Local Path Planning in Image Space for Autonomous Robot Navigation in Unstructured Environments

Michael W. Otte, Scott G. Richardson, Jane Mulligan, Gregory Grudic

***Abstract*—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 (e.g. 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 a projection of cost information from the image plane down onto a top-down 2D Cartesian cost map. We discuss three cost metrics for path selection in image space. We present a basic image based planning system, discuss its susceptibility to rotational and translational oscillation, and present and implement two extensions to the basic system 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. We find 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.**

I. Introduction

AUTONOMUS 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 its own kinematic limitations. 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—we do not have such high confidence *a priori* knowledge about the relationship between the appearance of a scene and its traversability.

Visual perception involves decoding the 2D projection of 3D Cartesian space as it is captured by a robot's imaging sensors [1], [2]. This 2D projection is referred to as *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 then projected down from image space onto the ground plane and inserted 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 because in order to maintain a map with a computationally feasible search space, the world must be resampled at a non-native resolution. This produces a projected image with low fidelity. 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 the image space has not been studied on a robot platform in unstructured environments. There are, however, examples of image based visual servoing in semi-structured and structured environments.

Autonomous highway driving algorithms such as Navlab and its many implementations operate in a semi-structured environment [14]–[18]. 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.

Both Navlab and VSRR type models develop a control strategy as a function of the perceived scene. However, both

Manuscript received March 28, 2007. This work was supported by DOD AFRL award no. FA8650-07-C-7702 (DARPA LAGR) and NSF IIS-0535269.

M. W. Otte, S. G. Richardson, J. Mulligan, and G. Grudic are with the Computer Science Dept., University of Colorado, Boulder, CO 80309 USA (e-mail: Michael.Otte@colorado.edu, Scott.Richardson@colorado.edu, Jane.Mulligan@colorado.edu, Gregory.Grudic@colorado.edu).

Navlab and VSRR 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.

Our task involves not only identifying traversable terrain from non-traversable terrain, but also finding and staying on the optimal path to the goal. We present an approach to path planning that allows local path search to take place directly in the image plane, thereby preserving the flexibility of the occupancy grid paradigm and avoiding the corresponding transformation distortion induced by the projection into a Cartesian coordinate system. In our scheme, a real-world GPS coordinate is projected into image space as a goal. Next, a variant of A* is used in image space to identify the 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 run-time complexity of the system to allow the robot a suitable reaction time.

We call our basic image based planning system the *Image planner*, and introduce it in Section III-*A*. We then discuss its susceptibility to rotational and translational oscillations. That is, because the Image planner lacks memory of the world, 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 forgetting that the obstacle exists. These limitations are addressed with a series of extensions to the Image planner. The *Cylindrical planner*, introduced in Section III-*B*, is created by augmenting the rotational memory of the Image planner to include the world beyond its FOV, and a hybrid hierarchical planner, introduced in Section III-*C*, combines the strengths of a local image planner with those of a global Cartesian planner. In Section IV we describe our experiments, and in Section V we discuss our results.

II. Experimental Apparatus

Our mobile robot platform is provided in conjunction with the DARPA Learning Applied to Ground Robotics (LAGR) program. It measures roughly 1m x 1.5m x 1.5m. Its sensors include: two forward facing Point Grey BumbleBee 2 stereo cameras, a Garmin GPS receiver, a magnetic compass, and wheel odometers. 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.

III. Planning Systems

A. Image Planner

Let R denote the 3D Cartesian real-world space. Our work

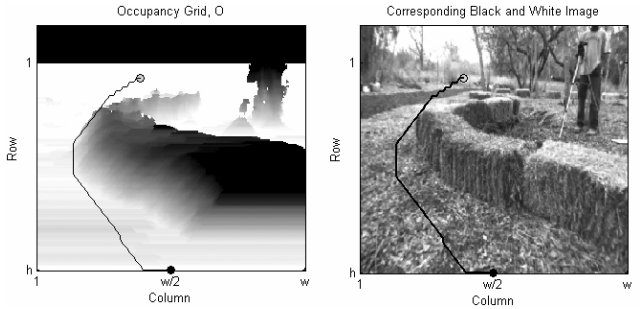

Fig. 1. A path through O from the robot position to a goal in the far-field, where light to dark corresponds to low to high cost (left). The path projected into a black and white image of the scene (right).

focuses on 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. We build an occupancy grid O in image space based on S, and then find the path $P^{optimal}$ in the set of paths P through O that minimizes a quantity W that is analogous to mechanical work (i.e. force multiplied by distance). See Fig. 1. Because any path found in O is a projection of some path existing in R, it is possible to navigate through R using P. This can be done directly, or via a projection of P from image space to R. We define the traversability of R with an occupancy grid O:

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

where S is organized in an h by w Cartesian grid based on the camera's physical pixel layout, and 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. $S_{n,m}^{t}$ 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}. In our experiments, 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 $O_{start}=O_{h,w/2}$. We interpret the traversability values stored in O as forces F that impede robot progress, and we search for paths through O that minimize the amount of work W that must be exerted to reach O_{goal} from O_{start}.

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

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.

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

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, $P^{optimal}$, we implement a version of the A* algorithm that uses W as its cost function. A* will return the path that minimizes $W_{\min}$, the ammount of work required to reach the goal.

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

In our implementation of A*, F_i=1+$O_{n,m}$ to impose a positive minimum force in the case of flat-ground traversal.

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. We investigate three possible distance functions for D_i. The most straightforward method for calculating D_i is to project the endpoints of i into R, with the help of S, and then use the standard Euclidian distance metric in 3-space. We call this distance 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 is often to traverse 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 in O and not behind the obstacle on R^{flat}. 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. Thus, the shortest path to the goal appears to require climbing the tree.

The second function we evaluate, $D_i^{R^{flat}}$, estimates 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 Fig. 2. Projecting i back to R^{flat} avoids the tree climbing problem because the 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}. Note that the tree will be avoided due to high F_i values.

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:

$$D_i^{R^{flat}} = \sqrt{(d_1 - d_2)^2 + 4 d_1 d_2 \sin^2\left(\frac{\psi}{2}\right)}, \quad (5)$$

where d_1 and d_2 are the magnitudes of ξ_1 and ξ_2, respectively, and ψ is the angle between them. We develop equations for d and ψ in the Appendix and show that, given certain assumptions, a function exists for d that is dependent on grid row (n or j) and four intrinsic values associated with the robotic system in general. Likewise, a function exists for ψ that is dependent on $|m-k|$ and two intrinsic values. 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.

The final distance metric we investigate, D_i^O, is the L^2 norm between grid locations in O, assuming that horizontal and vertical neighbors are spaced unit length apart.

$$D_i^O = \sqrt{(n-j)^2 (m-k)^2} \quad (6)$$

The calculation of D_i^O forgoes the projection between image space and Cartesian space, allowing D_i^O to be calculated relatively easily compared to $D_i^{R^{flat}}$. Note that D_i^O will always be 1, 1, and $\sqrt{2}$ for vertical, horizontal, and diagonal neighbors, respectively.

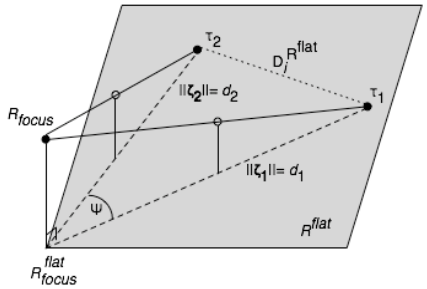

Fig. 2. Calculation of $D_i^{R^{flat}}$. R_{focus} is the focal point of the robot's camera. τ_1 and τ_2 are the endpoints of i projected onto R^{flat}.

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], we increase the width of obstacles in the occupancy grid as a function of robot width λ, allowing the robot to be treated as a particle 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. We approximate this relationship 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.

$$O_{n,m} = \max\left(O_{n,m+k}\right) \quad (7)$$

where k is an integer such that $1 \le (m+k) \le$ w and

$$\left\lfloor -\frac{\mathrm{w}}{\theta}\sin^{-1}\left(\frac{\lambda+\varepsilon}{2d_{n,m}}\right)\right\rfloor \le k \le \left\lceil \frac{\mathrm{w}}{\theta}\sin^{-1}\left(\frac{\lambda+\varepsilon}{2d_{n,m}}\right)\right\rceil \quad (8)$$

where θ is the angle of the camera's FOV parallel to R^{flat}, 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 θ. The assumption 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; however, this is not generally the case. The factor ε can be increased to address this discrepancy as the operational environment requires.

O is preprocessed to enable rotation around the central axis of the robot by setting $O_{h,m}$=0. Pixels above the horizon are ignored in O because sky traversal should be impossible. The horizon is assumed to be generated from the ground plane R^{flat} at infinity.

Servoing is accomplished by steering toward a target location P_{target}= $O_{nTarget,mTtarget}$ 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. We use the latter method in our experiments to calculate *steering angle* and *speed* where

$$speed = \frac{maxSpeed(nTarget - \mathrm{h}/2)}{\sqrt{(mTarget - \mathrm{w}/2)^2 - (nTarget - \mathrm{h}/2)^2}}, \quad (9)$$

$$steering\ Angle = \frac{\theta(mTarget - \mathrm{w}/2)}{\mathrm{w}}. \quad (10)$$

We assume that the robot has reached the goal when $P_{target}=O_{h,w/2}$. If $P_{target}=O_{h,m\neq w/2}$, then there is only a rotational component to movement. If $P_{target}=O_{n\neq h,w/2}$, then there is only a translational component to movement. Otherwise, movement consists of a combination of translation and rotation.

B. 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 field of view in R. The model uses a cylindrical representation of O that can be thought of as a radially panoramic mosaic of what the robot has experienced. Radially 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:

$$C_{n,m+f(\varphi)} = \left| S^{flat}_{n,m} - S^{t}_{n,m} \right|. \quad (11)$$

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:

$$f(\varphi) = \left\lfloor \frac{\mathrm{p}}{2\pi}((\varphi - \pi)(\mathbf{mod}\ 2\pi)) \right\rfloor + 1, \quad (12)$$

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.

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 distance between the R^{flat}_{focus} and the goal on R^{flat} and the compass heading of the goal relative to the robot. (21), derived in the Appendix, defines this projection. Fig. 3 depicts a typical search through C.

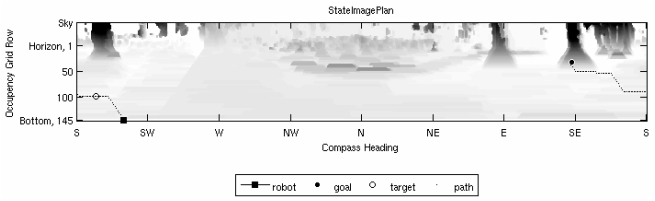

Fig. 3. A path from the robot position to a goal located at the base of a tree through the Cylindrical planner's occupancy grid. Light to dark corresponds to low to high cost.

A function exists that describes how elements in C should be updated for any combination of translation and rotation that the robot executes in R^{flat}. However, we find that it is computationally prohibitive to calculate within the robot's reaction time.

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,

$$C_{t+1} = C_t \left(\max\left(0, \frac{d_{forget} - \sqrt{(\Delta East)^2 + (\Delta North)^2}}{d_{forget}} \right) \right), \quad (13)$$

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. We manually tune d_{forget} to mimic the information loss observed in the translation scheme.

C. 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 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 for the global planning component and the Cylinder planner for the local planning component. 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 $\left|S_{n,m}^{flat} - S_{n,m}^{t}\right|$ onto R^{flat}. In our experiments, the resolution of B is 0.5 meters. 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. Sub-goals are chosen to be 5 meters to 10 meters away from the robot.

IV. Experiments

We compare implementations of our three planning systems that use the D_i^O distance metric (described in section III-*A*) 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. Courses 1, 2, and 3 are depicted in Fig. 4, Fig. 5, and Fig. 6, respectively. The actual paths that the robot took are overlaid on a top-down occupancy grid map of the environment. All maps were generated independent 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 of 10 centimeters to 1 meter. Courses 2 and 3 are similar to Course 1, except that Course 2 adds an obstacle of 10 meter girth, and Course 3 contains two adjoining obstacles each 1 meter wide and 30 meters and 10 meters long, respectively.

A version of the hierarchical planner implemented to use the $D_i^{R^{flat}}$ distance metric was also tested on course 3. The rout taken by this system is depicted in Fig. 7.

V. 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, we would like 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 our algorithms.

We 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 (Fig. 8). Thus, the least

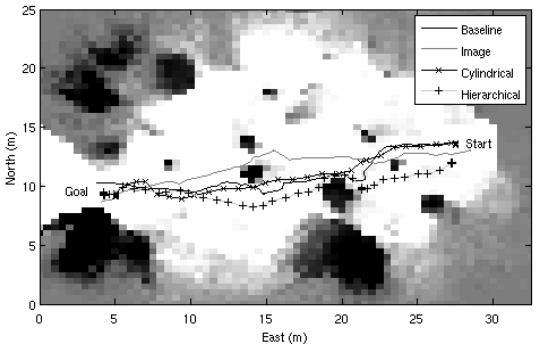

Fig. 4. Course 1: obstacles of small radii.

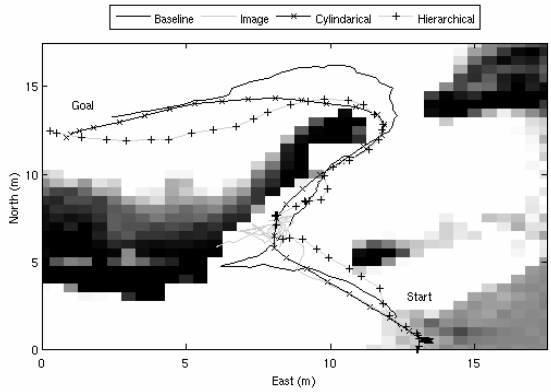

Fig. 5. Course 2: obstacle of 10 meter girth.

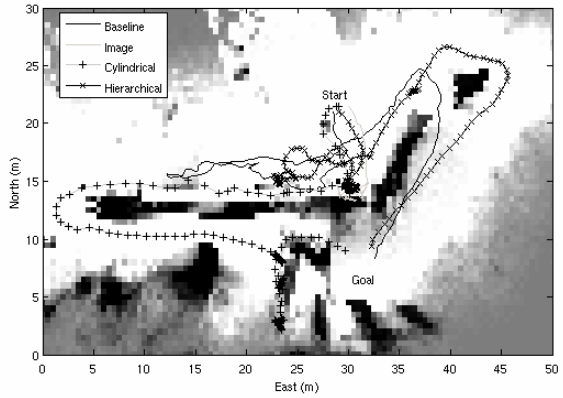

Fig. 6. Course 3: two adjoining long thin obstacles.

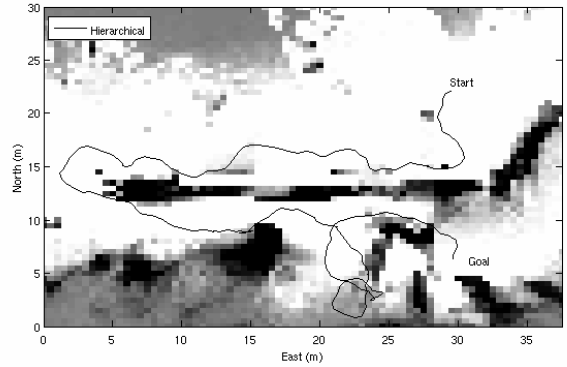

Fig. 7. $D_i^{R^{flat}}$ performance on Course 3.

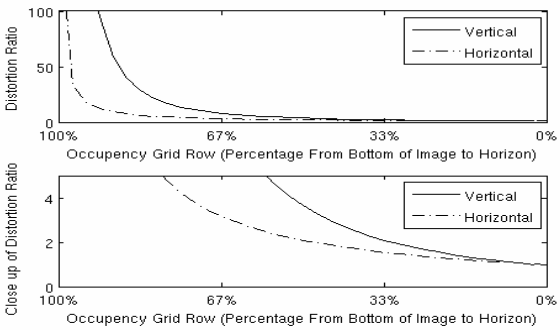

Fig. 8. Distortion Ratio as a function of occupancy grid row (percentage from bottom of image to horizon), where Distortion Ratio is $D_i^{R^{flat}}$ for vertical and horizontal neighbors divided by $D_i^{R^{flat}}$ for the bottom most vertical and horizontal neighbors, respectively (top). Note that this is proportional to $D_i^{R^{flat}} / D_i^{O}$. Close up of distortion ratio (bottom).

expensive path around an obstacle will 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. Fig. 6 and Fig. 7 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 Fig. 6 is much smoother than the one in Fig. 7.

D_i^{O} tends to distort R^{flat} distances, especially in the far field (Fig. 8). However, D_i^{O} works well in practice. By defining the distance between neighbors to be invariant of grid location, it avoids the near-field noise sensitivity observed with $D_i^{R^{flat}}$. This is because paths are penalized equally for near and far field detours, so the path is free to follow the geodesic around an obstacle or pseudo-obstacle without making an immediate correction. Also, because the range of our stereo sensors is effectively 12 meters, severe far-field distance distortion is somewhat irrelevant. Note that in Fig. 8 the horizontal distortion ratio is less than 2 for nearly half of the occupancy grid.

We found 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 a translational component and a rotational component. Without both components, the robot risks never finding a path to the goal.

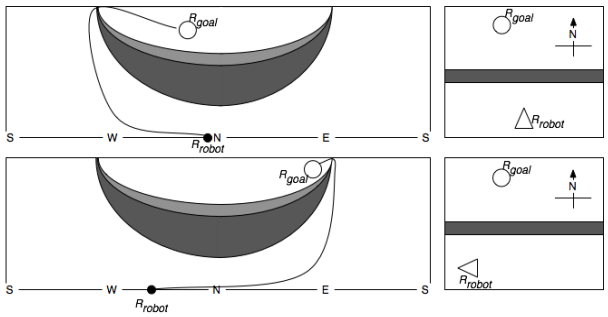

Fig. 9. Translational oscillation induced in the Cylindrical planner by a long thin wall. The initial path around the wall (top), and the path at a later time (bottom).

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. We observed the Image planner displaying this behavior on Courses 2 and 3, Fig. 5 and Fig. 6, 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 Fig. 5 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 (Fig. 6) 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 Fig. 9, 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 (Fig. 9 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.

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, if confronted with a large obstacle, will eventually find a way around it—if one exists. However, solutions can be suboptimal. For example, the robot may backtrack many times as it explores for a way around the wall [33]. This is observed in Fig. 6 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 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 systems like Navlab, 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.

VI. Conclusion and Future Work

We have demonstrated the efficacy of image based path planning. However, any robust path-planning algorithm must address two environmental scenarios: those that lead to rotational oscillation and those that lead to translational oscillation. Our 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. We find that, 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 high fidelity occupancy grid used in image space planning provides a natural framework to include more sophisticated models about the traversability of terrain. Color and texture models could be combined with stereo information to allow for more robust path planning. One unaddressed limitation of the Cylindrical planning system is its inability to plan behind obstacles; this is one of our current research focuses. The end goal of our efforts is a principled interaction between Cylindrical and Cartesian path planning. This paper proposed the first such successful framework, and sets the stage for future research efforts.

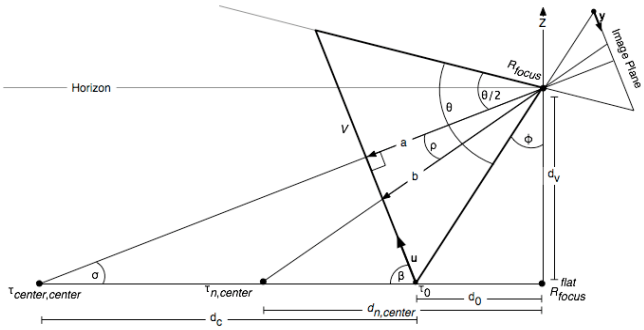

Fig. 10. FOV and accompanying variables used to calculate d and ψ.

VII. Appendix

A. Derivation of $d_{n,m}$

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 (Fig. 10). 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 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

$$\tau_0 = q_{center}\left(\mathrm{h} - \frac{\|\mathbf{y}\|}{2}\right) = g_{center}\left(\mathrm{h} - \frac{\|\mathbf{y}\|}{2}\right). \qquad (14)$$

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**

$$\rho = \tan^{-1}\left(\|\mathbf{u}\|\frac{(\mathrm{h}/2)-(\mathrm{h}-n)}{\|\mathbf{a}\|}\right) \qquad (15)$$

where h is the number of rows in the image plane and

$$\|\mathbf{a}\| = \frac{d_c + d_0}{\cos(\sigma)} - d_c\cos(\sigma). \qquad (16)$$

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}.

$$\sigma = \tan^{-1}\left(d_v/(d_c + d_0)\right) \qquad (17)$$

where d_v is the measured distance between R^{flat}_{focus} and R_{focus}.

We can now calculate $d_{n,w/2}$, the distance between R^{flat}_{focus} and $g_{center}(n)$.

$$d_{n,\mathrm{w}/2} = \|\mathbf{u}\|(h-n)\frac{\sin(\pi/2-\rho)}{\sin(\pi/2-\beta+\rho)} + d_0, \qquad (18)$$

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

$$\beta = \pi/2 - \sigma, \qquad (19)$$

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

$$\|\mathbf{u}\| = \frac{2d_c}{h}\sin(\sigma). \quad (20)$$

The inverse function to (18) is given by:

$$n = \left\lfloor h - \frac{(d_{n,w/2} - d_0)\sin\left(\tan^{-1}(d_v / d_{n,w/2})\right)}{\|\mathbf{u}\|\sin\left(\pi/2 - \tan^{-1}(d_v / d_{n,w/2}) + \sigma\right)} + \frac{1}{2} \right\rfloor. \quad (21)$$

If the image distortion caused by a rotation of $\theta/2$ can be ignored, e.g. θ is small such that $\tan(\theta/2) \approx \sin(\theta/2)$, then

$$d_{n,m} \approx d_{n,w/2}. \quad (22)$$

B. Derivation of ψ

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 in *O*, then given (22)

$$\psi = \frac{|m_2 - m_1|\theta}{w}, \quad (23)$$

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

ACKNOWLEDGMENT

We would like to thank Dan Lee for providing a top-down Cartesian global planner, and Adam Bates for robot support.

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.