PATH PLANNING IN IMAGE SPACE FOR THE AUTONOMOUS NAVIGATION OF UNMANNED VEHICLES IN UNSTRUCTURED OUTDOOR ENVIRONMENTS

By

MICHAEL WILSON OTTE

B.S., Clarkson University, 2005

A thesis submitted to the Faculty of the

College of Engineering and Applied Science of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Master of Science

Department of Computer Science

2007

This thesis entitled:

Path Planning in Image Space for the Autonomous Navigation of Unmanned Vehicles in Unstructured Outdoor Environments
written by Michael Wilson Otte
has been approved for the Department of Computer Science

Gregory Grudic

Jane Mulligan

Mike Mozer

Date______________

The final copy of this thesis has been examined by the signatories, and we Find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Otte, Michael Wilson (M.S., Computer Science)

Path Planning in Image Space for the Autonomous Navigation of Unmanned Vehicles in Unstructured Outdoor Environments

Thesis directed by Assistant Professor Greg Grudic

An approach to stereo based local path planning in unstructured environments is presented. The approach differs from previous stereo based and image based planning systems (i.e. top-down occupancy grid planners, autonomous highway driving algorithms, and view-sequenced route representation), in that it uses specialized cost functions to find paths through an occupancy grid representation of the world directly in the image plane, and forgoes the standard projection of cost information from the image plane down onto a top-down 2D Cartesian cost map. Three cost metrics for path selection in image space are discussed. A basic image based planning system is presented, and its susceptibility to rotational and translational oscillation is discussed. Two extensions to the basic system are presented that overcome these limitations—a cylindrical based image system and a hierarchical planning system. All three systems are implemented in an autonomous robot and are tested against a standard top-down 2D Cartesian planning system on three outdoor courses of varying difficulty. It was found that the basic image based planning system fails under certain conditions; however, the cylindrical based system is well suited to the task of local path planning and for use as a high resolution local planning component of a hierarchical planning system.

This Thesis is Dedicated to My Family.

ACKNOWLEDGEMENTS

I would like to thank Daniel Lee at the University of Pennsylvania for providing me with computer code for a top down Cartesian planning system. I would also like to thank Greg Grudic, Jane Mulligan, Scott Richardson, Adam Bates, and all of the other members of the Intelligence in Action Laboratory at the University of Colorado, Boulder, for their help with programming the sensory components of our robotic system, field testing, and advice on this research. Finally, I would like to thank the Defense Advanced Research Projects Agency, and the National Science Foundation for supporting this work.

CONTENTS

FIGURES

FIGURE PAGE

CHAPTER I, INTRODUCTION

Autonomous robot navigation aims to identify a series of movements that, when executed in a sequence, will translate the robot from a starting position to a goal position. The search for this path is constrained by the robot's sensor information and kinematics. Ideally, the path is chosen to minimize (or maximize) some criteria, such as energy expenditure. In highly structured environments, such as those encountered by a manipulator arm on a factory floor, an objective function can be found that describes the manifold on which the arm is constrained in actuator space. In this case, however, uncertainty about the world is limited. On the other hand, in unstructured environments—particularly outdoor environments beyond the city streets and paths of human infrastructure—there is not such high confidence *a priori* knowledge about the relationship between the appearance of a scene and its traversability.

Visual perception involves decoding 2D projections of 3D Cartesian space as they are captured by a robot's imaging sensors [1], [2]. These 2D projections are said to exist in *image space*. Many approaches to path planning in unstructured environments derive an obstacle vs. safe representation of a scene—referred to as an occupancy grid—which is created by projecting information from image space down onto the ground plane and then inserting it into an X-Y Cartesian map [3], [4]. Path planning systems have also used 3D occupancy grids to represent the world [5]. The A* algorithm [6] (or some variant [7]–[9]) is then used to find a path through the occupancy grid between the robot's position and the goal [3]. Work has also been done to model the path planning problem with various types of potential fields, as in [10] and [11], and as a hybrid of A* and potential fields, as in [12].

There are a number of advantages to planning a mobile robot's movement in a Cartesian map. However, this representation is not ideal for near-field planning—in order to maintain a map with a computationally feasible search space, the world must be resampled at a non-native resolution. Although there are some planners that maintain a higher resolution map for local path planning, e.g. [13], We propose that the transformation onto the Cartesian plane is superfluous.

To the best of our knowledge, planning and actuation in image space has not been studied on a robotic platform for use in unstructured environments. There are, however, examples of image based visual servoing in semi-structured and structured environments.

Autonomous highway driving algorithms [14]–[18] operate in a semi-structured environment. Information from image features such as lane markings, other automobiles, road color/texture, etc, allow these algorithms to follow the road while avoiding obstacles.

A robotic arm on a factory floor can be controlled via a constraint optimization function that maps the current field of view (FOV) to a reference or target frame through a series of movements [19], [20]. This idea has been extended to mobile robots in semi-structured environments in various forms

[21]–[23]. For instance, View-Sequenced Route Representation (VSRR) is a mapless navigation technique that calculates the displacement between a target image and the current FOV [24], [25]. This displacement is then translated into steering commands.

Autonomous highway driving algorithms and VSRR type models develop a control strategy as a function of the perceived scene. However, both make assumptions about the information that is available to them from the scene; for instance, the existence of lane markings or a clear view of a predefined goal state, respectively. These may be reasonable constraints in structured or semi-structured environments; however, planning through ambiguous terrain renders them infeasible.

The task that we are concerned with involves not only identifying traversable terrain from non-traversable terrain, but also finding and staying on a path to the goal. Specifically, the best path to the goal given one or more predefined optimality constraints and the current information about the world. We present an approach to path planning that allows local path search to take place directly in the image plane, preserving the flexibility of the occupancy grid paradigm while avoiding the corresponding transformation distortion induced by the projection into a Cartesian coordinate system. In this scheme, a real-world GPS coordinate is projected into image space as the goal. Next, a variant of A* is used directly in image space to identify an optimal path to the goal. Finally, robot servoing in the real world is accomplished via the image space path that is found by A*. Special attention must be placed on the runtime complexity of the system to allow the robot a suitable reaction time.

The basic image based planning system is called the *Image Planner*, and is introduced in Chapter III. The Image Planner lacks memory of the world and, therefore, planning can quickly degenerate into an infinite loop of the form: move away from the goal to avoid an obstacle, and then move back toward the goal (and thus the obstacle), after the obstacle's existence has been forgotten. These limitations are addressed with a series of extensions to the Image Planner. The *Cylindrical Planner*, introduced in Section IV-*A*, is created by augmenting the rotational memory of the Image Planner to include the world beyond its FOV. A hybrid Hierarchical Planner, introduced in Section IV-*B*, combines the strengths of a local Image Planner with those of a global Cartesian planner. Experiments are presented in Chapter V and a discussion of the results is presented in Chapter VI.

CHAPTER II, EXPERIMENTAL APPARATUS

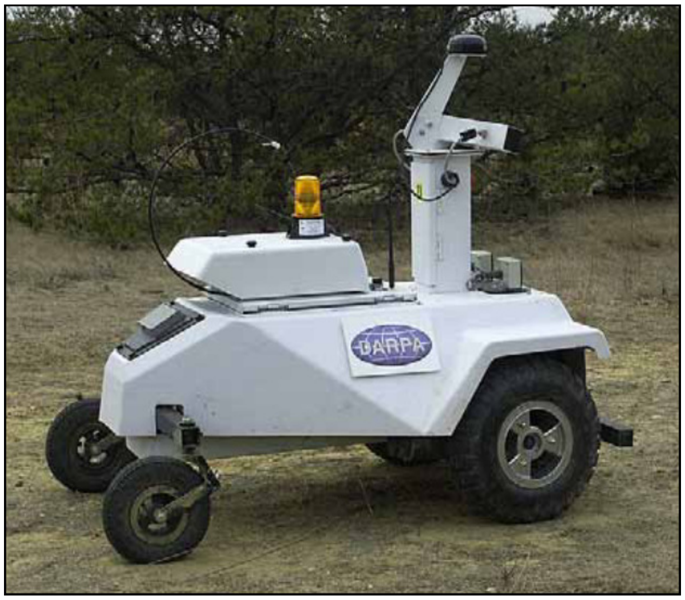

Figure 1. The DARPA LAGR Robotic Platform.

The mobile robot platform used for this work is provided in conjunction with the DARPA Learning Applied to Ground Robotics (LAGR) program. It measures roughly 1.2m x .8m x 1.2m. Its sensors include: two forward facing Point Grey BumbleBee 2 stereo camera pairs, a Garmin GPS receiver, a magnetic compass, and wheel odometers. There are also two forward facing infrared sensors and a front bumper sensor. The computational units include: one computer dedicated to each of the two stereo camera pairs, a third for the planning system, and a fourth that acts as a servo controller.

The stereo camera pairs are used to compute stereo disparity information, as well as color and texture data. The disparity information is accurate to approximately 15 meters, while color and texture data do not suffer from this limitation. The infrared sensors have a range of approximately

1 meter and are used for passive obstacle detection and avoidance—that is, the robot is decelerated if the infrared sensors detect an obstacle, however, information from the infrared sensors is not placed into the cost map.

Translation and rotation are achieved via two independently driven front wheels. The wheels are located on either side of the vertical axis that passes through the midpoint of the sensor mast, thus rotation around the mast axis is achieved by driving the wheels in opposite directions at the same speed.

CHAPTER III, THE IMAGE PLANNER

Section A, Occupancy Grids and Cost Functions

Let *R* denote the 3D Cartesian real-world space. A primary goal of this work is to achieve navigation through *R* toward a goal via paths found in image space. The robot perceives *R* as a stereo disparity image *S*, provided by a pair of stereo CCD cameras. The idea is to build an occupancy grid *O* in image space based on *S*, and then find the path $P^{optimal}$ that minimizes a quantity *W* that is analogous to mechanical work (i.e. force multiplied by distance). See Figure 2 for an example of such a path. Because any path found in *O* is a projection of some path existing in *R*, it is possible to navigate through *R* using $P^{optimal}$. This can be done directly, or via a projection of $P^{optimal}$ from image space into *R*.

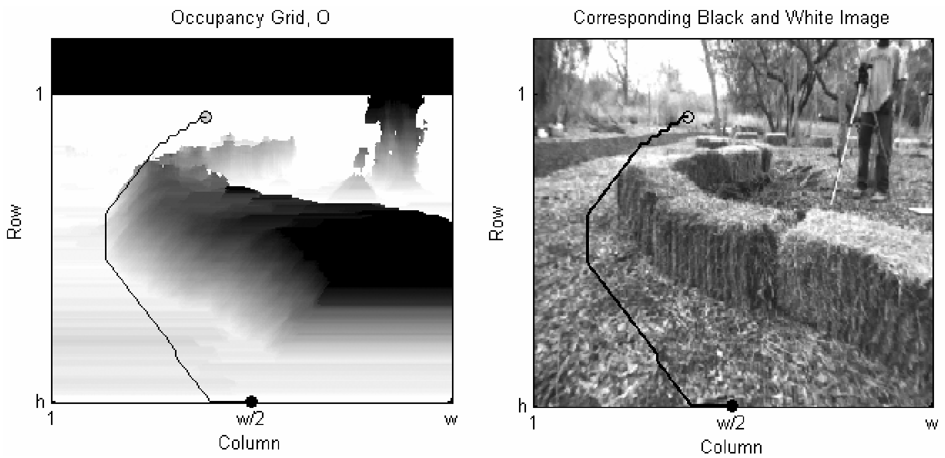

Figure 2. A path through *O* from the robot's position to a goal in the far-field—light to dark corresponds to low to high cost (left). The path projected into a black and white image of the scene (right).

S is organized in an h by w Cartesian grid based on the camera's physical pixel layout. The traversability of *R* is defined with the occupancy grid *O*:

$$O_{n,m} = f\left(S_{n,m}\right) = \left|S_{n,m} - S_{n,m}^{flat}\right|, \tag{1}$$

where n = 1…h and m = 1…w. Note that n = 1 and m = 1 correspond to the top row and left most column of *O*, respectively. $S_{n,m}$ is the disparity of pixel (n, m) in the scene at time t and $S_{n,m}^{flat}$ is the nominal disparity of a flat ground plane R^{flat}. The goal R_{goal} is defined by a GPS coordinate in *R*. R_{goal} is mapped into *O* as O_{goal}, assuming that both R_{goal} and the robot exist on R^{flat}. The robot's starting location in *O* is defined $O_{start} \equiv O_{h,w/2}$. The traversability values stored in *O* are interpreted as forces *F* that impede robot progress, and the planning system searches for paths through *O* that minimize the amount of work W_p that must be exerted to reach O_{goal} from O_{start}.

$$W_P = \int_{O_{start}}^{O_{goal}} F(P)dP \tag{2}$$

where dP is the differential of position along *P*. O_{goal} and O_{start} are nodes in *O* that anchor the endpoints of *P*. *P* contains $\|P\|$ connected subsections i in *O*, each starting at the center of a grid location $O_{j,k}$ and terminating at $O_{n,m}$, one of the 8-connected neighbors of $O_{j,k}$. Therefore, the work required to traverse *P* is found by the summation of work over its subsections.

$$W_P = \sum_{\forall i \in P} W_i = \sum_{\forall i \in P} F_i D_i \,, \tag{3}$$

where W_i is the work required to navigate path subsection i, F_i is the force that impedes robot progress along i, and D_i is the length of i (i.e. the distance between $O_{j,k}$ and $O_{n,m}$). In order to find the optimal path, $P^{optimal}$, a version of the A* algorithm has been implemented that uses *W* as its cost function. The path returned by A* will have $W = W_{\min}$, where $W_{\min}$ is the minimum amount of work required to reach the goal.

$$W_{\min} = \sum_{\forall i \in P^{optimal}} F_i D_i \tag{4}$$

The estimated cost that is used for A* is distance multiplied by one unit of force. In order to maintain the constraints of A*, force values must be scaled between 1 and some number greater than 1 to impose a positive minimum force that is at least as great as the estimated cost for flat-ground traversal:

$$F_i = 1 + c_{scale} O_{n,m} \tag{5}$$

c_{scale} is a scaling factor that controls the ratio between the cost of flat ground traversal and the maximum cost of traversal. For example, scaling F_i to have the range [1,5] causes flat ground traversal to be relatively more expensive than scaling F_i to have the range [1, 50]. In practice, we scale F_i to have the range [1 10].

Any metric used to calculate $P^{optimal}$ must account for the fact that paths found in O will determine navigation through R. Thus, care must be taken when choosing a distance metric D_i. In the next section, three possible distance functions for D_i are described.

Section B, Distance Metrics

The most straightforward method for calculating D_i, the length of a path segment in image space, is to project the endpoints of path segment i from O into R, with the help of S, and then use the standard Euclidian distance metric in 3-space. Let this distance be called D_i^R.

Although this metric seems very appropriate, a problem arises when the goal is projected into a high cost region (i.e. an obstacle). The optimal path often involves a traversal directly through the obstacle. This is due to the fact that, as far as the planner is concerned, the goal exists within the high cost region of *O* and not behind the obstacle in *R*. For instance, if a tree is located between the robot and a goal, then it will appear in *O* as if the goal has been projected onto the front of the tree (Figure 3). Thus, the shortest path to the goal appears to require climbing the tree. This phenomenon is a consequence of the two dimensional nature of image space, and as a result, D_i^R is not a suitable distance metric for use in image space.

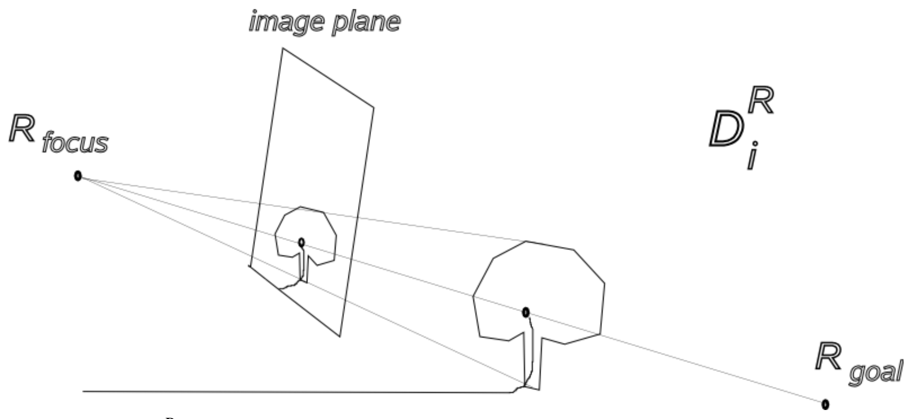

Figure 3, D_i^R is used to find a path in O that is then projected back into *R*. The goal appears to be on the front of the tree; therefore, the shortest path to the goal involves going up into the tree.

$D_i^{R^{flat}}$ aims to correct for the problems of D_i^R by defining the length of *i* to be the Cartesian distance between τ_1 and τ_2, the endpoints of *i* projected from the camera through the image plane and onto R^{flat}—refer to Figure 4.

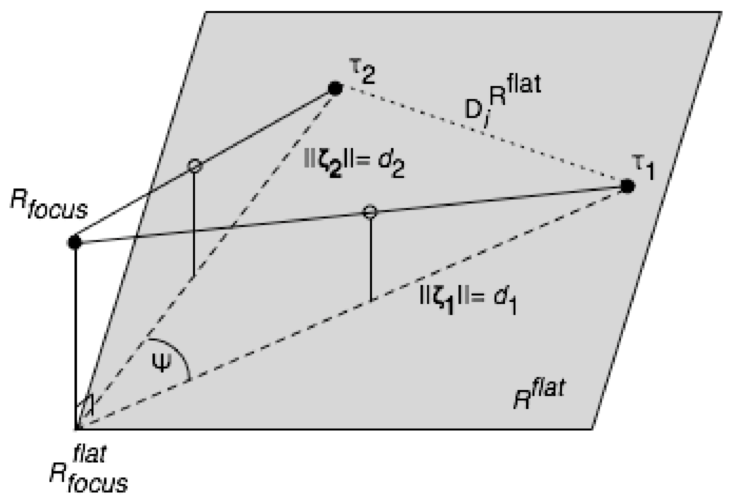

Figure 4. Calculation of $D_i^{R^{flat}}$. R_{focus} is the focal point of the robot's camera array. τ_1 and τ_2 are the endpoints of *i* projected onto R^{flat}.

Let ξ_1 and ξ_2 be the vectors that travel from the base of the robot R_{focus}^{flat} to τ_1 and τ_2, respectively. $D_i^{R^{flat}}$ is calculated as follows:

$$D_i^{R^{flat}} = \sqrt{(d_1 - d_2)^2 + 4d_1 d_2 \sin^2\left(\frac{\psi}{2}\right)}, \tag{6}$$

where d_1 and d_2 are the magnitudes of ξ_1 and ξ_2, respectively, and ψ is the angle between ξ_1 and ξ_2. Equations for *d* and ψ are now derived.

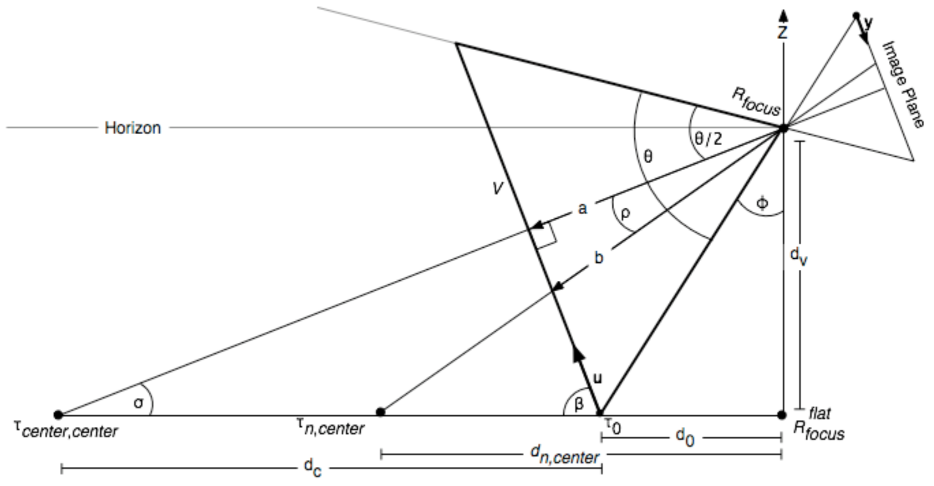

Figure 5. Quantities used in the calculations of *d* and ψ.

Assume that the robot is on R^{flat} at R^{flat}_{focus} and that its FOV is oriented such that the center pixel in the image is below the horizon (Figure 5). The angle of the vertical field of view is denoted θ, while the angle of the camera's FOV parallel to R^{flat} is denoted $\theta_{horizontal}$. Let R_{focus} be the focus of the camera in R, and let τ_0 be the first point on R^{flat} that is visible in the camera's FOV. Let ϕ be the angle that is formed between τ_0, R_{focus}, and R^{flat}_{focus}. Let V be the plane that contains τ_0 and is parallel to the image plane. **y** is the unit vector associated with the vertical length of a pixel in the image plane, and **u** is the projection of **y** through R_{focus} onto V in R. $q_{center}(n)$ is a function that maps pixels' centers from the center column of the image plane onto points on V contained in R. $g_{center}(n)$ is a function that maps pixels' centers from the center column of the image plane onto points on R^{flat}. Note that

$$\tau_0 = q_{center}\left(\mathrm{h} - \frac{\|\mathbf{y}\|}{2}\right) = g_{center}\left(\mathrm{h} - \frac{\|\mathbf{y}\|}{2}\right). \tag{7}$$

Let **a** be the vector between R_{focus} and $q_{center}(\mathrm{h}/2)$ and let **b** be the vector between R_{focus} and $q_{center}(n)$. ρ is the angle between **a** and **b**

$$\rho = \arctan\left(\|\mathbf{u}\|\frac{(\mathrm{h}/2)-(\mathrm{h}-n)}{\|\mathbf{a}\|}\right) \tag{8}$$

where h is the number of rows in O and

$$\|\mathbf{a}\| = \frac{d_c + d_0}{\cos(\sigma)} - d_c\cos(\sigma). \tag{9}$$

d_c is the measured distance from τ_0 to $g_{center}(\mathrm{h}/2)$, d_0 is the measured distance from R^{flat}_{focus} to τ_0, and σ is the angle between **a** and R^{flat}.

$$\sigma = \tan^{-1}\left(d_v / \left(d_c + d_0\right)\right) \tag{10}$$

where d_v is the measured distance between R^{flat}_{focus} and R_{focus}. $d_{n,w/2}$ is the distance between R^{flat}_{focus} and $g_{center}(n)$, and is calculated:

$$d_{n,\mathrm{w}/2} = \|\mathbf{u}\|(\mathrm{h} - n)\frac{\sin(\pi/2 - \rho)}{\sin(\pi/2 - \beta + \rho)} + d_0, \tag{11}$$

where β is the angle between V and R^{flat},

$$\beta = \pi/2 - \sigma, \tag{12}$$

and the magnitude of **u** is calculated by:

$$\|\mathbf{u}\| = \frac{2\mathrm{d}_c}{\mathrm{h}}\sin(\sigma). \tag{13}$$

The inverse function to (11) is given by

$$n = \left\lfloor h - \frac{\left(d_{n,\mathrm{w}/2} - d_0\right)\sin\left(\tan^{-1}\left(d_v / d_{n,\mathrm{w}/2}\right)\right)}{\|\mathbf{u}\|\sin\left(\pi/2 - \tan^{-1}\left(d_v / d_{n,\mathrm{w}/2}\right) + \sigma\right)} + \frac{1}{2}\right\rfloor, \tag{14}$$

If the image distortion caused by a rotation of $\theta_{horizontal}/2$ can be ignored, e.g. if $\theta_{horizontal}/2$ is small such that $\tan(\theta_{horizontal}/2) \approx \sin(\theta_{horizontal}/2)$, or the camera surface is curved such that the distance from the focus to the sensor array is constant, then

$$d_{n,m} \approx d_{n,\mathrm{w}/2}. \tag{15}$$

Let ψ be the angular distance in R^{flat} associated with the R^{flat} projection of the endpoints of i. If the endpoints of i exist in columns m_1 and m_2 of O, then given (15)

$$\psi = \frac{|m_2 - m_1| \theta_{horizontal}}{w}, \tag{16}$$

where *w* is the number of columns in *O*.

Note that the assumptions of (15) allow *d* to be calculated as a function of grid row (*n* or *j*) and four intrinsic values associated with the robotic system in general. Likewise, ψ is dependent on the difference between two grid columns |*m*-*k*| and two intrinsic values. Thus, the calculation of $D_i^{R^{flat}}$ can be performed offline, once for each combination of *n*, *j*, and |*m*-*k*|, and stored for later use in a three dimensional look-up table.

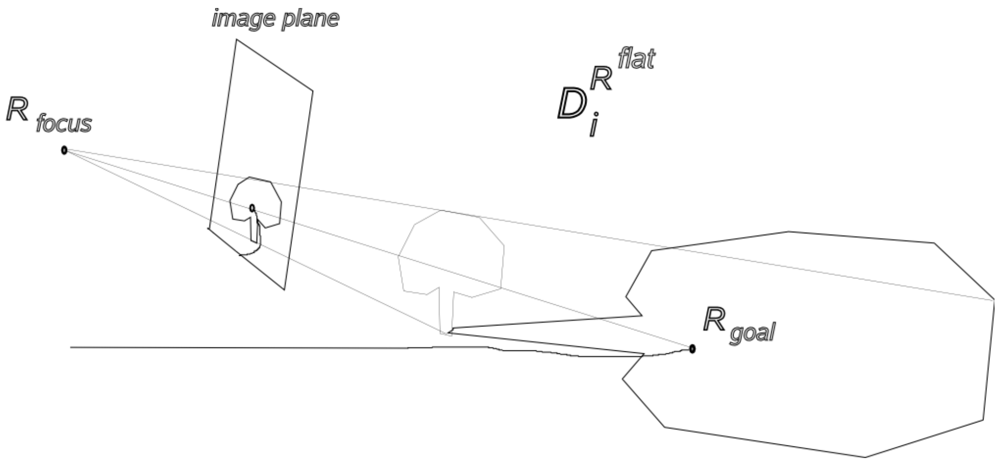

Figure 6. $D_i^{R^{flat}}$ is used to find a path in O that is then projected into R^{flat}. The $D_i^{R^{flat}}$ distance required to reach the goal by going around the base of the tree is about the same as the distance required to reach the goal by going up the trunk of the tree. The optimal path avoids the trunk of the tree if the force values created by the tree are slightly higher than those of the neighboring ground.

By projecting *i* all the way down to R^{flat}, the $D_i^{R^{flat}}$ distance required to go up the front of the tree is the same as the distance required to reach the goal by traversing along R^{flat}. As a result, the distance required to reach the goal by going just to the side of the trunk is about the same as going along (i.e. up) the trunk (Figure 6). Paths chosen to minimize W calculated with $D_i^{R^{flat}}$ avoid the tree climbing problem because, as long as the F_i values associated with the tree are slightly greater than those associated with unimpeded terrain, the path up the tree will be avoided in favor of the path directly next to the tree.

Even though both the D_i^R path and the $D_i^{R^{flat}}$ path initially terminate at the same goal—apparently in the middle of the tree. The $D_i^{R^{flat}}$ path is more likely than the D_i^R path to take the robot around the base of the tree. Once this has happened, the tree will no longer obscure the goal and the goal will no longer appear to be in the tree.

D_i^O, the third distance metric evaluated, is simply the L^2 norm in O, assuming that horizontal and vertical neighbors are spaced unit length apart. The A* search algorithm often computes the actual cost of moving from a given grid location to one of its neighbors. A slight speed increase is achieved by noting that the distance between any two neighboring locations will always be one of two values, depending on if the movement is diagonal, or strictly horizontal or vertical.

$$D_i^O = \sqrt{(n-j)^2 + (m-k)^2} = \begin{cases} 1 & j = n \pm 1 \\ 1 & k = m \pm 1 \\ \sqrt{2} & j = n \pm 1, k = m \pm 1 \end{cases} \tag{17}$$

This simplification is not possible for the estimated cost step of A*, which will generally require the standard computation of the L^2 norm. The calculation of D_i^O forgoes any projections from image space into Cartesian space, allowing D_i^O to be calculated relatively easily compared to $D_i^{R^{flat}}$.

Section C, Image Space Occupancy Grid Preprocessing

Subsection 1, Horizontal Obstacle Dilation

The A* search algorithm finds a path to the goal that minimizes the work expenditure as a function of both the distance traveled and the difficulty of travel. However, this model accounts for neither the physical extension of the robot, nor its ability to rotate in place around its central axis. As suggested by [4], [13], and [26], the width of obstacles in the occupancy grid are increased as a function of robot width λ, allowing the robot to be treated as a point during path search. Note that the apparent width of an obstacle in O is related to the distance between the robot and the obstacle in R. This relationship can be approximated by assuming that obstacles exist on R^{flat}. With this assumption, the distance to an obstacle is $d_{n,m}$, and obstacle dilation becomes a function of n that can be calculated offline.

$$O_{n,m} = \max\left(O_{n,m\pm k}\right) \tag{18}$$

where dilation width k is an integer such that $1 \leq (m \pm k) \leq \mathrm{w}$ and

$$\left\lfloor -\frac{\mathrm{w}}{\theta_{horizontal}} \sin^{-1}\left(\frac{\lambda/2+\varepsilon}{d_{n,m}}\right)\right\rfloor \leq k \leq \left\lceil \frac{\mathrm{w}}{\theta_{horizontal}} \sin^{-1}\left(\frac{\lambda/2+\varepsilon}{d_{n,m}}\right)\right\rceil \tag{19}$$

where $\theta_{horizontal}$ has the same definition as in III-*B*, and ε is the minimum clearance allowed between the robot and an obstacle. This assumes that each row in O represents an approximately equal angle of $\theta_{horizontal}$.

It was found that linear approximations to k performed well in practice. Such an approximation is calculated as a linear function of grid row:

$$-\max\left(1,\left(n-n_{start}\right)c_{expansion}\right)\leq k\leq\max\left(1,\left(n-n_{start}\right)c_{expansion}\right), \tag{20}$$

where n_{start} and $c_{expansion}$ are constants. n_{start} is the row at which $D_i^{R^{flat}}=\lambda/2+\varepsilon$ between the two neighboring grid locations (n_{start}, m) and (n_{start}, m+1). In other words, n_{start} is the row in O for which the width of a single pixel contains as much $D_i^{R^{flat}}$ distance as one half of the robot width plus the minimum obstacle clearance. If the robot is placed in the center of a hallway of width λ+2ε, such that the center of the robot's FOV is parallel to the two walls of the hallway, then the slope factor, $c_{expansion}$, is tuned so that the dilated left and right boundaries of the hallway barley touch on the bottom row of O.

Subsection 2, Vertical Obstacle Dilation

The assumption in (19) that obstacles exist on R^{flat} is only valid for portions of obstacles that are in direct contact with the ground plane (i.e. their bases). In many environments navigation around the base of an obstacle is sufficient to avoid collision, but this is not generally the case. For instance, consider the case of an obstacle that increases in radius as a function of height. The factor ε can be increased to address this discrepancy; however this is not a robust solution because each specific value of ε will only work for a subset of all obstacles. A better (although more time consuming) solution is to force an obstacle's width on R^{flat} to be indicative of the maximum width of that obstacle that presents a navigational hazard to the robot. This can be achieved by performing a vertical dilation to propagate a given obstacle's width information down to its base before the horizontal dilation occurs. In other words, setting the width of the obstacle's base to be equal to the obstacle's maximum width, in order to ensure that navigation around the base is always sufficient to avoid collision with the obstacle.

The vertical length of the dilation must be a function of how far a given point is away from the robot (note that this is also a function of disparity). This is because the obstacle that generates the cost information located at a particular row of *O* may exist at an infinite number of elevation and distance combinations. The vertical length of the dilation should reflect the robot's height, as perceived in *O*, at the distance that the obstacle is detected. Additionally, if the robot can safely travel under an obstacle, then it does not make sense to increase the cost of the area under the obstacle. A vertical length dilation equation is now derived.

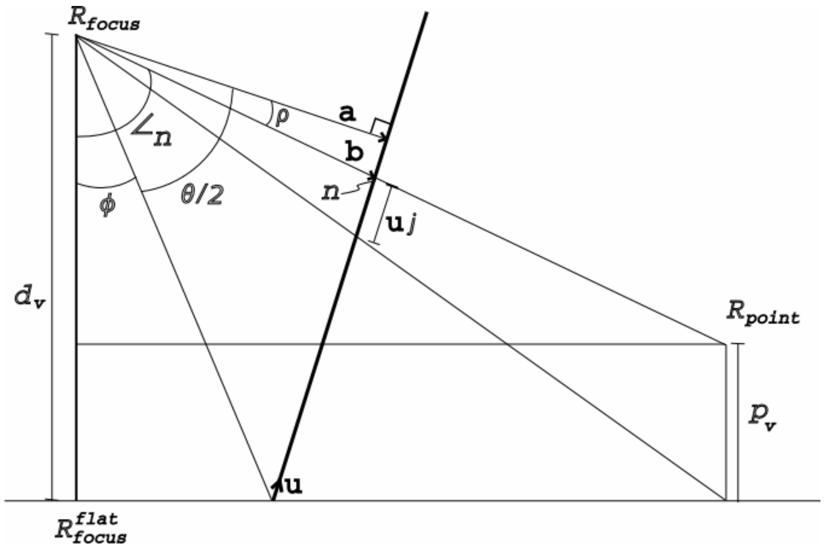

Figure 7, Quantities used to calculate vertical length dilation.

Let R_{focus}, R_{focus}^{flat}, h, n, m, d_v, **a**, **b**, **u**, θ, σ, ρ, and ϕ be defined as they were in III-*B* and Figure 5. Note that the angle between **a** and the vector that travels from R_{focus} to R_{focus}^{flat} can be expressed as:

$$\phi + \theta / 2 = \pi / 2 - \sigma \tag{21}$$

Let R_{point} be a point in R that generates disparity information, S_{point}, that is perceived by the camera. Let p_v be the height of R_{point} above the ground plane, and let $\vartheta(S_{point})$ be the distance from R_{focus} to R_{point} as a function of S_{point}:

$$\vartheta(S_{point}) = \frac{c_{camera}}{S_{point}} \tag{22}$$

where c_{camera} is calculated by multiplying the baseline of the stereo camera array by the focal length of the cameras. Note that c_{camera} is a constant associated with the stereo pair. Let $\angle_n$ be the angle between **b** and the vector that travels from R_{focus} to R_{focus}^{flat}.

$$\angle_n = \pi / 2 - \sigma - \rho \tag{23}$$

$$p_v = d_v - \vartheta(S_{n,m})\cos(\angle_n) \tag{24}$$

Dilation should extend from $O_{(n,m)}$ down to $O_{(j,m)}$ where j is given by:

$$j = \left\lceil \frac{\|\mathbf{a}\|}{\|\mathbf{u}\|} \tan\left(\pi / 2 - \sigma - \arctan\left(\frac{\vartheta(S_{n,m})\sin(\angle_n)}{d_v} \right) \right) + \frac{\mathrm{h}}{2} \right\rceil \tag{25}$$

Note that the vertical length dilation should only be performed for $O_{(n,m)}$ if the corresponding p_v is greater than 0 and less than the height of the robot.

A separate dilation calculation is required for each combination of grid row and disparity encountered by the system. If the computation of j is too time consuming, then a speed increase can be achieved via a pre-computed two dimensional look-up table of arbitrary precision.

Subsection 3, Rotation and Horizon Considerations

O is preprocessed to enable rotation around the central axis of the robot by setting $O_{\mathrm{h},m} = 0$. The horizon is assumed to be generated from the ground plane R^{flat} at infinity, and pixels above the horizon are ignored in O. Note that this is only necessary if the distance metric D_i^O is being used because $D_i^{R^{flat}}$ mandates that the cost of going above the horizon line is infinite. Initially, we believed that allowing sky traversal would provide the system with a method for dealing with navigation behind large obstacles. In practice, however, it was found that if the environment became sufficiently obstacle ridden, then the cheapest path to the goal nearly always traversed some portion of the sky. In extreme cases, the robot moved away from the goal indefinitely. For example, if the robot was not facing in the direction of the goal and a path through the sky induced the robot to move away from the goal, then the goal would appear to approach the horizon from the robot's point of view. Thus, after the movement, the path through the sky would appear even more desirable.

Section D, Using an Image Space Path for Navigation

Servoing is accomplished by steering toward a target location

$P_{target} = P_{nTarget,mTarget}$ located some predetermined distance along P in O. This is either achieved by mapping P_{target} into R^{flat} from O and then steering toward the resulting location, or by implementing the servoing function directly in O. In the experiments described in Chapter V, the latter method is used to calculate *steering angle* and *speed* where:

$$speed = \frac{maxSpeed(\mathrm{h} - nTarget)}{\sqrt{(mTarget - \mathrm{w}/2)^2 + (\mathrm{h} - nTarget)^2}} \tag{26}$$

$$steering\ Angle = \frac{\theta_{horizontal}(mTarget - \mathrm{w}/2)}{\mathrm{w}}. \tag{27}$$

There is only a rotational component to movement if $P_{target} = O_{h,m \neq w/2}$, and there is only a translational component to movement if $P_{target} = O_{n \neq h,w/2}$. Otherwise, movement consists of a combination of translation and rotation. It is defined that the robot has reached the goal when $P_{target} = O_{h,w/2}$, or when the goal cannot be projected into O because R_{goal} is too close to the robot.

CHAPTER IV, EXTENSIONS TO THE IMAGE PLANNER

Section A, The Cylindrical Planner

The Cylindrical Planner is created by adding additional elements to *O* that allow for storage of information that has passed outside of the robot's rotational field of view in *R*. The model uses a cylindrical representation of *O* that can be thought of as a radialy panoramic mosaic of what the robot has experienced. Radialy panoramic mosaics have been used in the past for landmark detection and pose estimation [25], [27], [28]. For implementation purposes, *O* is represented as a simple 2D grid *C,* with the added requirement that $C_{n,1}$ is considered a neighbor of $C_{j,\mathrm{p}}$, and $C_{j,1}$ is considered a neighbor of $C_{n,\mathrm{p}}$, for all rows *n* and *j* in *C*, where $j = \{n+1, n, n-1\}$ and p is the number of columns in *C*. Information is added to *C* by:

$$C_{n,m+f(\varphi)} = \left| S_{n,m} - S_{n,m}^{flat} \right|. \tag{28}$$

That is, information destined for storage in *C* is offset horizontally by a function of φ, robot yaw relative to North. $f(\varphi)$ is calculated as:

$$f(\varphi) = \left\lfloor \left(\frac{\mathrm{p}(\varphi - \pi)}{2\pi} \right) \mathbf{mod}\ \mathrm{p} \right\rfloor, \tag{29}$$

In other words, stereo disparity data is placed into *C* as a function of the compass direction that the robot is facing when the image is captured. This implies that the cardinal directions South, West, North, East, and South, will be mapped from *R* into the following columns of *C*: 0, $\lfloor \mathrm{p}/4 \rfloor$, $\lfloor \mathrm{p}/2 \rfloor$, $\lfloor 3\mathrm{p}/4 \rfloor$, and p, respectively.

$f(\varphi)$ is calculated ignoring the distortion that is caused by approximating multiple planes as a cylinder, and ignoring the fact that the image plane is not parallel to the cylinder's longitudinal axis. If the FOV is such that these distortions cannot be ignored, then two possible solutions exist: either a projection can be used that reconstructs the image plane correctly on the cylinder, or the FOV can be restricted in width such that the distortion is no longer a problem. For the LAGR vehicle, a linear shearing transformation is found to be effective at increasing the rotational accuracy the cylinder. The transformation involves increasing the row in *C* that a given camera pixel belongs to, as a function of the distance (in number of columns) that the pixel is located away from the center of the robot's FOV. Note that this is only an approximation to the true transformation that would place camera information perfectly into the cylinder, however the linear transformation is fast, and hence desirable for the online application.

The A* search algorithm is modified for use on *C* by allowing path sections to exist across the South-South border, and by setting the robot's location in *C* according to its pose: $C_{robot} = C_{\mathrm{h},f(\varphi)}$. The goal is projected into *C* based on the compass heading of the goal relative to the robot and the distance between R_{focus}^{flat} and the goal on R^{flat}. (14), derived in III-*B*, defines this projection. Figure 8 depicts a typical search through C.

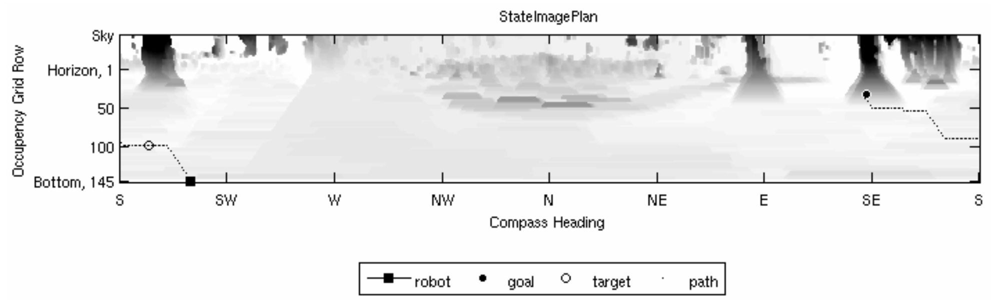

Figure 8. A path from the robot position to a goal located at the base of a tree through the Cylindrical Planner's occupancy grid. Light to dark corresponds to low to high cost, respectively.

A function is now derived that describes how elements outside of the FOV in *C* should be updated for any combination of translation and rotation that the robot executes on R^{flat}.

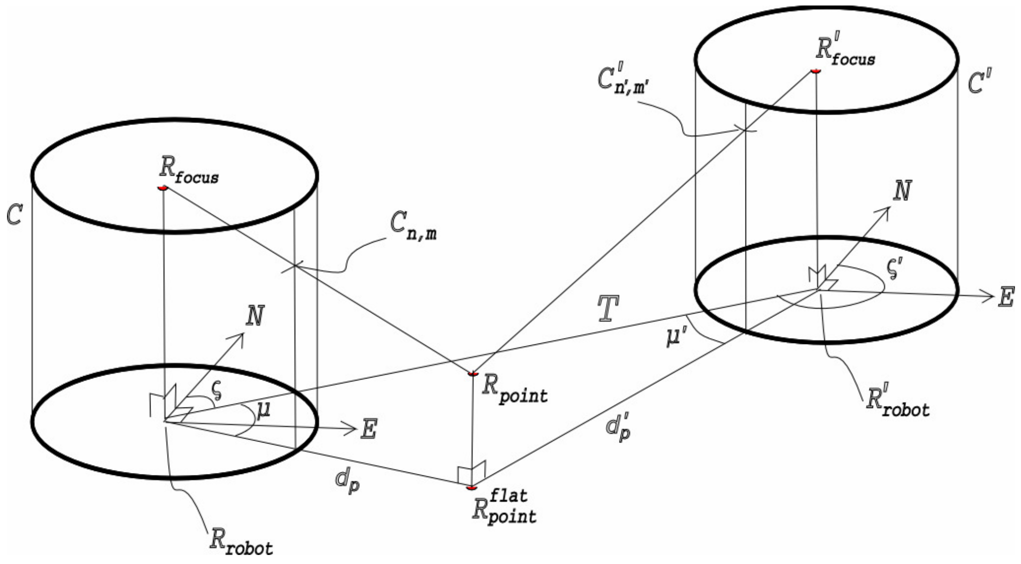

Figure 9, Quantities used to compute the updating function for *C*.

Let point R_{point} be a point in *R* that generates the cost information stored at row *n* and column *m* of *C* when the robot is located at R_{robot}. If the robot is moved to a new location, $R'_{robot} \neq R_{robot}$, then R_{point} will subsequently generate the cost information stored at row *n'* and column *m'* of the new cylinder *C*'. Note that the $C_{n,m}$ may not be equal to $C'_{n',m'}$, *n* may not be equal to *n'*, and *m* may not be equal to *m'*. This is because the directions and distances to R_{point} from R_{robot} and R'_{robot} may not be the same at the new location as they were at the old location.

When the FOV is not 2π, as is generally the case, this updating can be simulated by solving for *n*', *m*', and $C'_{n',m'}$, given *n*, *m*, $C_{n,m}$, and the movement of the robot between R_{robot} and R'_{robot}. Because *C* is a SWNES mosaic (and assuming that any change in height can be ignored) the only components of movement that affect these calculations are ΔE and ΔN, the robot's relative displacement in the East and North directions, respectively. Let h, ||**a**||, ||**u**||, ρ, and β have the same definitions as in (8) through (15) and Figure 5. Let *T* be the robot's translation on R^{flat}.

$$T = \sqrt{\Delta N^2 + \Delta E^2} \tag{30}$$

Let ς and ς' be the radial locations of R'_{robot} and R_{robot}, relative to North, from the vantage point of R_{robot} and R'_{robot}, respectively.

$$\varsigma = \arctan_2(\Delta E, \Delta N), \tag{31}$$

$$\varsigma' = \arctan_2(-\Delta E, -\Delta N), \tag{32}$$

Let R^{flat}_{point} be the point on R^{flat} directly below R_{point}, and let μ be the angle on R^{flat} between the vector from R_{robot} to R'_{robot} and the vector from R_{robot} to R^{flat}_{point}

$$\mu = \frac{2\pi(m-1)}{p} - \pi - \varsigma\,, \tag{33}$$

where *p* is the number of columns in *C*. Let d_p be the distance between R_{robot} and R^{flat}_{point}, and let d'_p be the distance between R'_{robot} and R^{flat}_{point}.

$$d_p = \vartheta(S_{n,m})\cos\left(\frac{\pi}{2}+\rho-\beta\right) = \frac{c_{camera}}{S_{n,m}}\cos\left(\frac{\pi}{2}+\rho-\beta\right) \tag{34}$$

$$d'_p = \sqrt{T^2 + d_p^2 - 2d_pT\cos(\mu)} \tag{35}$$

The function $\vartheta(S)$ is defined in (22). $\vartheta(S_{n,m})$ is the depth associated with pixel (*n*,*m*) when the robot is at R_{robot}, and $\vartheta(S_{n,m}^{flat})$ is the depth associated with the same pixel when the robot exists on a flat plane. $d_{n,m}$ is defined in (15), and is a distance along R^{flat} associated with pixel (*n*,*m*) when the robot is at R_{robot}. Note that the assumptions of (15) are valid because *C* is cylindrical. *n'* and *m'* are computed as follows:

$$n' = \frac{\mathrm{h}}{2} + \frac{\|\mathbf{a}\|}{\|\mathbf{u}\|}\tan\left(\beta - \frac{\pi}{2} + \tan^{-1}\left(\frac{d_p d_v}{d_{n,m}d'_p}\right)\right) \tag{36}$$

$$m' = \frac{p}{2\pi}\left((\pi + \varsigma' - \mu')\bmod 2\pi\right) + 1 \tag{37}$$

and then rounded to the nearest integer. μ' is the angle on R^{flat} between the vector from R'_{robot} to R_{robot} and the vector from R'_{robot} to R_{point}^{flat}. The sine and cosine of μ' can be calculated as follows:

$$\sin(\mu') = \left(\frac{d_p}{d'_p}\sin(\mu)\right) \tag{38}$$

$$\cos(\mu') = \frac{T^2 - d_p^2 + d'^2_p}{2Td'_p}. \tag{39}$$

Note that the relations $\mu' = \arcsin(\sin(\mu'))$ and $\mu' = \arccos(\cos(\mu'))$ do not hold over the range $0 \le \mu' \le 2\pi$. Therefore, μ' must be calculated as follows:

$$\begin{aligned} \mu' &= \arcsin(\sin(\mu')) && \textit{if} \quad \cos(\mu') \geq 0 \\ \mu' &= \pi - \arcsin(\sin(\mu')) && \textit{if} \quad \cos(\mu') < 0 \wedge \sin(\mu') \geq 0 \\ \mu' &= -\pi - \arcsin(\sin(\mu')) && \textit{if} \quad \cos(\mu') < 0 \wedge \sin(\mu') < 0 \end{aligned} \tag{40}$$

The use of $\vartheta(S_{n,m})$ in (34) implies that a second cylinder *Q*, containing the depth values associated with *C*, must either be maintained separately or calculated from *C*:

$$Q_{n,m} = \vartheta(S_{n,m}) = \frac{c_{camera}}{C_{n,m} + S^{flat}_{n,m}} \tag{41}$$

Let *Q* and *Q'* represent the depth value cylinder when the robot is located at R_{robot} and R'_{robot}, respectively. Note that in practice it is easier to maintain *Q* separately from *C*, instead of calculating it from *C*. However, if the latter method is used, then the absolute value on the right hand side of (28) should be relocated to (5), in order to insure that (41) is correct.

When *T*=0 the updating functions for *Q* and *C* are defined respectively Q'=*Q* and C'=*C*; otherwise, *Q'* is populated by

$$Q'_{n',m'} = \sqrt{\left(\frac{d_v d_p}{d_{n,m}}\right)^2 + d'^2_p}\ , \tag{42}$$

and C' is populated by

$$C'_{n',m'} = \vartheta^{-1}(Q'_{n',m'}) = c_{camera}\left(\frac{1}{Q_{n,m}} - \frac{1}{Q^{flat}_{n,m}}\right), \tag{43}$$

$Q'_{n',m'}$ cannot be calculated if d'_p is inside the cylinder, and thus below the cylindrical field of view. In that case, *n'* will not exist as a row in *Q'* or *C'*.

A strategy must be defined for dealing with what happens when one obstacle is occluded by another during translation, in which case more than one point in Q may be mapped to $Q'_{n',m'}$. Generally, obstacles closer to the robot represent more of a navigational hazard; therefore, it is advisable to retain the minimum $Q'_{n',m'}$ value. As previously noted, it may be easier to maintain Q instead of C. If this is done, then force values are calculated:

$$F_i = 1 + c_{scale} c_{camara} \left| \frac{1}{Q_{n,m}} - \frac{1}{Q_{n,m}^{flat}} \right| \tag{44}$$

Currently, it is computationally prohibitive to calculate this transformation within the robot's reaction time. Therefore, An alternative memory-updating scheme is implemented by having C gradually forget information outside of the robot's FOV as a function of the distance that the robot has traveled,

$$C' = C' \max\left(0, \frac{d_{forget} - \sqrt{(\Delta E)^2 + (\Delta N)^2}}{d_{forget}} \right), \tag{45}$$

where d_{forget} is the distance required to erase all rotational memory in a single update [26]. In this scheme, no translational updating takes place, and the values in C outside of the FOV will decay toward zero. d_{forget} is manually tuned to mimic the information loss observed in the translation scheme. Note that d_{forget} is also a function of the rate at which (45) is applied.

Section B, The Hierarchical Planner

A Hierarchical Planner attempts to solve the path planning problem by dividing it up into the parallel problems of global and local planning. The local planner is charged with obstacle avoidance and navigation toward sub-goals. Meanwhile, the global planner concerns itself with a coarse representation of the entire world and returns appropriate sub-goals to the local planner. Hierarchical Planners have been used in a variety of robot path planning schemes [29], [30]. For instance, [31] models the global world as a graph of connected nodes in which each node acts as the local map. [13] also models the global world as a graph of connected nodes, but views the local world in top-down Cartesian space. In [32], both the local and global planners are versions of the top-down occupancy grid model. In standard Hierarchical Cartesian Planners, the local cost-map is high resolution, fixed in size, and remains centered on the robot; the global cost-map maintains a lower resolution, expands with exploration, and remains fixed to some global frame of reference.

We implement a Hierarchical Planner that uses a top-down occupancy grid and a Cylindrical Planner for its global and local planning components, respectively. This configuration combines the local path planning strengths of image based path planning—high resolution obstacle avoidance and servoing—with the global strengths of the birds-eye view occupancy grid—translational memory. Data is stored in the global planner's occupancy grid, *B,* as a projection of *F* onto R^{flat}, and the resolution of *B* is 50 centimeters. Path planning through *B* is accomplished via a version of the work minimization A* search algorithm (4), where D_i is the Euclidean distance between grid locations in *B* multiplied by the resolution of *B*. Sub-goals are chosen to be 10 meters away from the robot.

CHAPTER V, EXPERIMENTS

The three implementations of image based planning systems, described in sections III-*A* , IV-*A*, and IV-*B*, respectively, are compared to a baseline top-down planner on three courses in unstructured outdoor environments. The Baseline Planner has an occupancy grid granularity of 50 centimeters and is nearly identical to the global half of the Hierarchical Planner. All three image based systems use the D_i^O distance metric (described in section III-*B*). Courses 1, 2, and 3 are depicted in Figures 10, 11, and 12, respectively. The actual paths that the robot took are overlaid on a top-down occupancy grid map of the environment. For completeness of map information, all maps were generated independently of the test runs by teleoperation. The granularity of each occupancy grid is 50 centimeters. Course 1 is a simple course that consists of randomly placed obstacles with radii varying from 10 centimeters to 1 meter. Courses 2 and 3 are similar to Course 1, except that an obstacle of 10 meter girth is added on Course 2, and Course 3 contains two adjoining obstacles each 1 meter wide and approximately 30 and 10 meters long, respectively. Low to high cost is represented by light to dark, respectively.

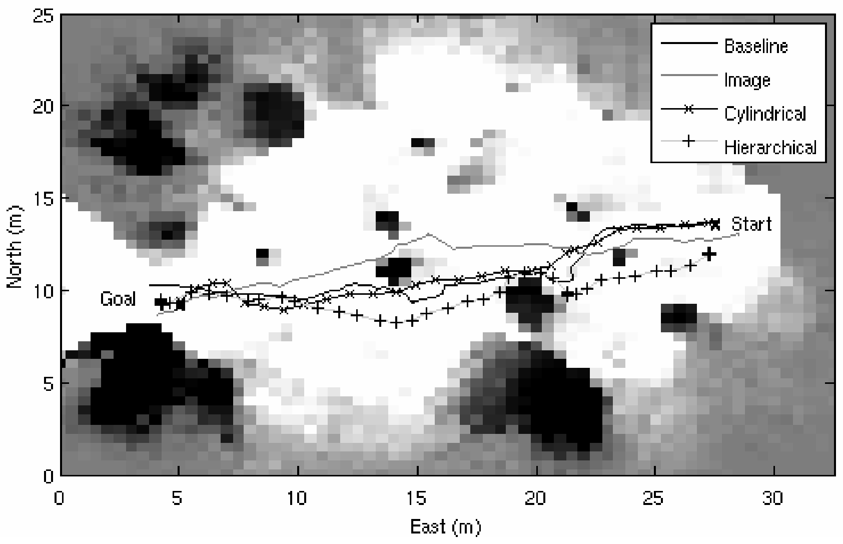

Figure 10. Course 1: obstacles of small radii.

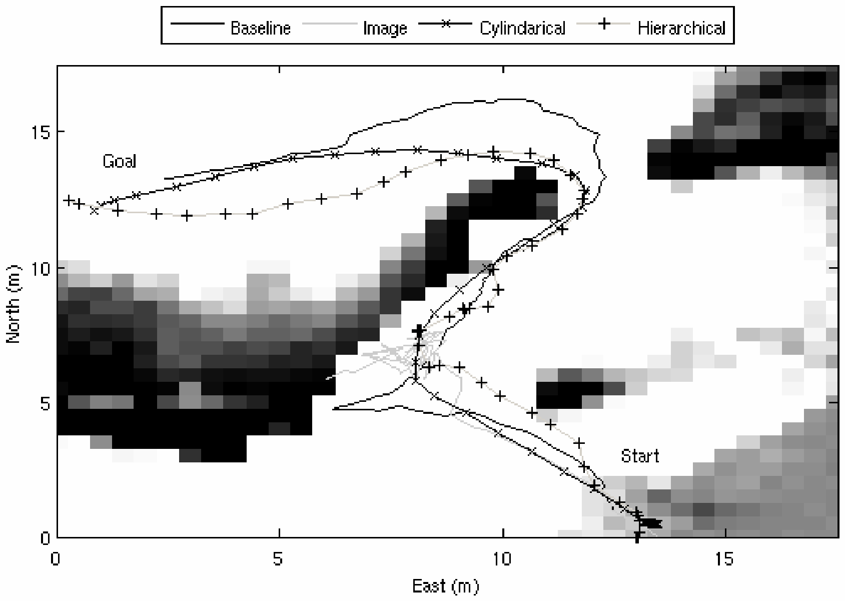

Figure 11. Course 2: an obstacle of 10 meter girth.

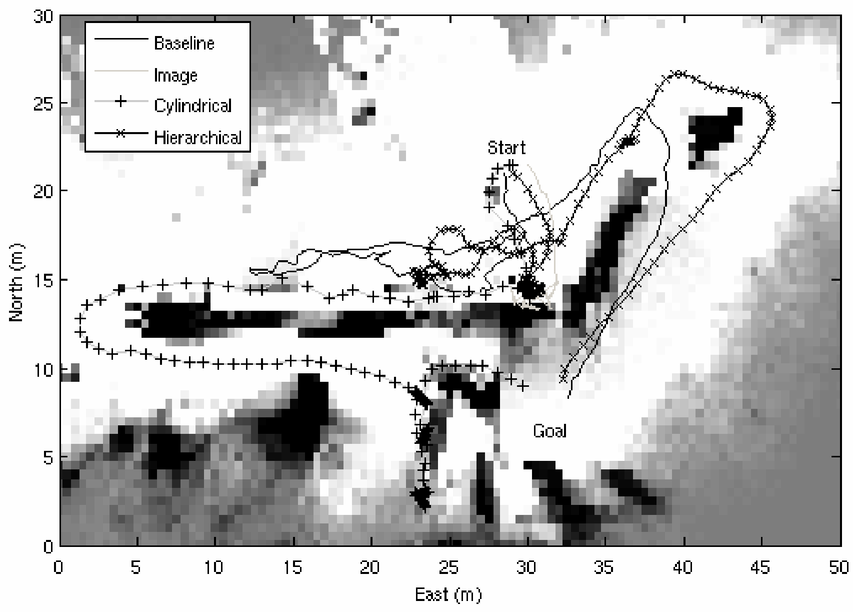

Figure 12. Course 3: two adjoining long thin obstacles.

A version of the Hierarchical Planner using the $D_i^{R^{flat}}$ distance metric was also tested on course 3. The rout taken by this system is depicted in Figure 13.

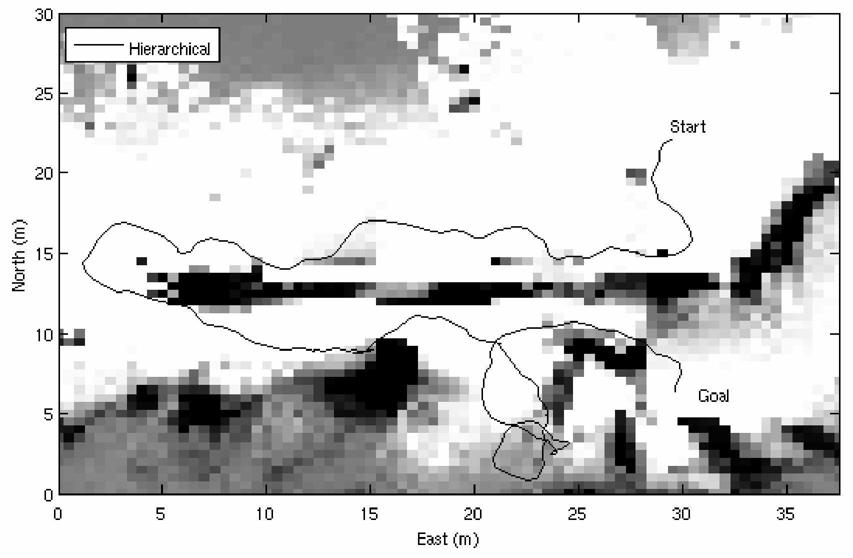

Figure 13. $D_i^{R^{flat}}$ distance metric performance on Course 3.

CHAPTER VI, DISCUSSION AND RESULTS

Path planning for robot navigation is a real-time system in which the robot must be able to observe the world and react quickly enough to guarantee safety and reliability. At the robot's minimum speed (approximately 0.125 m/s), robust navigation requires that the robot perceive the world and react at least every quarter meter, or 0.5 Hz. Ideally, it is desirable for the robot to translate at a rate of 0.5 m/s or greater, which means the robot must plan at least 2 Hz. Improving frame-rate beyond this is not unreasonable given state of the art CPUs. Nonetheless, care is taken to limit the time complexity of the algorithms, particularly the distance calculations. As noted in section III-*B*, $D_i^{R^{flat}}$ can be calculated off-line, thus reducing the online distance calculation to a table-lookup.

In practice, it was found that the $D_i^{R^{flat}}$ distance metric causes the path to be extremely sensitive to noise. When noise occurs in an otherwise traversable area, it creates a pseudo-obstacle that the planning system attempts to avoid like any other high cost region. $D_i^{R^{flat}}$ mandates that the cost associated with traveling between neighboring grid locations decreases as a function of occupancy grid row (Figure 14). Thus, the least expensive path around an obstacle located in the far field will try to avoid the obstacle in the near field—often by an immediate rotation. This would not be a problem in the absence of noise. However, because pseudo-obstacles pop in and out of existence, erratic behavior is induced by the planning system's continuous attempts to avoid new pseudo-obstacles (i.e. with immediate rotation after immediate rotation). Figures 12 and 13 show, respectively, the performance of the Hierarchical Planner using the D_i^O and $D_i^{R^{flat}}$ metrics on Course 3. The route taken by the hierarchical planning system in Figure 12 is much smoother than the one in Figure 13.

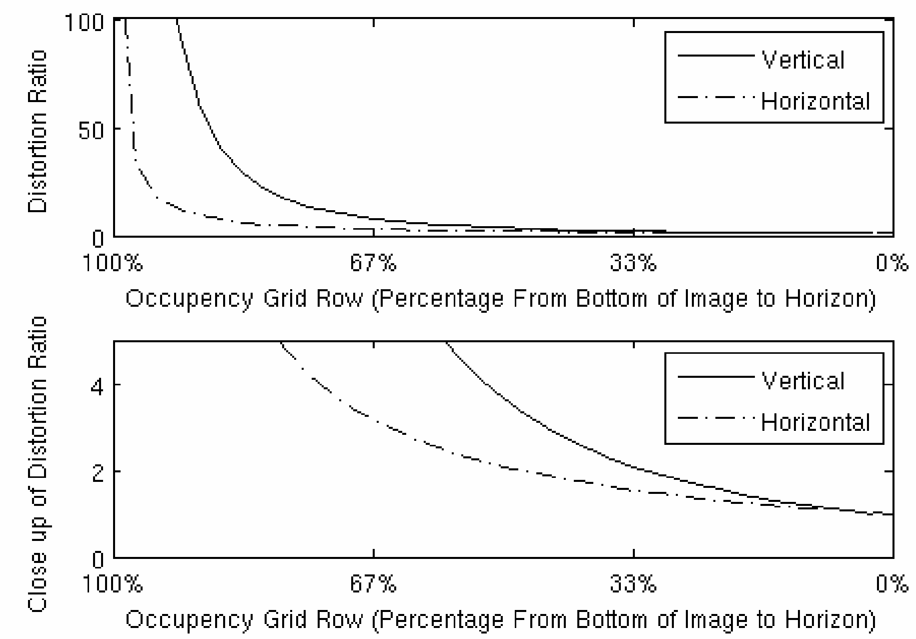

Figure 14. Distortion Ratio as a function of occupancy grid row (percentage from bottom of image to horizon), where Distortion Ratio is $D_i^{R^{flat}}$ for vertical and horizontal neighbors divided by $D_i^{R^{flat}}$ for the bottom most vertical and horizontal neighbors, respectively (top). Note that this is proportional to $D_i^{R^{flat}} / D_i^O$. Close up of distortion ratio (bottom).

D_i^O tends to distort R^{flat} distance, especially in the far field (Figure 14). However, D_i^O works well in practice. By defining the distance between neighbors to be invariant of grid location, it avoids the noise induced near-field path corrections that were observed with $D_i^{R^{flat}}$. This is because paths are penalized equally for near or far field detours, so the path is free to follow the geodesic around an obstacle or pseudo-obstacle without making a significant and immediate correction. Also, because the range of our stereo sensors is effectively 15 meters, severe far-field distance distortion is somewhat irrelevant. Note that in Figure 10 the distortion ratio is less than 2 for approximately one third of the occupancy grid, where Distortion Ratio is defined to be $D_i^{R^{flat}}$ for vertical and horizontal neighbors divided by $D_i^{R^{flat}}$ for the bottom most vertical and horizontal neighbors, respectively. Because D_i^O is always 1 for horizontal and vertical neighbors, this is also proportional to $D_i^{R^{flat}} / D_i^O$.

The experiments illustrated that the basic Image Planner is able to navigate through simple courses, such as Course 1; however, it is not a robust planning system. For instance, when R_{goal} is not in the robot's FOV it cannot be mapped into O. This will happen if the robot starts in such an orientation, is close to the goal, or has rotated away from R_{goal} in order to avoid an obstacle. Consequently, the Image Planner fails unless some predefined course of action is hard-coded into the system. The first case is solved by requiring the robot to rotate in the direction of the goal upon start-up. The second case can be ignored because it will only happen once the robot has completed its task. The final case is non-trivial and plans of action must involve movement containing both a translational component and a rotational component.

Without both translation and rotation, the robot risks never finding a path to the goal. Purely forward movement will carry the robot away from the goal indefinitely, whereas movement in the reverse direction risks obstacle collision. Pure rotation may induce oscillatory behavior, as the robot alternately rotates away from the obstacle and then back toward the goal after forgetting that the obstacle exists. The Image Planner was observed to display this behavior on Courses 2 and 3, Figures 11 and 12, respectively—note that each test was manually aborted after the robot oscillated for two minutes. A naive procedure that translates some distance before allowing rotation in the direction of the goal may perturb the system enough to overcome this condition. However, this does not address the deeper problem at the heart of rotational-oscillatory behavior—namely, the lack of rotational memory. The rotational memory of the Cylindrical Planner allows it to remember the obstacle's existence, even when the obstacle is outside the robot’s field of view. Note that in Figure 11 the Cylindrical Planner navigates around the obstacle to the goal.

The Cylindrical Planner was able to find the goal in all three tests. However, on Course 3 (Figure 12) it was the only planning system that opted to travel around the lengthier of the two obstacles. We speculate that this behavior would have degenerated into translational oscillation if the obstacle had been longer. Consider the case of Figure 15, top. A goal is placed directly North of the center of a long thin wall that runs East to West (e.g. the length of the wall is 1km and the width of the wall is 1m). The robot starts South of the center of the wall. At first, given the information in *C*, it will appear possible to navigate around the wall in either direction. However, as the robot moves toward one end of the wall, the goal will appear to move toward the opposite end of the wall from the robot's point of view (Figure 15, bottom). Eventually, it will appear cheaper to reverse direction and attempt to reach the goal by going around the opposite end of the wall. This will repeat each time the robot travels a certain distance away from the goal in either direction.

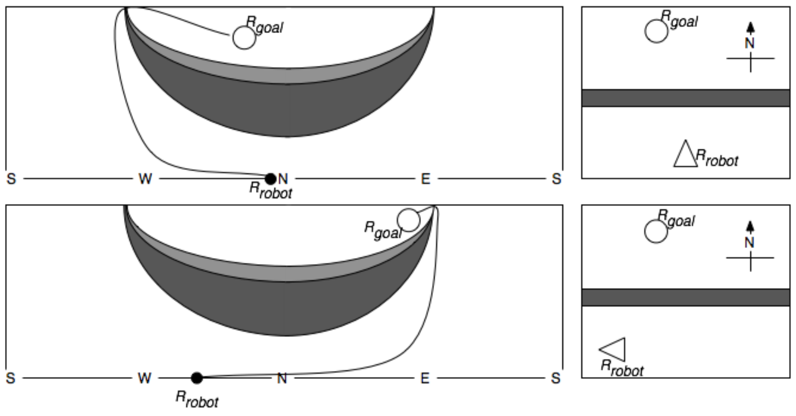

Figure 15, Translational oscillation induced in the Cylindrical Planner by a long thin wall. The initial path around the wall (top), and the path at a later time (bottom).

The only way to avoid this problem is to introduce some form of global translational memory, such as a global 3D or 2D top-down Cartesian Planner. Local versions of these planners do not suffice—they are, by definition, only concerned with portions of the world near the robot and will always be vulnerable to translational oscillation induced by obstacles larger than their translational memory. The Hierarchical Planner, on the other hand, will eventually find a way around a large obstacle—if one exists—with the help of its global Cartesian Planner. However, solutions can still be suboptimal. For example, the robot may backtrack many times as it explores for a way around the wall [33]. This is observed in Figure 12 for both the

Baseline Planner and the Hierarchical Planner. This suboptimal behavior can be described as translational quasi-oscillatory, and is related to (but not identical to) the translational oscillatory problem previously addressed. Any planning system that must make decisions based on limited information is susceptible to quasi-oscillatory behavior because any currently optimal solution may change as new information is discovered. Work has been done on this complex global planning phenomenon by [34].

If the system has sufficient prior knowledge of the domain (e.g. a perfect map) then the planner is able to make piece-wise optimal decisions that form a globally optimal decision. Highly structured environments, for instance those encountered by autonomous highway driving algorithms, may contain sufficient information to use a local planner in a global setting. Similarly, the Cylindrical Planner is equipped to navigate through environments similar to Courses 1 and 2 without the help of the Hierarchical Planner.

CHAPTER VII, FUTURE WORK

The high fidelity occupancy grid used in image space planning provides a natural framework to include more sophisticated models about the traversability of terrain. A natural extension to this work is to combine color and texture models with stereo information to incorporate more environmental knowledge, and allow for more robust path planning.

One unaddressed limitation of the Cylindrical planning system is its inability to plan behind obstacles that the robot cannot see over. This is a direct result of the fact that the Image Planner only has the ability to store complete information about two of the three dimensions that it uses. Attempting to solve this problem by allowing sky traversal caused failure in certain circumstances. A similar idea that may work is to artificially place low cost regions at specific places in the map. For instance, setting the force associated with a grid location to one if it happens to be in the same row that the goal is projected into, and within a certain number of columns away from the goal. This is similar to an idea from Applied Perception, Inc., a fellow LAGR participant, who suggested allowing easy horizontal movement everywhere in O, in order to reflect the fact that the robot may be able to go behind things in the cost map. Interestingly, the $D_i^{R^{flat}}$ distance required to move to a horizontal neighbors is always much less than the $D_i^{R^{flat}}$ distance required to move to a vertical neighbor, given the same starting location. This suggests that if the noise observed in our system could be reduced, then the $D_i^{R^{flat}}$ metric may be a principled way to account for the possibility of traversing behind obstacles.

It is worth noting that humans, given a similar world view, are able to use logic to reason about such things. For instance, they know that they can probably traverse horizontally through a forest, but that they probably will not be able to travel directly through a cliff face or a cement wall. Hence, It may be possible to use supervised or simi-supervised machine learning to build traversibility models of scenes in order to give the robot a similar knowledge base.

If a faster way of updating the part of the cylinder that is outside of the robot's FOV could be implemented, then it would allow the cost map of the cylindrical model to be more robust during translation. It is also possible that approximations to the full translational updating scheme could provide more useful information than what is currently provided by exponentially forgetting the part of the map that is outside of the robot's FOV. Several possible schemes include: assuming that all obstacles lie on two walls that are parallel to the direction of robot translation, one on either side of the robot and both a fixed distance away from the robot; adding an additional two walls, one in front of the robot and another behind it, that are perpendicular to the first two walls; and performing the entire translation at a limited disparity granularity—thus allowing the offline pre-computation of updated cost values and their corresponding locations.

CHAPTER VIII, CONCLUSIONS

We have demonstrated the efficacy of using image based path planning. However, We have also discovered that any robust path-planning algorithm must address two environmental scenarios: those that lead to rotational oscillation and those that lead to translational oscillation. The Image Planner is susceptible to both, a limitation not shared by the traditional top-down Cartesian Planners. We address these situations with a series of extensions to the Image Planner. By augmenting the memory of the Image Planner to include the world beyond the FOV, the Cylindrical Planner is capable of overcoming rotational oscillations and reducing translational oscillations. In general, the translational oscillation problem can only be solved by a planner that maintains global translational memory. Although planning in image space does not displace the Cartesian Planner, it does relegate it to the more aptly suited function of global planning. Local planning in image space is robust, and provides a simple framework for maintaining a high resolution world-view. A Hierarchical Planner combines the strengths of both systems and is able to plan a more natural path, which can then be executed more fluidly.

The end goal of these efforts is a principled interaction between Cylindrical and Cartesian path planning. This is the first such successful framework, and sets the stage for future research efforts.

REFERENCES

[1] D. Murray and C. Jennings, “Stereo vision based mapping and navigation for mobile robots,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA ’97)*, New Mexico, pp. 1694-1699, April 1998.

[2] W. van den Mark, F. Groen, and J. C. van den Heuvel, “Stereo based navigation in unstructured environments,” at IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary, 2001.

[3] A. Elfes, “Using occupancy grids for mobile robot perception and navigation,” in *IEEE Computer*, pp. 46–57, June 1989.

[4] S. Kolski, D. Ferguson, M. Bellino and R. Siegwart, “Autonomous driving in structured and unstructured environments,” Lausanne, Switzerland & Pittsburgh, USA, in *IEEE Intelligent Vehicles Symposium*, 2006.

[5] M. Herman, "Fast, three-dimensional, collision-free motion planning" in *IEEE Proc. Int. Conf. Robotics Automat., 2,* pp. 1056-1063, April 1986.

[6] P. Hart, N. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” in *IEEE Trans. On System Science and Cybernetics SSC-4, 2*, pp. 100-107, July 1968.

[7] E. Dijkstra, “A note on two problems in connection with graphs,” in *Numer. Math. 1*, pp. 269-271, 1959.

[8] G. Krishnaswamy and A. Stentz, “Resolution independent grid-based path planning,” Tech. Report CMU-RI-TR-95-08, Robotics Institute, Carnegie Mellon University, April 1995, unpublished.

[9] Anthony Stentz, “The focussed D* algorithm for real-time replanning,” in *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1995.

[10] O. Khatib, “Real-Time obstacle avoidance for manipulators and mobile robots,” in *The Int. Journal of Robotics Research, 5(1)*, pp. 90-98, Spring 1986.

[11] Y. Koren, J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 1991.

[12] D. Murray and J. Little, “Using real-time stereo vision for mobile robot navigation,” in *Proc. of the IEEE Workshop on Perception for Mobile Agents*, Santa Barbara, CA, June 1998.

[13] M. Sugiyama, Y. Kawano, M. Niizuma, M. Takagaki, M. Tomizawa, and S. Degawa, “Navigation system for an autonomous vehicle with hierarchical map and planner,” in *Proc. of the Intelligent Vehicles '94 Symposium*, pp. 50 – 55, 24-26, Oct. 1994.

[14] S. Tsugawa, T. Yatabe, T. Hirose, and S. Matsumoto, “An automobile with artificial intelligence,” in *Proc. Sixth Int Joint Conf. Artificial Intelligence*, pp. 893-895, 1979.

[15] C. Thorpe, M. H. Herbert, T. Kanade, and S. A. Shafer, “Vision and navigation for the Carnegie-Mellon Navlab,” in *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362-372, May 1988.

[16] D. Mateus, G. Avina, and M. Devy, “Robot visual navigation in semi-structured outdoor environments,” in *ICRA*, 2005.

[17] D.A. Pomerleau, “ALVINN: An autonomous land vehicle in a neural network,” Technical Report CMU-CS-89-107, Carnegie Mellon Univ., 1989, unpublished.

[18] T.M. Jochem, D.A. Pomerleau, and C.E. Thorpe, “Vision-based neural network road and intersection detection and traversal,” in *Proc. IEEE Conf. Intelligent Robots and Systems*, vol. 3, pp. 344-349, Aug. 1995.

[19] N. Cowan, I. Weingarten, and D. Koditschek, “Visual servoing via navigation functions,” in *IEEE Transactions on Robotics and Automation, 18(4)*, pp. 521-533, 2002.

[20] J. Feddema and O. Mitchell, “Vision-guided servoing with feature-based trajectory generation,” in *IEEE Trans. Robot. Automat.*, vol. 5, pp. 691–700, Oct. 1989.

[21] H. Zhang and J. Ostrowski, "Visual motion planning for mobile robots," in *IEEE Trans. Robot. Automat.*, vol. 18, pp. 199–208, April 2002.

[22] R. Vidal, O. Shakernia, and S. Sastry, "Formation control of nonholonomic mobile robots omnidirectional visual servoing and motion segmentation," in *Proc. IEEE Conf. Robotics and Automation*, pp. 584–589, 2003.

[23] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor, "Omni-directional vision for robot navigation," at IEEE Workshop on Omnidirectional Vision (OMNIVIS'00), Hilton Head, South Carolina, June 2000.

[24] Y. Matsumoto, K. Sakai, M. Inaba, H. Inoue, "View-based approach to robot navigation," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2000)*, vol 3, pp. 1702 – 1708.

[25] P. Gaussier, C. Joulain, S. Zrehen, J. P. Blanquet, A. Revel, "Visual navigation in an open environment without map," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)*, Grenoble, pp. 545-550, 1997.

[26] A. Kelly. "Adaptive perception for autonomous vehicles," Tech. Report CMU-RI-TR-94-18. The Robotics Institute, Carnegie Mellon University, 1994, unpublished.

[27] A. Kelly. "Mobile robot localization from large-scale appearance mosaics," in *Int. Journal of Robotics Research, 19*, pp. 1104–1125, 2000.

[28] A. Argyros, K. E. Bekris, S. C. Orphanoudakis, and L. E. Kavraki, "Robot homing by exploiting panoramic vision," in *Autonomous Robots, 19(1)*, pp. 7–25, 2005.

[29] S. Chen, "A spherical model for navigation and spatial reasoning," 1990.

[30] B. H. Krogh and C. E. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, pp. 1664–1669, 1986.

[31] J. Hong, X. Tan, B. Pinette, R. Weiss, and E. M. Riseman, "Image-based homing," in *Proc. IEEE Int. Conf. on Robotics and Automation*, New York, pp. 620–625, 1991.

[32] E. Gat, M. Slack, D.P. Miller and R.J. Firby, "Path planning and execution monitoring for a Planetary rover," in *IEEE Int. Conference on Robotics and Automation*, Cincinnati, USA, 1990.

[33] K. N. Kutulakos, V. J. Lumelsky, and C. R. Dyer, "Vision guided exploration: A step toward general motion planning in three dimensions," in *Proc. IEEE Robotics Automat. Conf.*, pp. 289-296, 1993.

[34] B. Nabbe, "Extending the path-planning horizon," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2005.