Accurate On-Line 3D Occupancy Grids Using Manhattan World Constraints

Brian Peasley and Stan Birchfield
Electrical and Computer Engineering Dept.
Clemson University, Clemson, SC 29634

Alex Cunningham and Frank Dellaert
School of Interactive Computing
Georgia Institute of Technology, Atlanta, GA 30332

***Abstract*— In this paper we present an algorithm for constructing nearly drift-free 3D occupancy grids of large indoor environments in an online manner. Our approach combines data from an odometry sensor with output from a visual registration algorithm, and it enforces a Manhattan world constraint by utilizing factor graphs to produce an accurate online estimate of the trajectory of a mobile robotic platform. We also examine the advantages and limitations of the octree data structure representation of a 3D environment. Through several experiments in environments with varying sizes and construction we show that our method reduces rotational and translational drift significantly without performing any loop closing techniques.**

I. INTRODUCTION

Three-dimensional (3D) modeling of the environment is an important tool in robotics that has received much attention in past decades. The advent of ubiquitous range sensing has made it more relevant than ever, with applications in both the exploration and navigation of unknown environments, as well as the manipulation of objects in those environments.

Of the many available 3D representations, multi-resolution volumetric occupancy grids are a promising approach for robotics. While both point clouds and surface representations are relevant for many applications, occupancy grids have the advantage that they provide the ability to reason about known versus unknown regions of space. In addition, by fusing the high-volume stream of measurements into a finite grid, both storage and computational requirements remain bounded and manageable, especially if multi-resolution storage schemes are used. A recent implementation of these ideas is OctoMap [22], which uses an octree-based data structure to accumulate data probabilistically while at the same time compressing the required storage down to a mere couple of bits per child node, maintaining the distinction between occupied, unoccupied, and unknown cells.

However, the main strength of an occupancy-based map, namely its ability to provide a compact representation of the scene, is perhaps also its greatest weakness. Promising to obviate the need to store raw range data for long periods of time, the representation is unable to correct large mistakes because the data are discarded as soon as they are assimilated into the map. This drawback is particularly apparent in the case of loop closure, where a single frame of data can necessitate large adjustments in the map representation. Because of this limitation, current implementations (such as [22]) assume that the robot's pose throughout a sequence is known at map construction time.

We are motivated to consider the problem of building 3D maps using an octree representation. The overwhelming amount of data available from an RGBD sensor makes it impractical to store all raw data prior to constructing the map, if online processing is desired. To facilitate the assimilation and discarding of such data as it is acquired, we propose to take advantage of the "Manhattan world" assumption in order to ensure the accuracy of the pose estimate. As the robot drives around a previously unexplored environment, the data acquired by the sensor is used not only to populate the map but also to compute the transformation of the robot between consecutive frames. These transformations, along with readings acquired by an odometry sensor, are fed to a pose-based SLAM (Simultaneous Localization and Mapping) algorithm to estimate the robot's pose on-line. Feature correspondence along with the Manhattan world assumption combine in a powerful way to significantly reduce translational drift and to essentially remove rotational drift. Results on several large environments validate the method's ability to build online octree-based maps without the need for correction, even in the presence of loop closure, demonstrating the benefits of our approach. It is important to note that our approach is not limited to occupancy grids but can be applied regardless of the map representation.

II. PREVIOUS WORK

Occupancy grids have been a popular representation for robot mapping since the pioneering work of Moravec and Elfes [15]. However, as pointed out by a number of researchers [9], [21], the grid-based approach does not facilitate loop closing because it is unable to handle pose uncertainty. The most common approach to

simultaneous localization and mapping (SLAM) is to store a separate map with each particle, so that when information is obtained that renders previous calculations invalid, the data stored in the particle set can be used to correct the mistake [8]. However, this requires either all the data to be stored, or for multiple maps to be retained, both of which negate one of the main strengths of the grid-based representation. One solution would be to quickly rasterize the map into an occupancy grid whenever requested, as in [19], but this solution also requires the raw data to be stored.

Several researchers have extended the idea of grid-based representations to height maps that include the distance above the ground for each grid cell. Such an approach is explored by Marks et al. [14], in which the robot is run in an environment with high visibility, so that the large overlap in field of view between various viewpoints minimizes the effects of loop closure. Another approach is that of Pfaff et al. [18], in which a graph-based algorithm operating on all the data is used for loop closure, though an occupancy grid is used for the final representation. A similar approach for a flat ground is adopted by [7], which also builds on the idea of Lu and Milios [13] that requires all data to be retained.

In the computer vision literature, several methods have been developed in recent years to use the Manhattan world assumption for reconstruction. Furukawa et al. [5] describe an algorithm that employs a multiview stereo approach for estimating the 3D coordinates of a sparse set of feature points. From these points, dominant plane directions are extracted, from which plane hypotheses are generated. Markov random fields are then used to compute per-view depth maps, even for relatively textureless scenes. In followup work [6], an automated system for 3D reconstruction of architectural scenes is described using a combination of Manhattan world multiview stereo, structure-from-motion, and graph cuts for axis-aligned depth map integration. Additional research endeavors [16] [17] demonstrate the ability to perform online SLAM using planes extracted from point clouds.

The approach of Flint et al. [4] uses visual SLAM to obtain key frames in a video sequence, along with the pose of the sensor for each key frame. Using these poses, along with line segments detected in the key frames, an EM algorithm is used to estimate the rotation of the SLAM coordinate frame with the axis-aligned coordinate frame. This rotation yields the vanishing points in the images, which imposes a powerful constraint for detecting even faint axis-aligned edges, from which the wall, ceiling, and floor planes can be reconstructed.

Our approach combines the simplicity and compactness of the occupancy grid representation, with the power of the Manhattan world constraint to overcome the problem of rotational drift. Combined with visual registration, our system provides a simple but compelling way to build accurate, online 3D maps without the need to store all the data.

III. Octree Scene Representation

An occupancy grid [3], [20] is an efficient way to integrate sensor readings, while an octree efficiently represents a 3D occupancy grid. An octree is a hierarchical data structure, where each node represents a cubic volume of space (voxel), and each node is either a leaf node or has eight children representing eight equally-sized cubic subsets of the parent's cubic volume. Octrees are flexible representations, able to capture arbitrarily shaped environments at any desired level of resolution, the resolution being determined by the minimum voxel size. Research has shown [1] that octrees are able to efficently represent scenes, requiring approximately 2.6 bits to store each cubic volume. An illustrative example of the octree data structure can be seen in Figure 1.

One of the more compelling implementations of octrees is the OctoMap, recently introduced by Wurm *et al.*[22]. By explicitly representing three types of voxels (occupied, unoccupied, and unknown), the data structure is able to differentiate between areas of the environment that have been determined by the sensor to be free of obstacles and areas for which no information has yet been obtained. Each node is represented by two bits capturing one of four states, that is, whether the voxel is a leaf node, and therefore one of the three types just mentioned, or whether it is a parent node. Utilizing a clamping update policy, nodes that are saturated to either a minimum or maximum value indicate with a high degree of certainty whether they are occupied, leading to a binarized maximum likelihood decision. In combination, these implementation details yield a compact representation that, when binarized, can represent sub-meter resolution of areas more than 10,000 square meters in size with considerably less than one megabyte of storage.

However, this tremendous gain in efficiency comes at a price. The reason that occupancy grid maps are able to save so much space is that they discretize the sensor readings prior to storage. This discretization discards information and is a reasonable approach only when the pose of the sensor is known. Therefore, occupancy grid-based approaches typically perform in a batch fashion, first estimating the robot pose throughout the entire data collection process, then compressing the data in the occupancy grid structure. Such an approach does not naturally extend to online operation because drift in the pose estimation causes increased errors in the map over time. We will incorporate a constraint on the environment to reduce online pose estimation errors.

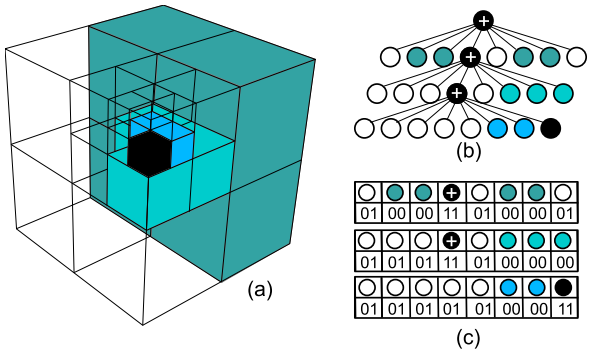


Fig. 1. An octree representation (b) and corresponding compact bit encoding (c) of a simple 3D model (a). White indicates areas of the map that are unoccupied, color indicates areas that are unknown, black indicates occupied areas, and black with a white cross indicates nodes that have children. At each level the 3D model is scanned in clockwise order around the top half, then the bottom half.

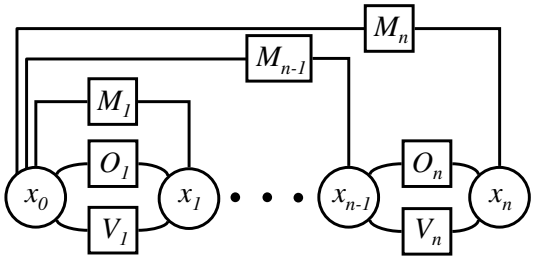


Fig. 2. Factor graph used to estimate trajectory of robot. Three types of factors are used: relative pose by robot odometry (O_i), relative pose by visual registration (V_i), and Manhattan world constraint (M_i). The ith pose is given by x_i, $i = 0, \ldots, n$.

IV. Factor Graphs for PoseSLAM

The underlying inference technique used in this paper employs the Smoothing and Mapping (SAM) technique for representation and incremental solving of the SLAM problem as inference over an undirected graphical model; a detailed explanation of the approach can be found in [2]. In a pose-only formulation of SLAM, we solve for the trajectory $X \triangleq \{x_i\}$, given the measurements $Z \triangleq \{z_k\}$, which we represent in an undirected, bipartite *factor graph.*

The measurements are typically connected to a small number of variables, such as binary pose constraints calculated by visual registration or odometry. As an inference problem, we compute a MAP estimate over all measurements

$$X^* \triangleq \underset{X}{\text{argmax}} \; P(X|Z) = \underset{X}{\text{argmax}} \; P(X,Z) \quad (1)$$
$$= \underset{X}{\text{argmin}} \; -\log P(X,Z),$$

using Bayes' rule to cast inference as a nonlinear least-squares optimization problem in which we minimize the negative log likelihood:

$$X^* = \underset{X}{\text{argmin}} \; \frac{1}{2} \left\| h(X) - Z \right\|^2_\Sigma , \quad (2)$$

where $h(X)$ is a generative measurement model that predicts all sensor measurements given the poses.

The sparsity of the relationships between variables motivates the use of a graphical formulation, in which we factor the optimization problem into separate *factors* $f_k(X_k, z_k)$, where each factor is a loss function $f_k(X_k, z_k) = \frac{1}{2} \left\| h_k(X_k) - z_k \right\|^2_{\Sigma_k}$ operating on the subset X_k of X associated with z_k. In this framework, $h_k(X_k)$ is the generative measurement model for the given sensing modality, with a local measurement covariance Σ_k. We can write the full loss function as $L(X) = \sum^k f_k(X, z_k)$. As these factors are independent, we can easily add different types of constraints to the graph.

Direct nonlinear optimization algorithms, such as Levenberg-Marquardt, can solve this problem in batch through recursive linearization of the full system around the current estimate X, successively computing updates δ until convergence:

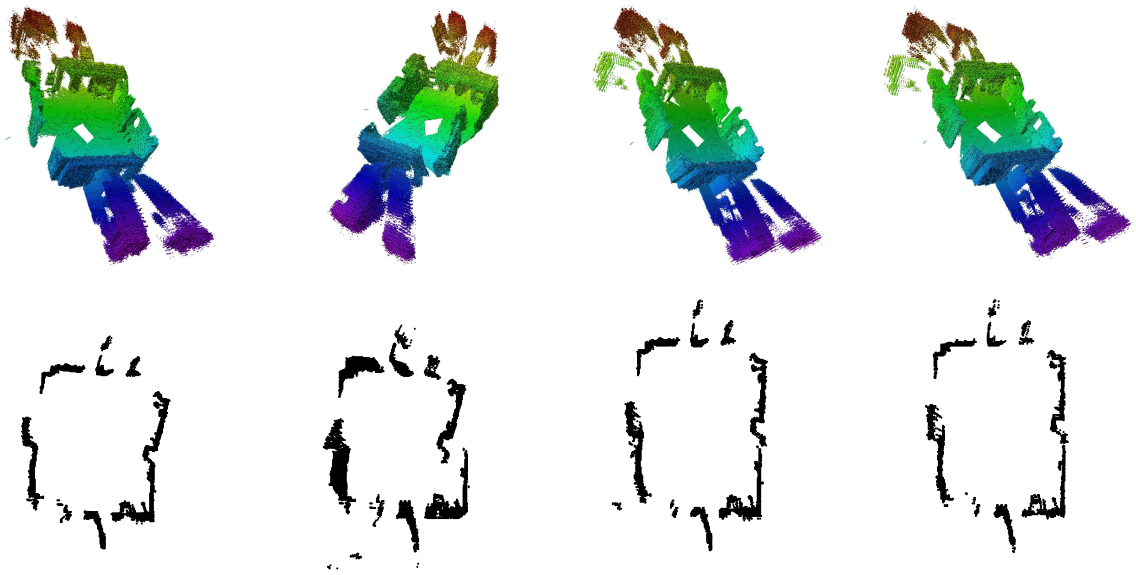
$$\delta^* = \underset{\delta}{\text{argmin}} \; \frac{1}{2} \left\| h(X) + H(X)\delta - Z \right\|^2_\Sigma \quad (3)$$
$$= \underset{\delta}{\text{argmin}} \; \frac{1}{2} \left\| A\delta - b \right\|^2_\Sigma , \quad (4)$$

where $H(X)$ is the Jacobian of $h(X)$ at X.

We reduce the full linearized system of (3) to a large block-wise sparse least-squares problem (4) to solve for δ^*. To avoid repeatedly solving a large system online, we again exploit sparsity and represent the solution process with a Bayes tree [11], which performs incremental multi-frontal Cholesky factorization to update the current estimate as we add new pose constraints. For more details on the iSAM (incremental SAM) algorithm, see [12].

V. Manhattan Constraint

While the visual registration between consecutive frames helps significantly to reduce drift in the pose estimation of the robot, errors nevertheless persist. For large environments, even small rotational errors cause large errors over time, because positional errors are on the order of $\ell \sin \Delta\theta$, where ℓ is the length traveled, and $\Delta\theta$ is the rotational error. For example, even a



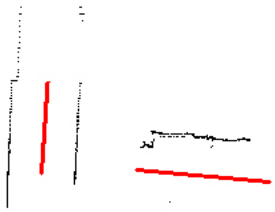
(a) Robot Odometry

(b) Visual Registration + Robot Odometry

(c) Manhattan Constraint + Robot Odometry

(d) Manhattan Constraint +Visual Registration + Robot Odometry

Fig. 4. Top: A 3D octree-based map of a laboratory environment of size 10.6 by 20.6 meters, constructed using 590 scans from the RGBD sensor and modeled with 30 mm resolution. Bottom: A 2D plan view of the map obtained by taking a horizontal slice through the 3D map. From left to right: results from various versions of the algorithm, demonstrating the ability of visual registration to reduce translational drift, and the Manhattan constraint to remove rotational drift. The final map required just 1.9 MB of disk space.



(a) (b)

Fig. 3. Output from the RANSAC line fitting algorithm. The black dots indicate the points along a scan line from the point cloud. The red line is in the dominant direction of the points in the scan line. (a) The robot in the middle of a corridor, (b) The robot in an area where only one wall is visible.

rotational error of just 1 degree will produce positional errors of nearly two meters when traversing a length of 100 meters.

To overcome this rotational drift error, we propose to use a Manhattan world assumption. According to this assumption, every pair of surfaces of interest are either parallel or perpendicular to one another. One key advantage of the Manhattan world assumption is that its enforcement does not require precise correspondence to be established between pairs of frames. Rather, in the context of a 3D sensor, all that is required is that planes be clustered appropriately into one of three mutually orthogonal bins. Because the relative rotation between consecutive frames is on the order of a few degrees at most, and because the bins are 90 degrees apart, essentially *zero rotational drift for indefinite periods of time* can be achieved in environments in which the assumption holds, with only mild assumptions on the ability of the algorithm to associate planes correctly. This removal of the most dangerous of the two types of drift enables the compression abilities of the occupancy grid-based approach to be fully utilized without significant fear of regretting the loss of data that would otherwise have been imperative for proper handling of loop closure.

Unlike the other factors which are added to join consecutive robot poses in the graph, the Manhattan constraint always connects the current pose to the initial pose, where the world coordinate frame is defined. This is illustrated in Figure 2.

In order to apply the Manhattan constraint on the geometry of the scene it is necessary to isolate features in the environment that will allow us to find the rotation parameters for this constraint. The most obvious features in indoor environments that are either parallel or orthogonal are walls. As mentioned earlier, in

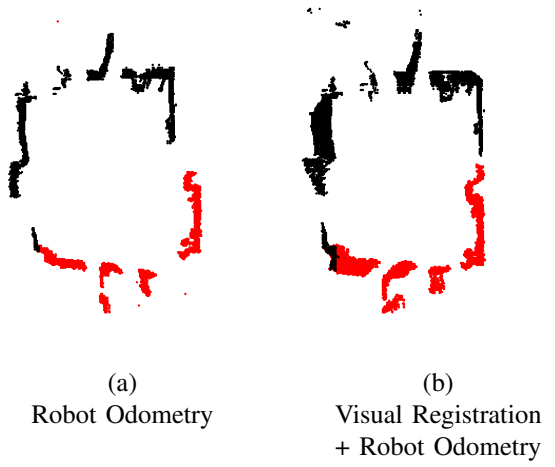
order to calculate the rotation appropriate for geometric alignment, explicit correspondence of items in the point cloud is not necessary. Rather, we only need to find the normal of a single wall in the current frame, along with the assignment to the same plane sensed in the previous frame. To determine such a plane, we apply RANSAC to the depth data in a horizontal scan of the RGBD sensor to find the dominant line. Output from this approach can be seen in Figure 3.

Once the relative rotation between walls of consecutive frames has been established, the orientation of the current frame and the global coordinate frame is automatically achieved, since the orientation of the previous frame is already known. This zero-drift principle of the Manhattan World constraint is similar to the driftless approach of matching the current sensor reading to the model rather than to a previous sensor reading, employed in KinectFusion [10].

VI. Experimental Results

To evaluate the proposed approach we constructed maps from data recorded in three different indoor environments. The first environment was a small laboratory where our mobile robot drove around the perimeter of the room. The purpose of this experiment was to test the ability of our method to handle rotational and translational drift without explicitly handling any loop closures. The second environment was a building on our campus consisting of a long main corridor and two side corridors, with no opportunity for loop closure. This experiment tested the performance of the rotational constraint imposed by the Manhattan world assumption over a long distance. The third environment was another large building on our campus containing many opportunities for loop closure, thus allowing us to measure the error in our results with and without such techniques. Our hardware platform consisted of an ActivMedia Pioneer P3AT mobile robot with a forward-facing Kinect RGBD sensor.

Results from the first experiment in the laboratory can be seen in Figure 4. Four different reconstructed maps demonstrate the influence of the various terms in the factor graph. Figure 4(a) shows the constructed map using only odometry data. As expected, both rotational and translational drift are present, causing noticeable errors in the map. Figure 4(b) shows the map constructed using cues from both the robot odometry and the visual registration. Although visual registration could be used to reduce both translational and rotational drift, we employ it only for the former in order to better show the power of the Manhattan assumption. As a result of this limitation, the addition of the visual registration causes the right wall to move to the left. At first inspection it may not be obvious that the map constructed in Figure 4(a) is worse than that of Figure 4(b). However, if the coincidental combination of translational and rotational drift is separated, then the errors due to odometry alone are more readily apparent, see Figure 5. While the Manhattan world constraint is sufficient for removing the rotational drift from the map, it does not address the problem of translational drift. The effects of the latter can be seen by the slight misalignment of the two pieces of the wall on the right side of the map (just above the concavity) in Figure 4(c). This gap is removed in Figure 4(d) by the addition of visual registration.

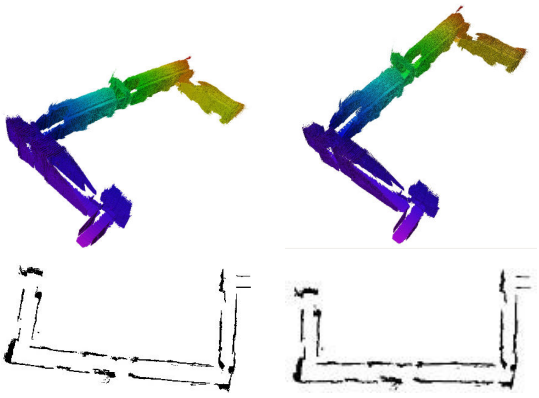


(a) Robot Odometry (b) Visual Registration + Robot Odometry

Fig. 5. A comparison of the system with and without visual registration, showing the rather larger translational error reduction. The points in red were rotated about the bottom left corner of their respective maps to isolate the translational drift from the rotational drift.

Figure 6 shows the resulting maps of the second environment. Due to the size of the building (the length of the main corridor is approximately 55 meters), there is much room for the robot odometry and visual registration to drift. This drift is shown in Figure 6(a), where significant rotational drift causes noticeable errors in the map. By adding in the Manhattan world constraint to the factor graph we are able to remove all rotational drift from the map, see Figure 6(b), even though there is no opportunity to perform loop closure.

In the third experiment the robot was driven around the floor of a large building containing several intersecting hallways. This experiment shows not only the ability of the Manhattan world constraint to remove rotational drift over an extended period of time, but also the ability of the visual registration to reduce the translational drift to a surprisingly low level, without any loop closure. Figure 7(a) shows the map with robot odometry and visual registration, which exhibits noticeable distortions over the length of the path. Of course,



(a) Robot Odometry + Visual Registration

(b) Manhattan Constraint + Visual Registration + Robot Odometry

Fig. 6. A large building with no loops. (Size was 23.9 by 47.8 meters, modeled with 30 mm resolution using 3,300 scans). The addition of the Manhattan constraint enforces perpendicularity of the walls. The map was saved to disk using only 2.1 MB.

existing techniques can handle such environments, but only by requiring that raw data are kept until such a time as loop closure is performed. In contrast, our approach, shown in Figure 7(b), is able to significantly reduce rotational and translational drift over an extended period of time, thus enabling data to be discarded as they are assimilated into the map.

The particular path driven by the robot is illustrated in Figure 8. Starting at intersection 1, the robot drove (from a bird's eye point of view) north, then east, then south to 2. Turning west, it traveled through 3, then south to 4, then east to 5, after which it encountered 3 and 2 again before heading south to 6 and then completing the bottom loop to end at 6. Opportunities for loop closure therefore occurred at intersections 1, 3, 2, 4, 5, and 6, in that order. Table I shows the errors occurring at the six different potential loop closure locations for the particular path driven. These errors were obtained by manually viewing the video and selecting, for each intersection, two key frames in which the robot was approximately in the middle of the intersection; the distance between the two estimated robot locations yielded the error. Due to imprecision in this measurement technique, these numbers should be used as relative rather than absolute assessments of error. Nevertheless, the Manhattan assumption reduces the error by about an order of magnitude.

The amount of memory saved in using the octree-based representation rather than retaining all the raw data is approximately three orders of magnitude, as shown in Table II and Figure 9.

Intersection	RO	VR + RO	VR + RO + Manhattan
1	12.0 m	1.0 m	0.1 m
2	31.5 m	8.3 m	0.5 m
3	16.1 m	3.7 m	1.0 m
4	25.0 m	6.7 m	0.8 m
5	12.7 m	10.7 m	0.5 m
6	15.5 m	4.5 m	0.5 m

TABLE I

ERROR FOR SIX INTERSECTIONS FROM THE ENVIRONMENT SHOWN IN FIGURE 7. THE COLUMNS SHOW THE RESULTS USING VARIOUS COMBINATIONS OF ROBOT ODOMETRY (RO), VISUAL REGISTRATION (VR), AND THE MANHATTAN CONSTRAINT. THE PATH OF THE ROBOT AND THE INTERSECTION POINTS CAN BE SEEN IN FIGURE 8.

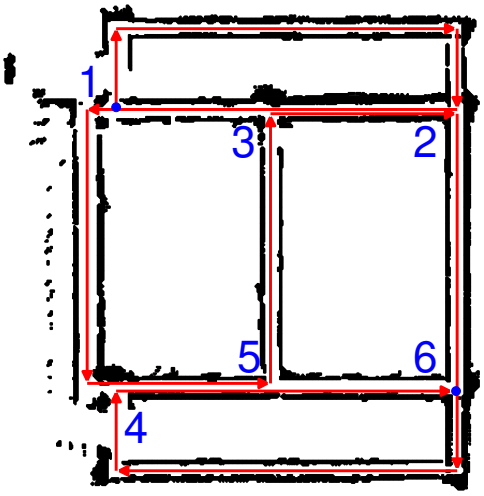
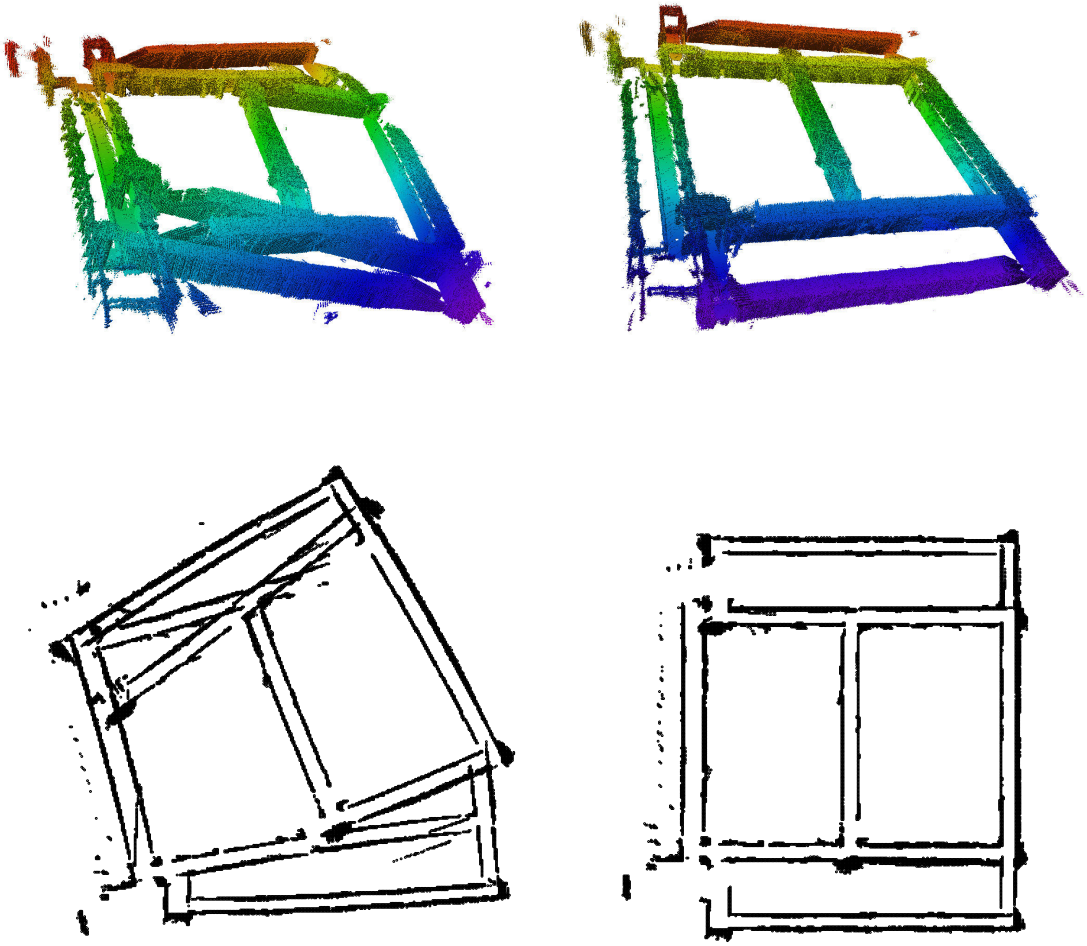


Fig. 8. Path taken by the robot in the generation of map in Figure 7. The robot moved in the direction of the arrows, encountering the intersection points in the following order: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 6$. Therefore, the potential loop closures would have been, in order, $(1, 3, 2, 4, 5, 6)$. The blue dots indicate the start and end points of the path.

	in memory	on disk
Point Cloud	4,350,000 kB	590,000 kB
Octree	941 kB	363 kB
Compression	4622:1	1625:1

TABLE II

AMOUNT OF SPACE REQUIRED TO STORE THE ENTIRE MAP IN FIGURE 6 IN BOTH MEMORY AND ON DISK.



(a)
Robot Odometry
+ Visual Registration

(b)
Manhattan Constraint
+ Visual Registration
+ Robot Odometry

Fig. 7. An environment with several intersecting corridors. The building is 52.6 by 53.2 meters and modeled with 30 mm resolution using 7,789 RGBD scans. The building was traversed multiple times in order to map the environment in its entirety. The benefit of the Manhattan assumption is evident. The map required just 5.6 MB of disk space.

VII. Conclusion

In this paper we have presented a system that builds online 3D maps by overcoming one of the primary disadvantages of occupancy-grid-based representations, namely their rigidity. By focusing on the reconstruction of man-made environments, we are able to remove essentially all rotational drift by enforcing a Manhattan world constraint on the geometry of the map. In addition to removing rotational drift in this manner, we are also able to reduce translation drift significantly by combining the odometry readings with output from a visual registration algorithm. Both relative pose measurements and the Manhattan world constraint are added to a factor graph to yield an optimized online trajectory of the mobile robot with greatly reduced rotational and translational drift. Together, these aspects of the system result in geometrically accurate maps for large indoor environments while utilizing the ability of octrees to perform significant data compression on the 3D maps. Experimental results on several large environments show the ability of our system to accurately construct 3D maps from RGBD data without having to perform any loop closures. Future work will be aimed at further characterizing the resulting error, generalizing the tech-

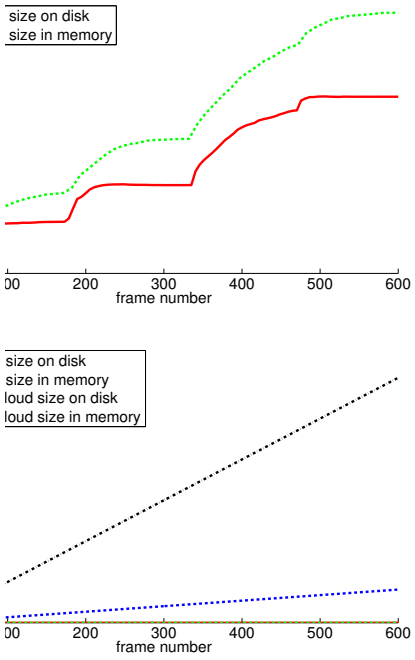


Fig. 9. The amount of space required by the map on disk and in memory by point cloud and octree representations, as a function of frame number. The plots show the sizes for the map constructed in Figure 4. The top plot is a zoomed-in view of the bottom plot (notice the red and green lines overlaid on the x axis in the bottom). The octree reduces storage requirements by more than three orders of magnitude.

nique to non-Manhattan worlds with other types of global constraints, and performing interleaved mapping and navigation in dynamic environments. We will also investigate the use of a more robust plane matching algorithm to handle cluttered areas where simple plane matching will not necessarily yield the walls.

VIII. ACKNOWLEDGEMENTS

This research was supported by the U.S. National Science Foundation under grant IIS-1017007.

REFERENCES

[1] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In *Thirteenth Eurographics Workshop on Rendering*, 2002.

[2] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, Dec 2006.

[3] A. Elfes. Occupancy grids: A probabilistic framework for robot perception and navigation. *Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

[4] A. Flint, C. Mei, I. Reid, and D. Murray. Growing semantically meaningful models for visual SLAM. In *Computer Vision and Pattern Recognition CVPR*, 2010.

[5] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Manhattan-world stereo. In *Computer Vision and Pattern Recognition CVPR*, June 2009.

[6] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Proceedings of the International Conference on Computer Vision*, Sept. 2009.

[7] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *IEEE Intl. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, November 2000.

[8] D. Häehnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[9] G. Q. Huang, A. B. Rad, and Y. K. Wong. Online SLAM in dynamic environments. In *Proceedings of the 12th International Conference on Advanced Robotics*, pages 262–267, July 2005.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th ACM Symposium on User Interface Software and Technology*, pages 559–568, 2011.

[11] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236, Feb 2012.

[12] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, Dec 2008.

[13] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, pages 333–349, Apr 1997.

[14] T. K. Marks, A. Howard, M. Bajracharya, G. W. Cottrell, and L. Matthies. Gamma-SLAM: Using stereo vision and variance grid maps for SLAM in unstructured environments. In *Proceedings of the International Conference on Robotics and Automation*, 2008.

[15] H. Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 116–121, Mar. 1985.

[16] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga. Online three-dimesional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27(1):52–84, Jan. 2010.

[17] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga. Fast registration based on noisy planes with unknown correspondences for 3-D mapping. *IEEE Transactions on Robotics and Automation*, 26(3):424–441, June 2010.

[18] P. Pfaff, R. Triebel, and W. Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *Intl. J. of Robotics Research*, 2007.

[19] J. Strom and E. Olson. Occupancy grid rasterization in large environments for teams of robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[20] S. Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, 2003.

[21] S. Thrun. Robotic mapping: a survey. In *Exploring artificial intelligence in the new millennium*, pages 1–35. Morgan Kaufmann, Inc., 2003.

[22] 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 *Proceedings of the ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, May 2010.