Textured Occupancy Grids for Monocular Localization Without Features

Julian Mason, Susanna Ricco, and Ronald Parr

***Abstract*— A textured occupancy grid map is an extremely versatile data structure. It can be used to render human-readable views and for laser rangefinder localization algorithms. For camera-based localization, landmark or feature-based maps tend to be favored in current research. This may be because of a tacit assumption that working with a textured occupancy grid with a camera would be impractical. We demonstrate that a textured occupancy grid can be combined with an extremely simple monocular localization algorithm to produce a viable localization solution. Our approach is simple, efficient, and produces localization results comparable to laser localization results. A consequence of this result is that a single map representation, the textured occupancy grid, can now be used for humans, robots with laser rangefinders, and robots with just a single camera.**

I. Introduction

The construction and use of maps is a fundamental problem in mobile robotics. Without a map of its environment, a robot cannot make long-range planning decisions. A map which directly encodes the presence or absence of objects in the world (an *occupancy map*) is the most natural representation for path-planning and obstacle-avoidance problems. As an added advantage, these maps are intuitive for human users, making it possible for a robot to scout ahead and return useful information to an operator. Occupancy maps are typically used on robots with active range sensors like sonars or laser rangefinders.

Because occupancy does not correspond directly to appearance, mapping for camera-based robots focuses on *features* (also called *landmarks*). The sparsity of feature-based maps provides computational benefits, but limits their usefulness in path-planning and obstacle-avoidance tasks. Since features are chosen for their mathematical properties, rather than human saliency, these maps are far harder for people to use.

A natural fusion of occupancy and appearance is the three-dimensional *textured occupancy grid map*. In such a map, three-dimensional space is discretized into a set of voxels, each of which stores an occupancy measure and (if occupied) information describing its appearance. Such a map permits use as a pure occupancy map (by ignoring the texture), and can be rendered to produce the appearance of the map from a given pose. Therefore, one map can be used for both camera- and range-sensor-equipped robots. Furthermore, the ability to render views makes these maps extremely intuitive for human users.

Department of Computer Science, Duke University, Box 90129, Durham NC 27708. {mac, sricco, parr}@cs.duke.edu

This work supported by NSF CAREER award IIS-0546709 and NSF grant IIS-1017017. Any opinions, findings, conclusions, or recommendations are those of the authors only.

In this work, we describe a simple algorithm for narrow-field-of-view monocular localization of a mobile robot in a textured occupancy grid map. We then use the same map, without modification, to perform laser-based localization. We demonstrate empirically that the two techniques have comparable performance. Importantly, our camera-based algorithm neither extracts nor uses features; instead, it relies on the ability to simulate a camera view given the textured occupancy grid.

II. Prior Work

What we call a textured occupancy grid is a form of *evidence grid*, as introduced by Moravec and Elfes [1]. In an evidence grid, the world is discretized into cells, and each cell stores some information about that location. In mobile robotics, this information is usually an *occupancy*, a probability distribution which describes a belief about the presence of an object at that location. The original evidence grid work focused on two dimensions; Moravec later extended the idea to three dimensions (where each cell represents a volume), and added color, producing textured occupancy grid maps [2]. Moravec clearly considered using these maps for localization, but did not directly address it. Similar work using appearance to inform stereo-vision-based depth sensing is presented by Sáez and Escolano [3].

Recently, there has been a resurgence in work on three-dimensional occupancy grid maps. Wurm et al. [4], Triebel et al. [5], and Fairfield [6] present examples of techniques for building and working with three-dimensional occupancy grids. None, however, address texture.

A state of the art approach to single-camera localization (and mapping) is MonoSLAM, (see, e.g. Strasdat et al. [7]), which builds feature-based maps. Other techniques rely on cameras with a very wide field of view (so-called "omnivision" cameras). Omnivision techniques traditionally extract features from panoramic images (for example, Scaramuzza et al. [8] use vertical lines, while Menegatti et al. [9] use Fourier coefficients of the image) and perform matching, either between local features or to a database of reference images. One advantage of panoramic images is that they greatly ease the problem of determining the robot's orientation, as the view from many (or all) directions is captured simultaneously. An interesting variation is to assume that the camera can (almost) always see a consistent, textured surface, commonly the ceiling. Dellaert et al. [10] applied this to a mosaic of the ceiling for a robot tour guide, and similar approaches were taken by Gamallo et al. [11] and Jeong and Lee [12].

Kitanov et al. [13] and Koch and Teller [14] present near neighbors to our approach, using hand-built 3D models of an environment. Images rendered from this model then have features extracted; these features are compared to features extracted from camera images of the actual scene. These approaches do not store texture, but assume that occupancy information from an available CAD model of the scene will correspond to 2D features in a captured image. These efforts produced good results when seeded with clean CAD models and when sufficient effort was invested in robustification of the feature detection process. To our knowledge, there are no end-to-end demonstrations of these approaches which start with raw sensor data, build a CAD model, then demonstrate that the inferred CAD model can induce features useful for monocular tracking.

We demonstrate that a textured occupancy grid of the type that could easily be built by a robot endowed with a laser rangefinder and camera can be used for localization *either* with a laser rangefinder alone *or* a camera alone. Our camera based localization approach does not rely upon feature detection and uses only some very minor robustness modifications to deal with deficiencies (missing data) in the map.

III. Datasets

A. Maps

We created 3D textured occupancy grids of two environments, which we call "lab" and "hallway." Both were generated using data from a Riegl LMS-Z390 laser rangefinder, with an attached calibrated Nikon D200 camera. This standalone sensor generates colorized three-dimensional point clouds; that is, each (x, y, z) point includes an RGB color. The lab dataset consists of nine such scans, for a total of $20,272,859$ points; the hallway contains ten scans, for $21,740,817$ points. The scans were manually aligned using retroreflective targets placed in the environments. To convert point clouds to our textured occupancy grids we use a three-dimensional octree-based variant of the occupancy model described by Eliazar and Parr in DP-SLAM [15]. (Note that we have known poses for map building, and are not performing SLAM.) The resulting maps are thresholded on occupancy, and only those voxels with sufficient occupancy are kept. Retained voxels are colored according to the average color (in RGB) of all Riegl points contained within the voxel. Our technique and results are not dependent on the use of the Riegl for map construction. The Riegl's 6mm range accuracy and 360° horizontal field of view made it a convenient sensor, but not a necessary sensor: any three-dimensional range sensor that includes color could create usable maps.

The lab dataset is an open room, roughly $12 \times 15 \times 2.5$ meters in size. A two-dimensional slice of this environment can be seen in Fig. 6.

The hallway dataset is of a larger environment, roughly $17.5 \times 22.5 \times 3$ meters in size, which can be seen in Figs. 1, 4, and 7. This dataset is a classic hallway environment: it contains large regions of uniform color, windows with reflections, and doorways that are subject to visual aliasing. When being used for localization, this map requires roughly 500MB of main memory.

B. Robot Data

Our mobile robot is an iRobot Create with a Logitech Webcam Pro 9000, with a field of view of roughly 48°. The robot also carries a Hokuyo URG-04LX laser rangefinder, with a maximum range of 4 meters, and a field of view of 240°. The robot was manually driven through both environments, taking one picture (see Figures 7(a) and (d) for example images) and one 2D laser scan every half-meter of linear distance or 25° of angular distance. For both the hallway and lab datasets, one robot trajectory was taken simultaneously with Riegl data acquisition; for the hallway dataset, another robot trajectory (the "new hallway") was taken several months later (see Section VI-C).

We independently localize in the maps using only the Hokuyo or robot-mounted camera at a time. The Riegl is far larger than an iRobot Create, and is not robot-mounted.

C. Missing Data and Perspective Issues

Our maps suffer from some problems of missing data. The Riegl's vertical field of view is only 80°, and although our scans overlap, a few small regions remain unmapped. In other locations the scanner encountered laser-absorbent surfaces or reflections which result in no laser return. (We found windows and bookcases to be particularly challenging.) The hallway contains two glass doors which open onto an unmapped central classroom. Poses that see through these doors suffer from a particularly extreme case of missing data.

Any algorithm relying on a fixed map is subject to difficulties with missing data, but a further issue arises when there is a substantial difference in the perspective from which the map was built and the perspective from which it is rendered. The Riegl's center of projection is roughly 150 cm off the floor, while the robot's camera is only 19 cm off the floor. Therefore, our maps do not contain observations of the colors of the bottoms of objects between 19 and 150 cm from the floor, but such surfaces may be visible to the robot. For example, consider the table in Fig. 7(a). This table is white on top and therefore white from the Riegl's point of view. It is actually black underneath, which can be seen in this image. Because we can only render according to the Riegl data, such views from the robot's height could not be expected to appear the correct color. As a result, bottom faces of overhanging objects should be treated as having missing texture data. As discussed in Section V, our camera sensor model deals with missing data gracefully.

Our datasets (both maps and trajectories) are available at `http://www.cs.duke.edu/~parr/textured-localization`. They include information not used here, including extra camera images and three-dimensional laser data taken from the robot on some trajectories.

IV. The Problem

Robot localization is the problem of determining a robot's pose at time t, $\mathbf{x}_t = [x\ y\ \theta]^\top$, given a sequence of robot actions $\mathbf{a}_{1:t}$ and observations $\mathbf{y}_{1:t}$. The key difficulty is that the robot's actions are noisy; the $\mathbf{a}_t$ reported by the robot (from, for example, wheel odometers) is a noisy reading of the robot's actual action. This means that even given a known $\mathbf{x}_0$ (the *robot tracking* problem), the robot's reported pose can diverge arbitrarily from reality. In the more general case, the robot's initial pose is unknown (the *kidnapped robot* problem). The standard approach is to use a particle filter (introduced for mobile robot localization by Dellaert et al. [16]) to maintain a probability distribution (the *belief state*) over the robot's pose. Particle filter models (and, more generally, recursive filtering techniques) model the error in the robot's actions and sensor readings probabilistically. Specifically, they assume that the distribution decomposes according to the Markov property:

$$P\left(\mathbf{x}_t|\mathbf{a}_{1:t}, \mathbf{y}_{1:t}\right) \propto M \cdot S \cdot P\left(\mathbf{x}_{t-1}|\mathbf{a}_{1:t-1}, \mathbf{y}_{1:t-1}\right)$$

The distribution $M = P\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{a}_t\right)$ is the *motion model*; the distribution $S = P\left(\mathbf{y}_t|\mathbf{x}_t, M\right)$ is the *sensor model*, given a map M.

In a particle filter, the distribution over robot poses is stored as a set of samples from that distribution (called particles). Each particle represents a hypothesis about the robot's pose. Each particle is also assigned a weight which is proportional to its probability; therefore, the filter represents a discrete probability distribution over robot poses.

V. The Algorithm

The main loop of a particle filter updates the particles' weights and poses in response to a single action and observation. Let p denote a single particle, with pose $p_{\mathbf{x}}$ and weight p_w. Given a set P containing n particles, an action $\mathbf{a}_t$, and a sensor reading $\mathbf{y}_t$ taken *after* performing the action, each iteration proceeds as follows:

1) For each particle $p \in P$:
   a) Draw a sample $\hat{\mathbf{x}}$ from $P\left(\mathbf{x}_t|p_{\mathbf{x}}, \mathbf{a}_t\right)$, and set $p_{\mathbf{x}}$ to $\hat{\mathbf{x}}$.
   b) Set p_w to $P\left(\mathbf{y}_t|p_{\mathbf{x}}, M\right)$.
2) Resample (with replacement) n new particles from P, and replace P with these new particles.

To specify the algorithm completely, we must provide both a motion model and a sensor model.

A. Motion Model

For both monocular-camera and laser-based localization, we use the motion model described by Eliazar and Parr [17]. This decomposes the robot's motion into three Gaussian terms: one parallel to the direction of motion, one perpendicular to the direction of motion, and one rotational component. The parameters of the motion model were set by hand, but were purposefully left with relatively high variance to guarantee good coverage of the motion space.

B. Sensor Models

For laser-based localization, we use the laser propagation sensor model described by Eliazar and Parr [15].

For a given particle pose $p_{\mathbf{x}}$ in our monocular-camera sensor model, we begin by using OpenGL to render an image of the map, as seen from $p_{\mathbf{x}}$. In this sense, we are using OpenGL as a "camera simulator." Due to the discretization in the occupancy grid, the rendered image will be "blocky," as can be seen in Fig. 7. We then independently normalize each color channel of the render to have mean 0 and standard deviation 1. The same normalization is applied to the image captured by the robot's camera. We then take the L_2 norm of the difference between the normalized render and normalized camera image, treating each channel independently. The probability of a particle with L_2 norm l is then proportional to $e^{-l^2/2\sigma^2}$, for an empirically chosen value of σ^2. This corresponds to assuming that errors per-pixel, per-channel, are drawn from independent Gaussian distributions. Our normalization step provides some robustness against global changes in illumination. In particular, it accounts for both additive and multiplicative changes in lighting (by normalizing the mean and standard deviation, respectively). Note that unmodeled local changes (for example, the open window seen in Fig. 2) occur in our data.

A more accurate model would require processing the camera image to account for the "blockiness" in the rendered image; this is not a straightforward transformation. Despite this, our normalized L_2 works quite well in practice, as can be seen in Section VI.

In some cases, it will not be possible to render every pixel; this happens when the view contains regions with missing occupancy or texture. When occupancy is missing, we complete the rendered image by inpainting [18], [19]. This produces realistic images when the missing regions are small. When more than 50% of the pixels cannot be rendered, we assign that particle probability zero. In our experiments, this applies almost exclusively to particles which have strayed into unreachable parts of the map.

Missing texture due to perspective issues is more subtle; in these cases, inpainting generally produces the wrong colors. The standard approach is to ignore such pixels, and renormalize the image score according to the number of observed pixels. This effectively assigns unobserved pixels a score equal to the average score of the observed pixels, but also has the effect of making unobserved pixels wild cards that could align with any part of the real image. Empirically, this tended to give poses with many unobserved pixels very high scores. A better solution is to recognize explicitly that these situations do not provide useful information to the filter, and their information should be ignored. We set a threshold t_1 on the number of pixels that can be missing texture, and count the number of particles that exceed it. Should that count exceed a second threshold t_2, we do not resample the filter on that iteration, and assign every particle a uniform probability. This allows us to maintain uncertainty in the filter until the particles return to more informative poses. Should

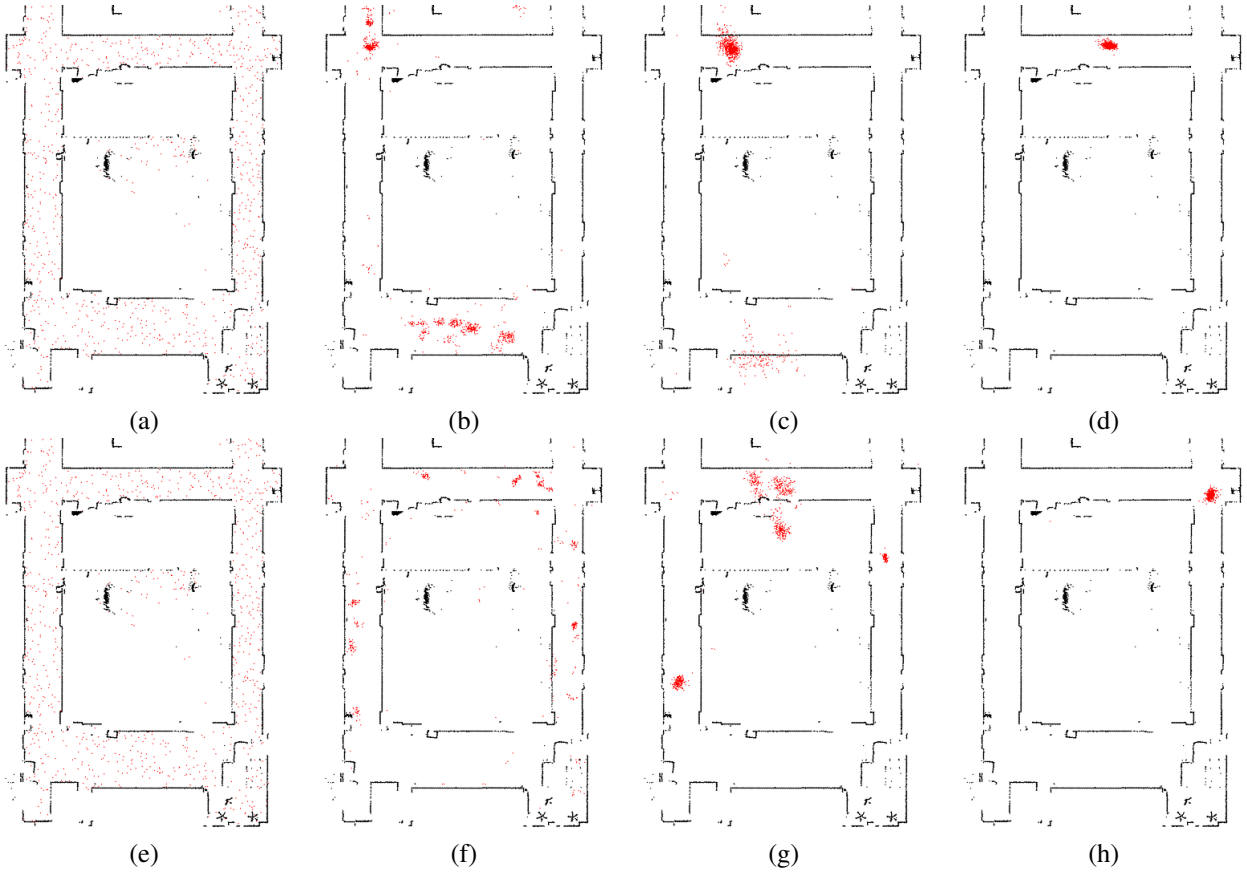

Fig. 1. Examples of camera localization on a robot with an unknown starting pose. Images (a) through (d) are steps 0, 6, 12, and 20 of localization on the original hallway trajectory. Of particular note is (c), an example of the delayed-resampling strategy discussed in Section V-B. Images (e) through (h) are steps 0, 5, 16, and 27 on the new hallway trajectory. All images are with 1000 particles, shown as red dots. This figure is best viewed in color, and should be compared to the tracking results in Figs. 4 (top row) and 5 (bottom row).

the t_1 count not exceed t_2, we inpaint. For the experiments presented, we set these thresholds conservatively, with $t_1 = 2.5\%$ and t_2 equal to half the number of particles. Unlike the N_{eff} criterion [20], we detect regions of uncertainty directly, rather than inferring them from our belief state. Missing-texture regions (rendered in yellow) and missing-occupancy regions (rendered in green) can be seen in Fig. 7.

Our sensor model uses a total of four parameters. Two parameters, t_1 and t_2, are required to deal with missing texture caused by perspective mismatch between the mapping and localizing sensors. We anticipate that these two parameters could be eliminated if the map were created and then used from similar perspectives. The third parameter (missing-occupancy pixel count) controls when proposed poses are considered to be outside the mapped region. It could be replaced by another technique for detecting particles that have left the map. The final parameter, σ^2, sets the variance of our Gaussian error model.

VI. Empirical Analysis

To validate our approach, we performed robot localization with both a known and unknown initial pose using both laser localization and camera localization, and occupancy grids produced from Riegl data. The occupancy grids have a resolution of 20 grid cells per linear meter (voxels 5cm on a side).

A. Unknown Initial Pose

Tracking from an unknown initial pose is the classic kidnapped robot problem and is a good measure of the flexibility of the filter and sensor model. We solved the kidnapped robot problem in both the lab and hallway datasets, using all three trajectories, but due to space limitations we show only the more challenging hallway problem in Fig. 1. (The top row shows the old hallway trajectory; the bottom row shows the new hallway trajectory, discussed in Section VI-C.) We generated a set of 1000 particles uniformly at random from the reachable part of pose space, using rejection sampling to avoid sampling in unreachable parts of the map. At step 6 (Fig. 1(b)), the particles show a tight cluster around the true position of the robot, with several remaining symmetric clusters. Step 12 (Fig. 1(c)) shows the effect of delaying resampling due to missing texture in the occupancy grid. Convergence to a single particle cluster occurs on step 11, which is not shown.

B. Known Initial Pose

To demonstrate quantitatively that camera localization and laser localization are comparable in performance, we

measured the maximum and mean difference between the X–Y positions of the robot in both camera and laser localization for the lab dataset for varying numbers of particles, shown in Fig. 3. The maximum and mean are computed over all poses in the trajectories of the highest probability particles at the end of six independent filtering runs (with varying random seeds) for each sensor type. We needed a minimum of 12 particles to track successfully through the entire trajectory. At 50 particles, the differences between camera and laser are quite small. Trajectories differ by less than one robot width on average and never by more than two robot widths. While the disagreement decreases with increasing particle count, we do not expect substantial improvement above 100 particles. One reason for this is that areas with missing textures, along with resulting delayed resampling, would ensure variance (at a few specific points in the map) in the particle distribution for camera localization regardless of the particle count. For laser localization, there will be places where the limited range of the URG-04LX introduces ambiguity.

For a more qualitative assessment of localization performance we plot the trajectories of the highest probability particles in both the lab (Fig. 6) and hallway (Fig. 4) datasets. These figures demonstrate wide agreement, with some pockets of disagreement between camera and laser localization. Fig. 7(a)-(c) shows a position from the hallway dataset (top right in Fig. 4), where camera and laser localization disagree. In this case, camera localization suffers from the problem of missing textures because the underside of the table was not seen by the Riegl. Fig. 7(d)-(f) shows another position in the hallway dataset (top left of Fig. 4) where camera and laser localization disagree. In this case, camera localization is more accurate, most likely because many scans in this open expanse of hallway are reaching the maximum range of the URG-04LX laser, while strong visual cues remain available.

C. Lighting and Map Changes

To demonstrate the robustness of our algorithm to changes in lighting and the physical structure of the world, we took a second robot trajectory (the "new hallway") in the hallway environment, roughly three months after the original data were collected. In that interval, objects both large (desks and couches) and small (trash cans) moved, making the Riegl-generated map incorrect. The new trajectory was taken at a different time of day, resulting in different global lighting conditions as well as extreme changes in naturally-lit regions. Lastly, wall textures, including posters and bulletin boards, changed. An example of these changes can be seen in Fig. 2. Finally, we traversed the loop counter-clockwise (where the original trajectory was clockwise), to guarantee no dependence on the direction of travel.

We performed both robot tracking and robot kidnapping on this new trajectory. Using 1000 particles, kidnapping converged to a single cluster of particles in 27 steps, as can be seen in Fig. 1(h); the trajectory from robot tracking with 100 particles can be seen in Fig. 5. This trajectory shows more pronounced disagreement between the laser and camera trajectories. When compared to the original trajectory

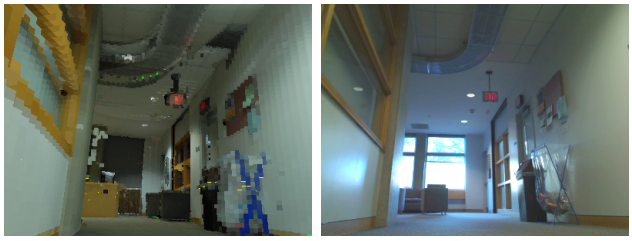

Fig. 2. An example of map errors encountered in the new hallway trajectory. The left image is a render from the Riegl-generated map; the right image is from the robot's camera, at roughly the same pose. Note the abundant natural light (due to an open windowshade) and the structural change (the chairs have been moved to the left, replacing the desk).

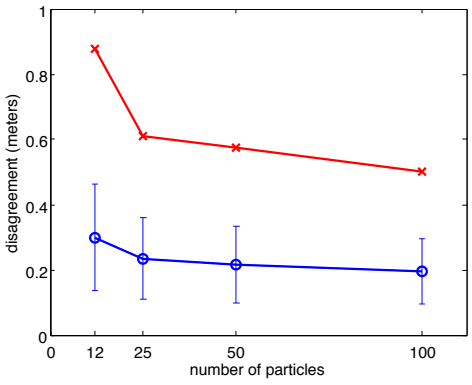

Fig. 3. Quantitative comparison of localization methods for 12, 25, 50, and 100 particles on the lab dataset. For each particle count, we computed six trajectories for camera localization and six for laser localization, each using a different random seed. The trajectory used is the trajectory of the highest probability particle at the end of the run. Red (top) shows the maximum disagreement in X–Y position in meters over all corresponding poses in the resulting 36 laser-camera trajectory pairs. Blue (bottom) shows the mean disagreement over all poses. Error bars on the blue curve show one standard deviation.

(Fig. 4), the mean disagreement has increased from 0.25 meters to 0.59 meters, and the maximum disagreement has increased from 0.74 meters to 1.47 meters. An iRobot Create is roughly 0.34 meters in diameter.

VII. Future Work

We see many opportunities for continued work using textured occupancy grid maps. One limitation of our current sensor model is that it assumes the lighting conditions remain similar between the mapping and localization steps. Normalization proved to be sufficient to handle the conditions tested in our experiments. In general, however, maps may be created and then used under a wide range of illuminations. We acknowledge that more sophisticated pre-processing and something more robust than L_2 distance in normalized RGB may result in better performance under more challenging changes in lighting conditions. One could also imagine combining ideas from feature-based localization with the rendered views textured occupancy grid maps provide, for example by weighting particles based on the distance between detected SIFT features [21]. Because we currently use a single color per voxel, our rendered images appear

Fig. 4. Robot trajectories for the hallway dataset (22.5×17.5 m) with 100 particles. Dead reckoning is rendered in green (lightest gray). It exits the figure on the top left but returns again in the bottom right. The camera localization trajectory is rendered in red (medium gray), and the laser localization trajectory is rendered in blue (darkest gray). Two red lines and blue lines are visible on the left side because the robot covered this area twice; it started in the bottom left, completed a clockwise loop, then finished in the top center after rounding the corner. The mean disagreement between the laser and camera trajectories shown here is 0.25 meters; the maximum is 0.74 meters. All trajectories are the trajectories of the highest probability particle at the end of the run.

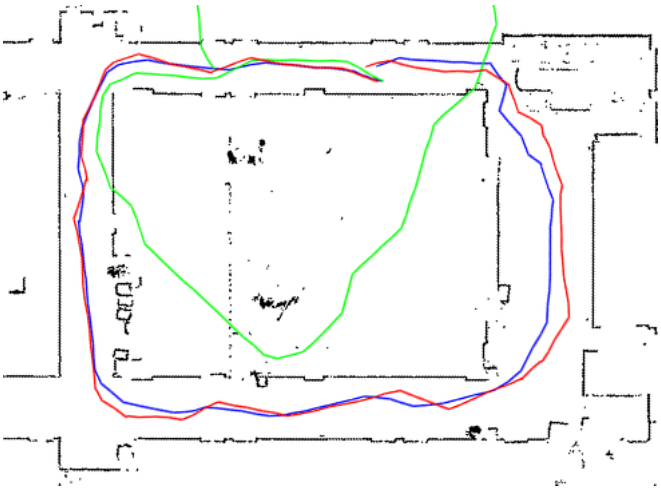

Fig. 5. Robot trajectories (tracking, 100 particles) for the hallway map, using the new hallway trajectory. (Compare to Fig. 4.) The robot started at top-center, moving left, and completed a single counter-clockwise loop. Dead reckoning is rendered in green (lightest gray), and exits the map at the top-right, before re-appearing at the top-left. The laser trajectory is rendered in blue (darkest gray), and the camera trajectory in red (medium gray). The lower-left, lower-right, and upper-right corners are regions of substantial natural lighting (and therefore major lighting changes); the lower-right and upper-right corners also have substantial structural changes. A view of the changes in the lower-right corner can be seen in Fig. 2. In both right corners, the camera's field of view is dominated by changed regions. In contrast, the laser's wider field of view allows it to measure the interior wall, which is unchanged. This gives the laser a substantial advantage. The mean disagreement between the laser and camera trajectories shown here is 0.59 meters; the maximum is 1.47 meters. All trajectories are the trajectories of the highest probability particle at the end of the run.

Fig. 6. Robot trajectories for the lab dataset (15×12 m) with 100 particles. The robot starts in the top right of the map and moves in a counterclockwise loop. Dead reckoning is rendered in green (lightest gray). The camera localization trajectory is rendered in red (medium gray), and the laser localization trajectory is rendered in blue (darkest gray). In this particular example, the mean disagreement between camera and laser localization is 0.17 meters, and the maximum is 0.41 meters. See Section VI-B and Fig. 3 for more information. All trajectories are the trajectories of the highest probability particle at the end of the run.

blocky; this makes it difficult to detect corresponding SIFT features. However, adding richer texture to the occupancy grid would make the rendered images more realistic. Our prototype implementation is fast enough to perform robot tracking in real time on our robot (which pauses briefly after image capture) but does not scale to full-speed video. The bottleneck is rendering, a problem which has been studied in great detail in the graphics community. We plan to optimize our renderer to support solving the kidnapped robot problem in real time. Finally, a map built using a mobile robot (rather than the Riegl) would improve our camera localization performance because it would eliminate (or at least greatly reduce) the perspective mismatch, reducing the frequency of missing texture regions. Very recently, maps similar to ours have been built using the Microsoft Kinect [22].

VIII. Conclusion

We constructed textured occupancy grid maps for two environments using a Riegl laser rangefinder and then used the resulting maps for localization on an inexpensive iRobot Create platform using (separately) a Hokuyo URG-04LX laser rangefinder and a single camera. For camera localization, we modified the standard particle filter localization algorithm to use a simple observation model based upon the L_2 norm of the difference between the observed image and an image rendered from the textured occupancy grid map. We also introduced a delayed-resampling strategy to account gracefully for missing texture in the map. Our results demonstrate that a single textured occupancy grid can be used effectively for localization with either a laser rangefinder or a camera, even when there are substantial map errors in color and structure. This demonstrates the viability of this data structure as a universal map representation for robots with a variety of sensors.

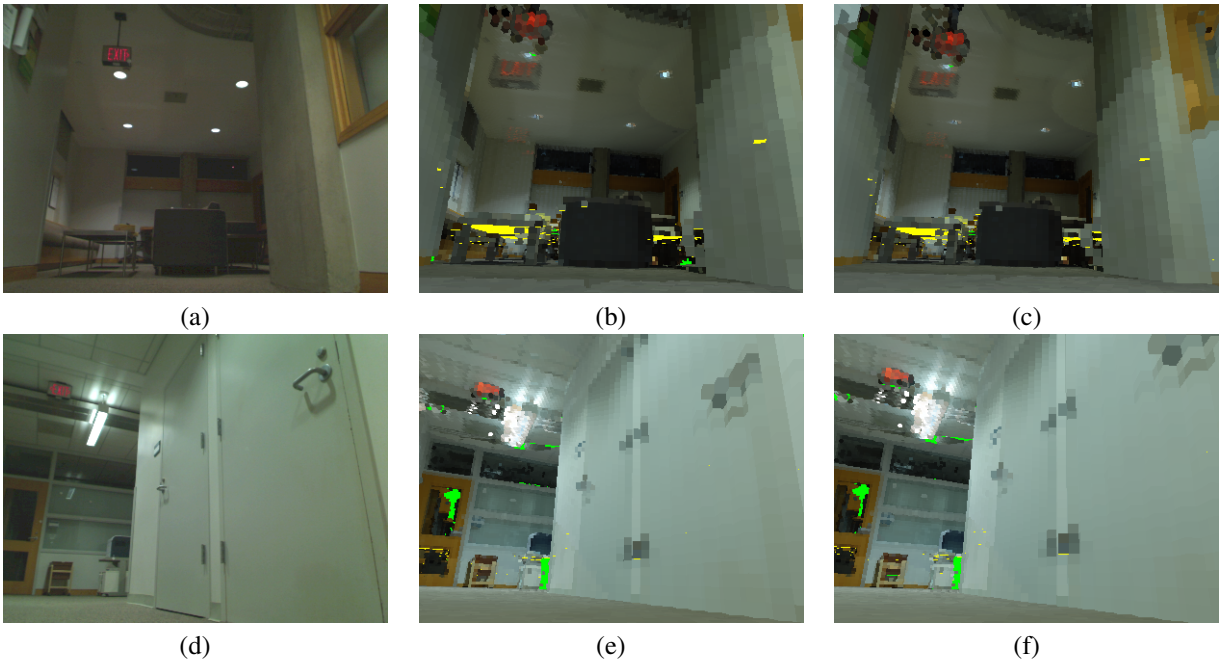

Fig. 7. Results for localization with 100 particles in the hallway dataset. Images (a) through (c) show an example where laser localization outperforms camera localization. Image (a) is the image captured by the camera. Image (b) is a corresponding render of the scene from the pose reported by the camera localization trajectory seen in Fig. 4. Image (c) is the comparable image from laser localization. Both camera and laser localization are too far forward, but laser localization is more correct. Images (d) through (f) repeat the pattern of (a) through (c), but show an example where camera localization outperforms laser localization. At this point, the far wall is outside the Hokuyo's range. As a result, laser localization cannot disambiguate poses along the major axis of the hallway. Camera localization can use the visual structure of the door handle (top right) to infer the correct pose. Regions with both missing texture (rendered in yellow) and missing data (rendered in green) are visible in this figure.

REFERENCES

[1] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 1985.

[2] H. Moravec, "Robust navigation by probabilistic volumetric sensing," Carnegie Mellon University, Tech. Rep., 2002, http://www.frc.ri.cmu.edu/~hpm/project.archive/robot.papers/2002/ARPA.MARS/Report.0202.html.

[3] J. M. Sáez and F. Escolano, "Monte Carlo localization in 3D maps using stereo vision," in *Proc. Advances in Artificial Intelligence - IBERAMIA 2002*, 2002.

[4] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: a probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[5] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proc. IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[6] N. Fairfield, "Localization, mapping, and planning in 3D environments," Ph.D. dissertation, Carnegie Mellon University, 2009.

[7] H. Strasdat, J. M. M. Montiel, and A. Davison, "Scale drift-aware large scale monocular SLAM," in *Proc. Robotics: Science and Systems*, 2010.

[8] D. Scaramuzza, N. Criblez, and A. Martinelli, "Robust feature extraction and matching for omnidirectional images," in *Proc. 6th International Conference on Field and Service Robotics (FSR)*, 2007.

[9] E. Menegatti, M. Zoccarato, and E. Pagello, "Image-based Monte Carlo localisation with omnidirectional images," *Robotics and Autonomous Systems*, vol. 48, no. 1, pp. 17–30, 2004.

[10] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[11] C. Gamallo, C. Regueiro, and P. Quintía, "Omnivision-based KLD-Monte Carlo localization," *Robotics and Autonomous Systems*, vol. 58, pp. 295–305, 2010.

[12] W. Jeong and K. M. Lee, "CV-SLAM: a new ceiling vision-based SLAM technique," in *Proc. IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[13] A. Kitanov, S. Bisevac, and I. Petrovic, "Mobile robot self-localization in complex indoor environments using monocular vision and 3D model," in *Proc. IEEE/ASME Inter. Conf. on Advanced Intelligent Mechatronics*, 2007.

[14] O. Koch and S. Teller, "Wide-area egomotion estimation from known 3D structure," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[15] A. Eliazar and R. Parr, "DP-SLAM 2.0," in *Proc. IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 2004.

[16] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proc. IEEE Inter. Conf. on Robotics and Automation (ICRA)*, 1999.

[17] A. Eliazar and R. Parr, "Learning probabilistic motion models for mobile robots," in *Proc. 21st Inter. Conf. on Machine Learning (ICML)*, 2004.

[18] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media Inc., 2008.

[19] M. Bertalmío, A. Bertozzi, and G. Sapiro, "Navier-Stokes, fluid dynamics, and image and video inpainting," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[20] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[21] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[22] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *Proc. 12th Inter. Symposium on Experimental Robotics (ISER)*, 2010.