New Techniques for Path Planning in Image Space

Michael W. Otte, Dan Knights, Joseph J. Pfeiffer III, Jane Mulligan, and Greg Grudic

University of Colorado at Boulder
Technical Report CU-CS-1052-09

New Techniques for Path Planning in Image Space

Michael W. Otte **Dan Knights** **Joseph J. Pfeiffer III**

Jane Mulligan **Greg Grudic***

Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309-0430

Abstract

We propose two new methods for performing local path search directly in the image space of a camera sensor. The first technique utilizes a high-resolution local image space subsystem to determine the farthest point along a global Cartesian path that can be reached along an unobstructed heading, thereby short-cutting as much of the global path as possible. To the best of our knowledge, this is the first image space planning technique that is both suitable for navigation in unstructured environments and does not use a graph-search algorithm in the image space subsystem. The second technique that we propose improves the local image space subsystem of a hierarchical image/Cartesian space planner by determining when it is safe to translate at full speed. Both techniques are implemented on an autonomous robot and experimentally evaluated against a hierarchical top-down Cartesian planner and a previously proposed hierarchical image/Cartesian space planner. Each system is tested three times on three different course layouts. The first proposed method performs comparably to existing systems when evaluated on the criteria of path length and total runtime. The second technique consistently outperforms the other three systems with respect to total runtime.

1 Introduction

Path planning is the task of finding a sequence of actions that will allow a robot to achieve a goal state given a representation of the environment and a predefined set of state transition constraints. In the context of robotic rovers, the environmental representation often includes a map and a set of sensor inputs. The map may be predefined and/or modified as the robot explores the environment. It is common to store the map in a discretized Cartesian model of the world, although many other representations have been proposed ([1–8]). The sensor input set is determined by the sensors available to the system and may include anything

*This work was supported by DOD AFRL award no. FA8650-07-C-7702 (DARPA LAGR) and NSF IIS- 0535269.

from measurement readings (GPS, velocimeter, thermometer, manometer, etc.) to image sensor data (ladar, sonar, camera, etc.).

There has recently been an interest in path planning directly in the array-based projections of the world that are captured by an image sensor ([9, 10]). This type of planning is referred to as *image space planning*, and the space in which the path search occurs is called *image space*. There is typically a one-to-one mapping between image space array locations (i.e. pixels) and image space occupancy grid locations. Image space planning contrasts with the more traditional practice of projecting environmental data into a separate Cartesian space map prior to path-planning ([11–15]).

Image space planning avoids the localization errors and systematic information distortions caused by projecting data into a model with resolution and/or coordinate system different than that of the sensor. However, image space planning still assumes the existence of a transformation function between robot and image space coordinates. This can be achieved directly if distance data is available; however, it is also common to approximate the transformation by assuming the robot exists on a flat ground plane. In either case, the relative positions of the camera and robot are assumed to be known.

Image space planning is unique among other image-based approaches. View-Sequenced Route Representation ([4–6, 8, 16–18]) uses the relationship between a target image and the camera's current image to calculate servoing commands, while autonomous highway driving algorithms ([19–23]) utilize environmental features that are extracted from the image—such as lane markings, prior vehicle tire tracks, or footprints. These techniques make assumptions about the existence of predefined information or scene appearance that prohibit their use in unknown/unstructured terrain. In contrast, image space planning makes no assumptions about environmental appearance or the existence of other prior knowledge.

Image space planning is distinct from reactive local planning techniques that use 1D vectors of sensor data [24] or similar 1D histograms of global Cartesian map data [25] to determine safe directional headings. The height-distance dimension of 2D image-space maps allows image planners to find more complex paths than those available to reactive local planners.

To date, image space path planning in environments that are both unknown and unstructured has been accomplished by modifying existing graph-search algorithms to work in image space. For instance, path-planning directly in a panoramic cylinder of cost-from-disparity images to achieve high-resolution local planning [9], or path-planning within a cost-from-color image in order to find superior long-range paths [10]. The cost-from-color image is obtained by passing an RGB image pixel-wise through a learned safe-vs.-obstacle model of the environment. In both of these methods, the image space planner operates in conjunction with a more traditional Cartesian space planner (i.e. a 2D top-down map of the world). In [9] the Cartesian space planner acts as a global subsystem that feeds sub-goals to the local image space subsystem. In [10] the robot follows paths from the Cartesian space planner by default, incorporating waypoints from image space paths that are determined to be sufficiently good.

A hierarchical planner consists of two or more subsystems that operate in parallel, often using separate models of the environment ([4–8]). A common arrangement consists of a global

subsystem and a local subsystem. The global subsystem maintains a coarse representation of the entire environment, while the local subsystem models an immediate region of the environment in high resolution. The global and local subsystems are charged with finding a complete path to the actual goal and a near-field path to a subgoal, respectively; where the local subgoal is determined from the global path.

In this paper we propose two new techniques for merging image space and Cartesian space planning methods in a hierarchical planner. We refer to the two proposed methods as *Image Space Verification of Global Planning* (IVG) and *Fast Local Cylindrical Planning* (FLCP). IVG and FLCP both consist of a local image space subsystem and a global Cartesian space subsystem.

IVG differs from previous image space planning techniques that have been designed for use in unstructured environments in that the local image space subsystem does not use a graph-search algorithm. Instead, IVG projects the global path provided by the global Cartesian space subsystem into image space. Next, IVG uses high-resolution image space data to determine the furthest point along the global path that the robot can move toward safely and directly. IVG encourages a robot to avoid unnecessary detours in the near-field by short-cutting the global path whenever possible.

If the local image space subsystem finds that a global path is impassable, IVG propagates this information back to the global Cartesian space subsystem. If the entire Cartesian path is determined to be invalid by the image space subsystem, then a message is sent to the Cartesian space subsystem and a high cost is inserted into the global map at the beginning of the invalid path. This form of subsystem interaction has previously been used in hierarchical planners ([26–28]).

IVG is similar to the system described in [29], in which candidate movement trajectories are projected into image space and then evaluated for quality based on image data. The technique proposed in [29] evaluates a fixed sized set of possible trajectories and assumes the existence of scene information such as prior vehicle tire tracks, pedestrian footprints, or predefined cost from color information. In contrast, IVG evaluates trajectories determined by the dynamic set of currently visible global path coordinates and does not make scene-appearance assumptions.

IVG is also similar to reactive local planning ([30–32]) in that the robot may deviate from a high-level Cartesian path as a result of online sensor observations. The main difference between reactive planning and IVG is that the former causes the robot to modify the global Cartesian path when an immediate obstacle is detected, while the latter encourages the robot to modify the global path in the absence of obstacles.

FLCP is a modified version of the panoramic cylindrical image space system proposed in [9], except that the servoing function is modified to allow the robot to travel at full speed whenever the high resolution image space map is free of robot-blocking obstacles. In [9] the robot's speed is adjusted as a function of the contour of the local image space path—even if the path in front of the robot is obstacle free. This causes the robot to move unnecessarily slow in safe environments.

The implementation details of IVG and FLCP are presented in Section 2, and our robotic apparatus and system parameters are outlined in Section 3. In Section 4 we describe three experiments in which we compare the proposed systems against two other planning systems. A discussion of the results is given in Section 5, our conclusions are presented in Section 6, and directions for future work are outlined in Section 7.

2 Methodology

We now describe the implementation details of all four planning systems that are experimentally evaluated in Section 4. These include the two methods proposed in this paper, Image Space Verification of Global Planning (IVG) and Fast Local Cylindrical Planning (FLCP), as well as two comparison planning systems including the hierarchical image/Cartesian space method from [9] and a double top-down Cartesian-based system. We refer to the latter two systems as *Local Cylindrical Planning* (LCP) and *Double Top-Down Planning* (DTP), respectively.

All four planning systems are hierarchical with self-contained local and global subsystems. The two subsystems run in parallel, on separate processing cores, and communicate via a message-passing protocol. All four planners use identical global subsystems, with the exception that IVG can send lethal coordinates from the local subsystem to the global subsystem.

2.1 IVG Local Image Space Subsystem

The image-based subsystem of IVG attempts to find the farthest point along the global path that the robot can navigate toward directly. This allows the robot to short-cut unnecessary parts of the global path and reach the goal with as little movement as possible. 'Unnecessary' global path sections include detours around obstacles that have disappeared since the creation of the global path, as well as places where the coarse granularity of the global map has prohibited it from noticing a safe passage.

Let $\mathbf{S}$ represent the current image (i.e. output) of an image sensor. $\mathbf{S}$ is h pixels high and w pixels wide. Let $\mathbf{S}_{n,m}$ denote the image pixel that exists at row n and column m of $\mathbf{S}$. Note that $1 \leq n \leq h$ and $1 \leq m \leq w$. We adopt the convention that pixel $\mathbf{S}_{1,1}$ is located in the top row and left-most column of $\mathbf{S}$. Let $\mathbf{O}$ represent an image-space occupancy-grid cost-map that is derived from $\mathbf{S}$. Let $\mathbf{O}_{n,m}$ represent the value stored at row n and column m of $\mathbf{O}$. Note that there is a one-to-one and onto mapping from image pixels in $\mathbf{S}$ to occupancy grid locations in $\mathbf{O}$.

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

IVG can be used with any form of image sensor data, as long as useful cost information can be obtained from $\mathbf{S}$. In the context of this paper, $\mathbf{S}$ is defined to be a stereo disparity image and $\mathbf{O}$ is calculated using a simple heuristic cost function based on a similar heuristic

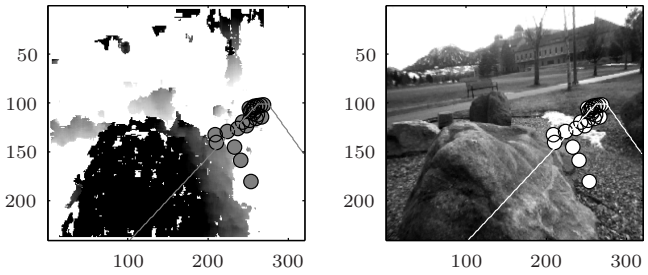

Figure 1: Image space occupancy grid $\mathbf{O}$, left, and the corresponding grayscale image, right. The quadrilateral outlined by the solid line is the image space projection of the uninhibited terrain surface required for safe navigation to the last coordinate of the global path. The quadrilateral displayed here is not safe because it contains an obstacle.

proposed in [9].

$$\mathbf{O}_{n,m} = 1 + c_{scale} \left| \mathbf{S}_{n,m} - \mathbf{S}^{flat}_{n,m} \right| \tag{2}$$

$\mathbf{S}^{flat}_{n,m}$ is the disparity image that would be returned from a horizontal flat ground plane, and c_{scale} is a scaling constant. Equation 2 has the desirable quality that cost increases as the environment deviates from a horizontal flat surface.

Assume that the Cartesian-based subsystem has found a path $\mathbf{P}$ from the robot's current location to the goal; where $\mathbf{P}$ is defined by l GPS coordinates. $\mathbf{P}$ is sent to the image-based subsystem in a message. Let $\mathbf{P}_i$ represent the ith GPS coordinate along the path. Note that $1 \leq i \leq l$. The most recent $\mathbf{P}$ is actively maintained in the image-based subsystem by periodically truncating it based on forward robotic movement. This is accomplished by removing points $\mathbf{P}_j$ for $1 \leq j \leq k$, given that $\mathbf{P}_k$ is the closest point along the path to the robot's current location.

The image space planning subsystem consists of three different states of operation: *ImagePlan*, *Scan*, and *Blocked*. State *ImagePlan* and State *Scan* utilize the subroutine *ApprovePoint*($\mathbf{P}_i$) to determine if it is safe to travel directly toward the path coordinate $\mathbf{P}_i$ given the information in $\mathbf{O}$. State *ImagePlan* is initialized by default and determines what visible point (if any) the robot should drive toward. State *Scan* is used if all of the currently visible coordinates $\mathbf{P}_i$ are determined to be unsafe. If the entire path $\mathbf{P}$ is found to be unsafe, then state *Blocked* is invoked. State *Scan* causes the robot to rotate toward unchecked $\mathbf{P}_i$, while State *Blocked* sends a message to the global Cartesian subsystem and waits for a new global path.

2.1.1 Subroutine *ApprovePoint*($\mathbf{P}_i$)

Let c_{width} be the actual width of the robot, and let c_{safety} be the minimum safety clearance allowed between the robot and an obstacle. The robot is assumed to exist on a flat plane for the purposes of projecting points between real-world coordinates and their corresponding image space locations.

ApprovePoint($\mathbf{P}_i$) operates by checking the image space projection of a swath of terrain

surface of width $(c_{width} + 2c_{safety})$ along the direct path from the robot's current location to $\mathbf{P}_i$ (see Figure 1). This represents the unoccupied space required for safe navigation to $\mathbf{P}_i$. Let the boundary of this area be defined by the four image space coordinates $[v_l, u_l]$, $[v_r, u_r]$, $[d_l, c_l]$, and $[d_r, c_r]$. Let $[v_l, u_l]$ and $[v_r, u_r]$ be associated with the left and right path boundaries at $\mathbf{P}_i$, respectively, and let $[d_l, c_l]$ and $[d_r, c_r]$ be associated with the left and right path boundaries at the robot's current location, respectively. If the orientation of the camera is fixed, relative to the robot, then $[d_l, c_l]$ and $[d_r, c_r]$ are constant. Together, the four points describe a quadrilateral in image space. Note that portions of the quadrilateral corresponding to places outside of the field-of-view will be located outside of $\mathbf{O}$.

For the sake of computational ease, the quadrilateral is redefined by making the top edge parallel to an image row.

$$v_l = v_r = \min(v_l, v_r) \tag{3}$$

Note that the area of the resulting quadrilateral is always greater than the original quadrilateral.

The interior of the quadrilateral is checked for cost values greater than c_{cost}, where c_{cost} is the per-grid maximum cost value that the robot is allowed to traverse. If such a value is found, then the subroutine returns $false$ because it is unsafe to travel directly toward $\mathbf{P}_i$. Otherwise, the path coordinate can be reached safely and the subroutine returns $true$.

2.1.2 State *ImagePlan*

As previously mentioned, this state is initialized by default. State *ImagePlan* is responsible for determining which currently visible GPS coordinate along $\mathbf{P}$, if any, the robot should travel toward.

IVG attempts to minimize the distance and time required to reach the goal by short-cutting the Cartesian path. It attempts to minimize the time required to reach the goal by traveling at the maximum speed c_{max} whenever possible. This is accomplished by finding the maximum i (i.e. the index of the most distant GPS coordinate) for which $ApprovePoint(\mathbf{P}_i)$ returns $true$, given the subset of $\mathbf{P}$ that is currently visible. In order to prevent the image subsystem from leading the robot toward subgoals that are blocked by obstacles beyond the robot's sensor range, path coordinates are only considered if they are within sensor range. If such an i can be found, then translational and angular velocity servoing commands are defined as follows:

$$trans = c_{max} \tag{4}$$

$$angular = \frac{\theta\,(u - w/2)}{w} \tag{5}$$

where θ is the horizontal field-of-view of $\mathbf{O}$, and u is the column of the image-space projection of $\mathbf{P}_i$.

The frame rate of the image space subsystem is assumed to be sufficiently high that a point will remain invalid (as defined by $ApprovePoint(\mathbf{P}_i) = false$) for the duration of a control loop iteration. If a valid target point cannot be found, then the image subsystem transitions to State *Scan*.

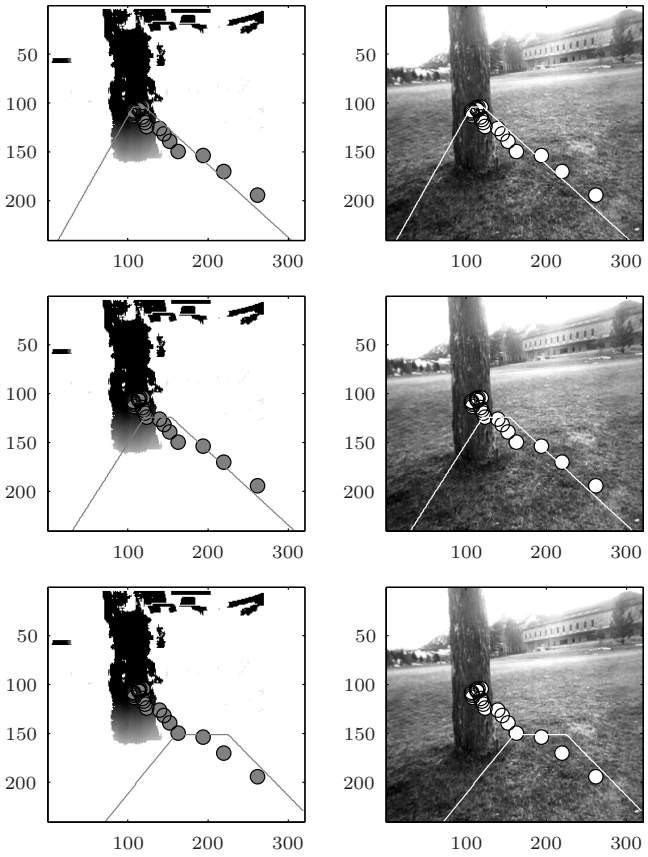

Figure 2: Image space occupancy grid $\mathbf{O}$, left, and the corresponding grayscale image, right. Top to bottom: State *ImagePlan* checks the visable points along the global path until it finds a coordinate that is safe to travel toward along a straight trajectory.

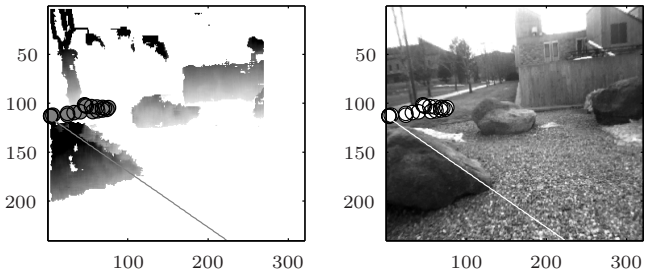

Figure 3: Image space occupancy grid $\mathbf{O}$, left, and the corresponding grayscale image, right. None of the currently visible global path points are safe; therefore, the system will transition to State *Scan*.

2.1.3 State *Scan*

This state is invoked if a safe straight-line path cannot be found to any currently visible $\mathbf{P}_i$ (as is the case in Figure 3). State *Scan* attempts to locate a valid $\mathbf{P}_i$ by causing the robot to look at the points in $\mathbf{P}$ that have not yet been checked by $ApprovePoint(\mathbf{P}_i)$.

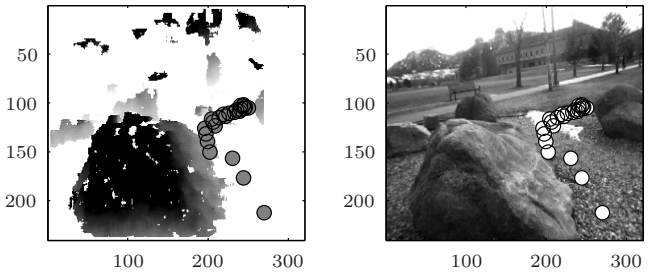

Figure 4: Image space occupancy grid $\mathbf{O}$, left, and the corresponding grayscale image, right. None of the global path points are safe; therefore, the system will transition to State *Blocked*.

Let $\mathbf{P}^{visible}$ be a boolean vector that defines whether or not points in $\mathbf{P}$ are visible given the camera's current position and orientation. If $\mathbf{P}_i$ is visible, then $\mathbf{P}_i^{visible}$ is defined to be $true$, otherwise $\mathbf{P}_i^{visible}$ is defined to be $false$. Let $\mathbf{P}^{checked}$ be a boolean vector that keeps track of whether or not points in $\mathbf{P}$ have ever been checked with $ApprovePoint(\mathbf{P}_i)$. If $\mathbf{P}_i$ has been checked, then $\mathbf{P}_i^{checked}$ is set to $true$ otherwise it is set to $false$. If State *Scan* is invoked, then all of the currently visible points have already been determined to be invalid; therefore, $\mathbf{P}^{checked}$ is initialized to $\mathbf{P}^{visible}$. When all currently visible points have been checked, the robot rotates toward the unchecked point, $\mathbf{P}_i^{checked} = false$, that requires the least amount of rotation to observe.

State *Scan* loops until either a valid point can be found or all $\mathbf{P}_i$ have been determined to be invalid. During each loop iteration $ApprovePoint(\mathbf{P}_i)$ is used to check any previously unchecked points that are now visible. If such a point is found to be valid, then servoing is performed using Equations 4 and 5 and the system returns to State $ImagePlan$. Otherwise, $\mathbf{P}_i^{checked}$ is set to $true$. If all of the points in path $\mathbf{P}$ have been checked and no valid points exist (that is, if $\mathbf{P}_i^{checked} = true$ for all i), then State *Blocked* is invoked.

2.1.4 State *Blocked*

This state is used when every point in $\mathbf{P}$ is invalid. State *Blocked* is charged with telling the Cartesian subsystem about the unusable path, and then waiting for a new path to be provided. This is accomplished by sending a message to the Cartesian subsystem containing the GPS coordinates of $\mathbf{P}_1$ (the first point in $\mathbf{P}$ and the beginning of the old path), as well as a unique key value. Upon receiving this notification, the Cartesian subsystem inserts a high cost at $\mathbf{P}_1$, calculates a new path, and then sends this new path to the local image space subsystem. The local subsystem remains in State *Blocked* until a new path arrives from the global subsystem that is tagged with the appropriate key value. This ensures that control will only be passed back to State $ImagePlan$ once information about $\mathbf{P}_1$ has been incorporated into the global path plan.

2.2 FLCP and LCP Local Image Space Subsystems

The FLCP image space subsystems is nearly identical to the LCP system proposed by [9]. Both model the world with an image-space cylindrical occupancy grid map $\mathbf{C}$. Assuming that the field-of-view of the camera sensor is less than 2π radians, $\mathbf{C}$ simulates a 2π radian panorama of the world in memory. $\mathbf{C}$ is fixed in orientation and populated by inserting values from $\mathbf{O}$ based on the robot's compass heading. $\mathbf{C}$ contains p columns and h rows.

The A* search algorithm is used to find a local image-space path between the image-space projections of the robot's current location and a subgoal; where the subgoal is obtained from the global path. In practice, the robot's location in $\mathbf{C}$ is defined to be the bottom (i.e. closest) row of $\mathbf{C}$ along with the column corresponding to the robot's current compass heading. Because the robot is represented as a point-particle during path search, $\mathbf{C}$ is preprocessed with a row-wise dilation to account for robot width. Dilation width is defined by the image-space projection of robot half-width $c_{width}/2$ plus a safety buffer c_{safety}, assuming a flat ground plane. Due to perspective, the column width of dilation is a function of occupancy grid row.

Servoing commands are calculated based on the position of a target point $\mathbf{C}_t$ located a predetermined distance d_t along the local path. In the case that the path is shorter than d_t, $\mathbf{C}_t$ is defined to be the final path coordinate. Note that the frame rate is sufficiently high that the local subsystem will generate a new local path before the robot reaches the real world projection of $\mathbf{C}_t$. Let v_t and u_t be the image space row and column of $\mathbf{C}_t$, respectively, and let h and p_1 be the image space row and column of the robot's starting location in image space, respectively. FLCP and LCP both control the robot's angular velocity as follows:

$$angular = \frac{\theta\left((u_t - p_1) \bmod p\right)}{w} \tag{6}$$

LCP controls the robot's translational velocity as follows:

$$trans = \frac{c_{max}\left(h - v_t\right)}{\sqrt{\left(\left((p/2 + u_t - p_1) \bmod p\right) - p/2\right)^2 + \left(h - v_t\right)^2}} \tag{7}$$

Let A be a point that is a predefined distance directly ahead of the robot. FLCP differs from LCP in that the servoing function is modified to allow the robot to travel at maximum speed whenever $ApprovePoint(A)$ returns $true$. Thus, if $ApprovePoint(A)$ returns $true$ or $false$, FLCP uses Equation 4 or 7 to determine the robot's speed, respectively. Cost-map dilation expands obstacles across the direction of travel, and each column of $\mathbf{C}$ represents a unique compass direction. Therefore, $ApprovePoint(A)$ only needs to check cost values stored in the specific column of $\mathbf{C}$ that A is projected into, when used in conjunction with FLCP.

2.3 DTP Local Cartesian Space Subsystem

The DTP system is a hierarchical planner composed of local and global subsystems that both use 2D top-down Cartesian models of the world. The local subsystem represents the

world with a robot-centric fixed-size map that has a relatively fine granularity. As with the other local subsystems, DTP's local subsystem searches for a path to a subgoal, where the subgoal is a function of the global path. The rest of the DTP local Cartesian subsystem is identical to the Global Cartesian space subsystem described in the next section.

2.4 Global Cartesian Space Subsystem

All four planning systems use identical global subsystems, with the exception that IVG allows the global subsystem to receive lethal coordinates from the local subsystem. The Cartesian space subsystem models the world as a 2D top-down occupancy grid map **G**. The map **G** expands with robot exploration, is fixed to a global coordinate system, and has a relatively coarse resolution. Cost information is inserted into **G** based on the relationship between stereo disparity and distance. Cost has the same definition as in Equation 2, except that movement between diagonal neighbors is multiplied by $\sqrt{2}$ to account for distance.

Paths through **G** are found using the A* algorithm ([33]), although any similar graph-search algorithm could also be used ([34, 35]). When a path is found, the **G**-based coordinates are translated into real-world GPS coordinates and then sent to the image subsystem via a message.

The IVG message also includes the most recent key value received from the image subsystem (a key value of *null* is sent if no messages have been received). The list of lethal coordinates from the local subsystem is maintained separately from the map, and then superimposed on the map prior to path search. The list is updated with new GPS coordinates as they arrive from the image-based subsystem. To account for dynamic environments, points are removed from the list after moving a predefined distance away from the robot.

To account for the point-particle robot representation used for path search, the map (including IVG lethal list) is dilated by robot half-width as a preprocessing step ([2, 7, 22]).

3 Apparatus and Implementation Details

We use the DARPA Learning Applied to Ground Robotics (LAGR) platform for the experiments presented in this paper. The dimensions of the robot are approximately 1.2 x .8 x 1.2 meters (length x width x height, respectively). Relevant hardware includes: two Point Grey Bumblebee 2 stereo camera pairs, a Garmin GPS receiver, and wheel odometers. The camera pairs output stereo disparity and RGB color information (RGB is unused in this work). We have found disparity to be accurate up to 15 meters. However, this range is often less depending on environmental conditions. Translation and rotation are achieved via two independently controlled wheels located on either side of the sensor mast. There is one computer dedicated to each of the camera pairs, another for the planning system, and a fourth that functions as a servo controller. The camera and planner computers have dual-core processors. Camera data is sent from the camera computers to the planning subsystems via network messages.

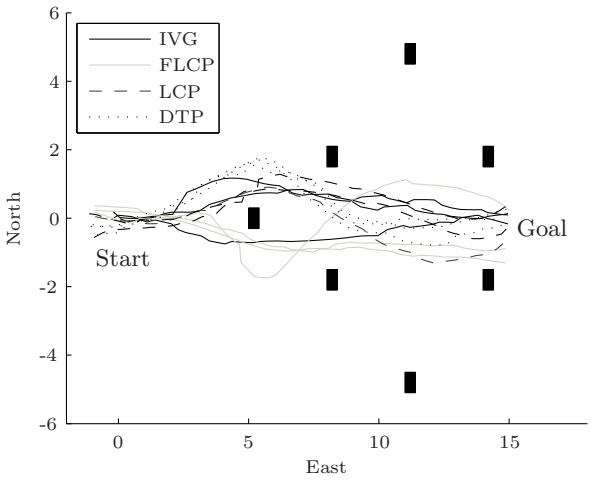

Figure 5: Experiment 1, a scattering of small obstacles on a field.

The image space occupancy grid used in the local subsystem of IVG contains $h = 240$ rows and $w = 320$ columns. The cylindrical image space map used in the local subsystems of FLCP and LCP contains $h = 40$ rows and $p = 200$ columns. The local Cartesian map used in the local subsystem of DTP is 18 meters North by 18 meters West and has a resolution of 0.2 meters. The subgoal used for local search in FLCP, LCP, and DTP is defined to be the first point along the global path that is at least 5 meters away from the robot. The target point used for the servoing functions in FLCP and LCP is chosen to be the twelfth node along the local image space path. The point A, used in FLCP to determine the safety of driving at the maximum speed, is defined to be 3 meters directly ahead of the robot. The global Cartesian subsystem uses a global map with a resolution of 0.5 meters. The IVG global Cartesian subsystem removes points from the lethal list if they are more than 15 meters away from the robot.

4 Experiments

All four planning systems are implemented on an autonomous robot and evaluated on three outdoor courses. Each planning system is tested three times per course. The maximum speed of all systems is set to 1 meter per second. GPS-based paths for each run are displayed in Figures 5-7, and the corresponding distances and runtimes are displayed in Tables 1 and 2, respectively. Traversal distance is calculated from the list of actual GPS coordinates that the robot is believed to have traversed. However, due to GPS drift, this may vary slightly from the actual path that was taken (within 8% for the experiments presented here). Note that if GPS drift artificially shortens the length of a quick test run, then the average robot speed may falsely appear greater than 1 meter per second.

Experiment 1 (Figure 5) is designed to test a planner's ability to navigate around small obstacles. The obstacles are arranged in a diamond pattern on a field.

Experiment 2 (Figure 6) contains 1.8 and 2.4 meter wide walls placed 10 and 20 meters

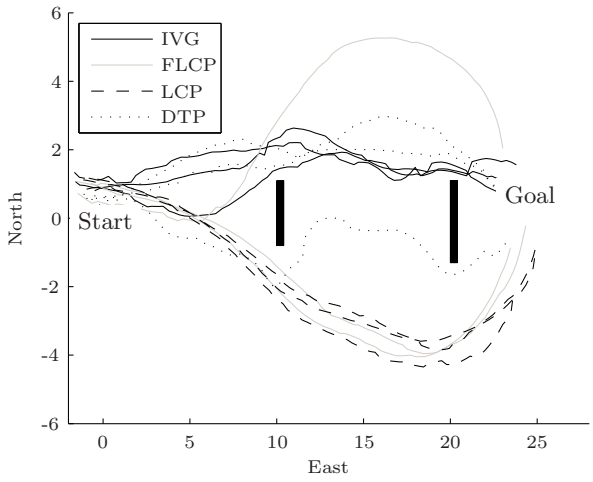

Figure 6: Experiment 2, two wall like obstacles on a field.

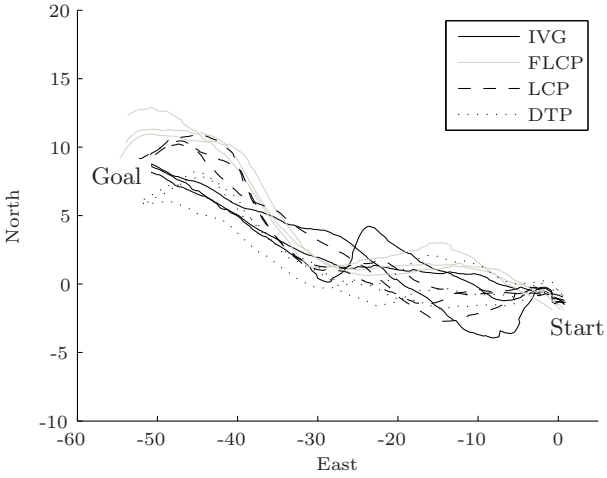

Figure 7: Experiment 3, slightly rolling terrain with natural obstacles such as trees.

in front of the robot, respectively. Experiment 2 is designed to test a planner's ability to navigate around medium-sized obstacles. The course is constructed such that that, after avoiding the first obstacle, the straightest path to the goal is blocked by the second obstacle.

Experiment 3 (Figure 7) is performed on rolling terrain with natural obstacles such as trees. This experiment is designed to test a planner's ability to navigate in a natural environment.

5 Results

The average runtime of FLCP is consistently less than that of other systems; however, it follows longer than average paths. In contrast, IVG requires more time to complete a course, while moving along shorter than average paths. Although no system consistently

Table 1: Experimental Traversal Distances In Meters

Experiment 1					
	Run 1	Run 2	Run 3	Mean	Std.
IVG	15.34	15.93	17.06	16.11	0.71
FLCP	17.91	15.96	16.00	16.62	0.91
LCP	16.38	16.88	16.67	16.64	0.20
DTP	17.20	16.86	16.55	16.87	0.27
Experiment 2					
	Run 1	Run 2	Run 3	Mean	Std.
IVG	25.27	25.50	25.68	25.48	0.17
FLCP	27.34	27.50	27.03	27.29	0.20
LCP	27.44	27.97	27.03	27.48	0.38
DTP	25.03	26.62	23.86	25.17	1.13
Experiment 3					
	Run 1	Run 2	Run 3	Mean	Std.
IVG	58.30	58.29	55.83	57.47	1.16
FLCP	58.99	59.60	60.39	59.66	0.57
LCP	58.81	60.57	60.59	59.99	0.83
DTP	55.42	57.08	55.48	55.99	0.77

Table 2: Experimental Runtimes In Seconds

Experiment 1					
	Run 1	Run 2	Run 3	Mean	Std.
IVG	29.92	30.36	31.51	30.60	0.67
FLCP	26.37	15.66	15.41	19.15	5.11
LCP	28.90	24.15	20.16	24.40	3.57
DTP	24.74	25.88	24.25	24.96	0.68
Experiment 2					
	Run 1	Run 2	Run 3	Mean	Std.
IVG	40.44	40.30	40.78	40.51	0.20
FLCP	26.05	30.37	25.57	27.33	2.16
LCP	30.79	31.56	30.54	30.96	0.43
DTP	33.01	37.88	31.61	34.17	2.69
Experiment 3					
	Run 1	Run 2	Run 3	Mean	Std.
IVG	72.37	71.41	69.94	71.24	1.00
FLCP	59.06	61.29	61.29	60.55	1.05
LCP	63.85	78.15	72.67	71.56	5.89
DTP	64.23	66.00	64.92	65.05	0.73

takes shorter or longer paths than the other three, IVG uses the shortest paths of any system with an image-space subsystem.

Even though both systems translate at the maximum speed whenever possible, IVG travels

at a slower average velocity than FLCP. We speculate that this result is caused by IVG's method of handling invalid global paths. If the local subsystem decides that none of the coordinates along the global path can be reached safely, then the robot must halt until a new global path can be found. This is evidenced by the fact that IVG has a relatively short average runtime on Experiment 3, which contains fewer obstacles than Experiments 1 and 2. On the other hand, the strategy of halting and waiting for a new global path conserves distance whenever a more efficient path can be found. FLCP finds a local path regardless of the validity of the global path. Even when it is dangerous to travel at the maximum speed (i.e. $ApprovePoint(A)$ returns $false$) the local path can still be used for navigation at a reduced speed. This increases the average velocity of FLCP relative to IVG.

The average runtimes of FLCP are less than those of LCP, although their average path lengths are within one standard deviation of each other. This suggests that FLCP is an improvement over LCP.

6 Conclusions

We have proposed two new image space planning techniques: Image Space Verification of Global Planning (IVG) and Fast Local Cylindrical Planning (FLCP). IVG uses a high-resolution local image space subsystem to determine the furthest global path coordinate that can be reached along an unobstructed heading, thereby short-cutting as much of the global path as possible. IVG is the first image space planning technique suitable for navigation in completely unstructured environments that does not use a graph-search algorithm in image space. Using functionality borrowed from IVG, FLCP improves upon previous cylindrical/Cartesian planners by determining when it is safe to translate at the maximum speed.

IVG and FLCP are experimentally evaluated on an autonomous robot and compared against a double 2D top-down Cartesian planner and a previously proposed hierarchical image/Cartesian space planner. All four systems are tested three times per course on three different course layouts. We find that IVG tends to follow shorter than average paths but requires more time to reach the goal. FLCP has consistently shorter runtimes than the other systems.

7 Future Work

The current image-based subsystem of IVG only uses information from a single camera pair. Given that many robotic platforms contain more than one imaging sensor, a natural extension to this work simultaneously incorporates data from multiple sensors into the IVG framework. Two possible ideas include:

1. Running one image-based subsystem per imaging sensor, and then defining GPS coordinates as valid only if they are independently valid in each of the subsystems.
2. Fusing the cost information provided by each sensor into a single image space cost

map, and then using the combined map as input to an image space planner.

The average speed of IVG is considerably reduced by the current method of dealing with invalid global paths. Another extension to this work involves creating a way for IVG to handle blocked paths without stopping.

References

[1] M. Herman, "Fast, three-dimensional, collision-free motion planning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'86)*, San Francisco, California, 1986, pp. 1056–1063.

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

[3] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," in *IEEE Computer*, 1989, p. 4657.

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

[5] B. H. Krogh and C. E. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'86)*, San Francisco, California, 1986, pp. 1664–1669.

[6] J. Hong, X. Tan, B. Pinette, R. Weiss, and E. M. Riseman, "Image-based homing," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'91)*, New York, 1991, pp. 620–625.

[7] M. Sugiyama, Y. Kawano, M. Niizuma, M. Takagaki, M. Tomizawa, and S. Degawa, "Navigation system for an autonomous vehicle with hierarchical map and planner," in *Proc. of the Intelligent Vehicles '94 Symposium*, Oct. 1994, p. 5055.

[8] E. Gat, M. Slack, D. Miller, and R. Firby, "Path planning and execution monitoring for a planetary rover," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'90)*, vol. 1, Cincinnati, USA, 1990, pp. 20–25.

[9] M. Otte, S. Richardson, J. Mulligan, and G. Grudic, "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, 2007.

[10] M. Ollis, W. H. Huang, M. Happold, and B. A. Stancil, "Image-based path planning for outdoor mobile robots," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, 2008.

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

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

[13] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetory exploration," in *IEEE Aerospace Conference Proceedings*, Big Sky, MT, USA, 2002.

[14] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara, "Obstacle avoidance and path planning for humanoid robots using stereo vision," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'04)*, 2004.

[15] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, "Global path planning on board the mars exploration rovers," in *IEEE Aerospace Conference*, 2007.

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

[17] K. N. Kutulakos, V. J. Lumelsky, and C. R. Dyer, "Vision guided exploration: A step toward general motion planning in three dimensions," in *Proc. IEEE Conference on Robotics and Automation*, vol. 1, 1993, pp. 289–296.

[18] B. Nabbe, "Extending the path-planning horizon," *Ph.D. dissertation*, 2005.

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

[20] Y. Matsumoto, K. Sakai, M. Inaba, and H. Inoue, "View-based approach to robot navigation," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00)*, vol. 3, 2000, pp. 1702–1708.

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

[22] A. Kelly, "Adaptive perception for autonomous vehicles," *Technical Report CMU-RI-TR-94-18*, 1994.

[23] ——, "Mobile robot localization from large-scale appearance mosaics," *International Journal of Robotics Research*, vol. 19, pp. 1104–1125, 2000.

[24] J. Minguez and L. Montano, "Nearness diagram (nd) collision avoidance in troublesome scenarios," *IEEE Transactions On Robotics and Automation*, vol. 20, pp. 45–59, 2004.

[25] J. Borenstein and Y. Koren, "The vector field histogram–fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[26] S. Lee and T. M. Adams, "Spatial model for path planning of multiple mobile construction robots," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, pp. 231–245, 2004.

[27] B. Faverjon and P. Tournassound, "The mixed approach for motion planning: Learning global strategies from a local planner," in *International Joint Conference on Artificial Intelligence*, 1987.

[28] X. Yang, M. Moallem, and R. V. Patel, "A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, pp. 1214–1224, 2005.

[29] D. Song, H. N. Lee, J. Yi, and A. Levandowski, "Vision-based motion planning for an autonomous motorcycle on ill-structured roads," *Auton Robot*, vol. 23, pp. 197–212, 2007.

[30] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 14–23, 1986.

[31] R. C. Arkin and D. C. Mackenzie, "Planning to behave: A hybrid deliberative/reactive robot control architecture for mobile manipulation."

[32] M. Hassoun and C. Laugier, "Reactive motion planning for an intelligent vehicle," *Proceedings of the Intelligent Vehicles Symposium*, 1992.

[33] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," in *Proc. IEEE Transactions On System Science and Cybernetics (SSC-4)*, 1968, pp. 100–107.

[34] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'94)*, 1994.

[35] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'02)*, 2002.