EXTENDING THE PATH-PLANNING HORIZON

Bart Nabbe

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

July 2005

Submitted in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy

Thesis Committee:
Martial Hebert, Chair
Howie Choset
Tony Stentz
Vijay Kumar, University of Pennsylvania

ABSTRACT

SINCE typical mobile robotic vehicles have mobility sensors (such as LADAR or stereo) that can only acquire data up to a few tens of meters, a navigation system has no knowledge about the world beyond this sensing horizon. As a result, path planners that rely only on this knowledge to compute paths are unable to anticipate obstacles sufficiently early and has no choice than to plan inefficient paths that trace obstacle boundaries. To alleviate this problem, We present an opportunistic navigation and view planning strategy that incorporates look-ahead sensing of possible obstacle configurations.

This planning strategy is based on a "what-if" analysis of hypothetical future configurations of the environment. Candidate vantage positions are evaluated based on their ability of observing anticipated obstacles. These vantage positions identified by this forward-simulation framework are used by the planner as intermediate waypoints.

The validity of the strategy is supported by results from simulations as well as field experiments with a real robotic platform. These results also show that opportunistically significant reduction in path length can be achieved by using this framework.

TABLE OF CONTENTS

ABSTRACT iii

LIST OF FIGURES vii

CHAPTER 1. Introduction 1
1. Problem Components 2
1.1. Navigation in Unstructured Environments 3
1.2. Mid-range Sensing 4
1.3. Path Planning 5
2. Proposed Solution 6
3. Contributions 8

CHAPTER 2. Prior Work 9
1. Path Planning for Ground Vehicles 9
2. View planning 11
3. Sensor Based Path Planning 15
4. Planning with Uncertainty 17
5. Remarks 21

CHAPTER 3. Mid-range Sensing and Planning Algorithm 23
1. Planning a Path 25
2. Forward Simulation 26
3. Computational Complexity 27
4. Conclusion 28

CHAPTER 4. Hallucinating Worlds 29
1. Hallucinating 29
2. Data Term 31
3. Field Model 32
4. Inference 33
5. Conclusion 35

CHAPTER 5. Where to Look 37
1. Expected Path Costs 38
2. Restricting Sensor Orientation 39
2.1. Sampling from the Field Model 39
2.2. Interesting Areas 40
2.3. Considering the Topological Map 43
3. Summary 45

CHAPTER 6. System Integration 47
1. Sensor Model 47
2. Computing Navigation Maps 48
3. Earliest Observation Position 50
4. System Overview 50

CHAPTER 7. Empirical Evaluation 59
1. Example Experiments with Laser Range Finder and Wide-baseline Stereo 60
1.1. Experiment with Wide-baseline Stereo 60
1.2. Experiment with Laser Range Finder 63
1.3. Discussion 64
2. Detailed Analysis 67
2.1. Overview 67
2.2. Analysis 68
2.3. Discussion 84

CHAPTER 8. Discussion 85

APPENDIX A. Wide baseline stereo for unstructured environments 87
1. Algorithm Description 89
2. Experimental Results 98

REFERENCES 101

LIST OF FIGURES

1.1 Typical example of poor performance due to lack of sensor planning and mid-range sensing. 2
1.2 A panoramic view of natural unstructured terrain. 3
1.3 The General Dynamics XUV autonomous vehicle for Unmanned Ground Navigation. 3
1.4 A typical sparse 3D wide-baseline stereo reconstruction from an unstructured environment. 4
1.5 The difference between the two path lengths. 6
1.6 The most likely inferred world configuration. 7

2.1 NAVLAB (a) navigating an unstructured area (b). 10
2.2 Pacman lives in a MDP maze. 18

3.1 Poor performance due to lack of sensor planning. 23
3.2 The partial map in (a) suggests a gap between the two obstacle regions. A large deviation from the path will be incurred if the gap is actually blocked (b). 24
3.3 Late detection of the obstacle results in an expensive detour. . 25
3.4 Illustration of the path computed with (C_s) and without sensing (C_{ns}) in an example world configuration. 26
3.5 The utility map displayed for a very simple partially known world. 27

4.1 Example showing the influence of the obstacle hallucination. . 33
4.2 Example demonstrating the influence of the obstacle hallucination over the course of a typical traverse. 34

5.1 The relation between θ and *point of interest* (POI). 37
5.2 Alternative traverses depending on the interpretation of inferred obstacle. 38
5.3 Illustration showing the relation between an interesting area (marked unknown) and observation location (in magenta). . . 40
5.4 Interest response from a comb style obstacle. 41

5.5 The measure of interest due to expected traversal for an example path and $\eta = 10$ 42

5.6 All the paths marked in blue are discarded for evaluation, since we use only one singe representative per homotopic class (marked in green). 43

5.7 The location of the most interesting point changes slightly for different paths in the same homotopic class. 44

6.1 Sensor models for the mobility sensor and mid-range sensor. . 48

6.2 Mid-range sensor measurements and their expansions. 49

6.3 Example robot navigation scenario and the corresponding utility map. 50

6.4 Mid-range planning sensing algorithm overview. 52

6.6 Step by step view point planning example. 56

6.7 Data display utility showing a typical candidate viewpoint evaluation. 57

6.8 A radial scan line sweeps a binary obstacle map. 58

7.1 The Pioneer mobile robot as used in a wide-baseline stereo experiment. 61

7.2 Experimental scenario. 61

7.3 a birds-eye view of the example test scenario. 62

7.4 The robotic Deere eGator use in a laser range finder experiment. 63

7.5 Example experiment "On the Cut": scenario. 64

7.6 Example experiment "On the Cut": internal maps. 65

7.7 Example experiment "In Shenley park": scenario. 65

7.8 Example experiment "In Shenley park": internal maps. 66

7.9 Top: a panorama view of the open coal mine test site from which an elevation map was collected. Bottom: a rendering of this elevation map. 69

7.10 The Mid-range planning sensing algorithm compared to the standard navigation approach (example 1). 70

7.11 Bad performance is due to the fact that little gain can be expected if the area between the start and goal points is completely unobstructed. 71

7.12 A random sample of runs drawn from the set of 500. 71

7.13 The Mid-range planning sensing algorithm compared to a continuous sensing method with an unlimited range, 360° view range finder. and also compared to the true shortest path ($\circ$) (example 1). . . 72

7.14 Evaluation of the timeout parameter. 73

7.15 Evaluation of the neighborhood interaction term β. 75

7.16 Obstacle inference as a function of the neighborhood interaction term β. 75

7.17 A rendering of the USGS elevation map from a terrain patch in the vicinity of Albuquerque, NM. 76
7.18 The Mid-range planning sensing algorithm compared to the standard navigation approach (example 2). 77
7.20 The Mid-range planning sensing algorithm compared to a continuous sensing method with an unlimited range, 360° view range finder (example 2). 78
7.21 The Mid-range planning sensing algorithm compared to the true shortest path ($\circ$) (example 2). 79
7.22 A rendering of the USGS elevation map from a terrain patch in the vicinity of Flagstaff,AZ. 80
7.23 The Mid-range planning sensing algorithm compared to the standard navigation approach (example 3). 81
7.24 The Mid-range planning sensing algorithm compared to a continuous sensing method with an unlimited range, 360° view range finder (example 3). 82
7.25 The Mid-range planning sensing algorithm compared to the true shortest path ($\circ$) (example 3). 83

A.1 A few typical outdoor scenes. 88
A.2 General approach to finding matching regions. 90
A.3 Lines extracted from an image (a), a hypothesized line grouping (b) and the reference normalized patch (c) 92
A.4 a typical segment from an image segmentation (a), the principal axis (b) and the normalized reference patch (c). 93
A.5 Unfiltered matches (top), candidate matches after filtering based on feature vectors and normalized patch similarity (bottom). . 95
A.6 The reconstructed epipolar geometry from the man-made structured environment. 99
A.7 The reconstructed epipolar geometry from the more natural environment. 100
A.8 The reconstructed epipolar geometry from the very wide baseline 100
A.9 The ratio of the height over the width of the window on the left is 1.5456 and the window on the right 1.8356, which gives us an indication of the reconstruction quality. 100

CHAPTER 1

Introduction

At the core of many autonomous robotics systems is a mobility system that takes data from sensors as input, reconstructs the 3-D geometry of the terrain around the vehicle, assesses the drivability of the terrain, detects obstacle regions, and modifies its currently planned path to avoid newly discovered non-drivable areas. The cycle is repeated many times a second until the vehicle reaches its goal destination. Typically, such a system is implemented by maintaining a representation of the world in the form of a discrete 2.5D grid which is used for planning.

Irrespective of the implementation details of such mobile robot systems, their performance is always severely limited by the so-called myopic planning effect (Figure 1.1);

This effect is due to the fact that the planner is limited by the maximum range of the mobility sensors. Since typical mobility sensors, such as Laser Radar (LADAR) or passive stereo vision, will only acquire data up to a few tens of meters, the planner has no knowledge about what to encounter beyond the sensed perimeter. As a result, the planner is unable to anticipate obstacles sufficiently early and has no choice but to plan paths close to obstacle boundaries.

For autonomous navigation over long distances, this issue degrades the performance of the system by greatly increasing the length of the path traveled by the vehicle. Consequently, the power consumed is increased and, more importantly,

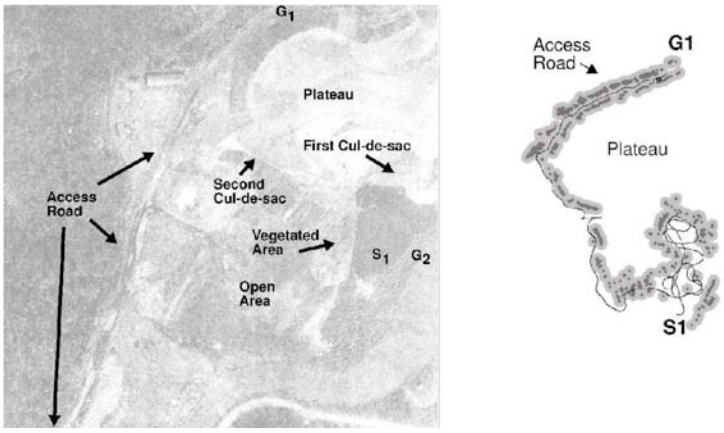


FIGURE 1.1. Typical example of poor performance due to lack of sensor planning and mid-range sensing (Left: Overhead view of terrain; Right: Executed path with detected obstacles shown as shaded regions). The path from S1 to G1 intersects a large hill which is discovered only when the vehicle enters a large cul-de-sac, causing the executed path to be substantially more expensive than the path that would have been followed, had the obstruction been discovered earlier. (From [114])

the risk of exposure to threats is also increased. In addition, the relative short range of the mobility sensors forces the vehicle to drive closer to terrain obstructions than is safe or necessary.

One solution is to use sensors with longer range to acquire information about the terrain further ahead. However, to use these sensors one needs to take into account certain constraints about the sensing geometry acquisition method. We will present in the Chapters 3 to 6 a method that plans for taking these range measurements to allow for the path planning algorithm to compute a better informed path.

1. Problem Components

Before attempting to devise an algorithm that can successfully navigate a mobile vehicle in an unstructured environment, we will describe the problem components in more detail. A logical division of these components includes first the aspects of navigating in unstructured terrain, second the effects of the sensing method used and third the influence of the planning strategy applicable.

1.1. Navigation in Unstructured Environments

FIGURE 1.2. A panoramic view of natural unstructured terrain. It features tree lines as a typical example of a natural obstacle boundary.

In contrast to path planning in man-made environments, the navigation problem in unstructured environments is more complicated since, the autonomous system cannot rely on well-defined structures such as roads, buildings or corridors and hallways to simplify the navigation task. Unstructured environments (Figure 1.2) are typically sparse and have no well defined layout. Vehicles navigating in these environments cannot rely on the presence of roads and will therefore need to estimate about the approximate global structure of the environment in order to efficiently navigate. Typically these autonomous vehicles are quite capable of negotiating rough terrain (Figure 1.3) and feature an ensemble of sensors to aid in determining the traversability of the terrain.

FIGURE 1.3. The General Dynamics XUV autonomous vehicle for Unmanned Ground Navigation.

The absence of structure means we cannot make assumptions about the environment and makes it necessary for the autonomous vehicle to gather information

about the structure of the environment while traversing it. Having longer range sensor measurements available will help to reason about the topological layout of the environment.

1.2. Mid-range Sensing

Solving the myopic planning problem suggests the use of range sensors with a much longer range than typically used for robot navigation and safeguarding. Among the different types of range sensors, a distinction is often made between active and passive sensing techniques. Active sensing techniques use some sort of transmitter to emit radiation, whose reflection is measured by the sensor. Examples of this type of systems are sonar, Laser Range Finders (LADAR) and conventional RADAR. Examples of passive sensing techniques are fixed baseline stereo [1] and structure from motion [128].

FIGURE 1.4. A typical sparse 3D wide-baseline stereo reconstruction from an unstructured environment.

These sensors are constrained by the resolution and the field of view they operate with. For example, for mid-range laser range finders, a single range measurement is returned. Collecting a useful description of the terrain ahead will therefore require taking many measurements, which is impossible for real time navigation because of the low repetition rate. The wide-baseline stereo suffers from different problems making it even more expensive to use. Typically for a mid-range stereo system, the baseline needs to be a couple of tenths of meters (see also appendix A), which implies that the two images from a stereo pair will have to be collected separately. The planner will need to divert the vehicle to this other observation location

or request this measurement from a cooperating agent [83]. Obviously, this is also a very time consuming operation.

The geometry of obstructions in close proximity of the robot may prevent observing terrain further ahead. The observation location needs to be carefully chosen such that the area of interest is in line of sight (if at all possible).

These sensing limitations affect our navigation and view planning strategy in two ways. The planner:

- needs to be able to handle sparse mid-range sensor data.
- has to schedule intermediate way-points and corresponding viewing directions to take a mid-range measurement.

1.3. Path Planning

Mobile robots traditionally navigate by using the *Sense Plan Act* cycle, in which the robot would process the sensor data typically into an occupancy grid, which is used in turn by a path planner to compute a path. The next waypoint on this path is then handed to the mobility system which will control the actuators such that the robot will reach that location. However, the planners are not restricted to these simple traversability cost maps. In addition, instead of cell traversability one could use a utility function to define cell cost. Utility functions or cost functions have been used in [54,100,101,126] and many others to take into account not only traversability, but also other planning parameters such as vehicle dynamics, or even the size of unexplored regions environment [36,82]. Path planning then becomes heuristic searching on utility maps in order to satisfy the utility objective.

Another common way of integrating multiple planning strategies is a hierarchical approach in which a high level planner reasons about different scenarios and passes down subtasks. The navigation and sensor planning method proposed in this thesis is of this kind.

2. Proposed Solution

With the problem set in this framework, a solution can be outlined to solve the navigation and view planning problem. The solution comprises of two parts. The first part deals with *when and where* to take mid-range sensor measurements. The second part deals with anticipating future obstacles and using the sparse mid-range sensor data.

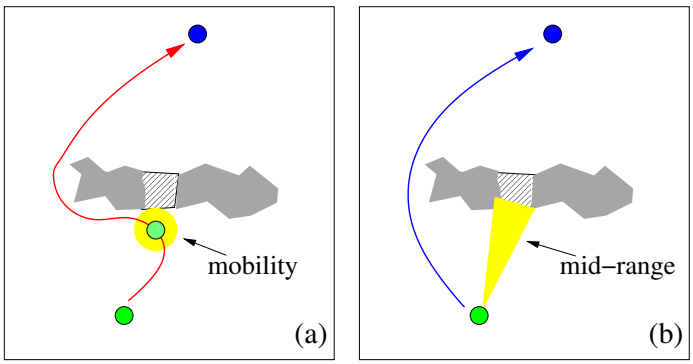


FIGURE 1.5. The difference between the two path lengths in diagram (a) and (b) yields a positive utility from taking an observation. In (a) no mid-range measurement was taken. In (b), a more optimal path was planned due to observing missing data.

A *forward simulation* approach is proposed to evaluate the benefit of going through a scheduled way-point instead of going to the goal immediately. This *"what-if"* analysis considers the additional incurred path length as a measure of detecting an obstacle late rather than early by taking an additional mid-ranging sensor measurement (Figure 1.5). In Chapter 3 we will develop a theoretical framework to compute the benefit (utility) from visiting a sensing location and taking a measurement. The formulation considers all possible future worlds and assumes perfect sensor geometry.

Since such an unlimited range and omni-directional sensor does not exist and because considering all possible worlds is intractable, we propose an approximation that is tractable and allows for the usage of a more typical range finder. The approximation strategy we use reduces the number of worlds to be considered. Our approximation considers only the most anticipated or likely world. To anticipate obstacles to be encountered in the future, we employ an inference mechanism that

uses a prior model of the environment and the current accumulated sensor view of this environment. These anticipated obstacles also allow for grouping sparse mid-range sensor data into more meaningful obstacle regions (Figure 1.6). This inference technique is described in more detail in Chapter 4.

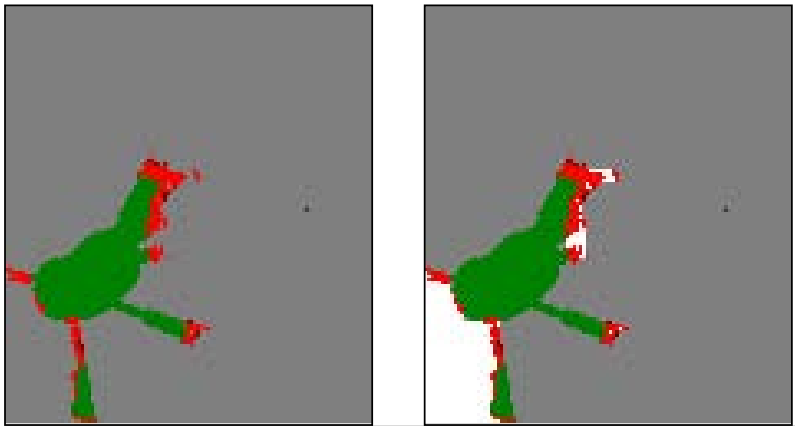


FIGURE 1.6. The inferred world as shown on the right has the inferred obstacles marked in white, the inferred empty space marked gray, observed traversable marked green and observed obstacles marked red. The sensed world is shown on the left for reference. The inferred part captures the most likely configuration of the world based on the current observations.

The theoretical framework from Chapter 3 is adapted in Chapter 5 to incorporate the inferred world model. Using the likelihood model from this inferred world, we show in Chapter 5 how to compute the expected utility from visiting an observation location. We also show in this chapter how we can use the same inferred world to allow the usage of a limited range sensor (limited in range and field of view).

In Chapter 6 we present the components that are necessary in using the forward simulation and inference framework as an online navigation and view planning algorithm. Included are details about the implementation and a system overview. Also presented is an annotated viewpoint planning example that shows step by step how an observation location is planned.

3. Contributions

This thesis presents a navigation and sensor planning strategy that actively uses mid-range sensors to aid a path planner in navigating large unstructured environments. It addresses the typical issues for navigating in large unstructured environments by taking strategically placed mid-range sensor measurements. In addition, the algorithm uses a prior model to anticipate obstacles and navigate accordingly.

The use of *"Hallucinated"* or inferred worlds in combination with probabilistic techniques to plan for view points is new to the field and the key to our navigation and sensor planning strategy.

CHAPTER 2

Prior Work

THIS section reviews the literature that is closely related to the navigation algorithm presented in this thesis. First, we will briefly describe the type of navigation regimes that can be typically found on current outdoor mobile robots. The class of grid-based heuristic path planners is of great importance since the proposed navigation algorithm uses this technique to computate paths. The second section covers some of the view planning concepts for 3D modelling that are related to addressing the question of where we should look to insure we have seen the complete environment. Third, we discuss the research that covers the relationship between sensing and navigating. A class of path planning algorithms is discussed that incorporate a sensing model. We will discuss methods that incorporate uncertainty in the planning framework. Since unknown and partial information will have to be addressed in our problem, this section will also detail a commonly known probabilistic framework (POMDP) that is well suited to capture our navigation problem. We will conclude this section with the relevance of the discussed techniques to the algorithm detailed in the remainder of this thesis.

1. Path Planning for Ground Vehicles

It is challenging to extend path planning algorithms that work well in lab conditions to unstructured environments. These environments are often partly unknown and there is never a well defined geometric structure that can be exploited. Many groups are researching and experimenting in this area. Most notable are

the experiments with the Autonomous Land Vehicle "ALV" [21,22], the Australian Center for Field Robotics High Speed Vehicle "HSV" [3,86] and Carnegie Mellon University's "NAVLAB" [35,44] (Figure 2.1). The groups at the Jet Propulsion Laboratory (JPL) [68,121] and Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) [66,78] have focused on the extraterrestrial application domain of sensing and navigating in unstructured environments. Whereas the groups at Space and Naval Warfare Systems Center (SPAWAR) [41,90], National Institute of Standards and Technology NIST [46,97], SAIC [103], Perceptek [102,104] and also JPL [7] have been looking at navigating large Unmanned Ground Vehicles under one of the many DARPA and ARL programs [112].

(a) (b)

FIGURE 2.1. NAVLAB (a) navigating an unstructured area (b) [44].

Many of these systems follow the classical robotic sense plan act cycle. The sensor acquisition system processes the raw data into an obstacle grid that represents the traversability of the environment. A heuristic search algorithm such as A^* [87,88] would uses this map to plan a path towards the goal. This path would then be handed off to the controller [67] to be executed.

A few other methods have been explored, but these have mostly been applied for local navigation and safeguarding. Cang and Borenstein [57,126] present a local navigation method that computes polar obstacle densities to represent traversability. By computing the histogram of these obstacle densities, a trajectory with low obstacle density can be followed by choosing bins with low obstacle densities.

The same concept was approached by Kelly in [54] a predictive classical control framework. The controller takes the vehicle dynamics and the current vehicle

attitude into account for predicting what is safe to navigate in the immediate future.

Lacaze et al. [62] extend this idea in include a further planning lookahead step. They pre-compute the so called "eco-graph" in which possible vehicle trajectories are evaluated based on vehicle dynamics. Possible waypoints at a fixed lookahead time are removed from the selection if the are not reachable under the current sensed world configuration. This is done successively for discrete multiples of the lookahead distance, creating a thinned eco-graph that has only traversable branches in it. The planning algorithm presented in this thesis takes this idea of lookahead or forward simulation a step further.

Unstructured environments are large in contrast to typical lab setups and methods of efficient planning have been researched in different directions. One approach is to represent the grid at multiple resolutions and to use a coarse-to-fine approach to efficiently compute a path [23, 91]. Another approach to make planning more efficient is to reduce the amount of computation. Stentz [114, 115] reduces the computation time by reusing previous planning results. Another approach is to reduce the search space, Spero [113] uses Rapidly-exploring Random Trees to do so. These approaches are not directly related to work presented in this thesis. However, our algorithm does rely on some of these results to reduce computation time [115].

2. View planning

Most related to our work is the problem of "View Planning" in which we need to decide where to acquire the next sensor reading given certain constraints. Although most of the related work in this area exploits task specific constraints such as a constrained turntable setup that are not valid for our task, it does provide insight into our problem.

The work of Chen [16] and Kleinberg [55] make the common assumption that the world is 2D. They approach the problem from a computational geometry perspective. Their approach to the problem is based on on-line search of polygon.

Kleinberg introduces in [55] rectilinear polygons that allows him to give a bound on the complexity of the view planning problem. Chen considers in [16] any type of polygon, and shows an algorithm for the incremental exploration of the environment. The algorithm that Chen presents is not as efficient as the algorithm described by Kleinberg, since the algorithm is not restricted to a particular type of polygons. Both of these results are interesting because of the provable performance. They both assume that there are no physical sensor limitations, such as range, infinite resolution, sensor noise and field of view restrictions. However, they are still important results since they give some indication about the complexity of the problem.

There is also extensive literature on view planning in the area of map building. For example, the work of [16, 32, 82, 127] with respect to sensing for exploration, focuses on complete exploration of the environment. The robot is driven by a measurement of utility that represents some notion of information gain for exploring certain areas. It does not deal with view planning in the context of deploying a sensing device, but merely deals with visiting locations and ensuring that the complete environment has been observed. Kondo [51] presents an algorithm for sensing and motion for exploration that considers the sensor model of real sensors (sonars), and copes with collision-avoidance issues. The algorithm incrementally builds a topological map of the partially known environment. It updates this map at every sensing location by incorporating local structure that has been computed from the current sensory data. These exploration types of approaches solve the navigation problem while ensuring that the whole environment is covered. The navigation map produced by the exploration approach is a connected graph of reachable nodes, which simplifies path planning. However, this type of approach is not useful if the objective is to navigate from a single point to another in a fairly efficient way since there is no notion of goal location.

An interesting question is whether the navigation task requires the construction of a model? Here also, the view planning literature provides some insight. Wixson [124] questions the necessity of these models in search problems, and presents a compromise between model-based and fixed-increment strategies. He shows

with empirical results that model-free methods that evaluate view-points spread out in fixed-increments in the environments compare favorably in computational speed and traversed distance to model-based ones in these search tasks. He also finds that the simplicity of these model-free approaches compensate for the advantages a model-based method may have. Whaite and Ferie state the opposite in [122]. They present a model based approach that uses a set of primitives for building a model. They argue that it is reasonable to compute a measure of uncertainty for such a primitive. The next best view [122] is then computed such that this next view will reduce the uncertainty of these primitives in the model currently being constructed. The computation time is fairly high since the whole environment is constructed from super-quadratic primitives. The flavor of this approach however is certainly a good starting point in determining which areas in our environment might need a closer look based on maximizing the information gain.

An objective function based solely on information gain is not appropriate for active sensing for planning since it does not account for additional costs for sensor deployment. Wahl [58] describes a more useful function for active sensor deployment. A "planning-sensing-updating" cycle is described, which based on the current, incomplete model and motion constraints, computes the next-best view such that an objective function in configuration space is maximized. The objective function consists of two terms; an information gain quantity per putative viewpoint and the traversal cost between view points. This approach of maximizing an objective function can be used in general as long as this function captures the desired behavior.

More insight can be gained from the "active vision" community. For example Kutulakos gives in [60,61] a good mathematical formalization of vision-guided exploration. This paper describes a method for exploration of arbitrary surfaces in 3D. The authors present in their paper two provably correct exploration methods. The first method uses a range sensor, and is complete for smooth and connected surfaces of finite area. The second uses a camera instead of a range sensor, and

is complete for smooth, non-concave, and connected surfaces. Their mathematical analysis of the next best view problem is of importance to any vision-guided exploration method. In their paper they address the problem of determining if a particular surface can be explored in a finite number of steps using common sensor types.

Another exploration approach is presented in [5]. This approach features a method that can plan for a next view very quickly, but unfortunately there are no guarantees about optimality or even an approximation. They present a method that identifies occlusion boundaries in range images and selects next views based on the information gain from a ray-traced model. To validate their method, they show plots of how the ratio of unknown versus the recovered volume increases over the number of views collected.

The *planetarium* algorithm as described by Connolly in [20] is another practical approach which exploits a constraint workspace to solve for sensor planning. The scene is represented by an octree and the putative sensing locations are distributed uniformly around a viewing sphere. The algorithm returns from these locations the location that yields the greatest unseen volume. They also present the so called *normal* algorithm in which each node in the octree describes a part of the scene by the six faces. The three faces that are neighboring the unseen volume are used to compute the viewing direction. This algorithm is faster, but does not deal so well with occlusions.

Maver's [79] approach extracts regions with no information content and then moves the sensor around the constrained workspace to fill in this data. This approach works well for their turntable setup, but has no practical application for sensing for navigation purposes. Another approach that exploits a constrained workspace is presented by Pito [94]. He describes a system that can automatically collect a complete surface model based on next best view calculation. The objective function used in this method contains only a measure for information gain and does not address sensing cost. The paper does address registration issues and selects the next best view with overlapping data in order to facilitate registration.

3. Sensor Based Path Planning

Different from the view planning methods discussed earlier are the sensor based path planning methods. They are different in the way that views are not planned for covering a certain region, but vehicle motion is planned with respect to sensor constraints and/or coverage of the area to traverse. In other words, the motion or path of the robot is planned simultaneously with the sensing action. For example the algorithm described in [17,18] continuously senses around the robot's perimeter and constructs a topological map while it navigates towards the goal.

While the algorithm in [17,18] continuously senses and plans a path, the sensing itself does not involve any particular view planning. If we would reason about the environment that we have observed so far, we can actively take measurements so that we can compute better paths. In [33], [34] and also [36] such active sensor based planning method is presented. The algorithm is based on detecting regions that are safe to navigate. These regions will be extended at every iteration according to newly sensed data. Candidate view points are generated at the boundaries of these safe regions, the viewpoint that exposes most of the unknown boundary is selected for the next viewing position. Although the focus of their paper is on exploration, the viewing for planning approach is completely integrated. A drawback of their algorithm is that it assumes that the range data does not contain any errors. Noisy data can cause mis-registering of range scans which may cause the result of the algorithm to become invalid. The sensing is also solely based on coverage and cannot be tailored for navigation purposes. The approach we have taken to overcome these issues is to use a confidence label on each data item and evaluate paths only with respect to the goal destination.

An alternative idea is explored in the "Bug" family of algorithms [74,96]. These algorithms assume a simple contact sensor and simple rules to navigate through unknown environments. Kamon et al. [52] extend this technique to incorporate the usage of range sensors by using the reduced visibility graph, better known as the tangent graph. The local tangent graph, which is the visible subset of the tangent graph to the robot, can be constructed by using the measurements of a laser range

finder. Instead of driving up to an obstacle, as in the bug algorithm the TangentBug algorithm, senses the perimeter and follows the local tangent graph.

Laubach extends in [69] and [68] the TangentBug algorithm even further into the WedgeBug (RoverBug) algorithm which explicitly deals with a sensor model. She adds some virtual states to the typical bug states (Motion towards goal and Boundary following), in these states the WedgeBug algorithm senses more from its environment to determine the local tangent graph which it uses to generate new path segments from. This algorithm is very elegant and clean in its underlying concept, however it still suffers from the same problems as the Safe region algorithm, that there is no notion of uncertainty in sensor data and it needs continuous detectable obstacle boundaries.

Most similar to our work is the work by Gancet and Lacroix [31]. They present in their paper the Perception-guided path planning (PG2P) approach which is a hybrid sensing and path planning method. Some of the same philosophies that we have embraced have been incorporated in the PG2P algorithm as well, although there are some notable differences between the two approaches.

Most importantly, the PG2P algorithm focusses on their definition of Confidence of Perception or COP. Their COP depends only on prior knowledge about the sensor model. For the type of range sensors they consider, the COP can be stated as a function of the perception distance. The COP is high for a location in close proximity and low for a sensed site that is further away. The COP can only be used if the site to be viewed already has an (obstacle/traversable) label and corresponding probability distribution. In contrast, we focus on the prior over the terrain itself and can therefore infer and leverage global terrain information.

Using the COP function together with the label probability, a traversability cost is defined [31]. The result is that the terrain's traversability is reinforced by the confidence in this measurement. Similar to what we propose is that given this metric, we can now reason about taking measurements. Taking a measurement and therefore reducing the confidence level will have an influence in the new traversability cost. They also define a utility for sensing that takes into account how much the

traverse cost changes as these simulated measurements are taken. The robot will then take a sensor measurement when this utility is positive.

Alternatively they select nodes on the boundary of the already observed area (so they can be reached safely) as locations for observations. However, this heuristic will not alleviate the problem of the myopic sensing horizon, since when we get to this boundary sensing location, we might discover that we encounter an obstacle.

Relevance to Thesis

This thesis will extend these independently derived ideas of sensing and simultaneously include mid-range sensing to actually alleviate the myopic path planning problem.

4. Planning with Uncertainty

Traditionally the path planning problem has been viewed in the context of perfect information. However, in practice this is rarely the case. Robots in the real world need to cope with partial information, which may also not be accurate and erroneous. In order to successfully plan under these constraints, a different framework is required.

One way of modeling this uncertainty is to pose the navigation task in a two player game-theoretic framework, in which the mobile robots system is one player and the environment is the other [26, 70, 71]. This is a very conservative approach, because the world in practice is not conspiring against the planning algorithm. These closed loop formulations do however provide great insights on the influence of uncertainty and how far the algorithm plans ahead against its opponent (the partially known environment).

In the remainder of this section we review a more general and commonly used probabilistic framework which models many, if not most aspects of our mobile

robot sensing and navigation problem. The notation used in the next few sections follows the conventions from [50].

4.1. Markov Processes

If all available information can be collapsed into the current state, and if the transition from this state to another state is not dependent on the history, this decision process is called a *Markov Process*.

A Markov process can be completely described by the tuple $(\mathcal{S}, T, s_0)$, in which $\mathcal{S}$ is a finite set of states, from which s_0 is the initial state and $T(s, s')$ is the stochastic transition function that brings the Markov Process from the current state s_t to the next state s_{t+1}. The behavior of the process is then defined by

$$Pr\left(s_{t+1} = s' | s_t = s\right) = T(s, s'). \tag{2.1}$$

4.2. Markov Decision Process

We specify a reward function $R(s)$ in addition to assuming the world as being Markovian. With a reward function $R(s)$ we can expand the Markov Process to specify a task in a Markovian environment.

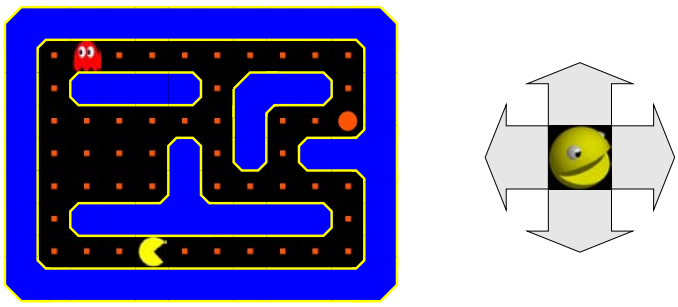


FIGURE 2.2. After Russel and Norvig [106]: Pacman lives in his favorite maze. He is currently at the initial state s_0, each move will have a reward of -0.01, and can be done in the four cardinal directions. A move into a wall will cause the agent to stay in place. In addition, visiting the ghost is rewarded -1, visiting the big • is rewarded +1.

In the example shown in Figure 2.2 the reward function $R(s)$ penalizes the agent by rewarding a -0.01 for moving from one cell to another, visiting the ghost

will penalize even more by a -1 reward. However if the big $\bullet$ is reached, the agent gets a $+1$ reward.

In order for the agent to maximize its reward, it can execute any of the following actions from the action set $\mathcal{A}$. with $\mathcal{A} = \{North, East, South, West\}$. Since each move has a negative reward of -0.01, the agent will want to make as few moves as possible to get to the goal reward of $+1$, while avoiding incurring the -1 reward altogether. We can formalize the problem now completely as a *Markov Decision Process MDP* with the tuple $(\mathcal{S}, \mathcal{A}, T, R, s_0)$.

Since the agent's choice of an action at the current moment, $a_t \in \mathcal{A}$ influences how the MDP transits to the next state, the transition function becomes now: $T(s, a, s')$. With this modified transition function, the evolution of the Markov Process is therefore given by

$$Pr\left(s_{t+1} = s' | s_t = s, a_t = a\right) = T(s, a, s'). \tag{2.2}$$

If the world has a terminal state (goal) and that state is reachable, this would be sufficient. However, if that is not the case, rewards will grow towards infinity and you will therefore often see the tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma, s_0)$ used. This tuple describes the same MDP with discounted rewards. Rewards are in these MDPs discounted with γ ($0 < \gamma < 1$), resulting in the agent preferring current rewards over future rewards.

For the agent to achieve the maximum expected reward, the agent should select its actions according to a policy π for each state there is. Since the agent is in a Markovian world, it only needs information available in the current state for it to act optimally. The expected long term reward or *Value function* is defined as

$$J^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | \pi\right]. \tag{2.3}$$

An optimal policy π^* is then defined as

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J^{\pi}(s). \tag{2.4}$$

The *Value Iteration* algorithm that uses the *Bellman equation* for the expected cost of subsequent states solves for the optimal policy π^* [106].

4.3. Partially Observable Markov Decision Process

The limitation of the MDP modeling incomplete information was reason for Cassandra et al. [15] to develop the *Partially Observable Markov Decision Process* (POMDP) framework. The complete state in the POMDP is hidden from the agent and can only be partially observed by a noisy sensor measurement.

The POMDP inherits all the elements from the MDP, but has also an observation model $O(s, o)$ and a belief state $b(s)$. The observation model models the probability of receiving an observation o from all possible observation $\mathcal{O}$, given the state s. For the example shown in Figure 2.2 the observation model consists of only one observation; the empty observation that occurs with a likelihood of 1 for each state.

Since the optimal action can not depend on the state the robot is in (since this state is not known to the robot), the best policy is described in respect to the current belief state. The POMDP transits from one belief state to the next according to

$$\tau(b, a, b') = \sum_{o} P(b'|o, a, b) \sum_{s'} O(s', o) \sum_{s} T(s, a, s')b(s). \tag{2.5}$$

Similarly, the expected reward for the actual state the agent is in, is defined by

$$\rho(b) = \sum_{s} b(s)R(s). \tag{2.6}$$

With τ and ρ defined as in 2.5 and 2.6, the POMDP can be expressed as an observable MDP on the belief states. Furthermore an optimal policy $\pi^*(b)$ for this MDP in belief space can be shown to be also the optimal policy for the original POMDP formulation.

However, the belief space is a continuous and in most real problems a high dimensional space. It is therefore very difficult to find optimal policies in this space. Researchers have therefore looked into exploiting structure that is inherent to the

problem to reduce the problem [105]. Other researchers focus on making it more tractable to solve larger POMDPs [10,11,43,73,93,129].

Relevance to Thesis

The navigation problem that we are interested in is an excellent example of a POMDP problem. However the number of states, actions and observations that a typical instance would have would make finding a optimal policy impractical. Let us look at a very coarse description of our navigation problem:

$\mathcal{S}$	$s_0 \ldots s_n$	each state representing a cell in an obstacle map. A typical state space size n of 100^2 is nothing out of the ordinary to model an obstacle map.
$\mathcal{A}$	$a_{MN}, a_{MNE}, \ldots a_{MNW}$	a very limited 8 connected move set.
	$a_{s0^\circ}, \ldots a_{s355^\circ}$	assuming we discretize the pointing angle for the sensor to a 5° interval. For a typical midrange set-up, 0.5° would be more reasonable.
$\mathcal{O}$	$o_{mob000}, \ldots o_{mob255}$	observation subset of a neighboring cell only mobility sensing system. More typical would be a 5 neighborhood radius, yielding approximately 2^{68} different observations.
	$o_{mid0^\circ r00}, \ldots o_{mid355^\circ r50}$	observation subset discretized on a 5° by 50 cells radial lattice.

For each cell traversed the robot will get a small negative reward to represent the cost of traversing. Traversing an obstacle cell will incur a steep negative reward and reaching the goal will be rewarded a very high positive reward. Although the way the problem is described is fairly coarse, it is clear that the problem is from a very high dimensionality. Because of the branching factor of the alternative belief states and the fact that the belief state is continuous, it is unfortunately computationally intractable to solve this problem with the current POMDP solving methods [106].

5. Remarks

The algorithm that will be described in the following sections combines ideas from the approaches that are traditionally used for outdoor navigation and a complete probabilistic framework. Although the approach presented still relies on a

grid based path planner, we use a probabilistic model to reason about what we expect to discover as we move through the environment and how to best verify this hypothesis with observations.

CHAPTER 3

Mid-range Sensing and Planning Algorithm

As a solution to the navigation in unstructured and partially known terrain problem we propose a combined view planning and navigation strategy. As indicated in Chapter 1, a path planning navigation algorithm can perform better if it has more knowledge about the environment.

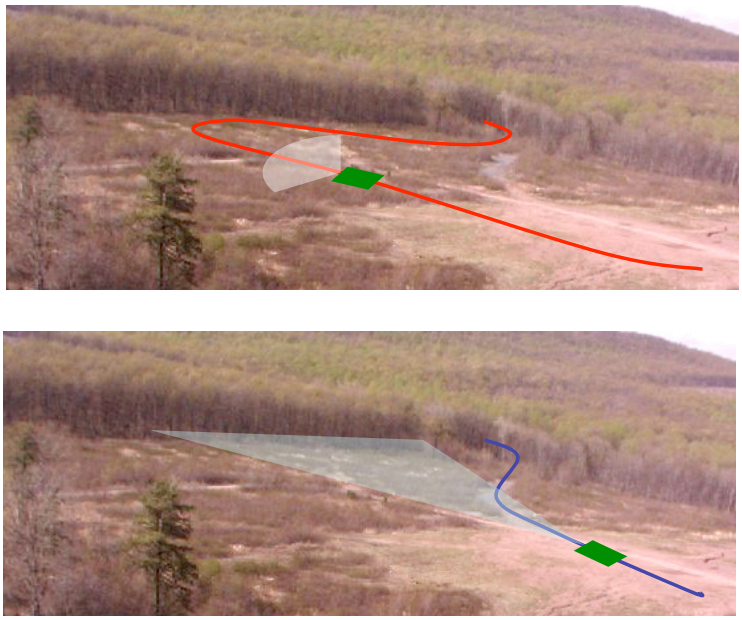


FIGURE 3.1. Top: poor performance due to lack of sensor planning and mid-range sensing, the robot has to rely on mobility sensor data only. Bottom: the pinch point is detected earlier which allows the planner to adjust the path accordingly.

For navigating in unstructured environments, the use of longer range laser range finders or wide-baseline stereo approaches could potentially provide this

needed knowledge about the environment. In order to alleviate this finite horizon problem for path-planners, we would like to use mid-range sensor data (Figure 3.1). However, there are some issues to be addressed to successfully use this type of sensing modality:

- The sensor data rate is low and it does not allow for continuous sensing
- The field of view is very narrow, the sensor needs to be pointed in the most "useful" direction
- The data is sparse, the registered data will be incomplete so we need to deal with missing data

Figure 3.2, illustrates the influence of missing data on the path computed by a planner. The absence of data could mean that there is traversable space, or a possible unobserved obstruction. In the latter case, in which the planner will discover the obstacle at the very last moment, the path is far from desirable.

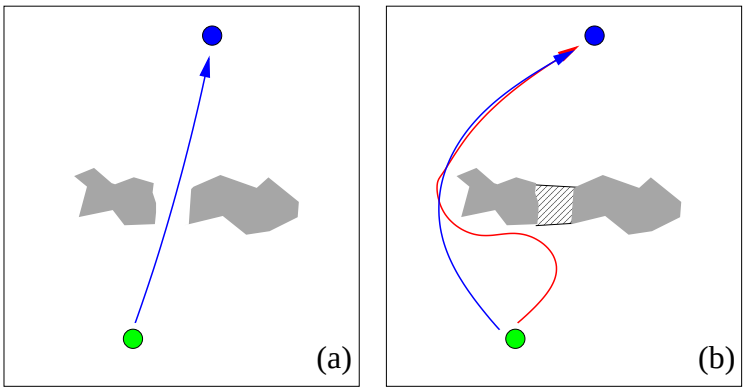


FIGURE 3.2. The partial map in (a) suggests a gap between the two obstacle regions. A large deviation from the path will be incurred if the gap is actually blocked (b).

This thesis will address the problems associated with using mid-range data for path-planning. Key to the solution is the insight that we can simulate the environment as we would observe it in the future. Sensor measurements and sensor positions can now be planned to observe the differences between the current and the expected environment in the future and therefore sense anticipated obstacles in a timely manner. The algorithm presented uses the current perceived world and

prior knowledge about the obstacle/free space distributions to compute intermediate observation locations.

1. Planning a Path

At the core of this solution is a grid based D* planner [115,116]. D* is a dynamic A* planning algorithm that produces a path based on the current assumed sensor information. It then incrementally repairs the plan as new information is sensed. The repaired D* plan is optimal (under the used cost metric) and is equivalent to replanning from scratch. In our case, we use D* to compute the best path from the current vehicle location to the goal point as usual. In addition to the usual traversability costs, we also evaluate the expected path costs for paths that first go through an intermediate observation location, take a measurement, and then go to the final goal destination. We define a positive utility for a location in the map if the expected traverse from current location to the goal through an observation location is cheaper than the expected cost for going to the goal directly (Figure 3.3).

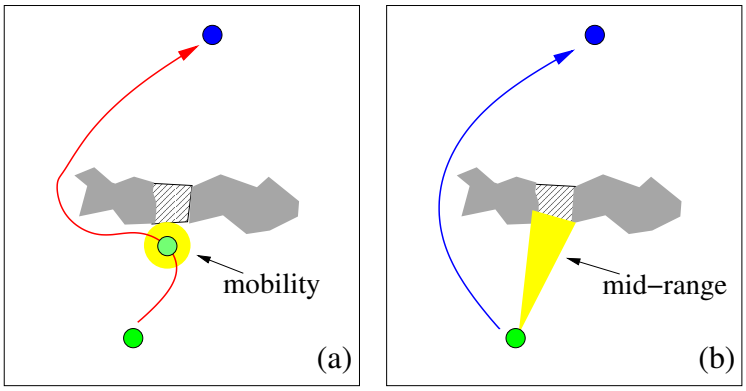


FIGURE 3.3. The obstacle is detected late, resulting in an expensive detour as in (a). In (b) a mid-range sensor measurement supplies the missing information to plan a more optimal path. The difference between the two path lengths yields a positive utility for the observation location.

We use D* as our path planning method because it is critical to our approach that we can reuse as much of our path planning computation as possible. The nature of our algorithm warrants evaluation of many different alternative paths which D* can do efficiently by incrementally updating and readjusting its search space (see also Chapter 6, Section 4).

This approach of incorporating utility into a cost is similar to the approach Rosenblatt [100, 101] takes in his arbiter. However, his utility map only looks at a few possible actions over a limited horizon and does not deal with additional utility for sensing purposes.

2. Forward Simulation

Let us now formulate further this notion of "sensing utility". Consider a candidate observation position $\mathbf{u}$ in our map. At this position, we can point our sensor at an angle θ. Given the field of view and minimum and maximum sensing distance of the sensor, we can define a sensor footprint. In addition we define $\mathcal{L}$ as the set of all possible obstacle/traversable labellings of the map. Each labelling

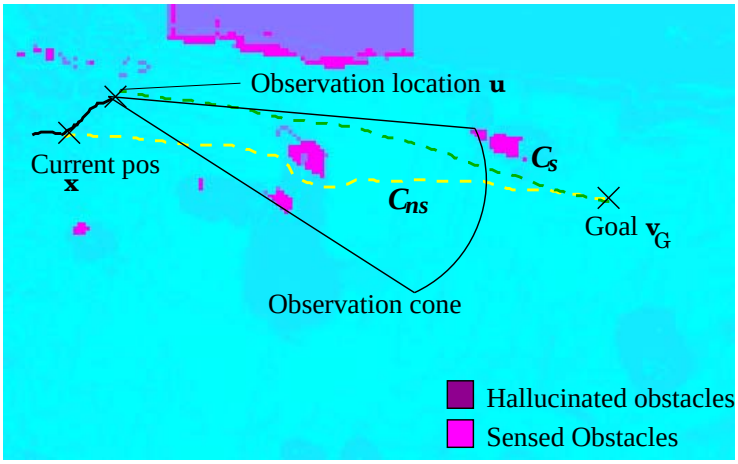


FIGURE 3.4. Illustration of the path computed with (C_s) and without sensing (C_{ns}) in an example world configuration.

($L_j \epsilon \mathcal{L}$) has an associated probability $P(L_j)$. Given such a labelling and the sensor footprint, one can compute the cost of getting from the current robot location $\mathbf{x} \epsilon \mathbb{R}^2$ to the goal $\mathbf{v}_G \epsilon \mathbb{R}^2$ by not making the observation at $\mathbf{u}$: $C_{ns}(L_j, \mathbf{x}, \mathbf{v}_G)$, and the cost of getting to the goal making the observation: $C_s(L_j, \mathbf{x}, \mathbf{u}, \theta, \mathbf{v}_G)$ (Figure 3.4). Intuitively, $C_{ns}(L_j, \mathbf{x}, \mathbf{v}_G)$ is the cost of the path executed if no sensors other than the short-range mobility sensors are used and the map is in configuration (L_j). The utility ψ for visiting $\mathbf{u}$ while navigating to the goal is defined as:

$$\psi_{\mathbf{u}}(\mathbf{x}, \mathbf{v}_G) = 1 - \operatorname*{argmax}_{\theta} \left(\sum_j \frac{P(L_j) \left(C_{ns}(L_j, \mathbf{x}, \mathbf{v}_G) - C_s(L_j, \mathbf{x}, \mathbf{u}, \theta, \mathbf{v}_G) \right)}{C_{ns}(L_j, \mathbf{x}, \mathbf{v}_G)} \right) \tag{3.1}$$

In principle, the utility can be computed everywhere in the map and the resulting utility map (see also Figure 3.5) can be used by the planner to select the most favorable sensing position. This "forward simulation" [26, 53, 70] is at the heart of our approach.

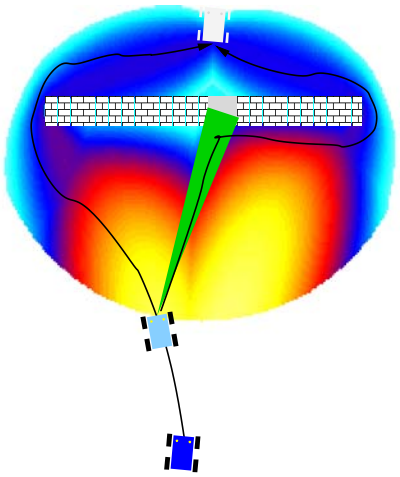


FIGURE 3.5. The utility map displayed for a very simple partially known world. The robot depicted at the bottom of the figure denotes the current robot position. A high measure of interest is represented with a bright color. The brightest point (highest utility) is used as the next sensing location.

3. Computational Complexity

Both C_s and C_{ns} can be evaluated by running the planner on different "virtual" configurations of the world map corresponding to different configurations L_j. In reality, however, this would require the enumeration of *all* possible configurations of the world which is clearly a combinatorially large set. Furthermore, the sum above must be evaluated, in principle, not only for all cell locations, but also for all possible sensor orientations. Therefore, the computation of the optimal utility function defined above is not tractable and an approximation must be used

to reduce the computation while retaining near-optimal evaluation of the utility. In the following chapter such an approximation will be presented that will limit the number of labellings that need to be considered. Incidentally, the number of orientations to be evaluated can also be reduced as a side effect.

4. Conclusion

With the forward simulation framework presented here we have now a way of evaluating potential mid-range observation locations and thus being able to decide "when to look". When to look, or taking a mid-range sensor measurement, can now be limited to the most beneficial locations. However the formulation presented is intractable to compute this, we will therefore use an approximation instead.

CHAPTER 4

Hallucinating Worlds

In order to be able to successfully perform forward simulation, we will need to predict how the world is likely to look like in the future. More specifically, we would like to know which parts of the environment to be observed in the future will be traversable or not. This problem can be expressed as an inference problem: given what we have observed so far and some prior model, we can infer, or hallucinate, the underlying world map and hence the traversability. With this hallucinated world we can perform forward simulation and thus compute the paths that are needed to compute the utility for taking additional mid-range sensing measurements.

In order to be able to hallucinate the world as it will in the future, we could use a previously learned distribution of worlds and use an inference mechanism to draw most likely world configurations from it. Since we are only interested in knowing if a particular cell is traversable, a binary representation of the world will be sufficient. In the remainder of this chapter, we will formulate our inference problem in the well known Markov Random Field framework.

1. Hallucinating

A hallucinated traversability label for a particular map cell needs to adhere to the underlying spatial distribution of obstacle vs. non-obstacle. This problem of grouping features together has been well studied in the field of computer vision [8,

72, 125]. Because of the similarity between hallucinating our traversability map and inference from noisy images, we have been looking at applying techniques from the computer vision community to our problem.

For our forward simulation to work, we will need to infer a traversability labelling. Let us denote such a traversability labelling with L. The sensing system that the robot uses to observe the world will only yield a partial terrain map. This partial terrain map is denoted by D. We can now formalize how to hallucinate our anticipated world L from the current observed terrain D.

Each L is a world configuration hallucinated from the data. If $\mathcal{N}$ is the number of cells in the map, $|L| = 2^{\mathcal{N}}$ becomes computationally intractable in practice even with small values of $\mathcal{N}$. In order to make it possible to compute the utility $\psi_{\mathbf{u}}$ we will use a common approximation that assumes that $P(L \mid D)$ can be approximated by a delta function [59]. This amounts to replacing the average configuration of the world with the most probable configuration or Maximum A Posteriori (MAP) estimate.

$$P(L \mid D) \approx \delta(L, \hat{L}) \tag{4.1}$$

with

$$\hat{L} = \underset{L}{\operatorname{argmax}}\, P(L \mid D) \tag{4.2}$$

Intuitively, $\hat{L}$ is the most likely world configuration inferred from the observed data.

The next problem we need to address is how to compute $P(L \mid D)$. We start with the observation that the labels on cells expressing traversability are not independent. If a cell is classified as non-traversable, then there is high probability that the neighboring cells are also non-traversable except at the discontinuities. This is also true for the traversable cells. This kind of contextual dependency in the labels on 2D lattices has been well studied and is often referred to as spatial smoothness.

A standard approach for representing this spatial smoothness is to use Markov Random Fields (MRFs), which can incorporate local contextual constraints in labelling problems in a principled manner [72]. These MRFs provide an efficient method to group features based on their labelling while adhering to an underlying spatial model. This inference technique has been successfully applied in the field of computer vision [8,72,125] to infer the original image from noisy data. Because of the similarity of the problems addressed in the computer vision community with our problem, inferring an obstacle labelling for a partially known terrain led us into formulating our inference problem into the MRF framework.

MRFs are generally used in a probabilistic generative framework that models the joint probability of the observed data and the corresponding labels. In other words, let D be the observed data from the terrain, where $D = \{d_i\}_{i\epsilon\mathcal{S}}$ and $d_i \epsilon \mathbb{R}$, d_i is the terrain data from the i th site, and $\mathcal{S}$ is the set of sites. Let the corresponding labels at the terrain sites be given by $L = \{l_i\}_{i\epsilon\mathcal{S}}$ with $l_i \epsilon \{-1, 1\}$. In our case, l_i indicates the presence or absence of an obstacle at location i. In the MRF framework, the posterior over the labels given the data is expressed using the Bayes rule as,

$$P(L \mid D) \propto p(L, D) = p(D \mid L)P(L) \tag{4.3}$$

2. Data Term

The first term in the product, $p(D \mid L)$, in Equation 4.3 is known as the likelihood of the data. In general, it is assumed that the data at each site is conditionally independent given the labels at that site, i.e. $p(D \mid L) = \prod_{i\epsilon S} p(d_i \mid l_i)$ [8,72]. We model the likelihood for each class, traversable/non-traversable, as a Gaussian.

$$P(d_i \mid l_i) = \frac{1}{\sqrt{2\pi\sigma_{l_i}^2}} \exp\left\{\frac{1}{2\sigma_{l_i}^2}(d_i - \mu_{l_i})^2\right\} \tag{4.4}$$

The parameters of the two Gaussians representing the likelihood for these classes can be easily learnt from labeled training data. This learning takes place

for a typical type of environment where the robot is to navigate in, for example on Mars, an exemplar is used for learning. There after the robot will use these learned parameters for online classification/hallucination of terrain. For the current model, this learning boils down to fitting a single Gaussian to each of the two classes.

3. Field Model

After modeling the likelihood of the data, we need to model the prior over label configurations $P(L)$, which encodes the notion of spatial smoothness of the terrain labels except at discontinuities. Since we have only two classes (traversable/non-traversable), this can be expressed in a MRF formulation as a binary classification problem. The data likelihood $p(D \mid L)$ is assumed to be conditionally independent given the labels and the label interaction field $P(L)$ is assumed to follow a homogeneous and isotropic Ising model [80] with only a pairwise interaction term: $\beta l_i l_j$ where β is the interaction parameter of the MRF and l_i, l_j are the labels at two neighboring cells. The Ising model favors neighboring sites with the same labels and penalizes the dissimilar labels by cost β [59].

Combining the likelihood model $P(D \mid L)$ with the prior over labels $P(L)$, we can write the overall posterior over labels as follows:

$$P(L \mid D) = \frac{1}{Z_m} \exp\left(\sum_{i \epsilon \mathcal{S}} \log p(d_i \mid l_i) + \sum_{i \epsilon \mathcal{S}} \sum_{j \epsilon \mathcal{N}_i} \beta l_i l_j\right) \tag{4.5}$$

In which Z_m is a normalizing constant, often referred to as the partition function. For this MRF (Equation 4.5), computing the Maximum A Posteriori (MAP) configuration of the labels, $\hat{L}$ given the data can be solved exactly using graph min-cuts/max-flow algorithms if $\beta > 0$ [8,37]. With such a min-cuts algorithm [59], we now have a way of computing $\hat{L} = \mathrm{argmax}_L \, P(L \mid D)$.

4. Inference

The min-cuts/max-flow algorithm that solves the MRF equation (4.5) produces a labelling $\hat{L}$ of traversable/non-traversable cells that is most consistent with the world model and the current data. This binary labelling of traversable and non-traversable cells is updated for a given instance of the currently observed terrain data and represents the most anticipated configuration of the world (Figure 4.1).

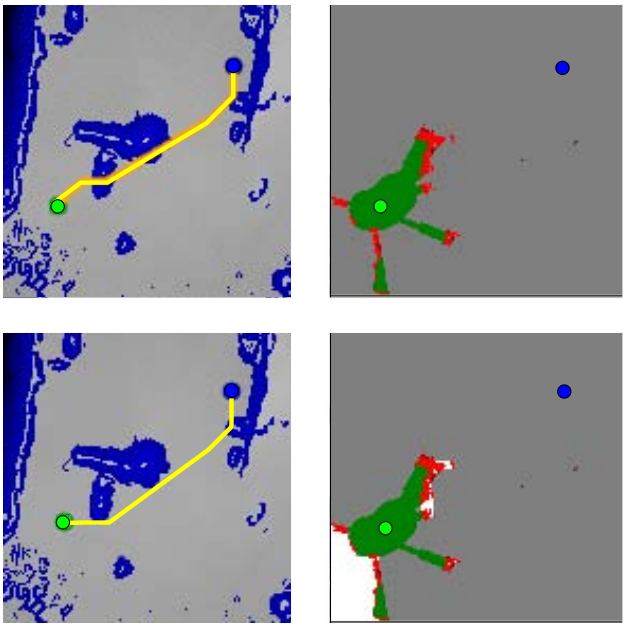
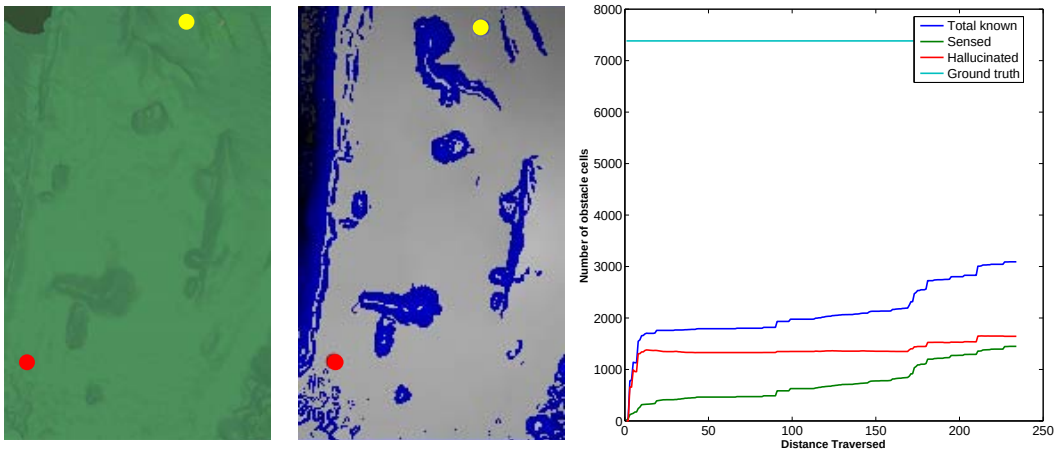


FIGURE 4.1. Typical example that shows the influence of the obstacle hallucination. The robot (in green) travels to the goal (blue). The path is shown in yellow. The hallucinated obstacles are in white, hallucinated empty space in gray, observed traversable in green and observed obstacles are red. In the top panel, the planner uses only the information from the sensors and it plans a path through the obstacle. In the bottom panel, the hallucination algorithm merges obstacles correctly together and produces a better path.

It is interesting to examine how the influence of the inference behaves over a typical traverse of the robot. To illustrate the operation of the obstacle inference algorithm, Figure 4.2 shows an example experiment. In Figure 4.2(a) and (b), the example terrain is shown with its ground truth obstacle labelling. The planner simulated a robot traversal from the red to the yellow markers.

The number of map cells that are detected as obstacles at every position along the path is shown in Figure 4.2(c), along with the number of hallucinated cells. The ground truth, that is, the total number of actual obstacle cells computed directly from the underlying elevation map of Figure 4.2(a) is also included. This number provides a baseline for reference: It is the total number of obstacle cells that would be included in the map if the entire world had been explored and sensed.



(a) (b) (c)

FIGURE 4.2. Example demonstrating the influence of the obstacle hallucination over the course of a typical traverse. (a) is an overhead view of the terrain, with start and goal location marked. The ground truth obstacle labelling is displayed in (b). Obstacle cells are marked in blue. The number of obstacle cells is plotted in (c). The graphs confirm the intuition that while more of the environment gets explored, less needs to be inferred. In the limit, the number of inferred cells will go to zero and the total number of known obstacle cells will equal the number for the ground truth labelling.

When we look at the number of cells hallucinated in Figure 4.2(c) over the course of a traverse, we see that the number of inferred cells first increases rapidly and then tapers down when we have sensed more of the environment. This is because initially there is no data available, therefore the hallucination algorithm has no evidence for any possible obstacles. It will therefore not infer any obstacles. As some sparse terrain data has been sensed, there is enough evidence for some cells to be inferred and classified as obstacles. However, when even more of the environment becomes known and inferred obstacles get confirmed, the number of

inferred obstacles decreases. In the limit, when all cells have been explored there will be no inferred obstacles.

5. Conclusion

By modeling the traversability of the terrain as a MRF, we can use an efficient graph-cut algorithm to hallucinate the most likely traversability labelling of a partially known terrain map. This will allow us to use the formulation developed in Chapter 3 to compute the expected path costs for the most likely world configuration and thus identify beneficial observation locations for mid-range sensing.

CHAPTER 5

Where to Look

WITH the algorithm laid out in Chapter 3 and a method to infer the most likely world in Chapter 4 we have all the components to implement the selection of intermediate viewpoints that are beneficial for navigation. The following sections revisit the original forward simulation definition to reflect the MRF framework. In addition, we will also define several metrics to eliminate an extensive search over the viewing angle θ.

In the following sections, we will not define θ explicitly, instead we will simplify the discussion by using the term *"Point Of Interest" (POI)*. It is sufficient to limit the discussion to just the point of interest since, the gazing angle θ is fixed such that it will bring the point of interest in the center of the view (Figure 5.1).

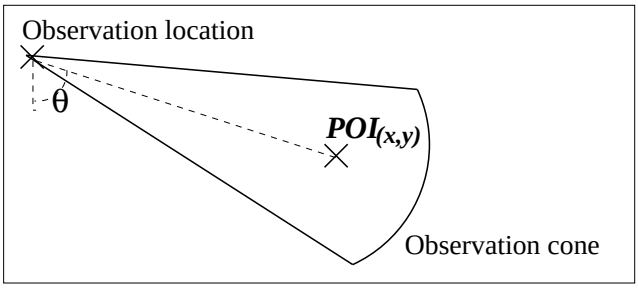


FIGURE 5.1. The relation between θ and *point of interest* (POI).

1. Expected Path Costs

The MAP estimate yields the most likely world obstacle label configuration and its corresponding likelihood. Given this labelling and the individual cell probabilities we can express in more detail the expected path costs as originally introduced in Chapter 3.

In Figure 5.2 we formulate the paths for both the non-sensing and the mid-range sensing case. Different from Chapter 3 is that we are considering only the most likely obstacle configuration of the world. Based on the probabilities from the MRF we can compute meaningful probabilities for the inferred regions [81].

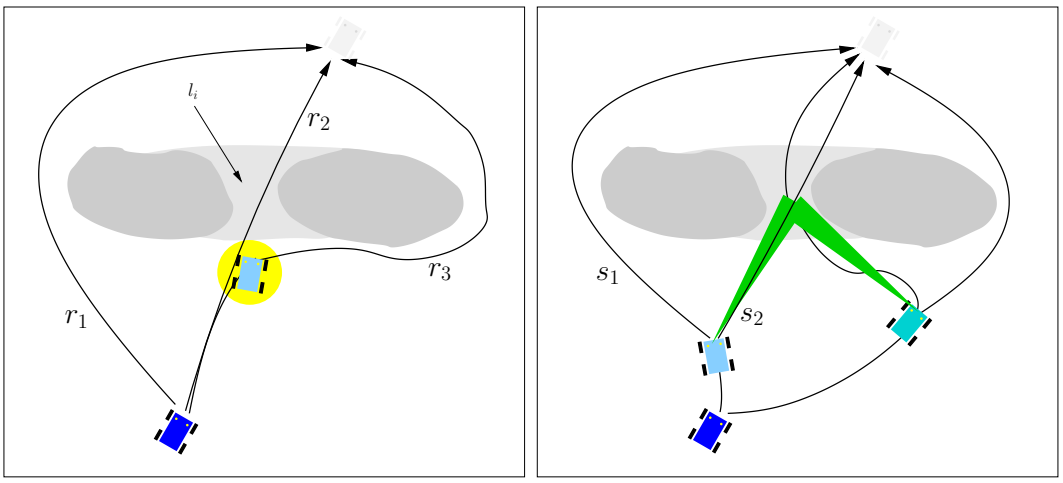


FIGURE 5.2. The paths displayed in the left figure are possible traverses depending on the interpretation of the inferred (shaded light gray) area of the obstacle. r_1 is the path the robot would take if it is very confident that this area is blocked. r_2, r_3 are the possible paths if the robot beliefs the inferred area is traversable. More specifically, r_2 when the region *is* actually traversable and r_3 when the region *is* actually blocked. The diagram shown on the right shows the paths that first go through an intermediate observation location and then to the goal. Path s_1 follows after observing the inferred area blocked, s_2 after observing the inferred area traversable.

Given a labelling, the paths denoted in Figure 5.2(a) with r_1, r_2, r_3 are examples of the three different classes of paths the robot can choose from when it chooses not to take a sensor measurement. Depending on the confidence of the inferred area l_i, the robot needs to decide between the two different interpretations of this area. A very high confidence ($P(l_j) > \tau$) of the interpretation that the area is untraversable, will lead to the path r_1. Whereas a lower confidence will yield an expected path

cost of the r_2, r_3 combination that is more favorable.

(5.1)
$$C_{ns}(l_j, \mathbf{x}, \mathbf{v}_G) = \begin{cases} C_{r_1} & \text{if } P(l_j) > \tau \\ (1 - P(l_j))C_{r_2}(l_j, \mathbf{x}, \mathbf{v}_G) + P(l_j)C_{r_3}(l_j, \mathbf{x}, \mathbf{v}_G) & \text{otherwise} \end{cases}$$

The introduction of the parameter τ is not strictly necessary because, if the expected cost of the r_2, r_3 combination is lower than C_{r_1}, the robot could just favor the conservative path r_1. With τ added to the definition, there is a parameter to influence how curious or conservative the robot behaves.

The expected costs of the paths (Figure 5.2(b)) that go through an intermediate observation can be updated in a similar fashion:

(5.2)
$$C_s(l_j, \mathbf{x}, \mathbf{u}, \mathbf{v}_G) = (1 - P(l_j))C_{s_2}(l_j, \mathbf{x}, \mathbf{u}, \mathbf{v}_G) + P(l_j)C_{s_1}(l_j, \mathbf{x}, \mathbf{u}, \mathbf{v}_G)$$

Notice that the sensor orientation θ has been dropped from the definition, because we can orient the sensor such that the inferred cells in l_i will be observed. Each location $\mathbf{u}$ for which the expected total path cost C_S is lower than the expected direct path cost C_{ns} we have a positive utility and therefore a candidate waypoint.

2. Restricting Sensor Orientation

So far we have shown how the computation over all possible map labellings can be reduced to evaluating a single, most likely instance. However, we have not defined yet how alternatives for the sensing direction θ are selected. The goal is to reduce the search space over θ. Depending on the evaluation mode (see also Chapter 6) different strategies apply. The following sections detail each approach.

2.1. Sampling from the Field Model

One possible approach would use the fact that we have modeled traversability as a Markov Random Field, which allows for sampling worlds from this distribution. Given such a sample world, we can plan a path to the goal and keep a tally at each cell that will be increased each single time the cell is traversed. If we sample our distribution correctly, the collection of these tallies gives us the distribution of

cells likely to be traversed, and therefore the distribution of cells that are interesting for us to observe.

Previous chapters have discussed efficient sampling methods for Binary Markov Random Fields [30, 48, 49]. However, we still have to draw many samples of inferred worlds and need to compute their corresponding paths, which becomes impractical for an online planning algorithm. For the online version of the algorithm, we can use the Maximum A Posteriori (MAP) estimate from Chapter 4, Section 1 and restrict ourselves to just observing those cells in the MAP estimate that differ from the current known labelling. Because if the labelling is identical, they will not influence the path and therefore do not need to be observed.

2.2. Interesting Areas

As an alternative approach, we can ask which sites are worthwhile to observe regardless if they can be observed from an observation location (Figure 5.3).

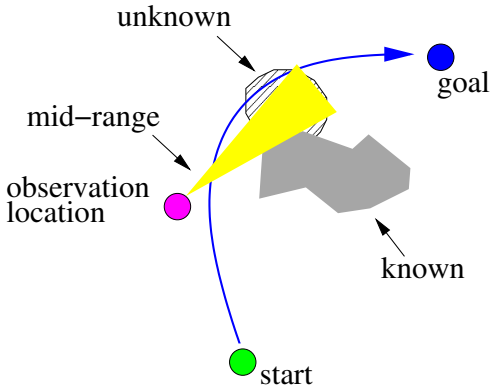


FIGURE 5.3. Illustration showing the relation between an interesting area (marked unknown) and observation location (in magenta).

We will look at some metrics that can express this. Ideally these would be the areas which our planning algorithm would like to collect more information in order to compute a better path.

In defining *"interesting"* areas, we use the following observations. An area is said to be interesting to observe if:

- an area has not been observed we are interested in observing it to determine its obstacle labelling.
- an area contains a lot of clutter, there is a mix of the different cell labellings "obstacle" and "free", this area is of interest.
- an area is likely to be traversed e.g. the path lays within, or is close to it, this area is of interest.

"Interesting" with respect to unknown cells can therefore be expressed as:

$$\sum_x (1 - Conf(x))$$

The confidence $Conf(x)$ expresses how much confidence we have in the sensed terrain data at site x. In addition, we can express interest due to clutter as:

$$\sum_x \left(1 - \left|2\left((G_\sigma \otimes Occ(x)) - \frac{1}{2}\right)\right|\right) \tag{5.3}$$

With G_σ a smoothing kernel with a support region of σ and Occ the occupancy map. Figure 5.4 illustrates the behavior of this definition on two example occupancy configurations. The definition of interest due to closeness to the path follows

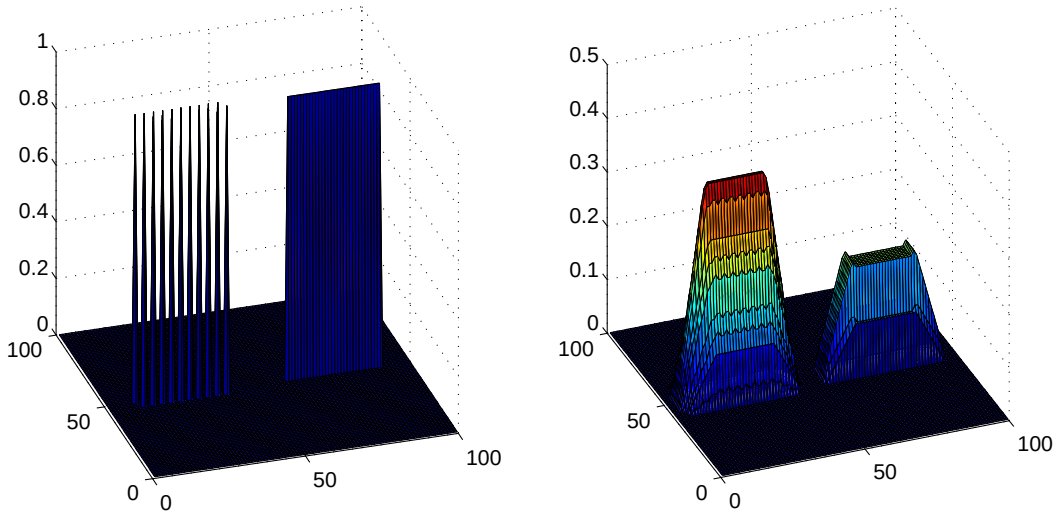


FIGURE 5.4. The Occupancy (labelling) of the example configuration is shown on the left. On the right, the response of the definition from equation 5.3 is shown. The comb style obstacle results in a higher degree of interest.

the following exponential definition:

$$\sum_{x} e^{-\frac{\mathrm{d}(\mathcal{P},x)}{\eta}} \tag{5.4}$$

In which η controls the range of influence of the path, and $\mathrm{d}(\mathcal{P})$ denotes the minimum distance to the path $\mathcal{P}$.

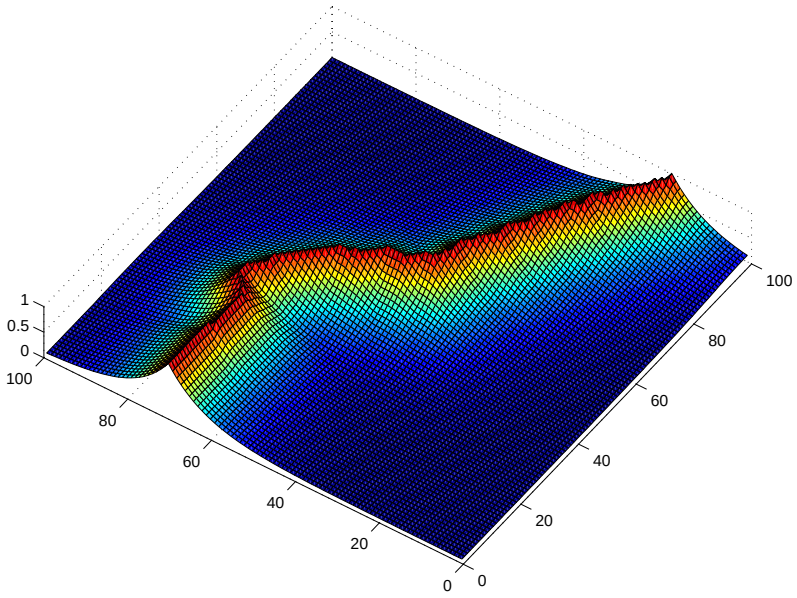


FIGURE 5.5. The measure of interest due to expected traversal for an example path and $\eta = 10$

Combined, we have a heuristic definition that peaks if locations close to the path are or unknown or cluttered:

$$\sum_{x} \operatorname{argmax}\left(1 - Conf(x), \left(1 - \left|2\left((G_\sigma \otimes Occ(x)) - \frac{1}{2}\right)\right|\right)\right) e^{-\frac{\mathrm{d}(\mathcal{P},x)}{\eta}} \tag{5.5}$$

One problem with this definition is that the path $\mathcal{P}$ is known only until after the robot completes its traverse and reaches the goal. Up until then we only have the current best estimate for the path given by the current knowledge of the world. In order for this definition to be more general we would need to summarize over all possible paths and their likelihood.

This heuristic for determining the viewing direction is only used if the inference mechanism does not yield any inferred cells that influence the path. This is

for example the case if we start out with an initial empty map and the inference needs some initial data to bootstrap the forward simulation.

2.3. Considering the Topological Map

Since the previous definition of the interest value (eq. 5.5) has the problem that the path $\mathcal{P}$ is not known, we should in principle evaluate over all possible paths. However, evaluating all possible paths is computationally intractable. Instead we suggest another approximation that only uses a small subset of paths.

When we consider the way that sensing data is collected, there will always be a so-called *sensing perimeter*. This perimeter is located at the maximum range that the robot is capable of sensing and it propagates when the robot moves. Obstacles will appear at this perimeter, when the robot moves while sensing its environment. Each obstacle that has been sensed at this perimeter can either be passed to the left or right side. Each of these two paths belongs to a different homotopic class. Two paths P_1 and P_2 are in the same homotopic class if and only if P_1 can be continuously distorted in P_2 without intersecting the obstacles [9, 12, 14, 25, 45]. For our evaluation, we consider only one representative path from each homotopic class (Figure 5.6).

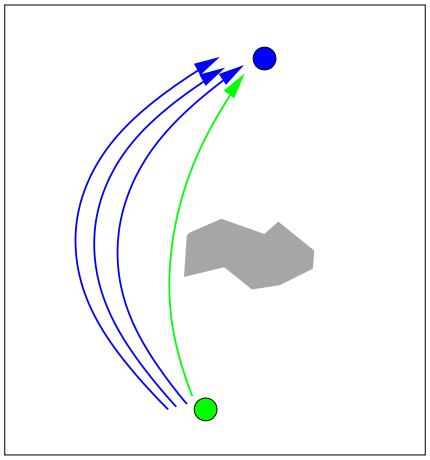


FIGURE 5.6. All the paths marked in blue are discarded for evaluation, since we use only one singe representative per homotopic class (marked in green).

To illustrate why we can justify evaluating only one single path per homotopic class, we have included the following example experiment. Figure 5.7 shows that even if the path planner chooses a different path around the obstacle, the location of the maximum of the interest metric (eq. 5.5) does not change substantially. In addition, unless the environment changes drastically, the shortest path actually executed by the planner will be very close to the representative path. However, if the environment changes drastically, it is likely that one of the other class representatives becomes preferable.

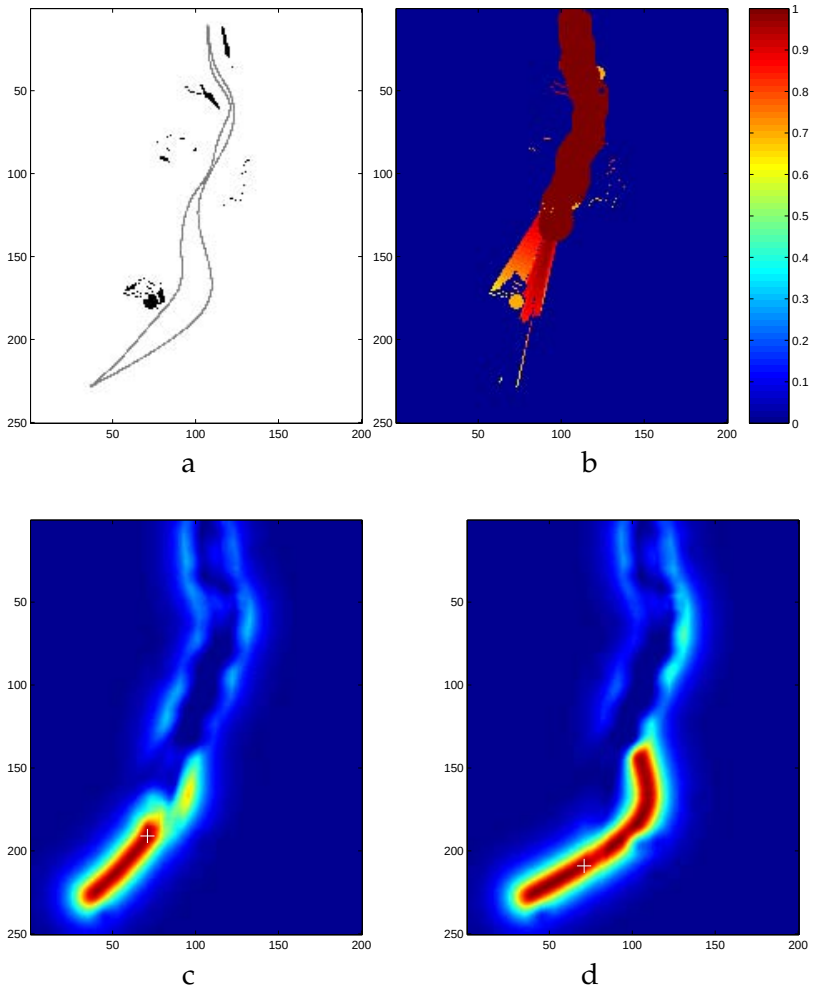


FIGURE 5.7. Example illustration that shows that for different paths in the same homotopic class, the interest metric and most importantly the most interesting point (marked by +) changes only slightly. (a) shows the obstacle labelling and the two example paths. In (b), the confidence for the cells is displayed. (c) and (d) show for each path the interest metric from equation 5.5.

This heuristic is used in the same way as the definition of interest in Equation 5.5. It differs however in that, instead of returning a single interest location, it returns one interest location for each homotopic class. The forward simulation view planner will therefore consider all these alternate viewing angles, resulting in sensor measurements being considered in a wider variety of areas. The data collected in these areas might be useful if the current pursued path turns out to be unattainable and an alternative one needs to be planned.

3. Summary

Having defined the point of interest in relation to the viewing angle, we can simplify the question "where to look" to "what is of interest". We also present several heuristics to determine these points of interest (POI's).

One method follows straight from our inference framework. All cells that have a different labelling (obstacle or traversable) in the inferred and sensed world are point of interest candidates.

As an alternate approach we use a heuristic measure that takes into account the confidence over the obstacle labels, the amount of clutter and the likelihood that the cells will be traversed. We consider a small subset of homotopic classes to determine if a particular cell is expected to be traversed.

These two methods are just two example heuristics that address the question "where to look". Other heuristics could possibly work as well, the ones presented here have shown to work well enough in our system. With the question of where to point our sensor answered, we can now complete the mid-range sensing and planning algorithm.

CHAPTER 6

System Integration

So far we have discussed the main details of the view-point selection mechanism. What remains are some implementation details that are important to make the algorithm work in practice. This chapter contains the collection of implementation details that together with the framework described in previous chapters make up a complete functional navigation algorithm. First, we will explain the sensor models used for both data aggregation and forward simulation. Then, we will explain how we compute the cost maps that the low level (navigation level) path planner uses to navigate. Third, we will explain how we further optimize the forward simulation computational cost. Finally, we will present an overview of the complete system.

1. Sensor Model

For the forward simulation module in the algorithm and for the data aggregation into the navigation map, we use a simplified sensor model (Figure 6.1). The mobility sensing system is modeled after close proximity range sensors. The local visibility is looked up (details later in this chapter) and the terrain is retrieved from the visible cells. For these visible cells, the gradient is computed. The local gradient is then inserted into the navigation map with full (1.0) confidence. Similar for the midrange data, with the following difference: Only a small but longer segment of the precomputed visible cells is used for computing the terrain gradient. The

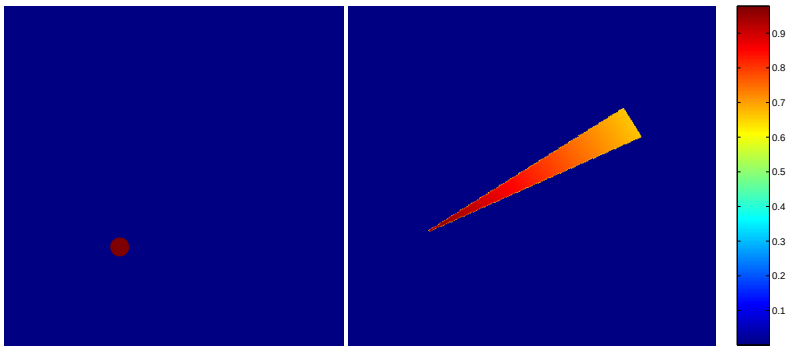


FIGURE 6.1. On the left, the sensor model used for the mobility sensing system. On the right, the sensor model used for the mid-range sensor. The color-bar indicates the confidence levels for both diagrams.

shape of this segment corresponds to the range, field of view and sensor orientation of the mid-range sensor modeled. In addition, a linear decay as a function of the measured range is used for the confidence model. Typically we use a value of 0.9 for the data points close to the robot and 0.6 for data points at the end of the sensor radius.

2. Computing Navigation Maps

The low level path planner [116] that is in charge of getting the robot from its current location to the next observation position uses a normalized cost map to find a path to the goal. Since the data gathered from the mid-range and mobility sensing systems have different properties, we use different techniques to integrate both into the same navigation map. Let us first define the separate cost:

ρ is the actual cost of traversing the cell based on data from the short range obstacle detection system. This is the cost that is used in the simplest mobility system in which mobility sensors, such as LADAR or stereo, insert obstacles in the world map. In practice we use a normalized and smoothed version of the observed terrain gradient:

$$\rho = f\left(\frac{|\nabla\left(G_\sigma \otimes \text{ObservedTerrain}\right)|}{\text{ObstacleThreshold}}\right)$$

with

$$f(x) = \begin{cases} x & \text{if } x < 1 \\ 1 & \text{otherwise} \end{cases}$$

While this is a simple cost model for traversability, this function can be arbitrarily complex and might encode many different other costs, our framework does not restrict the choice of this model [101, 114].

φ expresses our knowledge about the traversability of a grid cell as discovered by the mid-range sensing system. This cost is similar to ρ in that it encodes the local traversability of the terrain. However since we are only interested in the course layout of the environment and since the resolution of mid-range data is very low, we use only a binary representation $[\text{free}, \text{obstacle}]$. In addition, we also paint empty space from the sensor location up to the obstacle that was measured. Figure 6.2 shows an example of such a cost map. It is useful to separate the two costs because the mid-range data is less reliable.

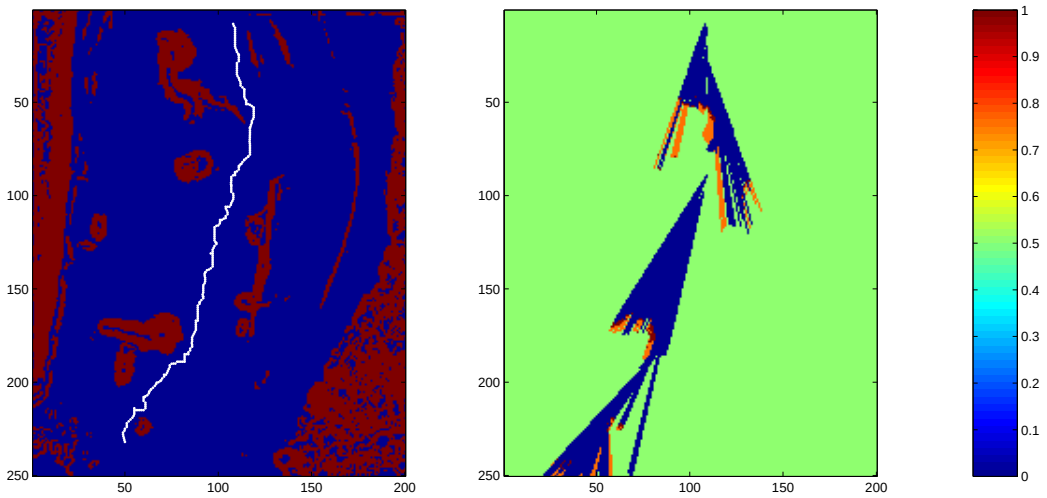


FIGURE 6.2. On the left, the ground truth obstacle labelling, the robot traverses the path (in white) from top to bottom. On the right, the mid-range sensor measurements and their expansions.

The final navigation map can now be computed as follows:

$$\text{NavMap}_{ij} = \begin{cases} \varphi_{ij} & \text{if } P(\varphi_{ij}) > 0.5 \\ \rho_{ij} & \text{otherwise} \end{cases} \tag{6.1}$$

3. Earliest Observation Position

To reduce the amount of computation, we can ask whether all the locations in the map need to be evaluated for potential observation locations. We can use the observation that the planner can find a cheaper detour if an obstruction is detected in an early stage. Hence there is no need to evaluate the utility for observation locations that are not on the boundary of the visibility contour (Figure 6.3).

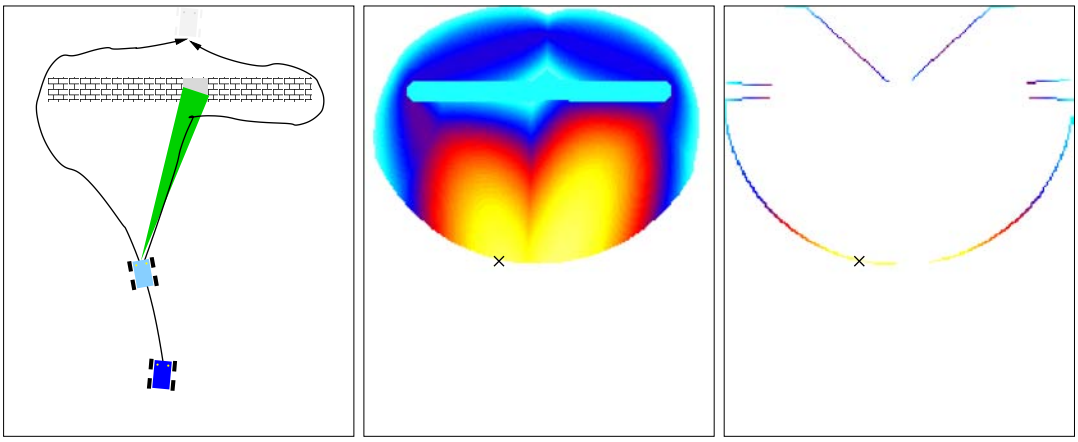


FIGURE 6.3. Example robot navigation scenario and the corresponding utility map. It is sufficient to evaluate only the earliest observation points (shown on the right) since the observation location with the maximum utility (marked by the $\times$) will be laying on this contour.

Based on the previous argument, and the heuristic from previous chapter that inferred obstacles are of interest, we can decrease the number of sites to be evaluated. The online version of our algorithm computes only the utility for those locations that are within mid-range range of inferred obstacles.

4. System Overview

The algorithm presented so far has only dealt with the high level mid-range sensing planning loop. The low level loop navigates the robot using the mobility sensing information. In principle we could forward simulate and plan for waypoints continuously. However, the amount of computation involved would unnecessarily slow down the low level path planning cycle.

The reason why we do not need to reevaluate possible vantage locations continuously comes from the fact that the changes caused by gathering more data from the mobility sensing system are small and very close to the robot. The mid-range view point planner benefits therefore little from this information, and need only to be run when new mid-range data comes available.

The mid-range sensing planning loop is therefore only invoked when we reach a waypoint (Figure 6.4). There are however two exceptions. First, when the system is started, we allow the robot to collect some initial mid-range data to bootstrap the inference process. Second, to insure the inference mechanism is not starved from data we explicitly take a measurement and reevaluate after a certain timeout period. This timeout is further explained in Chapter 7, Section 2.2.

In the robot experiments, the low-level navigation loop is executed on the robot computer, while the high level forward simulation is executed remotely. The loose coupling between the two loops allows for running this computationally more intensive task on a remote machine.

To further illustrate the inner workings of the algorithm, we have included the illustrations Figure 6.5 and Figure 6.6 that show how the algorithm navigates from one waypoint to another. The experimental results that will be presented in the next chapter are similar to the example presented here, however they are not discussed for each run individually. What is shown in the example is a typical viewpoint planning cycle. First some initial range measurements are acquired to bootstrap the forward simulation. Then the inference procedure produces the anticipated obstacles, which are used in turn to evaluate the alternative paths and select the best intermediate observation location.

We can zoom in even further and take a detailed look at one single evaluation (Figure 6.7). Since the computation is driven by the hallucinated obstacles, we start our *"what-if"* analysis by taking such an inferred obstacle and consider all the observation locations from which it can be observed (these locations are marked with magenta in the diagram). First we compute the paths from our current location to the goal with the obstacle present (r_1, left panel in red) and without (r_2, Figure 6.7

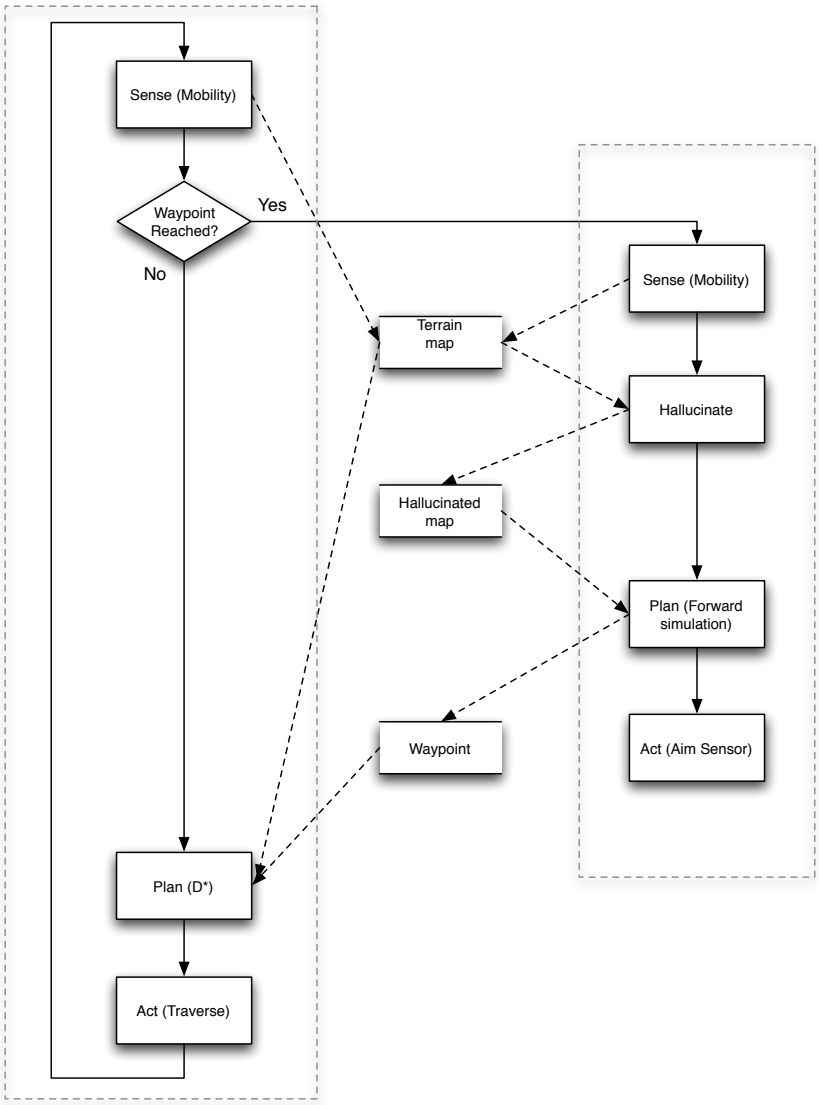


FIGURE 6.4. Mid-range planning sensing algorithm overview. The left-most flow is the low level navigation loop. The flow on the right shows the flow of the algorithm that does the forward simulation and sensor planning. Solid lines denote control flow of the algorithm, while dashed lines denote data transfer. Data items that have arrows entering on the sides are receiving incremental updates, whereas a top entry depicts the creation of the item.

left panel in green). If these two paths are the same, we need not to worry about this obstacle since it does not influence our path.

However, if these two paths are different, we compute the path that follows if we observe the obstacle late (r_3, left panel in yellow). We also compute the paths

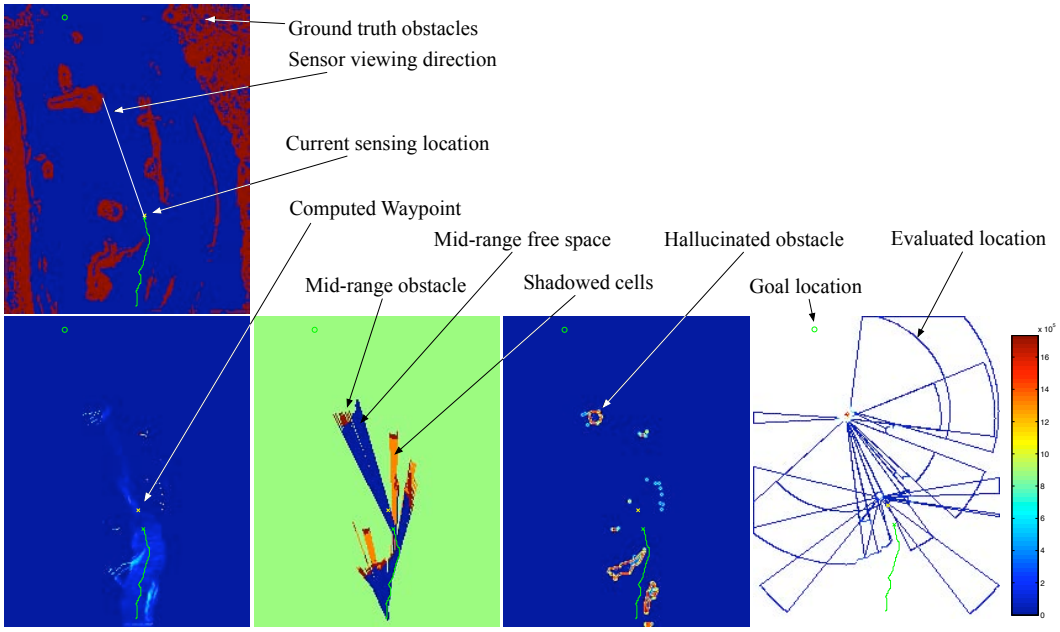


FIGURE 6.5. The diagrams that are used in the example presented in Figure 6.6 use the notation that is detailed out here. Counter clockwise from the top left is depicted: Terrain map (ground truth), Observed terrain map, Mid-range pencils of rays, binary hallucinated obstacle map and the utility map.

that first go through an observation location and then to the goal taking into account the observation. Observing the inferred obstacle at the sensed location yields s_1 (right panel in green), and observing free space leads to s_2 (right panel in red). With these paths s_1 and s_2 and their costs C_{ns} and C_s known, the utility ψ for this obstacle and observation location is computed. If as a result of another hallucinated obstacle a higher utility was already computed for this observation location, the previous value is retained.

Path Evaluation

Waypoints are in principle selected as the observation location with the highest utility. However, since we would like to enforce that we occasionally take a measurement and recompute the utility, a waypoint is returned within a certain timeout interval. This strategy will prevent the robot from travelling without sensing for a long period of time and being surprised by an obstacle (see also Chapter 7, Section 2.2).

Reusing computation results, to speed up the computation, all the path computation is cached. Specifically we can reuse much of the D^* [116] computation by keeping separate D^* navigation maps for the goal location and the current location. By setting each of these positions as goal location in its respective navigation maps, D^* can efficiently reuse most of the computation from previous queries to that location.

Over the course of evaluating waypoints as beneficial vantage points, the current and goal locations are fixed. We can thus exploit the way that D^* computes paths. With the separate maps for goal and current locations, D^* can keep all the heuristic values from previous path computations. Therefore D^* only needs to incrementally expand a few more nodes in the previous map for answering new queries.

Visibility computation as used for simulating range measurements is performed for every cycle, but the sensor masks and pixel indexes are all precomputed once upon initialization of the algorithm. The visibility computation uses a standard ray-casting algorithm [29]. Possible improvements from using graphics hardware for computing visibility can be achieved by indexing a 2.5D terrain surface model with a unique color that is generated as a hash from the pixel location. When we render in hardware a view of this terrain, the pixel colors in this rendered image will identify all visible terrain locations from the location that view was rendered. This enhancement was not included in the system. Relative indexes that map the sensor footprint are precomputed for a finite set of small orientation angles. These are then stored in a fixed size hash table using the angle as a hash. Finding all the pixels in a particular wedge is now a constant time operation.

Frequency of evaluation controls the forward simulation loop that through the timeout parameter. This parameter is used to enforce a mid-range sensor measurement after a certain number of cycles of the inner loop. Typically this value is set such that a mid-range sensor measurement is planned within a distance of

about twice the radius of the mobility sensing system. See also Chapter 7, Section 2.2 for the influence of this timeout parameter on the overall performance of the algorithm.

Homotopic paths are computed using a modified plane-sweep technique. A radius originating at the current location is swept clockwise through the binary obstacle-map giving rise to events at obstacle boundaries. If two obstacle pixels in the scan are separated by an empty pixel, a gap event is generated. In addition if two neighboring obstacle pixels in the scan are not neighboring obstacle pixels in the map, another gap event is generated.

For each gap we compute a homotopic class representative by closing all the gaps but one. We then compute a path through this open gap and repeat this procedure for each gap until for all gaps a path have been computed.

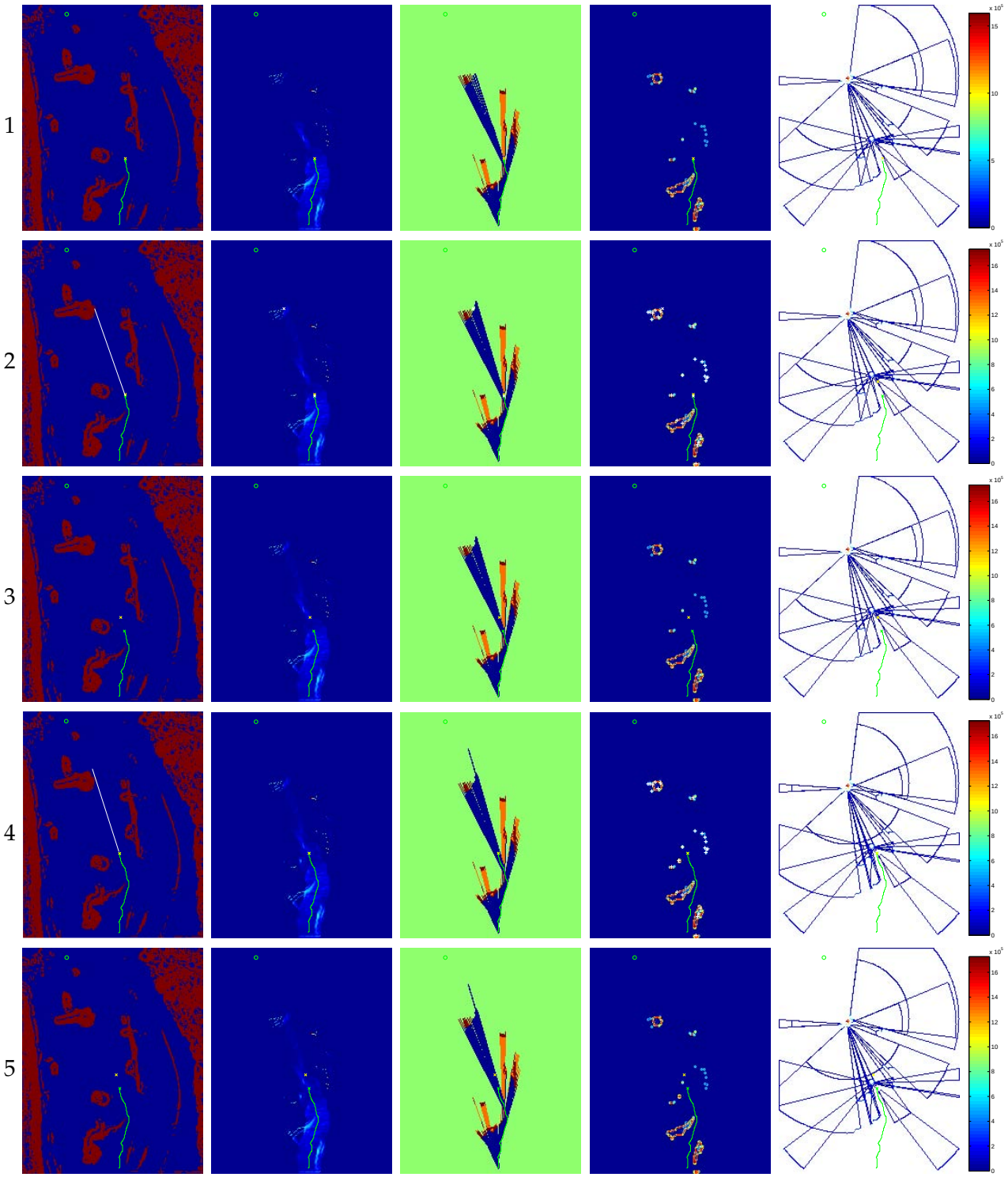


FIGURE 6.6. In row (1), the robot is about to arrive at a waypoint. This waypoint is reached and depicted in row (2). The robot takes a mid-range sensor reading in the shown direction. It then plans for a new waypoint, which is shown in row (3). This new waypoint is reached in row (4) and the cycle repeats.

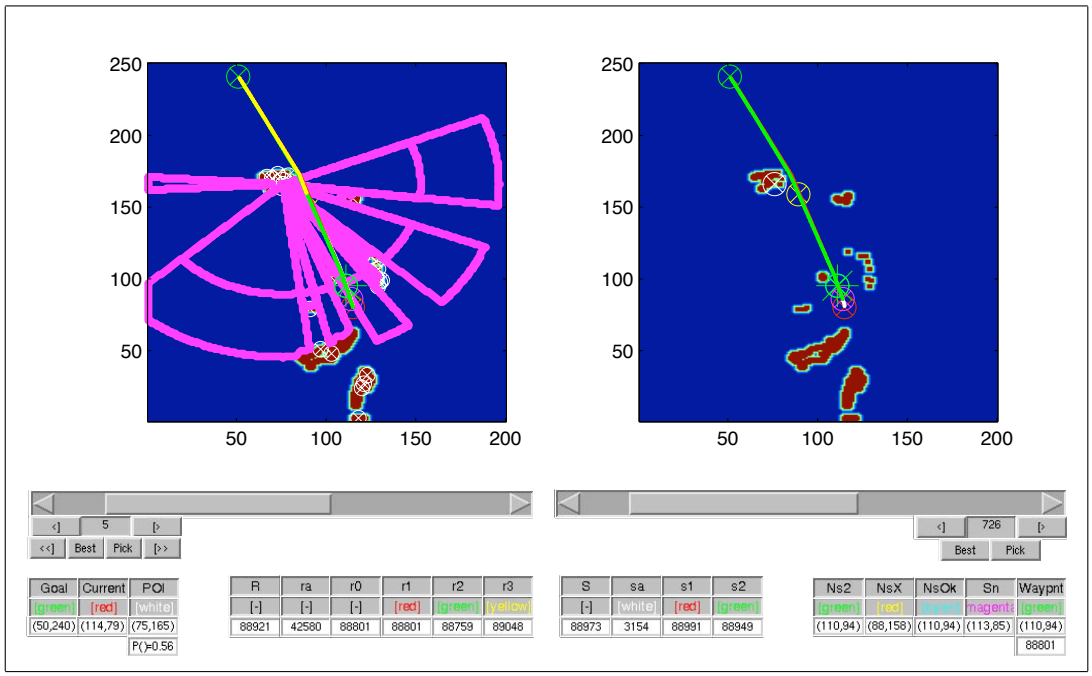


FIGURE 6.7. In our data display utility, we can see how the paths are computed. Shown here are the paths from one single evaluation location out of the many displayed on row (2) in Figure 6.6. In the displays, we only plotted those paths that correspond to the obstacle cells belonging to the segment marked with a white $\otimes$. All the cells marked in magenta have been evaluated since they are within range of observing the hallucinated obstacle. The view point planning algorithm has selected the location marked with a green $\otimes$ as an intermediate waypoint. The labelling of the paths follows the same convention as in Figure 5.2.

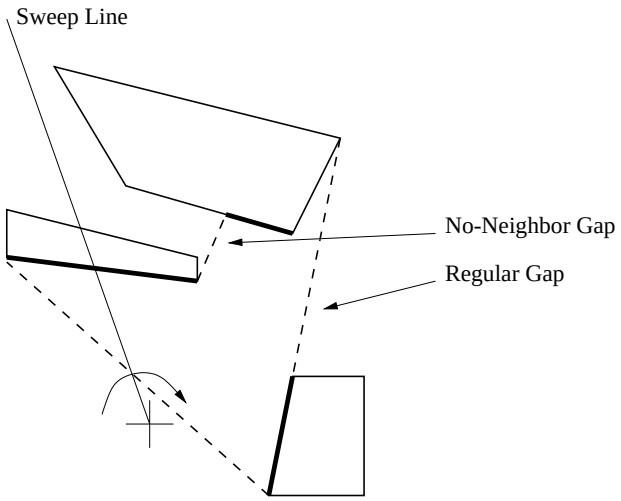


FIGURE 6.8. A radial scan line sweeps a binary obstacle map. Events are generated if two obstacle pixels in the scan are separated by an empty pixel and also if two neighboring obstacle pixels in the scan are not neighboring obstacle pixels in the map.

CHAPTER 7

Empirical Evaluation

To validate the proposed algorithm various experiments were carried out. These include results from field tests with an autonomous vehicle, as well as results from a much larger set of simulated experiments. The following sections will describe these experiments and show their results.

In the first section, we describe a set of experiments that serve as an example for the applicability of the algorithm. The example experiments described were carried out with real robotic platforms in outdoor environments. The first of these example experiments is using mid-range data from a wide-baseline stereo algorithm in a mostly structured environment. The second and third examples describe the usage of mid-range data from a laser range finder in a somewhat unstructured environment.

The second section looks in more detail at the performance of the algorithm and the influence of the parameters. To be able to evaluate performance and the influence of parameters properly, the algorithm was used in a controlled environment (simulation mode). All the simulation experiments use real terrain data for ground truth reference. The example simulation batch that is presented first is described in detail, whereas the following examples are summarized.

1. Example Experiments with Laser Range Finder and Wide-baseline Stereo

Since we have discussed two different types of mid-range sensing methods, we have conducted experiments for both of these methods. First, we will present some examples in which we use range data from a wide-baseline stereo setup. Second we will show some results from using laser range finders.

1.1. Experiment with Wide-baseline Stereo

For this example experiment in an outdoor environment we compare our approach to the standard proximity sensing and planning approach.

Experimental Setup. In this example we had our Pioneer DX robot navigate across the parking lot to find a goal behind a building. The Pioneer is equipped with a single camera, (the second image was captured manually at a fixed three meter baseline to collect wide-base line stereo images).

From these images a sparse set of 3D points were computed with a wide baseline stereo algorithm [83, 118]. The points were then used as the single source of information in an off-line algorithm to compute the next best vantage point. The off-line navigation and view planning algorithm used a camera field of view that was identical to the camera on the robot. With this particular configuration of baseline, range measurements of up to 100M can be acquired. Figure 7.1 shows a typical image pair that was used to compute this data[1]. See Table 7.1 for experimental details.

Map size x (M)	Map size y (M)	Resolution M/cell	Distance Start-Goal	Mobility Range (M)	Mid-range Range (M)	FOV mid-range degrees	Timeout Cycles	β $[0 \dots 1]$
100	100	0.5	45	1	100	45	20	0.7

TABLE 7.1. Smith Hall parking lot run, experiment parameters.

Results. The path that the robot followed in this example is indicated by a red trace in Figure 7.2. The mid-range sensing/planning approach did detect that

[1] see also appendix A

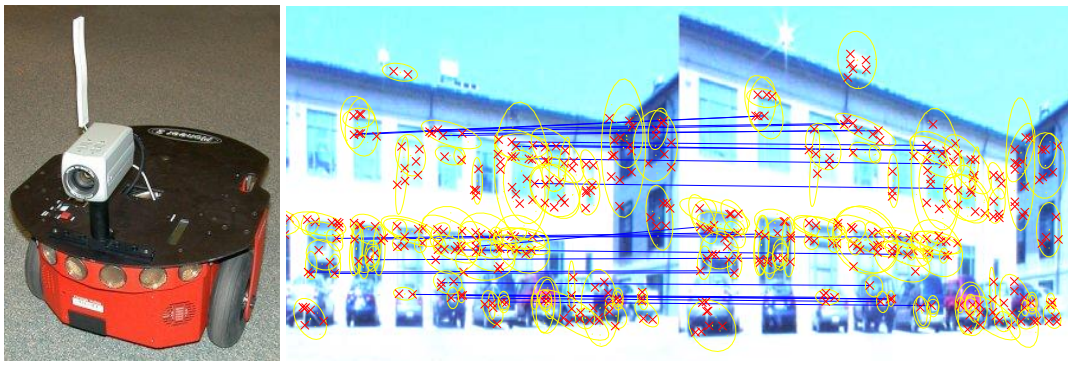


FIGURE 7.1. The Pioneer mobile robot with camera mounted is shown on the left. Shown on the right is the example range data that was computed by a wide-baseline stereo system, using a region based approach (See also appendix A).

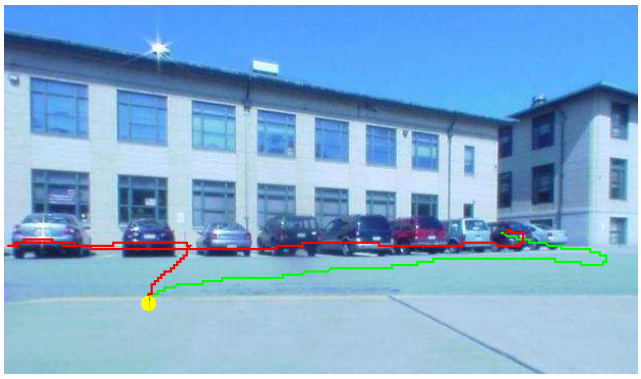


FIGURE 7.2. The experiment was conducted in the parking lot, with the start in front of the building and the goal (out of view) behind it. The red trace sketches the path as computed by a mobility only approach, the green trace shows approximately the path from the mid-range sensing approach.

the straight path to the goal was blocked and found immediately a passage around the obstacle, as indicated by the green trace in Figure 7.3.

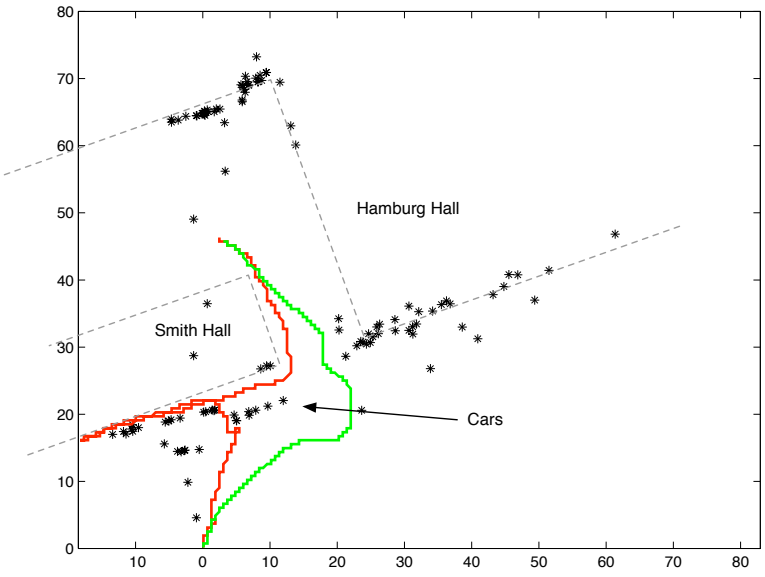


FIGURE 7.3. a birds-eye view of the example test scenario in a structured environment, in which the mobility sensing navigation skirts around the building (red) and the Mid-range sensing/planning algorithm (green) finds a passage and heads straight towards it.

1.2. Experiment with Laser Range Finder

Further experiments were done with a robotic Deere eGator (Figure 7.4). This allowed us to test our approach in much more natural environments.

Experimental Setup. The autonomous vehicle that we have been using for these experiments is an eGator from Deere. This vehicle was modified by the National Robotics Engineering Consortium to incorporate sensing, computation and actuation for autonomous navigation.

FIGURE 7.4. The robotic Deere eGator, equipped with a nodding laser range finder which gives us range measurements up to 60M in a 100° horizontal and 45° vertical FOV.

Our algorithm sits right on top of the vehicle navigator and receives its sensing data unprocessed from the vehicles sensing system. Since we do not have a mid-range type of sensor on the vehicle, we have simulated mid-range sensing in the following way:

Range: The range measurements are simulated by cutting scanner data off at 4M for the mobility system, the mid-range system is providing the full sensor range.

FOV: The Field of view of the sensor is limited to 6° by artificially selecting slices from the scan. Aiming the sensor is done in a similar matter, the slice that corresponds to the desired angle is returned.

Examples. Included are two experiments that were conducted with the eGator. The first experiment was staged on the "Cut" which is a grassy area that has a few trees and some other non-natural obstacles (Figure 7.5). The second example was set in Shenley park, this area is slightly larger and contains a higher density of obstacles.

FIGURE 7.5. Example experiment "On the Cut": In the left display: the robot is at its starting position, the path marked in green is planned by the mid-range sensing and planning framework. It runs through the planned vantage points (marked yellow) where a mid-range sensor measurement is taken. The path marked in red is taken if only mobility sensing was used. On the right: the robot has reached its goal position which is behind the initial obstruction.

Map size x (M)	Map size y (M)	Resolution M/cell	Distance Start-Goal	Mobility Range (M)	Mid-range Range (M)	FOV mid-range degrees	Timeout Cycles	β $[0 \ldots 1]$
150	150	0.5	75	4	40	6	20	0.7

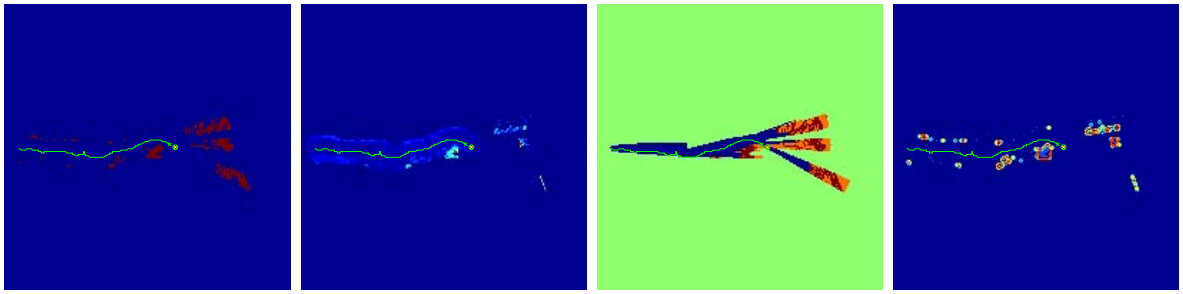
TABLE 7.2. "On the Cut", experiment parameters.

Map size x (M)	Map size y (M)	Resolution M/cell	Average distance Start-Goal	Mobility Range (M)	Mid-range Range (M)	FOV mid-range degrees	Timeout Cycles	β $[0 \ldots 1]$
150	150	0.5	140	4	40	6	20	0.7

TABLE 7.3. "In Shenley park", experiment parameters.

1.3. Discussion

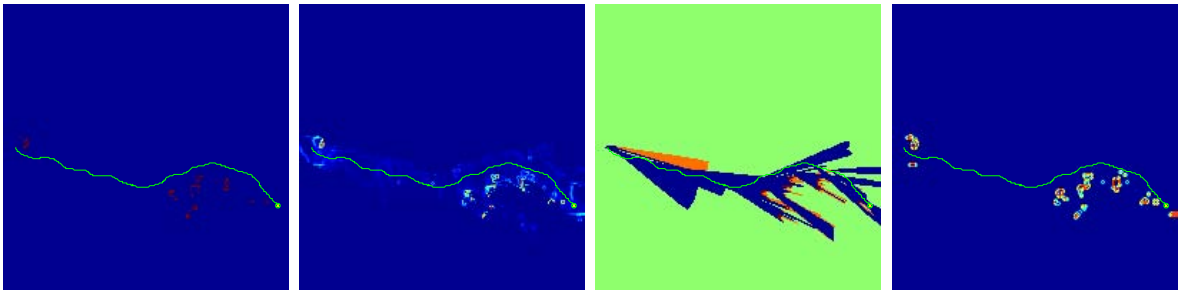
The examples show that our presented algorithm can successfully navigate autonomous vehicles in unknown terrain. The inference mechanism, the core of the forward simulation view planning and navigation strategy, assures that the most likely inferred obstacles will be observed.



(a) (b) (c) (d)

FIGURE 7.6. Example experiment "On the Cut": The internal maps of the mid-range sensing and planning algorithm are shown after the robot completed its run. In (a), the binary obstacle map as sensed during the run are shown. (b) shows the observed gradient from the mobility sensing system. The mid-range data is displayed in display (c). The hallucinated obstacles are shown in panel (d).

FIGURE 7.7. Example experiment "In Shenley park": In the left display: the robot is at its starting position, the path marked in green is planned by the mid-range sensing and planning framework. It runs through the planned vantage points (yellow marked) where a mid-range sensor measurement is taken. On the right: the robot has reached its goal position which is behind the initial obstruction.



(a) (b) (c) (d)

FIGURE 7.8. Example experiment "In Shenley park": The internal maps of the mid-range sensing and planning algorithm are shown after the robot completed its run. In (a), the binary obstacle map as sensed during the run are shown. (b) shows the observed gradient from the mobility sensing system. The mid-range data is displayed in display (c). The hallucinated obstacles are shown in panel (d).

2. Detailed Analysis

After the encouraging examples from previous section, we will now take a closer look at the performance of the algorithm. We describe an experimental protocol in the following few sections that allows us to run controlled experiments for a substantial number of runs on three different terrain datasets.

First we will describe how these experiments are carried out. Emphasis is on the experimental procedure. We then discuss in detail an experiment that shows clearly how the algorithm performs. This is followed with a discussion of the influence of the parameters used in our experiments which is supported by some example experiments. We then continue with the discussion of a few experiments that show results on publicly available terrain data.

2.1. Overview

For running our controlled experiments, we have setup up simulator that uses 2.5D elevation maps. We have successfully used maps from both the U.S. Geological Survey and also high resolution maps from the CMU Helicopter Lab and the CMU 3D Computer Vision Group [119, 120].

These maps are used in simulation as ground truth, the simulated vehicle will sense by using the same sensor model as used for the forward simulation (Chapter 6, Section 1) from this ground truth terrain map. Most of the forward simulation framework doubles as simulator in which the algorithm is evaluated.

The robot itself is modeled as a single point, and can freely move without any restrictions in the plane. This does not pose a serious limitation, since the view planning navigation algorithm we have presented is a high level planner, and we could ultimately use low level vehicle controller that does incorporate vehicle dynamics [54]. Because the robot is being modeled as a single point, its motion commands reduce to updating a system wide parameter for the current location. Sensing actions are performed using this system variable.

Typically an experiment is setup in the following way. Fist we acquire terrain data which is then scaled such that it matches the scale we would otherwise have sensed with. to match a realistic obstacle gradient threshold. With this parameter known, a script will then generate the largest connected traversable component for which all coordinates are stored. This set of components is then used to randomly draw the number of goal and start locations. These might also include trivial ones such as shown in Figure 7.11 on the left. Typically paths that span less than twice the range of the mobility sensing system are discarded since they tend to represent trivial scenarios in which no planning would be required.

We use then another script to generate the initialization files that include the settings for the simulator. This script includes computing the priors for the obstacles and free space. The priors for each of the two classes are computed by a standard nonlinear least squares fit of the terrain labels and gradient to a Gaussian distribution. The simulator is then started with the initialization file and data collection begins.

Depending on the verbosity level in which the simulator is initialized, we can either examine in full detail each single path evaluation (as shown in Figure 5.2) or summarize the results.

2.2. Analysis

Additional experiments were conducted using the environment as displayed in Figure 7.9. This terrain data from an open coal mine was collected by the CMU Helicopter Lab and is of very high spatial resolution. We sampled it down to a resolution of 0.5M/cell. See Table 7.4.

For a controlled experiment, we ran simulations of the system for different start and goal locations and different system configurations. A complete elevation map of the environment is used for generating sensor data in the simulation, but it is not known by the robot initially.

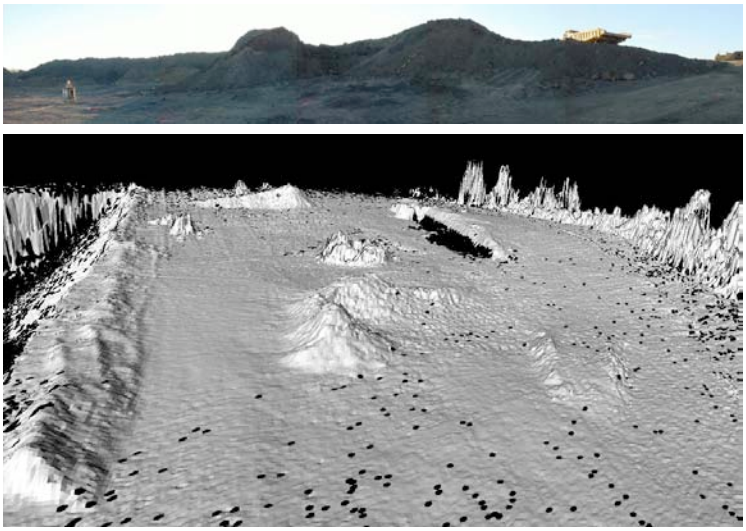


FIGURE 7.9. Top: a panorama view of the open coal mine test site from which an elevation map was collected. Bottom: a rendering of this elevation map.

Map size x (M)	Map size y (M)	Resolution M/cell	Average distance Start-Goal	Mobility Range (M)	Mid-range Range (M)	FOV mid-range degrees	Timeout Cycles	β $[0 \ldots 1]$
235	258	1	121	12	120	5	18	0.7

TABLE 7.4. Open coal mine data-set, experiment parameters.

We have executed 500 trial runs with randomly chosen start and goal positions, given the constraint that there exists a path from start to goal.

We analyze the data from these runs in three ways. First, we compare the lengths of the paths generated by using the mid-range sensor planning method with the paths executed by using mobility sensing only. This is shown in Figure 7.10, in which the runs are sorted in the order of increasing gain for the sensor-based planning method. This first type of analysis is necessary to assess the amount gained from using sensor planning. It is important to note that the gain can vary dramatically depending on the start and goal points. Intuitively, little gain can be expected if the area between the start and goal points is completely unobstructed, in which case *any* planning strategy would perform well.

The graph shows clearly that for an unobstructed path, the algorithm leads to a slightly longer path. This is because the vehicle might veer off the "path" in order to get better coverage (Figure 7.11). On average the gain in path length is

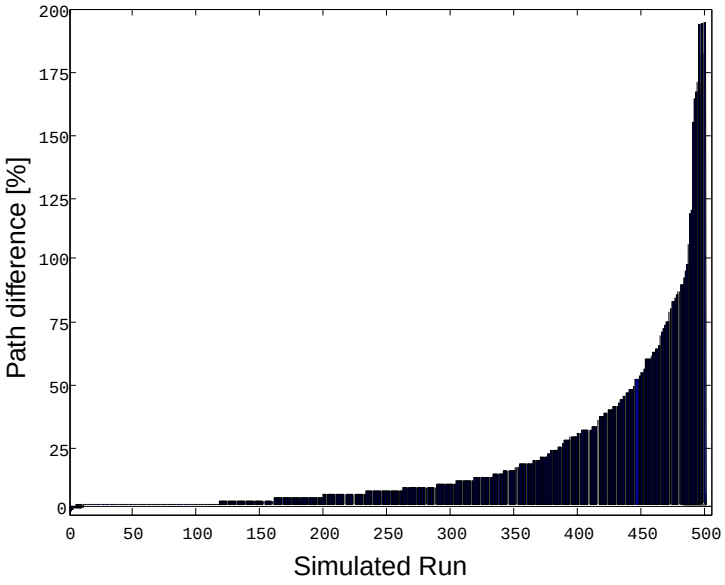
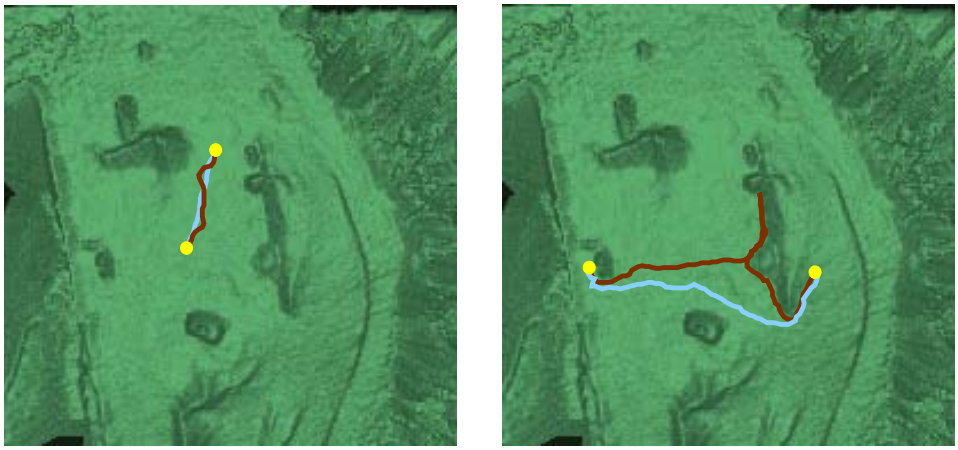


FIGURE 7.10. The Mid-range planning sensing algorithm compared to the standard navigation approach. A positive percentage shows how much shorter the path from the mid-range sense/planning is over the mobility only approach. The average gain in path length is +18%. However, more interesting is that +76% of the runs exhibit positive gain (up to almost +200%) and that of those that have negative gain, the maximum loss is -6%. This is due to the overhead introduced by the explore behavior that tries to find better paths. For 22% of the runs there was no gain or loss.

+18%. Also, 76% of the runs exhibit positive gain (up to almost 200%). And of those that have negative gain, the maximum loss is only -6%, furthermore, as the left illustration in Figure 7.11 shows, these cases are those for which the paths in the environment are unobstructed. A representative sample of runs was randomly selected from these 500 individual runs and plotted in Figure 7.11.

Second, it is important to compare the paths obtained with our sensor planning heuristics with the plan generated by using a mid-range sensor that senses all the time in every direction, since our claim is that the algorithm generates a "good" selection of when/where to sense during motion of the robot. If our planner were to generate paths that have substantially higher cost than those generated by using the sense all the time/everywhere strategy, it would indicates that our heuristics can be improved. This part of the analysis is summarized in Figure 7.13, in which the lengths of the paths generated from our planning approach and from the sense all the time/everywhere approach are plotted (with $(+)$) as a scatter plot.



(a) (b)

FIGURE 7.11. Typically, bad performance is due to the fact that little gain can be expected if the area between the start and goal points is completely unobstructed. This is due to the fact that the vehicle might veer off the "path" in order to get better coverage (diagram on the left). Much gain can be found if a major obstruction is anticipated early (diagram on the right).

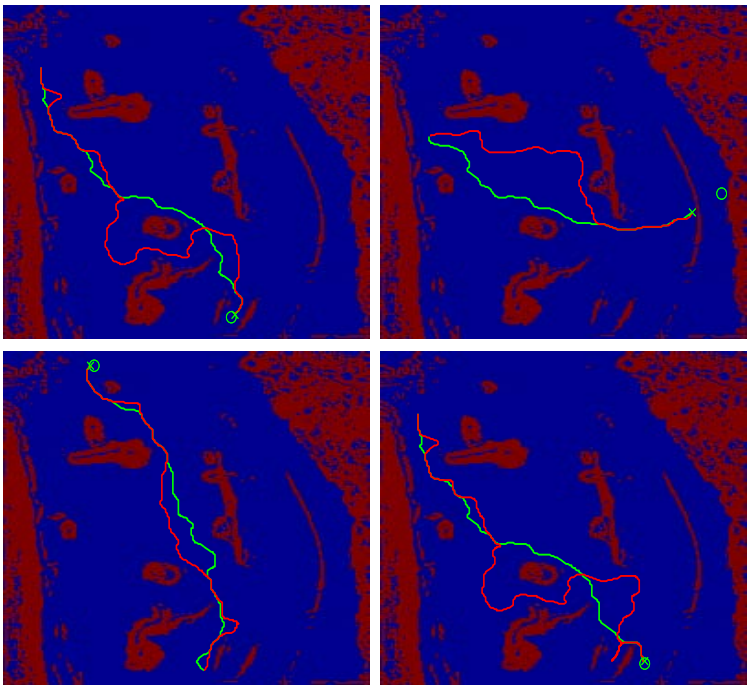


FIGURE 7.12. A random sample of runs drawn from the set of 500. In green are shown the paths resulting from our mid-range sensing and navigation framework. The paths shown in red are from a planner that has to rely on mobility only sensor data.

The plot indicate that the lengths are similar, i.e., the values are scattered near the diagonal (the correlation coefficient is $\rho = 0.99$). This result verifies empirically our hypothesis that continuous sensing of the environment is not necessary, provided that suitable heuristics are used for computing when and where to sense.

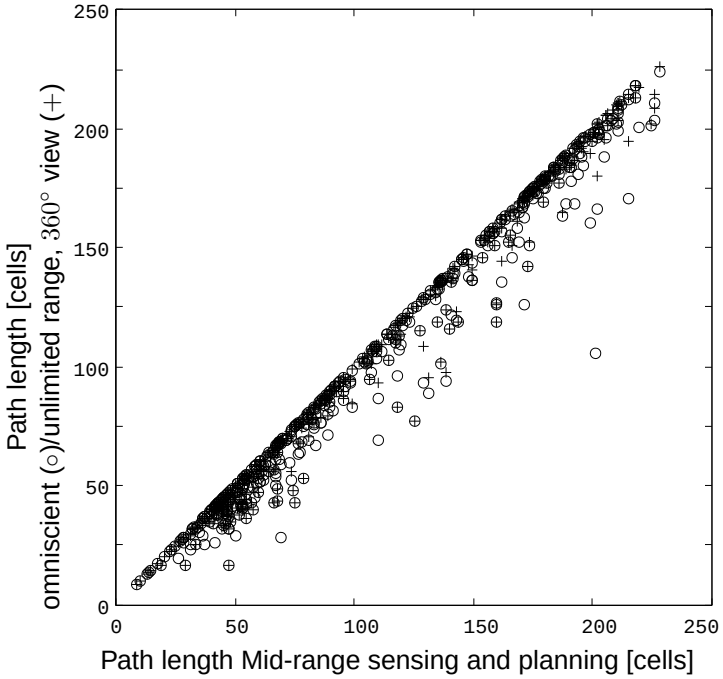


FIGURE 7.13. The Mid-range planning sensing algorithm compared in a scatter plot to a hypothetical continuous sensing method with an unlimited range, 360° view range finder (no viewpoint planning (+)). and also compared to the true shortest path (○). The vertical axis is the path length from the omniscient planner and also the planner using continuous sensing and planning. The horizontal axis is the path length from our Mid-range sensing/planning method. Results show that the performance of our mid-range sensing approach are almost as close to the optimal as they were to the continuous sensing planner.

Finally, a third type of analysis uses the paths generated by an omniscient planner that has complete knowledge of the entire map prior to execution as the baseline for comparison. Such an omniscient planner generates the shortest paths that can be generated given the environment and the selections of start and goal locations. As such, it is a useful baseline to quantify the degradation of performance due to limited sensor horizon. This analysis is shown in Figure 7.13 as well, in which the paths generated by using our mid-sensor planning strategy and the paths executed by omniscient planner (marked with ○) are plotted again as a scatter

plot. The graph shows that the paths generated with mid-range sensor planning are almost as close to the optimal from the omniscient planner as they were to the continuous sensing planner. Specifically, the correlation coefficient is $\rho = 0.98$ between the paths planned by the omniscient planner and our mid-range sensor planning approach.

In addition we have conducted experiments to determine the influence of the two main parameters of the system. The frequency of taking measurements (see also Chapter 6, Section 4) is determined by the timeout parameter. The timeout parameter enforces taking measurements even if the forward simulation does not expect that there is anything to sense. The following diagram (Figure 7.14), shows the result of the randomly drawn start and goal locations over a range of parameter settings. The plot suggests, that enforcing a measurement after a certain timeout seems to be unnecessary. However, there will be situations where not having had any data to infer from and not enforcing a measurement from time to time will lead to not having anticipated an obstacle and incurring a longer path. The timeout parameter has been varied in increments of 5 starting at the range of the mobility sensing system and extending well out into the range of the mid-range system.

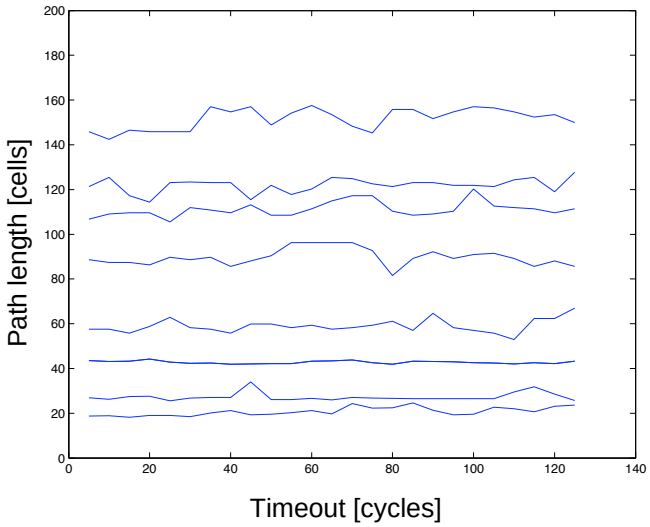


FIGURE 7.14. A random sample of start and goal tuples were evaluated over a wide range of the timeout parameter. Each line represents an individual experiment in which the start and goal situation are fixed, while the timeout parameter is changed.

Intuitively, one would expect the variation of this timeout parameter to have a substantial effect on the performance of the algorithm. However, after carefully studying these results it turns out that, in all these cases, the initial data that is collected at the start is sufficient to bootstrap the algorithm. To be more specific, when the system gets data initially the inference mechanism will keep us interested in looking at inferred obstacles until these are all confirmed and do not expand any further. At that time, we will have found an empty passage next to the obstacle and our laser range finder will have been targeted at this empty corridor. Moreover, when our ray hits this corridor it will have hit in all our cases another obstacle further away. This is also shown by example in Figure 6.2.

Having said that, it is easy to come up with a counter example that will break this hypothesis. If one would present a very sparse environment, with for example a single obstacle. The inference mechanism will not infer any new obstacles after the first obstacle have been steer clear from. If there would be yet another obstacle that was just out of range, then the algorithm will be surprised by this when it is eventually detected by the mobility sensing system.

In practice, this is however not a major issue, since we can set this timeout fairly high and still be safe, because we only need it to catch obstacles that are well beyond our mid-range sensing range.

The other algorithm parameter β controls the amount in which the obstacle inference mechanism weighs the importance of the labels from its neighbors. In Figure 7.15 and 7.16 the results of this example experiment are shown.

Figure 7.16 shows that even though the difference between the number of inferred cells is fairly high, the influence on the path is negligible. This behavior is expected, since the MRF models local structures especially well. The MRF will only influence large regions, if there is enough evidence to warrant a grouping, or the neighboring have a very weak confidence level.

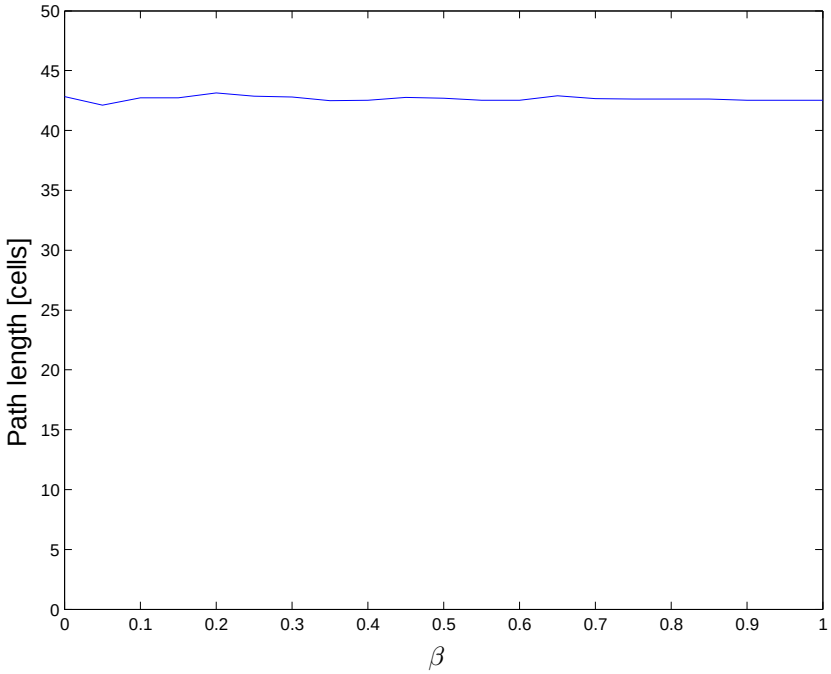


FIGURE 7.15. Variation of β on the final path length. In this example the neighborhood interaction term β was varied in steps of 0.05 over the whole range of the parameter space. The reason that β has a negligible effect on the path cost is that even when β is at its maximum, obstacles do not grow that much (See also Figure 7.16).

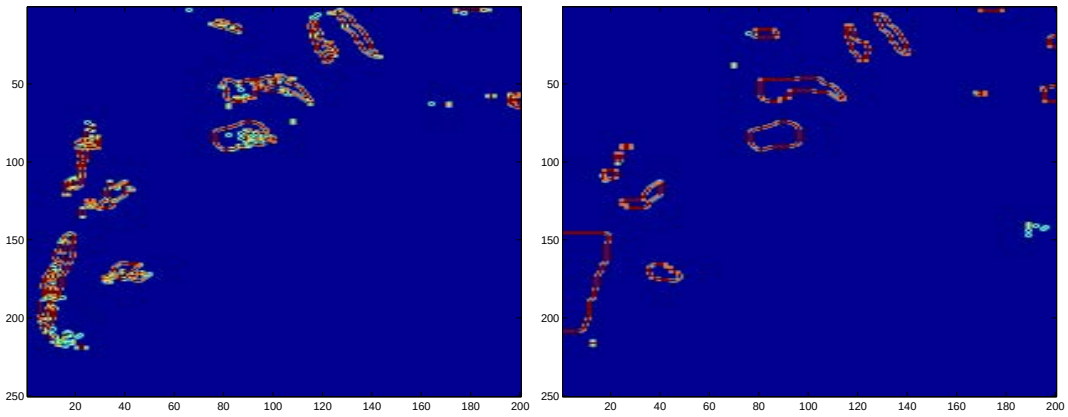


FIGURE 7.16. Obstacle inference as a function of the neighborhood interaction term β. On the left, obstacles were inferred with $\beta = 0$. On the right, inference with $\beta = 1$.

Much in the same flavor as the batch that was presented earlier in this section, we have acquired terrain maps from the USGS database and setup similar experiments. The examples shown here, have the same parameters (Table 7.5 and 7.6) but were generated independently.

The terrain map as rendered in Figure 7.17 is from the vicinity of Albuquerque, New Mexico, where as the second set uses a terrain patch from the vicinity of Flagstaff Arizona. The results are discussed in the same manner as before, but more briefly.

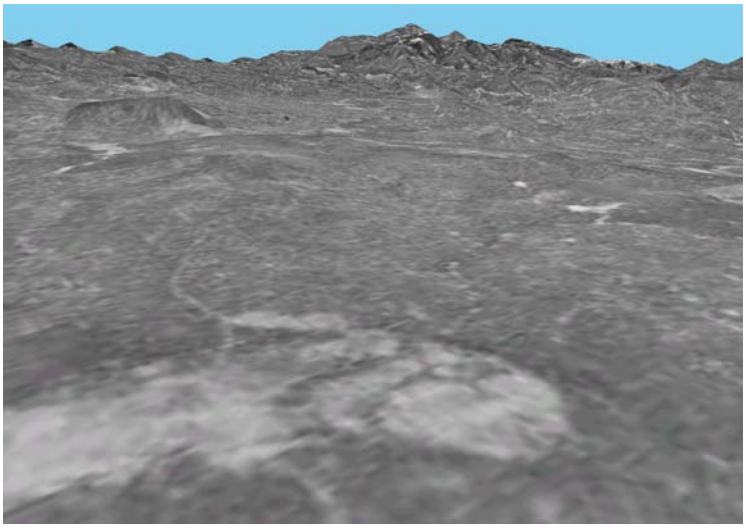


FIGURE 7.17. A rendering of the USGS elevation map from a terrain patch in the vicinity of Albuquerque, NM.

Map size x (M)	Map size y (M)	Resolution M/cell	Average distance Start-Goal	Mobility Range (M)	Mid-range Range (M)	FOV mid-range degrees	Timeout Cycles	β $[0 \ldots 1]$
3000	2000	10	634	40	1000	8	12	0.7

TABLE 7.5. New Mexico data-set, experiment parameters.

As before, we first compare the lengths of the paths generated by using the mid-range sensor planning method with the paths executed by using mobility sensing only. This is shown in Figure 7.18.

On average the gain in path length is +21%. Also, 75% of the runs exhibit positive gain (up to almost +200%). And of those that have negative gain, the maximum loss is -16%.

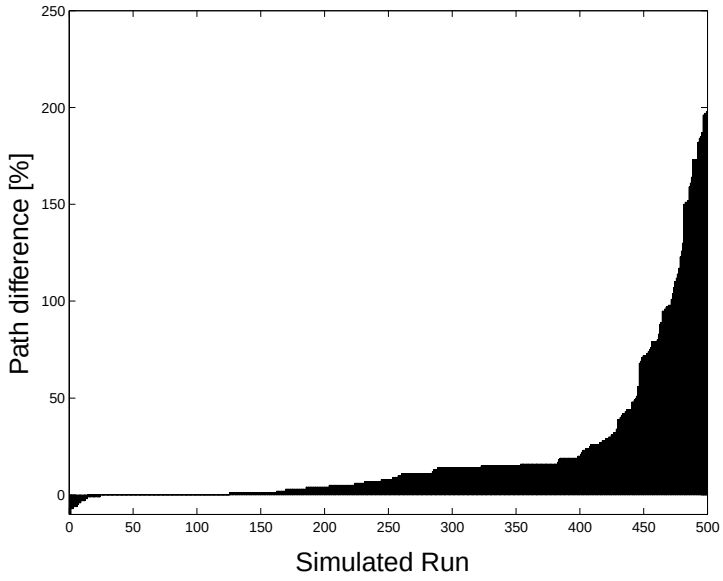


FIGURE 7.18. The Mid-range planning sensing algorithm compared to the standard navigation approach. A positive percentage shows how much shorter the path from the mid-range sense/planning is over the mobility only approach. The average gain in path length is +21%. In addition, +75% of the runs exhibit positive gain (up to almost +200%) and that of those that have negative gain, the maximum loss is -16%. This is due to the overhead introduced by the explore behavior that tries to find better paths (see also Figure 7.19). For 20% of the runs there was no gain or loss.

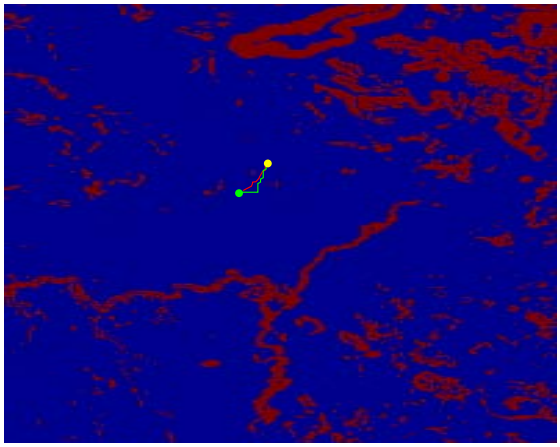


FIGURE 7.19. The path with a negative gain of 11% is due to the fact that the viewpoint that is planned is in comparison to the total path length far away. The start is marked yellow, the goal green.

Second, we compare the paths obtained with our sensor planning heuristics with the plan generated by using a mid-range sensor that senses all the time in every direction. The result of this analysis is summarized in Figure 7.20, in which the lengths of the paths generated from our planning approach and from the sense all the time/everywhere approach are plotted (with $(\times)$) as a scatter plot. The correlation coefficient for this experiment is ($\rho = 0.85$).

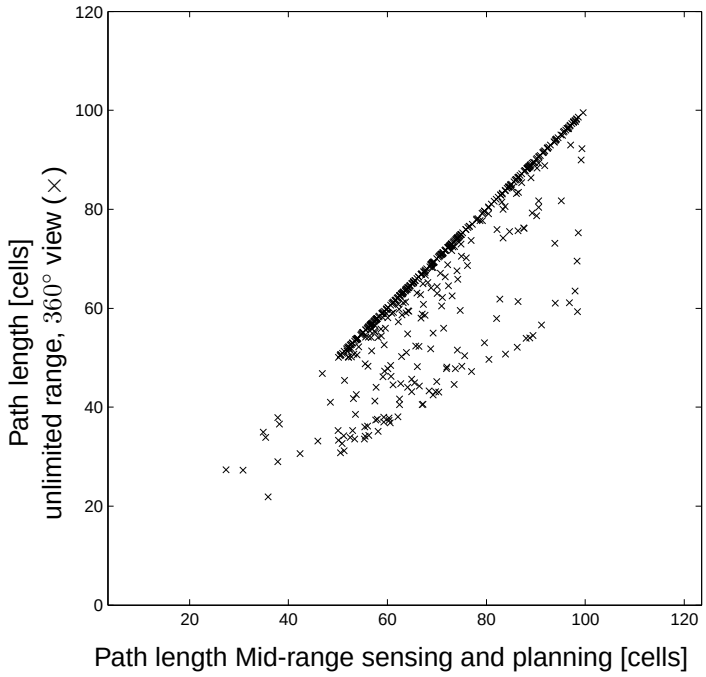


FIGURE 7.20. The Mid-range planning sensing algorithm compared in a scatter plot to a hypothetical continuous sensing method with an unlimited range, 360° view range finder (no viewpoint planning $(\times)$). The vertical axis is the path length for the planner using continuous sensing and planning. The horizontal axis is the path length from our Mid-range sensing/planning method.

Third, we use the paths generated by an omniscient planner that has complete knowledge of the entire map prior to execution as the baseline for comparison. This analysis is shown in Figure 7.21, in which the paths generated by using our mid-sensor planning strategy and the paths executed by omniscient planner (marked with $\circ$) are plotted again as a scatter plot. For this experiment, the correlation coefficient is $\rho = 0.77$.

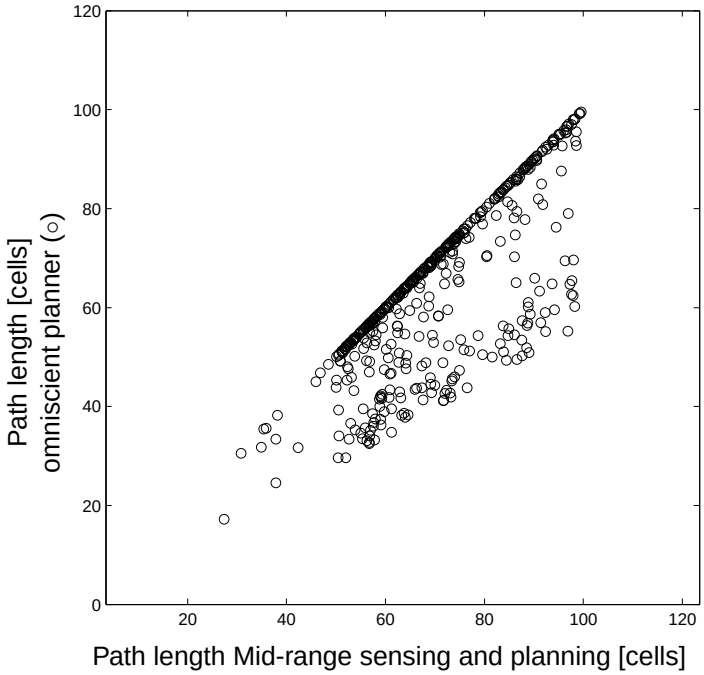


FIGURE 7.21. The Mid-range planning sensing algorithm compared in a scatter plot to the true shortest path (○). The vertical axis is the path length from the omniscient planner. The horizontal axis is the path length from our Mid-range sensing/planning method.

The second set of example experiments using a USGS database terrain patch from a location close to Flagstaff,AZ. The results from this experiment are presented in the Figures 7.23 to 7.25.

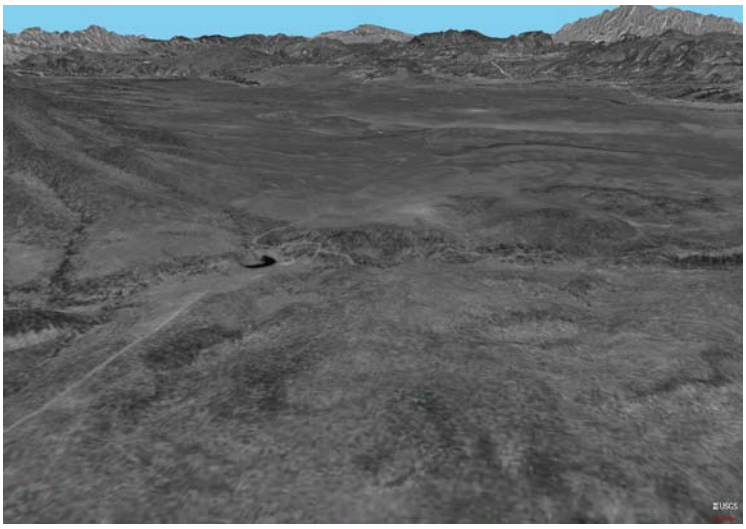


FIGURE 7.22. A rendering of the USGS elevation map from a terrain patch in the vicinity of Flagstaff,AZ.

Map size x (M)	Map size y (M)	Resolution M/cell	Average distance Start-Goal	Mobility Range (M)	Mid-range Range (M)	FOV mid-range degrees	Timeout Cycles	β $[0 \ldots 1]$
3000	2000	10	991	40	1000	8	12	0.7

TABLE 7.6. Arizona data-set, experiment parameters.

First, compare the lengths of the paths generated by using the mid-range sensor planning method with the paths executed by using mobility sensing only. This is shown in Figure 7.23.

On average the gain in path length is 19%. Also, 66% of the runs exhibit positive gain (up to almost 177%). And of those that have negative gain, the maximum loss is only -7%.

Second, we compare the paths obtained with our sensor planning heuristics with the plan generated by using a mid-range sensor that senses all the time in every direction. The result of this analysis is summarized in Figure 7.24, in which the lengths of the paths generated from our planning approach and from the sense all the time/everywhere approach are plotted (with $(\times)$) as a scatter plot. The correlation coefficient found for this experiment is $\rho = 0.98$).

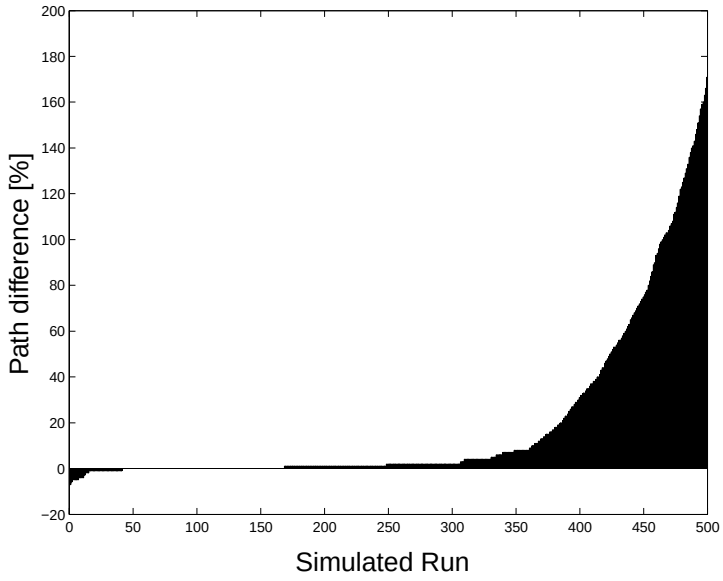


FIGURE 7.23. The Mid-range planning sensing algorithm compared to the standard navigation approach. A positive percentage shows how much shorter the path from the mid-range sense/planning is over the mobility only approach. The average gain in path length is +19%. In addition, 66% of the runs exhibit positive gain up to almost +177% and that of those that have negative gain, the maximum loss is -7%. This is due to the overhead introduced by the explore behavior that tries to find better paths. For 25% of the runs there was no gain or loss.

Third, we use the paths generated by an omniscient planner that has complete knowledge of the entire map prior to execution as the baseline for comparison. This analysis is shown in Figure 7.25, in which the paths generated by using our mid-sensor planning strategy and the paths executed by omniscient planner (marked with $\circ$) are plotted again as a scatter plot. For this experiment, the correlation coefficient is $\rho = 0.98$.

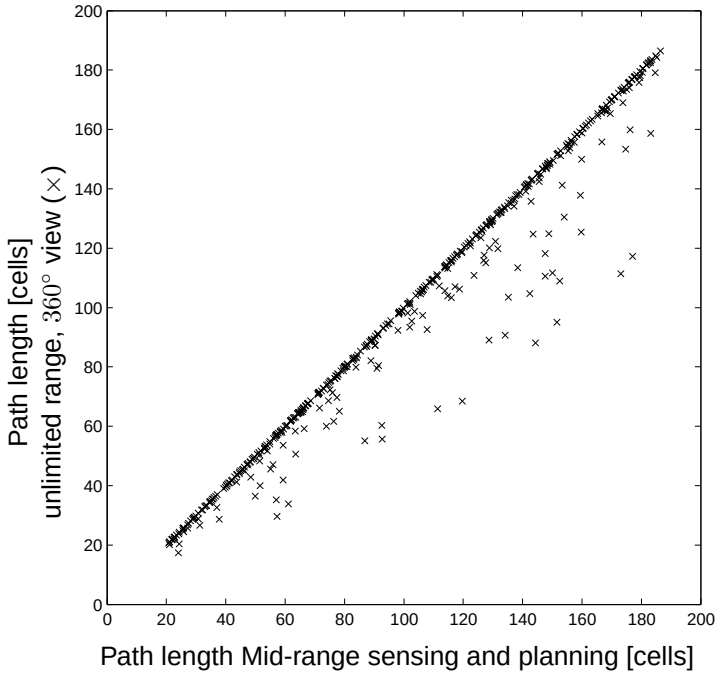


FIGURE 7.24. The Mid-range planning sensing algorithm compared in a scatter plot to a hypothetical continuous sensing method with an unlimited range, 360° view range finder (no viewpoint planning ($\times$). The vertical axis is the path length for the planner using continuous sensing and planning. The horizontal axis is the path length from our Mid-range sensing/planning method.

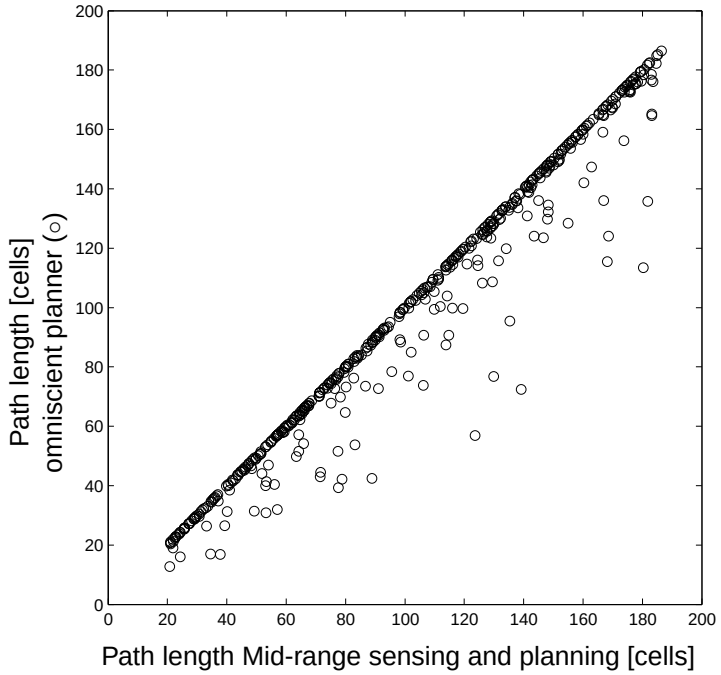


FIGURE 7.25. The Mid-range planning sensing algorithm compared in a scatter plot to the true shortest path (○). The vertical axis is the path length from the omniscient planner. The horizontal axis is the path length from our Mid-range sensing/planning method.

2.3. Discussion

We have shown examples in which our algorithm uses mid-range laser range finder data as well as data from a wide-baseline stereo system to successfully plan view points and navigate a real robot in an outdoor environment. In addition we have performed an extensive evaluation of the framework by means of experiments with a simulator that uses real terrain data.

Our analysis shows that the mid-range sensing and planning algorithm performs on average better than a mobility sensing only navigation approach. Moreover, opportunistically the algorithm has shown to outperform the mobility sensing only approach by a factor of two. In the few cases that our mid-range sensing and navigation is outperformed, the paths are trivial and a no sensing at all method would have performed as well.

If we compare our approach to an ideal, continuous unlimited range, unlimited field of view sensing approach, we often perform almost as good. Typically we find a correlation coefficient of 0.98.

CHAPTER 8

Discussion

NAVIGATION for ground vehicles is severely hindered by the limitations of sensor capabilities. The limited sensing range that is available often leads to inefficient paths because important terrain features or obstacles are not observed in a timely manner, so the navigation algorithm has no other choice than to skirt past these obstructions.

The use of mid-range sensing can alleviate this problem, since these types of sensors will provide data up to a few kilometers. However, applicability of these mid-range sensing devices is limited because of both geometric constraints and the type of data that is returned. Mid-range data is typically sparse and mid-range sensors need to be properly aimed at the region of interest such that the sensor's field of view is not limited by nearby obstructions.

In this thesis both of these issues are addressed. We propose a novel mid-range sensing and navigation strategy that relies on a new forward simulation technique that uses hallucinated worlds to determine "when and where" to look. This forward simulation is based on a novel application of a well studied inference mechanism in a probabilistic framework to reason about the most likely future world scenario. Based on this prediction, a cost benefit analysis is performed to determine which locations on our way to a goal destination could be used to acquire more mid-range data to aid navigation.

We demonstrate that the concept of forward simulation can be brought successfully into a working algorithm. This forward simulation framework evaluates alternative scenarios for navigation or sensing based on the most likely scenario. The algorithm is shown to be able to improve navigation of a robot in both simulated and real scenarios. The sensing and navigation strategy can opportunistically reduce the path cost by 50% in comparison with a mobility only navigation strategy. The algorithm is also shown to degrade gracefully. More specifically, the algorithm does not incur any significant extra cost by strategically planning waypoints.

Much work remains to be done. Most importantly we would like to extend our framework such that better prior models could be used. We have demonstrated examples on real autonomous vehicles.

The current implementation of the forward simulation is based upon a Markov Random Field (MRF) inference mechanism. Although quite powerful in expressing local spatial groupings, the MRF is not adequate to capture structures in real environments. Alternative inference mechanisms or possibly even a hierarchical multi resolution MRFs can be explored in the future such that more intricate structures could be inferred. Work in the pattern recognition community is directly related to this question, and new techniques from this field could possibly be tailored to suit our needs.

With respect to making the system ready for the field, more improvements may be made. The current system uses a fixed map. For an autonomous vehicle that might traverse long distances, this is somewhat impractical. This issue can be addressed by using so called “scrolling maps”. These maps stay centered around the vehicle while the vehicle travels. In addition, we would like to further reduce the cycle time, which could be achieved by using a multi-resolution representation of the maps. Forward simulation can initially be applied on a coarse scale and then in a higher resolution if necessary. The inference mechanism can likely benefit from this multi-resolution evaluation approach.

APPENDIX A

Wide baseline stereo for unstructured environments

RECOVERY of scene structure at long distances, *e.g.*, hundreds of meters is critical for mobile robot navigation in expansive outdoor environments. Structure recovery at such distances is not possible from fixed, on-board stereo systems because a much wider baseline is needed than can be achieved on a single robot. Generally speaking, very wide baseline between camera views can be achieved either by accumulating images over long distances, *i.e.*, structure from motion (SFM), or by using images taken from widely separated robots, *i.e.*, cooperative stereo. SFM is often favored because it simplifies the matching problem by using small incremental motion between images. In many cases, however, it is not possible to accumulate image data over a sufficiently long baseline to recover structure accurately. This is particular true for structure recovery in the direction of motion. Furthermore, in cluttered environments, it is difficult to track a large enough set of features to recover 3-D structure.

To address the potential shortcomings of SFM in robotics systems, we have investigated the feasibility of the alternative approach, cooperative stereo, to perform image matching over the very wide baselines necessary for long-range structure recovery. This approach is attractive because it makes no assumption on the robot's motion and, in principle, requires only images from a few positions of the robots.

Recent advances in wide-baseline matching in the Computer Vision literature provide a solid basis to tackle this problem. In particular, the landmark work of Zisserman and his colleagues [107] provides the basic tools for wide-baseline stereo. Our work is based in large part on that formulation; we extend it and evaluate it in the context of very wide baselines for robotic navigation.

For a mobile robot control standpoint, recent results in multi-robot cooperation [24] show that multi-robot planners can support the coordination of multiple robots to ensure coverage of a scene by their sensors.

Cooperative Stereo

In order to make a stereo measurement, a single robot would have to cooperate with another robot to take the second image [24] [117]. In a setup like this, there is only a limited amount of control over the camera position. It is therefore necessary for the stereo algorithm to deal with significant different views and occlusions, different scale and camera rotation. We want to perform cooperative stereo in three

a b c

FIGURE A.1. A few typical outdoor scenes. (a) man-made environments at moderate distances, *e.g.*, up to 100m. (b) environments with little regular structure. (c) a fair amount of structure but at a large distance.

broad classes of environments, as shown in Figure A.1. First, we want to address environments that contain a lot of structure, such as man-made environments, as shown in Figure A.1(a), at moderate distances, *e.g.*, up to 100m. This type of environment is the closest to the examples typically studied in the existing work on wide-baseline stereo. The second class of environments is shown in Figure A.1(b) in which little regular structure can be extracted from the images. In such cases,

we need to rely on ill-defined features such as regions uniform in color or texture rather than on geometric features such as straight lines and corners. A third case is shown in Figure A.1(c) in which the scene does contain a fair amount of structure but the distance to the camera and the baseline between the robots is considerably larger. The large aspect difference between the images induced by the very wide baseline complicates stereo matching substantially. The environments of Figure A.1(b) and Figure A.1(c) depart from the type of environments normally used in wide-baseline stereo work and are closer to target environments for colonies of mobile robots operating in outdoor environments.

1. Algorithm Description

In this section, we describe the algorithm for recovering epipolar geometry and scene structure, from widely separated viewpoints. After extracting features from the two images, as described in the next section, we apply progressively tighter filtering criteria to the set of candidate matches generated from the two sets of features from the two images.

The overall strategy for finding a set of matching regions can be summarized as follows. Each region i from image j, R_i^j, is represented by a feature vector f_i^j describing global characteristics of the regions, for example, color histogram. f_i^j is used for establishing initial matches between regions based on global properties. In addition, each R_i^j is warped to a new image ${R_i^j}^o$ in a normalized frame, a fixed square patch, using a homography $\boldsymbol{H}^{oj}_i$ computed from the region's parameter. If two image regions R_k^1 and R_l^2 correspond to the same, approximately planar, physical patch, then ${R^{oj}}_k$ and ${R^{oj}}_l$ should be similar. This fact is used to further refine the set of matches between regions. Finally, given a candidate set of matches, the homography H_n mapping the first region ${R^1}_{k_n}$ of correspondence n to the second region ${R^2}_{l_n}$ is estimated, using the homography induced by $\boldsymbol{H}^{o1}_{k_n}$ and $\boldsymbol{H}^{o2}_{l_n}$. The approach and the notations are summarized in Figure 1.

Below, we describe the three levels of filtering and refinement of the set of candidate matches, leading to the final evaluation of the epipolar geometry.

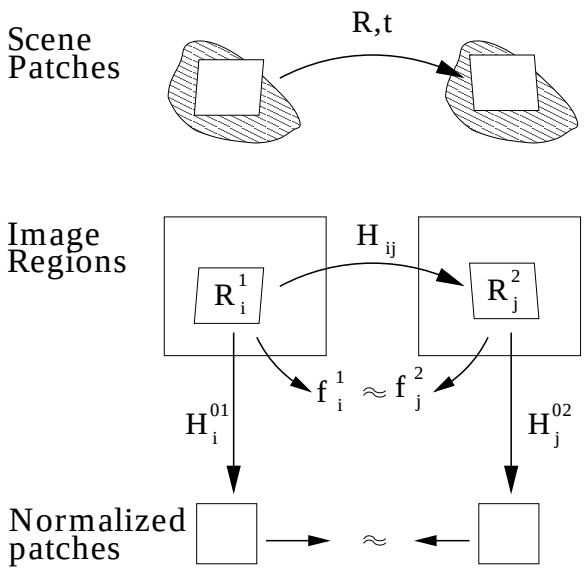


FIGURE A.2. General approach to finding matching regions.

Feature Extraction

The scenes we consider can be roughly divided in two classes: mostly geometric man-made and natural scenes. The approaches taken for these scenes differ only in the first phase of the algorithm. In both cases, we have ruled out approaches based solely on point features in favor of region-based techniques for the reasons stated above. Similarly, we rely on the approximate planarity of the image regions to reduce the number of matches needed for geometry recovery and to further constrain matching.

Feature Extraction: Line Groupings. It has been shown, in the context of 3-D modeling from multiple aerial images [4] for scenes with substantial geometric structure content, regions can be extracted by grouping line segments. Four line segments are normally needed for each region, although, in practice, three line segments are sufficient with the fourth one being inferred. To avoid searching through all possible 3-tuples of line segments, the endpoints of the line segments are first stored in bins equally spaced in the horizontal and vertical directions. Bins of 50-pixel width are used. For each segment, candidate segments with connecting

endpoints are retrieved by consulting the appropriate bins. Line segments are divided into two categories: near-horizontal and near-vertical lines. The two groups are defined based on large tolerance, $\pm\ 20^o$, between the lines and the horizontal and vertical directions. Groups are formed by finding connecting lines segments from the two categories. The use of the image axis as reference to aid in grouping restricts somewhat the generality of the approach. In practice, for images taken by mobile robots in which in-plane camera rotation is limited, the number of groups that are missed is small enough that it has no effect on the performance of the system. The result of this initial grouping approach is a set of cycles which is further filtered based on thresholds on the ratio of lengths of opposing line segments and on the elongation of the regions enclosed by the cycles.

Each region R_i^j corresponding to a cycle is mapped to a normalized reference patch by computing the transformation $H^{o}{}_i^j$ that maps its vertices to a 15x15 square. The intensity values (Y channel) in the region defined by the cycle are mapped to the normalized patch. Such a small size is used for the normalized patch so that comparison of regions for initial matching is fast. It is important to note that the normalized patches are used only for filtering out correspondences; the transformations between regions are estimated by using all the pixels within the regions.

Figure A.3 shows a typical set of line groupings and an example of normalized patch representation. Typical for these real images, a number of spurious cycles are detected and the normalized patches may not match even if the groups are geometrically correct. Filtering and refinement of the matches are described in the sections below.

Each region R_i^j is represented by a feature vector f_i^j which includes:

- Area and axes of the region. The axes are computed by taking the eigenvectors of the second moment matrix of the region.
- A color histogram computed in the U and V channels of the YUV colorspace. Each channel was converted into a 3bit color-depth. These two vectors were concatenated in a 64-bin histogram.

This feature vector is used for generating initial matches as described later in this section.

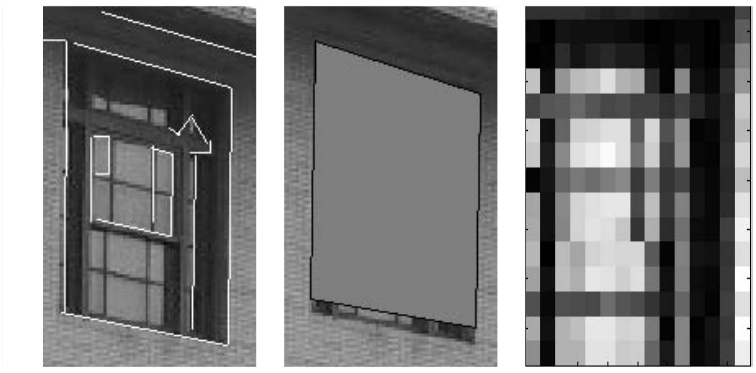


FIGURE A.3. Lines extracted from an image (a), a hypothesized line grouping (b) and the reference normalized patch (c)

Feature Extraction: Regions. A color segmenter [19] is used in natural scenes that lack strong geometric features. The same feature vectors as before, area and axes, and color histogram, are used for representing each region. Using the same feature vectors as in the case of regions computed from line groupings allows us to use exactly the same machinery for matching.

In addition, each region is warped to a normalized patch as in the case of the four-cycle representation above. More precisely, each segmented region is mapped to a reference square patch. The transformation is estimated by mapping the two axis and the two elongations of the second moment matrix to the reference axes.

Figure A.4 shows a typical example of a segment from an image segmentation. The ellipse computed from the second-order matrices is shown as is the normalized reference frame. As can be seen in this example, many regions are segmented incorrectly due to over-segmentation, occlusion, and photometric variations. This is typical of real outdoor images. It is the job of the matching algorithm described below to retain those few regions that can be matched reliably.

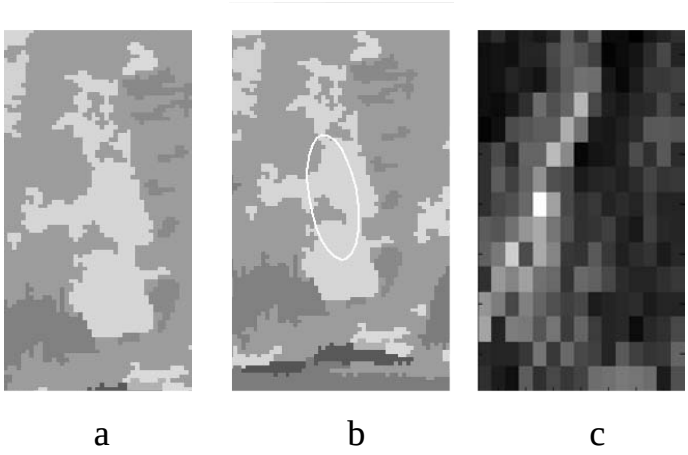


FIGURE A.4. a typical segment from an image segmentation (a), the principal axis (b) and the normalized reference patch (c).

Initial Filtering of Matches

Using the feature descriptor computed as described above, a first filtering step is applied to eliminate incorrect matches. Due to the large number of possible matches, this first step uses low-computation tests. Also, liberal thresholds are used to avoid filtering out acceptable matches at this stage.

We first reject all matches that have a significant mismatch in area, axis length and orientation. Matches are rejected if the area is off by a factor of 10, an axis mismatch by a factor of 3 or a 30° orientation difference. In order to further reduce the candidate matches, we only keep the 20 best color matches. These are the matches with the smallest sum of squared differences (L_2-distance) of the two color histograms.

This initial filtering reduces the number of potential matches typically to a dozen or so out of an initial set on order of one thousand candidates.

Initial Matching

For the remaining candidates the normalized gray-scale patch is used to compute a similarity measure. For this, we will use the result, as presented by Pritchett and Zisserman [95], that cross-correlation is geometrically invariant if the pixels of

two planar patches are mapped unto each other under a homography. So if the result of a cross-correlation between these warped patches yield a high correlation score, the underlying assumption that the patches are the same and planar holds.

To compute an initial assignment, we would like to find the best global consistent pairing between features in the left and right image. To do this, we extend the approach suggested in [92,108,109] in the context of matching point features to planar regions. In this approach, we define a cross-correlation similarity measure as a cost metric weighted by a proximity term [92].

The total cost for a match between feature i from one image and j from the other is given by:

$$G_{ij} = 1 - \frac{(C_{ij} + 1)}{2} e^{\frac{-r_{ij}^2}{2\sigma^2}} \tag{A.1}$$

In which C_{ij} is the normalized cross-correlation value between patch i and j and r_{ij} the Euclidean distance between the feature in the first and the second image. This distance metric follows a Gaussian decay over the support region σ. This distribution favors more distant matches, but because of the gradual decay and the large support region, matches within a shorter range are still accounted for. The costs in this cost matrix G ranges from 0 to 1, the smaller the cost, the better the match. In order to be able to assign a one to one mapping between the two feature sets, the cost matrix is padded to a square matrix with the rogue feature match cost set to a constant value (0.4).

The matching problem now follows the regime of a maximum weighted matching problem for bipartite graphs for which a standard solution is the *Hungarian method* [28], a polynomial-time algorithm. The output of this stage is a set of possible correspondences between regions, which has been filtered based on similarity between feature vectors, similarity between normalized patches, and global consistency.

Figure A.5 shows the set of candidate matches retained at this point in the matching algorithm for two regions.

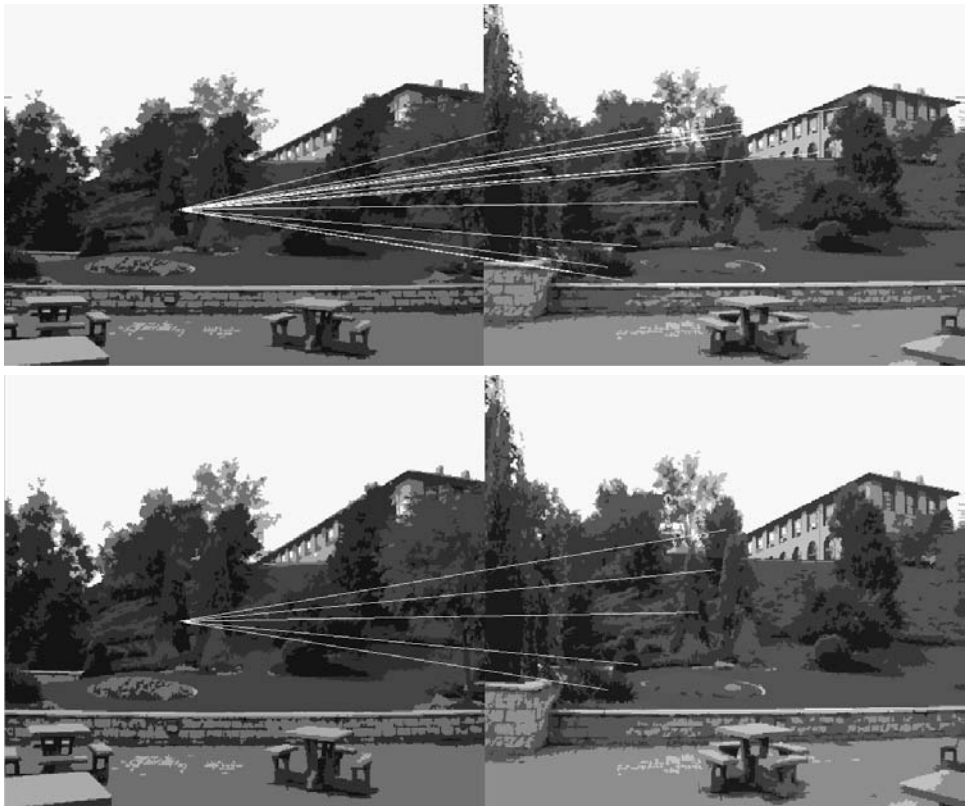


FIGURE A.5. Unfiltered matches (top), candidate matches after filtering based on feature vectors and normalized patch similarity (bottom).

Refinement

Since the number of potential correspondences is reduced to a small number N by using the steps above, it is now possible to use more expensive algorithms to compute the exact transformation between regions. In keeping with our general approach, we use all the pixels in the regions to estimate the transformation rather than estimating the transformation from sets of feature points extracted inside the regions. This may be potentially more expensive than using point matches, but it leads to more accurate estimates of the transformations. This is in contrast with other wide-baseline stereo work [107] in which the transformations are estimated from point features.

To simplify notations, we denote by (R_i^1, R_i^2), $i = 1, \ldots, N$, the set of matches identified at the previous steps. Since those candidate matches have fairly high

confidence, we can now compute explicitly the transformation $\boldsymbol{H}_i$ that maps R_i^1 to R_i^2 for every correspondence i. $\boldsymbol{H}_i$ is recovered by minimizing the objective function J defined as:

$$J(\boldsymbol{H}_i) = \sum_{\boldsymbol{x} \in R_i^1} (I_2(\boldsymbol{H}_i \boldsymbol{x}) - I_1(\boldsymbol{x}))^2, \tag{A.2}$$

where $\boldsymbol{H}_i$ the homography that maps region R_i^1 in the first image onto region R_i^2 in the second image. Levenberg-Marquardt is used to adjust the homography $\boldsymbol{H}_i$ such that the similarity between the patch in the second image and the warped patch from the first image is optimal. The LM iterations are initialized at the value of $\boldsymbol{H}_i$ induced by the homographies to the normalized patches, $\boldsymbol{H}^{o1}_i$ and $\boldsymbol{H}^{o2}_i$.

In practice, this type of optimization is known to be able to recover only small pixel discrepancies between patches (strictly speaking, at most one pixel motion can be recovered.) In practice, however, the initial homography computed from $\boldsymbol{H}^{o1}_i$ and $\boldsymbol{H}^{o2}_i$ may be relatively far from the optimum due, for example, to instability in region segmentation. To allow for a larger discrepancy between the regions, we can minimize the J first using lower resolution of the images and use the resulting $\boldsymbol{H}_i$ to start the estimation at the next higher level of resolution. Four levels of resolution seem sufficient to handle the typical errors between matching regions. The images are blurred between resolution levels by a Gaussian of $\sigma = 11$.

Epipolar Estimation

The homographies $\boldsymbol{H}_i$, $N = 1, \ldots, N$, estimated as described above are used for computing the initial epipolar geometry. As before, while the homographies can be used to find point correspondences [6], we prefer to first use the homographies directly to compute a first estimate without using additional point features. If $(\boldsymbol{R}, \boldsymbol{t})$ is the transformation between the two viewpoints, the epipolar geometry is characterized by $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$. This is assuming that the cameras are calibrated and

that the image coordinates are expressed as normalized coordinates. Classical results from multi-view geometry show that the epipolar geometry can be recovered exactly from two homographies [42].

Based on this classical result, we could pick pairs of homographies from the set $\boldsymbol{H}_i$, $i = 1, \ldots, N$ and compute the corresponding epipoles and epipolar geometry. A better approach is to use all the homographies simultaneously to compute $\boldsymbol{t}$ and $\boldsymbol{R}$. This is the approach we use, based on the formulation introduced in [27]. As a final step, the estimation is refined by using a robust estimator.

We use the standard representation of each homography as:

$$\boldsymbol{H}_i = \lambda_i \boldsymbol{R} + \boldsymbol{t}\boldsymbol{v}_i^T,$$

where $\boldsymbol{v}$ is the scaled normal to plane i: $\boldsymbol{v}_i = \lambda_i \frac{\boldsymbol{n}_i}{d_i}$ (the translation is normalized to unit length by convention.) We then find the unknowns, $\boldsymbol{R}, \boldsymbol{t}, \lambda_i, \boldsymbol{v}_i, i = 1, \ldots, N$ that minimizes the difference between $\boldsymbol{H}_i$ and $\lambda_i \boldsymbol{R} + \boldsymbol{t}\boldsymbol{v}_i^T$, summed over $i = 1, \ldots, N$. It turns out that the solution to this problem can be computed by first considering a virtual homography: $\boldsymbol{H} = \sum_i \boldsymbol{H}_i = \lambda \boldsymbol{R} + \boldsymbol{t}\boldsymbol{v}^T$, where $\boldsymbol{v}$ is the scaled normal to the virtual plane. If $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ is the SVD decomposition of $\boldsymbol{H}$, then the optimal $\boldsymbol{R}$ and $\boldsymbol{t}$ is obtained by equating λ to the largest singular value of $\boldsymbol{H}$ and $\boldsymbol{t} = \boldsymbol{U}\boldsymbol{t}', \boldsymbol{v} = \boldsymbol{V}\boldsymbol{v}'$. The components of the vectors $\boldsymbol{t}'$ and $\boldsymbol{v}'$ are computed as solutions of fourth-order polynomial equations whose parameters depend only of $\boldsymbol{S}$. This approach is similar to the one used in [27].

There are in fact four solutions for $\boldsymbol{R}, \boldsymbol{t}$ using this method. One pair of solutions corresponds to the usual ambiguity on the sign of $\boldsymbol{t}$ which is eliminated by enforcing a sign constraint on one of the coordinates of $\boldsymbol{t}$. The second pair of solutions corresponds to the fact that if $(\boldsymbol{R}, \boldsymbol{t})$ is a solution, then so is $(\boldsymbol{R}\boldsymbol{Q}, \boldsymbol{t})$ (with different values of $\boldsymbol{v}$,) where $\boldsymbol{Q}$ is a rotation of 180^o about $\boldsymbol{t}$. A simple test ensuring that both cameras are pointing in the same direction eliminates this ambiguity.

Robust Estimation

The previous step provides an estimate of $\boldsymbol{R}$ and $\boldsymbol{t}$ from the homographies. This estimation was performed so far without outlier rejection so that the current estimate may be corrupted by spurious matches. As a last step, to eliminate possible remaining spurious matches, we re-estimate $\boldsymbol{R}$ and $\boldsymbol{t}$ based on the vertices of the regions. We denote by $\boldsymbol{P}_{ij}^1$, $j = 1, \ldots, 4$ are the four vertices of region i in image 1 and $\boldsymbol{P}_{ij}^2 = \boldsymbol{H}_i \boldsymbol{P}_{ij}^1$ are the vertices in image 2 obtained by transformation by the current estimate of the homography $\boldsymbol{H}_i$. In the case of region extraction from lines, the vertices are the vertices of the polygon defining the region; in the case of free-form segmentation, the vertices are extremities of the principal axes of the second-moment matrix of the region. $\boldsymbol{R}$ and $\boldsymbol{t}$ are estimated by minimizing:

$$E(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1,j=1}^{i=N,j=4} \psi(d(\boldsymbol{P}_{ij}^2, \boldsymbol{E}\boldsymbol{P}_{ij}^1) + d(\boldsymbol{P}_{ij}^1, \boldsymbol{E}^T \boldsymbol{P}_{ij}^2)), \tag{A.3}$$

where $d()$ is the usual normalized distance between image point and epipolar line, and $\psi()$ is a Lorentzian function of the form $\psi(d) = \frac{d^2}{\sigma^2 + d^2}$. The scale σ is proportional to the median of the error $d()$ over all pairs of points. This approach is the one used, for example, in robust registration [123]. An alternative robust estimation technique would be to use a technique such as RANSAC. In practice, the outliers are effectively eliminated by using a Lorentzian directly, which also has the advantage to produce an optimal estimate of $(\boldsymbol{R}, \boldsymbol{t})$ at the same time.

2. Experimental Results

For the typical images as show in Figure A.1, we have included the epipolar reconstruction results.

In Figure A.6 the result for structured scene is shown. The result for this type of scene is very good as can be seen by the precise epipolar reconstruction. This is further illustrated in Figure A.9 by the fact that the ratio between the edges of two reconstructed windows matches fairly closely: 1.5456 for the left and 1.8356 for the right. The algorithm initially considered all the 262x305 matches, after filtering

only 369 of these matches were considered for refinement and 41 matches passed the global matching and robust optimization phase.

The result for the more natural scene (Figure A.7) was computed using only the patches from the segmentation stage. These 533x919 original patches were filtered down to 137 candidate matches, from which 17 made it through the global matching and robust optimization phase. Because the segmentations are much influenced by noise, this result is far less accurate as the previous result.

A much more exciting result is shown in Figure A.8 here we show a result that was computed from a 33m baseline. Locations in the scene cover a distance from a 100m to a 1000m. A small error has therefore quite some influence in the result. Nevertheless the reconstruction is fairly accurate. Here we started out with considering 982x669 matches, after filtering we had 1107 matches left from which 194 survived the global matching and robust optimization.

FIGURE A.6. The reconstructed epipolar geometry from the man-made structured environment.

FIGURE A.7. The reconstructed epipolar geometry from the more natural environment.

FIGURE A.8. The reconstructed epipolar geometry from the very wide baseline

FIGURE A.9. The ratio of the height over the width of the window on the left is 1.5456 and the window on the right 1.8356, which gives us an indication of the reconstruction quality.

REFERENCES

[1] M. Agrawal and K. Konolige. Wide-baseline image matching for motion estimation. Submitted to IEEE International Conference on Image Processing, 2004.

[2] R. Alami and T. Simeon. Planning robust motion strategies for a mobile robot. In IEEE, editor, *International Conference on Robotics and Automation*, volume 2, pages 1312 – 1318, 1994.

[3] T. Bailey, E. Nebot, J. Rosenblatt, and H. Durrant-Whyte. Data association for mobile robot navigation: a graph theoretic approach. In IEEE, editor, *International Conference Robotics and Automation*, volume 3, pages 2512–2517, 2000.

[4] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[5] J. Banta, Y. Zhien, X. Wang, G. Zhang, M. Smith, and M. Abidi. A best-nextview algorithm for three-dimensional scene reconstruction using range images. In *SPIE*, volume 2588, pages 418–429, 1995.

[6] A. Bartoli, P. Sturm, and R. Horaud. Projective structure and motion from two views of a piecewise planar scene. In *Proceedings IEEE International Conference on Computer Vision*, 2001.

[7] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for demo iii. In *Intelligent Vehicles Conference*, 2000.

[8] J. Besag. On the statistical analysis of dirty pictures. *Journal of Royal Statistical Soc.*, B-48:259–302, 1986.

[9] S. Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Algorithms*, 49(2):284–303, 2003.

[10] B. Bonet. An e-optimal grid-based algorithm for par-tially obserable markov decision processes. In *ICML*, 2002.

[11] R. I. Brafman. A heuristic variable grid solution method for pomdps. In *AAAI*, 1997.

[12] O. Brock and O. Khatib. Real-time re-planning in high-dimensional configuration spacesusing sets of homotopic paths. In IEEE, editor, *International Conference on Robotics and Automation*, volume 1, pages 550–555, 2000.

[13] R. Brooks and A. Flynn. A robust layered control system for a mobile robot. In *IEEE Transactions on Robotics and Automation*, volume 2(1), 1986.

[14] S. Cabello, Y. Liu, and A. Mantler. Testing homotopy for paths in the plane. *Discrete and Computational Geometry*, 31(1):61–81, 2004.

[15] A. Cassandra, L. Kaelbling, and M. Littman. Acting optimally in partially observable stochastic domains. In *Twelfth National Conference on Artificial Intelligence*, pages 1023–1028, Seattle, Washington, 1994. AAAI Press/MIT Press.

[16] Z. Chen and C. Huang. Terrain exploration of a sensor-based robot moving among unknown obstacles of polygonal shape. In *Robotica*, volume 12, pages 33–44, 1994.

[17] H. Choset. *Sensor-Based Motion Planning: The Hierarchical Generalized Voronoi Graph*. PhD thesis, California Institute of Technology, 1996.

[18] H. Choset and J. Burdick. Sensor based planning, part i: The generalized voronoi graph. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation (ICRA '95)*, volume 2, pages 1649 – 1655, May 1995.

[19] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico*, 1997.

[20] C. Connolly. The determination of the next best view. In *IEEE International Conference on Robotics and Automation*, pages 432–435, 1985.

[21] M. Daily, J. Harris, D. Keirsey, K. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong. Autonomous cross-country navigation with the ALV. In *Proceedings of the International Conference on Robotics and Automation*, pages 718–726, 1988.

[22] M. Daily, J. Harris, and K. Reiser. An operational perception system for cross-country navigation. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 794–802, June 1988.

[23] M. Devy, R. Chatila, P. Fillatreau, S. Lacroix, and F. Nashashibi. On autonomous navigation in a natural environment. *Robotics and Autonomous Systems*, 16(1):5–16, 1995.

[24] M. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *Proceedings of the 6 th International Conference on Intelligent Autonomous Systems, Venice, Italy*, July 2000.

[25] A. Efrat and S. G. Kobourov. *Computing Homotopic Shortest Paths Efficiently*, volume 2461. Springer, 2002.

[26] J. M. Esposito and V. Kumar. Closed loop motion plans for mobile robots. In IEEE, editor, *International Conference on Robotics and Automation*, pages 2777–2782, April 2000.

[27] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *Journal of Pattern Recognition and AI*, 2(3):485–508, 1988.

[28] G. Fielding and M. Kam. Applying the hungarian method to stereo matching. In *Proceedings 1997 IEEE Conference on Decision and Control*, December 1997.

[29] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer graphics (2nd ed. in C): principles and practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[30] C. Fox and G. Nicholls. Exact map states and expectations from perfect sampling: Greig, porteous and seheult revisited. In *AIP Conference Proceedings on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, volume 568, pages 252–263, 2001.

[31] J. Gancet and S. Lacroix. Pg2p: A perception-guided path planning approach for long range autonomous navigation in unknown natural environments. In IEEE, editor, *International Conference on Intelligent Robotics and Systems*, 2003.

[32] H. González-Baños and J. Latombe. Planning robot motions for range-image acquisition and automatic 3d model construction. In *Integrated Planning for Autonomous Agent Architectures*, AAAI Fall Symposium Series, 1998.

[33] H. H. González-Baños and J. C. Latombe. Robot navigation for automatic model construction using safe regions. In *International Symposium on Experimental Robotics*, pages 405–415, 2000.

[34] H. H. González-Baños and J. C. Latombe. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research*, 12(10-11):829–848, 2002.

[35] Y. Goto and A. Stentz. The CMU system for mobile robot navigation. In IEEE, editor, *International Conference on Robotics and Automation*, volume 4, pages 99–105, 1987.

[36] R. Grabowski, P. Khosla, and H. Choset. Autonomous exploration via regions of interest. In IEEE/RSJ, editor, *International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[37] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Royal Statistical Soc.*, page 271:279, 1989.

[38] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive navigation in outdoor environments using potential fields. In IEEE, editor, *International Conference on Robotics and Automation*, volume 2, pages 1232–1237, May 1998.

[39] A. Hait and T. Simeon. Motion planning on rough terrain for an articulated vehicle inpresence of uncertainties. In IEEE/RSJ, editor, *International Conference on Intelligent Robots and Systems*, volume 3, pages 1126–1133, Nov 1996.

[40] A. Hait, T. Simeon, and M. Taix. Robust motion planning for rough terrain navigation. In IEEE/RSJ, editor, *International Conference on Intelligent Robots and Systems*, volume 1, pages 11–16, 1999.

[41] S. Y. Harmon. The ground surveillance robot (GSR): An autonomous vehicle designed to transit unknown terrain. *Robotics and Automation*, RA-3(3):266–279, June 1987.

[42] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000., 2000.

[43] M. Hauskrecht. Value-function approximationsfor partially observable markov decision processes. In *Journal of Artificial Intelligence Research*, pages 33–94, 2000.

[44] M. Hebert, C. Thorpe, and A. T. Stentz. *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon*. KluwerAcademic Publishers, 1997.

[45] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.*, 4(2):63–97, 1994.

[46] T.-H. Hong, T. Chang, C. Rasmussen, and M. Shneier. Feature detection and tracking for mobile robots using a combination of ladar and color images. In IEEE, editor, *International Conference of Robotics and Automation*, pages 4340–4345, Washington, D.C., May 2002.

[47] T.-H. Hong, C. Rasmussen, T. Chang, and M. Shneier. Fusing ladar and color image information for mobile robot feature detection and tracking. In *7th International Conference on Intelligent Autonomous Systems*, 2002.

[48] M. Huber. Efficient exact sampling from the ising model using swendsen-wang. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 921–922. Society for Industrial and Applied Mathematics, 1999.

[49] M. Huber. A bounding chain for swendsen-wang. *Random Struct. Algorithms*, 22(1):43–59, 2003.

[50] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[51] K. Kakusho, T. Kitahashi, K. Kondo, and J. Latombe. Continuous purposive sensing and motion for 2d map building. In *IEEE International Conference on Systems, Man & Cybernetics*, volume 2, pages 1472–1477, 1985.

[52] I. Kamon, E. Rivlin, and E. Rimon. A new range-sensor based globally convergent navigation algorithm for mobile robots. In IEEE, editor, *International Conference on Robotics and Automation*, pages 429–435, 1996.

[53] A. Kelly and A. T. Stentz. Rough terrain autonomous mobility - part 2: An active vision, predictive control. *Autonomous Robots*, 5:163 – 198, May 1998.

[54] A. J. Kelly. Ranger–an intelligent predictive controller for unmanned ground vehicles. Technical report, The Robotics Institute, Carnegie Mellon University, 1994.

[55] J. Kleinberg. On-line search in a simple polygon. In *5th ACM-SIAM Symposium on Discrete Algorithms*, pages 8–15, 1994.

[56] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In IEEE, editor, *International Conference on Robotics and Automation*, volume 1, pages 968– 975, 2002.

[57] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In IEEE, editor, *International Conference on Robotics and Automation*, volume 2, pages 1398–1404, 1991.

[58] E. Kruse, R. Gutsche, and F. Wahl. Efficient, iterative, sensor based 3d map building using rating functions in configuration space. In IEEE, editor, *International Conference on Robotics and Automation*, volume 2, pages 1067–1072, 1996.

[59] S. Kumar. Models for learning spatial interactions in natural images. *The Robotics Institute, School of Computer Science, Carnegie Mellon University*, 2004.

[60] K. Kutulakos, C. Dyer, and V. Lumelsky. Provable strategies for vision-guided exploration in three dimensions. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1365–1372, 1994.

[61] K. N. Kutulakos and C. R. Dyer. Recovering shape by purposive viewpoint adjustment. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition, CVPR*, Los Alamitos, California, 1992. IEEE Computer Society.

[62] A. Lacaze, Moscovitz, Y. DeClaris, and K. N. Murphy. Path planning for autonomous vehicles driving over rough terrain. In IEEE, editor, *International Symposium on Computational Intelligence in Robotics and Automation*, pages 50–55, 1998.

[63] S. Lacroix and R. Chatila. Motion and perception strategies for outdoor mobile robot navigation in unknown environments. In *The 4th International*

Symposium on Experimental Robotics IV, pages 538–547, London, UK, 1997. Springer-Verlag.

[64] S. Lacroix, R. Chatila, S. Fleury, M. Herrb, and T. Simeon. Autonomous navigation in outdoor environment: adaptive approach and experiment. In IEEE, editor, *International Conference on Robotics and Automation*, volume 1, pages 426 – 432, 1994.

[65] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains. demonstrations in the space museum "cité de l'espace" at toulouse. In *7th International Symposium on Experimental Robotics*, 2000.

[66] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous Rover Navigation on Unknown Terrains: Functions and Integration. *The International Journal of Robotics Research*, 21(10-11):917–942, 2002.

[67] D. Langer, J. Rosenblatt, and M. Hebert. A reactive system for off-road navigation. In IEEE, editor, *International Conference on Robotics and Automation*, May 1994.

[68] S. Laubach and J. Burdick. An autonomous sensor-based path-planner for planetary microrovers. In IEEE, editor, *International Conference of Robotics and Autonomation*, volume 1, pages 347 – 354, 1999.

[69] S. L. Laubach. *Theory and Experiments in Autonomous Sensor-Based Motion Planning with Applications for Flight Planetary Microrovers*. PhD thesis, California Institute of Technology, California, CA, May 1999.

[70] S. LaValle. *A Game-Theoretic Framework for Robot Motion Planning*. PhD thesis, University of Illinois, Urbana, IL, 1995.

[71] S. LaValle and R. Sharma. On motion planning in changing, partially-predictable environments. In *International Journal of Robotics Research*, volume 16, pages 775–805, 1997.

[72] S. Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, Tokyo, 2001.

[73] W. S. Lovejoy. Computationally feasible bounds for partially observed markov decision processes. In *Operations Research*, volume 39, pages 162–175, 1991.

[74] V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape. In *Algorithmica*, volume 2, pages 403–430, 1987.

[75] D. Maksarov and H. Durrant-Whyte. Mobile vehicle navigation in unknown environments: a multiple hypothesis approach. In IEEE, editor, *Control Theory and Applications*, volume 142, pages 385–400, 1995.

[76] J. Malik, S. Belongie, J. Shi, and T. Leung. contours and regions: Cue combination in image segmentation. In *Proceedings International Conference on Computer Vision, Kerkyra, Greece*, September 1999.

[77] R. Mandelbaum, M. Hansen, P. Burt, and S. Baten. Vision for autonomous mobility: Image processing on the vfe-200. In *Intl. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1998.

[78] M. Maurette. Mars rover autonomous navigation. *Autonomous Robots*, 14(2-3):199–208, 2003.

[79] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. In IEEE, editor, *Pattern Analysis Machine Intelligence*, volume 15(5), 1993.

[80] B. M. McCoy and T. T. Wu. *The Two-Dimensional Ising Model*. Harvard University, Cambridge, Mass., 1973.

[81] T. Minka. The 'summation Hack' as an Outlier Model. http://www.research.microsoft.com/~minka/papers/minka-summation.pdf, August 2003.

[82] S. Moorehead. *Autonomous Surface Exploration for Mobile Robots*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2001.

[83] B. Nabbe and M. Hebert. Toward practical cooperative stereo for robotic colonies. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3328–3335. Omnipress, May 2002.

[84] B. Nabbe and M. Hebert. Where and when to look. In *IROS2003*. IEEE, October 2003.

[85] F. Nashashibi, P. Fillatreau, B. Dacre-Wright, and T. Simeon. 3-d autonomous navigation in a natural environment. In IEEE, editor, *International Conference on Robotics and Automation*, volume 1, pages 433–439, 1994.

[86] E. Nebot, T. Bailey, and J. Guivant. Navigation algorithms for autonomous machines in off-road applications. In *Autonomous Robots*, submitted 2000.

[87] N. J. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, 1969.

[88] N. J. Nilsson. *Problem Solving Methods of Artificial Intelligence*. McGraw-Hill, New York, 1971.

[89] C. Ó'Dúnlaing and C. K. Yap. A retraction method for planning the motion of a disc. In *Journal of Algorithms*, volume 6, 1982.

[90] E. B. Pacisx, H. Everett, N. Farrington, G. Kogut, B. Sights, T. Kramer, M. Thompson, D. Bruemmer, and D. Few. Transitioning unmanned ground vehicle research technologies. In SPIE, editor, *Unmanned Ground Vehicle Technology VII*, volume 5804, Orlando, FL, 2005.

[91] D. Pai and L.-M. Reissell. Multiresolution rough terrain motion planning. *Transactions on Robotics and Automation*, 14(1):19–33, Feb 1998.

[92] M. Pilu. A direct method for stereo correspondence based on singular value decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[93] J. Pineau. *Tractable Planning Under Uncertainty: Exploiting Structure*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2004.

[94] R. Pito. A sensor based solution to the next best view problem. In IEEE, editor, *13th International Conference on Pattern Recognition*, volume 1, pages 941–945, 1996.

[95] P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *Proceedings 6th International Conference on Computer Vision, Bombay*, 1998.

[96] N. Rao, S. Kareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. In *Oak Ridge National Laboratory Document ORNL/TM-12410*, July 1993.

[97] C. Rasmussen. Combining laser range, color and texture cues for autonomous road following. In IEEE, editor, *International Conference of Robotics and Automation*, pages 4320–4325, Washington, D.C., May 2002.

[98] E. Rimon and J. Canny. Construction of c-space road maps from local sensory data: What should the sensors look for? In IEEE, editor, *International Conference on Robotics and Automation*, volume 1, pages 117 – 123, 1994.

[99] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. In IEEE, editor, *Robotics and Automation*, volume 8(5), pages 501–518, 1992.

[100] J. Rosenblatt. *DAMN: A Distributed Architecture for Mobile Navigation*. PhD thesis, Carnegie Mellon University Robotics Institute, Pittsburgh, PA, 1997.

[101] J. Rosenblatt. Optimal selection of uncertain actions by maximizing expected utility. *Autonomous Robots*, 9(1):17–25, 2000.

[102] M. Rosenblum. Immune systems are not just for making you feel better: They are for controlling autonomous robots. In SPIE, editor, *Unmanned Ground Vehicle Technology VII*, volume 5804, 2005.

[103] M. Rosenblum and B. Gothard. A high fidelity multi-sensor scene understanding system for autonomous navigation. In IEEE, editor, *Intelligent Vehicles Symposium*, pages 637–643, October 2000.

[104] M. Rosenblum and V. Rajagopalan. The road "more" traveled: A foundation for autonomous roadway operations. In SPIE, editor, *Unmanned Ground Vehicle Technology VII*, volume 5804, 2005.

[105] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation – mobile robot navigation with uncertainty in dynamic environments. In *Proc. IEEE Conf. Robotics and Automation (ICRA)*, May 1999.

[106] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[107] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Proceedings 8th International Conference on Computer Vision, Vancouver, Canada*, July 2001.

[108] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. *Royal Society of London*, B-244:21–26, 1991.

[109] L. Shapiro and J. Brady. Feature-based correspondence: An eigenvector approach. *Image and Vision Computing*, 10(5):283–288, June 1992.

[110] M. Sharir. Algorithmic motion planning in robotics. *Computer*, 22(3):9–20, 1989.

[111] R. Sharma, D. Mount, and Y. Aloimonos. Probabilistic analysis of some navigation strategies in a dynamic environment. *T-SMC*, 23:1465–1474, 1993.

[112] C. M. Shoemaker and J. Borenstein. The Demo III UGV program: A testbed for autonomous navigation research. In *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, pages 644–651, Gaithersburg, MD, 1998.

[113] D. Spero and R. Jarvis. Path planning for a mobile robot in a rough terrain environment. In IEEE, editor, *Third International Workshop on Robot Motion and Control*, pages 417– 422, 2002.

[114] A. Stentz. *The NAVLAB System for Mobile Robot Navigation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, March 1990.

[115] A. Stentz. Optimal and efficient path planning for partially-known environments. In IEEE, editor, *International Conference on Robotics and Automation*, pages 3310–3317, 1994.

[116] A. Stentz. The focussed D* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.

[117] S. Thayer, B. Digney, M. Dias, A. Stentz, B. Nabbe, and M. Hebert. Distributed robotic mapping of extreme environments. In *Proceedings of SPIE: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, November 2000.

[118] T. Tuytelaars and L. V. Gool. Wide baseline stereo matching based on local affinely invariant regions. In *Proceedings British Machine Vision Conference*, 2000.

[119] N. Vandapel, O. Amidi, and J. R. Miller. Toward laser pulse waveform analysis for scene interpretation. In *IEEE International Conference on Robotics and Automation*, May 2004.

[120] N. Vandapel, R. R. Donamukkala, and M. Hebert. Experimental results in using aerial ladar data for mobile robot navigation. In *International Conference on Field and Service Robotics*, 2003.

[121] R. Volpe, J. Balaram, T. Ohm, and R. Ivlev. The rocky7 mars rover prototype. In IEEE/RSJ, editor, *International Conference on Intelligent Robotics and Systems*, volume 3, 1996.

[122] P. Whaite and F. Ferrie. From uncertainty to visual exploration. In *3rd International Conference on Computer Vision*, pages 690–697, 1990.

[123] M. Wheeler and K. Ikeuchi. Sensor modeling, probabilistic hypotheses generation, and robust localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3), 1995.

[124] L. Wixson. Viewpoint selection for visual search. In IEEE, editor, *Conference on Computer Vision and Pattern Recognition*, pages 800–805, 1994.

[125] C. S. Won and H. Derin. Unsupervised segmentation of noisy and textured images using markov random fields. *CVGIP*, 54:308–328, 1992.

[126] C. Ye and J. Borenstein. A method for mobile robot navigation on rough terrain. In IEEE, editor, *International Conference on Robotics and Automation*, volume 4, pages 3863– 3869, 2004.

[127] A. Zelinsky, R. Jarvis, J. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *International Conference on Advanced Robotics*, pages 533–538, 1993.

[128] W. Zhao, D. Nister, and S. Hsu. Alignment of continuous video onto 3d point clouds. *Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1305– 1318, 2005.

[129] R. Zhou and E. A. Hansen. An improved grid-based approximation algorithm for pomdps. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 707–716, 2001.