Rapid 3D Visualization of Indoor Scenes Using 3D Occupancy Grid Isosurfaces

Ran Zask and Matthew N. Dailey
Computer Science and Information Management
Asian Institute of Technology
ran@cs.ait.ac.th, mdailey@ait.ac.th

***Abstract*—In many mobile robotics applications involving exploration of unknown environments, it would be extremely useful to provide human operators with a real-time 3D visualization of the environment the robot is exploring. Although a great deal of progress has been made in the separate fields of photorealistic structure from motion and realtime vision-based robot localization and mapping (SLAM), the ultimate goal of real-time 3D visualization of the environment a robot is exploring has yet to be realized. In this paper, we present a simple and efficient incremental algorithm for 3D modeling amenable to real-time implementation. The algorithm creates a texture-mapped polygonal mesh model of the environment from a monocular video feed or sequence of images. The key to the algorithm's simplicity and efficiency is the use of the isosurface of a coarse 3D occupancy grid that is incrementally updated as new images arrive. The isosurface-based reconstruction provides low metric accuracy but helps to filter measurement noise and allows rapid construction of a 3D visualization. We demonstrate the practicality and effectiveness of the algorithm by using it to generate an OpenGL model of a real indoor environment.**

I. Introduction

Many existing systems for mobile robot navigation and control incorporating sonar, laser, or vision sensors are able to create 2D maps suitable for navigation of visualizing the environment. See [1]–[3] for just a few examples. However, although 2D maps are useful, Yanco and Drury [4] report that even with access to a 2D map during urban search and rescue operations, operators still tend to spend on average 30% of their time *acquiring situation awareness*, that is, trying to understand the robot's location, surroundings, and status.

One technology that could possibly improve search and rescue operators' situation awareness and mission planning is *textured 3D map visualization*. In this paper, we explore the feasibility of rapidly constructing a texture mapped 3D model of a mobile robot's environment based on a monocular image stream.

There is a very large literature on 3D modeling in computer vision and robotics. Using techniques from structure from motion (see, e.g., [5], [6]), structured lighting [7], or image-based rendering (see, e.g., [8]) it is possible to synthesize photorealistic views of complex real-world environments. At the same time, towards simultaneous localization and mapping (SLAM) for mobile robots, techniques for real-time vision-based estimation of 3D scene structure are beginning to emerge [9]. However, currently, the photorealistic methods require significant amounts of offline processing, and the real time vision-based structure estimation methods provide only sparse information about a scene, without surface estimation or texture maps.

In this paper, we show that if the requirement for photorealistic rendering is relaxed, it is possible to obtain the benefits of both the structure-from-motion and SLAM approaches. We construct 3D surface models, incrementally, based on a monocular image sequence, using a simple and efficient algorithm. We believe that the resulting models, though only roughly accurate, would be sufficiently detailed to enhance operators' situation awareness during teleoperation and help them command the robot.

There are two main ways to construct surface models: from 3D point sets or from volumetric data. The first method triangulates 3D point sets to obtain a polygonal mesh (see, for example, [5]). The main alternative method, the volumetric approach, uses a sensor model to update an occupancy grid as images arrive. Occupancy grids represent the environment as a set of cells that indicate the probability of occupancy by some obstacle. The 2D occupancy grid is commonly used with sonar and laser sensors for localization, mapping, and navigation [1], [2], [10]), and the 3D grid is a straightforward generalization. For example, Moravec [7] uses stereoscopic images and structured lighting to construct a fine-scale 3D grid then renders the resulting grid directly.

The volumetric grid-based approach has several advantages over point-based modeling. Measurement quantization helps to reduce noise, and incremental fusion of measurements from different sensor positions is trivial. When the goal is to rapidly generate a 3D model incrementally, grids offer the additional benefit of allowing us to manipulate the tradeoff between speed and accuracy by merely changing the cell size.

Our work is similar to some previous work using grid isosurfaces and omnidirectional multibaseline stereo [6], but in our work, recognizing that operator situation awareness only requires an approximate model that represents the large-scale structure of the environment, we incrementally construct our 3D models from a sequence of images using a standard monocular camera and a coarser-scale 3D occupancy grid. We use a grid cell size small enough that the environment can be readily visualized and explored yet large enough to make grid updates and isosurface calculation fast.

II. Occupancy Grid Isosurfaces

Our method for incremental 3D modeling takes as input an image sequence and returns as output a texture mapped 3D model of the scene. For each image, we extract feature points, obtain correspondences with previous frames, estimate 3D points and camera positions, update the occupancy grid with new 3D points, compute the grid's isosurface, and modify the current 3D mesh model accordingly. Here we describe each of the important steps of the algorithm in detail then summarize the entire algorithm in Section II-F.

A. The 3D data

After we acquire each image, we undistort it and extract SIFT [11] features from it. To establish initial correspondences between two sets of SIFT features, we find, for each keypoint in one image, the most similar keypoint in the other image according to the dot product. We accept that correspondence if the keypoint's similarity to the most similar keypoint is above threshold and higher than that of the next most similar keypoint by some ratio. After obtaining these initial correspondences, we eliminate outliers using RANSAC estimation of the fundamental matrix [12]. We then find the 3D points and relative camera positions using an incremental bundle adjustment procedure similar to that of Pollefeys et al. [5] except that we use a pre-calibrated camera and undistorted images, thus reducing the complexity of the optimization.

Although this straightforward sparse structure from motion method is relatively slow, it could be replaced by a more efficient real-time SLAM algorithm such as that of Davison et al. [9] without affecting the rest of the system, so long as sufficient care is taken to only return high-confidence points to the 3D model construction algorithm.

B. Occupancy grid

We use two types of 3D occupancy grids: a single *global* grid G that accumulates all of the occupancy information and a *local* L grid for each camera position that contains the camera position and the 3D points visible from that camera position.

For fine grids aimed at photorealistic reconstruction, sophisticated sensor models are necessary to add evidence of "emptiness" to the grid cells along each ray from the camera sensor to each detected 3D point and to add evidence of "fullness" to the grid cells corresonding to the detected 3D points. However, since our goal is to rapidly construct an approximate model using a coarse grid, we find that it is sufficient to simply "fill in" the grid cells interior to the convex hull of the full grid cells corresponding to detected points and the camera center. Our criterion for grid cell fullness is based simply on the number of visible point features that map to the given cell.

Clearly, when multiple viewpoints are considered, the filled-in grid is not what we want. For example, the convex hull of a set of points in an "L"-shaped corridor would not preserve concavities in the 3D structure of the environment. This is why we perform the filling operation in a local grid L containing only the 3D points visible from the current camera position. After filling, the information in the local grid for the current camera position is combined with the current global grid by taking the union of the filled cells in the two grids to obtain a new global grid G that is propagated to the next iteration.

C. Isosurfaces

Isosurfaces are commonly used in computer graphics and medical image processing to visualize volumes based on a 3D grid containing the discrete values of a density function. For example, when working on CT and MRI images, the grid might represent the densities of various bodily organs and cavities. In order to visualize volumetric data effectively, we normally need to convert it to surface form, then we can texture map and visualize the surface. The isosurface of a grid for a particular isovalue is the set of all points whose interpolated density equals that isovalue. Most implementations of isosurface extraction use the marching cubes algorithm [13] and return a tessellated triangle-based surface. We use Matlab's custom isosurface algorithm [14], which also returns a triangulated surface S given a grid G but avoids patent issues with the marching cubes algorithm.

D. Incremental modeling

When a new image introduces a new 3D point set X, we first associate the new points with their corresponding grid cells to get a set of voxels. When the total number of points associated with a cell exceeds a threshold k, the cell is deemed "occupied." Let the set of newly occupied cells (those previously unoccupied in G) be V^{new}. We create a local grid L containing V^{new} along with the current camera position C then fill the convex hull of the resulting voxel set as previously described. Finally, we merge L with G to obtain a new occupancy grid G and new isosurface S.

E. Texture mapping

As we explore new regions of a given environment, the volumetric 3D model and corresponding isosurface grows. In general, some of the polygons (triangles) from the previous 3D model M will disappear and some new polygons will appear. To accomplish this, we classify model triangles into the sets "new," "old," and "deleted." We project each "new" triangle into the current image, obtain the appropriate texture coordinates, and then add the new triangle with its texture map to M. Similarly, we remove from M any triangles classified as "deleted." To facilitate search and update, we uniquely identify each triangle by its centroid.

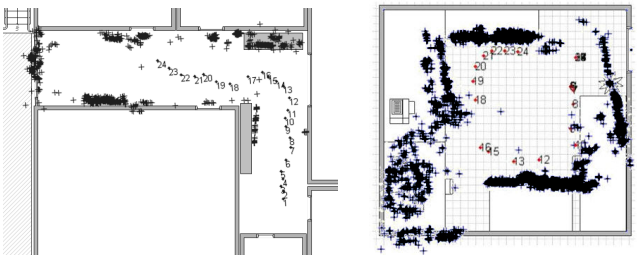
F. Incremental modeling algorithm summary

Here we provide a pseudocode summary of the incremental 3D modeling algorithm.

Algorithm Build-Incremental-Model

Input: image sequence

Output: 3D surface model M with texture map



(a) (b)

Fig. 1. Experimental scenes with reconstructed point sets and camera positions. (a) "L"-shaped corridor. (b) 360-degree panorama of a small office.

1: $M \leftarrow \emptyset$ {initial 3D model}
2: $G \leftarrow \emptyset$ {initial global grid}
3: **for each** image I **do**
4: $X \leftarrow$ GET-3D-POINTS(I)
5: $P \leftarrow$ GET-CAMERA-MATRIX(I)
6: $C \leftarrow$ GET-CAMERA-CENTER(P)
7: $V \leftarrow$ VOXELS(X, C, G)
8: $V^{\text{new}} \leftarrow$ GET-NEWLY-OCCUPIED-VOXELS(G, V)
9: **if** V^{new} is empty **then**
10: continue
11: **end if**
12: $L \leftarrow$ FILL-CONVEX-HULL(V, V^{new})
13: $G \leftarrow L \cup G$
14: $S \leftarrow$ GET-ISOSURFACE($G, 1 - \epsilon$)
15: $(P^{\text{old}}, P^{\text{new}}, P^{\text{deleted}}) \leftarrow$ CLASSIFY-POLYGONS(S, M)
16: $M \leftarrow$ DELETE-POLYGONS(P^{deleted}, M)
17: $T \leftarrow$ GET-TEXTURE-PATCHES(P^{new}, P, I)
18: $M \leftarrow$ ADD-NEW-POLYGONS(P^{new}, T, M)
19: **end for**

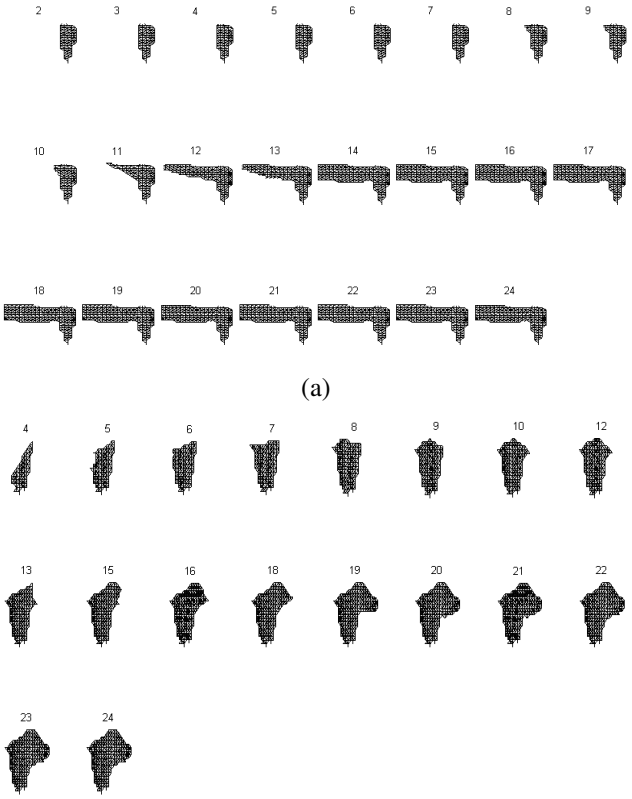
III. EXPERIMENTAL RESULTS

We have experimented with various scenes. In this paper we provide results for an "L"-shaped room and a 360-degree panorama of a small office (see Fig. 1).

We extracted 3D points and camera centers following the steps mentioned in section II-A. At every step, we automatically removed any outlier points with high reprojection error or a large distance from the centroid of the 3D points. Currently, our prototype requires that the empty grid be manually intialized to be large enough to fit the scene of interest.

The results of the incremental model growing process can be seen in Fig. 2, and detailed views of the final models are shown in Fig. 3.

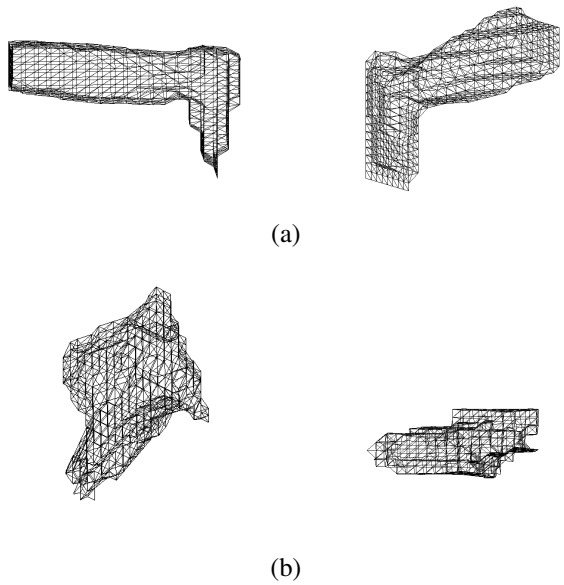
The approach has a few limitations that we plan to address in future work. The first is the assumption that we can fill the convex hull of the set of newly occupied voxels V^{new} (line 12 of Algorithm BUILD-INCREMENTAL-MODEL). This heuristic is appropriate when new regions of the scene are revealed gradually but can cause serious distortion of the scene's structure. For example, if in the very first image in the sequence, two adjacent rooms separated by a wall are visible, the algorithm will eliminate that wall. This problem can be avoided either by restricting the initial view to an



(a)

(b)

Fig. 2. Incremental modeling. A model is shown after updating it with each image of the sequence used for reconstruction. Some images have no effect, and some cause the model to expand. (a) "L"-shaped corridor. (b) Room panorama.



(a)

(b)

Fig. 3. Final 3D models. (a) Two views of the final "L"-shaped corridor model. (b) Two views of the final panorama model.

Fig. 4. Texture-mapped "L"-shaped corridor model. We captured two views from arbitrary camera positions while "flying" through the virtual OpenGL environment. The complete texture map integrates patches from 11 of the 17 total images used for the 3D model reconstruction.

approximately convex region of the scene and ensuring that each new image is acquired from a position very close to the previous image (we have confirmed that planting additional intermediate images into the sequence improves the results), or by using a different method to obtain the local grid L from V^{new}.

Another limitation of the approach arises from the coarseness of the grid. Any feature smaller than the grid cell size (approximately 0.2m $\times$ 0.2m $\times$ 0.2m in the models of Fig. 3) cannot be represented in the final model. When compounded with measurement errors, this means that important features such as a wall between two adjacent rooms could be eliminated from the model. The algorithm should diagnose this situation and ensure that we never "punch through" a wall without sufficient evidence of a door or window.

Figure 4 shows the result of incrementally texture mapping the "L"-shaped 3D model from Fig. 3(a). Although there are distortions due to imperfections in the 3D model structure and greedy selection of the texture patches, the model is sufficiently accurate for a human observer to understand the scene.

To compare these results with the state of the art, our texture-mapped reconstructions, being based on sparse point sets, are less accurate than those of Johnson and Kang [6]. However, whereas Johnson and Kang perform a 2D Delauney triangulation and the iterative closest point (ICP) algorithm for 3D mesh recognition for every update, our method is much more lightweight, directly associating point sets to the grid, and is therefore more amenable to real time implementation. We believe the result is sufficient to enhance teleoperation situation awareness.

IV. Conclusion

In this paper, we have demonstrated the effectiveness of a simple yet efficient method for incrementally constructing texture mapped polygonal mesh models of indoor environments using a single camera. The resulting models are sufficiently accurate for search and rescue situation awareness, but they might not be appropriate for other applications.

In future work, we plan to build a complete real time implementation including a user interface for high-level control of the robot and a Kalman filter-based or particle filter-based SLAM algorithm for camera pose and 3D point estimation. Some of the improvements that may be required to achieve real time performance include porting the software to a compiled language such as C or C++ (parts of the current prototype are implemented in Matlab), to perform bundle adjustment only locally (in the current prototype, bundle adjustment is performed over all images up to the current image), to work with smaller images (e.g. 640$\times$480 rather than the current 2048$\times$1536), to use a faster feature detector such as SURF [15] instead of SIFT, to use octrees instead of uniform grids, and to design a local version of the isosurface algorithm.

Acknowledgments

We thank Sumanta Guha, Kiyoshi Honda, and the AIT Computer Vision Group for valuable discussions and comments on this work. Special thanks are due to Suwan Tongphu, who assisted with the implementation of the OpenGL visualization.

References

[1] A. Elfes, "Sonar-based real world mapping and navigation," *IEEE Journal of Robotics and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[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 *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000, pp. 321–328.

[3] K. Konolige, "Large-scale map-making," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2004, pp. 457–463.

[4] H. Yanco and J. Drury, "'Where am I?' Acquiring situation awareness using a remote robot platform," in *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, vol. 3, 2004, pp. 2835–2840.

[5] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.

[6] A. E. Johnson and S. B. Kang, "Registration and integration of textured 3D data," *Image and Vision Computing*, vol. 17, no. 2, pp. 135–147, 1999.

[7] H. Moravec, "Robot spatial perception by stereoscopic vision and 3D evidence grids," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-96-34, 1996.

[8] M. Uyttendaele, A. Criminisi, S. B. Kang, S. A. J. Winder, R. Szeliski, and R. I. Hartley, "Image-based interactive exploration of real-world environments," *IEEE Computer Graphics and Applications*, vol. 24, no. 3, pp. 52–63, 2004.

[9] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[10] R. Parr and A. Eliazar, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 1135–1142.

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

[12] R. H. A. and Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[13] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[14] The MathWorks, "Source code for the Matlab `isosurface` function," 2007, implementation located in Matlab distribution directory `toolbox/matlab/specgraph/isosurface.c`.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.