The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

A Stereo Vision Based Mapping Algorithm for Detecting Inclines, Drop-offs, and Obstacles for Safe Local Navigation

Aniket Murarka
Computer Science
The University of Texas at Austin
aniket@cs.utexas.edu

Benjamin Kuipers
Computer Science and Engineering
University of Michigan
kuipers@umich.edu

Abstract— **Mobile robots have to detect and handle a variety of potential hazards to navigate autonomously. We present a real-time stereo vision based mapping algorithm for identifying and modeling various hazards in urban environments – we focus on inclines, drop-offs, and obstacles. In our algorithm, stereo range data is used to construct a 3D model consisting of a point cloud with a 3D grid overlaid on top. A novel plane fitting algorithm is then used to segment the 3D model into distinct potentially traversable ground regions and fit planes to the regions. The planes and segments are analyzed to identify safe and unsafe regions and the information is captured in an annotated 2D grid map called a *local safety map*. The safety map can be used by wheeled mobile robots for planning safe paths in their local surroundings. We evaluate our algorithm comprehensively by testing it in varied environments and comparing the results to ground truth data.**

I. INTRODUCTION

A wheeled mobile robot navigating in an urban environment has to deal with many potential hazards (Table I). The robot must be able to avoid obstacles and, critically, detect catastrophic hazards like drop-offs. It is also important that the robot is able to detect inclined surfaces and estimate their slopes so as to avoid rolling over or slipping.

Potential Hazards	Examples
Obstacles: Static	Walls, furniture
Dynamic	People, doors
Invisible	Glass doors and glass walls
Drop offs	Sidewalk curbs, downward stairs
Inclines	Wheelchair ramps, curb cuts, sloped sidewalks
Overhangs	Table tops, railings, tree branches
Rough surfaces	Gravel paths, grass beds
Narrow regions	Doorways, elevators

TABLE I: Potential hazards that a mobile robot must deal with in urban environments.

Research in mobile robotics has primarily focused on detecting obstacles [1]. Methods for detecting drop-offs [2], [3], inclines and other hazards have featured less prominently even though detecting them is just as important. Furthermore, laser range-finders have been used as the primary sensors on most robots because they provide reliable and high quality range information. Cameras have also been used extensively, but on almost all practical navigation systems they are accompanied with lasers [1]. Reasons for cameras not taking a central role include the sensitivity of cameras to environmental conditions and the difficulty in reliably interpreting images.

However, we focus on the use of cameras as the primary sensors, instead of lasers, because cameras are cheaper and smaller and an image provides a lot more information than a laser scan. For the price of a single laser scanner, it is possible to have several cameras on a robot providing a greater field-of-view and information. Furthermore, in addition to being useful for navigation, cameras are useful for a variety of other tasks such as detecting objects and recognizing places.

Therefore, in this paper we present a stereo vision based mapping algorithm for detecting inclines, drop-offs, and static obstacles for enabling wheeled mobile robots to navigate safely in urban environments. Our stereo mapping algorithm identifies safe and unsafe regions in the robot's local surroundings and represents this information using an annotated 2D grid map called a *local safety map*. The 2D local safety map can be used by the robot to plan safe local paths using standard planning algorithms [19]. In addition, we also present an evaluation framework that allows for the comprehensive evaluation of mapping systems, such as ours. An overview is provided in the following section.

A. OVERVIEW OF SYSTEM

As the robot explores its local surroundings it gets a constant stream of stereo images. Each new stereo image pair (or frame) is processed as outlined in the following steps (corresponding to Sections III through VI), to update the robot's current knowledge of the world:

A. A depth (or disparity) map is computed from the stereo image pair (Fig. 1). The depth readings are transformed into global coordinates (i.e., map coordinates).

B. A 3D model consisting of a 3D grid and 3D point cloud is updated with the new depth readings using an occupancy grid algorithm (Fig. 2).

C. Planes are fit to potentially traversable ground segments in the 3D model. The process consists of segmenting the 3D grid followed by fitting planes to points corresponding to the segments using linear least squares (Fig. 3).

D. Finally, the segments and planes are analyzed for safety to yield the local safety map (Fig. 4).

Cells in the local safety map are annotated with one of the following labels: *Level:* implying the region in the cell is level and free of obstacles; *Inclined:* the cell region is inclined; *Non-ground:* the cell has an obstacle or overhang or is lower in height than nearby ground regions; *Unknown:* there is insufficient information about the region. *Level* and *Inclined* cells are considered safe and whereas *Non-ground* cells are considered unsafe. Safe cells can be further annotated as being *Potential Drop-off Edges*, if a drop-off is

expected to be present in the vicinity of the cell. Additionally, *Inclined* cells are annotated with their surface normals.

To transform the depth readings into global coordinates in step **A**, we need to know the robot's pose in the global frame. Since the focus of this work is on building models of the environment, we assume that the robot is able to localize. For our experiments we use a laser based 3-DOF localization algorithm [4] for providing the robot's pose to our mapping algorithm (the 3-DOF method can be replaced by a camera based 6-DOF SLAM algorithm [5]). Since we use a 3-DOF localization module, our robot is restricted to travelling only on level surfaces for the experiments reported in this paper. However, our reconstruction algorithm is general and applicable without modification to the case when the robot's motion has 6-DOF.

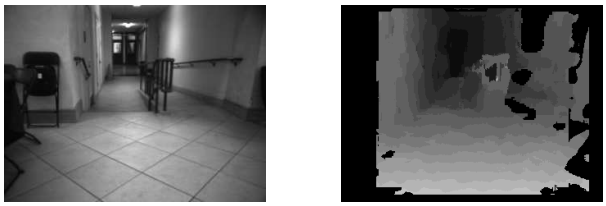
We evaluate our system quantitatively by constructing safety maps for five different datasets and measuring error rates by comparing the constructed maps against ground truth maps. We also measure latencies present in the system and the accuracy of the plane fitting process. The evaluation framework is very general and we believe it to be a comprehensive way of evaluating and comparing the performance of a variety of mapping systems. Towards this goal we have made our video datasets and associated ground truth data publicly available [6].

Our contributions in this work are as follows: (i) A real-time stereo vision based system for detecting inclines, drop-offs, and obstacles. (ii) A comprehensive evaluation framework for measuring the performance of various mapping systems. As part of the system, our contributions are: (a) A novel method for fitting planes to traversable ground segments in 3D point clouds. (b) A novel method for analyzing the segments and planes to build 2D annotated safety maps.

II. RELATED WORK

Here we discuss related prior work on hazard detection using vision and other sensors. We begin with two systems that are closely related to our own. Gutmann *et al.* [7] present a real-time stereo based navigation system for humanoid robots. To get accurate floor heights, the raw range data is first segmented into planes. They then build a 3D model consisting of an occupancy grid and floor height maps and use labels similar to ours. However, their system only works for level surfaces and does not handle inclines. Rusu *et al.* [8] present a real-time stereo algorithm that creates a 3D model consisting of polygonal patches. Each patch is analyzed locally to determine semantic labels similar to our labels. Since the analysis is very local we believe it can lead to incorrect labels unlike our work where we first segment surfaces into larger regions and then assign labels. Furthermore, they do not detect drop-off edges in their system.

Other stereo algorithms include work by Iocchi *et al.* [9] who present a stereo vision based system for building planar 3D models. However, they only consider vertical planes and a level ground. Singh *et al.* [10] fit planes to stereo data to construct 2D local grid maps and to estimate the traversability of cells. Again the analysis is local in nature meaning that it can lead to incorrect estimates of slopes. In addition no semantic labels, like ours, are determined.



(a) (b)

Fig. 1: (a) Left image from the stereo camera showing a scene from dataset 1, with a drop-off to the left, and a ramp to the right, of the center rail. (b) Computed disparity map (brighter areas closer).

Two vision-based methods for the detection of drop-offs are evaluated in [11]. One method looks for gaps in stereo range data, while another looks at height differences and gaps in local terrain/height maps of the environment. The results from both methods are combined to identify drop-offs. Ramps are not considered. Rankin *et al.* [12] merge stereo vision and thermal signatures to detect drop-offs at night, requiring the use of special infrared cameras.

There exist various methods for fitting planes to point cloud data [13], [14]. Of these, Expectation Maximization (EM) based methods are very popular [14]. However, EM based methods require that the number of planes be estimated in some manner beforehand and are subject to local minima that can be difficult to avoid. Recently, Gaussian Processes (GP) [15] have become popular for analyzing laser range data for traversability. However, GPs require heavy computation and it is unlikely that GPs can handle the sort of false readings produced by stereo unless coupled with other methods.

Laser range-finders are used extensively for detecting obstacles and other hazards. In their DARPA Grand Challenge vehicle, Thrun *et al.* [1] use lasers mounted on top of the car to detect obstacles by measuring height differences of regions in front of the car. Heckman *et al.* [3] find drop-offs using 3D laser data. They ray-trace to find occlusions in the laser grid and determine the cause of the occlusions. Wellington *et al.* [16] use lasers and cameras to find the true ground height and traversability of vegetation covered regions. These systems differ significantly from ours in the algorithms used and require the use of expensive laser sensors.

In the following sections we describe in detail the main steps in our system.

III. COMPUTING STEREO DEPTH MAPS

In the first step a disparity map is computed from the image pair returned by the stereo camera. We use a standard off-the-shelf multi-scale correlation stereo method [17] to compute the disparity maps (Fig. 1). In post-processing, we remove range readings that are significantly different from neighboring range readings. Such readings have a high likelihood of being incorrect and their removal noticeably improves our system's performance.

The disparity map provides distances to points in the world relative to the camera (and hence relative to the robot

assuming the camera's coordinates are known in the robot's frame). Since localization gives the robot's pose in the global coordinate frame, a point's 3D position in the global frame (denoted $(x,y,z)^T$) can be computed. Since we are only interested in safe local motion the localization method does not have to be globally consistent, only locally consistent.

The output of this step is a set of 3D range points in the global (or map) coordinate frame.

IV. Updating the 3D Model

In the second step, the 3D range points are used to update a 3D model, consisting of a point cloud and an occupancy grid, of the robot's surroundings. The grid is of a fixed size since we are interested in local reconstruction. We can have the grid move while centered on the robot thereby always providing the robot with a model of its local surroundings. However, in our experiments the distance moved by the robot is typically small, so a non-scrolling grid suffices. A fixed size grid has the benefit of bounded computation irrespective of the distance travelled.

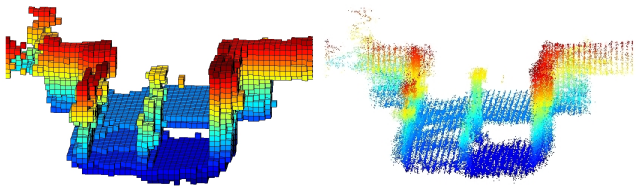
The occupancy grid is updated probabilistically [18]. For each 3D range point, a ray is cast from the camera to the 3D point and voxels along the ray have their log odds probability (LOP) of occupancy updated. Updates are done by incrementing or decrementing the LOP by the log odds likelihood of the voxel having produced the range point [18] (the sensor model). In our work, the LOP of the voxel in which the point falls is incremented by $incr_H$; this voxel's fore and aft neighbors have their LOP incremented by a lower amount, $incr_L$; and all other voxels between the fore neighbor and camera have their LOPs decremented by a fixed amount $decr$. A voxel starts off with a LOP of zero. Finding the correct sensor model amounts to tuning the *relative* values of the increment and decrement parameters. In Section VII-A we examine the effect of different $decr$ values (for fixed values of $incr_H = 10$ and $incr_L = 4$) on mapping accuracy.

For occupancy grid algorithms to work properly, range readings should be independent [18]. Unfortunately, this is not true of stereo range readings. Stereo range points that fall in a particular voxel are produced by neighboring pixels and hence are usually highly correlated. We reduce the effect of correlation by updating the grid for only one point per voxel per timestep – we use the first point that falls in a voxel and discard all other points (a side effect of this is an increase in computational efficiency). We also discard entire stereo frames when the robot is stationary, since range readings across such frames are highly correlated.

A point cloud database is also updated in this step. For each voxel a list of the points that fall in the voxel over time is maintained. The indices of the voxel in which a point $(x,y,z)^T$ falls are given by,

$$(u,v,w) = ([x/l_v],[y/l_v],[z/l_v]) \quad (1)$$

where $[\cdot]$ is the rounding operator and l_v is the length of a voxel's side (in our experiments we found $l_v = 0.1m$ to give a good balance between mapping accuracy and computational efficiency). The list is of a fixed size and at each frame the list is updated with the current point that falls in the voxel.



(a) (b)

Fig. 2: 3D model of dataset 1 (rendered from a viewpoint different from the image in Fig. 1(a)), built after processing 459 stereo frames. The 3D model consists of (a) a 3D grid and (b) the corresponding 3D point cloud model. Note the discretization imposed on the ramp by the 3D grid (figure better viewed in color).

Each voxel's list is ordered according to the distance of the camera from the point, when the point was seen. Thus, when the list is pruned, points seen from closer distances are given preference since such points have lower error.

Once the updates are done, voxels that have a high probability of occupancy (we use a threshold of $occ_t = 100$ on LOP to determine when a voxel is occupied) are identified in the 3D occupancy grid. Thus, the output of this step is a 3D grid of occupied voxels plus the list of points (the 3D point cloud) associated with the occupied voxels (Fig. 2).

V. Segmenting & Fitting Planes to the 3D Model

In this step, the 3D model is first segmented into potentially traversable ground regions (Section V-A) and then planes are fit to points corresponding to the segments using linear least squares (Section V-B).

A. Segmenting the 3D grid

The segmentation of the 3D grid is achieved as explained in Algorithm 1. The segments found for a 3D grid are shown in Fig. 3(a). Each segment is a collection of voxel columns that have the *same height* where a voxel column consists of voxels that have the same (u,v) indices (Eq. (1)). The index w is along the vertical direction and is called the *indexed height* of a voxel (u,v,w). The height of a voxel column is the *indexed height* of the highest occupied voxel in the column that is below the robot's height (the robot can travel under occupied voxels higher than it). The segments represent distinct level and inclined ground regions that might be potentially traversable. All remaining voxel columns that are not part of any segments either represent obstacles or unreachable areas and are considered *non-traversable*.

An important detail in the algorithm is that we only consider voxel columns within a given planning radius R_{max} of the robot's current pose. This is because stereo returns good range data only within a limited distance and considering cells far away results in many errors. Section VII-B provides an experimental justification for the choice of R_{max}.

Finding the segments utilizes the fact that robot can travel only on horizontal or slightly inclined surfaces such as ramps. Since we are interested in wheeled mobile robots, surfaces with high inclines are considered non-traversable. Therefore in the segmentation process we look for regions of the same height, that are potentially reachable from where the robot currently is. The segmentation algorithm is very fast

Algorithm 1 Find Traversable Ground Segments in a 3D Grid

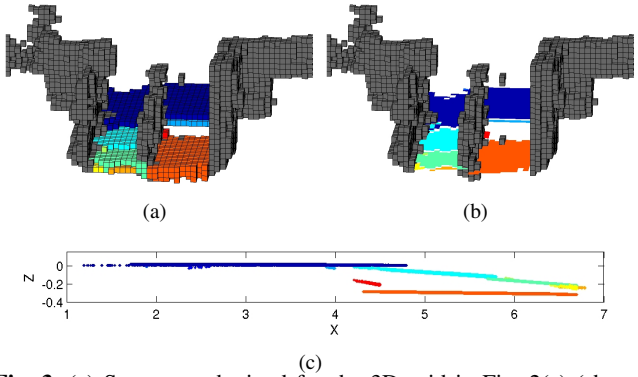
Require: 3D grid and the robot's pose in the grid.

1: Build a height map: a 2D grid with each (u,v) cell containing the height w of the corresponding (u,v) voxel column in the 3D grid.
2: Initialize $c = (u_r, v_r)$ – the cell in the height map where the robot is currently located.
3: Mark all cells that are greater than a planning radius R_{max} $(= 4m)$ away from the robot as "not to be examined".
4: Create an empty list L_U. The list will contain cells that have to be examined. Let $k \leftarrow 1$.
5: Create a new segment S_k and add to it the voxel column corresponding to cell c.
6: **while** A cell $c_i \in S_k$ can be found whose height has not yet been compared to that of its neighbors **do**
7: Let w_i be cell c_i's height.
8: Find the list of cells, N_c, neighboring c_i (do not include cells marked as "not to be examined").
9: **for each** $c_n \in N_c$ **do**
10: Let w_n be cell c_n's height.
11: **if** $w_n = w_i$ **then**
12: Add the voxel column corresponding to c_n to segment S_k.
13: **else if** $|w_n - w_i| \leq w_T$ $(= 1)$ **then**
14: Add c_n to L_U. c_n's height is not that different from c_i's and hence it can lead to the creation of a new segment. Cells whose height is very different are probably unsafe.
15: **end if**
16: **end for**
17: Mark c_i as a cell whose height has been compared to that of its neighbors.
18: **end while**
19: **if** $L_U \neq \phi$ **then**
20: Pop L_U until a cell c not yet part of any segment is found. If no such cell is found, **Return** the list of segments S_k, $\forall k$ found.
21: $k \leftarrow k+1$. Goto step 5.
22: **end if**
23: **Return** the list of segments S_k, $\forall k$ found.

(see Section VII-D) and finds appropriate ground segments. It over-segments, in that a single incline can get broken into two or more segments, but this allows our algorithm to find small changes in slope.

B. Least Squares Fit

Once we obtain a segmentation of the 3D grid, for each segment S we find the 3D points associated with all voxels in the segment (using the 3D point cloud). We then fit a plane of the form $z = p_{s_1}x + p_{s_2}y + p_{s_3}$ to the points using a standard linear least squares formulation to find the best fitting plane parameters $p_s = (p_{s_1}, p_{s_2}, p_{s_3})$. Fig. 3(b), 3(c) show the planes obtained for the segmentation in Fig. 3(a).



(c)

Fig. 3: (a) Segments obtained for the 3D grid in Fig. 2(a) (shown using different colors). (b) Corresponding planes obtained for each of the ground segments after the least squares fit. (c) Cross-sectional view of the planes showing the level ground, the ramp, and the below ground region (figure better viewed in color).

The output from this step is the list of ground segments, their associated plane parameters, and also the list of non-traversable voxel columns.

VI. Building the Local Safety Map

In this step we analyze the segments and plane parameters for safety. We assume that the segment on which the robot is located, is safe. We then find all segments that are reachable by the robot from the first segment and label those as *Safe*. All unreachable segments are labeled as *Non-ground*, i.e., unsafe.

We begin by finding interior and boundary cells of all segments (we can think of the segments in 2D terms as each voxel column corresponds to a cell in a 2D grid). Interior cells are those that have 8 neighbors, with all of those neighbors belonging to the segment itself. Cells for which this does not hold are boundary cells. Next we find neighboring segments: 2 segments are neighbors if any of their boundary cells are neighbors.

The segments are then analyzed for connectivity and labeled for safety as described in Algorithm 2. The local safety map is then obtained by creating a 2D grid (with the same (u,v) dimensions as the 3D grid) and assigning the same labels to the cells as the labels of their corresponding segments. Cells corresponding to the list of non-traversable voxel columns found above (previous section) are marked *Non-ground*. Cells that fall outside the planning radius, R_{max} are labeled as *Unexamined*. Cells with no labels are marked *Unknown*.

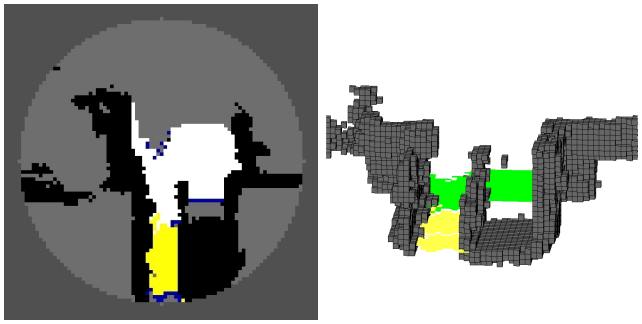
If desired, boundary cells of segments labeled *Level* or *Inclined*, can be further annotated as *Potential Drop-off Edges*. We consider a boundary cell as possibly being a drop-off edge if it is next to an *Unknown* cell and is *close* to cells that are lower in height than it. This condition finds cells that are drop-off edges but also find cells that are not. As a result, in a practical navigation system cells marked as *Potential Drop-off Edges* should be treated as requiring further observation (it is possible to find confirmed drop-off edges in front of the robot using the method described in [2]).

The parameters θ_{max}, h_T, and num_T in Algorithm 2 are determined using the motion capabilities and dimensions of

Algorithm 2 Labeling Segments and their Planes for Connectivity and Safety

Require: Segment and plane list with neighborhood information.

1: Initialize by labeling all segments in the segment list as *Unexamined*.
2: Label segments with planes with a slope greater than θ_{max} $(= 10^o)$ as *Non-ground*.
3: Label small segments with less than n_{small} $(= 6)$ cells as *Unknown*.
4: Label thin segments as *Unknown*. Thin segments are those for which: $\#interior\ cells < r_{thin} \times \#boundary\ cells$ where $r_{thin} = 0.1$. Small and thin segments are usually poorly observed and hence not considered.
5: Create an empty list L_S. L_S will contain segments to be examined further for safety.
6: Label the segment on which the robot is located as *Safe* and add it to L_S.
7: **while** $L_S \neq \phi$ **do**
8: Pop L_S until a segment S is obtained that has not had its neighbors examined.
9: Find neighboring segments, N_S, of S.
10: **for each** $R \in N_S$ **do**
11: Find neighboring boundary cells c_{RS} of S and R.
12: $num_{RS} \leftarrow 0$
13: **for each** cell $c \in c_{RS}$ **do**
14: Compute the xy-coordinates (x_c, y_c) of the center of cell $c = (u, v)$ as follows: $x_c = u \cdot l_v$ and $y_c = v \cdot l_v$ (From Eq. (1)).
15: Compute the expected ground height at cell c's center using the plane parameters of both segments R and S: $h_c^S = p_{s_1} x_c + p_{s_2} y_c + p_{s_3}$ and $h_c^R = p_{r_1} x_c + p_{r_2} y_c + p_{r_3}$.
16: Compute the difference, $h_c(S,R) = |h_c^S - h_c^R|$.
17: **if** $h_c(S,R) < h_T$ (where $h_T = 0.05$m) **then**
18: $num_{RS} \leftarrow num_{RS} + 1$
19: **end if**
20: **end for**
21: **if** $num_{RS} \geq num_T$ $(= 5)$ **then**
22: Segment R is reachable from S. Label R as *Safe* and add R to L_S.
23: **end if**
24: **end for**
25: Mark S as having had all its neighbors examined.
26: **end while**
27: Re-label all remaining segments still labeled *Unexamined* as *Non-ground* as they are unreachable from the robot's current position.
28: Re-classify all segments labeled *Safe* into *Level* or *Inclined* segments depending on the slope of their planes. If the slope is greater than $\theta_T = 3$ degrees label the segment as *Inclined* else label it as *Level*.
29: Return all segments with their labels.



(a) (b)

Fig. 4: (a) The safety map obtained after analyzing the segments and fitted planes in Fig. 3 for safety. Color scheme: *Non-ground* regions in black; *Level* regions in white; *Inclined* regions in yellow; *Unknown* regions in light grey; *Unexamined* regions in dark grey; and *Potential Drop-off Edges* in blue. (b) Corresponding hybrid 3D model obtained. *Non-ground* regions are represented using voxels (grey) and safe ground regions are represented using planes: green denotes *Level* planes and yellow denotes *Inclined* planes (figure better viewed in color).

the robot for which the safety maps are being created. θ_{max} is determined by the maximum incline that the robot can navigate. h_T is the maximum height difference between two adjoining surfaces that the robot can navigate and num_T is the number of cells that make up the robot's width.

Thus, the safety map as created, is weakly dependent on the robot's dimensions and navigation capabilities but independent of the trajectories that the mobile robot may take when navigating. We believe that it is the task of the path planner to generate and test the feasibility of different trajectories based on the safety map annotations. This clearly demarcates the goals of the mapping and path planning modules.

The output from this step of processing is a 2D local safety map (Fig. 4(a)) with various annotations for safety. In addition to the safety map, we can also construct a hybrid 3D model (Fig. 4(b)) that might be useful for other robot applications. This model is a hybrid of a 3D grid and 3D planes: the grid is used to represent *Non-ground* regions and the planes are used to represent safe ground regions. However, we believe that for most navigation applications the 2D local safety map should suffice.

VII. EVALUATION FRAMEWORK AND RESULTS

Our evaluation framework consists of three different parts. First, we measure the accuracy of the safety maps created by the system on five stereo datasets by comparing them to ground truth (GT) safety maps. Second, we measure latencies in the system that arise due to noise filtering, and third we measure the system's plane fitting accuracy. The datasets were collected using the robot shown in Fig. 5. Laser data was collected along with video data and used to provide ground truth for our evaluation. Laser based maps of the environment were created and then manually annotated and cleaned to give GT safety maps used in the first part of the evaluation (the laser maps provided excellent starting points). We manually fit planes to laser range points to give GT planes used in the third part of the evaluation.

Fig. 5: The wheelchair robot used for collecting the video datasets. The robot has a stereo camera and one horizontal and one vertical laser-rangefinder. The lasers are only used for localization and providing ground truth data.

A. Error Rates

The first part of the evaluation measures the classification accuracy of the system by comparing the stereo safety maps against ground truth safety maps. We do this for five stereo video data sets (between 350-500 stereo image pairs in each dataset) collected in a wide variety of environments. Two of the datasets have both ramps and drop-offs, one dataset has two ramps, and the remaining two datasets have drop-offs only. Most environments have large poorly textured regions, and both indoor and outdoor areas are represented. Lighting varies from good to fair.

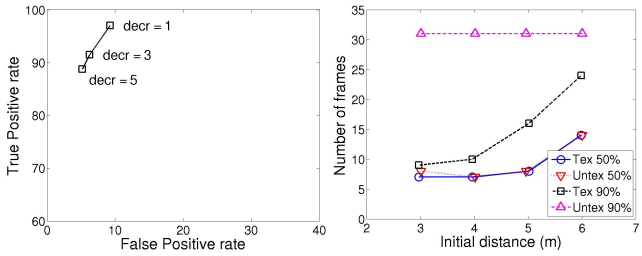
For each constructed safety map we identify the following cells: (i) False Positives (FP): cells marked unsafe in the constructed stereo map but labeled safe in the GT map. (ii) False Negatives (FN): cells marked safe in the constructed map but marked unsafe in the GT map. (iii) True Positives (TP): cells marked unsafe that are unsafe. (iv) True Negatives (TN): cells marked safe that are safe. For purposes of evaluation, cells marked *Level* or *Inclined* are considered safe and only cells marked *Non-ground* are considered unsafe. Cells marked *Unknown* or *Unexamined* are considered to be unclassified. For each safety map we find the number of FP, FN, TP, and TN cells. Also, of all classified cells in a safety map we find the number of true safe and unsafe cells present (the cells have to be classified by the GT map as well).

The errors are computed for all stereo image pairs for which a safety map is created - frames that capture exactly the same scene as previous images (this happens when the robot is stationary, see Section IV on why we do this) are discarded. Discarded frames only account for 10 to 20 percent of all images. For a given dataset we compute an overall false negative rate by summing the number of false negatives in all safety maps created and dividing by the sum of the number of unsafe cells in those safety maps. Other error rates are computed in a similar manner. Once error rates have been computed for each data set, they are averaged across all datasets and the standard deviations are computed.

Table II shows these averaged error rates for our system for three different values of the decrement parameter *decr* mentioned in Section IV. We deemed our system to be sensitive to this parameter and hence measured its effect on the error rates - Fig. 6(a) shows an ROC curve of the average FP and TP values in Table II. These results along with Fig. 7 and 8 demonstrate the good performance of our system. All figures and the movie associated with this paper are for *decr*=1.

Decrement Parameter Value	1	3	5
False Negative Rate (FN)	3.0 ± 1.8	8.5 ± 5.3	11.2 ± 7.3
False Positive Rate (FP)	9.3 ± 4.7	6.2 ± 3.7	5.2 ± 2.4
True Positive Rate (TP)	97.0 ± 1.8	91.5 ± 5.3	88.8 ± 7.3
True Negative Rate (TN)	90.7 ± 4.7	93.8 ± 3.7	94.8 ± 2.4

TABLE II: Safety map error rates for the system, averaged across all datasets, shown for three values of the decrement parameter *decr* (Section IV). Standard deviation values are also given.



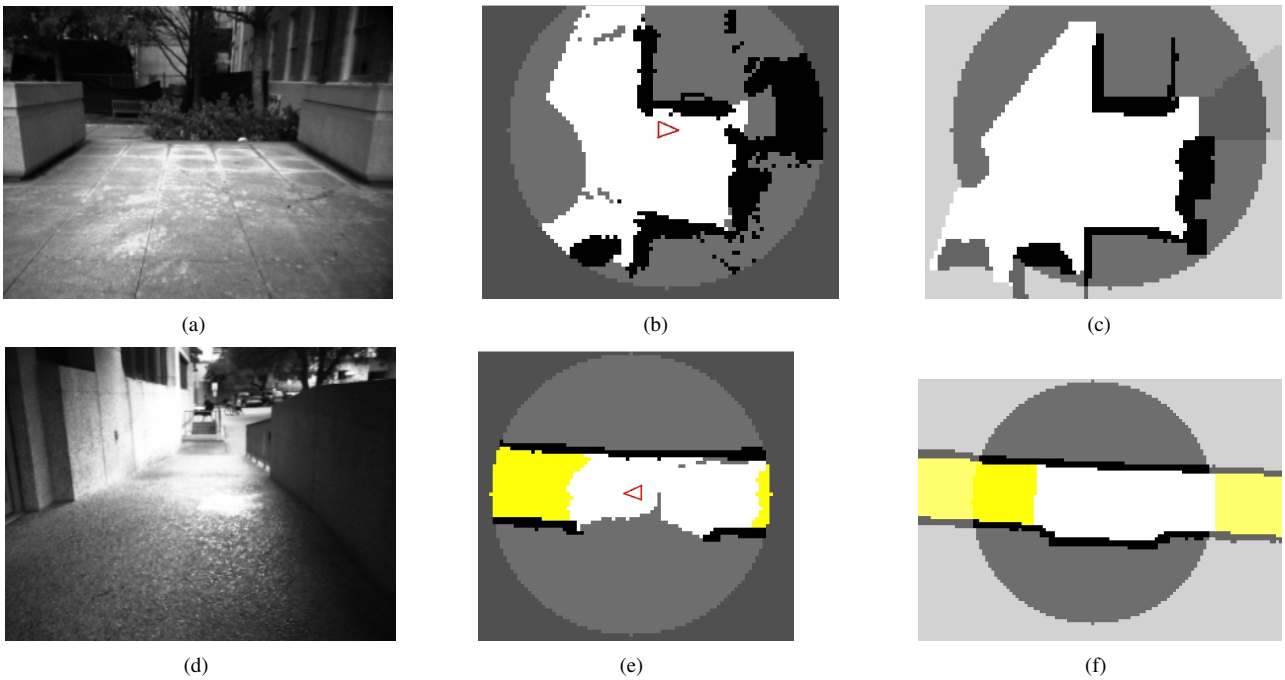
(a) (b)

Fig. 6: (a) ROC curve showing TP rate vs FP rate for the *decr* parameter (Section IV). Increasing the parameter increases the effect of negative evidence in the occupancy grid. As expected as *decr* increases both the FP and TP rates decrease (the FN rate increases). Depending on what is a minimum acceptable TP rate the parameter can be adjusted to get closer to a desired FP rate. (b) Frame Latency: Plot showing number of frames that go by, before 50% and 90% of the width of a board is visible in the robot's occupancy grid, as a function of the initial distance to the board and the board's texture. Both textured (Tex) and untextured (Untex) boards are detected in about the same number of frames for the 50% case whereas for the 90% case the robot reaches the untextured board before it is detected (we plot the maximum value of frame latency for this case).

The most important error metric is the FN rate which turns out to be very low for our system. It should be noted that a particular FN rate does not translate directly into the chances of an accident (e.g., a 3 percent FN rate does not mean the robot has a 3 percent chance of having an accident). The movie associated with this paper shows the nature and distribution of the FN errors the robot experiences. They are few and usually at some distance from the robot thereby not putting the robot in immediate danger. Furthermore robots usually have a margin of safety and so in general the effect of a particular FN rate is expected to be far less than what the numbers suggest - this is a direction of future study [19]. Nevertheless the FN rate is a very useful metric.

The FP rate of our system is higher. A high FP rate seems like a hindrance to navigation because the robot might see objects where there are none. This would indeed be the case if false positives were distributed randomly across the safety map. However, as the movie shows, almost all of the FPs occur adjacent to existing obstacles - that is most FPs are caused by obstacles "bleeding" into nearby regions - and that most false positives disappear as the robot comes closer to the obstacles. The high FP rate also indicates the noisy nature of stereo range data.

As the ROC curve shows (Fig. 6(a)) we can reduce the FP rate by changing the decrement parameter but at the expense



(a) (b) (c)

(d) (e) (f)

Fig. 7: Figures (a) and (d) show sample images from datasets 2 and 3. Figures (b) and (e) show stereo safety maps in the process of being constructed by the robot (i.e., after the robot has processed only a fraction of the images in the datasets). The robot is shown as a red triangle. Figures (c) and (f) show the ground truth safety maps for comparison - areas that are in common with the stereo safety maps on the left are highlighted using circles. The annotations used are the same as those in Fig. 4(a) except that *Potential Drop-off Edges* are not marked (figure better viewed in color).

of increasing FN rates. Additional ways to overcome such errors would be to fit surfaces to all the point cloud data much like the way we have fit planes to ground points. Another method would be to use active sensing - as we see in the movie, false positives disappear as the robot moves closer and more data is obtained.

B. Detection Latency and Distance

The use of the occupancy grid for noise filtering leads to latencies in hazard detection. However, such filtering is necessary because in its absence the robot may be too "paralyzed with fear" to move. To evaluate the effect of filtering we measure *Frame Latency*, defined as the number of camera frames between the appearance of an object in a video sequence and its detection by the robot. To make the notion of detection concrete, we defined it to be the event when $N\%$ of the width of the board is visible in the occupancy grid.

The Frame Latency depends on object texture and the initial distance of the object from the robot. To measure latency as a function of these quantities we drive the robot towards two boards, one textured and one without texture, placed at various initial distances from the robot. The results are plotted in Fig. 6(b).

As the figure shows, when the board is at an initial distance of 5 meters or less, the robot is able to detect $N = 50\%$ of both textured and untextured boards within 8 frames. However, for $N = 90\%$, it takes significantly more time for the untextured board to be detected as expected. In fact, the robot reaches the board before 90% of it is detected. This appears like it can be a bit dangerous for the robot, but what happens is that different parts of the untextured board are detected, making it appear as an impassable obstacle nevertheless.

It is worth noting that as the initial distance to the object increases from 4 to 5 meters, the number of frames taken to detect the board jumps for $N = 90\%$. This means that it can take a long time before objects at a distance of more than 4 meters are seen properly with our camera. This provides an experimental justification for the use of a restricted planning radius in Section V-A as analysis of objects far away is going to be unreliable.

C. Plane Fitting Accuracy

We evaluate the accuracy of the plane fitting process by comparing the detected planes against ground truth planes obtained using laser range data. We compute: (i) the angle between the normal of a detected plane and the GT plane, and, (ii) the average distance between the detected and the GT plane. From each of the five data sets, we randomly choose 5 frames and compute the above measures for all safe planes detected in those frames. We average across all planes and all data sets. The mean and standard deviation for the angle are found to be: $\mathbf{1.2} \pm \mathbf{0.5}$ degrees. The mean and standard deviation for the distance are found to be: $\mathbf{1.9} \pm \mathbf{0.8}$ centimeters. This shows that plane fitting works very well and that we are able to accurately estimate the normal and location of ground surfaces. Fig. 8 shows examples of planes found using stereo compared to laser data.

D. Computation Time

We implemented a real-time version of our algorithms on the wheelchair robot. The average computation times per

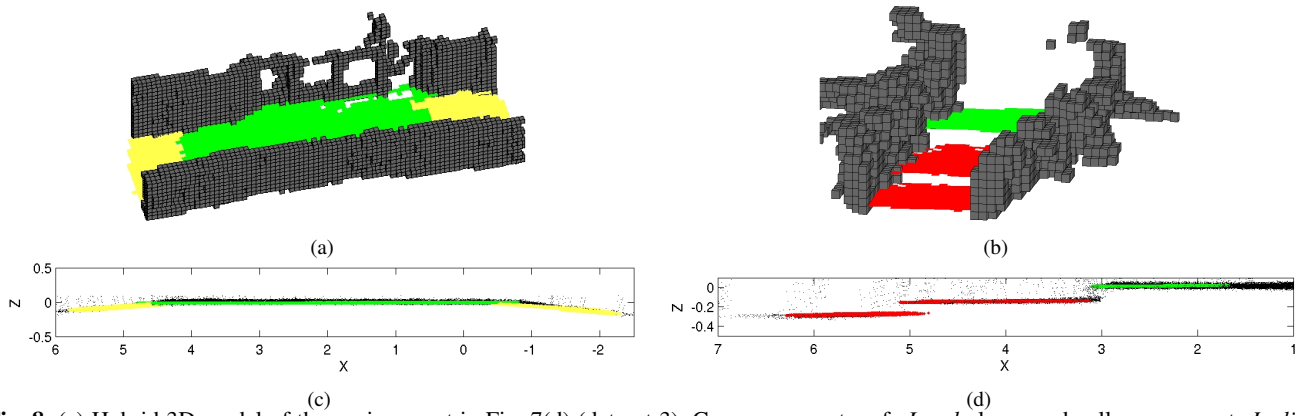


Fig. 8: (a) Hybrid 3D model of the environment in Fig. 7(d) (dataset 3). Green represents safe *Level* planes and yellow represents *Inclined* planes. (c) Cross sectional view of the planes in the model compared to laser range data shown as black dots. (b) Hybrid 3D model of dataset 4 with long low steps each about 15 cm high. The planes in red are found to be unsafe and marked as such - this demonstrates the system's ability to detect even small drop-offs. (d) Cross-sectional view of the planes. As can be seen from (c) and (d) the system does very well at finding *Level* and *Inclined* planes and distinguishing *Inclined* planes from steps (figure better viewed in color).

stereo frame of the algorithms for the real-time version for a $10 \times 10 \times 3$ m^3 map size are as follows: (i) Updating 3D model: 92ms. (ii) Grid segmentation: 4ms. (iii) Least squares fit: 61ms. (iv) Safety analysis: 71ms. This corresponds to an average cycle rate of 4.4Hz. The algorithms were run on a laptop with a 1.83 GHz dual core processor which was simultaneously running a laser-based SLAM algorithm and the GUI. The stereo depth computation algorithm was run on a 1.2 GHz machine on the robot at an average cycle rate of 9Hz.

VIII. CONCLUSION AND FUTURE WORK

We have presented a stereo vision based system for finding inclines, drop-offs, and static obstacles. Finding such hazards is very important if robots are to navigate autonomously. We have comprehensively evaluated our system on five different datasets and demonstrated good performance. We have made these datasets and the corresponding ground truth data publicly available so as to provide a common testing ground for other robot mapping systems [6].

Our results suggest several avenues for further research. We plan to estimate the true chances of failure that false negative error rates translate to. We would also like to explore the use of active sensing to further reduce error rates and the effect of stereo noise. Another possible method for reducing the effect of noise is by fitting surfaces to the entire point cloud data not just to points representing the ground. However, that is a much harder problem as obstacles and other objects can have complex topologies, e.g., due to the presence of holes.

IX. ACKNOWLEDGMENTS

This work has taken place in the Intelligent Robotics Lab at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Intelligent Robotics lab is supported in part by grants from the Texas Advanced Research Program (3658-0170-2007), and from the National Science Foundation (IIS-0413257, IIS-0713150, and IIS-0750011). The authors would like to thank members of the Intelligent Robotics Lab for their valuable comments.

REFERENCES

[1] S. Thrun and et al., "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, 2006.
[2] A. Murarka, M. Sridharan, and B. Kuipers, "Detecting obstacles and drop-offs using stereo and motion cues for safe local motion," in *IROS*, 2008.
[3] N. Heckman, J. Lalonde, N. Vandapel, and M. Hebert, "Potential negative obstacle detection by occlusion labeling," in *IROS*, 2007.
[4] P. Beeson, A. Murarka, and B. Kuipers, "Adapting proposal distributions for accurate, efficient mobile robot localization," in *ICRA*, 2006.
[5] A. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3D visual odometry," in *ICRA*, 2007.
[6] A. Murarka and B. Kuipers, 2009. [Online]. Available: http://eecs.umich.edu/∼kuipers/research/pubs/Murarka-iros-09.html
[7] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "3D perception and environment map generation for humanoid robot navigation," *Intl. Journal of Robotics Research*, 2008.
[8] R. Rusu, A. Sundaresan, B. Morisset, M. Agrawal, and M. Beetz, "Leaving flatland: Realtime 3D stereo semantic reconstruction," in *ICIRA*, 2008.
[9] L. Iocchi, K. Konolige, and M. Bajracharya, "Visually realistic mapping of a planar environment with stereo," in *ISER*, 2000.
[10] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, and K. Schwehr, "Recent progress in local and global traversability for planetary rovers," in *ICRA*, 2000.
[11] A. Rankin, A. Huertas, and L. Matthies, "Evaluation of stereo vision obstacle detection algorithms for off-road autonomous navigation," in *32nd AUVSI Symposium on Unmanned Systems*, 2005.
[12] ——, "Night-time negative obstacle detection for off-road autonomous navigation," in *SPIE*, 2007.
[13] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," in *IROS*, 2008.
[14] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard, "A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots," *IEEE Transactions on Robotics*, 2004.
[15] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "Learning predictive terrain models for legged robot locomotion," in *IROS*, 2008.
[16] C. Wellington, A. Courville, and A. Stentz, "Interacting markov random fields for simultaneous terrain modeling and obstacle detection," in *Robotics: Science and Systems*, 2005.
[17] "Videre Design," http://www.videredesign.com.
[18] K. Konolige, "Improved occupancy grids for map building," *Autonomous Robots*, vol. 4, no. 4, 1997.
[19] A. Murarka, "Building safety maps using vision for safe local mobile robot navigation," Ph.D. dissertation, The University of Texas at Austin, 2009.