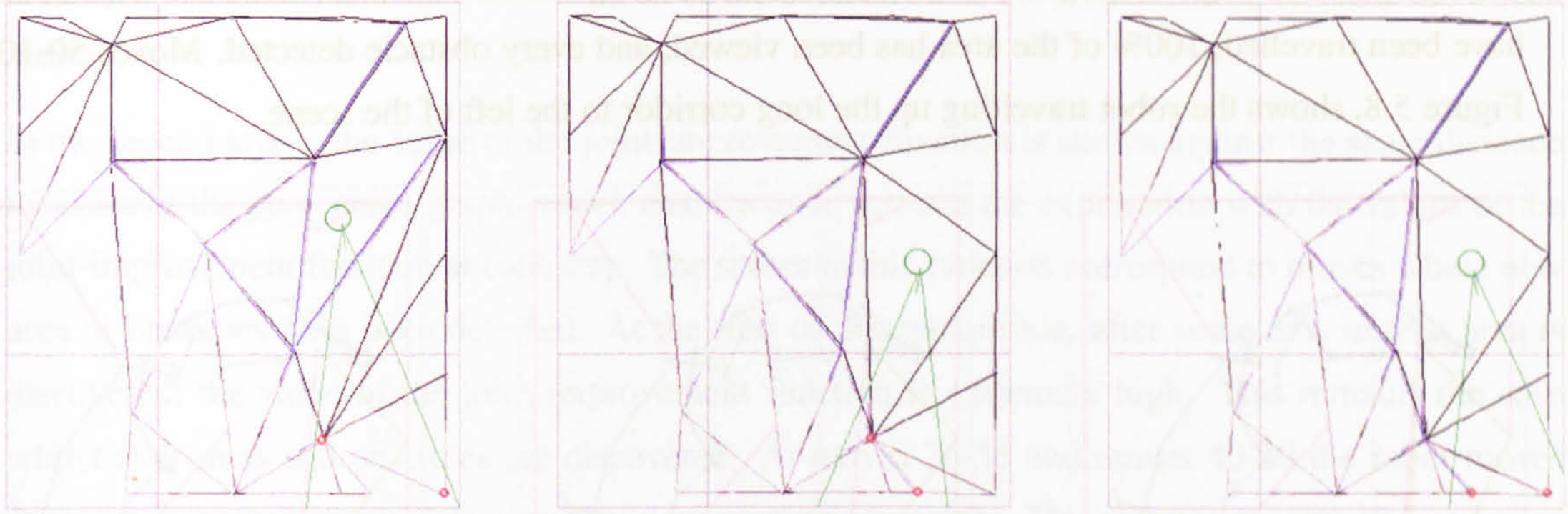


Figure 5.6: Basic scene exploration: moves 38–40: next move on obstacle histogram map, with selected obstacle highlighted with corner circles. Using position improvement to view new areas

Moves 38–40 take the robot to view the small unseen area at the bottom right of the scene. Between moves 40–45, the joint-improvement steadily decreases, and appears to be using the position-similarity alone to direct the robot towards the newly seen bottom right region.

Figure 5.7 shows the last new areas being discovered between moves 45–50. The position-similarity measure has pushed the position towards the bottom of the scene. At move 46 the only visible remaining incorrectly hypothesised obstacle is viewed and reveals the unseen area in the bottom left corner of the scene. The joint-improvement values for subsequent moves are controlled by the huge obstacle-importance values of the new obstacles, keeping the pioneering interest of the exploration towards this new region. Moves 47–50 reveal the remaining unseen areas. After 50 moves, 7500cm have been travelled, 100% of the area has been viewed, and every obstacle detected. Moves 50-100, Figure 5.8, shows the robot travelling up the long corridor to the left of the scene.

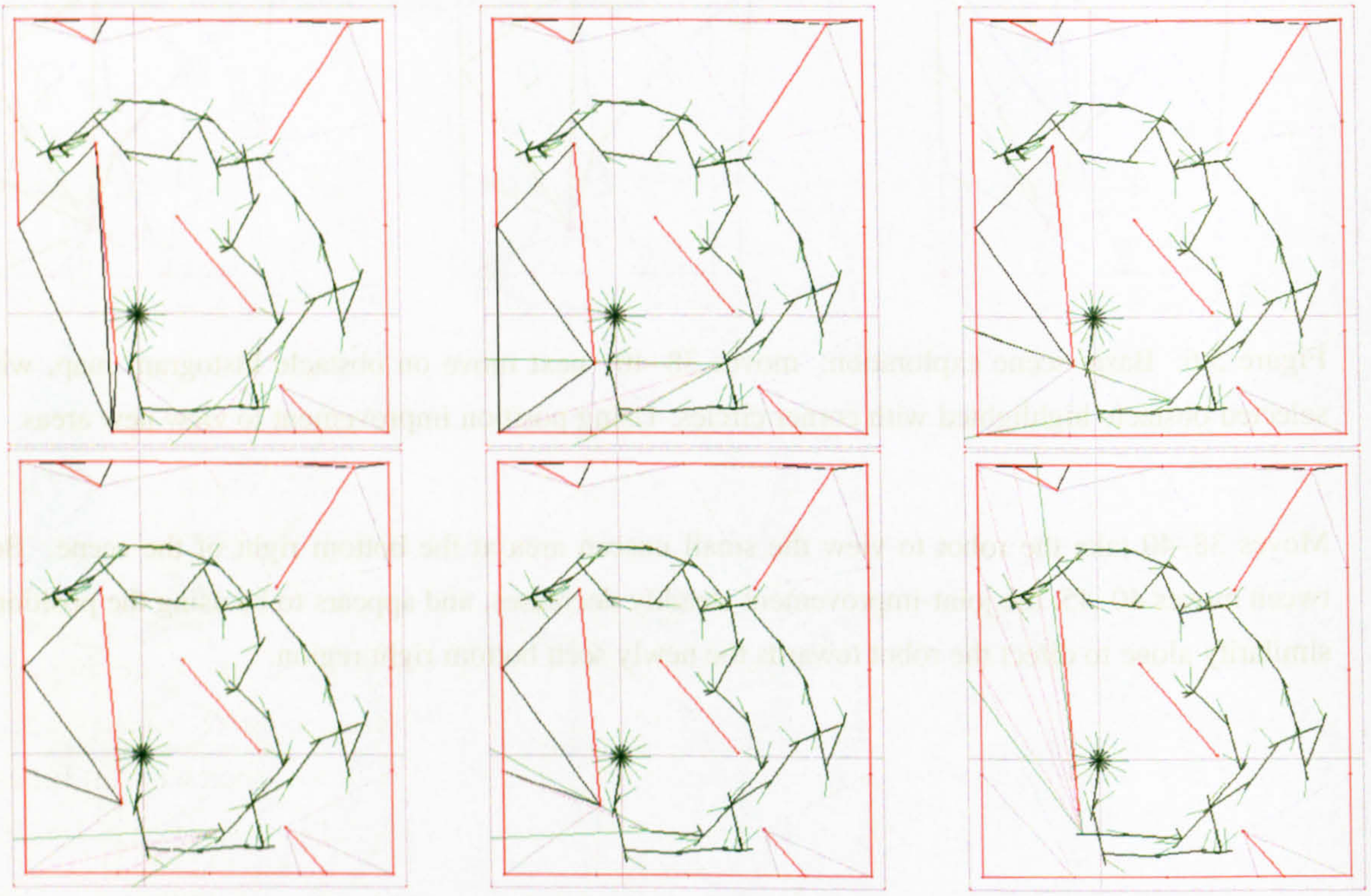


Figure 5.7: Basic scene exploration moves 45–50: remaining unseen areas discovered.

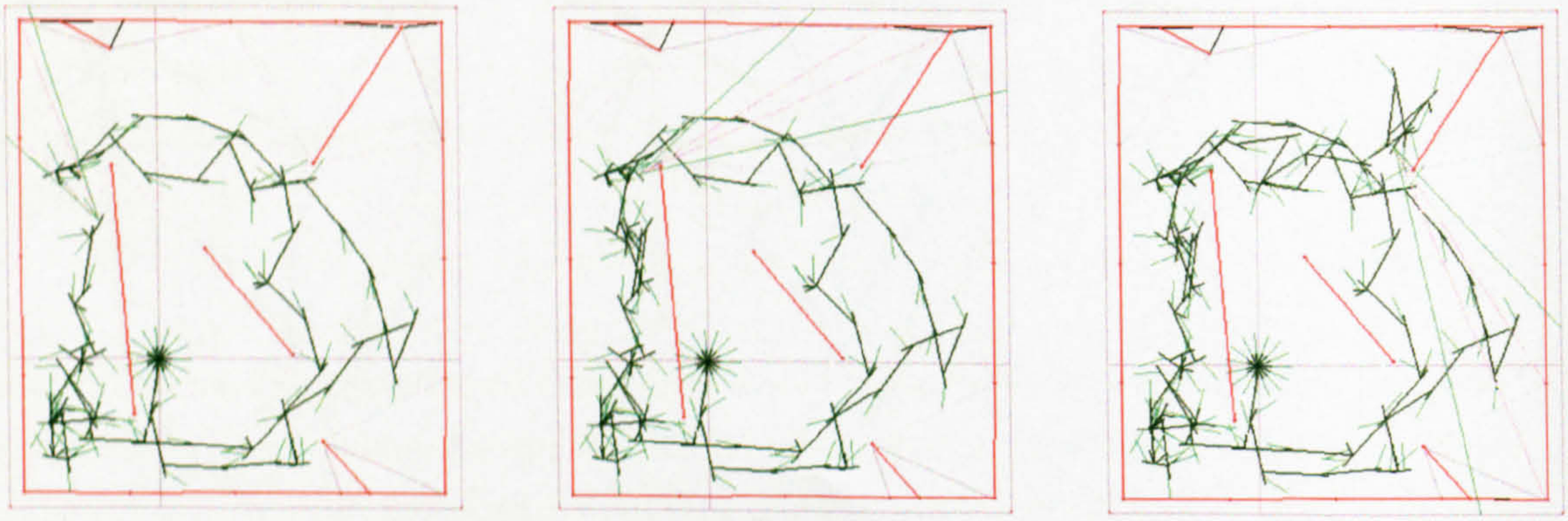


Figure 5.8: Basic scene exploration: moves 88, 100, 120.

5.3.2 Analysis

Figure 5.9 shows the percentage of the area and the number of obstacles seen as the robot travels through the scene. The percentage of obstacles detected and the percentage of the area seen follow closely as the exploration continues. After just twenty moves over 75% of both the obstacles and the area have been seen, and after 50 moves all of the obstacles and the area of the basic scene have been viewed.

In the second graph, the value of the joint-improvement function is shown against the same distance measure as the percentage graph, which enables us to re-trace the exploration with the values of the joint-improvement function at each step. The spikes in this function correspond to moves where new area or obstacles have been detected. At the start of the exploration, after some new unseen area is discovered, the value of the joint improvement function is extremely high. This remains the case whilst new areas and obstacles are discovered. At moves 26-36 and moves 40-50 the robot moves across the scene without detecting any new area or obstacles. The maximum joint-improvement values during these moves gradually decrease.

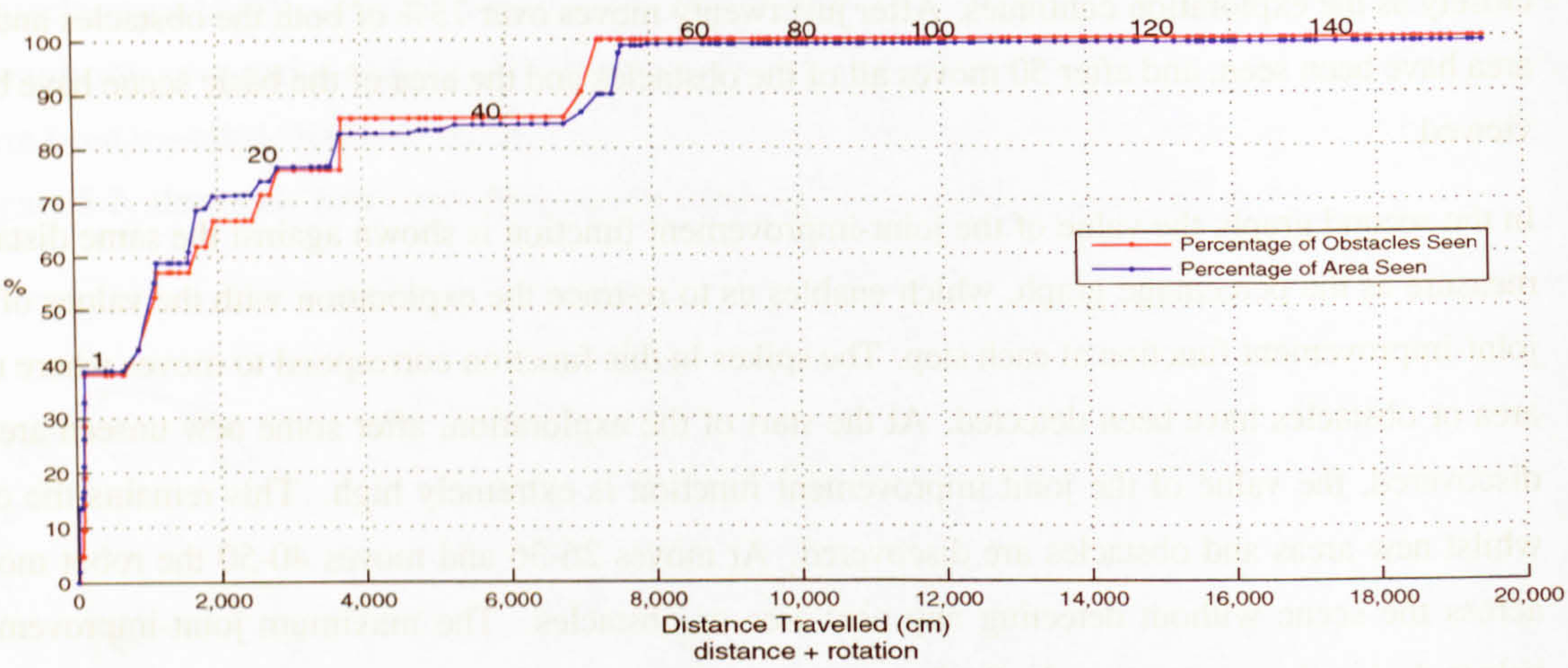
Move 50 is the last move where new area is discovered. After this point, the joint improvement value decreases to zero as the robot's exploration is allowed to continue. As each obstacle with a high obstacle-importance value is chosen for view its obstacle importance decreases, the chosen value for the maximum joint-improvement will therefore decrease over time.

From move 50-100 the robot travels up the long corridor to the left of the scene. Once the robot reaches the top of the corridor an obstacle previously not well seen becomes visible. Again at move 120 a large spike appears in the joint improvement value, due to an obstacle-importance left behind from the earlier exploration moves. The joint-improvement however is only in the order of 1×10^1 compared to the previous values at 1×10^{12}. After 120 moves all three improvement measures decrease. Since no further new areas are available to explore, the obstacle-importance plays no part in the obstacle selection choice and the joint-improvement falls to zero.

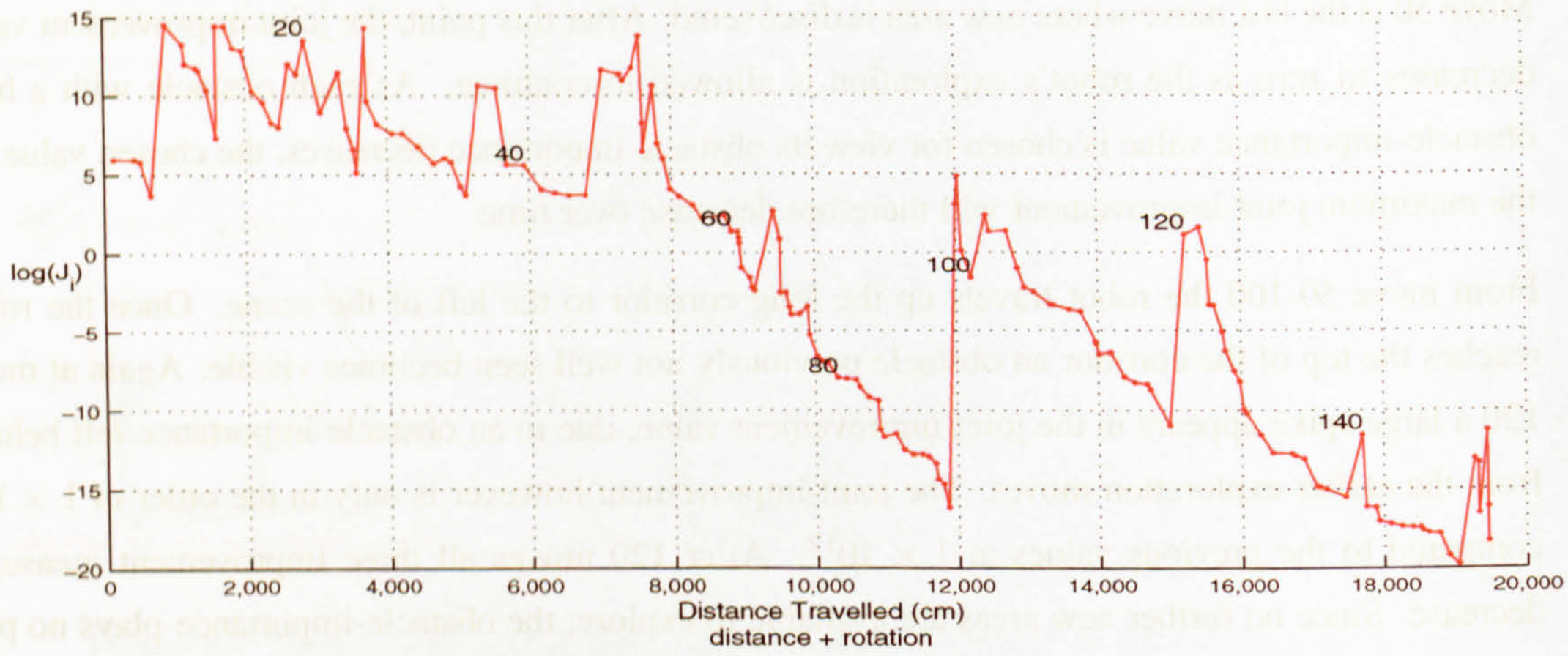
The exploration of this basic scene using the obstacle-view improvement was shown in the previous chapter. During that exploration, the chosen robot positions were those which were close to the obstacle and views along the obstacle so that the entire obstacle was visible. After thirty moves and 4000cm travelled in the obstacle-view exploration, over 70% of the obstacles and area had been seen, compared to over 80% for the joint improvement exploration after travelling a similar distance. The path of the robot at the beginning of the obstacle-view exploration is similar to that of the joint exploration in that the robot is moved away from the starting point and acquires information of the lengths of obstacles it is able to view. However the path of the joint-improvement exploration is much more direct towards the new discovered area and does not back-track upon itself as often.

The area-view improvement exploration of the previous chapter showed that the chosen robot positions were such that a larger area would be visible, in the centre of the spaces of the scene. This behaviour is also displayed in this exploration. After 50 moves and 4300cm travelled in the area-

view exploration, 75% of the obstacles and 80% of the scene area had been viewed; after 50 moves and 7500cm travelled in the joint-improvement exploration, 100% of the obstacles and the area had been viewed. Before the area-view exploration had viewed all of the area and obstacles, a total of 10,000cm had been travelled.



(a) Percentage of obstacles and areas seen versus total distance travelled.



(b) Joint-improvement during exploration.

Figure 5.9: Analysis of basic map exploration using joint improvement measure.

The path of the robot for the area-view exploration was noted to be characteristic of one large step followed by several rotations, although this was not apparent in the joint improvement exploration.

The position-similarity exploration of the previous chapter showed that the chosen robot path was irregular and generally didn't intersect itself, which is a characteristic also displayed in the joint-improvement exploration. After 50 moves and 3300cm travelled, 90% of the obstacles and 94% of the area had been viewed using the position-similarity exploration. This compares to 100% of both obstacles and the area which had been viewed after travelling that distance in the joint-improvement exploration.

In this basic scene example, the joint-improvement exploration performs better than each of the three techniques alone. Along with the obstacle improvement measure a sensible exploration path is constructed. Two structured environments to display certain characteristics of the joint-improvement exploration algorithm will now be considered.

5.4 Hidden Room

The previous scene map showed the general workings of the exploration algorithm and how a map can be successfully explored using the joint-improvement utility function with next-best-view movement options. A scene where the majority of the area and obstacles can be seen from the starting point, but contains a room whose entrance is hidden from view, Figure 5.10 will now be considered. For the robot to find the entrance to the room, it must travel up to the top of the scene and look towards the entrance at an angle such that the interior of the room becomes visible. The obstacle map after the initial move is shown, also the area-view histogram.

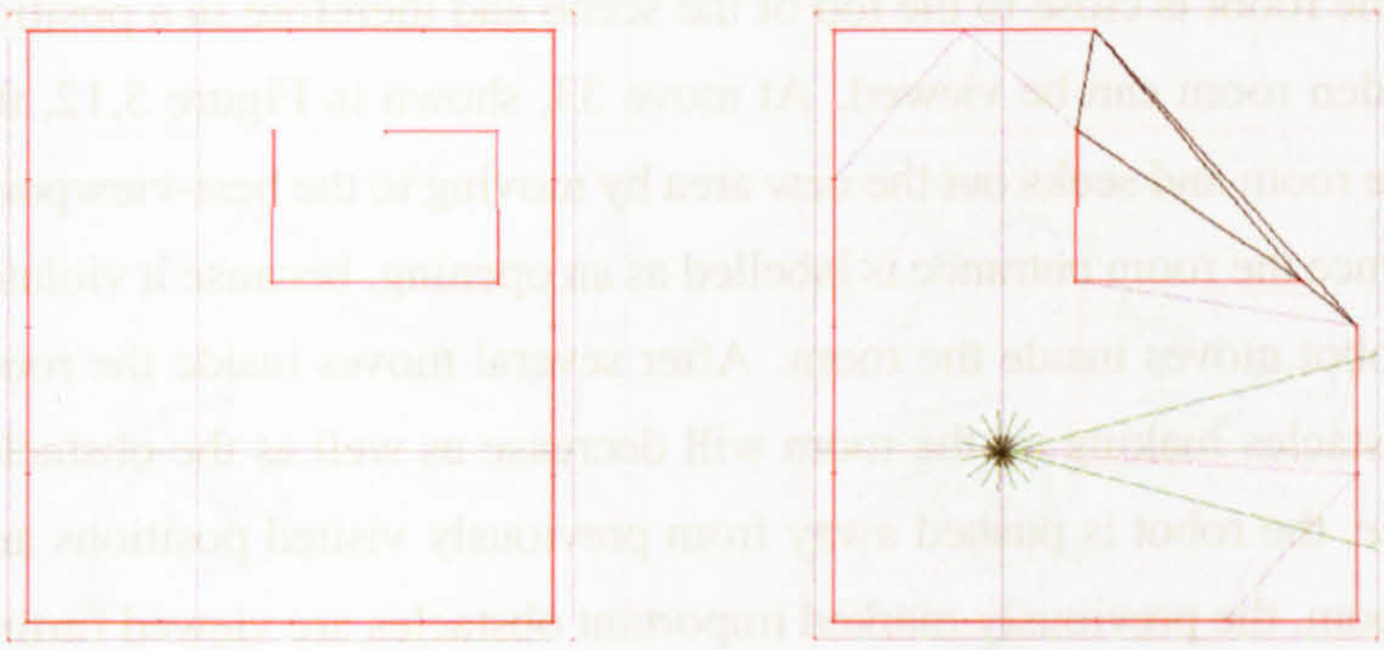


Figure 5.10: Hidden Room Exploration: Scene Map, Initial Move and Areas Seen. Most of the obstacles and the area is seen from the initial scan; entrance is hidden from initial view.

After this initial move, all of the construction lines labelled as obstacles have the same obstacle-importance, so are considered equally. The choice of move therefore is due only to improvement in obstacle- and area-view and position-similarity.

5.4.1 Exploration

Figure 5.11 shows the beginning of the exploration with move numbers 4, 6, 8 and 10. While no new obstacles are detected each obstacle has the same importance (O_i). The initial moves show the robot moving without an overall direction around the central area. It appears that each obstacle around the exterior of the known space are selected in turn, where a good sized area and large change in position is calculated. No new area or new obstacle features are discovered for some time. The joint improvement slowly decreases as each obstacle is viewed numerous times. A similar behaviour to

that of the position-similarity exploration shown in the previous chapter (Figure 4.15) is prominent during these first few moves.

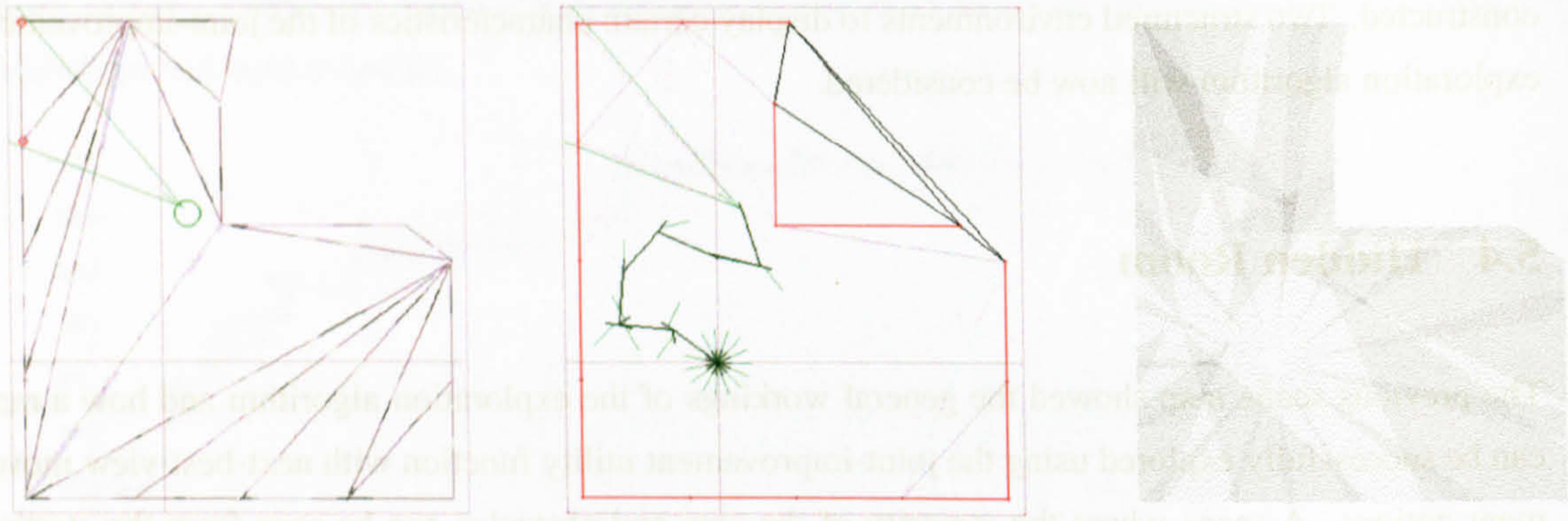


Figure 5.11: Hidden room exploration: up to move 10.

After 32 similar moves where the entrance to the hidden room is not seen and thus no new information of the scene has been discovered, the robot is close to the top of the scene and therefore in a position that the unseen area above the hidden room can be viewed. At move 33, shown in Figure 5.12, the robot detects the new area above the room and seeks out the new area by moving to the best-viewpoint for the newly detected obstacles. Once the room entrance is labelled as an opening, because it violates several visibility constraints, the robot moves inside the room. After several moves inside the room the obstacle-importance for the obstacles making up the room will decrease as well as the obstacle- and area-view improvement. Hence, the robot is pushed away from previously visited positions and exits the room. Upon leaving the room, the previously marked important obstacles are viewed further and the remaining unseen area of the scene is viewed.

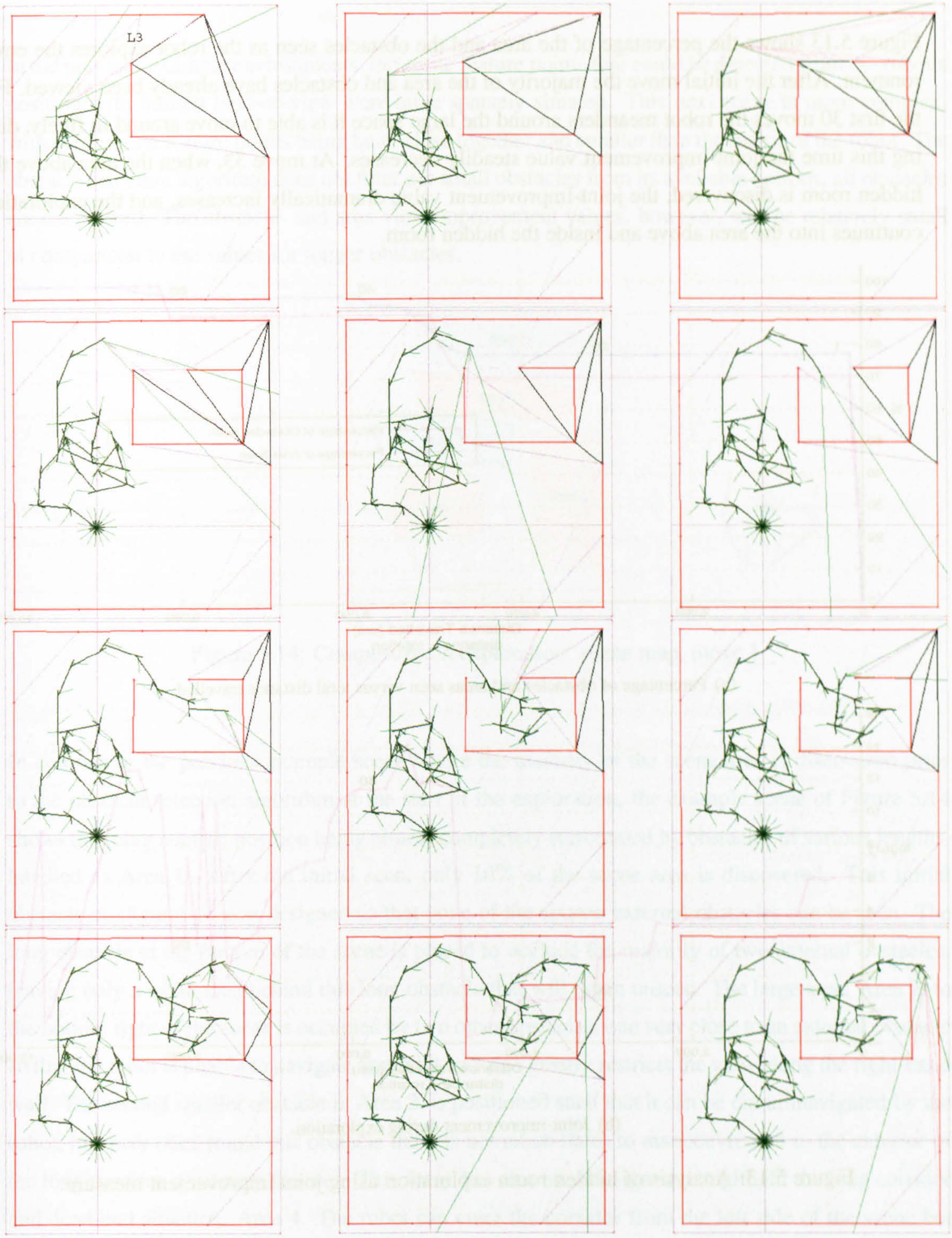
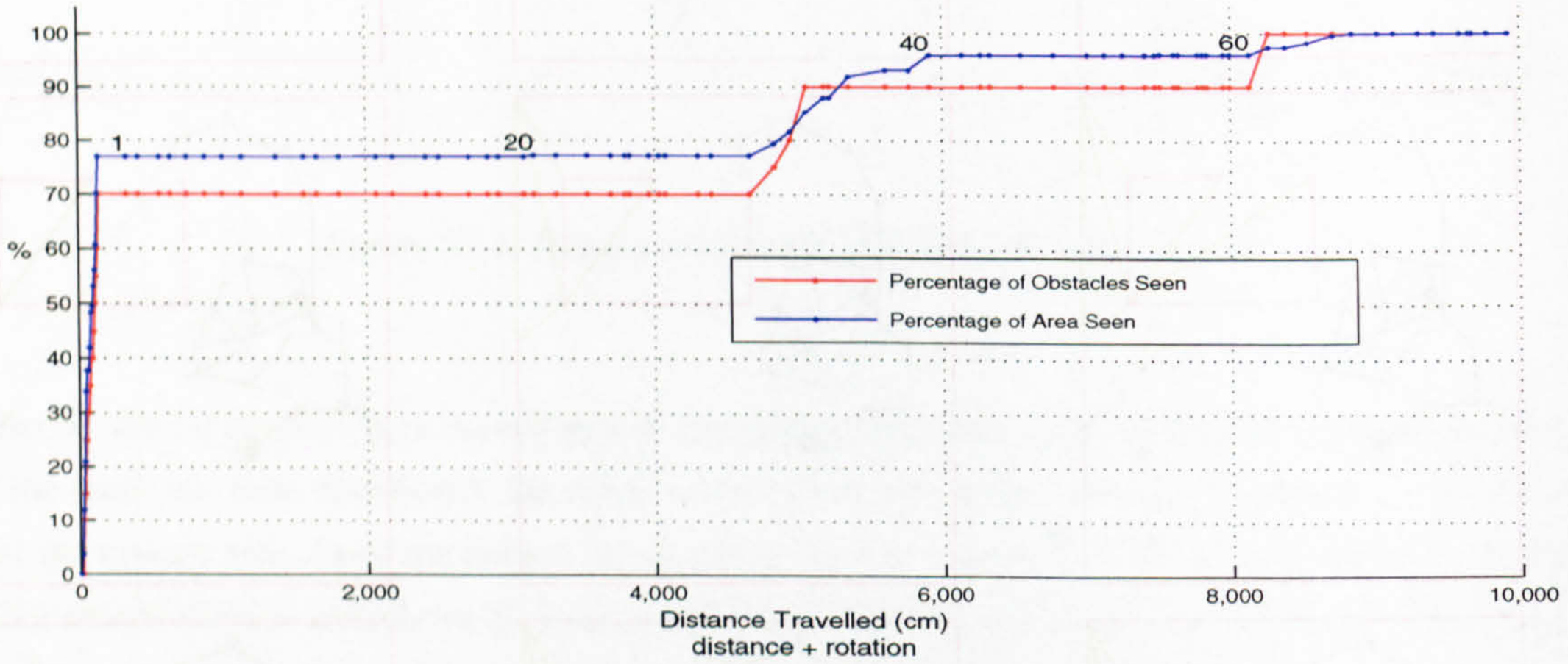


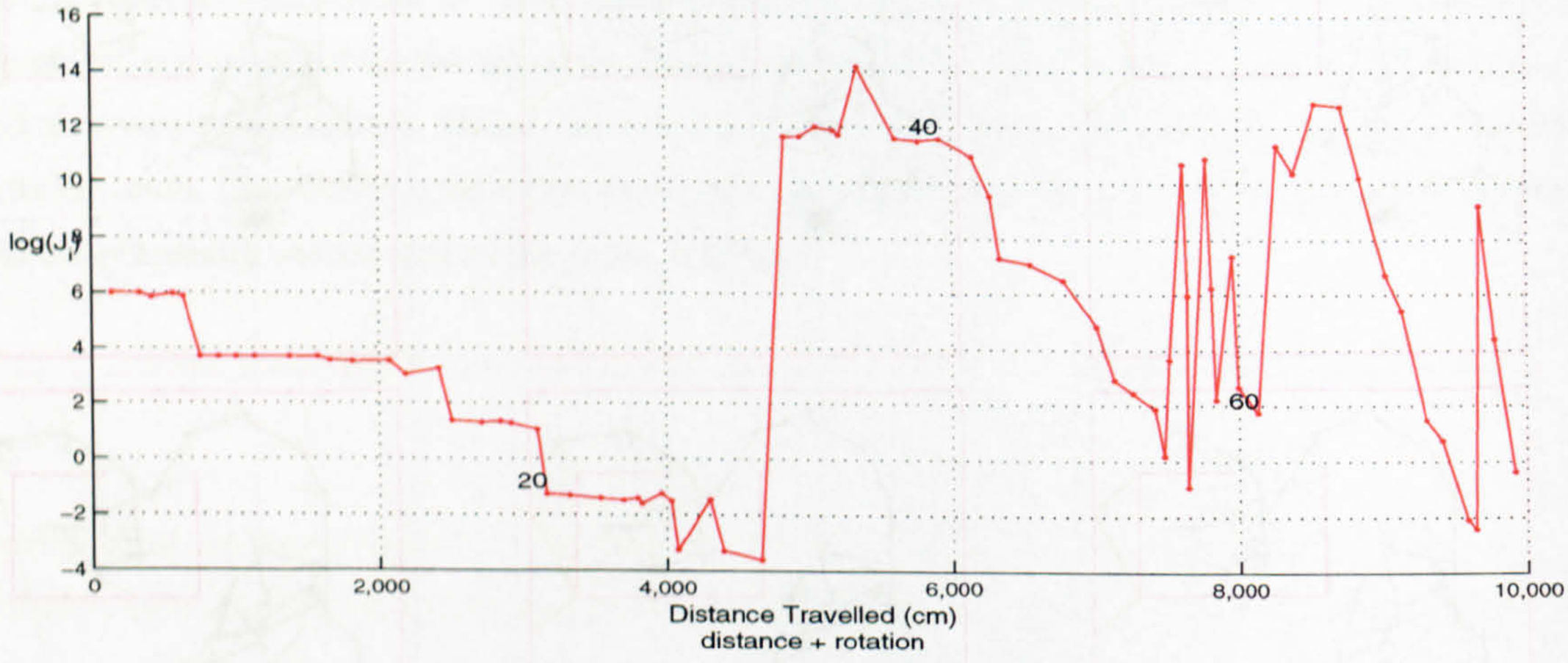
Figure 5.12: Hidden room exploration: obstacle map with robot path for selected moves 33–66. Move 33 selects the obstacle L3 to view, allowing new features to be detected. Subsequent hypothesised obstacles selected yield further information about the scene. After just 66 moves the entire area has been viewed and all obstacles discovered.

5.4.2 Analysis

Figure 5.13 shows the percentage of the area and the obstacles seen as the robot explores the environment. After the initial move the majority of the area and obstacles have already been viewed. For the first 30 moves the robot meanders around the large space it is able to move around in freely, during this time the joint-improvement value steadily decreases. At move 33, when the area above the hidden room is discovered, the joint-improvement value dramatically increases, and the exploration continues into the area above and inside the hidden room.



(a) Percentage of obstacles and areas seen versus total distance travelled.



(b) Joint-improvement during exploration.

Figure 5.13: Analysis of hidden room exploration using joint improvement measure.

5.5 Complex Scene with Dead-End

In the previous example environments, the scene feature points that could be detected from the robot's position with limited field-of-view were quite sparsely situated. This next scene is more complex, with some scene feature points being quite close together and smaller than the radius of the robot. The obstacle selection algorithm does not filter out small obstacles from its available options, all obstacles are considered. The obstacle- and area-view improvement values, however, will be relatively small in comparison to the values for longer obstacles.

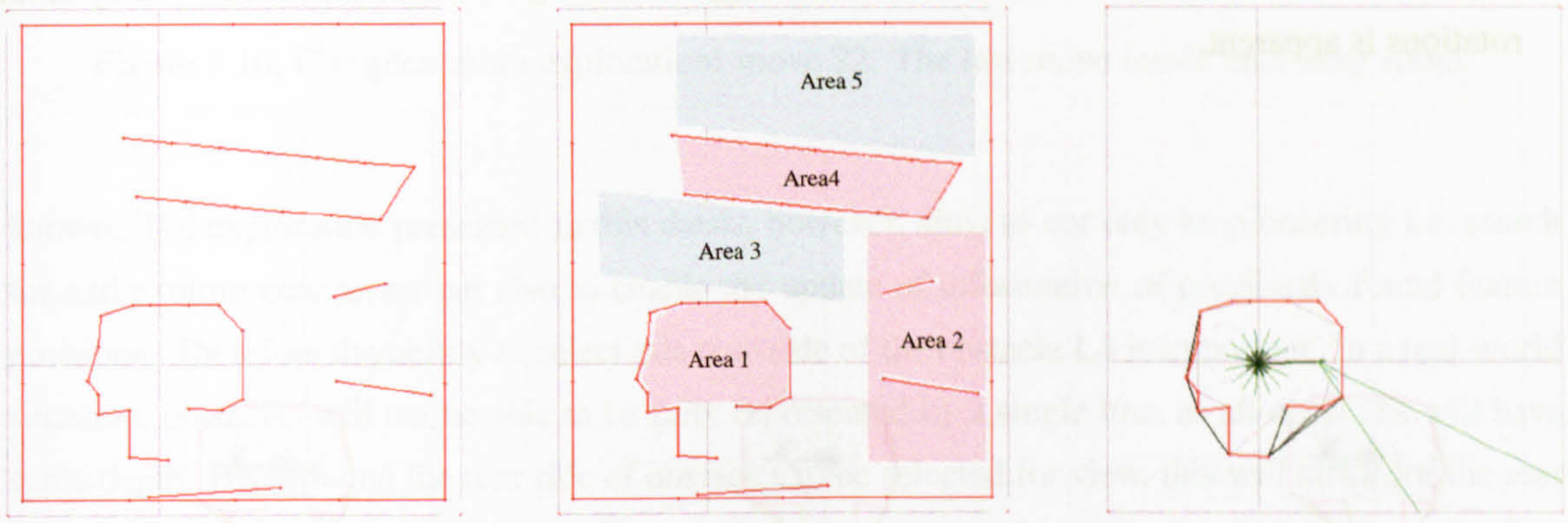


Figure 5.14: Complex room exploration: scene map, move 1.

In contrast to the previous example scene where the majority of the scene area is discovered prior to the obstacle selection algorithm at the start of the exploration, the example scene of Figure 5.14 shows the robot starting position being almost completely surrounded by obstacles of various lengths, labelled as Area 1. After the initial scan, only 10% of the scene area is discovered. This initial obstacle configuration was designed so that none of the sixteen external obstacles can be seen. The long obstacle at the bottom of the scene is placed to occlude the majority of two external obstacles, leaving only a small area behind this long obstacle that will begin unseen. The large area, Area 2, to the bottom right of the scene is occupied by two other obstacles, one very close to an external obstacle so that the robot is unable to navigate around it and also greatly restricts the view along the right-hand wall. The second smaller obstacle in Area 2, is positioned such that it can be circumnavigated by the robot, however once round this obstacle there is not much room to manoeuvre due to the exterior of the long corridor restricting the area. This scene has also been designed to highlight the long corridor and dead-end situation, Area 4. The robot can enter the corridor from the left side of the scene but must re-cover the area which has already been visited to come back out again.

5.5.1 Exploration

The first move of the exploration is shown in Figure 5.14, where the robot moves away from the starting position. The exploration continues in Figure 5.15, where after 13 moves the robot is still

inside Area 1 and has not discovered that one of the obstacles is incorrectly labelled. The behaviour of the robot in this case is similar to that of the hidden room example of Figure 5.11 where no new areas are discovered and the area- and obstacle-view improvement calculations become less prominent than the position-similarity; the obstacles appear to be selected in-turn from the longer obstacles to the shorter. Here, longer obstacles are selected before shorter obstacles due to the area- and obstacle-view improvement values generally being larger because of a greater number of histogram bins for long obstacles. In confined spaces, the obstacles being selected do not tend to be those that are viewed after moving large distances, in comparison to the previous exploration where several successive large distance moves were taken. Instead the strategy of one large distance move followed by several rotations is apparent.

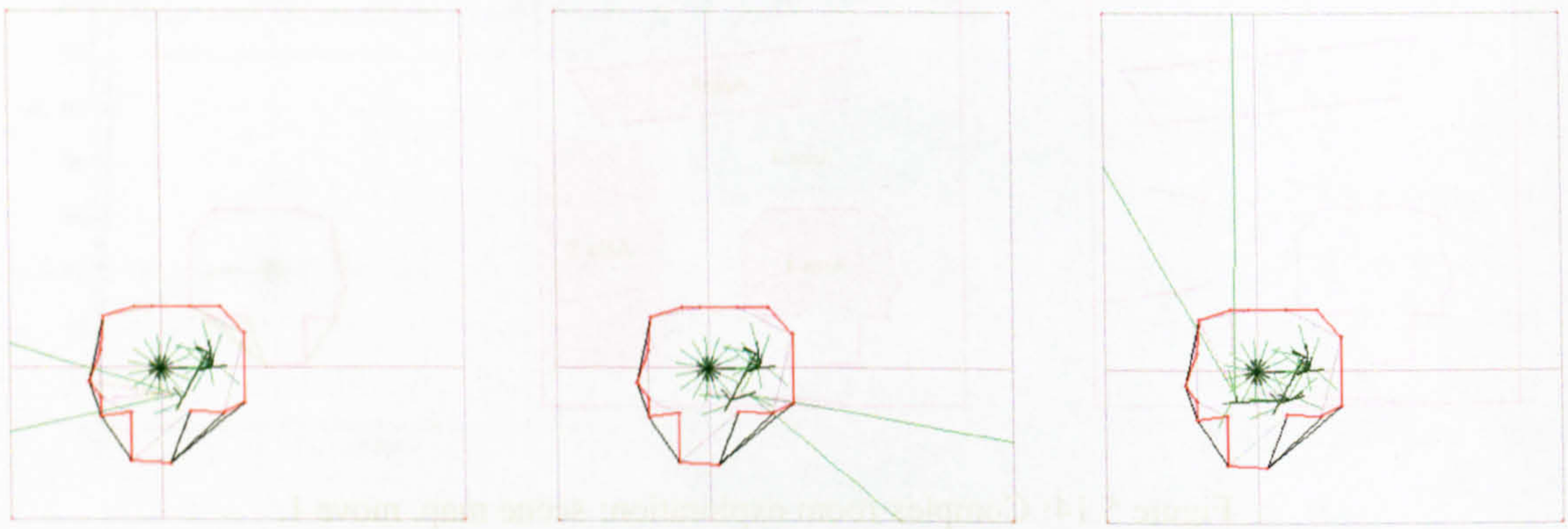


Figure 5.15: Complex room exploration: moves 13, 14 & 18. The obstacles appear to be selected in turn, with one large move followed by several smaller moves and rotations.

Figure 5.16 shows the last obstacle selection move which does not enable the robot to view any new area. Up to this point the selection algorithm has not chosen the incorrectly labelled obstacle to view since the viewing angle from the initial move scan is not significantly different from the available move options, most likely to be due to the confined space that the robot would have to move into in order to view this obstacle from the correct angle and the limited valid path required to access the next position.

The robot leaves the enclosed area upon discovering obstacles behind the previously incorrectly labelled obstacle, Figure 5.17. It should be noted that each of the newly discovered obstacles are now labelled with the same value of obstacle-importance, the unanticipated new area, Area A. This large obstacle-importance value will affect each of the new obstacles such that they are more likely to be chosen by the obstacle-selection algorithm than the obstacles that make up the interior of the initial enclosing obstacles.

In move 25 the rear side of an Area 1 obstacle is selected and viewed from outside the initially enclosed area. For a purely pioneering exploration algorithm, where the only goal is to discover uncharted areas, this obstacle, L4, should be removed from the interesting obstacle list, since no new unseen area will be exposed by looking through it at any angle since the area behind it is already

Figure 5.16: Complex room exploration: move 22. The last move inside enclosing room.

known. The exploration presented in this thesis, however, aims to not only be pioneering i.e. search for and explore new areas, but also to enable the update of information of previously found feature positions. Therefore the ability to select this rear-side of the obstacle L4 is important. In a real-world situation, obstacles will not be able to be fully represented by a single line, as all obstacles will have some depth. By allowing the rear side of obstacles to be selected for view, this will allow for the rear side of the 3D obstacle to be viewed and its other points to be added to the world map accordingly.

Moves 26-29 show, again, that obstacles appear to be selected in turn. At move 29 a new area to the bottom left of the scene (shaded region Area B) is discovered. Since the unanticipated area discovered (Area B) is relatively small compared to that from move 23 (Area A), the newly added obstacles are not immediately considered for further exploration. This holds some advantage over other methods that could be used to prioritise obstacles.

For example, one might consider ranking obstacles for selection in a list. As new obstacles are discovered they are pushed to the very top of the list, such that older obstacles are considered less important. However this does not account for the case of a minor adjustment being made to the world map. In this case a relatively unimportant new obstacle would be ranked higher than a much more interesting one. The strategy employed here does not encounter this problem.

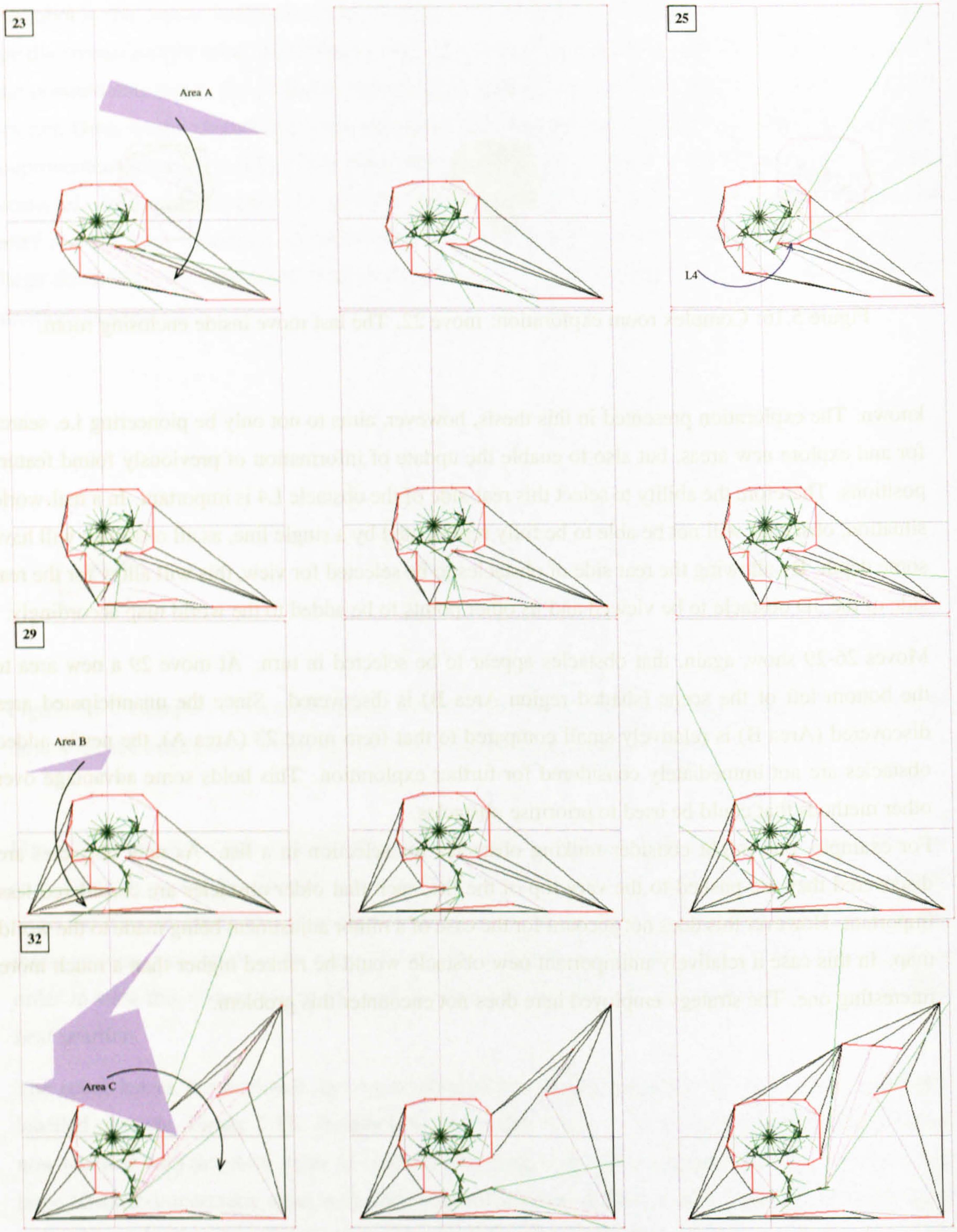


Figure 5.17: Complex room exploration: moves 23–34. Exploration lead by obstacle-importance; differing sizes of areas discovered which affect the obstacle's importance measures which overpower the obstacle selection algorithm.

This selection method allows the algorithm to consider these previously discovered obstacles to be rated more highly because they were found at a more prominent area. This allows for obstacles that are discovered behind a very small new area to not be rated higher than those which have enabled a much larger area to be explored. In move 32 a large area is discovered (shaded region Area C) and the new obstacles that surround this new area are all labelled with a very large obstacle-importance.

The exploration continues in Figure 5.18 as the robot enters Area 2. As the robot moves up the right hand side of the scene, an obstacle in the bottom right corner is not detected, which could be hazardous as a path could be planned into it. This is a downfall of the limited field-of-view sensors compared to 360° scanners. In this example exploration a path is not planned through the undetected obstacle; however, if such a path was chosen and executed the path would not be able to be completed. In this case the restricted motion would be recorded and the desired view would not be obtained, but the algorithm would proceed, nevertheless.

It should be noted that the obstacle selected in Figure 5.18 at move 40 was that which was previously labelled an obstacle, marked with an asterisk and not, as it would appear, the rear-side of the already detected obstacle, which was not thought visible from the start of the move. This obstacle was selected because of its extremely high obstacle-importance and reasonable joint-improvement.

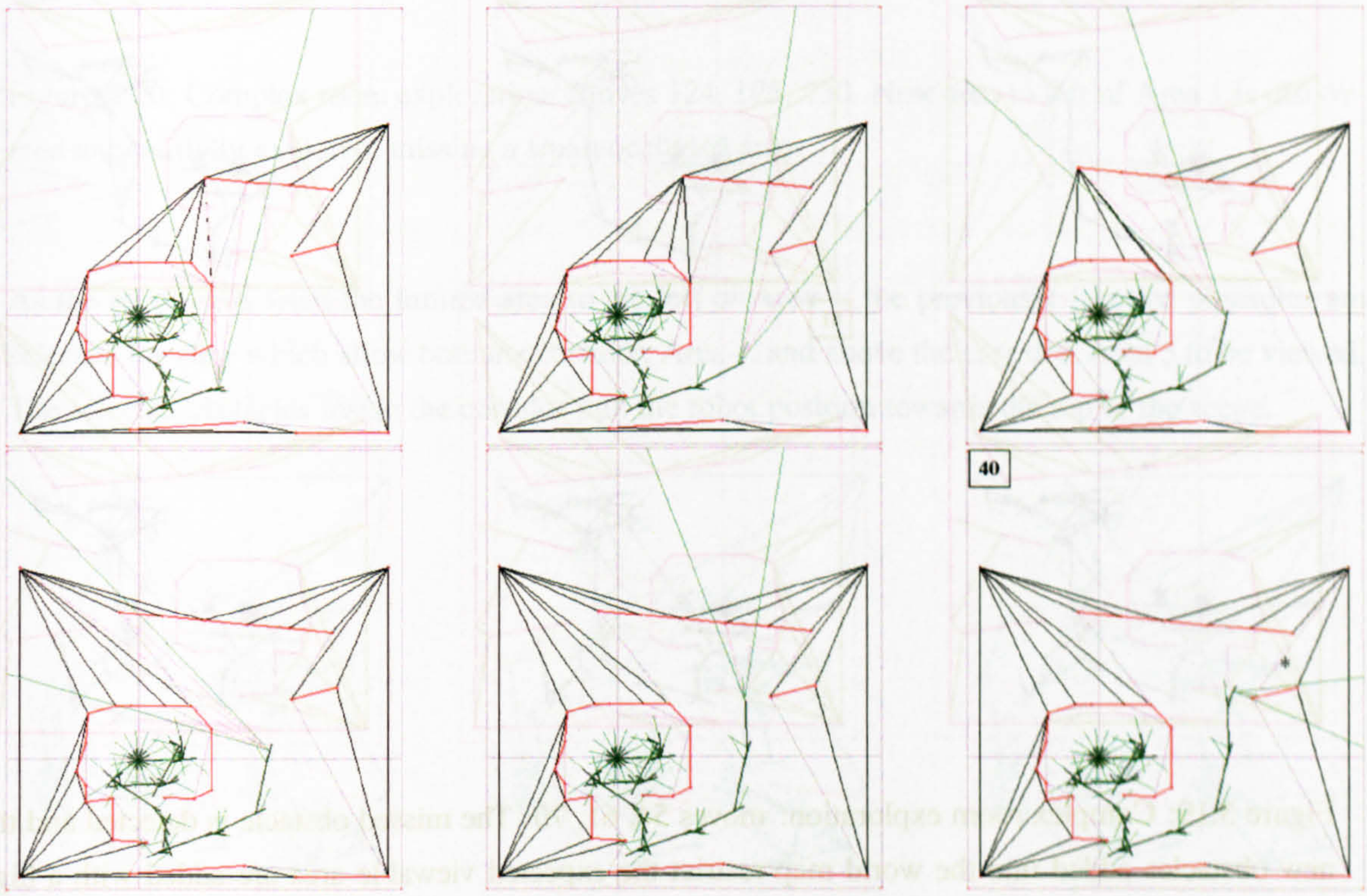


Figure 5.18: Complex room exploration: moves 35–40; moving through the scene an obstacle is initially not detected. The obstacle marked * is selected for view in move 40.

Figure 5.19 shows a summary of some of the moves as the robot back-tracks across the scene. The obstacle that was previously undetected in Figure 5.18 is now discovered when an exterior obstacle is selected for view. The large area improvement from viewing across the scene from the opposite direction is in fact not achieved due to the obstacle being detected and the area is much smaller than that which was anticipated. The area is updated once an obstacle is detected and is added into the world map, so that the area is correctly labelled as unseen. The measure of obstacle-importance is added to the newly discovered obstacles in the same way as it would be if the area had been expanded, as it is still new information and the new obstacles should be explored further.

At move 81 an obstacle at the bottom of the scene is selected, which for the first time is viewed from an angle such that the previously occluded exterior obstacle can be detected. The new area behind this obstacle is viewed further and shows that even very narrow openings can be viewed. The options for viewing positions of these newly discovered obstacles are limited since the robot is not able to travel into the new area; the new obstacles are not selected immediately, but are selected once the option for a good view of the obstacle through the opening can be achieved, three moves later.

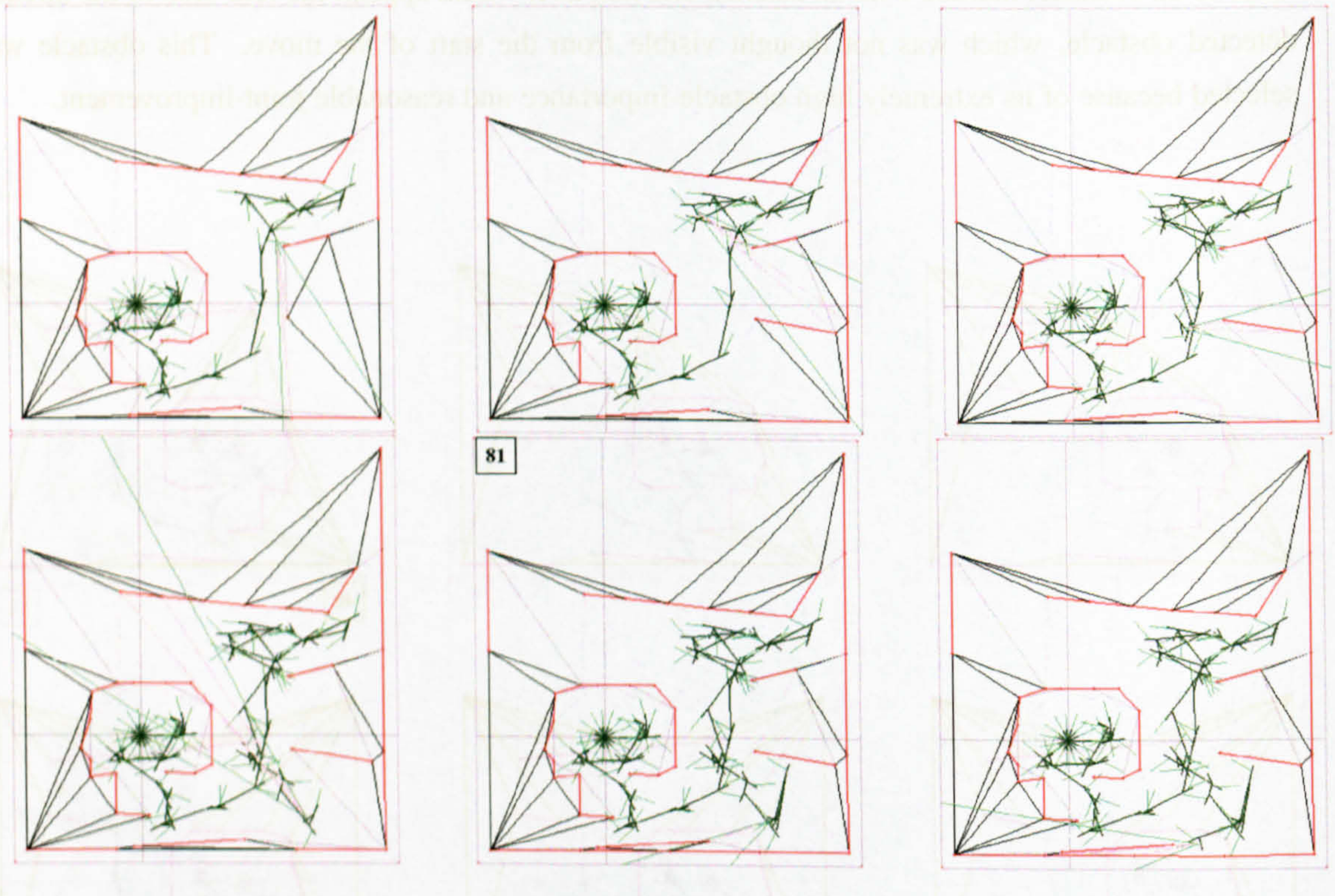


Figure 5.19: Complex room exploration: moves 54, 61, 70: The missed obstacle is detected and the new obstacles added into the world map restrict the expected viewable area are added with a high obstacle-importance. Moves 78, 81, & 84: a small area at the bottom of the scene is detected.

The exploration continues in Figure 5.20 where the path of the robot has been led up and over the initial enclosing area, as the remainder of the scene begins to be explored. A new area to the left of Area 1 is discovered, and upon further investigation the robot travels down into the area but does not actually view all the available area; a small section of the viewable area is missed. Other researchers (e.g. [Pito, 1996]) have written algorithms to calculate a position from which to view an area, taking into account occlusions from known obstacles; however, no such analysis of the area is carried out in this research as it is not assumed that the entire area will be visible. Therefore if an algorithm was in place to select views for a given area it may lead the robot off the path of pioneering exploration. Up to this point, move 130, 65% of the area and 73% of the obstacles have been seen.

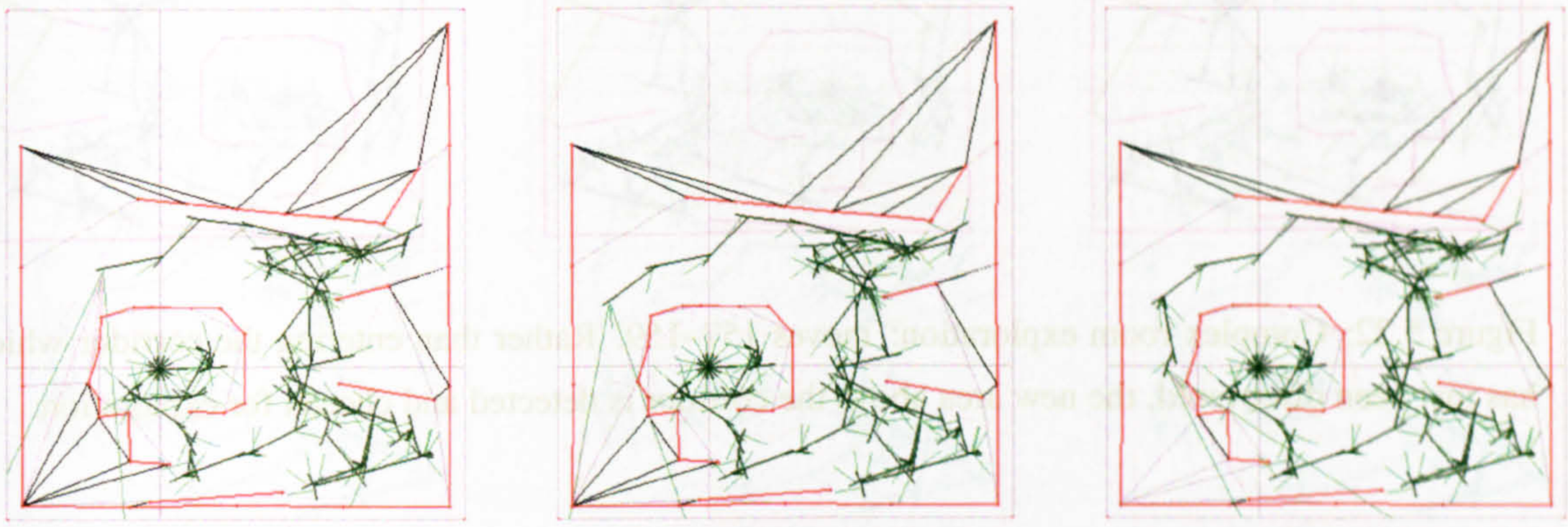


Figure 5.20: Complex room exploration: moves 124, 125, 130. New area to left of Area 1 is discovered and partially explored, missing a small occluded area.

As the robot exits from the limited area to the left of Area 1, the previously labelled obstacles are selected for view which allow both the corridor, Area 4, and above the corridor, Area 5 to be viewed. The selected obstacles inside the corridor pull the robot position towards the top of the scene.

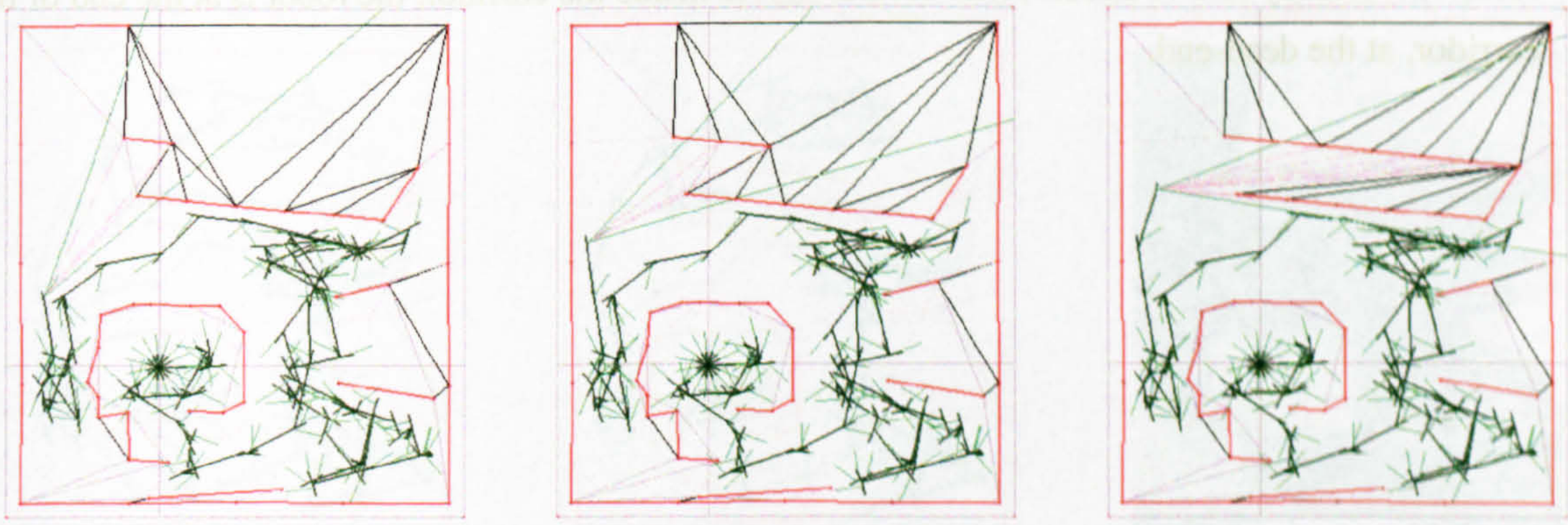


Figure 5.21: Complex room exploration: moves 147–149. New areas, Area 4 & 5, are discovered and initially viewed.

As the robot moves towards the top of the scene and its view direction is towards the end of the corridor, to get the best view of the obstacles that make up the corridor the robot has to travel inside the corridor. In Figure 5.22 however, the selected robot path is not to enter the corridor, but to view the new and larger area above the corridor. This is purely because of the magnitude of the obstacle-importance for those obstacles that surround Area 5.

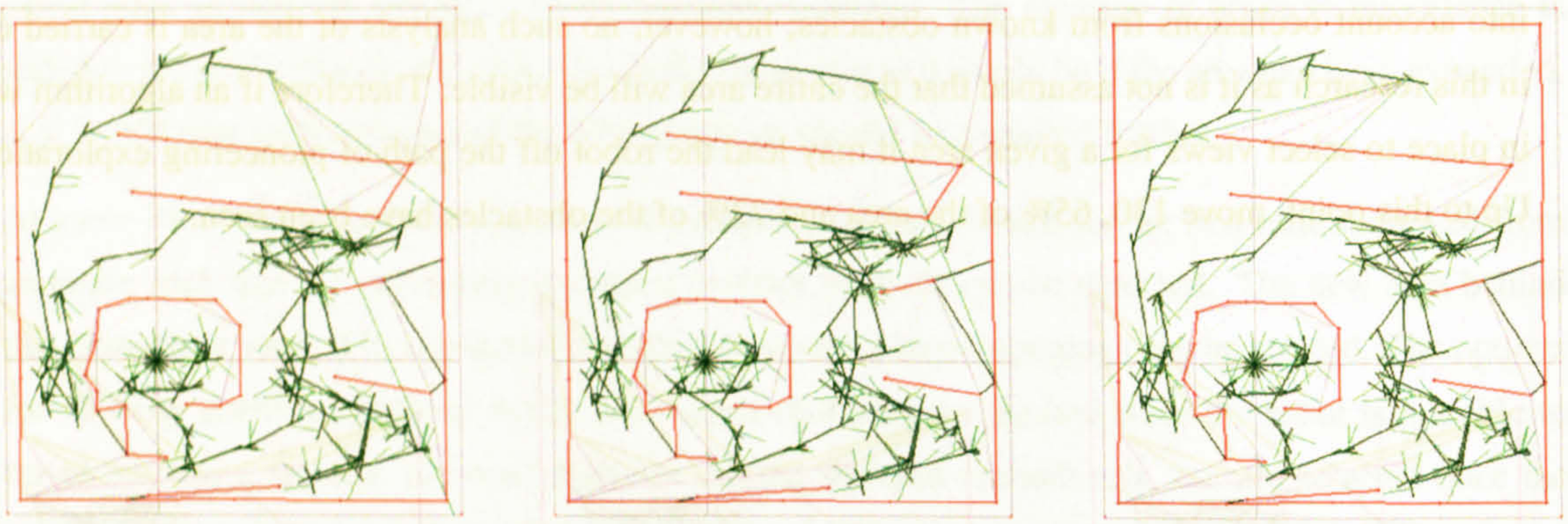


Figure 5.22: Complex room exploration: moves 157–159. Rather than entering the corridor which has just been discovered, the new area above the corridor is detected and chosen for exploration.

The robot travels around Area 5 for thirty moves until the position of the robot allows for the next-viewing position to again allow the interior of the corridor, Area 4, to be visible. Figure 5.23 shows the robot entering the corridor. As the robot enters the corridor and moves towards the dead-end, the first set of moves will add viewing orientations into obstacle's histograms from only one side. As the robot proceeds down the corridor the angles added into the obstacle's histograms will be significantly different if the viewing angle is chosen to view towards the exit of the corridor. This is useful since it enables the map building algorithm to keep track of the position of the robot with respect to scene features it already knows about. After several moves inside the corridor, the robot is at the end of the corridor, at the dead-end.

The last snapshot of Figure 5.23 is the area-view histogram, which shows that the majority of the scene area has been viewed at least once except for a few small areas to the bottom left and one to the right of the scene.

The exploration continues inside the corridor. When obstacles are selected for view their obstacle-importance is divided by a factor of ten. This makes those obstacles less important to the selection algorithm. Of course if all available obstacles have a very low importance value, then the maximum joint-improvement function will have a very low value. As the selection algorithm is limited by the number of choices of moves available to it, the robot's path can be non-direct. This is clear in Figure 5.24 as the robot struggles to exit the corridor.

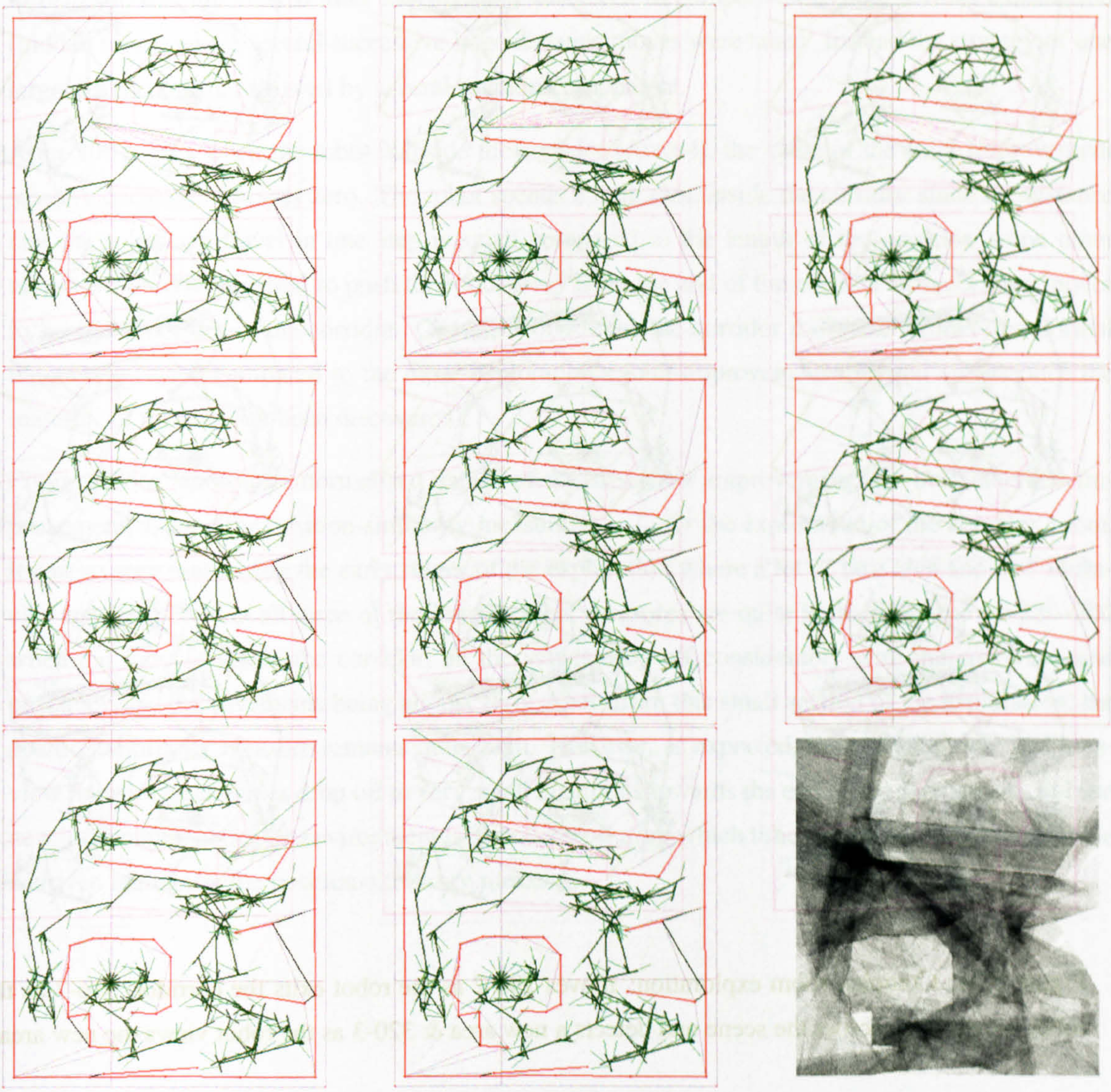


Figure 5.23: Complex room exploration: moves 192–194, 202–204, 206, & 207 (obstacle map and seen areas histogram): Robot travelling down the corridor towards the dead end.

Figure 5.24 shows the final steps of the exploration for the complex scene. Moves 290-2 shows the robot exiting the corridor (one hundred moves after entering it); moves 316-20 shows the robot moving back across the scene, using areas of the scene that have not yet been visited (since the joint-improvement measure will be higher for these regions). Moves 321-3 shows the obstacle selection algorithm finding obstacles with left-over obstacle-importance and discovers a small new area to the right of the scene (in Area 2) which was previously missed.

Figure 5.24: Complex room exploration: moves 290-2 as the robot exits the corridor, 316-7 as the robot moves back across the scene and detects a new area & 320-3 as the robot views the new area.

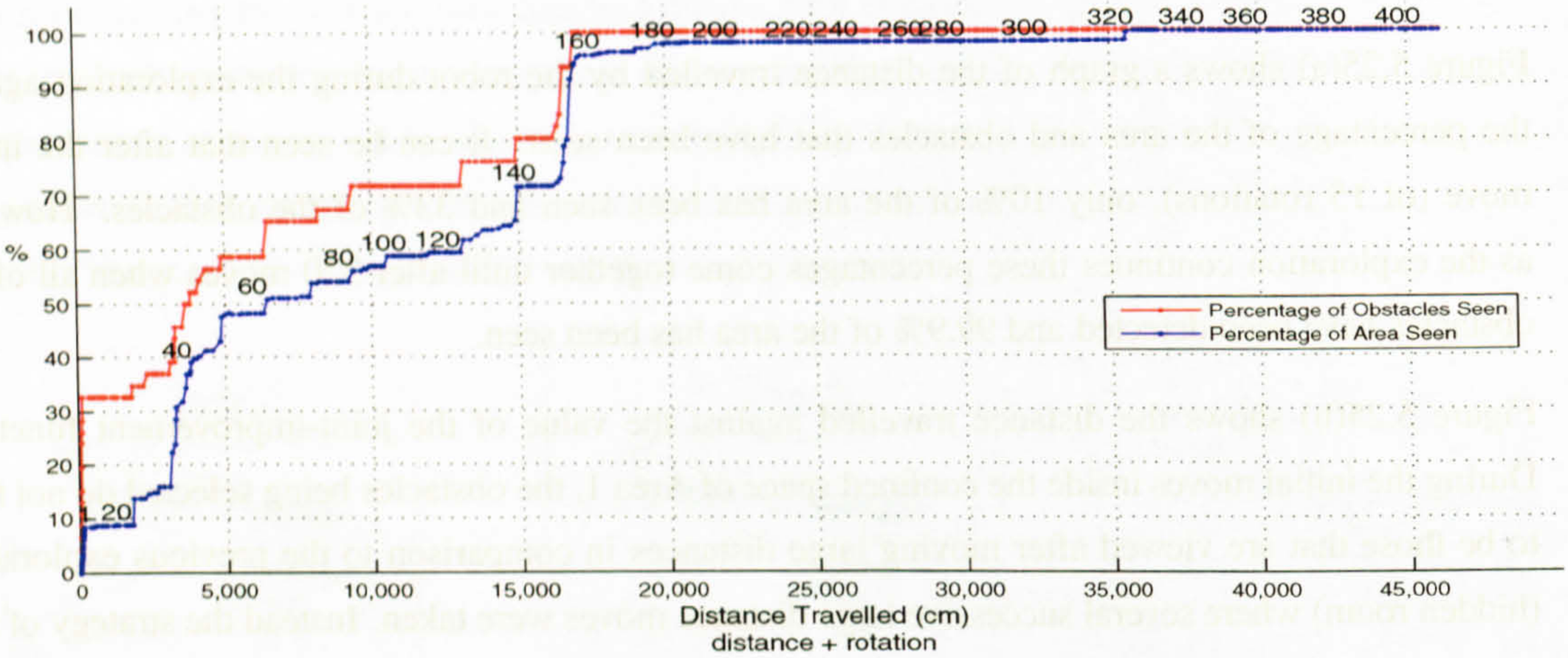
5.5.2 Analysis

Figure 5.25(a) shows a graph of the distance travelled by the robot during the exploration against the percentage of the area and obstacles that have been seen. It can be seen that after the initial move (of 15 rotations), only 10% of the area has been seen and 33% of the obstacles. However as the exploration continues these percentages come together until after 320 moves when all of the obstacles have been detected and 99.9% of the area has been seen.

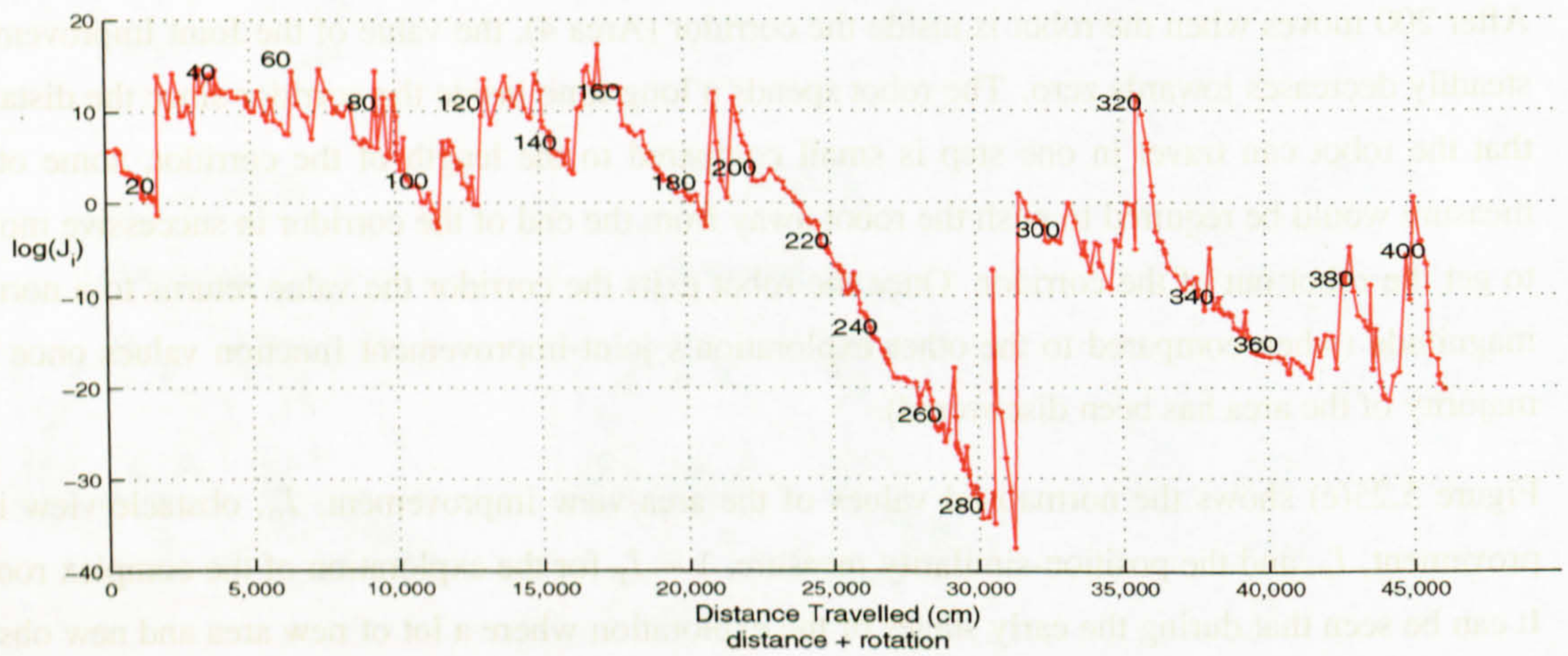
Figure 5.25(b) shows the distance travelled against the value of the joint-improvement function. During the initial moves inside the confined space of Area 1, the obstacles being selected do not tend to be those that are viewed after moving large distances in comparison to the previous exploration (hidden room) where several successive large distance moves were taken. Instead the strategy of one large distance move followed by several rotations is apparent.

After 200 moves when the robot is inside the corridor (Area 4), the value of the Joint Improvement steadily decreases towards zero. The robot spends a long time inside the corridor since the distance that the robot can travel in one step is small compared to the length of the corridor, some other measure would be required to push the robot away from the end of the corridor in successive moves to get the robot out of the corridor. Once the robot exits the corridor the value returns to a normal magnitude (when compared to the other exploration's joint-improvement function values once the majority of the area has been discovered).

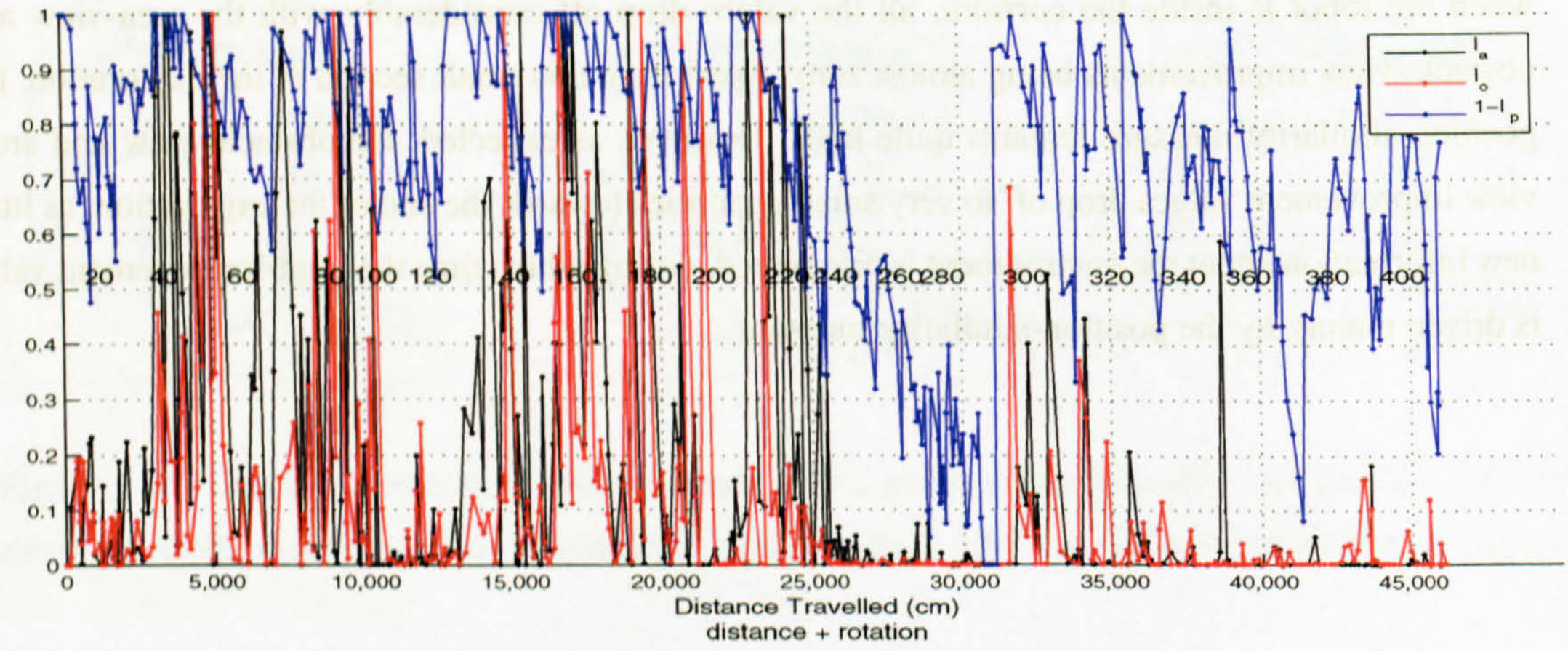
Figure 5.25(c) shows the normalised values of the area-view improvement, I_a, obstacle-view improvement, I_o, and the position-similarity measure, $1 - I_p$ for the exploration of the complex room. It can be seen that during the early stages of the exploration where a lot of new area and new obstacles are detected, that all three of the improvement measures are quite high. From moves 220–300 when the robot is inside the corridor, all the values drop off considerably with the area-view and obstacle-view improvements being almost zero. Apart from this small section of the exploration, the position-similarity measure remains quite high. However, as expected, the obstacle-view and area-view improvement values drop off to very small quantities towards the end of the exploration, as little new information about the environment is discovered, during which time the joint-improvement value is driven mainly by the position-similarity measure.



(a) Percentage of obstacles and areas seen versus total distance travelled.



(b) Joint-improvement versus total distance travelled.



(c) Obstacle-view, area-view and position-similarity values versus total distance travelled.

Figure 5.25: Analysis of complex room exploration using joint improvement measure.

5.6 Conclusions

This chapter has reviewed the amalgamation of three improvement measures along with an obstacle improvement measure to form a joint-improvement function that has been shown to successfully explore unknown environments. Using all three measures (obstacle-view improvement, area-view improvement and position-similarity along with the obstacle-importance, the behaviours of the individual improvements complement each other well. The obstacle-view improvement alone did not allow the robot to move. The area-view improvement moved in sequences of a large step followed by a series of scans. The position-similarity randomly walked around the scene in a similar fashion to the well known lawn-mower algorithms. Used together a sensible exploration is achieved without navigation algorithms for specific scenarios being hard-coded.

Chapter 6

Discussion

A new approach to the unknown environment exploration problem has been presented. The goal of the system was to maximise the improvement in knowledge of the environment as an autonomous agent moves about the area. From no prior knowledge of the environment, a world map representation is built up incrementally using selected viewing positions calculated using a next-best-view technique. The next move is chosen from a set of options to maximise the improvement on all previous views of obstacles. The presented strategy was shown to successfully explore a number of environments, [Gartshore and Palmer, 2005], [Gartshore et al., 2005].

A new viewpoint planning algorithm was presented to calculate the position for the next-best-view, and was shown to optimise the viewing angle through an opening, and around a corner, for a given distance to move.

The incremental map building was shown to use an accumulator grid technique to gather evidence of features during the robot's motion, [Gartshore et al., 2002]. Features with a high confidence are added to a world map representation, and errors in the position of these features were shown to reduce over time.

6.1 Review of the Method Approach

A view-improvement strategy as been introduced, as the initial step towards the aim of maximising the knowledge of an unknown environment. Three improvement measures were presented where a limited field-of-view was imposed: obstacle-view improvement, area-view improvement and position-similarity. Each of these techniques displayed promising characteristics for an autonomous exploration, however they did not appear to handle a variety of situations well. Also considered were the three measures combined in a joint improvement function, and along with an importance value for recently discovered obstacles a successful exploration algorithm was achieved.

The exploration algorithm is repeatable since there is no random calculation, there is always a deciding choice. This however is partly based on which obstacle was discovered first rather than which

can be predicted to provide more new information, since this cannot be calculated. It would be possible to use an estimate of the area behind an obstacle as a prediction to the likelihood of the obstacle allowing more information of the unexplored environment to be discovered. This, however, may hamper the algorithm in the case of a dead end–where it may assume that one of the obstacles must at some point allow the unseen area behind to be viewed, without being able to retreat as they become less interesting when no new information is provided.

Another addition to the strategy could be to evaluate the viewed area at a pre-defined stage in the exploration process, say after 50 moves. The areas which had at this point not yet once been viewed could be targeted. However, not only would this alter the approach to the exploration problem to more of a target-oriented problem, the question of which area to explore first would still need to be considered. Throughout this work no assumption was made that all of the environment area would be able to be traversed or viewed. The target-oriented nature of such an extension would contradict that assumption.

The next-best-view algorithm developed an optimal approach to the problem of calculating the best view around a corner and through an opening. This calculation was shown to be a useful tool to traverse an environment based on seeking the greatest amount of new information for a limited distance for a robot to move. During the exploration of an unknown environment, the next-best-view calculation for each obstacle in the agent's vicinity provided a limited choice of next moves with their corresponding view-improvement for comparison, where the obstacle with the highest improvement was selected for view. The first 20 moves of each exploration of the environments were very successful in seeking out the incorrectly labelled obstacles, allowing uncharted areas to be discovered. However, for some of the situations shown, e.g. the confined space of the corridor with dead-end presented in Figure 5.14, this meant that the overall path of the agent was not what may be considered as sensible behaviour. This is discussed in the future work section.

Also discussed was the problem of constructing a map of the obstacles to the robot in two-dimensions. The position information of obstacles in the environment was used by the exploration algorithm, where the confidence of obstacle features existing in the environment allowed the confidence of the map's construction lines being obstacles to be inferred. Due to the modular nature of the approach to exploration developed in this work, any map building algorithm that provides position and confidence information of obstacle features relative to a world reference frame could be used.

6.2 Extensions for Future Research

The view-improvement exploration and next-best-view strategies presented in this thesis have only been tested on scenarios for a simulated agent. Although theoretical analysis of the error in the calculation of the next-best-viewpoint has been carried out, the full impact of this cannot be fully known until a real-world system has been employed to test it.

The environments considered for analysis with the view-improvement exploration were constructed

from straight thin obstacles. Inevitably obstacles detected in real-world experiments will be more detailed, perhaps with several features on the obstacle surface, and obstacles which are not straight flat walls. In these environments, the map being built may be too detailed for the purposes of exploration. Perhaps it is unnecessary to maintain all the knowledge about the world obstacles in the map queried for exploration decisions, such that a hierarchical structure of maps required for exploration, obstacle avoidance and environment modelling could be considered.

For the exploration algorithm in its current state, a factor when errors are involved will be the calculation of the predicted and acquired view area for the obstacle importance measure introduced in section 5.2.1. The current approach requires the predicted and acquired areas for a given obstacle to be identical, this of course will not be the case for a real system as the position of features that define the obstacle will alter with most view updates due to errors in the pose of the robot and feature position calculations. One simple approach would be to consider the percentage of the difference in predicted and acquired view areas, allowing for a given acceptable difference. A more robust solution however would be to feed the information about the shift in the obstacle's feature positions through to the calculation of the obstacle importance so that the calculation of the area could incorporate this feature position update in the predicted area calculation. For example: for the case of feature positions being updated but no new features added to the world map, the predicted and acquired areas should be the same; for the case of new features being added to the world map as well as already mapped feature positions being updated, the predicted area would reflect the view area of the updated known features as well as the true newly acquired area.

Map building algorithms designed for real-world environments must take the effects of errors in the position of the robot and the subsequently added world features into account. Although considered in this work, the effects of such errors was not fully evaluated. The Kalman Filter is a tool often used to estimate the world feature positions and to store information of the errors involved in the mapped data. For the system presented in this work to be useful as a real-world system, the map building module could be replaced by other well tested approaches.

The exploration strategy is restricted to choosing the best view-improvement from a limited set of options. For situations when the robot is free to move with several obstacle view-improvements to choose from, the exploration proceeds nicely. However, when the choice of obstacles is limited e.g. in a long corridor with dead end, the improvement of knowledge dramatically reduces given the limited distance to travel on a given move. For exploration approaches where time or energy are major factors, the exploration could be improved by extending the next-best-view calculation to incorporate the cost of moving against the expected improvement. For situations when the number of move decisions are limited because of many surrounding obstacles limiting the valid paths, this problem could be alleviated by the next-best-viewpoint calculation incorporating a variable distance for the robot to move.

The restriction placed upon this work that the robot travels on the horizontal ground plane rather limits the choice of environments that this work could be directly applied to. For the system to be

employed in a variety of real-world scenarios, this assumption would need to be removed. There remains a choice for mapping the environment in a simplified three-dimensional view, such as 2.5-dimensions where different heights of obstacles are considered, or whether a fully three-dimensional map would be needed.

Another extension to the work presented in this thesis could be to consider obstacles that disappear after a period of time, e.g. a parked car which then moves after a number of hours. With the current algorithm, if an obstacle, l, is removed and features previously occluded by this obstacle become visible, the confidence of obstacle l will decrease. Similarly, when obstacle features are not detected at a position from which they should be visible, the confidence of those obstacle features existing will also decrease. A simple solution to deal with such obstacles would be to remove obstacle features from the world mesh once they fall below a given threshold. This design feature would need to be more thoroughly evaluated depending upon the possible situations being considered. With the parked car example, it may be more sensible to continue considering the empty parking space as an obstacle even when no car exists in the space, in order to aid exploration of new areas – otherwise the repeatedly empty car space and newly occupied car space may become more interesting to the algorithm than the exploration of uncharted areas.

Appendix A

Error Analysis on Next-Best-View Calculation

The viewpoint planning strategy is used to calculate the direction θ which the robot should move in order to reach the best viewing position given a fixed distance to travel. Since the calculation of θ depends upon the values of R_1, R_2, γ_1 and γ_2 (the distances and angles to the corners), any errors in these values will affect the direction calculation. Recall that

$$\theta = \left(\frac{\gamma_1 + \gamma_2}{2}\right) - \tan^{-1}\left(\frac{q}{p}\right) - \cos^{-1}\left(\frac{1}{\sqrt{\lambda}}\right)$$

$$\text{where} \quad p = l\cos\mu \qquad q = m\sin\mu \qquad \mu = \left(\frac{\gamma_1 - \gamma_2}{2}\right) \tag{A.1}$$

$$l = \frac{R_1R_2 - r^2}{r(R_1 - R_2)} \quad m = \frac{R_1R_2 + r^2}{r(R_1 + R_2)} \quad \lambda = p^2 + q^2$$

The values of R_1, R_2, γ_1 and γ_2 will be affected by small changes to the positions of the robot and of the corners.

A co-ordinate system is defined to allow the values of R_1, R_2, γ_1, and γ_2 to be expressed in terms of the robot position, robot orientation, and the corner positions and is shown in figure A.1. Here we assume that the errors are linear and local to the point being considered i.e. the robot or corner position is within two to three standard deviations from the estimated point position (and orientation in the case of the camera). We will first consider the problem with errors on the camera position alone, and then with errors on the positions of the corners.

A.1 Calculation of Variance of θ given Errors on Camera Position

Initially let us consider errors in the camera position (P_x,P_y) and view orientation (P_h) only. The position measurements of the robot are taken from the position estimation module discussed in chapter 2.7, and are assumed to have a zero-mean Gaussian error with a diagonal covariance matrix such

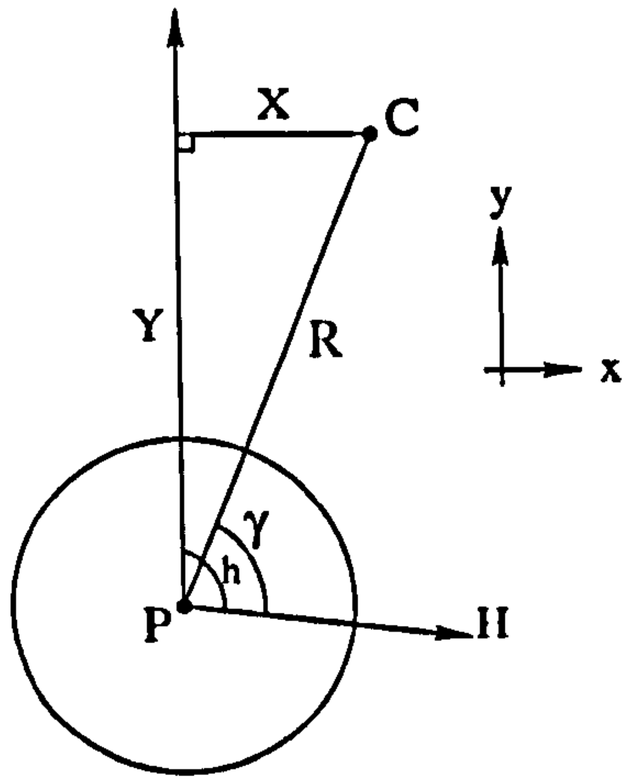


Figure A.1: Co-ordinate system used for the error analysis on the next-best-view calculation of θ. Shown is the robot position at P with original heading H, and one corner C in the vicinity.

$$X = C_x - P_x$$

$$Y = C_y - P_y$$

$$R = \sqrt{X^2 + Y^2}$$

$$\gamma = h - \tan^{-1}\left(\frac{X}{Y}\right)$$

that

$$cov(L) = cov\begin{pmatrix} P_x \\ P_y \\ P_h \end{pmatrix} = \begin{pmatrix} \sigma_{Px}^2 & 0 & 0 \\ 0 & \sigma_{Py}^2 & 0 \\ 0 & 0 & {\sigma_{Ph}}^2 \end{pmatrix} \tag{A.2}$$

where σ_{Px}, σ_{Py}, and σ_{Ph} are the standard deviations of the robots x and y position and orientation in a world co-ordinate frame.

The variance of θ, ${\sigma_\theta}^2$, can be calculated from a series of Jacobian matrix calculations, where S are the parameters of the equation for θ and L is the position of the camera.

$${\sigma_\theta}^2 = \frac{\partial \theta}{\partial S} cov(S) \frac{\partial \theta}{\partial S}^T \quad \text{where} \quad S = [\ \gamma_1 \quad \gamma_2 \quad R_1 \quad R_2\ ]^T \tag{A.3}$$

$$cov(S) = \frac{\partial S}{\partial L} cov(L) \frac{\partial S}{\partial L}^T \quad \text{where} \quad L = [\ P_x \quad P_y \quad P_h\ ]^T \tag{A.4}$$

We first need to calculate $cov(S)$ from the Jacobian error matrix of $\partial S/\partial L$ where

$$\frac{\partial S}{\partial L} = \begin{pmatrix} \frac{\partial \gamma_1}{\partial P_x} & \frac{\partial \gamma_1}{\partial P_y} & \frac{\partial \gamma_1}{\partial P_h} \\ \frac{\partial \gamma_2}{\partial P_x} & \frac{\partial \gamma_2}{\partial P_y} & \frac{\partial \gamma_2}{\partial P_h} \\ \frac{\partial R_1}{\partial P_x} & \frac{\partial R_1}{\partial P_y} & \frac{\partial R_1}{\partial P_h} \\ \frac{\partial R_2}{\partial P_x} & \frac{\partial R_2}{\partial P_y} & \frac{\partial R_2}{\partial P_h} \end{pmatrix}$$

These partial derivatives are calculated such that

$$\begin{aligned}\frac{\partial\gamma}{\partial P_x} &= \frac{-1}{1+\frac{X^2}{Y^2}}\cdot\frac{-1}{Y}\\ &= \frac{Y}{X^2+Y^2}\\ \frac{\partial\gamma}{\partial P_y} &= \frac{-1}{1+\frac{X^2}{Y^2}}\cdot\frac{X}{Y^2}\\ &= \frac{-X}{X^2+Y^2}\\ \frac{\partial\gamma}{\partial P_h} &= 1\end{aligned}$$

$$\begin{aligned}\frac{\partial R}{\partial P_x} &= \frac{1}{2}[X^2+Y^2]^{-\frac{1}{2}}.2X.-1\\ &= \frac{-X}{\sqrt{X^2+Y^2}}\\ \frac{\partial R}{\partial P_y} &= \frac{1}{2}[X^2+Y^2]^{-\frac{1}{2}}.2Y.-1\\ &= \frac{-Y}{\sqrt{X^2+Y^2}}\\ \frac{\partial R}{\partial P_h} &= 0\end{aligned}$$

By introducing the following notation:

$$\begin{aligned} i &= {X_1}^2+{Y_1}^2 & V_x &= {\sigma_{Px}}^2\\ j &= {X_2}^2+{Y_2}^2 & V_y &= {\sigma_{Py}}^2\\ & & V_h &= {\sigma_{Ph}}^2\end{aligned}$$

we can calculate the covariance matrix $cov(S)$ using equation [A.4] giving us

$$\begin{aligned} cov(S) &= \begin{pmatrix} \frac{Y_1}{i} & \frac{-X_1}{i} & 1\\ \frac{Y_2}{j} & \frac{-X_2}{j} & 1\\ \frac{-X_1}{\sqrt{i}} & \frac{-Y_1}{\sqrt{i}} & 0\\ \frac{-X_2}{\sqrt{j}} & \frac{-Y_2}{\sqrt{j}} & 0\end{pmatrix}\cdot\begin{pmatrix} V_{P_x} & 0 & 0\\ 0 & V_{P_y} & 0\\ 0 & 0 & V_{P_h}\end{pmatrix}\cdot\begin{pmatrix} \frac{Y_1}{i} & \frac{Y_2}{j} & \frac{-X_1}{\sqrt{i}} & \frac{-X_2}{\sqrt{j}}\\ \frac{-X_1}{i} & \frac{-X_2}{j} & \frac{-Y_1}{\sqrt{i}} & \frac{-Y_2}{\sqrt{j}}\\ 1 & 1 & 0 & 0\end{pmatrix}\\ &= \begin{pmatrix} \frac{Y_1V_{P_x}}{i} & \frac{-X_1V_{P_y}}{i} & V_{P_h}\\ \frac{Y_2V_{P_x}}{j} & \frac{-X_2V_{P_y}}{j} & V_{P_h}\\ \frac{-X_1V_{P_x}}{\sqrt{i}} & \frac{-Y_1V_{P_y}}{\sqrt{i}} & 0\\ \frac{-X_2V_{P_x}}{\sqrt{j}} & \frac{-Y_2V_{P_y}}{\sqrt{j}} & 0\end{pmatrix}\cdot\begin{pmatrix} \frac{Y_1}{i} & \frac{Y_2}{j} & \frac{-X_1}{\sqrt{i}} & \frac{-X_2}{\sqrt{j}}\\ \frac{-X_1}{i} & \frac{-X_2}{j} & \frac{-Y_1}{\sqrt{i}} & \frac{-Y_2}{\sqrt{j}}\\ 1 & 1 & 0 & 0\end{pmatrix}\\ &= \begin{pmatrix} \frac{V_{P_x}{Y_1}^2+V_{P_y}{X_1}^2}{i^2}+V_{P_h} & -- & -- & --\\ \frac{V_{P_x}Y_1Y_2+V_{P_y}X_1X_2}{ij}+V_{P_h} & \frac{V_{P_x}{Y_2}^2+V_{P_y}{X_2}^2}{j^2}+V_{P_h} & -- & --\\ \frac{X_1Y_1(V_{P_y}-V_{P_x})}{i\sqrt{i}} & \frac{-V_{P_x}X_1Y_2+V_{P_y}X_2Y_1}{j\sqrt{i}} & \frac{V_{P_x}{X_1}^2+V_{P_y}{Y_1}^2}{i} & --\\ \frac{-V_{P_x}X_2Y_1+V_{P_y}X_1Y_2}{i\sqrt{j}} & \frac{X_2Y_2(V_{P_y}-V_{P_x})}{j\sqrt{j}} & \frac{V_{P_x}X_1X_2+V_{P_y}Y_1Y_2}{\sqrt{ij}} & \frac{V_{P_x}{X_2}^2+V_{P_y}{Y_2}^2}{j}\end{pmatrix}\end{aligned} \tag{A.5}$$

where $--$ is used in place of the diagonally symmetric counterpart, and is just used for clarity.

Now that we have $cov(S)$, we now differentiate θ with respect to γ_1, γ_2, R_1 and R_2 to calculate the variance of θ. Using the definition of θ from equation [A.1], we split the equation of θ into three

parts: $\left(\frac{\gamma_1+\gamma_2}{2}\right)$, $\tan^{-1}\left(\frac{q}{p}\right)$, and $\cos^{-1}\left(\frac{1}{\sqrt{\lambda}}\right)$ which can be differentiated separately. The latter two are simplified further so that the 'chain rule' can be applied.

$$\frac{\partial}{\partial\gamma_{1,2}}\left(\frac{\gamma_1+\gamma_2}{2}\right)=\frac{1}{2} \qquad \frac{\partial}{\partial R_{1,2}}\left(\frac{\gamma_1+\gamma_2}{2}\right)=0$$

Using $u=\frac{q}{p}$, $\frac{\partial}{\partial u}\left(\tan^{-1}u\right)=\frac{1}{1+u^2}=\frac{p^2}{\lambda}$

Using $v=\sqrt{p^2+q^2}$, $\frac{\partial}{\partial v}\left(\cos^{-1}v\right)=\frac{-1}{\sqrt{1-v^2}}=\frac{-1}{\sqrt{1-\lambda}}$

p, l, q, and m are used to simplify the calculation of θ. Since they are used repeatedly throughout the workings, their derivatives are

$$\frac{\partial p}{\partial\mu}=-l\sin\mu \qquad \frac{\partial q}{\partial\mu}=m\cos\mu$$

$$\frac{\partial p}{\partial l}=\cos\mu \qquad \frac{\partial q}{\partial m}=\sin\mu$$

$$\frac{\partial l}{\partial R_1}=\frac{R_2r(R_1+R_2)-r(R_1R_2+r^2)}{r^2(R_1+R_2)^2} \qquad \frac{\partial m}{\partial R_1}=\frac{R_2r(R_1-R_2)-r(R_1R_2-r^2)}{r^2(R_1-R_2)^2}$$

$$=\frac{{R_2}^2-r^2}{r(R_1+R_2)^2} \qquad =\frac{-({R_2}^2-r^2)}{r(R_1-R_2)^2}$$

$$\frac{\partial l}{\partial R_2}=\frac{R_1r(R_1+R_2)-r(R_1R_2+r^2)}{r^2(R_1+R_2)^2} \qquad \frac{\partial m}{\partial R_2}=\frac{R_1r(R_1-R_2)+r(R_1R_2-r^2)}{r^2(R_1-R_2)^2}$$

$$=\frac{{R_1}^2-r^2}{r(R_1+R_2)^2} \qquad =\frac{{R_1}^2-r^2}{r(R_1-R_2)^2}$$

With these simplified derivatives, the following base derivatives are formed.

$$\frac{\partial p}{\partial\gamma_{1_{1,2}}}=-l\sin\mu.\frac{1}{2} \qquad \frac{\partial q}{\partial\gamma_{1_{1,2}}}=m\cos\mu.\frac{1}{2}$$

$$=\frac{-l}{2}\sin\mu \qquad =\frac{m}{2}\cos\mu$$

$$\frac{\partial p}{\partial R_1}=\cos\mu.\frac{{R_2}^2-r^2}{r(R_1+R_2)^2} \qquad \frac{\partial q}{\partial R_1}=\sin\mu.\frac{-({R_2}^2-r^2)}{r(R_1-R_2)^2}$$

$$=\frac{{R_2}^2-r^2}{r(R_1+R_2)^2}cos\mu \qquad =\frac{-({R_2}^2-r^2)}{r(R_1-R_2)^2}\sin\mu$$

$$\frac{\partial p}{\partial R_2}=\cos\mu.\frac{{R_1}^2-r^2}{r(R_1+R_2)^2} \qquad \frac{\partial q}{\partial R_2}=\sin\mu.\frac{{R_1}^2-r^2}{r(R_1-R_2)^2}$$

$$=\frac{{R_1}^2-r^2}{r(R_1+R_2)^2}\cos\mu \qquad =\frac{{R_1}^2-r^2}{r(R_1-R_2)^2}\sin\mu$$

We can now differentiate u to be used with the chain rule for the tan^{-1} term.

$$\frac{\partial u}{\partial \gamma_1} = \left[\frac{mp}{2}\cos\mu + \frac{lq}{2}\sin\mu\right]\frac{1}{p^2}$$

$$\Rightarrow \frac{\partial}{\partial \gamma_1}\left[\tan^{-1}\left(\frac{q}{p}\right)\right] = \frac{p^2}{\lambda}\frac{1}{p^2}\left[\frac{lm}{2}\cos^2\mu + \frac{ml}{2}\sin^2\mu\right] = \frac{lm}{2\lambda}$$

$$\frac{\partial u}{\partial \gamma_2} = \left[\frac{-m}{2}\cos\mu\cdot p - \frac{l}{2}\sin\mu\cdot q\right]\cdot\frac{1}{p^2}$$

$$\Rightarrow \frac{\partial}{\partial \gamma_2}\left[\tan^{-1}\left(\frac{q}{p}\right)\right] = \frac{p^2}{\lambda}\cdot\frac{1}{p^2}\left[\frac{-lm}{2}\cos^2\mu - \frac{ml}{2}\sin^2\mu\right] = \frac{-lm}{2\lambda}$$

$$\frac{\partial u}{\partial R_1} = \left[\frac{-({R_2}^2 - r^2)}{r(R_1 - R_2)^2}\sin\mu\cdot p - \frac{{R_2}^2 - r^2}{r(R_1 + R_2)^2}\cos\mu\cdot q\right]\cdot\frac{1}{p^2}$$

$$\Rightarrow \frac{\partial}{\partial R_1}\left[\tan^{-1}\left(\frac{q}{p}\right)\right] = \frac{p^2}{\lambda}\cdot\frac{-1}{p^2}\left[\frac{{R_2}^2 - r^2}{r(R_1 - R_2)^2}l\sin\mu\cos\mu + \frac{{R_2}^2 - r^2}{r(R_1 + R_2)^2}m\cos\mu\sin\mu\right]$$

$$= \frac{-({R_2}^2 - r^2)\sin\mu\cos\mu}{r\lambda}\left[\frac{l}{(R_1 - R_2)^2} + \frac{m}{(R_1 + R_2)^2}\right]$$

$$\frac{\partial u}{\partial R_2} = \left[\frac{{R_1}^2 - r^2}{r(R_1 - R_2)^2}sin\mu\cdot p - \frac{{R_1}^2 - r^2}{r(R_1 + R_2)^2}\cos\mu\cdot q\right]\cdot\frac{1}{p^2}$$

$$\Rightarrow \frac{\partial}{\partial R_2}\left[\tan^{-1}\left(\frac{q}{p}\right)\right] = \frac{p^2}{\lambda}\cdot\frac{1}{p^2}\left[\frac{{R_1}^2 - r^2}{r(R_1 - R_2)^2}l\sin\mu\cos\mu - \frac{{R_1}^2 - r^2}{r(R_1 + R_2)^2}m\cos\mu\sin\mu\right]$$

$$= \frac{({R_1}^2 - r^2)\sin\mu\cos\mu}{r\lambda}\left[\frac{l}{(R_1 - R_2)^2} - \frac{m}{(R_1 + R_2)^2}\right]$$

Also v for the cos^{-1} term.

$$\frac{\partial v}{\partial \gamma_1} = \frac{-1}{2\lambda\sqrt{\lambda}}\left(2p\cdot\frac{-l}{2}\sin\mu + 2q\cdot\frac{m}{2}\cos\mu\right)$$

$$\Rightarrow \frac{\partial}{\partial \gamma_1}\left[\cos^{-1}\frac{1}{\sqrt{\lambda}}\right] = \frac{-1}{\sqrt{1 - 1/\lambda}}\cdot\frac{-\sin\mu\cos\mu}{2\lambda\sqrt{\lambda}}(m^2 - l^2)$$

$$= \frac{-\sin\mu\cos\mu}{2\lambda\sqrt{\lambda - 1}}(l^2 - m^2)$$

$$\frac{\partial v}{\partial \gamma_2} = \frac{-1}{2\lambda\sqrt{\lambda}}\left(2p \cdot \frac{l}{2}\sin\mu + 2q \cdot \frac{-m}{2}\cos\mu\right)$$

$$\Rightarrow \frac{\partial}{\partial \gamma_2}\left[\cos^{-1}\frac{1}{\sqrt{\lambda}}\right] = \frac{-1}{\sqrt{1-1/\lambda}} \cdot \frac{-1}{2\lambda\sqrt{\lambda}}(l^2 \sin\mu\cos\mu - m^2\cos\mu\sin\mu)$$

$$= \frac{\sin\mu\cos\mu}{2\lambda\sqrt{\lambda-1}}(l^2 - m^2)$$

$$\frac{\partial v}{\partial R_1} = \frac{-1}{2\lambda\sqrt{\lambda}} \cdot \left[l\cos\mu\cos\mu\frac{{R_2}^2 - r^2}{r(R_1+R_2)^2} + m\sin\mu\sin\mu\frac{-({R_2}^2 - r^2)}{r(R_1-R_2)^2}\right]$$

$$\Rightarrow \frac{\partial}{\partial R_1}\left[\cos^{-1}\frac{1}{\sqrt{\lambda}}\right] = \frac{-1}{\sqrt{1-1/\lambda}} \cdot \frac{-1}{2\lambda\sqrt{\lambda}} \cdot \frac{{R_2}^2 - r^2}{r}\left[\frac{l\cos^2\mu}{(R_1+R_2)^2} - \frac{m\sin^2\mu}{(R_1-R_2)^2}\right]$$

$$= \frac{{R_2}^2 - r^2}{r\lambda\sqrt{\lambda-1}}\left(\frac{l\cos^2\mu}{(R_1+R_2)^2} - \frac{m\sin^2\mu}{(R_1+R_2)^2}\right)$$

$$\frac{\partial v}{\partial R_2} = \frac{-1}{2\lambda\sqrt{\lambda}} \cdot \left[l\cos\mu\cos\mu\frac{{R_1}^2 - r^2}{r(R_1+R_2)^2} + m\sin\mu\sin\mu\frac{{R_1}^2 - r^2}{r(R_1-R_2)^2}\right]$$

$$\Rightarrow \frac{\partial}{\partial R_2}\left[\cos^{-1}\frac{1}{\sqrt{\lambda}}\right] = \frac{-1}{\sqrt{1-1/\lambda}} \cdot \frac{-1}{2\lambda\sqrt{\lambda}} \cdot \frac{{R_1}^2 - r^2}{r}\left[\frac{l\cos^2\mu}{(R_1+R_2)^2} + \frac{m\sin^2\mu}{(R_1-R_2)^2}\right]$$

$$= \frac{{R_1}^2 - r^2}{r\lambda\sqrt{\lambda-1}}\left(\frac{l\cos^2\mu}{(R_1+R_2)^2} + \frac{m\sin^2\mu}{(R_1-R_2)^2}\right)$$

We can now bring together each section of the derivative of θ with respect to γ_1, γ_2, R_1 and R_2 (using G and R to represent these derivatives).

$$G_a = \frac{\partial\theta}{\partial\gamma_1} = \frac{1}{2} - \frac{lm}{2\lambda} + \frac{\sin\mu\cos\mu}{2\lambda\sqrt{\lambda-1}}(l^2 - m^2)$$

$$G_b = \frac{\partial\theta}{\partial\gamma_2} = \frac{1}{2} + \frac{lm}{2\lambda} - \frac{\sin\mu\cos\mu}{2\lambda\sqrt{\lambda-1}}(l^2 - m^2)$$

$$R_a = \frac{\partial\theta}{\partial R_1} = a\sin\mu\cos\mu\left[\frac{l}{(R_1-R_2)^2}+\frac{m}{(R_1+R_2)^2}\right] - \frac{a}{\sqrt{\lambda-1}}\left[\frac{l\cos^2\mu}{(R_1+R_2)^2}-\frac{m\sin^2\mu}{(R_1-R_2)^2}\right]$$

$$R_b = \frac{\partial\theta}{\partial R_2} = -b\sin\mu\cos\mu\left[\frac{l}{(R_1-R_2)^2}-\frac{m}{(R_1+R_2)^2}\right] - \frac{b}{\sqrt{\lambda-1}}\left[\frac{l\cos^2\mu}{(R_1+R_2)^2}+\frac{m\sin^2\mu}{(R_1-R_2)^2}\right]$$

where $a = \dfrac{{R_2}^2 - r^2}{r\lambda}$ $\quad b = \dfrac{{R_1}^2 - r^2}{r\lambda}$

We can then write the Jacobian matrix multiplication for the variance of θ as

$$V_\theta = \begin{pmatrix} G_a & G_b & R_a & R_b \end{pmatrix} \cdot cov(S) \cdot \begin{pmatrix} G_a \\ G_b \\ R_a \\ R_b \end{pmatrix} \tag{A.6}$$

Such that

$$\begin{aligned} V_\theta = {} & V_{P_x}\left[\frac{{Y_1}^2}{i^2}G_a^2+\frac{Y_1Y_2}{ij}G_aG_b-\frac{X_1Y_1}{i\sqrt{i}}G_aR_a-\frac{X_2Y_1}{i\sqrt{j}}G_aR_b\right]+ \\ & V_{P_y}\left[\frac{{X_1}^2}{i^2}G_a^2+\frac{X_1X_2}{ij}G_aG_b-\frac{X_1Y_1}{i\sqrt{i}}G_aR_a-\frac{X_1Y_2}{i\sqrt{j}}G_aR_b\right]+ \\ & V_{P_h}\left[G_a^2+G_aG_b\right]+V_{P_h}\left[G_aG_b+G_b^2\right]+ \\ & V_{P_x}\left[\frac{Y_1Y_2}{ij}G_aG_b+\frac{{Y_2}^2}{j^2}G_b^2-\frac{X_1Y_2}{j\sqrt{i}}G_bR_a-\frac{X_2Y_2}{j\sqrt{j}}G_bR_b\right]+ \\ & V_{P_y}\left[\frac{X_1X_2}{ij}G_aG_b+\frac{{X_2}^2}{j^2}G_b^2-\frac{X_2Y_1}{j\sqrt{i}}G_bR_a-\frac{X_2Y_2}{j\sqrt{j}}G_bR_b\right]+ \\ & V_{P_x}\left[\frac{-X_1Y_1}{i\sqrt{i}}G_aR_a-\frac{X_1Y_2}{j\sqrt{i}}G_bR_a+\frac{{X_1}^2}{i}R_a^2+\frac{X_1X_2}{\sqrt{ij}}R_aR_b\right]+ \\ & V_{P_y}\left[\frac{-X_1Y_1}{i\sqrt{i}}G_aR_a-\frac{X_2Y_1}{j\sqrt{i}}G_bR_a+\frac{{Y_1}^2}{i}R_a^2+\frac{Y_1Y_2}{\sqrt{ij}}R_aR_b\right]+ \\ & V_{P_x}\left[\frac{X_2Y_1}{i\sqrt{j}}G_aR_b-\frac{X_2Y_2}{j\sqrt{j}}G_bR_b+\frac{X_1X_2}{\sqrt{ij}}R_aR_b+\frac{{X_2}^2}{j}R_b^2\right]+ \\ & V_{P_y}\left[\frac{X_1Y_2}{i\sqrt{j}}G_aR_b-\frac{X_2Y_2}{j\sqrt{j}}G_bR_b+\frac{Y_1Y_2}{\sqrt{ij}}R_aR_b+\frac{{Y_2}^2}{j}R_b^2\right]+ \end{aligned} \tag{A.7}$$

And simplified to provide our solution for the variance of θ given errors on the position and orientation of the robot, V_{P_x}, V_{P_y}, V_{P_h}.

$$\begin{aligned} V_\theta &= V_{P_h} + \\ & V_{P_x}\left[\left(\frac{G_a Y_1}{i} + \frac{G_b Y_2}{j} - \frac{R_a X_1}{\sqrt{i}} - \frac{R_b X_2}{\sqrt{j}}\right)^2\right] + \\ & V_{P_y}\left[\left(\frac{G_a X_1}{i} + \frac{G_b X_2}{j} - \frac{R_a Y_1}{\sqrt{i}} - \frac{R_b Y_2}{\sqrt{j}}\right)^2\right] \end{aligned} \tag{A.8}$$

And G_a, G_b, R_a and R_b are the derivatives of θ with respect to γ_1, γ_2, R_1, and R_2 respectively.

When $R_1 = R_2$,

$$\theta = \frac{\gamma_1 + \gamma_2}{2} \tag{A.9}$$

therefore

$$G_a = G_b = 1/2 \qquad R_a = R_b = 0 \tag{A.10}$$

and the solution simplifies to

$$V_\theta = V_{P_h} + V_{P_x}\left(\frac{Y_1}{2i} + \frac{Y_2}{2j}\right)^2 + V_{P_y}\left(\frac{X_1}{2i} + \frac{X_2}{2j}\right)^2 \tag{A.11}$$

A.2 Calculation of Variance of θ given Errors on Corner Positions

The same co-ordinate frame is used from figure A.1, so that the same definitions for R_1, R_2, γ_1 and γ_2 exist.

With the addition of errors on the corner positions, the covariance matrix of γ_1, γ_2, R_1 and R_2 with respect to P_x, P_y, P_h, $P_{C_{1x}}$, $P_{C_{1y}}$, $P_{C_{2x}}$ and $P_{C_{2y}}$ is calculated from

$$\sigma_\theta{}^2 = \frac{\partial \theta}{\partial S} cov(S) \frac{\partial \theta}{\partial S}^T \quad \text{where} \quad S = [\ \gamma_1 \quad \gamma_2 \quad R_1 \quad R_2\ ]^T \tag{A.12}$$

$$cov(S) = \frac{\partial S}{\partial M} cov(M) \frac{\partial S}{\partial M}^T \quad \text{where} \quad M = [\ P_x \quad P_y \quad P_h \quad P_{C_{1x}} \quad P_{C_{1y}} \quad P_{C_{2x}} \quad P_{C_{2y}}\ ]^T \tag{A.13}$$

Using the same definitions for R and γ of figure A.1,

$$\frac{\partial \gamma}{\partial C_x} = \frac{-1}{1+\frac{X^2}{Y^2}} \cdot \frac{1}{Y} \qquad \frac{\partial R}{\partial C_x} = \frac{1}{2}[X^2+Y^2]^{-\frac{1}{2}} \cdot 2X \cdot 1$$

$$= \frac{-Y}{X^2+Y^2} \qquad = \frac{X}{\sqrt{X^2+Y^2}}$$

$$\frac{\partial \gamma}{\partial C_y} = \frac{-1}{1+\frac{X^2}{Y^2}} \cdot \frac{-X}{Y^2} \qquad \frac{\partial R}{\partial C_y} = \frac{1}{2}[X^2+Y^2]^{-\frac{1}{2}} \cdot 2Y \cdot 1$$

$$= \frac{X}{X^2+Y^2} \qquad = \frac{Y}{\sqrt{X^2+Y^2}}$$

Again we assume independent errors on the input information

$$cov(M) = \begin{pmatrix} V_{P_x} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & V_{P_y} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & V_{P_h} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & V_{C_{1x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & V_{C_{1y}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & V_{C_{2x}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & V_{C_{2y}} \end{pmatrix} \tag{A.14}$$

Where $V_a = {\sigma_a}^2$.

The covariance matrix of γ_1, γ_2, R_1 and R_2 with respect to P_x, P_y, P_h $P_{C_{1_x}}$ $P_{C_{1_y}}$ $P_{C_{2_x}}$ $P_{C_{2_y}}$ is

$$cov(S) = cov(S_1)cov(M)cov(S_1)^T$$

where

$$\begin{pmatrix} \frac{Y_1}{i} & \frac{-X_1}{i} & 1 & \frac{-Y_1}{i} & \frac{X_1}{i} & 0 & 0 \\ \frac{Y_2}{j} & \frac{-X_2}{j} & 1 & 0 & 0 & \frac{-Y_2}{j} & \frac{X_2}{j} \\ \frac{-X_1}{\sqrt{i}} & \frac{-Y_1}{\sqrt{i}} & 0 & \frac{X_1}{\sqrt{i}} & \frac{Y_1}{\sqrt{i}} & 0 & 0 \\ \frac{-X_2}{\sqrt{j}} & \frac{-Y_2}{\sqrt{j}} & 0 & 0 & 0 & \frac{X_2}{\sqrt{j}} & \frac{Y_2}{\sqrt{j}} \end{pmatrix}.$$

$$= \begin{pmatrix} \frac{Y_1 V_{P_x}}{i} & \frac{-X_1 V_{P_y}}{i} & V_{P_h} & \frac{-Y_1 V_{C_{1_x}}}{i} & \frac{X_1 V_{C_{1_y}}}{i} & 0 & 0 \\ \frac{Y_2 V_{P_x}}{j} & \frac{-X_2 V_{P_y}}{j} & V_{P_h} & 0 & 0 & \frac{-Y_2 V_{C_{2_x}}}{j} & \frac{X_2 V_{C_{2_y}}}{j} \\ \frac{-X_1 V_{P_x}}{\sqrt{i}} & \frac{-Y_1 V_{P_y}}{\sqrt{i}} & 0 & \frac{X_1 V_{C_{1_x}}}{\sqrt{i}} & \frac{Y_1 V_{C_{1_y}}}{\sqrt{i}} & 0 & 0 \\ \frac{-X_2 V_{P_x}}{\sqrt{j}} & \frac{-Y_2 V_{P_y}}{\sqrt{j}} & 0 & 0 & 0 & \frac{X_2 V_{C_{2_x}}}{\sqrt{j}} & \frac{Y_2 V_{C_{2_y}}}{\sqrt{j}} \end{pmatrix} \cdot cov(S_1)^T$$

Which is equal to the covariance matrix of A.4 in addition to

$$\begin{pmatrix} \frac{Y_1^2 V_{C_{1_x}} + X_1^2 V_{C_{1_y}}}{i^2} & - & - & - \\ \frac{V_{C_{2_x}} Y_2^2 + V_{C_{2_y}} X_2^2}{j^2} & \frac{Y_2^2 V_{C_{2_x}} + X_2^2 V_{C_{2_y}}}{j^2} & - & - \\ \frac{X_1 Y_1 (V_{C_{1_y}} - V_{C_{1_x}})}{i\sqrt{i}} & 0 & \frac{X_1^2 V_{C_{1_x}} + Y_1^2 V_{C_{1_y}}}{i} & - \\ 0 & \frac{X_2 Y_2 (V_{C_{2_y}} - V_{C_{2_x}})}{j\sqrt{j}} & 0 & \frac{X_2^2 V_{C_{2_x}} + Y_2^2 V_{C_{2_y}}}{j} \end{pmatrix} \tag{A.15}$$

where –– is again used for clarity in place of the diagonally symmetric counterpart.

We can now formulate variance of θ for robot and corner errors using A.6 such that

$$\begin{aligned} V_\theta \quad &= V_{P_h} + \\ &V_{P_x}\left[\left(\frac{G_a Y_1}{i} + \frac{G_b Y_2}{j} - \frac{R_a X_1}{\sqrt{i}} - \frac{R_b X_2}{\sqrt{j}}\right)^2\right] + \\ &V_{P_y}\left[\left(\frac{G_a X_1}{i} + \frac{G_b X_2}{j} - \frac{R_a Y_1}{\sqrt{i}} - \frac{R_b Y_2}{\sqrt{j}}\right)^2\right] + \\ &V_{C_{1x}}\left[\frac{Y_1^2}{i^2}G_a^2 - \frac{X_1 Y_1}{i\sqrt{i}}G_a R_a\right] + V_{C_{1y}}\left[\frac{X_1^2}{i^2}G_a^2 + \frac{X_1 Y_1}{i\sqrt{i}}G_a R_a\right] + \\ &V_{C_{2x}}\left[\frac{Y_2^2}{j^2}G_b^2 - \frac{X_2 Y_2}{j\sqrt{j}}G_b R_b\right] + V_{C_{2y}}\left[\frac{X_2^2}{j^2}G_b^2 + \frac{X_2 Y_2}{j\sqrt{j}}G_b R_b\right] + \\ &V_{C_{1x}}\left[\frac{-X_1 Y_1}{i\sqrt{i}}G_a R_a + \frac{X_1^2}{i}R_a^2\right] + V_{C_{1y}}\left[\frac{X_1 Y_1}{i\sqrt{i}}G_a R_a + \frac{Y_1^2}{i}R_a^2\right] + \\ &V_{C_{2x}}\left[\frac{-X_2 Y_2}{j\sqrt{j}}G_b R_b + \frac{Y_1^2}{j}R_b^2\right] + V_{C_{2y}}\left[\frac{X_2 Y_2}{j\sqrt{j}}G_b R_b + \frac{Y_2^2}{j}R_b^2\right] \end{aligned} \tag{A.16}$$

We therefore find our solution for the variance of θ given errors on the postition and orientation of the robot in addition to errors on the corner positions that form the opening (V_{P_x}, V_{P_y}, V_{P_h},$V_{P_{C_{1x}}}$, $V_{P_{C_{1y}}}$, $V_{P_{C_{2x}}}$ and $V_{P_{C_{2y}}}$.

$$\begin{aligned} \Rightarrow V_\theta \quad &= V_{P_h} + \\ &V_{P_x}\left[\left(\frac{G_a Y_1}{i} + \frac{G_b Y_2}{j} - \frac{R_a X_1}{\sqrt{i}} - \frac{R_b X_2}{\sqrt{j}}\right)^2\right] + \\ &V_{P_y}\left[\left(\frac{G_a X_1}{i} + \frac{G_b X_2}{j} - \frac{R_a Y_1}{\sqrt{i}} - \frac{R_b Y_2}{\sqrt{j}}\right)^2\right] + \\ &V_{C_{1x}}\left(\frac{Y_1 G_a}{i} - \frac{X_1 R_a}{\sqrt{i}}\right)^2 + V_{C_{1y}}\left(\frac{X_1 G_a}{i} + \frac{Y_1 R_a}{\sqrt{i}}\right)^2 + \\ &V_{C_{2x}}\left(\frac{Y_2 G_b}{j} - \frac{X_2 R_b}{\sqrt{j}}\right)^2 + V_{C_{2y}}\left(\frac{X_2 G_b}{j} + \frac{Y_2 R_b}{\sqrt{j}}\right)^2 \end{aligned} \tag{A.17}$$

Again when $R_1 = R_2$, θ is defined in equation A.9 such that

$$G_a = G_b = 1/2 \qquad R_a = R_b = 0$$

$$\begin{aligned} V_\theta \quad &= \quad V_{P_h} + \\ &V_{P_x}\left[\left(\frac{Y_1}{2i} + \frac{Y_2}{2j}\right)^2\right] + V_{P_y}\left[\left(\frac{X_1}{2i} + \frac{X_2}{2j}\right)^2\right] + \\ &V_{C_{1x}}\left(\frac{Y_1}{2i}\right)^2 + V_{C_{1y}}\left(\frac{X_1}{2i}\right)^2 + V_{C_{2x}}\left(\frac{Y_2}{2j}\right)^2 + V_{C_{2y}}\left(\frac{X_2}{2j}\right)^2 \end{aligned} \tag{A.18}$$

Bibliography

[Albers et al., 1999] Albers, S., Kursawe, K., and Schuierer, S. (1999). Exploring unknown environments with obstacles. In *10th Symposium on Discrete Algorithms*, pages 842–843.

[Althaus and Christensen, 2003] Althaus, P. and Christensen, H. I. (2003). Automatic map acquisition for navigation in domestic environments. In *Proc of International Conference on Robotics and Automation*, volume 2, pages 1551–1556.

[Arbel and Ferrie, 1999] Arbel, T. and Ferrie, F. (1999). Viewpoint selection by navigation through entropy maps. *Proc of International Conference on Computer Vision*, pages 248–254.

[Arkin et al., 1993] Arkin, E. M., Fekete, S. P., and Mitchell, J. S. B. (1993). The lawnmower problem. *5th Canadian Conference on Computational Geometry*, pages 461–466.

[Ayache and Faugeras, 1989] Ayache, N. and Faugeras, O. D. (1989). Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819.

[Bar-Shalom and Li, 1993] Bar-Shalom, Y. and Li, X.-R. (1993). *Estimation and Tracking: Principles, Techniques and Software*. Artech House, Inc.

[Bhattacharyya, 1948] Bhattacharyya, A. (1948). On some analogues of the amount of information and their use in statistical estimation. *The Indian Journal of Statistics*, 8(4):315–328.

[Borenstein and Koren, 1988] Borenstein, J. and Koren, Y. (1988). Obstacle avoidance with ultrasonic sensors. *Journal on Robotics and Automation*, 4(2):213–218.

[Borenstein and Koren, 1989] Borenstein, J. and Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187.

[Borenstein and Koren, 1991] Borenstein, J. and Koren, Y. (1991). The vector field histogram: Fast obstacle avoidance for mobile robots. *Journal on Robotics and Automation*, 7:278–288.

[Briggs and Donald, 2000] Briggs, A. and Donald, B. (2000). Visibility-based planning of sensor control strategies. *Algorithmica*, 26:364–388.

[Brooks, 1983] Brooks, R. A. (1983). Solving the findpath problem by good representation of free space. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3):190–233.

[Brooks, 1986] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47(1-3):139–159.

[Burgard et al., 2000] Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2000). Collaborative multi-robot exploration. In *Proc of International Conference on Robotics and Automation*, volume 1, pages 476–481.

[Cameron and Durrant-Whyte, 1990] Cameron, A. J. and Durrant-Whyte, H. (1990). A bayesian approach to optimal sensor placement. *International Journal of Robotics Research*, 9(5):70–88.

[Cartwright and Collett, 1983] Cartwright, B. A. and Collett, T. S. (1983). Landmark learning in bees. *Journal of Comparative Physiology A*, 151:521–543.

[Chaimowicz et al., 2003] Chaimowicz, L., Campos, M. F. M., and Kumar, V. (2003). Hybrid systems modeling of cooperative robots. In *Proc of International Conference on Robotics and Automation*, volume 3, pages 4086–4091.

[Chvátal, 1975] Chvátal, V. (1975). A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory*, 18:39–41.

[Collins and Tsin, 1998] Collins, R. and Tsin, Y. (1998). Calibration of an outdoor active camera system. Technical Report CMU-RITR-98-36, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

[Comaniciu and Meer, 1997] Comaniciu, D. and Meer, P. (1997). Robust analysis of feature spaces: Color image segmentation. In *IEEE Computer Vision and Pattern Recognition*, pages 750–755.

[Comaniciu and Meer, 1999] Comaniciu, D. and Meer, P. (1999). Mean shift analysis and applications. In *International Conference on Computer Vision*, pages 1197–1203.

[Connolly, 1985] Connolly, C. (1985). The determination of next best views. *Proc of International Conference on Robotics and Automation*, pages 432–435.

[Connolly et al., 1990] Connolly, C., Burns, J., and Weiss, R. (1990). Path planning using laplace's equation. *Proc of International Conference on Robotics and Automation*, pages 2102–2106.

[Cowan and Kovesi, 1988] Cowan, C. and Kovesi, P. (1988). Automatic sensor placement from vision task requirements. *Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416.

[Davison and Murray, 2002] Davison, A. J. and Murray, D. W. (2002). Simultaneous localization and map-building using active vision. *Transactions on Pattern Analysis and Machine Intelligence*, 24(7):865–880.

[Deng et al., 1996] Deng, X., Milios, E., and Mirzaian, A. (1996). Landmark selection strategies for path execution. *Journal of Robotics and Autonomous Systems*, 17(3):171–185.

[Dissanayake et al., 2001] Dissanayake, M. W. M. G., Newman, P. M., Durrant-Whyte, H. F., Clark, S., , and Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

[Duckett and Nehmzow, 1997] Duckett, T. and Nehmzow, U. (1997). Experiments in evidence based localisation for a mobile robot. In *Proceedings of the AISB Workshop on Spatial Reasoning in Animals and Robots*.

[Faugeras, 1993] Faugeras, O. (1993). *Three-Dimensional Computer Vision*. The MIT Press.

[Folkesson and Christensen, 2003] Folkesson, J. and Christensen, K. (2003). Outdoor exploration and slam using a compressed filter. In *Proc of International Conference on Robotics and Automation*, pages 419–426.

[Foux et al., 1993] Foux, G., Heymann, M., and Bruckstein, A. (1993). Two-dimensional robot navigation among unknown stationary polygonal obstacles. *IEEE Transactions on Robotics and Automation*, 9(1):96–101.

[Fukunaga, 1990] Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*, pages 99–100. Computer Science and Scientific Computing. Academic Press.

[Gartshore et al., 2002] Gartshore, R., Aguado, A., and Galambos, C. (2002). Incremental map building using an occupancy grid for an autonomous monocular robot. In *The 7th International Conference on Control, Automation, Robotics and Vision*, pages 613–618.

[Gartshore and Palmer, 2005] Gartshore, R. and Palmer, P. (2005). Exploration of an unknown 2D environment using a view improvement strategy. In *Towards Autonomous Robotic Systems*, pages 57–64. London, UK.

[Gartshore et al., 2005] Gartshore, R., Palmer, P., and Illingworth, J. (2005). A novel exploration algorithm based on a improvement strategy. *International Journal of Advanced Robotic Systems*, 2(4):287–294.

[Gonzalez and Woods, 1992] Gonzalez, R. C. and Woods, R. E. (1992). *Digital Image Processing*. Addison-Wesley Publishing Company.

[Grabowski et al., 2003] Grabowski, R., Khosla, P., and Choset, H. (2003). Autonomous exploration via regions of interest. In *IEEE Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1691–1696.

[Guibas and Stolfi, 1985] Guibas, L. and Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123.

[http://www.activmedia.com, 2000] http://www.activmedia.com (2000). Activmedia robotics.

[Huang and Beevers, 2005] Huang, W. H. and Beevers, K. R. (2005). Topological map merging. *International Journal of Robotics Research*, 24(8):601–613.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98.

[Kirkpatrick, 1979] Kirkpatrick, D. (1979). Efficient computations of continuous skeletons. In *Annual Symposium on Foundations of Computer Science*, pages 18–27.

[Kutulakos et al., 1994a] Kutulakos, K., Brent Seales, W., and Dyer, C. (1994a). Building global object models by purposive viewpoint control. In *Proc. 2nd CAD-based Vision Workshop*, pages 169–182.

[Kutulakos et al., 1994b] Kutulakos, K., Dyer, C., and Lumelsky, V. (1994b). Provable strategies for vision-guided exploration in three dimensions. In *Proc of International Conference on Robotics and Automation*, volume 2, pages 1365–1372.

[Kutulakos et al., 1993] Kutulakos, K., Lumelsky, V., and Dyer, C. (1993). Vision-guided exploration: A step toward general motion planning in three dimensions. *Proc of International Conference on Robotics and Automation*, pages 289–296.

[Kutulakos and Dyer, 1994] Kutulakos, K. N. and Dyer, C. R. (1994). Recovering shape by purposive viewpoint adjustment. *International Journal of Computer Vision*, pages 113–136.

[Lacroix et al., 1992] Lacroix, S., Grandjean, P., and Ghallab, M. (1992). Perception planning for a multisensory interpretation machine. *Proc of International Conference on Robotics and Automation*, pages 1818–1824.

[Langer et al., 1994] Langer, D., Rosenblatt, J., and Hebert, M. (1994). A behavior based system for off road navigation. *IEEE Transactions on Robotics and Automation*, 10(6):776–782.

[Laubach et al., 1999] Laubach, S. L., Olsen, C. F., Burdick, J. W., and Hayati, S. (1999). Long range navigation for mars rovers using sensor-based path planning and visual localisation. In *5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

[Lazanas and Latombe, 1992] Lazanas, A. and Latombe, J.-C. (1992). Landmark-based robot navigation. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 816–822. AAAI Press.

[Lee and Recce, 1994] Lee, D. and Recce, M. (1994). Quantitative evaluation of the exploration strategies of an intelligent vehicle. In *Proceedings of the Intelligent Vehicles Symposium*, pages 538–543.

[Lewin et al., 1998] Lewin, M., Bowden, R., and Sarhadi, M. (1998). Tools and techniques for camera calibration. VRR Technical Report, Dept M & ES, Brunel University.

[Lorigo et al., 1997] Lorigo, L., Brooks, R., and Grimson, W. (1997). Visually-guided obstacle avoidance in unstructured environments. *Proc of IEEE Conference on Intelligent Robots and Systems*, pages 373–379.

[Lozano-Pérez, 1983] Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computing*, C-32:108–120.

[Lumelsky and Stepanov, 1987] Lumelsky, V. J. and Stepanov, A. A. (1987). Path-planning strategies for a point mobile automation moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430.

[Manessis, 2001] Manessis, A. (2001). *3D Reconstruction from Video Using a Mobile Robot*. PhD thesis, University of Surrey, UK.

[Maver and Bajcsy, 1993] Maver, J. and Bajcsy, R. (1993). Occlusions as a guide for planning the next view. *Transactions on Pattern Analysis and Machine Intelligence*, 15(5):417–432.

[Murray et al., 1996] Murray, D. W., Reid, I. D., and Davison, A. J. (1996). Steering and navigating behaviours using fixation. *British Machine Vision Conference*, pages 635–644.

[Nagatini and Choset, 1999] Nagatini, K. and Choset, H. (1999). Toward robust sensor based exploration by constructing reduced generalized Voronoi graphs. *Proc of IEEE Conference on Intelligent Robots and Systems*, pages 1687–1692.

[Nehmzow and Owen, 2000] Nehmzow, U. and Owen, C. (2000). Robot navigation in the real world: Experiments with manchester's *fortytwo* in unmodified, large environments. *Journal of Robotics and Autonomous Systems*, 33:223–242.

[Nelson and Khosla, 1994] Nelson, B. and Khosla, P. (1994). Integrating sensor placement and visual tracking strategies. *Proc of International Conference on Robotics and Automation*, pages 1351–1356.

[Newman et al., 2003] Newman, P., Bosse, M., and Leonard, J. (2003). Autonomous feature-based exploration. In *Proc of International Conference on Robotics and Automation*, volume 1, pages 1234–1240.

[O'Dunlaing and Yap, 1985] O'Dunlaing, C. and Yap, C. (1985). A "retraction" method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111.

[Oommen et al., 1987] Oommen, B., Iyengar, S., Rao, N., and Kashyap, R. (1987). Robot navigation in unknown terrains using learned visibility graphs. part 1: The disjoint convex obstacle case. *IEEE Journal on Robotics and Automation*, RA-3(6):672–681.

[O'Rourke, 1994] O'Rourke, J. (1994). *Computational Geometry in C*. Cambridge University Press.

[Patel et al., 2002] Patel, N., Ellery, A., Welch, C., and Curley, A. (2002). Preliminary analysis of mobility and suspension systems for a mars micro rover. In *2nd World Space Congress*.

[Petrou and Bosdogianni, 1999] Petrou, M. and Bosdogianni, P. (1999). *Image Processing. The Fundamentals.* John Wiley & Sons, Ltd.

[Pirjanian and Christensen, 1997] Pirjanian, P. and Christensen, H. I. (1997). Behavior coordination using multiple-objective decision making. In Schenker, P. S. and McKee, G. T., editors, *Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, volume 3209, pages 78–89. SPIE - The International Society for Optical Engineering.

[Pirjanian et al., 1997] Pirjanian, P., Christensen, H. I., and Fayman, J. A. (1997). Experimental investigation of voting schemes for fusion of redundant purposive modules. *Fifth Symposium for Intelligent Robotics Systems*, pages 131–138.

[Pito, 1996] Pito, R. (1996). A sensor based solution to the next best view problem. In *International Conference on Pattern Recognition*, volume 1, pages 941 – 945.

[Polesel et al., 2000] Polesel, R., Rosati, R., Speranzon, A., Ferrari, C., and Pagello, E. (2000). Using collision avoidance algorithms for designing multi-robot emergent behaviors. In *Proc of IEEE Conference on Intelligent Robots and Systems*, volume 2, pages 1403–1409.

[Rao and Iyengar, 1990] Rao, N. and Iyengar, S. (1990). Autonomous robot navigation in unknown terrains: Incidental learning and environmental explorations. *IEEE Transactions on Systems, Man and Cybernetics*, 20(6):1443–1449.

[Rao et al., 1991] Rao, N., Stoltzfus, N., and Iyengar, S. (1991). A "retraction" method for learned navigation in unknown terrains for a circular robot. *IEEE Transactions on Robotics and Automation*, 7(5):699–707.

[Russell and Norvig, 1995] Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence. A Modern Approach*, chapter 4. Prentice-Hall International, Inc.

[Schiele and Crowley, 1994] Schiele, B. and Crowley, J. (1994). Comparison of position estimation techniques using occupancy grids. In *Proc of International Conference on Robotics and Automation*, pages 1628–1634.

[Sim and Dudek, 2003] Sim, R. and Dudek, G. (2003). Effective exploration strategies for the construction of visual maps. In *Proc of IEEE Conference on Intelligent Robots and Systems*, volume 3, pages 3224–3231.

[Stentz, 1995] Stentz, A. (1995). Optimal and efficient path planning for unknown and dynamic environments. *International Journal on Robotics and Automation*, 10(3):89–100.

[Thrun, 2002] Thrun, S. (2002). Robotic mapping: A survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann.

[Trullier et al., 1997] Trullier, O., Wiener, S., Berthoz, A., and JA, M. (1997). Biologically based artificial navigation systems: review and prospects. *Progress in Neurobiology*, 51(5):483–544.

[Ulrich and Borenstein, 2000] Ulrich, I. and Borenstein, J. (2000). VFH*: Local obstacle avoidance with look-ahead verification. In *International Conference on Robotics and Automation*, pages 2505–2511.

[Wan and Xu, 1996] Wan, X. and Xu, G. (1996). Camera parameters estimation and evaluation in active vision system. *Pattern Recognition*, 29(3):439–447.

[Whaite and Ferrie, 1991] Whaite, P. and Ferrie, F. (1991). From uncertainty to visual exploration. *Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1038–1049.

[Willson and Shafer, 1993] Willson, R. and Shafer, S. (October 1993). A perspective projection camera model for zoom lenses. In *Proc. 2nd Conf. on Optical 3-D Measurement Techniques*.

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151.

[Yamauchi et al., 1998] Yamauchi, B., Schultz, A., and Adams, W. (1998). Mobile robot exploration and map-building with continuous localization. In *Proc of International Conference on Robotics and Automation*, volume 4, pages 3715–3720.

[Youngblood et al., 2000] Youngblood, G. M., Holder, L. B., and Cook, D. J. (2000). A framework for autonomous mobile robot exploration and map learning through the use of place-centric occupancy grids. In *Proceedings of the Machine Learning Workshop on Learning from Spatial Information.*