Probabilistic Estimation of Multi-Level Terrain Maps

Cesar Rivadeneyra, Isaac Miller , Jonathan R. Schoenberg and Mark Campbell
Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, USA
Email: {crr62, itm2, jrs55, mc288}@cornell.edu

***Abstract*— Recent research has shown that robots can model their world with Multi-Level (ML) surface maps, which utilize 'patches' in a 2D grid space to represent various environment elevations within a given grid cell. Though these maps are able to produce 3D models of the environment while exploiting the computational feasibility of single elevation maps, they do not take into account in-plane uncertainty when matching measurements to grid cells or when grouping those measurements into 'patches.' To respond to these drawbacks, this paper proposes to extend these ML surface maps into Probabilistic Multi-Level (PML) surface maps, which uses formal probability theory to incorporate estimation and modeling errors due to uncertainty. Measurements are probabilistically associated to cells near the nominal location, and are categorized through hypothesis testing into 'patches' via classification methods that incorporate uncertainty. Experimental results comparing the performances of the PML and ML surface mapping algorithms on representative objects found in both indoor and outdoor environments show that the PML algorithm outperforms the ML algorithm in most cases including in the presence of noisy and sparse measurements. The experimental results support the claim that the PML algorithm produces more densely populated, conservative representations of its environment with fewer measurements than the ML algorithm.**

I. INTRODUCTION

As military and commercial applications of field robotics become more prevalent, there is a growing need to improve a mobile robot's ability to understand and map its environment. For example, when autonomous robots are sent into a collapsed building in search of survivors or put in charge of exploring a building that might house explosives or hostile enemies, they must be able to navigate through an environment that is largely unknown *a priori*. Perception is a critical precursor to planning because successful navigation relies on accurate modeling and understanding of the environment.

In the past, there has been a surge of methods aimed to empower robots to map complex indoor and outdoor environments. The simplest, yet most informative method is to represent the environment with binary 3D grids that divide the robot environment into voxels. The environment is then defined by each of its voxels' occupancy status. This method is implement in [1] using a stereo camera to gather disparity and depth information about an indoor environment. [2] utilizes this method to solve the simultaneous localization and mapping problem using dual laser scanners. Though voxel representations of the environment are most informative in the sense that they fully-model the 3D space, they suffer from significant computational burdens in maintaining a dense 3D grid that grows with the cube of distance travelled.

Another popular method for representing environment topology is the use of elevation maps. An elevation map represents the space using a 2D grid where each cell stores the dominant height of the terrain contained within the grid cell. This 2D representation has lower computational demands than the voxel representation while still maintaining a rich model of the environment. This method is employed in [3] to build elevation surface models using laser range data to navigate their walking robot. [4] represents the terrain at an excavation site with elevation maps using a laser rangefinder. Though these 2D methods work well in environments where each grid cell has one dominant height, they fail to accurately represent the environment when employed in multi-level environments containing trees, overhangs, tunnels, tables, and chairs. To alleviate these shortcomings, [5] creates an extension for elevation maps to handle objects such as bridges and underpasses in outdoor environments. This extended elevation algorithm allows the map to choose which elevation to keep in its representation, and whether it is the dominant one or not. While this enables planning and navigation under bridges or through tunnels, it is limiting in the sense that the robot must choose which of the multiple elevation levels to represent. As a result, a robot choosing to drive under a bridge would never be able to drive on the bridge at later times.

Miller and Campbell [6] implement a real-time terrain estimator that takes into account common sources of uncertainty to estimate terrain height in a 2D grid. They fit a Gaussian distribution over each measurement's nominal location, and use this probability distribution to generate minimum mean-squared error elevation estimates on neighboring grid cells near the nominal measurement. After these estimates are generated, Bayesian data fusion is used to generate a minimum mean-squared error estimate of the terrain. [6] implemented this algorithm on Cornell University's 2005 DARPA Grand Challenge robot, using laser rangefinders to produce a probabilistic elevation map of the outdoor environment. Though this method suffers from limitations of only having one dominant height in the 2D environment maps, it is the only one that produces true statistical estimates that account for all sources of uncertainty, even across in-plane cells.

In order to provide mobile robots with a broader navigation space, Triebel *et al.* [7] propose a method called Multi-Level (ML) surface mapping for maintaining multiple elevations in the grid space. They maintain 'patches' for each cell, which represent traversable height levels in the

environment at that cell; thus, the environment features are represented using either vertical or horizontal patches depending on the feature. As new measurements are taken, they are either assimilated into older patches or used to generate new ones. This method allows mobile robots to represent multiple elevation levels in the environment simultaneously, without incurring the computational burden of a dense voxel representation. This method, however fails to adopt a formal probabilistic approach, and suffers from several drawbacks as a result:

1) it ignores important sources of uncertainty, such as robot pose and sensor calibration, which may result in significant distortions in terrain estimates.

2) it assumes an arbitrary model for laser height uncertainty instead of mapping errors, which will result in suboptimal estimation.

3) it does not consider in-plane measurement errors, such that the estimate of each terrain feature is suboptimal.

4) it relies on ad hoc clustering algorithms rather than Bayesian decision theory to decide the structure of the terrain (i.e. whether to start a new patch or not).

In order to model the environment accurately, a robot must account for both the multi-level nature of its environment and the measurement uncertainty. In response to this need, this paper proposes a Probabilistic Multi-Level (PML) surface map that extends the ML surface map to include a probabilistic representation of the terrain using formal Bayesian techniques. In particular, the PML surface mapping approach extends the ML surface map to rigorously include multiple sources of uncertainty as in [6] to reduce distortions on the terrain estimates. In addition, the approach considers in-plane measurement uncertainty to perform optimal terrain estimation on the locations of terrain features in addition to the elevations. Once elevation estimates for the various cells are calculated, the PML surface mapping algorithm utilizes Bayesian decision theory to classify these elevation estimates into cell patches according to a modified version of the classification methodology in [7]; it will be shown statistically that the horizontal patches can be discarded without loss of performance. Finally, each elevation estimate will be used to modify the properties of the patch in the cell it belongs (i.e. its mean, depth and probability).

This paper is outlined as follows. Section II describes the methodology for creating PML surface maps in two main steps: 1) in-plane probabilistic assignment of measurements into cells and elevation estimate generation and 2) classification of elevation estimates into patches and combination of correlated patches. Section III presents and discusses the experimental results obtained from implementing the ML and PML surface mapping algorithms on generic objects that can be composed to represent both indoor and outdoor environments for varying performance parameters. Finally, the conclusions are presented in Section IV.

II. PML Methodology

The proposed PML surface map augments Triebel's ML surface map through rigorous treatment of sensor errors and formal probability techniques to incorporate elevation estimates into patches and patch combination. First, elevation estimates are generated for each single raw measurement for both the originating cell location and its neighboring cells according to the method described in [6]. This produces a probabilistic elevation estimate $z_k^{i,j} = \mathcal{N}(\hat{U}_k^{i,j}, \sigma^2_{\hat{U}_k^{i,j}})$ for each measurement k for every cell $c_{i,j}$. Each estimate is then either assimilated into a patch within the cell it belongs to or used to create a new patch for that cell. Finally the correlated patches, defined as patches that are spatially and probabilistically similar, are combined to produce the final PML surface map. Notice the proposed PML surface map utilizes two key features: A) incorporation of measurement uncertainties in the probabilistic assignment of measurements to the cells, similar to [6], and B) development of the multiple levels of the map building on the concepts in [7].

A. Measurement Assignment and Multiple-Level Patches

The probabilistic elevation estimates $z_k^{i,j}$ obtained as described in [6] are assigned into patches for cells $c_{i,j}$, in a method similar to [7]. For every cell $c_{i,j}$, a database of one or more patches, each represented by a mean $U_e^{i,j}$, variance $\sigma^2_{U_e^{i,j}}$, depth $d_e^{i,j}$ and probability $p_e^{i,j}$, is maintained for the e^{th} patch as shown in Figure 1. The depth parameter represents the length of the patch below its mean; it is only non-zero for vertical objects. As mentioned earlier, a study was conducted on the need to use a horizontal-and-vertical (HnV) patch representation similar to [7] versus a vertical-only (Vonly) patch representation. An environment map was generated with both representations, and a statistical comparison was made. The results shows that the V-only representation had transformed nearly 97.5% of the horizontal patches from the HnV map into vertical patches with non-zero depth, leaving less than 2.5% as zero-depth vertical patches. As a result it was determined the horizontal patches could be eliminated from the algorithm.

Once an elevation estimate $z_k^{i,j}$ is determined, it must be subjected to the classification procedure in Figure 2.

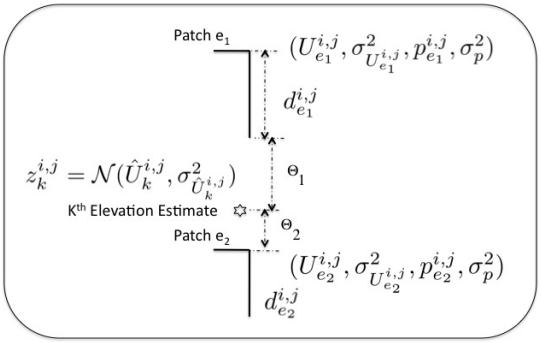

Fig. 1. Elevation estimate and two patches for cell $c_{i,j}$.

This procedure seeks to answer one main question: does the elevation estimate $z_k^{i,j}$ extend a current patch, or belong to a new patch? This question is answered via hypothesis testing on two parameters: spatial correlation and probabilistic correlation.

1) Spatial Correlation: In order to answer the question of whether an elevation estimate extends a current patch, it first must be determined if its spatially correlated to that patch. To determine this correlation, hypothesis testing is performed to compare the elevation of the measurement estimate with respect to the elevation of the closest section of the current patch (i.e. if the estimate is below the patch, it compares the estimate with the elevation of the bottom of the patch). The null hypothesis H_0 is defined as the elevation estimate being spatially correlated, and thus possibly extending the patch, while the alternate hypothesis H_1 is that the elevation estimate is not spatially correlated to the patch. Since the measurement noise is assumed to be Gaussian, the observation likelihoods are defined relative to the elevation of the closest point of the patch to the estimate by

$$p(z_k|H_0) = \mathcal{N}(0, \sigma^2_{\hat{U}_k^{i,j}} + \sigma^2_{U_e^{i,j}}) \quad (1)$$

$$p(z_k|H_1) = \mathcal{N}(\theta, \sigma^2_{\hat{U}_k^{i,j}} + \sigma^2_{U_e^{i,j}}) \quad (2)$$

where θ is the noise-free elevation relative to the closest patch elevation for the estimate $\delta\hat{U}_k^{i,j} = \theta - \omega$, and $\omega = \mathcal{N}(0, \sigma^2_{\hat{U}_k^{i,j}})$ is the estimate noise. Notice that the likelihood uncertainty is a composition of the estimate uncertainty and $\sigma^2_{U_e^{i,j}}$, the elevation uncertainty of the patch in question. The hypothesis is then tested using the likelihood ratio

$$\Lambda(z_k) = \frac{p(z_k|H_1)}{p(z_k|H_0)}, \quad (3)$$

where $\Lambda(z_k) > \Lambda_0$ gives $1-\alpha$ probability that H_1 is true, when Λ_0 is determined via $p(\Lambda(z_k) > \Lambda_0|H_0) = \alpha$.

Generally there are three types of cases where this test would be applied: 1) elevation estimate is above the patch, 2) elevation estimate is below the patch and 3) elevation estimate is found within the patch. For all cases, however a height test point in the patch is chosen to apply the hypothesis test against the elevation estimate. If the estimate is above the patch, the closest point in the patch to the estimate is the patch mean, and $U_e^{i,j}$ would be used as the test point against the estimate. If the estimate is below the patch, the bottom of the patch $U_e^{i,j} - D_e^{i,j}$ would be used instead, where $D_e^{i,j}$ is the patch depth. If the elevation estimate is found within the patch, the closest height section of the patch will have the same elevation as the estimate. Notice that [7] take a heuristic approach to this spatial correlation through the use of the performance parameter γ, which represents the minimum distance between two completely uncorrelated patches. A study of the effect of this performance parameter is presented in this paper's experimental results.

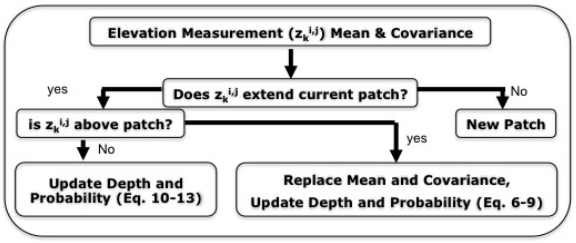

Fig. 2. Measurement-patch classification procedure for multiple elevation levels.

2) Probabilistic Correlation: Once an estimate is found to be spatially correlated to a patch, its probabilistic correlation [may be use another name because in literature it means different] must be assessed to finally determined if it is extending said patch. An estimate is probabilistically correlated to a patch if the probability of the estimate is statistically 'close' to that of the patch (i.e. if an estimate has a very low probability, while the patch has a very high probability, it can be said that the two are probabilistically uncorrelated). To determine this closeness, hypothesis testing is performed on the elevation estimate's and the patch's probabilities. In this case, the probability distribution of the patch is assumed to be Gaussian $\mathcal{N}(p_e^{i,j}, \sigma_p^2)$, where $p_e^{i,j}$ is the patch probability and σ_p^2 is the probability uncertainty, which can be selected by the user. This hypothesis test is done in a similar manner to the spatial correlation test. In this case, however, the null hypothesis H_0 is defined as the estimate being probabilistically correlated, while the alternate hypothesis H_1 as lacking correlation. A similar likelihood ratio is formed with the observation likelihood

$$p(z_k|H_0) = \mathcal{N}(0, \sigma^2_{p_k^{i,j}} + \sigma_p^2) \quad (4)$$

$$p(z_k|H_1) = \mathcal{N}(\phi, \sigma^2_{p_k^{i,j}} + \sigma_p^2) \quad (5)$$

where ϕ is the noise-free probability of the elevation estimate $\delta p_k^{i,j} = \phi - \nu$ relative to the probability of the patch, and $\nu = \mathcal{N}(0, \sigma^2_{p_k^{i,j}})$ is the estimate probability noise.

Once an estimate is found to be correlated both spatially and probabilistically to a patch, the patch characteristics must be updated. If the elevation estimate is located above the patch mean $U_e^{i,j}$, the algorithm uses the following to update the patch

$$D_e^{i,j} = D_e^{i,j} + \hat{U}_k^{i,j} - U_e^{i,j} \quad (6)$$

$$U_e^{i,j} = \hat{U}_k^{i,j} \quad (7)$$

$$\sigma_{U_e^{i,j}} = \sigma_{\hat{U}_k^{i,j}} \quad (8)$$

$$p_e^{i,j} = \frac{1}{2}(p_e^{i,j} + p_{\hat{U}_k^{i,j}}) \quad (9)$$

while if the estimate is below the patch mean, it is updated by

$$D_e^{i,j} = max(U_e^{i,j} - \hat{U}_k^{i,j}, D_e^{i,j}) \quad (10)$$

$$U_e^{i,j} = U_e^{i,j} \quad (11)$$

$$\sigma_{U_e^{i,j}} = \sigma_{U_e^{i,j}} \quad (12)$$

$$p_e^{i,j} = \frac{1}{2}(p_e^{i,j} + p_{\hat{U}_k^{i,j}}) \quad (13)$$

B. Patch Consolidation

Once all the elevation estimates are incorporated into their corresponding patches, a consolidation process is applied that combines patches that 'belong' together by applying the same hypothesis tests described in the previous section. The algorithm evaluates spatial and probability correlations of all patches within each cell and then combines those that are correlated.

III. Experiments and Results

Both the ML and PML surface mapping algorithms have been implemented and validated using the experimental testbed shown in Figure 3. The mobile robot platform is a *Pioneer P3DX* equipped with a *URG-04LX* laser rangefinder with range up to 4 m and scanning field of view of 240° with a beam width of 0.35°/beam for a total of 681 points per scan. The laser position is gimbaled, and orientation is controlled with two servo-motors atop the robot. The robot is equipped with a 2GHz *mini-ITX* computer with 1GB RAM memory. The data collected from the laser rangefinder is post-processed to obtained both the ML and PML surface maps. Position and orientation information for the robots is obtained from a *Vicon* system, which estimates pose by tracking IR reflective balls attached to the equipment. Both ML and PML algorithms were implemented on a 65x65 grid space, with 5 cm^2 cells. The coordinate axes is set so the EN plane is parallel to the floor and the U axis is pointed towards the testbed ceiling.

Three of the four experimental set-ups are shown in Figure 3. In order to gauge each algorithm's ability to map both indoor and outdoor environments, a set of four representative objects were chosen as tests. The four objects were a tunnel, a wall, a shallow hill and a flat surface. Data was collected by driving the robot back and forth in a straight line and sweeping the laser up and down the object. Several tests were completed varying the performance parameters for each mapping algorithm: for ML algorithm the minimum distance between patches γ was varied, while for the PML the probabilistic correlation uncertainty σ_p was varied. In the case of the ML algorithm, the smallest γ was chosen to match the average uncertainty in range measurements so as to accurately compare to the PML algorithm, while the two larger ones were chosen to represent typical robot heights (25 & 100 cm). For the PML, the smallest probability uncertainty σ_p was chosen slightly below half the average uncertainty in the range measurements, while the larger two were chosen two consecutive orders of magnitude higher so as to loosely match the larger γ parameters for the ML algorithm. Both algorithms were tested on three different scenarios: 1) noisy-dense data, where additional Gaussian noise, ($\mathcal{N}(0, \sigma_{noise} = 0.001$ cm)), is introduce on all the range data obtained ($DS = \#$ laser beams skipped $= 0$), 2) clean-dense data, where no additional noise is introduced and all the range data is used ($DS = 0$), and 3) clean-sparse data, where no additional noise is introduced, but a number of the range measurements are discarded ($DS = 30$) to simulate sparse data, thus taking only 23 points per laser scan.

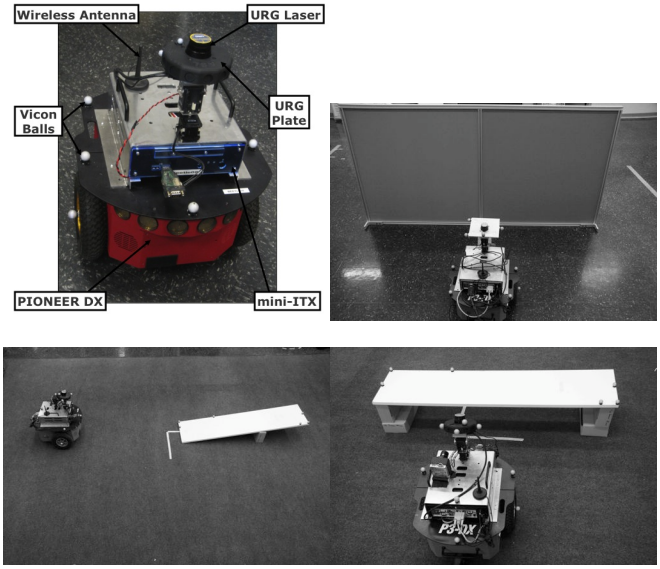

Fig. 3. Experimental platform (top-left) and experiment set-up: Wall (top-right), Shallow Hill (bottom-left), Tunnel (bottom-right)

In order to measure and compare the performance of each mapping algorithm, a metric was created that quantified the accuracy of the surface map generated against truth data obtained for the object using the *Vicon* localization system. For both the truth data and the ML and PML maps created, a full binary (occupied or not) 3D grid representation was created. Then, the 3D binary voxel representation for the ML and PML surface maps were compared against the truth. Finally, disparities were added and a score S was generated according to

$$S = w_1 * \Sigma(miss) + w_2 * \Sigma(extra), \tag{14}$$

where w_1 and w_2 are weighting constants, and $\Sigma(miss)$ is the total number of voxels the mapping algorithm missed from the truth, while $\Sigma(extra)$ is the total number of extra voxels the mapping algorithm added over the truth. In this paper, w_1 and w_2 were chosen to be 75% and 25%, thus giving more penalty to missing occupied voxels and less to wrongly occupying empty voxels. The reasoning behind this decision is the need for a conservative mapping algorithm; in general, it is better to have a map that overestimates obstacles in the environment. In addition, the truth data obtained from the *Vicon* system only represents the object itself, and therefore is missing data such as the floor beneath the object. Though this metric is not perfect, it serves well to qualitatively compare the performance of each mapping algorithm.

The results from all the tests performed are tabulated in Table I. The section called $PMLScores * 100$ show the scores determined according to Eq. 14 for all the tests. The $ScoreRates$ section shows the rate of change of the scores when the performance parameters are increased (i.e. Tunnel rate is 1.1 when changing the σ_p from 1.5 to 15 cm). Finally,

TABLE I

PML AND ML EXPERIMENT RESULTS

		PML Scores*100				Score Rates				σ_n/clean or clean/DS			
σ_p [cm]		Tunnel	Floor	Wall	Hill	Tunnel	Floor	Wall	Hill	Tunnel	Floor	Wall	Hill
1.5	clean	6.3	26.7	0.1	0.2	–	–	–	–	–	–	–	–
15	clean	6.7	39.3	0.1	1.3	1.1	1.5	1.9	6.1	–	–	–	–
150	clean	24.1	40.3	0.4	2.4	3.6	1.0	2.8	1.8	–	–	–	–
1.5	noise	8.6	34.5	0.1	3.1	–	–	–	–	0.4	1.3	1.1	14.6
15	noise	16	48.4	0.1	6.6	1.9	1.4	1.7	2.1	1.4	1.2	1.0	5.0
150	noise	32.5	49.6	0.9	10.9	2.0	1.0	6.4	1.7	0.3	1.2	2.3	4.6
1.5	DS=30	1.8	5.2	0.0	0.1	–	–	–	–	3.6	5.2	2.0	2.0
		ML Scores*100				Score Rates				σ_n/clean or clean/DS			
γ [cm]		Tunnel	Floor	Wall	Hill	Tunnel	Floor	Wall	Hill	Tunnel	Floor	Wall	Hill
4.4	clean	1.6	0.3	1.4	0.1	–	–	–	–	–	–	–	–
25	clean	1.9	1.6	2.1	0.1	1.2	5.2	1.5	1.3	–	–	–	–
100	clean	6.6	1.6	8.2	0.1	3.4	1.0	3.9	1.2	–	–	–	–
4.4	noise	2.0	0.1	1.4	0.1	–	–	–	–	1.3	0.2	1.0	1.2
25	noise	3.1	6.2	2.6	0.7	1.5	113.5	1.8	7.7	1.6	3.9	1.2	6.6
100	noise	6.9	7.0	7.7	4.0	2.2	1.1	3.0	6.2	1.1	4.4	0.9	33.5
4.4	DS=30	0.8	0.3	0.5	0.0	–	–	–	–	2.0	1.1	2.6	2.9

the last subdivision shows the ratio of the scores between σ_n and clean or clean and DS according to the respective tests (i.e. the ratio of σ_n/clean = 1.6 for ML Tunnel represents the ratio between the σ_n score of the ML Tunnel with noise and that of the ML Tunnel clean, both at γ= 4.4 cm, while the ratio of clean/DS = 2.0 for the Wall represents the ratio between the PML Wall clean score and that of the PML Wall test that skips 30 laser rays both at σ_p = 1.5). Notice also that Figure 4 shows, in a top down order, the resulting PML surface maps for $\sigma_p = 0.15$, $\sigma_p = 0.015$, as well as the ML surface map for $\gamma = 0.025$. The results in Table I show:

1) the PML algorithm generally performs better than the ML algorithm except for the Wall object, which can be seen by comparing the scores in the corresponding PML columns with those of the ML columns for the different objects.

2) increasing the algorithms' performance parameters σ_p for PML and γ for ML generally increases the score of both algorithms, however, there are cases where further increase on those parameters decreases performance. By looking at the PML and ML Scores columns, a monotonic increase on the raw scores is seen for all the objects. Nevertheless, by focusing on the rate columns, which show the rate of performance increase from one parameter to its increased value, depending on the object the rate drops as the parameter is increased (i.e., the PML Hill rate decreases from 6.1 to 1.8 as σ_p increases from 0.015 to 0.15 and the ML Floor rate decreases from 5.2 to 1.0 as γ increases from 0.025 to 0.1). This diminishing effect on the score can be deduced visually by looking at the top and middle maps generated on Figure 4. These are maps generated by the PML algorithm at $\sigma_p = 0.15$ and $\sigma_p = 0.015$ respectively. The plots show that the underpass feature of the tunnel is nonexistent for a large enough probability correlation variance σ_p. The reason why the algorithm shuts down the underpass is because it correlates patches on the top of the tunnel to estimates distributed at the center from the side walls of the tunnel. That is, estimates nominally originating from side walls of the tunnel around cells (0.5,0.2) and (-0.5,0.2) are associated to neighboring cells near the symmetric center of the tunnel around cell (0.0,0.2). Though these associated estimates have very low probability of originating at this center cell, with a very large probability correlation variance they are incorporated into the patches directly above the center cell, thus closing the tunnel entrance. In the limit, as the σ_p increases, the PML algorithm reduces to a terrain elevation estimator alike that of [6].

3) the PML algorithm generally deals better with noise than the ML algorithm except for the Floor object. This is visible by comparing the σ_n/clean columns in Table I, which show the performance of each algorithm on noisy data over its performance on clean data. Notice, however that the values in the σ_n/clean column are greater than unity, meaning the score of the map on the noisy data is better than that on the clean data, though the opposite is expected. The reason for this discrepancy is the manner in which the performance metric works. Adding noise to the data increases its entropy, meaning more chances that voxels the algorithm would have left empty in the clean case would become occupied now. If these voxels happen to be ones that should, according to truth data, be occupied, this will increase the score of the map given the metric's heavy weight on true positives. This is one of the downfalls of this metric, however, this metric is still useful for comparing the performance of the PML and ML algorithms.

4) the PML algorithm generally outperforms the ML algorithm when using sparse data, except for the Hill case. This can be seen by comparing the clean/DS rows, which show the performance of the clean, dense map over that of the sparse map.

In addition, from the bottom two plots in Figure 4 it is easy to see that the PML algorithm is able to generate a more populated and conservative surface map than the ML algorithm. Notice that both surface maps were generated from the same data, yet the ML algorithm is not quite able to produce a tunnel-like object. In the limit, as the number of measurements taken is increased, the ML algorithm is able to provide a more populated representation of its environment; however, along with an increase of measurements comes an increase in computation and in time required to produce the map.

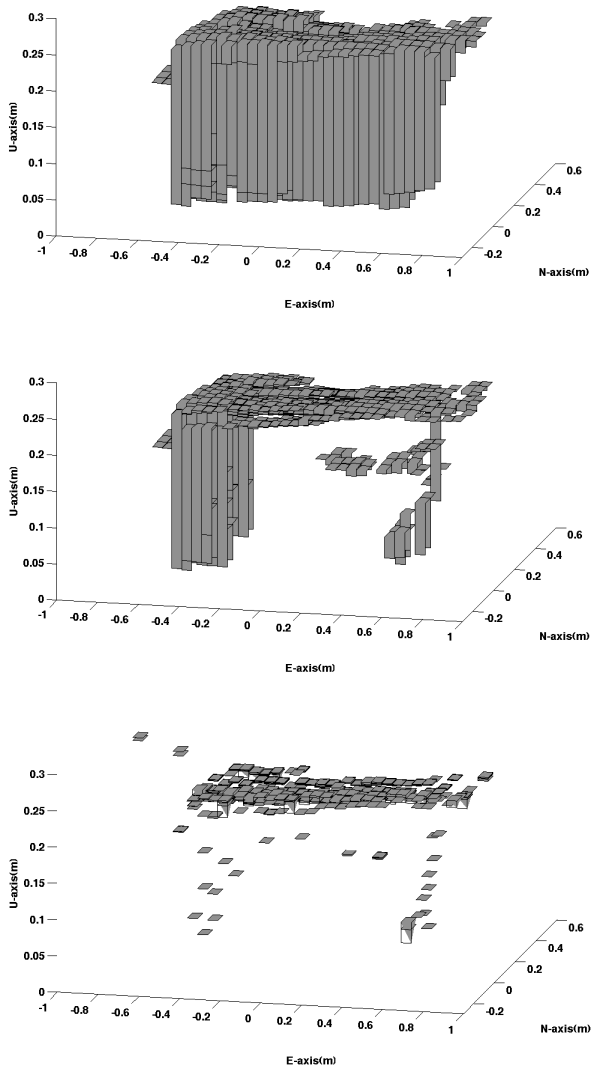

Fig. 4. Experimental results for mapping Tunnel object. PML surface map for tunnel object at $\sigma_p = 0.15$ (top) and for $\sigma_p = 0.015$ (medium). ML surface map for $\gamma = 0.025$ (bottom).

IV. CONCLUSIONS

This paper develops a Probabilistic Multi-Level (PML) terrain map using formal probability methods to better and more accurately model complex environments. This algorithm was implemented and validated using a mobile robotic testbed through experiments on four generalized representative objects. Experimental results were then compared to those of a benchmark ML surface mapping algorithm and a truth set of data. The results show that PML algorithm was generally able to outperform the ML algorithm in most cases. Under some circumstances, the performance of the PML algorithm was overtaken by that of the ML algorithm, such as those of the Wall object. The results also point out that the PML algorithm is better at handling noise in the measurements than the ML algorithm. Further, the experiments show that the ML algorithm performance drops more significantly when dealing with sparse data than the PML algorithm. Visual inspection of the results give a better picture at the performance of the PML surface map, since it is able to produce a better populated and denser representation of the environment with less measurements in less time and lower computational cost. It is also noted that though in some circumstances the PML algorithm was outperformed by the ML, it still provides a more accurate representation in many cases. Finally, though the PML algorithm does have a tuning parameter σ_p, it does not have the same detrimental effect that the performance parameter γ has for the ML algorithm. The σ_p is merely a parameter affecting the probabilistic correlation among patches and measurement estimates. The γ parameter dictates the way in which the robot sees the world according to the ML algorithm because it heuristically determines the incorporation of measurements into patches. In some specific cases, this can be quite problematic. For example, if a set of robots of different sizes are working together to represent an environment, they will produce radically different representations given that their γ parameters, which are dictated by the robot's individual height, would be different. On the other hand, a set of different robots working together using the PML algorithm do not have to utilize different performance parameters. In conclusion, the PML algorithm generally produces more accurate and more conservative surface maps, which are more suitable for robot navigation and cooperative exploration.

REFERENCES

[1] A. P. Tirumalai, B. G. Schunck, and R. C. Jain, "Evidential reasoning for building environment maps," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 1, pp. 10–20, 1995.

[2] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping," in *International Conference on Robotics and Automation*, vol. 1. San Francisco, CA: IEEE, 2000, pp. 321–8.

[3] R. Hoffman and E. Krotkov, "Terrain mapping for long-duration autonomous walking," in *International Conference on Intelligent Robots and Systems*, vol. 1. Raleigh, NC: IEEE, 1992, pp. 563–568.

[4] S. Singh and A. Kelly, "Robot planning in the space of feasible actions:two examples," in *IEEE International Conference on Robotics and Automation*, vol. 4. Minneapolis, MN: IEEE, 1996, pp. 3309–3316.

[5] P. Pfaff, R. Triebel, and W. Burgard, "An ecient extension of elevation maps for outdoor," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 217–230, 2007.

[6] I. Miller and M. Campbell, "A mixture-model based algorithm for real-time terrain estimation," *Journal of Field Robotics*, vol. 23, no. 9, pp. 755–775, 2006.

[7] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems,*. Beijing, China: Freiburg University, 2006, pp. 2276–2282.