

Positive and Negative Obstacle Detection using the HLD Classifier

Ryan D. Morton and Edwin Olson

Abstract—Autonomous robots must be able to detect hazardous terrain even when sensor data is noisy and incomplete. In particular, negative obstacles such as cliffs or stairs often cannot be sensed directly; rather, their presence must be inferred. In this paper, we describe the height-length-density (*HLD*) terrain classifier that generalizes some prior methods and provides a unified mechanism for detecting both positive and negative obstacles. The classifier utilizes three novel features that inherently deal with partial observability. The structure of the classifier allows the system designer to encode the capabilities of the vehicle as well as a notion of risk, making our approach applicable to virtually any vehicle. We evaluate our method in an indoor/outdoor environment, which includes several perceptually difficult real-world cases, and show that our approach out-performs current methods.

I. INTRODUCTION

Robots in complex environments encounter hazardous terrain in the form of positive and negative obstacles. Positive obstacles are objects that extend ‘up’ from the ground, such as walls. Negative obstacles occur when the ground drops off, presenting a falling or tipping hazard for the robot, i.e. the edge of a cliff (see Fig. 1). The terrain classification task is to detect these obstacles and output a map, typically a 2D grid, that will be used by the robot’s path planner as the final word regarding the safe/dangerous regions in the robot’s vicinity.

In the process of detecting obstacles and drivable terrain, robots have incomplete information regarding the surrounding terrain due to sensor sparseness and occlusions (see Fig. 2). In fact, negative obstacles are not directly observable from most sensor configurations and must be inferred; for example, by using the change in height and the intervening distance between the two closest observations, from the upper and lower surfaces; we term these the h and l features, respectively. Additionally, the data density returned from most sensors varies according to the geometry of the sensor configuration, e.g., see the 3D point cloud’s line-like structure in Fig. 2. This leads to the third feature, d , based on the point density in the vicinity of each discretized cell.

We present a new terrain classifier that uses the h , l , and d features to detect both positive and negative obstacles even with incomplete information. The classifier generalizes some previous methods, which we show are special cases of the

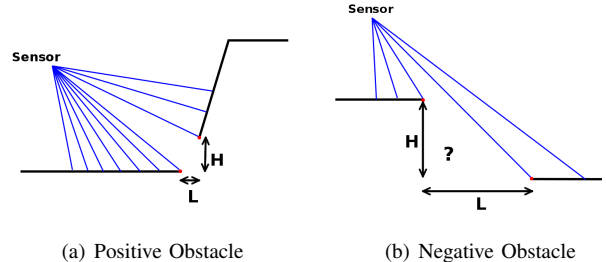


Fig. 1. Positive vs. Negative Obstacles. (a) A positive obstacle extends upward from the ground while in (b) the negative obstacle drops off and occludes the immediately adjacent terrain. Both obstacle types can be parameterized by h and l .

proposed *HLD* classifier. The primary contributions of this paper include a new terrain classifier that:

- Detects both positive and negative obstacles
- Handles partial observability
- Gives designers ability to manage risk via a notion of classification confidence
- Subsumes and outperforms previous methods

In the next section we discuss related work in terrain classification. The h , l , and d features are described in detail in Section III. The *HLD* classifier and confidence-based classification are shown in Section IV, followed by an evaluation in Section V.

II. RELATED WORK

Many robot terrain classification methods detect obstacles using features similar to h , l , and d . The first method, termed *bucket model*, discretizes the world into cells and analyzes the data within each cell independently [1], [2], [3]. The simplest such method thresholds the h within a cell. Although more complicated statistics could be used, e.g., variance or plane-normal direction, none allow inter-cell

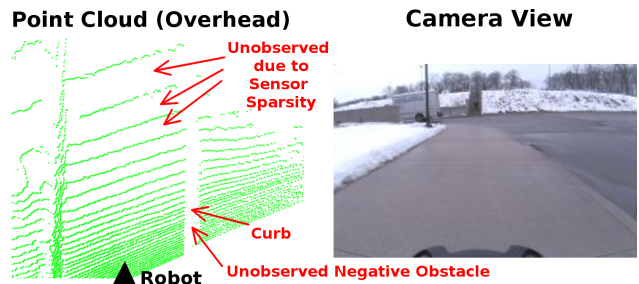


Fig. 2. Partial Observability. The 3D point cloud (left) shows how unobserved regions come from both sensor sparsity and occlusions, such as the untraversable 12 cm curb (negative obstacle).

Ryan D. Morton is a Student rmorton@umich.edu
Edwin Olson is an Assistant Professor ebolson@umich.edu
Computer Science and Engineering, University of Michigan, 2260 Hayward Street, Ann Arbor, Michigan

This research was supported in part by the Ground Robotics Reliability Center (GRR) at the University of Michigan, with funding from government contract DoD-DoA W56H2V-04-2-0001 through the US Army Tank Automotive Research, Development, and Engineering Center.

UNCLASSIFIED: Dist. A. Approved for public release

inference and thus cannot detect negative obstacles unless the grid cells are excessively large. However, increasing the grid cell size causes obstacles to dilate and grow out into drivable terrain. Minimization of obstacle dilation is a major criteria of the terrain classification task, as robots must often work in already constricted environments, e.g., a 50 cm wide robot attempting to fit through a 70 cm doorway.

Inter-cell inference can be added with a simple extension to the bucket model by thresholding the h between nearby cells. We term this method the *flat model* approach because it ignores l (distance has no impact on the classification) and, thus, models terrain as flat [4], [5]. The *constant slope* model uses both the h and l features to model drivable terrain as a linear function ($h_{max}(l) = ml + b$ for some constants m and b), thus, allowing inclined terrain [6], [7]. These binary classification approaches allow inference between cells and thus, through unobserved regions, but fail to model the risk associated with doing so.

A notion of confidence has been used before in a few areas of terrain classification. For example, in [8] a notion of confidence is used based on color features. In [9], confidence is associated with a complete feature, for example during learning, rather than during the classification process as used here. In [10], kernels and visibility are used to classify rough terrain by bounding the best and worst case for terrain in unobserved regions.

Other methods look at the drivability of occlusions for terrain analysis. Potential negative obstacles are classified by modeling occlusions and propagating information through unobserved regions via an expensive ray-casting operation in [11]. However, the robot must wait until it is closer to conduct a mobility assessment and has separate detectors for positive and negative obstacles. The *HLD* classifier allows analysis that is agnostic to the position of the robot and handles both positive and negative obstacles through explicit use of unobserved regions. A principal component analysis based approach to obstacle detection is used in [12] with a measure of surface roughness. Although they use a notion of distance disturbance, h , l , and d are not used.

Some robot systems have been deployed in constrained environments that simplify the terrain classification problem. For example, work in the high-speed road driving arena has allowed robotic cars to drive successfully on roadways [1], [2], [13], [14]. However, a robot traveling between the lane markers on a maintained road can assume that it will not encounter a dangerous negative obstacle. Similarly, indoor-only robots generally only detect positive obstacles [15], [16] or move so slowly that they can simply look down with a proximity sensor to detect negative obstacles. Plane-fitting techniques ([17], [18]) have been shown to work well in environments where planes are good approximators of the underlying terrain, but often have difficulty with negative obstacles. We desire a classifier for environments with missing information, no road signs, and dangerous negative obstacles.

III. THE *HLD* FEATURES

The steps to classify a 3D point cloud with the *HLD* algorithm are 1) discretize the world, 2) run the feature detectors on each cell, and 3) classify each cell based on the model. Each 3D point is added to the respective 2.5D cell in the discretization while each cell keeps track of the number of observations and its observed z_{min} , z_{max} , and z_{avg} . We use the terms z and z_{avg} interchangeably.

Each cell has a single d feature value, which measures the point density near the cell. However, a cell may have multiple (h, l) pairs, each corresponding to a path, of length l , through the cell with a $h = \Delta height$ between two nearby observed cells.

A. Message Passing and the h and l Features

We propagate information to account for missing data in the world, thus, allowing inference using observations from nearby cells. The h and l features require cells to receive information from nearby cells in order to infer the terrain between them. The actual information propagated between cells always originates as the z_{avg} value of an observed cell and only propagates between direct neighbors and through unobserved regions, see Fig 3(c).

Every observed cell will generate the (h, l, d) tuple ($z_{max} - z_{min}, 0, d$) to later be used for classification. In addition, the cell initiates message passing by sending its z_{avg} value to its neighbors. The message passing scheme is broken into two steps for computational efficiency. First, information propagates once from each observed cell to its 4-connected neighbors. Next, $s - 1$ iterations of information propagation from each unobserved cell to its 4-connected neighbors. The number of propagation steps s is set by the designer for the desired propagation distance in order to ensure geometric detection of expected obstacles, e.g., ability to detect a 12 cm curb from a distance of 1.5 m. See Algorithm 1 for the pseudo-code for the message passing scheme.

Every cell on the propagated path between cells A and B needs to know $h = |z_{avg_A} - z_{avg_B}|$ and l , which equals

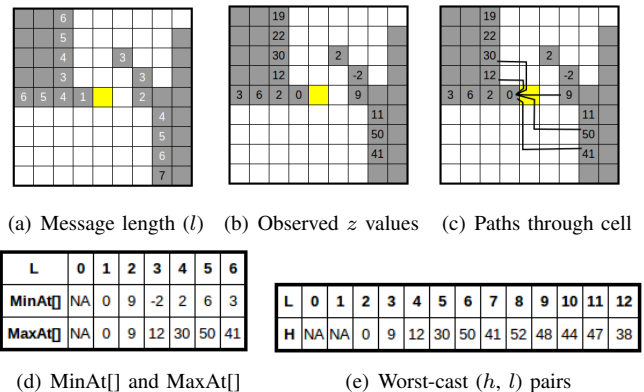


Fig. 3. Message Passing Example (gray = observed, white = unobserved, yellow = cell in question). Propagation distances are shown in (a) (white font indicates observed cells within 6 propagation steps) and z values in (b). (c) shows a subset of information paths through the yellow cell. (d) and (e) show the information that is propagated to the yellow cell.

Algorithm 1 Message passing for h and l features

```
1: {Initialize and send from observed cells}
2: for each observed cell  $c$  do
3:    $c.minAt[0] = c.z_{avg}$ 
4:    $c.maxAt[0] = c.z_{avg}$ 
5:   for each neighbor cell ( $n$ ) of  $c$  do
6:      $n.minAt[1] = \min(n.minAt[1], c.minAt[0])$ 
7:      $n.maxAt[1] = \max(n.maxAt[1], c.maxAt[0])$ 
8:   end for
9: end for
10: {Propagate through unobserved cells}
11: for  $i = 2 : \text{number of propagations}$  do
12:   for each unobserved cell  $c$  do
13:     for each  $n \in (4\text{-connected})$  neighbor of  $c$  do
14:        $n.minAt[i] = \min(n.minAt[i], c.minAt[i-1])$ 
15:        $n.maxAt[i] = \max(n.maxAt[i], c.maxAt[i-1])$ 
16:     end for
17:   end for
18: end for
```

the propagation distance based Manhattan distance but only through unobserved regions. But, to efficiently implement the message passing algorithm, the only information propagated between cells is the maximum and minimum observed z values at each discretized distance from the cell. For example, the yellow cell in Fig. 3(a) must know both the minimum and maximum z values among all the observed cells 1 unit away; similarly for distances of 2, 3, etc. as shown in Fig. 3(d). Thus, each cell needs to store two arrays representing the minimum and maximum z values at each discretized distance; we call these $minAt[i]$ and $maxAt[i]$, respectively. Then the cell's actual value for h at a distance of l is

$$h(l) = \max_{i+j=l} (maxAt[i] - minAt[j]) \quad (1)$$

Thus, with s propagation steps, a cell may have up to $2s$ total (h, l, d) tuples, where the d is the same in each. The complexity of each full round of message passing is $O(sn)$, where n is the number of cells.

B. d Feature

The density feature at a cell should be a function of the observations in the region around the cell. We found that the best formulation for d is the sum of the nearby observations (out to the propagation radius of the message passing scheme) weighted by a zero-mean Gaussian distribution on distance, parameterized by σ .

C. Discretization Size

The underlying terrain is best fit with small grid cells, yet often detection capabilities decrease as the cells get smaller. Smaller cells result in more cells, more inter-cell inferences, and, generally, more unobserved cells. However, since the actual distances between observations do not change (except for discretization errors) the detections with the *HLLD* classifier are not affected. Thus, the only effects of grid cell size

on the *HLLD* terrain classifier are computational. Specifically, an arbitrarily small grid cell can be used (within computation limits).

IV. *HLLD* CLASSIFICATION

A. Binary Classification with a Confidence

Each 3D sweep is analyzed and produces a terrain map. These maps are combined by compositing consecutive maps into a larger map, which is sent to the path planner. This final map can be easily thresholded into a binary map, detailing each cell as drivable or as an obstacle. The individual maps must be aligned via the simultaneous localization and mapping (SLAM) system but they need not be binary themselves. In fact, if they are binary then the composition step is very tricky or trivial, e.g., always picking most recent data over older data. Instead, we adopt a notion of confidence to augment the binary classification for each cell. To accomplish this we propose a real valued output at each cell; positive for obstacles and negative for drivable terrain, with magnitude signaling the confidence. The notion of confidence affects two distinct pieces of the overall system: compositing the individual terrain maps into the final output map and during classification of individual cells from a single 3D sweep.

The idea behind using confidence during composition is that when deciding between two classifications for a cell, each from a different 3D sweep, we accept more confident one. For example, if a cell is classified as drivable with confidence 80% from one 3D sweep but the next sweep declares it as an obstacle with 1% confidence, we keep the former and output the cell as drivable (see Fig. 4).

The confidence is also used to classify individual cells from a single 3D sweep due to each cell having multiple (h, l, d) tuples. The cell is annotated pessimistically, via the max operator, based on the values in the model. The max operator ensures the safety of the robot by taking any hazardous evidence over evidence of drivability. Thus, individual 3D scans mark cells pessimistically (the most unsafe evidence

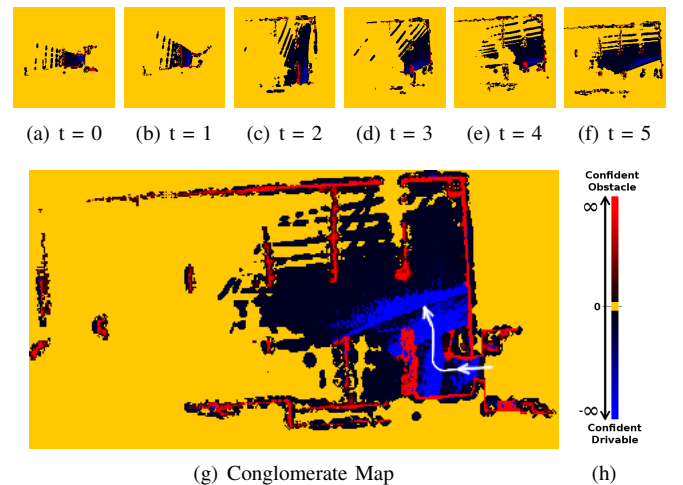


Fig. 4. Map Composition. The sequential maps (a)-(f) are merged into a composite map (g) using the single scan confidence values (color scheme shown in (h) with the robot trajectory in white).

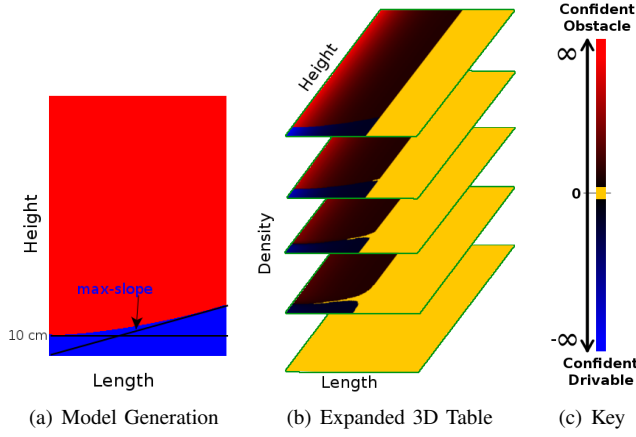


Fig. 5. *HLD* Classification Model. (a) Tuning the *HLD* model is based on the kinematics of the robot. (b) The full 3D table comes from the kinematics and confidence losses due to changes in l and d .

in that scan) while the compositing process uses the most confident evidence.

B. Classification Model

Designers have the ability to tailor the *HLD* terrain classifier specifically to their robot, risk level, and environment via the classification model. The actual encoding of the model can take many forms such as a parametric function, 3D table, or some other factorization. The classification model represents the classification and confidence throughout the feature-space.

For this work we used a 3D table with real values, notated as $HLD(h, l, d)$, for example the graphical depiction shown in Fig. 5(b). The generation of the model comes primarily from the kinematics of the robot. For example, a single *HL* slice of the table may look like Fig. 5(a) if a robot can go over 10cm curbs and handle terrain up to some max-slope. The complete 3D table can be generated by copying this slice or by a more complicated mechanism. In Sec. V-A we describe the generation of our model in more detail. The process of *HLD* model generation as explained here requires an expert. However, fundamentally it is a direct consequence of the vehicle capabilities. The slice shown in Fig. 5(a) would result in a binary classifier and the expert need only modify the model to account for the confidence in the binary classification w.r.t. changes in l and d .

Interestingly, the bucket, flat, and constant slope models are special cases of the *HLD* classifier. These methods are binary by convention, produce no confidence estimate, and do not depend on d . Thus, the slices shown for each method's model in Fig. 6 are the same for all values of d .

C. Classifying Individual Cells

After detecting the h , l , and d features at each cell we use the model to classify the cell. The only special case is when the density is zero (no observations nearby) resulting in a value of 0; zero confidence. For all other cells, the assigned value is computed via direct observations and messages received during message passing. All the paths through cell

c and their values can be computed using the initialized values in the $\text{minAt}[]$ and $\text{maxAt}[]$ arrays at the cell. The features are $l = i + j$ and $a = h = c.\text{maxAt}[j] - c.\text{minAt}[i]$. Thus, the value for a particular path is $HLD(c.\text{maxAt}[j] - c.\text{minAt}[i], i + j, c.d)$, see Algorithm 2. The cell's final value is the maximum, or most unsafe, of the individual path values.

Algorithm 2 Classification

```

1: for each cell c do
2:   if c.d == 0 then
3:     c.val = 0
4:     continue
5:   end if
6:   c.val = -∞
7:   if c.observed then
8:     c.val = HLD(c.zmax - c.zmin, 0, c.d)
9:   end if
10:  {Pessimistically mark cell from (h, l, d) tuples}
11:  for each initialized c.minAt[i], c.maxAt[j] pair do
12:    val = HLD(c.maxAt[j] - c.minAt[i], i + j, c.d)
13:    c.val = max(c.val, val)
14:  end for
15: end for

```

V. EVALUATION

To evaluate the *HLD* method we first describe our testing apparatus (our robot) and give additional details about the alternate methods. Then we explain some perceptually challenging environments in order to illustrate the qualitative differences between the methods. Finally, we show a quantitative error analysis with a ground truth dataset.

A. Testing Apparatus and Method Parameters

The 4-wheeled skid-steered robots used for evaluation can safely traverse 20° inclines and overcome 10 cm vertical discontinuities. The robot uses an actuated 2D LIDAR (Hokuyo 30LX) that returns a 3D point cloud every 1.25 seconds (see Fig. 7). This particular sensor configuration gives line-like returns, which often contain large unobserved regions, even on flat ground.

We discretize the world into 5 cm cells and produce 15×15 m maps, centered about the robot, from each 3D sweep. The same discretization is used for the composite map sent to the path planner. We set the propagation distance to 40 cm (or 8 steps) for the feature detectors and set $\sigma = 0.1$ m for the d feature.

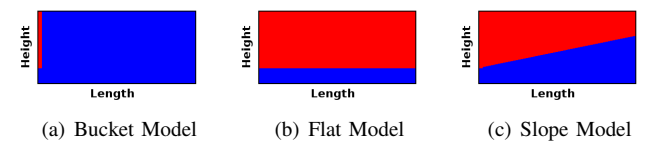


Fig. 6. Alternate Method Classification Models. Showing a single slice of possible *HLD* representations for the bucket, flat, and slope models; none have a dependence on d . (blue = drivable, red = obstacle)

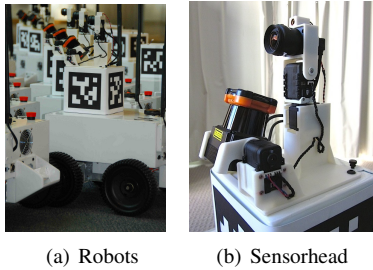


Fig. 7. Robot Testbed. The robots (a) use an actuated 2D LIDAR (b) which cause the line-like structure shown in Fig. 2.

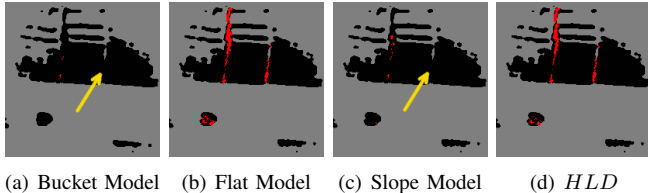


Fig. 8. Negative Obstacle - Curb. (gray=unknown, black=drivable, and red=obstacle). The bucket and constant slope models fail to detect the 12 cm negative obstacle. (scene shown in Fig. 2)

For our *HLD* model we use a 3D table discretized h into 250 bins $\in [0, 25\text{ cm}]$, l into 200 bins $\in [0, 50\text{ cm}]$, and d into 100 bins $\in [0, 1]$. The kinematics of the robot and the designer specified risk associated with missing information were used to generate the hand-tuned model, as shown in Fig. 5. To generate our model we began by simply putting the physical abilities of the robot into an *HL* slice, as shown in Fig. 5(a). We then decaying the confidence linearly with l to create the $d = 1$ slice (top slice shown in Fig. 5(b)). Finally, the additional slices (for values of d) were generated by decaying the confidence quadratically w.r.t. d . The decrease in confidence associated with l and d depended upon the capabilities of our robot, our sensor configuration, and our confidence in the particular region of the feature-space.

We compare our tuned *HLD* model against the bucket, flat, and constant slope models. Since these are special cases of the *HLD* classifier, we implement them with the models depicted in Fig. 6. Each of these models use a h threshold of 10cm and the constant slope model uses 20° for the additional slope parameter.

B. Qualitative Classification Comparison

We compare these methods on a variety of indoor/outdoor environments that include both positive and negative obstacles. Qualitatively, the algorithms all properly detect large vertical surfaces (e.g. walls) and flat surfaces near the robot (e.g. sidewalks). However, negative obstacle and inclined terrain handling differed between the algorithms.

Negative obstacles must be detected at a safe distance or else the robot could be endangered. Thus, for our robot, we need to detect the negative obstacle before the robot is within 0.5m (the distance traveled during one 3D sweep). The bucket model did not detect a single negative obstacles at this distance. Small, yet still dangerous, negative obstacles

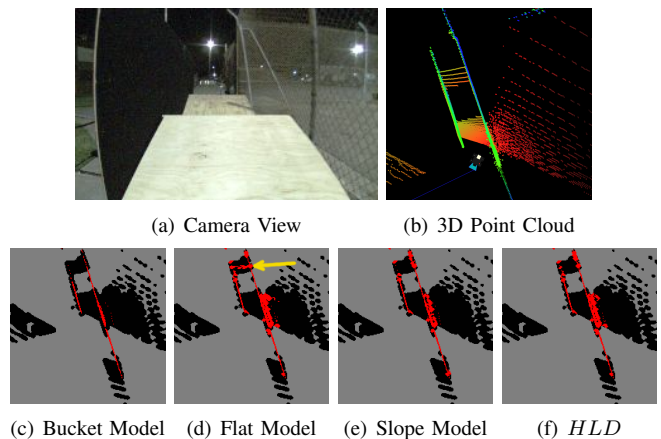


Fig. 9. Inclined Terrain (15° ramp). The flat model hallucinates obstacles, while the other methods properly mark the ramps as drivable. The colors of the points in (b) represents various heights in the overhead view.

were correctly classified by both the flat model and *HLD* method but not by the constant slope method. For example, the constant slope method could not detect the 12 cm curb in Fig. 8, because the unobserved region was approximately 20 cm wide and at that distance only curbs taller than $(\tan(20^\circ)20 + 10) = 17\text{ cm}$ can be detected. The slope model can only detect the curb as an obstacle when the robot is 18 cm from the curb (for our sensor position), but this is too late for an autonomous robot with a scanning sensor. The constant slope method can be tuned to detect the curb by lowering either the 20° slope or the 10 cm vertical obstacle threshold, which causes other undesirable side-effects as explained below.

We tested the algorithms on inclined terrain, specifically ramps with slopes up to 15° (see Fig. 9). These traversable inclines are challenging for the flat model. The constant slope model can handle the inclines, but not with the values tuned for the 12 cm curb. The performance of the bucket method on ramps is very sensitive to discretization size; the algorithm worked with 5 cm cells, but hallucinations prevented traversal with 10 cm cells. However, the 5 cm cells lowered the obstacle detection rate elsewhere (e.g. curbs and small obstacles). The *HLD* method successfully classifies the 15° inclines and begins detecting slopes as obstacles when the slope increases toward 20° .

Overall, only the *HLD* method handles each of these perceptually challenging situation with a single model. Each of the other methods could be tuned to handle individual challenges, however, no single parameter tuning could handle

TABLE I
QUALITATIVE ANALYSIS ON PERCEPTUALLY DIFFICULT TERRAIN

	Flat	Walls	12 cm Neg. curb	15° ramp
Bucket model	Yes	Yes	No	Yes
Flat model	Yes	Yes	Yes	No
Slope model	Yes	Yes	No	Yes
<i>HLD</i>	Yes	Yes	Yes	Yes

TABLE II
QUANTITATIVE ANALYSIS ON ALL DATASETS

	Obstacle		Drivable	
	True	False	True	False
Bucket model	0.91	0.09	0.74	0.24
Flat model	0.78	0.22	0.85	0.15
Constant Slope	0.84	0.16	0.83	0.17
<i>HLD</i> model	0.87	0.13	0.97	0.03

the complete set, see Table I.

C. Quantitative Classification Comparison

We hand-labeled 30 ground truth maps from data gathered around the University of Michigan campus. See Table II for the class accuracy data for the 2.7 million data points from these 30 labeled maps.

The bucket model only marks obstacles detectable within single cells, which are generally vertical in nature. Because of the small grid size (5 cm) these typically correlate with actual obstacles in the real world. Thus, the fact that the bucket model has the highest true-obstacle rate is expected. Each of the other methods infer obstacles between cells and, thus, have higher error rates on obstacle detections, due to the inevitable incorrect inferences. The error rates for drivable terrain show that the tuned *HLD* model greatly outperforms the other models with only 3% error. This result is also expected based on how closely the model reflects the actual abilities of the robot.

D. Run-time Analysis

Since we implemented the previous methods via the *HLD* terrain classifier shown in this paper, we do not provide run-time analysis for those methods. However, the *HLD* method completes the terrain analysis for 5 cm cells over a 15×15 m area in 280 ms (average). These test ran on a Core 2 Duo processor at 2.4Ghz with 4GB of RAM running single-threaded in Java. The run-time compared nearly one-to-one with the 260 ms classifier used for Team Michigan's MAGIC 2010 classifier, which was based on the constant slope model and thus did not detect negative obstacles. Further run-time optimizations are desired, but for a small time premium our system gained the ability to detect negative obstacles.

VI. CONCLUSION

The *HLD* terrain classifier defines a new feature-space that allows detection of both positive and negative obstacles using a unified mechanism. The h , l , and d features are efficiently computed from 3D point cloud data on a robot operating in complex indoor/outdoor urban environments. Additionally, the new method generalizes and outperforms common approaches on several common mobile robot terrain topologies and this illustrates the versatility of our method. A single *HLD* model, properly set for the capabilities and risk level for our robot, was sufficient to handle these perceptually difficult terrain classification problems. However, this expressivity comes at a cost of need to specify the model, which currently requires an expert.

REFERENCES

- [1] F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H. Wuensche, "Driving with tentacles: Integral structures for sensing and motion," *Journal of Field Robotics*, vol. 25, no. 9, pp. 640–673, 2008.
- [2] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams, "A perception driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, September 2008.
- [3] D. Kim, J. Sun, S. Oh, J. Rehg, and A. Bobick, "Traversability classification using unsupervised on-line visual learning for outdoor robot navigation," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 518–525.
- [4] T. Hong, M. Abrams, T. Chang, and M. Shneier, "An intelligent world model for autonomous off-road driving," *Computer Vision and Image Understanding*, 2000.
- [5] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin, "Terrain perception for DEMO III," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 326–331.
- [6] L. Matthies, A. Kelly, T. Litwin, and G. Tharp, "Obstacle detection for unmanned ground vehicles: A progress report," in *Intelligent Vehicles '95 Symposium., Proceedings of the*. IEEE, 1996, pp. 66–71.
- [7] H. Schafer, A. Hach, M. Proetzsch, and K. Berns, "3d obstacle detection and avoidance in vegetated off-road terrain," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, pp. 923–928.
- [8] J. Lalonde, N. Vandapel, D. Huber, and M. Hebert, "Natural terrain classification using three-dimensional lidar data for ground robot mobility," *Journal of Field Robotics*, vol. 23, no. 10, pp. 839–861, 2006.
- [9] M. Hebert and N. Vandapel, "Terrain classification techniques from lidar data for autonomous navigation," in *Collaborative Technology Alliances Conference*. Citeseer, 2003.
- [10] R. Hadsell, J. Bagnell, and M. Hebert, "Accurate rough terrain estimation with space-carving kernels," in *Proc. of Robotics: Science and Systems (RSS)*. Citeseer, 2009.
- [11] N. Heckman, J. Lalonde, N. Vandapel, and M. Hebert, "Potential negative obstacle detection by occlusion labeling," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 2168–2173.
- [12] F. Neuhaus, D. Dillenberger, J. Pellenz, and D. Paulus, "Terrain drivability analysis in 3D laser range data for autonomous robot navigation in unstructured environments," in *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*. IEEE, 2009, pp. 1–4.
- [13] S. Thrun, M. Montemerlo, and A. Aron, "Probabilistic terrain analysis for high-speed desert driving," in *Proceedings of the Robotics Science and Systems Conference, Philadelphia, PA*. Citeseer, 2006.
- [14] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J. P. Gonzalez, J. Gowdy, A. Gutierrez, S. Harbaugh, M. Johnson-Roberson, H. Kato, P. L. Koon, K. Peterson, B. K. Smith, S. Spiker, E. Tryzelaar, and W. R. L. Whittaker, "High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004," Robotics Institute, Pittsburgh and PA, Tech. Rep. CMU-RI-TR-04-37, June 2004.
- [15] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation* 1," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.
- [16] H. Surmann, A. N. "luchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3-4, pp. 181–198, 2003.
- [17] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Autonomous Robots*, vol. 18, no. 1, pp. 81–102, 2005.
- [18] A. Murarka and B. Kuipers, "A stereo vision based mapping algorithm for detecting inclines, drop-offs, and obstacles for safe local navigation," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1646–1653.