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

**Michael W. Otte, Scott G. Richardson, Jane Mulligan, and Gregory Grudic**
*Department of Computer Science*
*University of Colorado at Boulder*
*Boulder, Colorado 80309-0430*
*e-mail: michael.otte@colorado.edu, scottric@gmail.com, jane.mulligan@colorado.edu, gregory.grudic@colorado.edu*

Received 13 April 2008; accepted 23 November 2008

Path planning systems using graph-search algorithms such as A* usually operate in uniform plan-view occupancy grids. However, the sensors used to construct these grids observe the environment in their own sample space based on sensor type and viewpoint. In this paper we present an *image space* technique for path planning in unknown unstructured outdoor environments. Our method differs from previous techniques in that we perform path search directly in image space—the native sensor space of the imaging sensor. After an image space path has been found, it is used for navigation in the real world. By operating at the resolution of the image sensor, image space planning facilitates accurate robot vs. obstacle localization and enables a high degree of movement precision. Our image space planning techniques can potentially be used with many different kinds of sensor data, and we experimentally evaluate the use of stereo disparity and color information. We present an extension to the basic image space planning system called the *cylindrical planner* that simulates a $2\pi$ field of view with a cylindrically shaped occupancy grid. We believe that image space planning is well suited for use in the local subsystem of a hierarchical planner and implement a *hybrid hierarchical planner* that utilizes the cylindrical planner as a local planning subsystem and a two-dimensional Cartesian planner as the global planning subsystem. All three systems are implemented and experimentally tested on a real robot. We evaluate the failure modes of image space planning and discuss how to avoid them. We find that image space enables precise real-time, near-field planning. © 2009 Wiley Periodicals, Inc.

## 1. INTRODUCTION

Autonomous robot navigation is the search for a series of actions that move a robot from an initial state to a goal state. Typically, an internal representation of the world is used to select actions from a movement set defined by the robot's kinematics. For autonomous navigation in unstructured environments, the world representation is often encoded in a map. Navigation is achieved by finding a map-based path between map-based start and goal locations, extrapolating that path to the real world, and then following it. Occupancy grid maps are discretized environmental representations in which each grid cell stores data about a particular piece of the real world (Elfes, 1989; Goldberg, Maimone, & Matthies, 2002; Herman, 1986; Kolski, Ferguson, Bellino, & Siegwart, 2006). Occupancy grids are traditionally uniform two-dimensional (2D) or three-dimensional (3D) Cartesian models containing traversability data (e.g., cost or obstacle probability). A path is found between the occupancy grid start and goal locations using a graph-search algorithm (Dijkstra, 1959; Ferguson & Stentz, 2006; Hart, Nilsson, & Raphael, 1968; Koenig & Likhachev, 2002; Krishnaswamy & Stentz, 1995; Stentz, 1995), a finite element potential field model (Khatib, 1986; Koren & Borenstein, 1991), or a combination of the two (Murray & Little, 2000). We focus primarily on the former paradigm, in which variants of A* and D* are commonly used.

In unknown environments, the occupancy grid is not known a priori and must be created as the robot observes the world with its sensors. Computer vision methods such as 3D stereo reconstruction can be used to obtain depth information (Matthies, 1992; Murray & Jennings, 1998; van den Mark, Groen, & van den Heuvel, 2001). Map building techniques that project stereo or laser depth data into a Cartesian model of the world depend on pose estimates that may be inaccurate in the short term. We investigate using the sensor image to guide robot motion without projection, particularly for real-time, near-field obstacle avoidance.

Planning in the space of image sensor samples, or *image space*, presents several challenges. Owing to a sensor's pose and projection, the sensor sample array does not represent a uniform grid in the world; i.e., near- and far-field samples arise from regions of differing size (several square centimeters to meters, respectively). We exploit occupancy grid planning methods in image space by using nonuniform graph-search edge costs. Once an image space path is computed, motion control is defined based on the first-person relative position of a near-field path waypoint. Because the image changes with the motion of the robot, the use of memory offers interesting trade-offs. To avoid oscillation as obstacles move in and out of the field of view (FOV), we present a method for accumulating image data in a cylindrical panorama as the robot moves. We call this type of system a *cylindrical planner*.

A *hierarchical planner* (as considered here) is a planning system that divides the path planning problem into the two parallel tasks of local planning and global planning. The global planner finds an approximate path to the goal based on a coarse world representation, whereas the local planner is concerned with obstacle avoidance and navigation toward subgoals (provided by the global planner) through a more precise world representation. A hierarchical planner allows the global map to expand as the robot encounters new parts of the environment, while facilitating time-bounded obstacle avoidance in the near field via the local map. A diverse variety of hierarchical planning systems have been proposed in the literature (Carsten, Rankin, Ferguson, & Stentz, 2007; Chen, 1990; Gat, Slack, Miller, & Firby, 1990; Hong, Tan, Pinette, Weiss, & Riseman, 1991; Krogh & Thorpe, 1986; Sugiyama et al., 1994, and many others). A common implementation uses two 2D Cartesian occupancy grids. The local planner maintains a fixed-size, high-resolution map that is centered on the robot, whereas the global planner accumulates a low-resolution map that expands with exploration and is anchored to a global frame of reference. We propose a *hybrid hierarchical planner* that performs local navigation in a cylindrical planner and global navigation in a 2D Cartesian occupancy grid. We believe that the strengths of image planning (discussed in Section 2.1) justify its use as a local planner, whereas Cartesian space is better suited to long-distance planning tasks requiring persistent memory. A hybrid hierarchical planner combines the strengths of both planning paradigms in a single system.

This paper is organized as follows: Section 2 contains an overview of image planing, a survey of related work, and a discussion on cost maps motivating our method. The basic image planner is introduced in Section 3, and the cylindrical and hybrid hierarchical planners are described in Sections 4 and 5, respectively. In Section 6 we outline our hardware, and in Section 7 we perform a series of experiments to evaluate the image planner, the cylindrical planner, and the hybrid hierarchical planner against a state-of-the-art 2D Cartesian hierarchical planer. Specific system parameter values are also stated in Section 7. Results are presented in Section 8 and conclusions in Section 9.

## 2. BACKGROUND

### 2.1. Image Space Path Planning

A *grid-based sensor* or *image sensor* captures multiple simultaneous observations about the world via a grid of sensor elements (Sonka, Hlavac, & Boyle, 2008). An *image* is the instantaneous output of an image

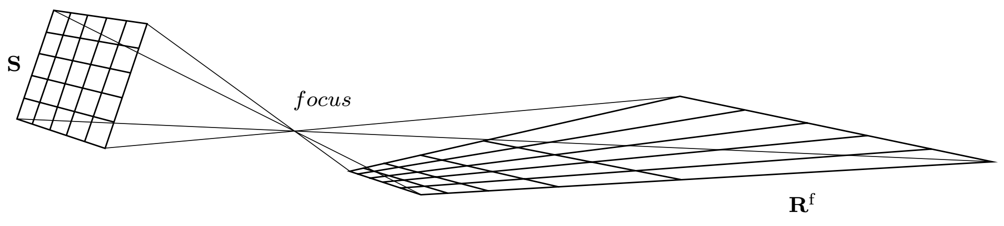


**Figure 1.** A projection of pixels from image space **S** onto a flat level groundplane $\mathbf{R}^f$.

sensor and can be represented as a matrix, the elements of which are called *pixels*. *Image space* is a coordinate space of the sensor grid with unit vectors defined by pixel height and width. In *image space planning* a path through an image is used for navigation. Digital cameras are commonly used for image space planning. We experiment with stereo disparity and RGB (red, green, blue) intensity values—both captured with cameras—and refer to image sensors as cameras. However, our general methods can also be used with other grid-based sensors (e.g., LIDAR, SONAR, infrared).

We restrict our discussion to scenarios in which a passive image sensor is mounted on an autonomous robot. We assume that spatial transformations are known between the sensor and robot coordinate frames, that the sensor grid observes the ground surface (or obscuring obstacles), and that the ground plane parameters ($\mathbf{n}$, $d$) are known relative to the calibrated camera frame. Only one real-world path can be described by projecting an image space path onto the real-world ground surface. We assume that connectivity along the image space path implies connectivity along that real-world path. Finally, we assume basic prior knowledge about traversability given a sensor type (e.g., stereo disparity or color) and also a known goal location on the ground surface.

Image space preserves the accuracy and precision of sampled scene information given an instantaneous sensor reading. Consequently, image space planning minimizes the number of map locations needed to accurately represent a scene projection captured by an image sensor. Map size is fixed by the properties of the sensor, which guarantees real-time obstacle avoidance by placing upper bounds on the time required for graph search. Owing to foreshortening and perspective projection, near-field pixels capture scene information at a higher resolution than far-field pixels, although both are represented at the finest granularity available (Figure 1). As a result, image space provides robust localization of near-field obstacles, enabling navigation through tight passages and cluttered environments.

If other information is available, in addition to the current image, then image space may not provide the most accurate world representation. For example, there is a point beyond which a grid-based Cartesian map will have a higher resolution than an image space map. This is because the granularity of a Cartesian map is fixed, relative to the real world, whereas the granularity of an image space map becomes coarser toward the horizon (Figure 1). In general, image space projections cannot represent what is behind view obstructions. This problem is initially encountered by any system that must discover the world. However, image space lacks the ability to remember previously explored areas that are currently outside the FOV. We discuss this problem, including solutions, in Section 3.4.4.

## 2.2. Related Work

The work most closely related to our own has been pursued by Ollis, Huang, Happold, and Stancil (2008). In this work, image space and Cartesian planners operate in parallel. If a low-cost, image-based path is found, then the image planner guides the default Cartesian planner by providing a waypoint from the image-based path. This technique is used in conjunction with cost from color data to find paths beyond the range of stereo disparity. In contrast, our approach focuses on high-resolution, near-field planning.

Autonomous highway driving algorithms rely on image features such as lane markings and road color/texture data to infer knowledge about a scene (Jochem, Pomerleau, & Thorpe, 1995; Mateus, Avina, & Devy, 2005; Pomerleau, 1989; Thorpe, Herbert, Kanade, & Shafer, 1988; Tsugawa, Yatabe, Hirose, & Matsumoto, 1979). A related off-road system assumes prior vehicle tire tracks or pedestrian footprints (Song, Lee, Yi, & Levandowski, 2007). However, these features are not guaranteed to exist, or be useful, in all outdoor environments.

Visual servoing methods use the appearance relationship between a camera's current FOV and a predefined target image to calculate steering commands (Cowan, Weingarten, & Koditschek, 2002; Feddema & Mitchell, 1989; Gaussier, Joulain, Blanquet, & Revel, 1997; Hutchinson, Hager, & Corke, 1996; Matsumoto, Sakai, Inaba, & Inoue, 2000; Vidal, Shakernia, & Sastry, 2003; Winters, Gaspar, Lacey, & Santos-Victor, 2000; Zhang & Ostrowski, 2002). Navigation through unknown terrain implies that we cannot obtain a predefined image of the goal state.

Reactive local planners often use polar models to determine navigational headings. Borenstein and Koren (1991) create a polar histogram from data accumulated in a global Cartesian map, and Minguez and Montano (2004) build a polar map from a ring of depth sensors. In either case, the local map is a one-dimensional (1D) vector and each element represents the cost of moving in a particular direction. These techniques are similar to our cylindrical planner in that they utilize polar coordinates. However, the additional dimension of the cylindrical planner enables it to formulate an entire path between the robot and goal.

Range data from stereo vision are frequently used in conjunction with Cartesian planning systems (Carsten et al., 2007; Goldberg et al., 2002; Murray & Little, 2000; Sabe et al., 2004; van den Mark et al., 2001). These approaches project data into top-down or 3D Cartesian representations of the world, instead of planning directly in image space.

## 2.3. Cost vs. Distance

Each cell or *map element* in an occupancy grid map represents a unique subset of the real world, or *world element*, and has a value associated with the cost of movement through the corresponding world element. Map elements are usually arranged in rows and columns of unit-length squares. This configuration can be stored as a 2D array $\mathbf{G}$, where $\mathbf{G}_{n,m}$ maintains the cost associated with the row $n$, column $m$ map element. If a graph structure is imposed on the map, then a path between two map locations can be found with a graph-search algorithm. The former is typically achieved by recasting map elements as graph nodes and then creating edges from each node to its neighbors. Common structures include the 4- and 8-connected graphs. Let the graph formalization of the cost map be denoted $\mathbf{G}^{\Gamma}$ to differentiate it from the map array $\mathbf{G}$. The graph node $\mathbf{G}^{\Gamma}_{n,m}$ is associated with the row $n$ and column $m$ map element. Graph search algorithms operate on edges, not nodes; therefore, a method of determining edge cost must be devised. This is often done by defining edge cost to be the map cost value $\mathbf{G}_{n,m}$ or $\mathbf{G}_{k,j}$ associated with the nodes $\mathbf{G}^{\Gamma}_{n,m}$ and $\mathbf{G}^{\Gamma}_{k,j}$ that an edge connects, or the average value of $\mathbf{G}_{n,m}$ and $\mathbf{G}_{k,j}$.

Consider a uniform world such that all elements in a 2D top-down Cartesian map have identical cost. Let the map graph be 8-connected. The Euclidean distance between neighboring world element centers is given, to a scale factor, by the distance between the corresponding map elements. The scale factor can be ignored without affecting search results, so we assume that it is 1. If edge cost is defined as the map cost $\mathbf{G}_{n,m}$ of the node $\mathbf{G}^{\Gamma}_{n,m}$ that the edge is directed to (or from, or their mean), then it costs 1 to move between neighboring nodes. An edge cost of 1 reflects the 1 Euclidean distance between horizontal and vertical neighbors but is inaccurate with respect to the $\sqrt{2}$ distance between diagonal neighbors. Edge cost should reflect both map cost and real-world distance. Therefore, we redefine edge cost to be *map cost integrated over the Euclidean distance between map elements*. In a discretized map this reduces to map cost multiplied by the distance through a map element.

The idea of integrating cost over distance is used in Ferguson and Stentz (2006) and elsewhere; however, we have found little discussion in the literature as to why this is a good idea. In fact, it is supported by a geometric argument and a physical analogy. *Geometric*: Let point cost be a measure of traversability associated with a point. Let the integration of point cost over a grid element's area yield that grid element's map cost. Grid elements are assumed to be unit-length squares of uniform point cost; thus, point cost is equal to the map cost of the grid element containing a particular point. Edges are defined as line segments spanning neighboring grid element centers, and 1D line cost is found by integrating point cost over a segment's length. Thus, edge cost is map cost integrated over Euclidean distance. *Physical*: Assume that map cost is a force opposing movement within a map element. The most efficient way through a map is along the path of minimum work. Work is force integrated over distance; therefore, the most efficient path is found by minimizing map cost integrated over Euclidean distance.

To disambiguate between map cost (i.e., point cost, force) and edge cost (i.e., line cost, work), we employ the terms used in the physical analogy: we refer to map cost as force and edge cost as work. The concepts of force, distance, and work are integral to our image space planning system. Owing to perspective, image space world elements are not square and vary in size from the near to far field. We hypothesize that image space paths will be more useful for navigation if the distance quantity reflects real-world spatial relationships.

## 2.4. Memory and Myopic Behavior

Let *rotational memory* and *translational memory* denote, respectively, an ability to remember map data after an arbitrarily large rotation or translation. Local Cartesian maps do not have translational memory because data are forgotten after a translation beyond the map's fixed size. They do have rotational memory because the robot can pivot without losing information. Global Cartesian maps have both rotational and translational memory. In our earlier work, we observe that lack of translational or rotational memory can induce myopic behavior in the form of *translational* or *rotational oscillation*, as the robot repeatedly moves away from, forgets, and then returns to a goal blocking obstacle (Otte, 2007; Otte, Richardson, Mulligan, & Grudic, 2007). Myopic behavior is not the only cause of oscillation; e.g., Carsten et al. (2007) observe oscillation from conflicting votes in a vote-based planner.

## 3. THE BASIC IMAGE PLANNER

We call our basic image space planner the *image planner*. It creates an image space occupancy grid from pixel data and makes no assumptions additional to those in Section 2.1.

## 3.1. Occupancy Grids and Cost Functions in Image Space

Let $\mathbf{S}$ be the $h$ row and $w$ column output of an image sensor. Pixels are denoted $\mathbf{S}_{n,m}$, where $1 \leq n \leq h$ and $1 \leq m \leq w$, and $n = 1$ and $m = 1$ correspond to the top row and left column, respectively. $\mathbf{O}$ is the image space occupancy grid built from $\mathbf{S}$. Map values are determined as a function of the corresponding image pixel data, and in the most basic implementation there is a one-to-one and onto mapping from pixels to grid locations:

$$\mathbf{O}_{n,m} = f(\mathbf{S}_{n,m}).$$

Let $\mathbf{O}^\Gamma$ be the 8-connected graph of $\mathbf{O}$. Node $\mathbf{O}^\Gamma_{n,m}$ is associated with map element $\mathbf{O}_{n,m}$ and is a neighbor of $\mathbf{O}^\Gamma_{k,j}$ for all $k = n \pm 1$ and $j = m \pm 1$, where $1 \leq k \leq h$ and $1 \leq j \leq w$. Let arc $i$ connect $\mathbf{O}^\Gamma_{k,j}$ to $\mathbf{O}^\Gamma_{n,m}$. The length of $i$ is $D_i$. We interpret $\mathbf{O}_{n,m}$ as a force $F_i$ impeding progress along $i$ and define the edge cost of $i$ as work $W_i$:

$$W_i = F_i D_i.$$

Let $\mathbf{R}$ denote the real world. To move from the current world location $\mathbf{R_s}$ to a goal location $\mathbf{R_g}$, the system transforms $\mathbf{R_s}$ and $\mathbf{R_g}$ into their image space graph equivalents $\mathbf{O}^\Gamma_\mathbf{s}$ and $\mathbf{O}^\Gamma_\mathbf{g}$, respectively, and then uses a variant of the A* algorithm to find the minimum work path $P^{\min}_\mathbf{O}$ between $\mathbf{O}^\Gamma_\mathbf{s}$ and $\mathbf{O}^\Gamma_\mathbf{g}$:

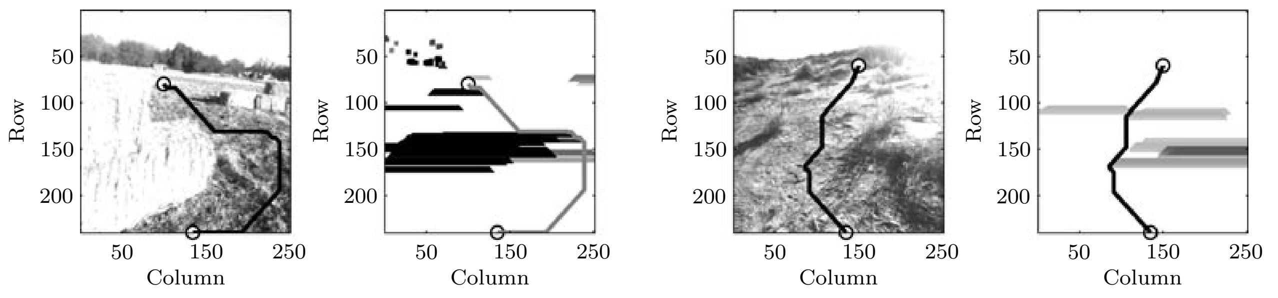
$$W_{P^{\min}_\mathbf{O}} = \min_{P_\mathbf{O}} \left( \sum_{i \in P_\mathbf{O}} F_i D_i \right).$$

The optimality of A* is ensured only if $f(\mathbf{S}_{n,m})$ returns nonnegative values over the entire range of $\mathbf{S}_{n,m}$ allowed by the sensor; thus, $f(\mathbf{S}_{n,m})$ is system dependent. Figure 2 shows two examples of $P^{\min}_\mathbf{O}$ projected into $\mathbf{O}$ and the corresponding grayscale images.

$\mathbf{O}^\Gamma_\mathbf{s}$ represents the robot's projected location in the image $\mathbf{S}$ and depends on the relationship between the robot and the camera. If the camera is forward facing and fixed in the center of the robot, then $\mathbf{O}^\Gamma_\mathbf{s}$ is the bottom center graph node $\mathbf{O}^\Gamma_{h,\lfloor w/2 \rfloor}$. The method used to position $\mathbf{O}^\Gamma_\mathbf{g}$ is task dependent. A system designed to seek a specific color/pattern target may extract the coordinates of $\mathbf{O}^\Gamma_\mathbf{g}$ directly from an RGB image. Goals defined by global positioning system (GPS) coordinates must be projected into image space, implying the existence of a transformation function. The robot is programmed to rotate toward $\mathbf{R_g}$ whenever $\mathbf{R_g}$ is outside the FOV.

## 3.2. Force Metrics

The map value $\mathbf{O}_{n,m}$ is a function of pixel data $\mathbf{S}_{n,m}$ and determines the force $F_i$ required to move along



(a) Hay bales on field (b) Shrubs in a desert

**Figure 2.** Image space paths through grayscale images and corresponding occupancy grids.

arc $i$ toward node $\mathbf{O}^{\Gamma}_{n,m}$:

$$F_i = \mathbf{O}_{n,m} = f(\mathbf{S}_{n,m}). \quad (1)$$

Equation (1) is a force metric defining the relationship between sensor data and force in $\mathbf{O}^{\Gamma}$. To be useful, $f(\mathbf{S}_{n,m})$ should be representative of real-world traversability. Obstacles should have high (or lethal) values relative to flat terrain. We experiment with two force metrics. The first metric $F^{\mathrm{d}}_i$ assumes that pixel $\mathbf{S}_{n,m}$ contains disparity information. The second metric $F^{\mathrm{c}}_i$ assumes that $\mathbf{S}_{n,m}$ is a triple of RGB intensity values.

### 3.2.1. Force from Stereo Disparity

Given two images of a scene from different viewpoints, stereo disparity is the difference between the image locations to which a point is projected. Given rectified images and cameras of equal focal length, disparity $\mathbf{S}_{n,m}$ is related to depth $z(\mathbf{S}_{n,m})$ by

$$z(\mathbf{S}_{n,m}) = \frac{c_{\mathrm{f}} c_{\mathrm{b}}}{\mathbf{S}_{n,m}},$$

where $c_{\mathrm{f}}$ and $c_{\mathrm{b}}$ are focal length and the baseline between optical centers, respectively (Sonka et al., 2008). Disparity approaches zero as depth approaches infinity and vice-versa. However, given a finite camera resolution, disparity may go to zero at less than 30 m for practical $w$, $h$, $c_{\mathrm{b}}$, and $c_{\mathrm{f}}$. We define a force metric from disparity as follows:

$$F^{\mathrm{d}}_i = \mathbf{O}_{n,m} = 1 + c_{\mathrm{scl}} \left| \mathbf{S}_{n,m} - \mathbf{S}^{\mathrm{f}}_{n,m} \right|, \quad (2)$$

where $\mathbf{S}^{\mathrm{f}}_{n,m}$ is the disparity observed from a known flat level ground plane $\mathbf{R}^{\mathrm{f}}$ and $c_{\mathrm{scl}}$ is a scaling constant. $c_{\mathrm{scl}}$ is tuned so that $F^{\mathrm{d}}_i$ will be greater than a threshold $c_{\mathrm{thd}}$ for obstacles within stereo range. A concept of lethality is added by explicitly resetting $F^{\mathrm{d}}_i$ to $10^8 c_{\mathrm{thd}}$ if $F^{\mathrm{d}}_i$ is greater than $1.5c_{\mathrm{thd}}$. Distance may be used as an admissible heuristic for estimating work during graph search because $F^{\mathrm{d}}_i \geq 1$.

### 3.2.2. Force from Color

Our color data are composed of three dimensions, red $\mathbf{S}^{\mathrm{red}}_{n,m}$, green $\mathbf{S}^{\mathrm{grn}}_{n,m}$, and blue $\mathbf{S}^{\mathrm{blu}}_{n,m}$. Values range between 0 and 255. In practice, a force from color metric can be provided by a machine-learning subsystem that is trained online (Grudic, Mulligan, Otte, & Bates, 2007; Ollis et al., 2008; Procopio, Mulligan, & Grudic, 2007). We implement a predefined metric $F^{\mathrm{c}}_i$ as a proof of concept. Any predefined force from color metric is valid only in a subset of environments, and the simple technique described here is even brittle in a single environment. Hand-labeled training examples, one each for traversable terrain and obstacles, are provided to the system prior to a run. The color dimension of largest magnitude is denoted gnd or obs for the traversable or obstacle example, respectively, gnd, obs ∈ {red, grn, blu}. This method of determining an example's associated color dimension is simple and works in practice, but it may not be superior to any other. $\hat{\mathbf{S}}$ is a normalized version of $\mathbf{S}$ with values between 0 and 1. $\hat{\mathbf{S}}^{\mathrm{obs}}_{n,m}$ and $\hat{\mathbf{S}}^{\mathrm{gnd}}_{n,m}$ are normalized intensity values along the obs and gnd dimensions, respectively:

$$F^{\mathrm{c}}_i = \mathbf{O}_{n,m} = 1 + c_{\mathrm{scl}}\left(1 + \hat{\mathbf{S}}^{\mathrm{obs}}_{n,m} - \hat{\mathbf{S}}^{\mathrm{gnd}}_{n,m}\right). \quad (3)$$

$F_i^c$ is scaled between 1 and $c_{\text{thd}}$ using $c_{\text{scl}}$ and defined as lethal if greater than $c_{\text{thd}}/3$. Distance may be used as an admissible heuristic for estimating work during graph search because $F_i^c \geq 1$. $F_i^c$ fails if obs = gnd and is valid only in environments where obstacles and traversable terrain differ in color. It is worth reiterating that $F_i^c$ is used as a proof of concept. Our goal is to show that the image planner can navigate using color data.

## 3.3. Distance Metrics

$D_i$ is defined to be the length of arc $i$ from node $\mathbf{O}_{k,j}^{\Gamma}$ to node $\mathbf{O}_{n,m}^{\Gamma}$ in the graph $\mathbf{O}^{\Gamma}$. In general, $D_i$ is a function of grid coordinates:

$$D_i = f(k, j, n, m).$$

$D_i$ must be nonnegative for the optimality of A* and nonzero to ensure that A* is useful. (A* ignores all force information and degenerates into an uninformed breadth-first search when $D_i = 0$.) We now discuss two distance metrics used in our experiments.

### 3.3.1. The $L^2$ Norm in Image Space

In 2D top-down Cartesian systems, the $L^2$ norm in map space is a constant multiple of the $L^2$ norm in the real world (assuming elevation can be ignored), and it makes sense to define arc length as the Euclidean distance between grid element centers. We experiment using the $L^2$ norm of image space as a distance metric $D_i^{\mathbf{S}}$:

$$D_i^{\mathbf{S}} = \sqrt{(n-k)^2 + (m-j)^2}. \tag{4}$$

$D_i^{\mathbf{S}}$ is the distance between $\mathbf{O}_{k,j}$ and $\mathbf{O}_{n,m}$, assuming that grid element side lengths are 1. The distance between horizontal, vertical, and diagonal neighbors is 1, 1, and $\sqrt{2}$, respectively. $D_i^{\mathbf{S}}$ does *not* accurately model the distances between image space world elements—which increase toward the horizon.

### 3.3.2. The Flat World Distance Metric

We define the flat world distance metric $D_i^{\mathbf{R}^f}$ to be the Euclidean distance between the two points $\mathbf{x}_1$ and $\mathbf{x}_2$ on a flat ground plane $\mathbf{R}^f$ that are projected to the centers of pixels $\mathbf{S}_{k,j}$ and $\mathbf{S}_{n,m}$, respectively (Figure 3). To calculate the positions of $\mathbf{x}_1$ and $\mathbf{x}_2$, we find the intersection of $\mathbf{R}^f$ and the rays extending from pixel centers through the camera's optical center. These are nominal world grid distances, assuming that traversable regions of the scene lie on $\mathbf{R}^f$. Tracing pixel rays to the ground plane reverses the projection process by following the line of sight from the pixel. Given the known camera intrinsic matrix $\mathbf{K}$, we convert from pixel coordinates to normalized coordinates $\mathbf{R_d}$:

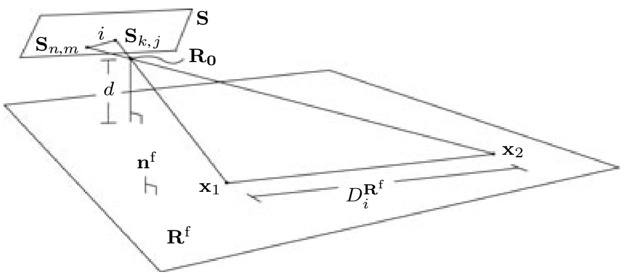


**Figure 3.** A 3D view of the relationship between image space arc $i$ and the distance $D_i^{\mathbf{R}^f}$.

$$\mathbf{R_d} = \mathbf{K}^{-1}[m, n, 1]^{\mathrm{T}}.$$

The rays emanate from the camera center $\mathbf{R_0}$, which is the origin of the camera coordinate frame. We assume that the ground plane parameters $\mathbf{R}^f = (\mathbf{n}^f, d)$ are known in the camera coordinate frame. $\mathbf{n}^f = [n_x^f, n_y^f, n_z^f]^{\mathrm{T}}$ is the plane normal, and $d$ is the perpendicular distance from the origin to the plane. $n_x^f x + n_y^f y + n_z^f z = d$ for points on the plane described by the vector $\mathbf{x} = [x, y, z]^{\mathrm{T}}$. To find the intersection such that $\mathbf{R_0} + t\mathbf{R_d}$ lies on the plane, we need to solve $\mathbf{n}^f \cdot (\mathbf{R_0} + t\mathbf{R_d}) - d = 0$ for $t$. Recall that $\mathbf{R_0} = [0, 0, 0]^{\mathrm{T}}$ is the origin:

$$t = \frac{\mathbf{n}^f \cdot \mathbf{R_0} + d}{\mathbf{n}^f \cdot \mathbf{R_d}} = \frac{d}{\mathbf{n}^f \cdot \mathbf{R_d}}.$$

We can compute the desired intersection points using $\mathbf{x} = t\mathbf{R_d}$. Finally, $D_i^{\mathbf{R}^f}$ is computed as the Euclidean distance between $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$D_i^{\mathbf{R}^f} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}. \tag{5}$$

$D_i^{\mathbf{R}^f}$ defines arc lengths in $\mathbf{O}^{\Gamma}$ as the flat level ground-surface distances they represent. Although the ground surface in unstructured outdoor environments may not be flat or level, we hypothesize

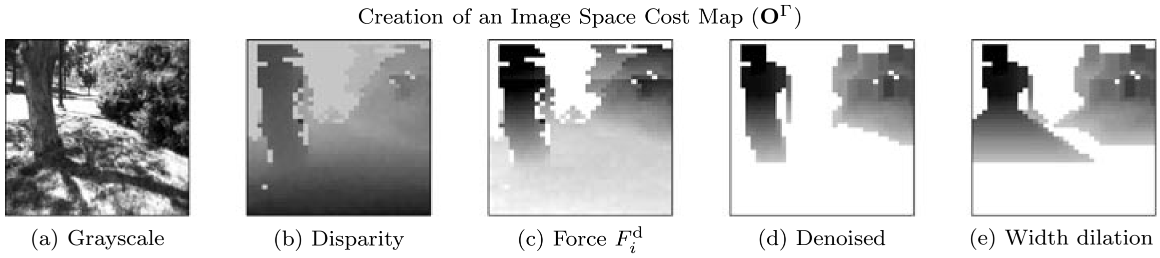

Creation of an Image Space Cost Map ($\mathbf{O}^{\Gamma}$)
(a) Grayscale
(b) Disparity
(c) Force $F_i^{\mathrm{d}}$
(d) Denoised
(e) Width dilation

**Figure 4.** A grayscale scene image, the corresponding stereo disparity, and various stages of the $F_i^{\mathrm{d}}$ cost map. Light to dark corresponds to high to low disparity and cost, respectively.

that $D_i^{\mathbf{R}^{\mathrm{f}}}$ more accurately models the world than $D_i^{\mathbf{S}}$. Movement through $\mathbf{O}^{\Gamma}$ becomes easier toward the bottom of the graph because $D_i^{\mathbf{R}^{\mathrm{f}}}$ increases toward the horizon line. $D_i^{\mathbf{R}^{\mathrm{f}}}$ also reflects the fact that image space world elements become increasingly elongated toward the horizon (see Figure 1).

## 3.4. Preprocessing

### 3.4.1. Noise Reduction

Small regions of noise are observed with both stereo and color data. We erode and then dilate $\mathbf{O}$ to remove errors introduced by this noise. The same mask is used for both operations. To eliminate small variations in nearly flat terrain, we reset map values less than a threshold $c_{\mathrm{flt}}$ to 1. Figures 4(c) and 4(d) show $\mathbf{O}$ before and after noise reduction, respectively.

### 3.4.2. Obstacle Width Dilation

The A* search algorithm assumes the robot is a point particle. As suggested by Kelly (1994), Kolski et al. (2006), and Sugiyama et al. (1994), we increase obstacle width in the occupancy grid by performing a dilation of radius $\alpha$, where $\alpha$ is the map space equivalent of one half robot width $c_{\mathrm{wth}}/2$ plus a buffer $c_{\mathrm{buf}}$. This ensures that paths through $\mathbf{O}^{\Gamma}$ will cause the robot to avoid obstacles by at least $c_{\mathrm{buf}}$. The dilation equation is given by

$$\mathbf{O}_{n,m} = \max_{j} (\mathbf{O}_{n,m\pm j}),$$

where $\lfloor -\alpha \rfloor \leq j \leq \lceil \alpha \rceil$ and $1 \leq (m \pm j) \leq w$. The apparent robot width in the image $\mathbf{S}$, and therefore $\mathbf{O}$, varies due to perspective. Our dilation function assumes a flat level plane $\mathbf{R}^{\mathrm{f}}$ and approximates each pixel as an equal portion of $\theta_w$, where $\theta_w$ is the horizontal FOV. Let $D_{0\to n,m}^{\mathbf{R}^{\mathrm{f}}}$ be the distance along $\mathbf{R}^{\mathrm{f}}$ from the robot to the projected center of $\mathbf{O}_{n,m}$. Equation (5) can be used to calculate $D_{0\to n,m}^{\mathbf{R}^{\mathrm{f}}}$ by letting $\mathbf{x}_1 = [0, 0, -d]^{\mathrm{T}}$. $\alpha$ is a function of $D_{0\to n,m}^{\mathbf{R}^{\mathrm{f}}}$:

$$\alpha = \frac{w}{\theta_w} \arcsin\left( \frac{c_{\mathrm{wth}}/2 + c_{\mathrm{buf}}}{D_{0\to n,m}^{\mathbf{R}^{\mathrm{f}}}} \right). \tag{6}$$

$\alpha$ can be calculated offline. The flat level plane assumption provides a useful approximation to reality. An alternative is to perform dilation as a function of depth, assuming that depth data exist (an idea not explored in this paper). Figures 4(d) and 4(e) show $\mathbf{O}$ before and after robot width dilation, respectively.

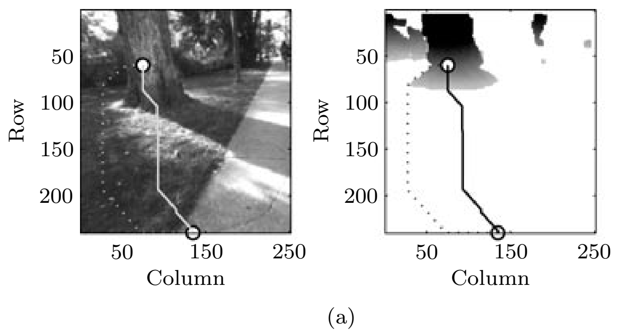
### 3.4.3. Enabling Rotation

If $\mathbf{S}$ does not capture the area immediately adjacent to the robot, then obstacles in the bottom grid row $\mathbf{O}_{h,1\ldots w}$ can prevent rotation, despite the existence of maneuvering space. We explicitly allow rotation by defining the bottom row to have the minimum force value 1:

$$\mathbf{O}_{h,1\ldots w} = 1.$$

### 3.4.4. Accounting for Finite Obstacle Depth

If the goal is a coordinate in $\mathbf{R}$, then a problem can arise when it is obscured by an obstacle. The goal will appear inside the obstacle in $\mathbf{O}^{\Gamma}$, requiring the path to penetrate the obstacle. The cheapest path may go "up" the obstacle in $\mathbf{O}^{\Gamma}$, leading the robot to collide with the obstacle in $\mathbf{R}$ [see Figure 5(a)]. Obstacles defined as infinitely lethal will cause the goal to



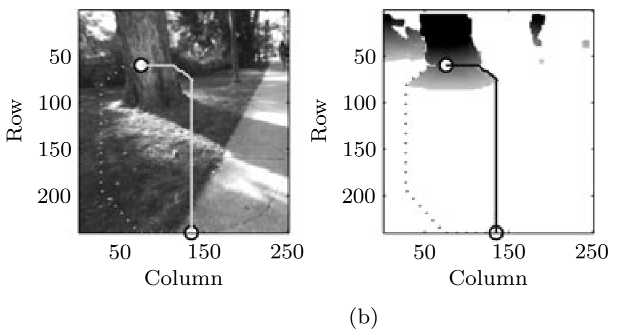


**Figure 5.** Image space paths based on the $D_i^{\mathbf{S}}$ and $D_i^{\mathbf{R}^f}$ distance metrics (solid and dotted lines, respectively), through grayscale scene images and the corresponding occupancy grids. (a) It requires too much work for $D_i^{\mathbf{S}}$ to go around the tree because the planner has no notion of going behind obstacles; this results in failure. (b) The occupancy grid has been preprocessed by setting force in the goal row equal to the minimum force value. This creates an artificial conception of obstacle depth that enables both paths to go around the tree.

be unreachable. Ollis et al. (2008) have addressed this problem by allowing cheap horizontal movement in the goal row of the image space map. This encourages paths to go around the obstacle before entering horizontally, causing the robot to travel around the obstacle as it moves. We modify our force values in a similar fashion. Let $\mathbf{O}^{\Gamma}_{n_g,m_g} = \mathbf{O}^{\Gamma}_{\mathbf{g}}$. The goal row $n_g$ of the map $\mathbf{O}$ is defined to have minimum force value 1:

$$\mathbf{O}_{n_g,1\ldots w} = 1.$$

Figures 5(a) and 5(b) display image space paths through identical occupancy grids, except that the latter has been preprocessed to account for obstacle depth.

## 3.5. Navigation

We control the robot based on the relative graph locations of $\mathbf{O}^{\Gamma}_{\mathbf{s}}$ and a target node $\mathbf{O}^{\Gamma}_{n_{\text{tgt}},m_{\text{tgt}}} = \mathbf{O}^{\Gamma}_{\mathbf{tgt}}$ located a predetermined distance $c_{\text{tgt}}$ along $P^{\min}_{\mathbf{O}}$. We find this to be simple and functional; however, Cartesian-based control can alternatively be used by projecting $\mathbf{O}^{\Gamma}_{\mathbf{tgt}}$ into robot coordinates. Assuming a center-mounted, forward-facing camera, $\mathbf{O}^{\Gamma}_{\mathbf{s}} = \mathbf{O}^{\Gamma}_{h,w/2}$. In this case, we define forward velocity as a function of the image-space cosine between the column $w/2$ and the vector from $\mathbf{O_s}$ to $\mathbf{O_{tgt}}$:

$$\text{speed} = \frac{\text{speed}_{\max}(h - n_{\text{tgt}})}{\sqrt{(w/2 - m_{\text{tgt}})^2 + (h - n_{\text{tgt}})^2}},$$

where speed$_{\max}$ is the maximum forward velocity allowed by the robot. Speed increases as $\mathbf{O_{tgt}}$ approaches the center of the FOV. In other words, the robot slows during turning maneuvers, which are likely to occur near obstacles. For turns in safe terrain, we increase the speed of the robot by resetting speed = speed$_{\max}$ if all force values in the bottom half of the center column $\lfloor w/2 \rfloor$ are less than $c_{\text{thd}}/3$ after map dilation. Turning angle is defined as the relative horizontal displacement between $\mathbf{O_s}$ and $\mathbf{O_{tgt}}$, normalized by the FOV $\theta_w$:

$$\text{turn} = \frac{\theta_w(m_{\text{tgt}} - w/2)}{w}.$$

This causes the robot to turn proportionally as the target approaches the edge of the FOV.

## 4. THE CYLINDRICAL IMAGE PLANNER

A shortcoming of the image planner is that the goal can exist outside the FOV. One solution is to use a sensor with a $2\pi$ radian FOV. In the absence of an omnicam or other panoramic sensor, we can map images to a synthetic panorama over time, based on direction of capture. We call a planner that uses a $2\pi$ radian panoramic image-space map a *cylindrical planner*. In addition to the assumptions outlined in Section 2.1, our cylindrical planner assumes the existence of global pose information, including orientation. This is necessary to populate and update the cylindrical panorama. One challenge with temporally accumulating panoramas is that information outside the camera's FOV becomes outdated with movement. In Section 4.2. we describe two techniques for updating data to reflect the robot's movement.

## 4.1. Panoramic Considerations

The cylindrical planner uses a cylindrical occupancy grid **C**. **C** is similar to **O** but contains additional columns to remember information outside the current FOV. Each column in **C** represents a specific compass direction, and data are inserted based on the camera's orientation at the time of capture. **C** is a radial panorama of what the robot has experienced. Radial panoramas and other forms of mosaics have previously been used for landmark detection and pose estimation (Argyros, Bekris, Orphanoudakis, & Kavraki, 2005; Chen, 1990; Kelly, 1994, 2000). Let $\varphi$ represent rotation about the vertical axis (yaw) relative to the compass direction north. Information is placed into **C** as a function of $\varphi$:

$$\mathbf{C}_{n,m+f(\varphi)} = f(\mathbf{S}_{n,m}).$$

Recall that $f(\mathbf{S}_{n,m})$ is a force metric (Section 3.2). Let $w_\mathrm{P}$ be the number of columns of **C**. We associate columns 1, $\lfloor w_\mathrm{P}/4 \rfloor$, $\lfloor w_\mathrm{P}/2 \rfloor$, $\lfloor 3w_\mathrm{P}/4 \rfloor$, and $w_\mathrm{P}$ with the cardinal directions south, west, north, east, and south, respectively (see Figure 6). $f(\varphi)$, the mapping from yaw angles to map columns, is given by

$$f(\varphi) = \left\lfloor \frac{w_\mathrm{P}\,(\varphi - \pi)}{2\pi} \bmod w_\mathrm{P} \right\rfloor. \tag{7}$$

Our robot has two stereo camera rigs that are angled toward the ground. For a virtual cylinder with radius

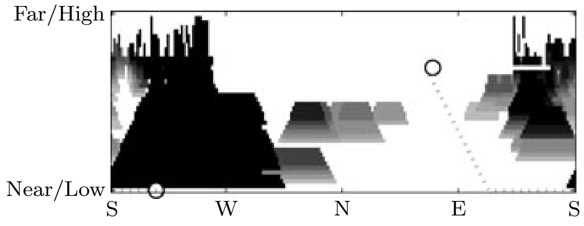


**Figure 6.** Image space path through the cylindrical occupancy grid. The robot is facing southwest, and the goal is east of the robot. It is easier for the robot to turn counterclockwise, which takes the image space path across the south–south border.

$r$ aligned with the vertical ($Z$) axis, we can identify the intersections of camera pixel rays with the cylinder. For camera position $\mathbf{p} = [p_x, p_y, p_z]^T$ and pixel ray $\mathbf{v} = [v_x, v_y, v_z]$ in the robot coordinate frame we want

$$p_x + kv_x = r\cos\theta,$$
$$p_y + kv_y = r\sin\theta,$$

for scale $k$. We square these expressions and add to yield

$$\left(p_x^2 + p_y^2\right) + 2k(p_x v_x + p_y v_y) + k^2\left(v_x^2 + v_y^2\right) = r^2.$$

This is a quadratic equation that can be solved for $k$ in the standard way:

$$a = \left(v_x^2 + v_y^2\right),$$
$$b = 2(p_x v_x + p_y v_y),$$
$$c = \left(p_x^2 + p_y^2\right) - r^2,$$
$$k = \frac{-b + \sqrt{b^2 - 4ac}}{2a}.$$

We now determine where each image-pixel ray intersects the cylinder using $\mathbf{p} + k\mathbf{v}$. This allows us to identify a range of height and angular values that bound the elements of the cylindrical image. Constructing a tessellation on the cylinder, we project these cylinder image points to the camera image to yield a mapping between the camera and cylinder images. Values are inserted into the cylinder using this mapping and Eq. (7)

Given depth data, another approach is to define a cylinder image origin on the yaw axis and then reproject reconstructed depth points $[X_w, Y_w, Z_w]$ to this coordinate frame $[X_c, Y_c, Z_c]$:

$$[X_c, Y_c, Z_c] = \frac{1}{\sqrt{X_w^2 + Y_w^2}}[X_w, Y_w, Z_w].$$

Maintaining the 3D locations of points from which cylinder values arise simplifies techniques for propagating and updating cost information as the robot moves.

Let $\mathbf{C}^\Gamma$ be the 8-connected graph of the occupancy grid **C**. We add arcs between the first and last columns of the occupancy grid graph so that paths can travel across the south–south border of $\mathbf{C}^\Gamma$. Arcs exist between $\mathbf{C}^\Gamma_{n,1}$ and $\mathbf{C}^\Gamma_{k,w_\mathrm{p}}$ for all $k = \{n-1, n, n+1\}$ and all $n = \{k-1, k, k+1\}$, subject to $1 \le n \le h$ and $1 \le k \le h$. The goal node $\mathbf{C}^\Gamma_\mathbf{g}$ is determined in a similar manner to $\mathbf{O}^\Gamma_\mathbf{g}$, unless the goal is too close to be projected into the cylinder—in which case we define the goal as achieved. The start node $\mathbf{C}^\Gamma_\mathbf{s} = \mathbf{C}^\Gamma_{h,f(\varphi)}$ is a function of yaw. Figure 6 shows a path from $\mathbf{C}^\Gamma_\mathbf{s}$ to $\mathbf{C}^\Gamma_\mathbf{g}$, projected into the occupancy grid $\mathbf{C}$ used to create $\mathbf{C}^\Gamma$ (the path traverses the south–south border).

As with $\mathbf{O}$, $\mathbf{C}$ is preprocessed to remove noise and improve path quality. Obstacle width dilation is modified to extend across the south–south border:

$$\mathbf{C}_{n,m} = \max_j (\mathbf{C}_{n,j}),$$

where $j$ is defined as follows:

$$\begin{array}{lll} \left.\begin{array}{l} 1 \le j \le m+\alpha \\ m-\alpha+w_\mathrm{p} \le j \le w_\mathrm{p} \end{array}\right\} & \text{if } m \le \alpha, & \\ m-\alpha \le j \le m+\alpha & \text{if } \alpha < m \le w_\mathrm{p}-\alpha, & (8) \\ \left.\begin{array}{l} 1 \le j \le m+\alpha-w_\mathrm{p} \\ m-\alpha \le j \le w_\mathrm{p} \end{array}\right\} & \text{if } w_\mathrm{p}-\alpha < m. & \end{array}$$

$\alpha$ is defined in Eq. (6). The denoising operations are similarly modified. Rotation is explicitly allowed by resetting values in the bottom row of $\mathbf{C}$ to the minimum force:

$$\mathbf{C}_{h,1\ldots w} = 1.$$

Similarly, finite obstacle depth is accounted for by resetting goal row values to the minimum force if they are within $\lfloor w_\mathrm{p}/4 \rfloor$ columns of $m_\mathrm{g}$:

$$\mathbf{C}_{n_\mathrm{g},j} = 1,$$

where $j$ is calculated by substituting $\alpha = \lfloor w_\mathrm{p}/4 \rfloor$ and $m = m_\mathrm{g}$ into Eq. (8).

Care must be taken to satisfy the constraints of the A* algorithm. If distance is defined by $D^\mathbf{S}_i$, the $L^2$ norm in image space is no longer an admissible heuristic for estimating work because it neglects the possibility of crossing the south–south border. Instead, we use a similar quantity $L^2_C$ that replaces the horizontal component of the $L^2$ norm with the shorter of the two horizontal candidates. The $L^2_C$ norm between $\mathbf{C}_{n,m}$ and $\mathbf{C}_{k,j}$ is calculated:

$$L^2_C = \sqrt{(n-k)^2 + \min\left[(m-j)^2, (w_\mathrm{p}-m+j)^2\right]}.$$

$D^{\mathbf{R}^\mathrm{f}}_i$ can be used as an admissible heuristic for estimating work because it is found by computing pixel ray intersections with the ground plane model and then computing the Euclidean distances between these points (Section 3.3.2). For the virtual cylinder camera, we compute the ground plane intersections for the cylinder image pixels directly.

## 4.2. Updating Techniques

### 4.2.1. Translation-Based Forgetting

Information in $\mathbf{C}$ is captured when the robot is at a specific location and facing in a particular direction. As the robot moves from that location, the data in $\mathbf{C}$ will cease to be an accurate representation of the associated direction. We propose forgetting data in $\mathbf{C}$ as a function of translation and call this technique *translation-based forgetting*. We assume that changes in elevation can be ignored to a first approximation and idealize the world as a flat level plane $\mathbf{R}^\mathrm{f}$. Let $\mathbf{R}^\mathrm{f}_\mathbf{bot}$ and $\tilde{\mathbf{R}}^\mathrm{f}_\mathbf{bot}$ be the current position of the robot and the position of the robot during the previous update, respectively. $\tilde{\mathbf{C}}$ and $\mathbf{C}$ denote the cylinder after the updates at $\tilde{\mathbf{R}}^\mathrm{f}_\mathbf{bot}$ and $\mathbf{R}^\mathrm{f}_\mathbf{bot}$, respectively. Let $\Delta N$, $\Delta E$ be the north and east components of the vector from $\tilde{\mathbf{R}}^\mathrm{f}_\mathbf{bot}$ to $\mathbf{R}^\mathrm{f}_\mathbf{bot}$, respectively. $T$ is the magnitude of the same vector:

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

The incremental updating function is given by

$$\mathbf{C}_{n,m} = \max\left(1, \tilde{\mathbf{C}}_{n,m}\frac{c_\mathrm{fgt} - T}{c_\mathrm{fgt}}\right), \qquad (9)$$

where $c_\mathrm{fgt}$ is the translation distance required to erase all information in a single update. $\mathbf{C}_{n,m}$ cannot decay to less than 1, the minimum force value. $c_\mathrm{fgt}$ is tuned for the desired system behavior: small values of $c_\mathrm{fgt}$ cause the robot to forget information more quickly than large values. The relationship between long-term decay behavior and translation is influenced by the frequency with which Eq. (9) is applied. $\mathbf{C}$ should be updated at a constant rate of time to ensure predictable decay vs. distance and velocity.

### 4.2.2. Depth-Based Updating

If depth data exist for $\mathbf{S}$, then $\mathbf{C}$ can be updated by calculating the motion of depth points relative to the robot. It is convenient to store depth data in a separate "cylindrical" array $\mathbf{Q}$. For each force value $f(\mathbf{S}_{n,m})$ placed into $\mathbf{C}$ as a function of pixel data $\mathbf{S}_{n,m}$, a corresponding 3D position $\mathbf{R_p}(\mathbf{S}_{n,m})$ in the local robot frame is placed into $\mathbf{Q}$ at the same location. Let $\tilde{\mathbf{Q}}$ and $\mathbf{Q}$ denote the depth cylinder at the robot's previous location $\tilde{\mathbf{R}}^{\mathrm{f}}_{\mathbf{bot}}$ and current location $\mathbf{R}^{\mathrm{f}}_{\mathbf{bot}}$, respectively. We assume that depth $\tilde{\mathbf{Q}}_{\tilde{n},\tilde{m}}$ was generated from a point projected to the center of pixel $\mathbf{S}_{\tilde{n},\tilde{m}}$ when the robot was at $\tilde{\mathbf{R}}^{\mathrm{f}}_{\mathbf{bot}}$. Let $\mathbf{R_p}$ be the point in $\mathbf{R}$ that generated $\tilde{\mathbf{Q}}_{\tilde{n},\tilde{m}}$. As the robot moves, the image projection of $\mathbf{R_p}$ will change. When 3D points are retained for the cylinder image, we use $\mathbf{R}^{\mathrm{f}}_{\mathbf{bot}}$ and $\tilde{\mathbf{R}}^{\mathrm{f}}_{\mathbf{bot}}$ to compute the differential position $\mathbf{R}_\Delta$ between frames. We then apply $\mathbf{R}_\Delta^{-1}$ to the points in $\tilde{\mathbf{Q}}$ and propagate the points to their new positions in $\mathbf{Q}$ if they are within the bounds of the cylinder. This provides a mapping from $\tilde{\mathbf{Q}}_{\tilde{n},\tilde{m}}$ to $\mathbf{Q}_{n,m}$. Information from multiple locations in $\tilde{\mathbf{Q}}$ may migrate to $\mathbf{Q}_{n,m}$ as one obstacle is occluded by another. $\mathbf{Q}_{n,m}$ is updated with the nearest point because closer obstacles present a more immediate navigational hazard. If a nondepth force metric is used to populate $\mathbf{C}$, then $\mathbf{C}$ is updated as follows:

$$\mathbf{C}_{n,m} = \tilde{\mathbf{C}}_{\tilde{n},\tilde{m}}.$$

If a depth-based force metric is used, then $\mathbf{C}$ is a synthetic disparity calculated from $\mathbf{Q}$ by modifying Eq. (2) to operate on $\mathbf{Q}_{n,m}$ instead of $\mathbf{S}_{n,m}$:

$$F_i^{\mathrm{d}} = \mathbf{C}_{n,m} = 1 + c_{\mathrm{scl}} \left| \frac{c_{\mathrm{f}} c_{\mathrm{b}}}{\mathbf{Q}_{n,m}} - \frac{c_{\mathrm{f}} c_{\mathrm{b}}}{\mathbf{Q}^{\mathrm{f}}_{n,m}} \right|,$$

where $\mathbf{Q}^{\mathrm{f}}_{n,m}$ is the depth cylinder observed from a flat level plane.

## 5. HIERARCHICAL PLANNERS

Image planning provides a variable world view that maximizes map resolution and minimizes the number of map elements required to model the world, given an image sensor reading. However, environmental knowledge is constrained by the current line of sight and far-field data are less refined than near-field data. We believe that the shortcomings of image planning are complemented by the strengths of Cartesian planning and combine the two in a *hybrid hierarchical planner*. The global planner uses a 2D, top-down Cartesian occupancy grid $\mathbf{G}$ to find a coarse path $P^{\min}_{\mathbf{G}}$, whereas the local cylindrical planner navigates toward subgoals $\mathbf{R_{sub}}$ a predetermined distance along $P^{\min}_{\mathbf{G}}$. To determine what advantages (if any) the local cylindrical planner provides to the hybrid hierarchical planner, we evaluate the proposed system against a double 2D, top-down Cartesian hierarchical planner. We refer to the latter system as the *Cartesian hierarchical planner*. Both hierarchical systems use an identical global planner that runs independently on its own processor.

In addition to the assumptions outlined in Section 2.1, the hierarchical planners assume knowledge of global position and orientation. This is necessary to populate, update, and place start and goal positions in the global map and to place $\mathbf{R_{sub}}$ in the local map. Because $\mathbf{R_{sub}}$ is defined to be a specific distance along $P^{\min}_{\mathbf{G}}$, any particular $\mathbf{R_{sub}}$ will be replaced by a new $\mathbf{R_{sub}}$ before the former is actually reached (except when the robot converges on the global goal). This method of planner interaction works well in practice, as long as $\mathbf{R_{sub}}$ is within the sensor range of the robot.

### 5.1. The Global Cartesian Planner

As with the image planner (Section 3) and the cylindrical planner (Section 4), our global Cartesian planner separates the ideas of work, force, and distance. Let $\mathbf{G}^\Gamma$ represent the 8-connected graph of $\mathbf{G}$. The cost of traveling from node $\mathbf{G}^\Gamma_{k,j}$ to node $\mathbf{G}^\Gamma_{n,m}$ along arc $i$ is determined by work $W_i = F_i D_i$, where force $F_i = \mathbf{G}_{n,m}$ and distance $D_i$ are determined by force and distance metrics, respectively. Map grids are square, and we use the $L^2$ norm [Eq. (4)] as a distance metric. Values are added to $\mathbf{G}$ by projecting $\mathbf{O}$ onto the ground plane using the robot's position, orientation, and stereo disparity information. Therefore, $F_i$ is determined by the force metric used to populate $\mathbf{O}$. $\mathbf{G}$ is preprocessed by a dilation of $\alpha$, the map equivalent of a robot half-width $c_{\mathrm{wth}}/2$ plus a buffer $c_{\mathrm{buf}}$:

$$\mathbf{O}_{n,m} = \max_{k,j} \left( \mathbf{O}_{n \pm k, m \pm j} \right),$$

where $\lfloor -\alpha \rfloor \le k \le \lceil \alpha \rceil$, $1 \le (n \pm k) \le h$, $\lfloor -\alpha \rfloor \le j \le \lceil \alpha \rceil$, and $1 \le (m \pm j) \le w$. $h$ and $w$ represent the number of rows and columns in $\mathbf{G}$, respectively. $\alpha$ is calculated:

$$\alpha = c_{\mathrm{msc}} \left( c_{\mathrm{wth}}/2 + c_{\mathrm{buf}} \right),$$

where $c_{\text{msc}}$ is the map scale. The A* algorithm is used to find the minimum work path $P_{\mathbf{G}}^{\min}$ from the start node $\mathbf{G}_{\mathbf{s}}^{\Gamma} = \mathbf{G}_{n_s,m_s}^{\Gamma}$ to the goal node $\mathbf{G}_{\mathbf{g}}^{\Gamma} = \mathbf{G}_{n_g,m_g}^{\Gamma}$, where $\mathbf{G}_{\mathbf{s}}^{\Gamma}$ and $\mathbf{G}_{\mathbf{g}}^{\Gamma}$ are found by projecting robot and goal coordinates into map space, respectively. $P_{\mathbf{G}}^{\min}$ is transformed into GPS coordinates $P_{\mathbf{R}}^{\text{gps}}$ before being passed to a local planner.

### 5.2. The Local Cylindrical Planner

The local cylindrical subsystem of the hybrid hierarchical planner is identical to the system described in Section 4., except that $\mathbf{C}_{\mathbf{g}}^{\Gamma}$ is found by projecting $\mathbf{R}_{\mathbf{sub}}$ into image space, where $\mathbf{R}_{\mathbf{sub}}$ is the first position along $P_{\mathbf{R}}^{\text{gps}}$ that is at least $c_{\text{sub}}$ away from the robot and $c_{\text{sub}}$ is a system parameter. Owing to the relative frame rates of the local and global planners, the same $P_{\mathbf{R}}^{\text{gps}}$ may be used to determine multiple $\mathbf{R}_{\mathbf{sub}}$.

### 5.3. The Local Cartesian Planner

The local Cartesian subsystem of the baseline hierarchical planner is similar to the global planner (Section 5.1.), except for five differences: (1) $\mathbf{G}_{\mathbf{g}}^{\Gamma}$ is found by projecting $\mathbf{R}_{\mathbf{sub}}$ into local map space, (2) the local path $P_{\mathbf{G}}^{\min}$ is used for servo control, and (3–5) the map has a higher resolution, is fixed in size, and is centered on the robot. $\mathbf{R}_{\mathbf{sub}}$ is defined to be the first position along $P_{\mathbf{R}}^{\text{gps}}$ that is at least $c_{\text{sub}}$ from the robot, and the same $P_{\mathbf{R}}^{\text{gps}}$ may be used to determine multiple $\mathbf{R}_{\mathbf{sub}}$. Servoing is accomplished by steering at $\mathbf{R}_{\mathbf{tgt}}$, where $\mathbf{R}_{\mathbf{tgt}}$ is found by projecting $\mathbf{G}_{\mathbf{tgt}}^{\Gamma} = \mathbf{G}_{n_{\text{tgt}},m_{\text{tgt}}}^{\Gamma}$ into the real world and $\mathbf{G}_{\mathbf{tgt}}^{\Gamma}$ is defined to be $c_{\text{tgt}}$ along the local path $P_{\mathbf{G}}^{\min}$. $c_{\text{sub}}$ and $c_{\text{tgt}}$ are system parameters. Forward velocity and turning angle are defined:

$$\text{speed} = \text{speed}_{\max} \max[0, \cos(\psi_{\mathbf{R}})],$$
$$\text{turn} = \psi_{\mathbf{R}},$$

where $\psi_{\mathbf{R}}$ is the angular distance between the current heading and $\mathbf{R}_{\mathbf{sub}}$ and $-\pi \leq \psi_{\mathbf{R}} \leq \pi$. Speed will increase as the target point approaches a position in front of the robot, and turning angle will steer the robot toward the target point.

## 6. THE LAGR ROBOT

Our mobile robot platform (Figure 7) is provided in conjunction with the DARPA Learning Applied to Ground Robotics (LAGR) program. The length, width, and height dimensions of the robot are approximately 1.2 × 0.75 × 1.0 m, respectively. Sensors include two forward-facing Point Grey BumbleBee 2 stereo camera pairs, a Garmin GPS receiver, a magnetic compass, wheel odometers, two forward-facing infrared sensors (unused in this work), and a front bumper sensor. The stereo camera pairs output stereo disparity and RGB color data. We have found disparity data to have a maximum range of approximately 15 m, although the usable range is generally between 5 and 10 m. There are four processing units: one dedicated to each of the two stereo camera pairs, one for path planning, and one that is a servo controller (the former three computers have dual-core processors). Translation and rotation are achieved via two independently driven front wheels.

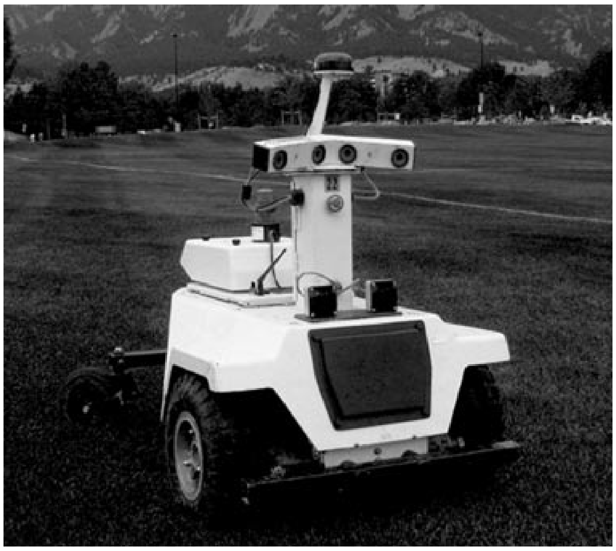


**Figure 7.** The LAGR robot platform.

## 7. EXPERIMENTS

All of the methods we have proposed are experimentally evaluated, including two force metrics ($F_i^{\text{d}}$ and $F_i^{\text{c}}$ of Sections 3.2.1 and 3.2.2, respectively), two distance metrics ($D_i^{\mathbf{S}}$ and $D_i^{\mathbf{R}^{\text{f}}}$ of Sections 3.3.1 and 3.3.2, respectively), and four planning systems (image, cylindrical, hybrid hierarchical, and Cartesian hierarchical from Sections 3, 4, 5, and 5, respectively). We also test two cylindrical updating techniques (translation-based forgetting and depth-based updating from Sections 4.2.1 and 4.2.2, respectively). Our toy color force metric $F_i^{\mathrm{c}}$ requires that safe terrain and obstacles be different colors; therefore, we evaluate $F_i^{\mathrm{c}}$ only in environments where this is the case. As previously stated, $F_i^{\mathrm{c}}$ is not useful in most environments and is used only to test whether our image space systems can function using color. The depth-based cylindrical updating scheme assumes the existence of depth information; therefore, it is evaluated only in conjunction with the disparity force metric $F_i^{\mathrm{d}}$.

We test the following three hypotheses: (1) The flat world distance metric $D_i^{\mathbf{R}^{\mathrm{f}}}$ will outperform the image distance metric $D_i^{\mathbf{S}}$ because $D_i^{\mathbf{R}^{\mathrm{f}}}$ more accurately models real-world distances. (2) The cylindrical planner is susceptible to translational oscillation near large obstacles, due to a lack of translational memory. (3) A local image space planner will outperform a local 2D Cartesian planner as a subsystem of a hierarchical planner because the former represents near-field information at a higher resolution.

## 7.1. System Parameters

We achieve an effective FOV of 110 deg (61 grid elements) by having each camera insert information into the cost map (image or cylinder) based on its relative pose. To achieve a frame rate of at least 10 Hz, we define the size of the image planner map **O** to be $h = 40$ grid elements high and $w = 61$ grid elements wide. Similarly, the cylindrical cost map **C** is defined to be $h = 40$ high and $w_{\mathrm{p}} = 200$ wide (i.e., around the perimeter). In either case, the image from a particular camera is 40 grid elements wide. This resolution is achieved by subsampling **S**. Map updates do not incorporate old information. The most recent image determines map values at positions of camera overlap. There is no longer a one-to-one mapping from **S** to **Q** or **C**. However, there is a one-to-one mapping from the subsampled version of **S** to **Q** and **C**. The mask used for denoising the cost map is defined to be 4 pixels high × 1 pixel wide and is chosen to modify

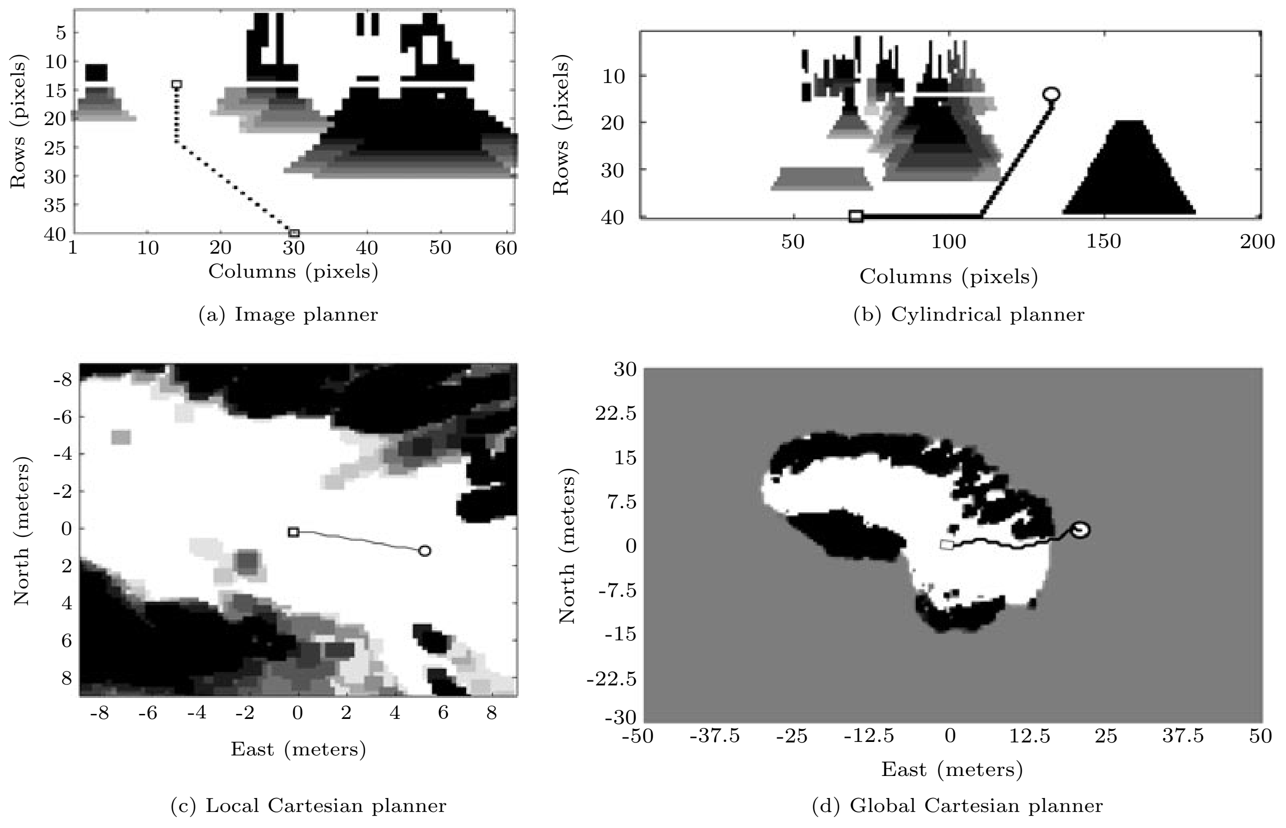


**Figure 8.** Cost maps with paths: light to dark represents low to high cost, respectively.

obstacle width as little as possible while still eliminating noise. A cost map from the image planner is displayed in Figure 8(a), and a cost map from the cylindrical planner is displayed in Figure 8(b).

We set the maximum speed of the robot to 1 m/s in all experiments. The parameter $c_{\text{sub}}$ (Section 5.2) used to place the local subgoal $\mathbf{R_{sub}}$ is defined to be 5 m. This distance was chosen to make the local subsystem responsible for navigation around small- to medium-sized obstacles (e.g., trees, rocks, bushes, and boulders). The image and cylindrical planner parameter $c_{\text{tgt}}$ (Section 3.5), used to place $\mathbf{O}^{\Gamma}_{\mathbf{tgt}}$, is defined to be the 12th node along $P^{\min}_{\mathbf{O}}$ or the goal node if the path contains fewer than 12 nodes. The parameter $c_{\text{scl}}$ (Section 3.2) is defined to be 10 or 30 if force is determined by disparity or color, respectively. The parameter $c_{\text{thd}}$ (Section 3.2) is defined to be 10, and the parameter $c_{\text{flt}}$ (Section 3.4.1) is defined to be 3. The parameter $c_{\text{fgt}}$ used in Eq. (9) to control translation-based forgetting is defined to be 0.4 m (obstacles are remembered farther than 0.4 m because the frame rate is at least 10 Hz). The values of $c_{\text{tgt}}$, $c_{\text{scl}}$, $c_{\text{thd}}$, $c_{\text{flt}}$, and $c_{\text{fgt}}$ have been tuned by human evaluation of the system's ability to navigate around a simple obstacle that is 0.5 m wide, 0.5 m tall, and 0.25 m deep.

The local Cartesian planner used in the Cartesian hierarchical system models an 18 × 18 m square of the world, and the robot is always located in the center of the map graph $\mathbf{G}^{\Gamma}$. This ensures that at least 5 m of the world is modeled between the robot and $\mathbf{R_{sub}}$ and at least 4 m is modeled between $\mathbf{R_{sub}}$ and the map boundary. We believe these distances are suitable for navigation around small- to medium-sized obstacles. A cost map from the local Cartesian planner is displayed in Figure 8(c). To make the comparison between the two hierarchical planners as fair as possible, we define the granularity of the local Cartesian occupancy grid $\mathbf{G}$ to be 0.2 m. Thus, $\mathbf{G}^{\Gamma}$ has 8,100 nodes and the local cylindrical map graph $\mathbf{C}^{\Gamma}$ has 8,000 nodes. The areas of local Cartesian map world elements are 0.04 m squared. The areas of image cylindrical map world elements vary from 0.019 to $\infty$ meters squared, from the near to far field, respectively. The frame rate of the local Cartesian planner is comparable to that of the local cylindrical planner.

## 7.2. Experiment 1: Shakeout Course

Experiment 1 consists of the NIST (National Institute of Standards and Technology) LAGR shakeout course [Figure 9(a)]. The goal is placed 20 m north of the starting location. The course contains three obstacles constructed out of four, six, and six large plastic tubs, respectively. The first obstacle is placed 5 m toward the goal, directly in front of the robot. The next two obstacles are placed 5 m farther, separated by 3 m. The widths of the obstacles are 1.2, 1.8, and 1.8 m, respectively, and the length and height of the obstacles are 0.4 and 0.8 m, respectively. The planning systems tested in Experiment 1 include all combinations of the image and cylindrical planners, $F_i^{\text{d}}$ and $F_i^{\text{c}}$ force metrics, and $D_i^{\mathbf{S}}$ and $D_i^{\mathbf{R}^{\text{f}}}$ distance metrics. Additionally, both cylindrical updating schemes (translation-based forgetting and depth-based updating) are tested in conjunction with a $F_i^{\text{d}}$ cylindrical planner. Each combination is run three times, resulting in 30 total runs.

The primary purpose of Experiment 1 is to show that image planning is possible over a wide range of system implementations. Additionally, we hope to observe some discriminating trends along the different dimensions of our system space. GPS data from each run are superimposed on an illustration

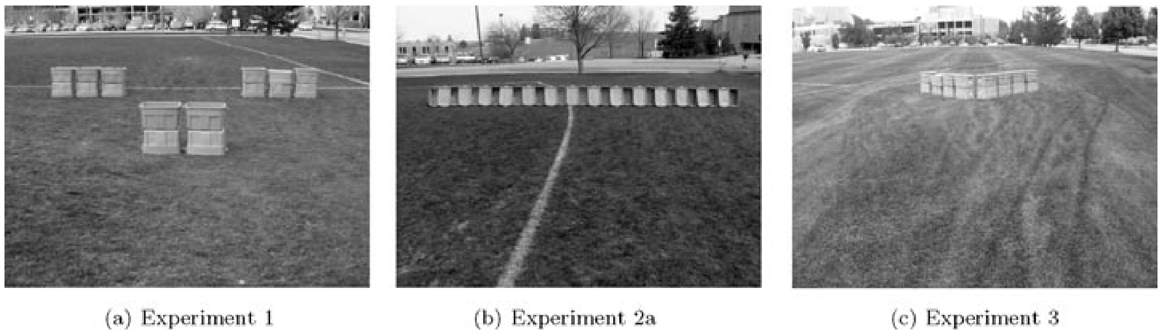

(a) Experiment 1 (b) Experiment 2a (c) Experiment 3

**Figure 9.** Photos of test courses from the start positions.

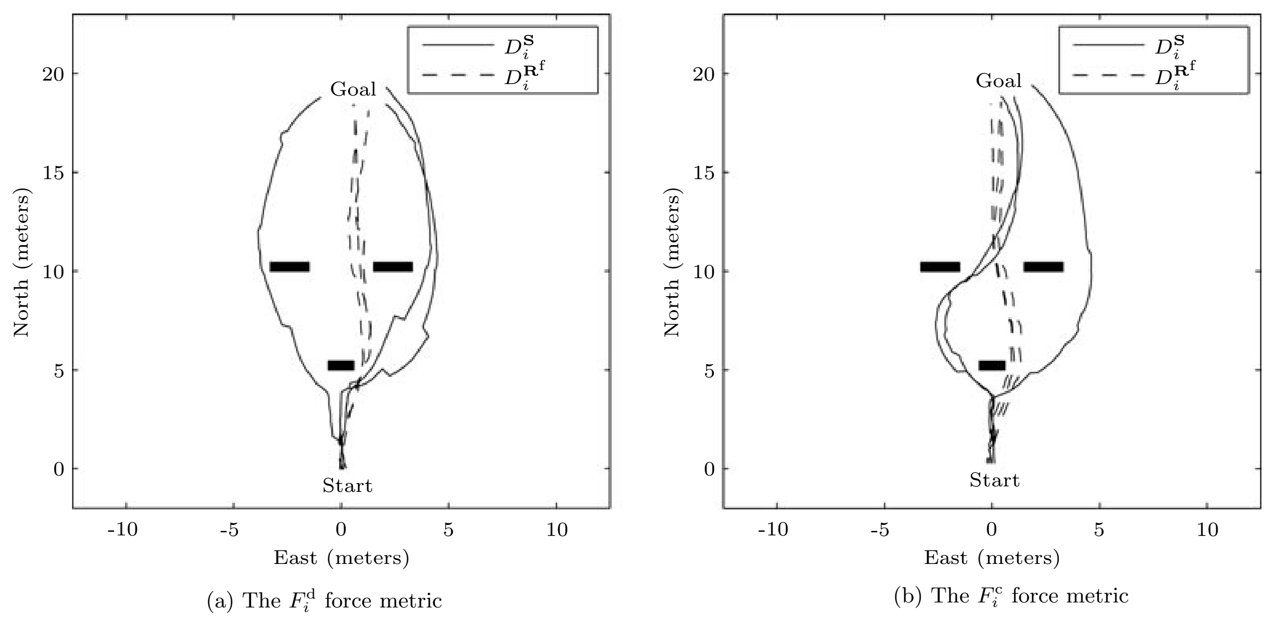


**Figure 10.** The image planner in Experiment 1: runs using the $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^{\mathrm{f}}}$ distance metrics appear solid or dashed, respectively.

of the course in Figures 10 and 11, and run times are displayed in Table I (note that TF and DU denote translation-based forgetting and depth-based updating, respectively). Image planning systems are displayed in Figure 10, and cylindrical planning systems appear in Figure 11. Figures 10(a) and 11(a) display tests using the $F_i^{\mathrm{d}}$ force metric, and Figures 10(b) and 11(b) show tests using the $F_i^{\mathrm{c}}$ force metric. The $D_i^{\mathbf{S}}$ and $D_i^{\mathbf{R}^{\mathrm{f}}}$ distance metrics are plotted with solid and dashed lines, respectively.

## 7.3. Experiment 2: Walls of 10 and 50 m

Experiment 2a consists of a single wall placed 10 m east of the starting location. The goal is positioned another 10 m east of the wall. The width, length, and height dimensions of the wall are 10, 0.4, and 0.6 m, respectively. A photo of the course is displayed in Figure 9(b). The course consists of the aforementioned blue tubs and grassy field. The same planning systems are tested as in Experiment 1, and trials are repeated three times. The actual GPS data from each run are superimposed on an illustration of the course in Figures 12 and 13. The planning systems displayed in Figures 12(a), 12(b), 13(a), and 13(b) are identical to those displayed in Figures 10(a), 10(b), 11(a), and 11(b), respectively. Run times are displayed in Table II.

Experiment 2a is designed to test our hypothesis that the cylindrical planner is vulnerable to translational oscillation when large obstacles obscure the goal. Additionally, we expect to observe rotational oscillation when the goal is blocked by obstacles wider than the FOV. A test is halted if the robot oscillates eight times without making any forward progress toward the goal (the oscillation count resets to 0 whenever progress is made). Elapsed time values of $\infty$ denote that a system did not complete the course.

Experiment 2b consists of a single wall placed 5 m north of the starting location. The goal is positioned another 5 m north of wall. The width, length, and height dimensions of the wall are 50, 0.1, and 1 m, respectively (Figure 14). We test the cylindrical planner using the force-from-disparity metric $F_i^{\mathrm{d}}$ in conjunction with $D_i^{\mathbf{S}}$ distance and depth-based updating. The system is run three times. Experiment 2b is designed to test the limits of this particular planner because it completed Experiment 2a all three times.

## 7.4. Experiment 3: V-Shaped Course

Experiment 3 consists of a symmetrical "V"-shaped obstacle placed 10 m north of the starting location.

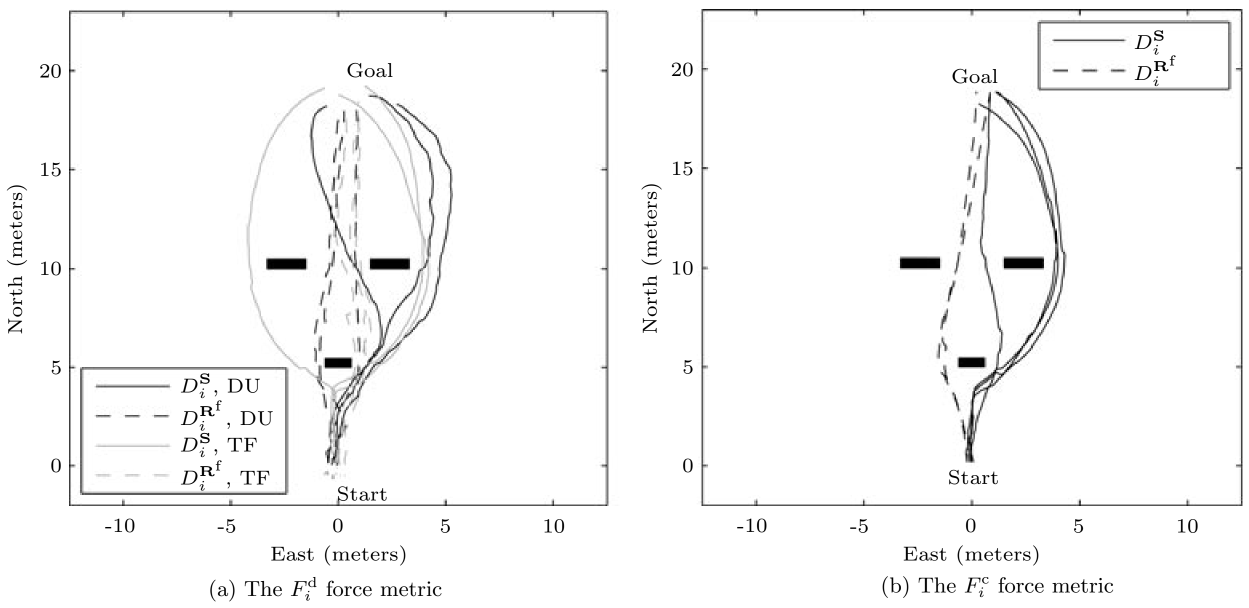


**Figure 11.** The cylindrical planner in Experiment 1. Runs using the $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^f}$ distance metrics appear solid or dashed, respectively. Runs using depth-based updating or translation-based forgetting are black or gray, respectively.

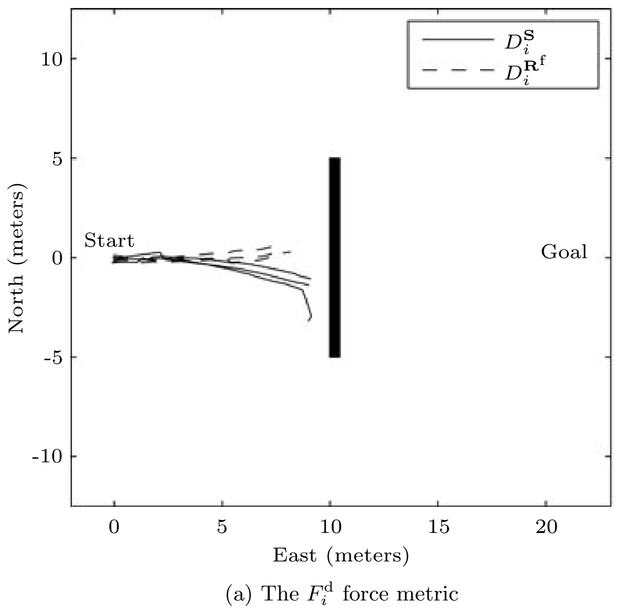
The V consists of two 2.4-m-long walls joined at an angle of 97 deg. The depth and height of the obstacles are 0.4 and 0.8 m, respectively. The goal is centered between the far ends of the two walls. A photo of the course is displayed in Figure 9(c). The course uses the same blue tubs and grassy field as Experiments 1 and 2a, and the same planning systems are also tested. Trials are repeated three times. The actual GPS data from each run are superimposed on an illustration of the course in Figures 15 and 16. The planning systems displayed in Figures 15(a), 15(b), 16(a), and 16(b) are identical to those displayed in Figures 10(a), 10(b), 11(a), and 11(b). Experiment 3 evaluates the image and cylindrical planners on a controlled course not specifically designed to induce pathological behavior. The run times are given in Table III.

**Table I.** Experiment 1 run times in seconds.

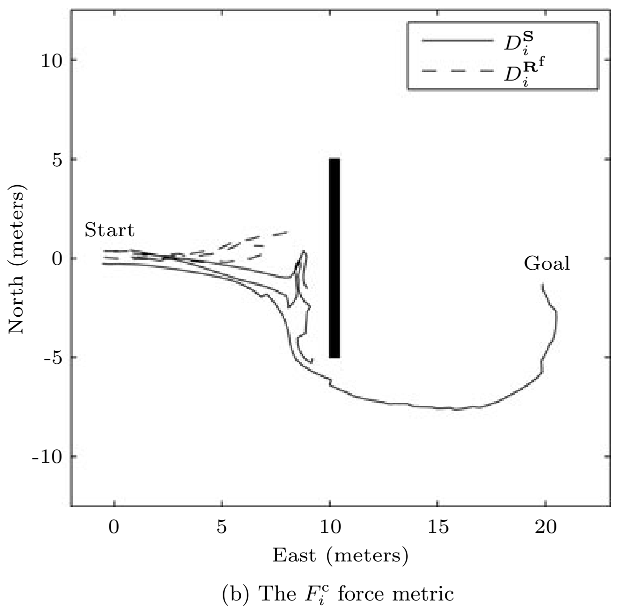
| System | Runs 1–3 | | | Mean |
|---|---|---|---|---|
| | Image planner | | | |
| $F_i^d$ $D_i^{\mathbf{S}}$ | 30.60 | 31.07 | 30.26 | 30.64 |
| $F_i^d$ $D_i^{\mathbf{R}^f}$ | 24.89 | 25.17 | 25.28 | 25.11 |
| $F_i^c$ $D_i^{\mathbf{S}}$ | 34.06 | 33.00 | 32.02 | 33.03 |
| $F_i^c$ $D_i^{\mathbf{R}^f}$ | 24.94 | 25.16 | 25.11 | 25.07 |
| | Cylindrical planner | | | |
| DU $F_i^d$ $D_i^{\mathbf{S}}$ | 30.66 | 34.43 | 33.15 | 32.75 |
| DU $F_i^d$ $D_i^{\mathbf{R}^f}$ | 25.25 | 28.62 | 25.02 | 26.30 |
| TF $F_i^d$ $D_i^{\mathbf{S}}$ | 30.75 | 30.54 | 31.87 | 31.05 |
| TF $F_i^d$ $D_i^{\mathbf{R}^f}$ | 24.19 | 26.02 | 25.09 | 25.10 |
| TF $F_i^c$ $D_i^{\mathbf{S}}$ | 31.14 | 30.25 | 29.40 | 30.26 |
| TF $F_i^c$ $D_i^{\mathbf{R}^f}$ | 25.64 | 25.22 | 25.43 | 25.43 |

## 7.5. Experiments 4–7: Natural Terrain Courses

Experiments 4–7 are in natural terrain. They are conducted to demonstrate that the hybrid and Cartesian hierarchical systems actually perform in unknown unstructured outdoor environments and also to determine whether the behavior of the stand-alone cylindrical planning systems transfers to hybrid hierarchical systems. Experiment 4 consists of boulders and trees on a lightly undulating ground surface with grass. The course is 95 m long. GPS paths are displayed in Figure 17(a). An approximate representation of the course can be seen in Figure 17(b). All systems are tested twice. The run times are displayed in Table IV.



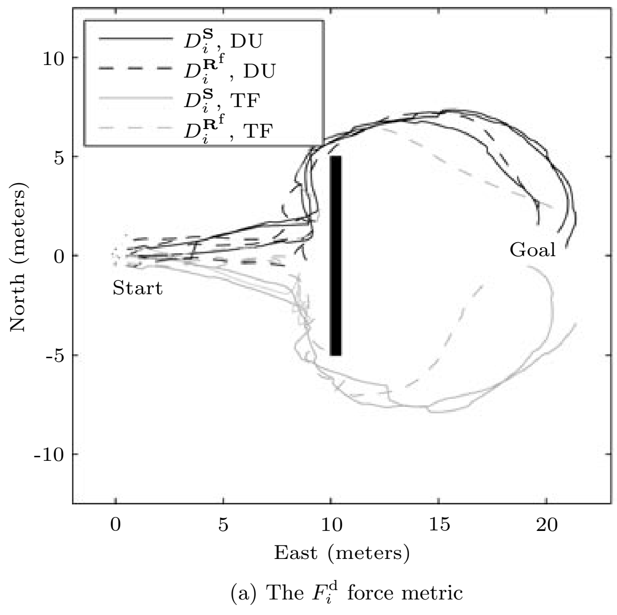
(a) The $F_i^{\rm d}$ force metric



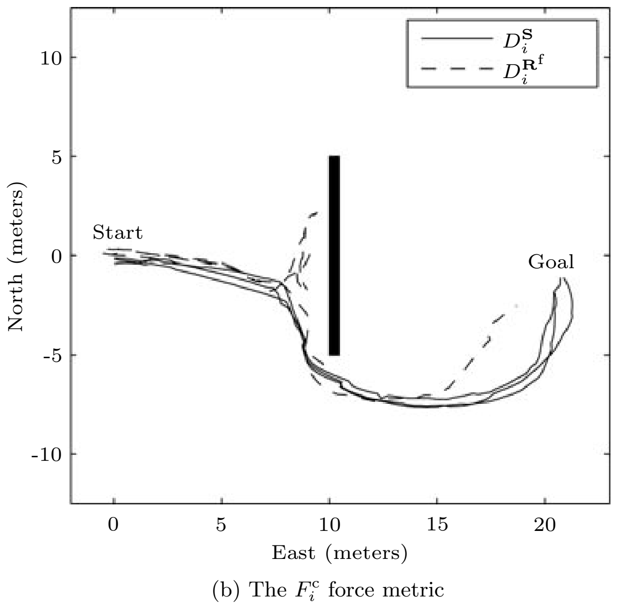
(b) The $F_i^{\rm c}$ force metric

**Figure 12.** The image planner in Experiment 2a: runs using the $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^f}$ distance metrics appear solid or dashed, respectively.

Experiments 5 and 6 contain shrubs, trees, and dead grass ranging in height from 3 cm to 0.5 m. The terrain is rocky, with rocks protruding above the ground surface 4–12 cm. Photos of the courses are displayed in Figures 18(a) and 18(b), respectively. In Experiment 6 there is a 10-m-wide stand of interwoven trees blocking the robot's view of the goal from the starting position. The courses are 30 and 35 m long,



(a) The $F_i^{\rm d}$ force metric



(b) The $F_i^{\rm c}$ force metric

**Figure 13.** The cylindrical planner in Experiment 2a. Runs using the $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^f}$ distance metrics appear solid or dashed, respectively. Runs using depth-based updating or translation-based forgetting are black or gray, respectively.

**Table II.** Experiment 2a run times in seconds.

| System | Runs 1–3 | | | Mean |
|---|---|---|---|---|
| | Image planner | | | |
| $F_i^{\text{d}}\ D_i^{\mathbf{S}}$ | ∞ | ∞ | ∞ | ∞ |
| $F_i^{\text{d}}\ D_i^{\mathbf{R}^{\text{f}}}$ | ∞ | ∞ | ∞ | ∞ |
| $F_i^{\text{c}}\ D_i^{\mathbf{S}}$ | 35.31 | ∞ | ∞ | ∞ |
| $F_i^{\text{c}}\ D_i^{\mathbf{R}^{\text{f}}}$ | ∞ | ∞ | ∞ | ∞ |
| | Cylindrical planner | | | |
| DU $F_i^{\text{d}}\ D_i^{\mathbf{S}}$ | 41.72 | 46.37 | 44.94 | 44.34 |
| DU $F_i^{\text{d}}\ D_i^{\mathbf{R}^{\text{f}}}$ | ∞ | ∞ | 53.92 | ∞ |
| TF $F_i^{\text{d}}\ D_i^{\mathbf{S}}$ | 39.52 | ∞ | 49.07 | ∞ |
| TF $F_i^{\text{d}}\ D_i^{\mathbf{R}^{\text{f}}}$ | 65.84 | ∞ | 46.51 | ∞ |
| TF $F_i^{\text{c}}\ D_i^{\mathbf{S}}$ | 39.46 | 40.42 | 49.22 | 43.03 |
| TF $F_i^{\text{c}}\ D_i^{\mathbf{R}^{\text{f}}}$ | ∞ | 69.26 | ∞ | ∞ |

respectively. GPS paths are displayed in Figures 19(a) and 19(b), respectively. All systems are tested three times, and the run times are given in Tables V and VI, respectively.

Experiment 7 evaluates the hierarchical planners on a grass and dirt field with trees and boulders as obstacles. A photo of the course is displayed in Figure 18(c). The course is 19 m long. The purpose of Experiment 7 is to compare the best performing hybrid hierarchical planners (as determined by previous experiments) against the Cartesian hierarchical planner with enough trials to provide statistical significance. Hybrid hierarchical planners using the $D_i^{\mathbf{R}^{\text{f}}}$ distance metric with either cylindrical updating strategy are tested. We selected this course because we believe that cluttered environments give hybrid hierarchical planners an advantage over the Cartesian hierarchical planner. All systems were tested 10 times. GPS paths are displayed in Figure 20, and the run times are given in Table VII.

## 8. DISCUSSION

### 8.1. Force Metrics: $F_i^{\text{d}}$, $F_i^{\text{c}}$

We use force metrics from disparity $F_i^{\text{d}}$ and color $F_i^{\text{c}}$ to evaluate the effects of different sensors and force definitions on image planning. Comparison between $F_i^{\text{d}}$ and $F_i^{\text{c}}$ is possible only on the subset of courses for which $F_i^{\text{c}}$ is tested: Experiments 1, 2a, and 3 (the NIST shakeout, 10-m wall, and V-shaped courses, respectively). In Experiment 1, both $F_i^{\text{d}}$ and $F_i^{\text{c}}$ are effective and there is no clear winner with respect to run time. In contrast, Experiments 2a and 3 show $F_i^{\text{c}}$ systems encountering oscillation less frequently and progressing farther when oscillation does occur. This suggests that $F_i^{\text{c}}$ has an advantage over $F_i^{\text{d}}$, which we attribute to the greater range of color vs. disparity. Systems using $F_i^{\text{c}}$ are more likely to see and avoid an obstacle before moving close enough to induce oscillation. It is worth restating that any *predefined* color force metric is useful only in a subset of environments. The obstacle and ground colors of Experiments 1, 2a, and 3 were chosen to be compatible with $F_i^{\text{c}}$; therefore, the superiority of $F_i^{\text{c}}$ to $F_i^{\text{d}}$ may be due to course appearance.

### 8.2. Distance Metrics: $D_i^{\mathbf{S}}$, $D_i^{\mathbf{R}^{\text{f}}}$

$D_i^{\mathbf{S}}$ and $D_i^{\mathbf{R}^{\text{f}}}$ define distance as the $L^2$ norm in image space and its projected distance along a flat level ground plane, respectively. We have hypothesized that $D_i^{\mathbf{R}^{\text{f}}}$ will outperform $D_i^{\mathbf{S}}$ because the former better approximates real-world distances than the latter. Our experiments validate this hypothesis for cases without rotational or translational oscillation. However, they also show that $D_i^{\mathbf{S}}$ is less susceptible to oscillation than $D_i^{\mathbf{R}^{\text{f}}}$. In experiments free from oscillation, systems using $D_i^{\mathbf{R}^{\text{f}}}$ achieve the goal faster than those using $D_i^{\mathbf{S}}$. Examples include the NIST shakeout and natural terrain courses (Experiments 1 and

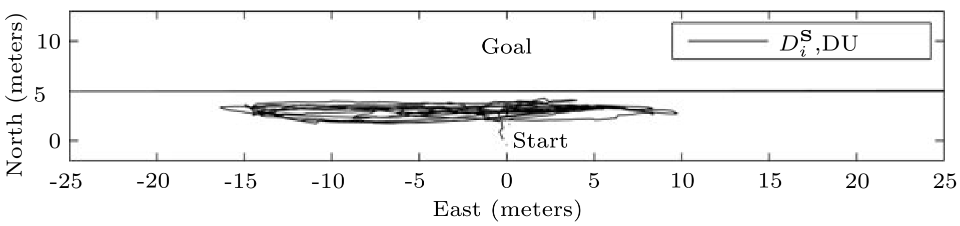
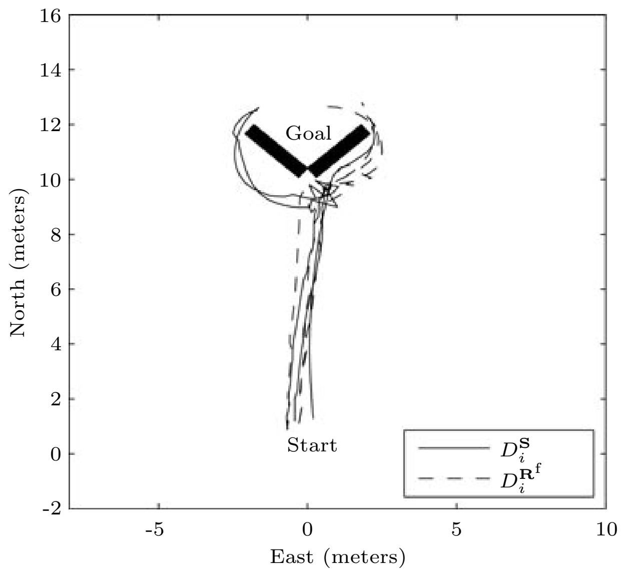
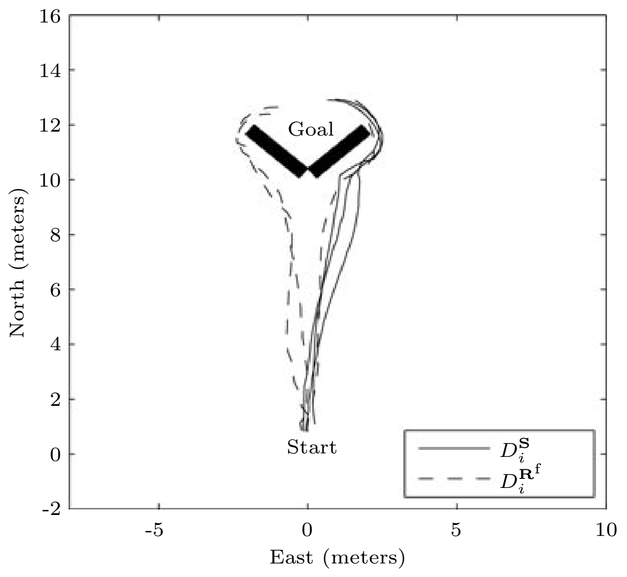


**Figure 14.** Experiment 2b: the $F_i^{\text{d}}$ cylindrical planner using $D_i^{\mathbf{S}}$ and depth-based updating.

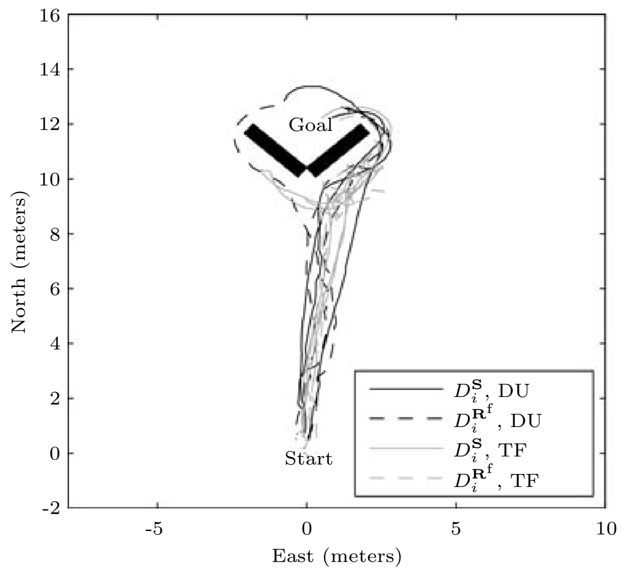


(a) The $F_i^d$ force metric

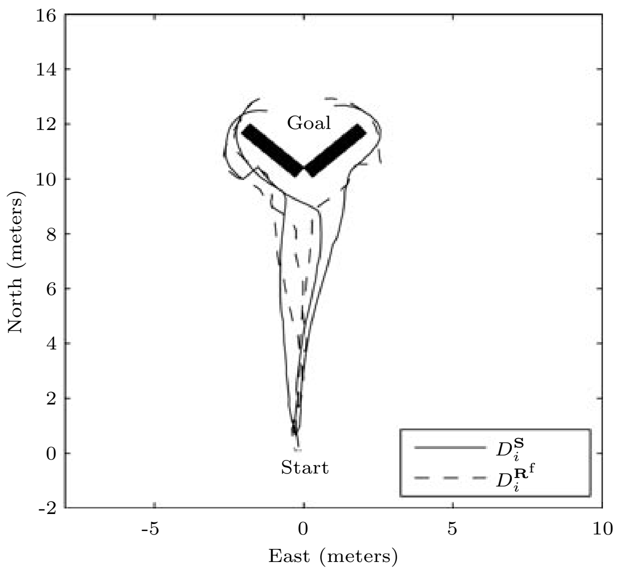


(b) The $F_i^c$ force metric

**Figure 15.** The image planner in Experiment 3: runs using the $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^f}$ distance metrics appear solid or dashed, respectively.



(a) The $F_i^d$ force metric



(b) The $F_i^c$ force metric

**Figure 16.** The cylindrical planner in Experiment 3. Runs using the $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^f}$ distance metrics appear solid or dashed, respectively. Runs using depth-based updating or translation-based forgetting are black or gray, respectively.

**Table III.** Experiment 3 run times in seconds.

| System | Runs 1–3 | | | Mean |
|---|---|---|---|---|
| | Image planner | | | |
| $F_i^{\mathrm{d}}\ D_i^{\mathbf{S}}$ | 20.53 | 33.01 | 56.42 | 36.65 |
| $F_i^{\mathrm{d}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 57.98 | 22.76 | 38.62 | 39.79 |
| $F_i^{\mathrm{c}}\ D_i^{\mathbf{S}}$ | 23.11 | 22.80 | 22.07 | 22.66 |
| $F_i^{\mathrm{c}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 21.69 | 23.14 | 22.13 | 22.32 |
| | Cylindrical planner | | | |
| DU $F_i^{\mathrm{d}}\ D_i^{\mathbf{S}}$ | 19.82 | 32.91 | 22.24 | 24.99 |
| DU $F_i^{\mathrm{d}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 26.35 | 32.68 | 22.81 | 27.28 |
| TF $F_i^{\mathrm{d}}\ D_i^{\mathbf{S}}$ | $\infty$ | 68.15 | 38.18 | $\infty$ |
| TF $F_i^{\mathrm{d}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 21.74 | 74.90 | 49.85 | 48.83 |
| TF $F_i^{\mathrm{c}}\ D_i^{\mathbf{S}}$ | 24.17 | 23.06 | 24.63 | 23.96 |
| TF $F_i^{\mathrm{c}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 21.46 | 32.47 | 32.78 | 28.90 |

4–6, respectively). On the other hand, in cases in which terminal oscillation is observed, $D_i^{\mathbf{S}}$ explores more terrain despite the oscillation (e.g., the Experiment 2 wall courses). Thus, $D_i^{\mathbf{S}}$ sometimes allows a system to discover a transition out of an oscillatory state and achieve the goal when $D_i^{\mathbf{R}^{\mathrm{f}}}$ cannot. This exploratory behavior also means that $D_i^{\mathbf{S}}$ risks wasting fuel and time if such a transition is never found. The observed trends extend to the V-shaped course (Experiment 3), with the exception of the $D_i^{\mathbf{R}^{\mathrm{f}}}$ translation-based-forgetting cylindrical planner. In the latter case, one $D_i^{\mathbf{S}}$ run is terminated after oscillating eight times, whereas similar behavior in the $D_i^{\mathbf{R}^{\mathrm{f}}}$ system resolves itself before eight oscillations. An interesting difference between the two distance metrics is that, in the absence of obstacles, $D_i^{\mathbf{R}^{\mathrm{f}}}$ induces the robot to move directly toward the goal, whereas $D_i^{\mathbf{S}}$ tends to produce navigation along "hook"-shaped trajectories. This behavior can be observed in all experiments. The extra distance traveled by systems using $D_i^{\mathbf{S}}$ is the primary reason they tend to achieve the goal more slowly than systems using $D_i^{\mathbf{R}^{\mathrm{f}}}$.

All of the observed differences are explained by the characteristics of the two metrics. $D_i^{\mathbf{S}}$ defines the distance between vertically or horizontally neighboring nodes as 1 and diagonally neighboring nodes as $\sqrt{2}$. The combined distance required to move horizontally from $\mathbf{O}^{\Gamma}_{n,m}$ to $\mathbf{O}^{\Gamma}_{n,j}$ and then vertically to $\mathbf{O}^{\Gamma}_{k,j}$ is equal to the distance required to move vertically from $\mathbf{O}^{\Gamma}_{n,m}$ to $\mathbf{O}^{\Gamma}_{k,m}$ and then horizontally to $\mathbf{O}^{\Gamma}_{k,j}$. In contrast, $D_i^{\mathbf{R}^{\mathrm{f}}}$ represents distance as the projected length of ground surface between pixel centers in $\mathbf{S}$, and the distance required to move between two vertically, horizontally, or diagonally neighboring nodes increases as grid row $n$ approaches the horizon. Thus, if $n > k$ (i.e., $n$ is closer to the bottom of the FOV than $k$), then the combined distance required to move horizontally from $\mathbf{O}^{\Gamma}_{n,m}$ to $\mathbf{O}^{\Gamma}_{n,j}$ and then vertically to $\mathbf{O}^{\Gamma}_{k,j}$ will be less than the combined distance required to move vertically from $\mathbf{O}^{\Gamma}_{n,m}$ to $\mathbf{O}^{\Gamma}_{k,m}$ and then horizontally to $\mathbf{O}^{\Gamma}_{k,j}$ (see Figure 21). This is because if $n > k$, then the distance between $\mathbf{O}^{\Gamma}_{n,m}$ and $\mathbf{O}^{\Gamma}_{n,j}$ is less than the distance between $\mathbf{O}^{\Gamma}_{k,m}$ and $\mathbf{O}^{\Gamma}_{k,j}$.

Given constant force values, $D_i^{\mathbf{R}^{\mathrm{f}}}$ will cause a path to perform as much horizontal movement in the bottom of the FOV (i.e., the near field) as possible. Thus, $D_i^{\mathbf{R}^{\mathrm{f}}}$ encourages a system to steer toward the goal immediately, whereas $D_i^{\mathbf{S}}$ allows turning movement to be postponed until later. This is illustrated in Figure 22. The $D_i^{\mathbf{S}}$ hook-shaped, real-world robot trajectories (shown in Section 7) are a result of this belated steering behavior, as is the propensity for increased exploration in situations that cause oscillation.

$D_i^{\mathbf{R}^{\mathrm{f}}}$ is relatively susceptible to translational oscillation because arc distance in $\mathbf{O}^{\Gamma}$ increases as a function of image height. Small changes in far-field information, including the relative position of the goal, can trigger large corrections in near-field portions of the path. This is observed on the 10-m wall course (Experiment 2a; Figure 13). Consider the case in which a large obstacle exists between the robot and the goal (Figure 23). The initial rotation away from the obstacle reinforces the decision to move in a particular direction by decreasing the optimal path length. As the robot moves, the relative location of $\mathbf{C}^{\Gamma}_{\mathbf{g}}$ moves in the opposite direction of travel. Eventually, $D_i^{\mathbf{R}^{\mathrm{f}}}$ induces a near-field correction and the robot changes direction. Assuming a unit force associated with nonobstacle portions of $\mathbf{C}$, this is guaranteed to repeat whenever the distance added to the path in the far field (due to the relative movement of the goal vs. the robot) is greater than the near-field distance required to change direction. Although $D_i^{\mathbf{S}}$ is susceptible to oscillation given a large enough obstacle (e.g., the 50-m wall in Experiment 2b), it allows movement to continue farther along the obstacle before a direction reversal occurs. This is because near-field movement involves the same amount of distance as far-field movement, and a simple rotation can substantially change the length of a path. The apparent location of $\mathbf{C}^{\Gamma}_{\mathbf{g}}$ must approach the opposite edge of the obstacle before a direction reversal occurs.

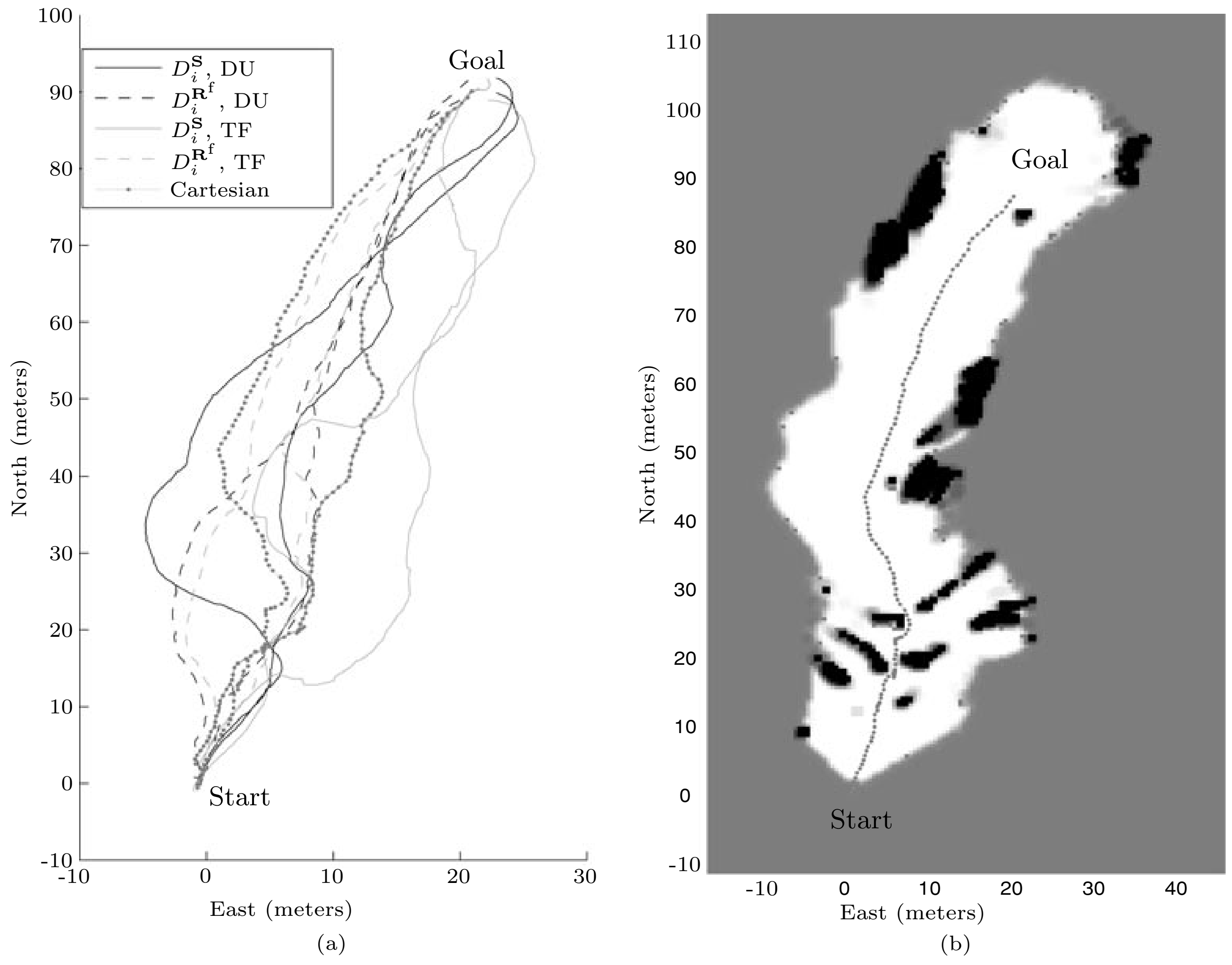


**Figure 17.** Experiment 4 GPS paths. (a) Hybrid hierarchical runs using $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^f}$ are solid or dashed, respectively, runs using depth-based updating or translation-based forgetting are black or gray, respectively, and all use $F_i^d$. Runs from the Cartesian hierarchical planner are dotted gray. (b) The Cartesian hierarchical planner's first run superimposed on the corresponding global occupancy grid. Low to high force values are displayed in white to black, respectively.

**Table IV.** Experiment 4 run times in seconds.

| System | Runs 1–2 | | Mean |
|---|---|---|---|
| | Hybrid hierarchical planner | | |
| DU $F_i^d$ $D_i^{\mathbf{S}}$ | 122.4 | 116.7 | 119.55 |
| DU $F_i^d$ $D_i^{\mathbf{R}^f}$ | 104.2 | 106.1 | 105.15 |
| TF $F_i^d$ $D_i^{\mathbf{S}}$ | 120.9 | 132.4 | 126.65 |
| TF $F_i^d$ $D_i^{\mathbf{R}^f}$ | 102.3 | 103.6 | 102.95 |
| | Cartesian hierarchical planner | | |
| Cartesian | 140.9 | 121.4 | 131.15 |

### 8.3. Cylindrical Updating: Translation-Based Forgetting, Depth-Based Updating

Translation-based forgetting decreases the force of an obstacle as a function of robotic displacement, whereas depth-based updating explicitly tracks obstacle movement relative to the robot. The former is tested in conjunction with all cylindrical and hybrid planners, whereas the latter is tested only with systems using the $F_i^d$ force metric. Translation-based forgetting slightly outperforms depth updating in situations without translational oscillation, but depth updating is more robust to translational oscillation.

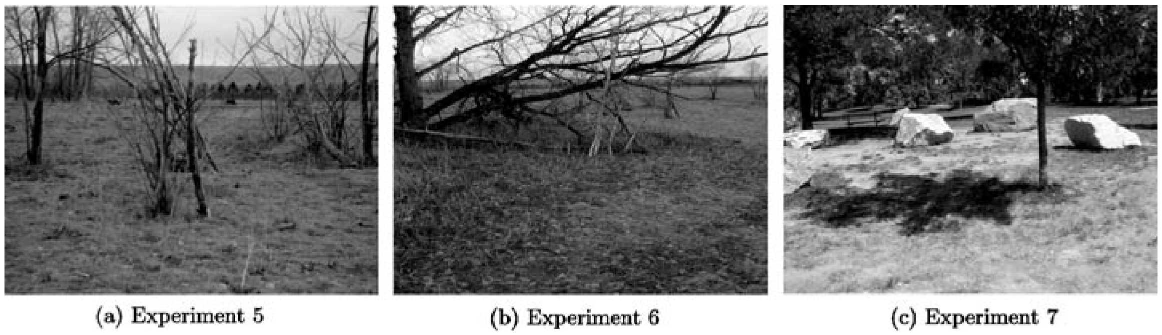


**Figure 18.** Photos of Experiments 5, 6, and 7 from the goal, start, and start, respectively.

On the 10-m wall course [Experiment 2a, Figure 13(a)], the $F_i^{\mathrm{d}} D_i^{\mathbf{S}}$ cylindrical planner experiences translational oscillation and systems using depth-based updating always escape, whereas systems using translation-based forgetting escape only twice. These trends extend to the V-shaped course [Experiment 3, Figure 16(a), Table III], where translation-based forgetting experiences more severe oscillation than depth-based updating.

With translation-based forgetting, the robot will always forget an obstacle of a particular force $\mathbf{O}_{n,m}$ after moving a predetermined distance $c_{\mathrm{dst}}$ at a constant speed. $c_{\mathrm{dst}}$ is calculated by recursively expanding Eq. (9), assuming it is used at a constant rate $c_{\mathrm{rt}}$:

$$1 \leq \mathbf{O}_{n,m} \prod_{u=1}^{\left\lceil \frac{c_{\mathrm{dst}} c_{\mathrm{rt}}}{\mathrm{speed}} \right\rceil} \frac{c_{\mathrm{fgt}} - \frac{\mathrm{speed}}{c_{\mathrm{rt}}}}{c_{\mathrm{fgt}}},$$

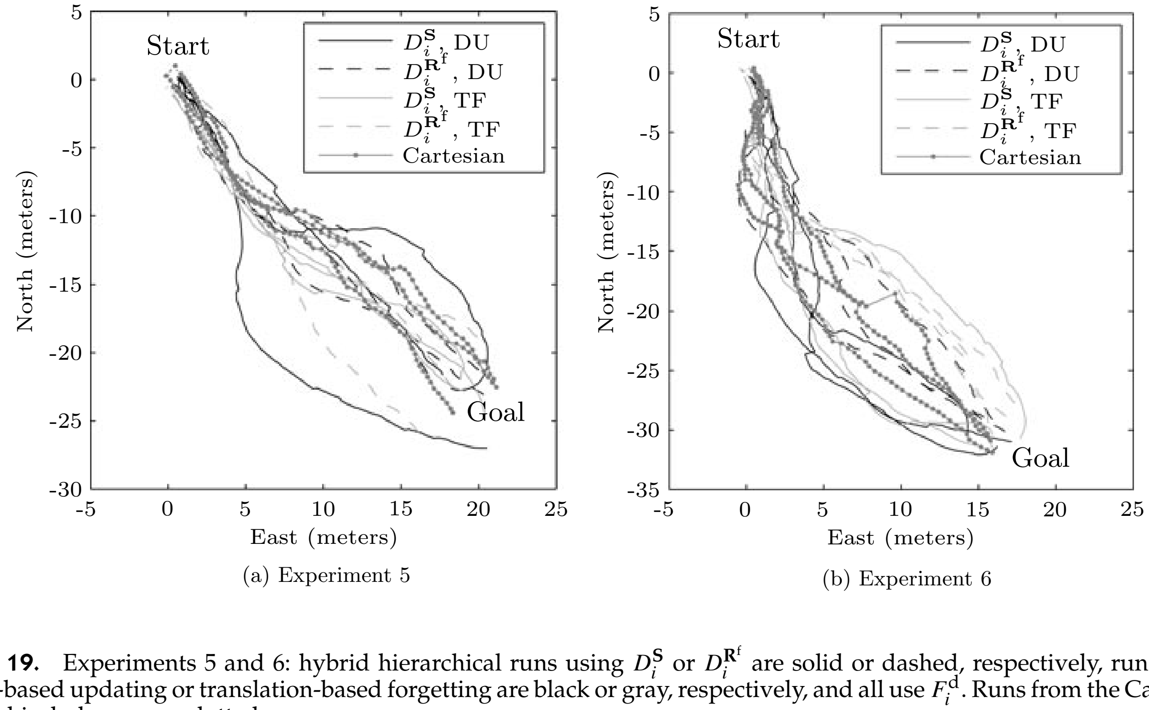


**Figure 19.** Experiments 5 and 6: hybrid hierarchical runs using $D_i^{\mathbf{S}}$ or $D_i^{\mathbf{R}^{\mathrm{f}}}$ are solid or dashed, respectively, runs using depth-based updating or translation-based forgetting are black or gray, respectively, and all use $F_i^{\mathrm{d}}$. Runs from the Cartesian hierarchical planner are dotted gray.

**Table V.** Experiment 5 run times in seconds.

| System | Runs 1–3 | | | Mean |
|---|---|---|---|---|
| | Hybrid hierarchical planner | | | |
| DU $F_i^{\mathrm{d}}\ D_i^{\mathbf{S}}$ | 35.99 | 48.54 | 47.03 | 43.85 |
| DU $F_i^{\mathrm{d}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 37.81 | 39.99 | 39.55 | 39.12 |
| TF $F_i^{\mathrm{d}}\ D_i^{\mathbf{S}}$ | 63.36 | 41.05 | 36.78 | 47.06 |
| TF $F_i^{\mathrm{d}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 37.38 | 35.48 | 36.26 | 36.37 |
| | Cartesian hierarchical planner | | | |
| Cartesian | 40.40 | 41.56 | 39.61 | 40.52 |

**Table VI.** Experiment 6 run times in seconds.

| System | Runs 1–3 | | | Mean |
|---|---|---|---|---|
| | Hybrid hierarchical planner | | | |
| DU $F_i^{\mathrm{d}}\ D_i^{\mathbf{S}}$ | 53.64 | 48.65 | 43.96 | 48.75 |
| DU $F_i^{\mathrm{d}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 53.24 | 46.69 | 42.76 | 47.56 |
| TF $F_i^{\mathrm{d}}\ D_i^{\mathbf{S}}$ | 44.39 | 45.15 | 48.91 | 46.15 |
| TF $F_i^{\mathrm{d}}\ D_i^{\mathbf{R}^{\mathrm{f}}}$ | 41.56 | 44.10 | 45.46 | 43.71 |
| | Cartesian hierarchical planner | | | |
| Cartesian | 47.77 | 48.36 | 43.64 | 46.59 |

where speed$/c_{\mathrm{rt}}$ is the distance traveled between calls to Eq. (9). The oscillation amplitude can be tuned by adjusting $c_{\mathrm{fgt}}$, $c_{\mathrm{rt}}$, and speed; however, it will be constant given a parameter set. If the robot encounters an obstacle longer than this distance, then the robot will turn back toward the goal before it has circumvented the obstacle. Oscillation occurs because the robot must turn around before rediscovering the forgotten portion of the obstacle. After the turn, it requires less graph distance to reach the goal by traveling in the new direction—especially in the case of $D_i^{\mathbf{S}}$. Depending on the specific system parameters and apparent location of $\mathbf{C}_{\mathbf{g}}^{\Gamma}$, translational oscillation may be induced by the distance metric before it is caused by translation-based forgetting. Force values are stored at constant headings in the cylinder; therefore, after turning away from an obstacle, the robot will avoid turning back toward its original heading until the obstacle is forgotten. Large values of $c_{\mathrm{fgt}}$ and slow frame rates tend to cause the robot to overavoid small obstacles (i.e., move unnecessarily far around them). The forgetting parameters must be tuned to elicit the desired balance between oscillation and obstacle overavoidance.

In Experiment 7 (the natural terrain course where each system is tested 10 times), depth-based updating has a quicker mean run time than translation-based forgetting—although not by a significant amount.

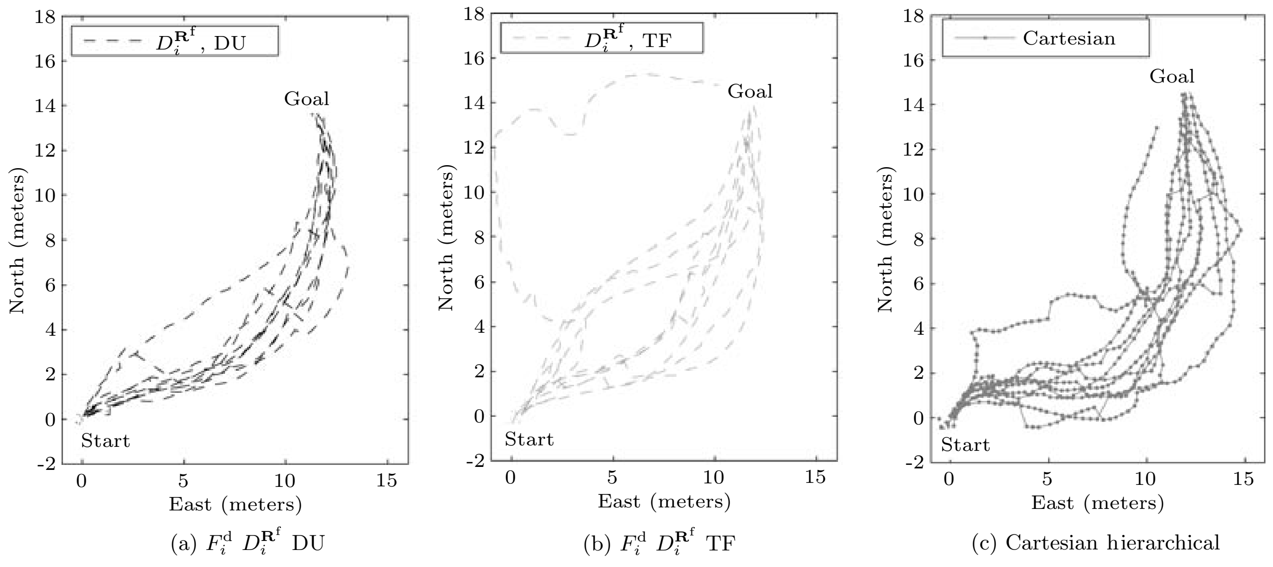


**Figure 20.** GPS paths of the hybrid and Cartesian hierarchical planners in Experiment 7.

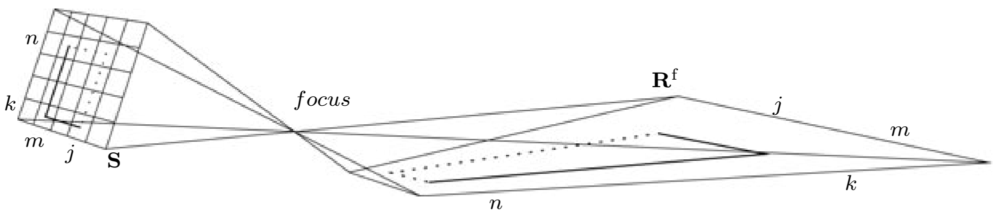


**Figure 21.** A projection of two paths from image space **S** onto a flat ground plane $\mathbf{R}^f$. The distance along $\mathbf{R}^f$ is used for the $D_i^{\mathbf{R}^f}$ metric. Although both paths have the same number of vertical and horizontal edges, the dotted path is shorter due to perspective. The solid and dashed paths appear reversed on the image plane due to projection through the focus.

(Using Student's *t*-test with the null hypothesis that the two systems' run times are sampled from the same normal distribution gives a *p*-value of 0.26.) We attribute this (nonsignificant) performance difference to the slight overestimation of obstacle size by the depth-based updating system. If the cylindrical planner is used as the local planning subsystem of a hierarchical planner, then forgetting enabled oscillation can be avoided by using appropriately close subgoals. However, depth updating has no parameters to tune, is less susceptible to translational oscillation, and will never overestimate the size of an obstacle—although its use is limited to systems with depth data.

## 8.4. Planning Systems: Image, Cylindrical, Hierarchical

The wall and V courses (Experiments 2 and 3) show that the basic image planner is susceptible to rotational oscillation whenever an obstacle wider than the FOV obscures the goal. The cylindrical planner is immune to rotational oscillation; hence, it is more robust than the basic image planner. Although the latter can succeed in a subset of environments navigable by the former (i.e., sparsely populated environments containing obstacles narrower than the FOV), it is still advisable to use the cylindrical planner whenever possible. Both the image planner and the cylindrical planner exhibit translational oscillation when a wide obstacle obscures the goal. As discussed in Section 8.2, this is manifested differently depending on the distance metric being used ($D_i^{\mathbf{S}}$ is less susceptible than $D_i^{\mathbf{R}^f}$). A separate type of translational oscillation, discussed in Section 8.3, is enabled by the translation-based forgetting cylindrical updating method.

All three forms of oscillation observed in our experiments are triggered when a large obstacle exists directly between the robot and the goal. This reinforces our belief that image planning is more suited to near-field (local) planning than far-field (global) planning. To provide quick navigation commands for obstacle avoidance, any local planner must limit the

**Table VII.** Experiment 7 run times in seconds.

| System | Runs 1–10 | | | | | Mean | Std. |
|---|---|---|---|---|---|---|---|
| | Hybrid hierarchical planner | | | | | | |
| DU $F_i^d$ $D_i^{\mathbf{R}^f}$ | 27.41 | 26.35 | 25.59 | 29.78 | 32.47 | | |
| | 31.40 | 27.38 | 28.84 | 28.82 | 26.59 | 28.46 | 2.24 |
| TF $F_i^d$ $D_i^{\mathbf{R}^f}$ | 24.40 | 27.33 | 35.89 | 25.59 | 42.84 | | |
| | 44.58 | 35.69 | 25.96 | 26.07 | 25.53 | 31.39 | 7.71 |
| | Cartesian hierarchical planner | | | | | | |
| Cartesian | 37.38 | 30.45 | 40.83 | 30.53 | 33.15 | | |
| | 34.78 | 42.55 | 51.09 | 34.23 | 57.64 | 39.26 | 9.01 |

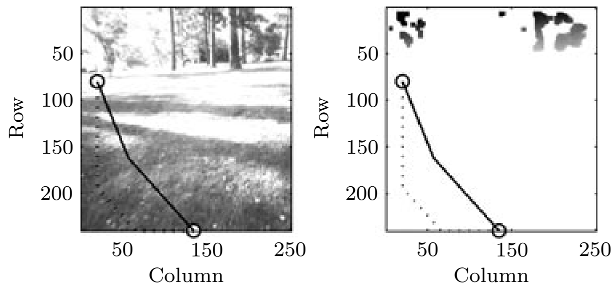


**Figure 22.** Image space paths using the $D_i^{\mathbf{S}}$ and $D_i^{\mathbf{R}^f}$ distance metrics, solid and dotted, respectively, through a scene image (left) and corresponding occupancy grid (right). In an empty field, $D_i^{\mathbf{S}}$ causes the path to change direction farther away from the robot than $D_i^{\mathbf{R}^f}$.

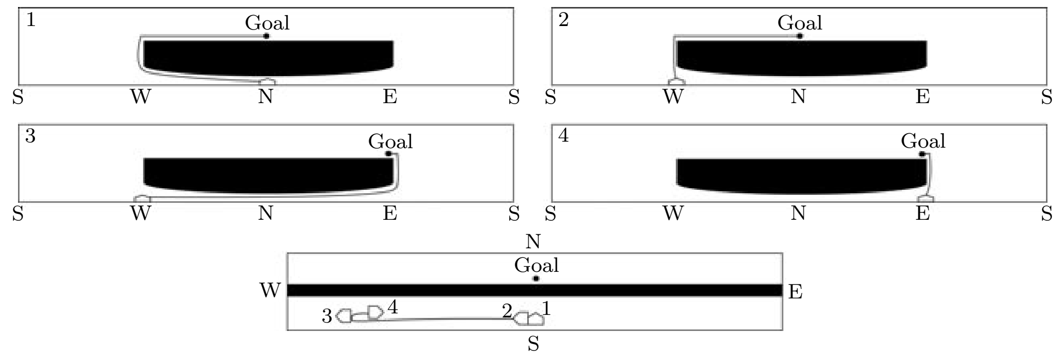


**Figure 23.** Translational oscillation caused by a large obstacle. Panes 1–4 show the paths through the cylinder that are generated at the respective locations in the bottom diagram. (1) The robot decides to move west. (2) Directional change reinforces the decision to go west. (2–3) Westward translation causes the goal to move east, relative to the robot. (3) It becomes cheaper to go east. (4) Directional change reinforces the decision to go east.

size of its world view. Therefore, any local planner is vulnerable to oscillation from obstacles larger than its world model. When the cylindrical planner is used as a local subsystem of a hybrid hierarchical planner, the global planner can maximize the effectiveness of the local planner by feeding it subgoals closer than its minimum oscillation amplitude. This ensures that pathological oscillation will never occur. Further motivation for using the cylindrical planner locally is provided by the fact that, due to perspective, the accuracy of an image-based representation of the world is maximized in the near field.

The natural terrain courses (Experiments 4–7) validate our hypothesis that the hybrid hierarchical planner will outperform the Cartesian hierarchical planner—but only for the specific planners using the $D_i^{\mathbf{R}^f}$ distance metric. Results for planners using the $D_i^{\mathbf{S}}$ metric are inconclusive. This illustrates the importance of choosing the correct representation of distance when planning in a nonrectilinear projection of the world. With respect to Experiment 7, using Student's *t*-test with the null hypothesis that two systems' run times are sampled from the same normal distribution, the $D_i^{\mathbf{R}^f}$ depth-based updating hybrid hierarchical planner vs. the Cartesian hierarchical planner gives a *p*-value of 0.0017, whereas the $D_i^{\mathbf{R}^f}$ translation-based forgetting hybrid hierarchical planner vs. the Cartesian hierarchical planner gives a *p*-value of 0.050. It is important to note that the hybrid and baseline hierarchical planners use an identical global planner, maintain local maps with similar memory requirements, and achieve comparable local frame rates. We attribute the positive performance of the $D_i^{\mathbf{R}^f}$ hybrid hierarchical planner to the local cylindrical planner's high-resolution, near-field view. This enables accurate robot localization vs. objects in the FOV and gives the local cylindrical planner the ability to disambiguate, track, and narrowly avoid obstacles that are distorted by the local Cartesian planner.

## 9. CONCLUSION

Image space path planning is a planning technique used to discover paths through the image output of a grid-based sensor. We present an image space planning technique for local path planning in unknown unstructured outdoor environments. The basic method involves creating an occupancy grid map directly from image data and then using a graph-search algorithm to find a minimum-cost path through a graph representation of the cost map. Once found, the graph path is used for navigation in the real world. Our method differentiates between the cost values that are stored in the cost map, the cost values assigned to graph edges, and the distances between cost-map elements. Borrowing terms derived from a physical analogy, we label these as force, work, and distance, respectively

We propose three image-based planning systems and perform experimental evaluation with a real robot on eight courses. We find that our method of minimizing work, given explicitly separated notions of force and distance, is robust to various force and distance metrics. Evaluation is performed with two different force metrics ($F_i^{\mathrm{d}}$ and $F_i^{\mathrm{c}}$) and two different distance metrics ($D_i^{\mathbf{S}}$ and $D_i^{\mathbf{R}^{\mathrm{f}}}$).

The greater usable range of color data, as compared to stereo disparity, gives our naive force from color metric $F_i^{\mathrm{c}}$ a reactive advantage over our force from disparity metric $F_i^{\mathrm{d}}$. Although the former is used only as a proof of concept, it demonstrates that accurate force from color data can improve the performance of an image space planning system by enabling it to detect obstacles sooner.

The distance metric based on the $L^2$ norm in image space ($D_i^{\mathbf{S}}$) postpones trajectory adjustments into the far field, whereas the flat world distance metric ($D_i^{\mathbf{R}^{\mathrm{f}}}$) prefers trajectory adjustments in the near field. Consequently, $D_i^{\mathbf{R}^{\mathrm{f}}}$ causes the robot to follow straighter paths than $D_i^{\mathbf{S}}$, whereas $D_i^{\mathbf{S}}$ has a greater oscillation amplitude than $D_i^{\mathbf{R}^{\mathrm{f}}}$. (Oscillation amplitude is the distance along the edge of a goal-blocking obstacle that the robot will travel before turning around.) A relatively large oscillation amplitude gives $D_i^{\mathbf{S}}$ an advantage in environments containing obstacles with radii larger than the oscillation amplitude of $D_i^{\mathbf{R}^{\mathrm{f}}}$ (but smaller than the oscillation amplitude of $D_i^{\mathbf{S}}$).

The basic image planner is susceptible to two types of oscillation: rotational oscillation caused by the goal moving outside the FOV and translational oscillation caused by the apparent location of the image space goal shifting as a function of robotic movement. The simulated $2\pi$ radian FOV maintained in the cylindrical planner eliminates rotational oscillation; however, it adds the complication of updating data that have moved outside the camera's FOV. Despite the cylindrical planner's immunity to rotational oscillation, it is still susceptible to translational oscillation. Of the two cylindrical updating techniques evaluated, translation-based forgetting can be used in conjunction with any type of sensor input. However, it must be tuned to balance an additional form of translational oscillation failure vs. moving unnecessarily far around small obstacles. In contrast, depth-based updating does not require tuning, does not enable additional oscillation, and will never overestimate the length of an obstacle—but requires depth data.

The hybrid hierarchical planner avoids all forms of oscillation by giving the cylindrical planner appropriate subgoals from a global Cartesian planner. This framework incorporates the strengths of both planning paradigms, allowing for better overall path planning. Hybrid hierarchical planners using the $D_i^{\mathbf{R}^{\mathrm{f}}}$ distance metric are able to outperform a double 2D top-down Cartesian planner on four real-world courses.

Image space planning provides a convenient framework for maintaining a high-resolution view of near-field terrain by performing path search at the maximum resolution of the image sensor. Assuming an image sensor of fixed resolution, other advantages include guarantees on search time and a stable memory footprint. A natural application for image space planning is as the local planning component of a hybrid hierarchical planner. This combination provides the high-resolution, near-field view, precise robot vs. obstacle localization, and dependable frame rate of a local image space planner with the translational memory and fixed-resolution, far-field view of a global 2D Cartesian planner. Overall, we find that image space planning is a viable technique for discovering and following paths through unknown unstructured outdoor environments.

## ACKNOWLEDGMENTS

This work has been supported by the DARPA LAGR program (Department of Defense Air Force Research Laboratory award no. FA8650-07-C-7702) and the National Science Foundation (NSF IIS-0535269). We would like to thank the members of the Intelligence in Action Laboratory at the University of Colorado at Boulder, our fellow participants in the LAGR program, and the LAGR government team. We would also like to thank the anonymous experts who reviewed this work; their comments and suggestions have made it a much better paper.

## REFERENCES

Argyros, A., Bekris, K. E., Orphanoudakis, S. C., & Kavraki, L. E. (2005). Robot homing by exploiting panoramic vision. Autonomous Robots, 19, 7–25.

Borenstein, J., & Koren, Y. (1991). The vector field histogram—Fast obstacle avoidance for mobile robots. IEEE Journal of Robotics and Automation, 7, 278–288.

Carsten, J., Rankin, A., Ferguson, D., & Stentz, A. (2007). Global path planning on board the Mars exploration rovers. In IEEE Aerospace Conference (pp. 1–11). IEEE.

Chen, S. (1990). A spherical model for navigation and spatial reasoning. In IEEE International Conference on Robotics and Automation, Cincinnati, OH (vol. 2, pp. 776–781). IEEE.

Cowan, N., Weingarten, I., & Koditschek, D. (2002). Visual servoing via navigation functions. IEEE Transactions on Robotics and Automation, 18, 521–533.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. In Numerical mathematics (vol. 1, pp. 269–271).

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. IEEE Computer, 22(6), 4657.

Feddema, J., & Mitchell, O. (1989). Vision-guided servoing with feature-based trajectory generation. IEEE Transactions on Robotics and Automation, 5, 691–700.

Ferguson, D., & Stentz, A. (2006). Using interpolation to improve path planning: The field D* algorithm. Journal of Field Robotics, 23, 79–101.

Gat, E., Slack, M., Miller, D. P., & Firby, R. J. (1990). Path planning and execution monitoring for a planetary rover. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'90), Cincinnati, OH (vol. 1, pp. 20–25). IEEE.

Gaussier, P., C. Joulain, S. Z., Blanquet, J. P., & Revel, A. (1997). Visual navigation in an open environment without map. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97), Grenoble, France (vol. 2, pp. 545–550).

Goldberg, S. B., Maimone, M. W., & Matthies, L. (2002). Stereo vision and rover navigation software for planetory exploration. In IEEE Aerospace Conference Proceedings, Big Sky, MT (vol. 5, pp. 2025–2036). IEEE.

Grudic, G., Mulligan, J., Otte, M., & Bates, A. (2007). Online learning of multiple perceptual models for navigation in unknown terrain. In International Conference on Field and Service Robotics (FSR'07), Chamonix, France (pp. 411–420). Berlin: Springer.

Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on System Science and Cybernetics, 4(2), 100–107.

Herman, M. (1986). Fast, three-dimensional, collision-free motion planning. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'86), San Francisco, CA (pp. 1056–1063). IEEE.

Hong, J., Tan, X., Pinette, B., Weiss, R., & Riseman, E. M. (1991). Image-based homing. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'91), New York (pp. 620–625). IEEE.

Hutchinson, S., Hager, G. D., & Corke, P. I. (1996). A tutorial on visual servo control. IEEE Transactions on Robotics and Automation, 12, 651–670.

Jochem, T. M., Pomerleau, D. A., & Thorpe, C. E. (1995). Vision-based neural network road and intersection detection and traversal. In Proceedings IEEE Conference Intelligent Robots and Systems, Pittsburgh, PA (vol. 3, pp. 344–349). IEEE.

Kelly, A. (1994). Adaptive perception for autonomous vehicles (Tech. Rep. CMU-RI-TR-94-18). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Kelly, A. (2000). Mobile robot localization from large-scale appearance mosaics. International Journal of Robotics Research, 19, 1104–1125.

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

Koenig, S., & Likhachev, M. (2002). Improved fast replanning for robot navigation in unknown terrain. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'02), Washington, DC (vol. 1, pp. 968–975). IEEE.

Kolski, S., Ferguson, D., Bellino, M., & Siegwart, R. (2006). Autonomous driving in structured and unstructured environments. In IEEE Intelligent Vehicles Symposium, Lausanne, Switzerland, and Pittsburgh, PA (pp. 558–563). IEEE.

Koren, Y., & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'91), Sacramento, CA (vol. 2, pp. 1398–1404). IEEE.

Krishnaswamy, G., & Stentz, A. (1995). Resolution independent grid-based path planning (Tech. Rep. CMU-RI-TR-95-08). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Krogh, B. H., & Thorpe, C. E. (1986). Integrated path planning and dynamic steering control for autonomous vehicles. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'86), San Francisco, CA (pp. 1664–1669). IEEE.

Mateus, D., Avina, G., & Devy, M. (2005). Robot visual navigation in semi-structured outdoor environments. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'05), Barcelona, Spain (pp. 4691–4696). IEEE.

Matsumoto, Y., Sakai, K., Inaba, M., & Inoue, H. (2000). View-based approach to robot navigation. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00), Takamatsu, Japan (vol. 3, pp. 1702–1708). Piscataway, NJ: IEEE.

Matthies, L. (1992). Passive stereo range imaging for semi-autonomous land navigation. Journal of Robotic Systems, 9, 787–816.

Minguez, J., & Montano, L. (2004). Nearness diagram (ND) collision avoidance in troublesome scenarios. IEEE Transactions on Robotics and Automation, 20, 45–59.

Murray, D., & Jennings, C. (1998). Stereo vision based mapping and navigation for mobile robots. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'97), Albuquerque, NM (pp. 1694–1699). IEEE.

Murray, D., & Little, J. J. (2000). Using real-time stereo vision for mobile robot navigation. Autonomous Robots, 8(2), 161–171.

Ollis, M., Huang, W. H., Happold, M., & Stancil, B. A. (2008). Image-based path planning for outdoor mobile robots. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'08), Pasadena, CA. IEEE.

Otte, M. (2007). Path planning in image space for the autonomous navigation of unmanned vehicles in unstructured outdoor environments. M.S. dissertation, University of Colorado, Boulder.

Otte, M., Richardson, S., Mulligan, J., & Grudic, G. (2007). Local path planning in image space for autonomous robot navigation in unstructured environments. In International Conference on Intelligent Robots and

Systems (IROS'07), San Diego, CA (pp. 2819–2826). IEEE.

Pomerleau, D. A. (1989). ALVINN: An autonomous land vehicle in a neural network (Tech. Rep. CMU-CS-89-107). Pittsburgh, PA: Department of Computer Sciences, Carnegie Mellon University.

Procopio, M., Mulligan, J., & Grudic, G. (2007). Long-term learning using multiple models for outdoor autonomous robot navigation. In International Conference on Intelligent Robots and Systems (IROS'07), San Diego, CA (pp. 3158–3165). IEEE.

Sabe, K., Fukuchi, M., Gutmann, J.-S., Ohashi, T., Kawamoto, K., & Yoshigahara, T. (2004). Obstacle avoidance and path planning for humanoid robots using stereo vision. In Proceedings IEEE International Conference on Robotics and Automation (ICRA'04) (vol. 1, pp. 592–597). IEEE.

Song, D., Lee, H. N., Yi, J., & Levandowski, A. (2007). Vision-based motion planning for an autonomous motorcycle on ill-structured roads. Autonomous Robots, 23, 197–212.

Sonka, M., Hlavac, V., & Boyle, R. (2008). Image processing, analysis and machine vision, 3rd ed. Toronto: Thompson Learning.

Stentz, A. (1995). The focussed D* algorithm for real-time replanning. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Montréal, Canada (pp. 1652–1659). Morgan Kaufmann.

Sugiyama, M., Kawano, Y., Niizuma, M., Takagaki, M., Tomizawa, M., & Degawa, S. (1994). Navigation system for an autonomous vehicle with hierarchical map and planner. In Proceedings of the Intelligent Vehicles '94 Symposium, Paris (p. 5055). Piscataway, NJ: IEEE.

Thorpe, C., Herbert, M. H., Kanade, T., & Shafer, S. A. (1988). Vision and navigation for the Carnegie-Mellon Navlab. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10, 362–372.

Tsugawa, S., Yatabe, T., Hirose, T., & Matsumoto, S. (1979). An automobile with artificial intelligence. In Proceedings Sixth International Joint Conference on Artificial Intelligence, Tokyo (pp. 893–895). Kaufmann.

van den Mark, W., Groen, F., & van den Heuvel, J. C. (2001). Stereo based navigation in unstructured environments. In IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary (vol. 3, pp. 2038–2043).

Vidal, R., Shakernia, O., & Sastry, S. (2003). Formation control of nonholonomic mobile robots omnidirectional visual servoing and motion segmentation. In Proceedings IEEE Conference on Robotics and Automation, Taipei, Taiwan (pp. 584–589). IEEE.

Winters, N., Gaspar, J., Lacey, G., & Santos-Victor, J. (2000). Omni-directional vision for robot navigation. In IEEE Workshop on Omnidirectional Vision (OMNIVIS'00), Hilton Head, SC (pp. 21–28). Los Alamitos, CA: IEEE Computer Society.

Zhang, H., & Ostrowski, J. (2002). Visual motion planning for mobile robots. IEEE Transactions on Robotics and Automation, 18, 199–208.