Live Structural Modeling using RGB-D SLAM

Nicolas Olivier[1], Hideaki Uchiyama[2], Masashi Mishima[2], Diego Thomas[2], Rin-ichiro Taniguchi[2], Rafael Roberto[3], João Paulo Lima[3,4] and Veronica Teichrieb[3]

Abstract— **This paper presents a method for localizing primitive shapes in a dense point cloud computed by the RGB-D SLAM system. To stably generate a shape map containing only primitive shapes, the primitive shape is incrementally modeled by fusing the shapes estimated at previous frames in the SLAM, so that an accurate shape can be finally generated. Specifically, the history of the fusing process is used to avoid the influence of error accumulation in the SLAM. The point cloud of the shape is then updated by fusing the points in all the previous frames into a single point cloud. In the experimental results, we show that metric primitive modeling in texture-less and unprepared environments can be achieved online.**

I. Introduction

Automated control of cars, drones and robots has been achieved owing to recent advances in odometry and simultaneous localization and mapping(SLAM) technologies [1]. Specifically, visual SLAM(vSLAM) using a monocular camera or an RGB-D camera has become a key technology in various applications because it needs only a camera, and can run in real time even on mobile devices [2]. Given that the stability of localization is degraded under fast camera motion, an IMU is integrated into the vSLAM to support orientation estimation. This framework is generally referred to as sensor fusion or visual-inertial SLAM [3]. Using these state-of-the-art technologies, device pose can be accurately estimated under any type of motion in unprepared environments.

In vSLAM, a map is generally represented by a set of points, lines, or a voxel grid. For example, early work on vSLAM was based on sparse feature points [4], [5]. The density of points increased for the robustness in the case of fast camera motion [6]. As the vSLAM using points fails in texture-less environments, edges extracted from object contours were used [7], [8]. A voxel grid-based approach was proposed to build a dense and accurate map by efficiently fusing 3D measurements from multiple viewpoints [9]. vSLAM using a RGB-D camera also employed the same map representation as when using a monocular camera [10], [11], [12]. These map representations are basically designed for fast and efficient localization tasks. However, a map is not sufficient for representing scene semantics, and therefore needs to be converted into primitive shapes to enable us to understand 3D scenes. Such structural information is also useful for robotic grasping tasks [13].

In our previous work, we proposed a method for incremental structural modeling using sparse points computed from monocular vSLAM [14]. The advantage of our method is that system users can interactively recognize the modeling progress to avoid repeating the measurement in an online manner. This interactivity is practically useful for 3D measurement applications. In the literature, existing structural modeling methods employed a batch-based approach such that data acquired with LiDAR scanners was processed only once [15]. As the estimated shapes were sensitive to the degree of noise in the points, we proposed a method using the generating process of the points in vSLAM so that shapes were stably estimated by analyzing the temporal history of estimating shapes. However, the result was strongly affected by texture in environments, such that the method did not work in texture-less construction fields. Also, the metric accuracy was not well investigated because estimated shapes were up to scale.

In this paper, we propose a method for live structural modeling using RGB-D SLAM so that a shape map containing only primitive shapes can be generated in texture-less and unprepared environments. Compared with our previous work, the processes in the shape matching and shape updating are improved to be robust to error accumulation in the SLAM. As the error accumulation gradually causes the mismatching of primitive shapes during camera movement, we propose to use the history of fusing shapes to suppress the influence of error accumulation. Next, the point cloud of the matched shape is updated by fusing the points extracted in the previous frames. In the experimental results, we show that metric primitive modeling can be achieved online.

II. Related Work

Estimating primitive shapes from a point cloud such as cylinders and planes was a main research issue in structural modeling. Such structural information effectively represents 3D structure with a few shape parameters and can reduce the size of memory required. Also, it is useful in various applications, such as reverse engineering in construction fields. For example, cylinder detection was required to detect pipes in manufacturing plants [16], [17]. Indoor 3D model of a building can be generated with plane detection techniques by approximating a room as a grid space [18]. Basically, these methods are based on a batch process with data acquired offline.

[1]Nicolas Olivier is with Ecole SupErieure d'IngEnieurs de Rennes, France `nicolasolivier@mail.com`

[2]Hideaki Uchiyama, Masashi Mishima, Diego Thomas and Rin-ichiro Taniguchi are with Kyushu University, Japan `uchiyama@limu.ait.kyushu-u.ac.jp`

[3]Rafael Roberto, João Paulo Lima and Veronica Teichrieb are with Voxar Labs, Centro de Informática, Universidade Federal de Pernambuco, Brazil `rar3@cin.ufpe.br`

[4]João Paulo Lima is with Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco, Brazil `jpsml@cin.ufpe.br`

To generate primitive shapes incrementally and online, structural modeling is incorporated into the process of the vSLAMs [19]. Generally, shape constraints can be incorporated into this mapping process. For example, a planar constraint is used to accurately build planes in a map and improve localization accuracy [20], [21]. In the plane map, the incidence and orthogonality of planar surfaces are identified for high-level structural models [22]. In these methods, the target shape was a plane only and more parametric shapes are required to be detected. As an alternative approach, SLAM++ tackled the issue with an object detection technique so that objects registered in the database can be mapped with the vSLAM [23]. As the parametric representation of primitive shapes allows more flexibility in representing a 3D shape [24], we propose a method for stably estimating primitive shapes based on the vSLAM.

III. Overview of Propose Method

The flow of our proposed method is based on our previous work [14], and is illustrated in Figure 1. A vSLAM algorithm is used as an external library that sequentially outputs point clouds in the global coordinate system. We use Real-Time Appearance-Based Mapping(RTAB Map) [25] because it includes all basic functions for RGB-D SLAM such as frame-to-frame/frame-to-map registration with ICP/visual features, and runs in both textured and texture-less environments. For the shape detection, we use Efficient RANSAC [26] to extract some primitive shapes in a point cloud. Given that the Efficient RANSAC does not stably extract primitive shapes from its noisy or partial measurement, we incrementally refine the estimated shapes by fusing the ones computed in the previous frames.

The processes at each iteration are summarized as follows. First, a point cloud represented in a global coordinate system is obtained from the RTAB Map. Next, primitive shapes are extracted from the point cloud using the Efficient RANSAC. Then, the extracted shapes are matched with those in the shape map. If shape parameters of an extracted shape are similar enough to a particular shape in the shape map, the shape parameters and the point cloud are updated. Otherwise, it is stored as a new shape in the shape map. These processes are iterated at every frame. Note that all of detected shapes are stored as new shapes at the first frame.

The following operations are of vital importance in each step. The Efficient RANSAC provides not only accurate primitive shapes but also inaccurate ones. Therefore, the inaccurate shapes are suppressed in the shape extraction. In the shape matching, inaccurate or duplicate objects are again eliminated, and distinct objects represented by similar shape parameters are not fused. In the shape update, the shape parameters and the point cloud in the shape map are refined.

The motivation for live modeling using the vSLAM comes from the fact that it was difficult to use primitives only for the mapping in the vSLAM, owing to the instability of shape detection with the Efficient RANSAC. In our preliminary experiments, we found that the localization accuracy was largely degraded, or localization failed if a shape

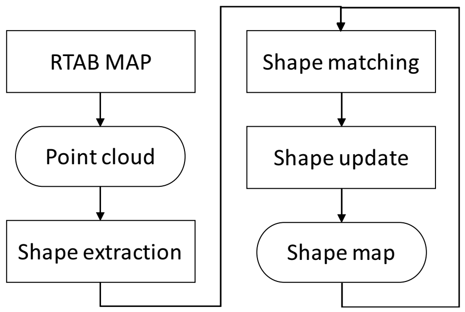


Fig. 1: Flow of proposed method. At every frame, primitive shapes are detected, fused with previously detected ones, finally the shape map is stably generated.

was wrongly extracted at a frame. Also, mapping only the primitives is not feasible in many situations because the scene does not always contain enough primitives for the mapping. Therefore, we propose live structural modeling at the top of the vSLAM, so that device pose can be stably estimated from the vSLAM. Following this, structural modeling runs only when there are primitive shapes in the scene.

IV. Shape extraction

Shape extraction is the process to extract primitive shapes from a point cloud, and is based on the Efficient RANSAC. A point cloud is represented in the global coordinate system defined in the vSLAM. To stably extract a shape from a point cloud, the distribution of points on the shape should be visually comprehensible. However, this is not valid in our case because only a part of an object can be captured from one view, and the acquired points are noisy owing to various factors such as the surface property of an object and a structured light based depth sensor. This partial and noisy point cloud causes a major issue because the Efficient RANSAC frequently outputs inaccurate results, such as a plane being a part of a large cylinder, as illustrated in Figure 2.

To prevent this, we propose to add a credibility condition for a sphere and a cylinder:

$$\frac{\sqrt{N}}{R} \geq TH \tag{1}$$

where N is the number of points associated to a shape, R is a radius, and TH is a threshold empirically determined to control the result. This condition represents the fact that a point cloud must be the representative part of a shape, such that a large shape needs many points. This is useful to reject inaccurate estimation because only half of an object at most can be captured with a RGB-D camera.

V. Shape Matching

To match a shape extracted in the latest frame with the one in the shape map, we use different shape parameters according to object types as described in Table I. The same type of shapes with similar parameters can be simply matched by using a similarity threshold set for each parameter. For instance, as illustrated in Figure 3, two spheres are matched only if the ratio of their radii is within a certain

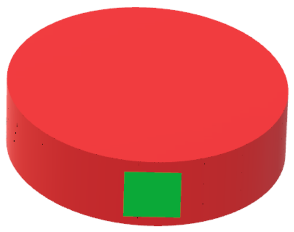


Fig. 2: Inaccurate estimation in Efficient RANSAC. Inaccurate estimation occurs with a partial object shape with noise such that a plane(green) is wrongly approximated by a large cylinder(red).

TABLE I: Shape parameters

Shape	Properties
Sphere	Position and radius
Cylinder	Axis position, orientation, and radius
Plane	Plane position and orientation

range; their distance is less than a certain value; and these matching conditions are subjects to defined parameters. In the following sections, the matching process for cylinders and planes, and the techniques for stable shape matching are explained.

A. Position distance for cylinders and planes

In the Efficient RANSAC, the position of a cylinder and a plane is defined as a point on a rotation axis or a plane. This is problematic because two positions can be distant even though they are on the same axis or plane. Therefore, their position comparison is obviously not meaningful using this approach. To solve this problem, we use an axis distance and a plane distance instead.

An axis distance between two cylinders is computed by

$$distance = \frac{\boldsymbol{D}_1 \times \boldsymbol{D}_2}{|\boldsymbol{D}_1 \times \boldsymbol{D}_2|} \cdot (\boldsymbol{P}_1 - \boldsymbol{P}_2) \quad (2)$$

where $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are the points on each axis, and $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are the directions of each axis, respectively. This distance is derived from a standard representation for computing the distance of skew lines.

A plane distance is meaningful only if they are parallel because they always intersect at some line in 3D space if they are not parallel. Therefore, we should assume that two planes are parallel to compute their distance. The distance is computed:

$$distance = \frac{|ax_1 + by_1 + cz_1 - (ax_0 + by_0 + cz_0)|}{\sqrt{a^2 + b^2 + c^2}} \quad (3)$$

where (a, b, c) is the average of the normals of both planes, (x_0, y_0, z_0) is a point on one plane, and (x_1, y_1, z_1) a point on the other plane.

B. Avoiding the influence of error accumulation in vSLAM

If a shape in the shape map was updated by fusing it with the shape extracted in the latest frame, the shape fusion with the latest frame will be influenced by only 10% if the

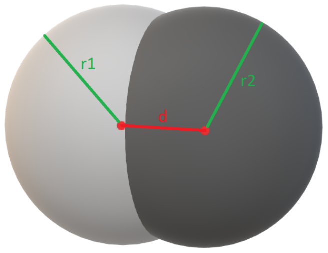


Fig. 3: Matching two spheres. Two spheres are matched only if the ratio of their radii and the distance between their centers are less than a threshold.

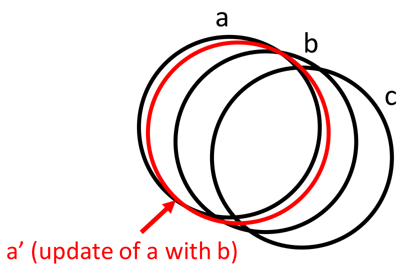


Fig. 4: Avoiding influence of error accumulation in vSLAM. In the shape matching, the shape in the latest frame(c) is matched with the one in the previous frame(b), not only with previously-updated one(a').

shape in the shape map is the result of 10 updates. Owing to error accumulation in vSLAM, a shape extracted in the latest frame can be far away from the one in the shape map. An example of this issue is illustrated in Figure 4. In this figure, the shape a in the shape map is being updated with the shape b. When the shape a has been updated many times before, the shape a is only slightly modified by the shape b as illustrated with the shape a'. As localization error accumulates in vSLAM, the shape c can be extracted further away from the shape a' and their distance becomes more than a threshold even though they should be matched.

The solution is to not only use the updated shape (a') in the previous frame to match with the shape (c) extracted in the latest frame, but also the shape (b) from the frame before the latest. To do this, the history of the shape matching, such as the affiliation between the updated shape and the lastly detected one, is stored. After detecting the shape at incoming frames again, it is matched with the shape that was last detected. This allows us to have to deal only with the frame-to-frame pose estimation error, and not with the accumulation of error since the beginning.

C. Adaptive thresholding for shape matching

In the shape matching, thresholds for parameters in Table I control whether a shape is matched or not. When a shape is approximated with few points, its error is generally more important, making it less likely to match properly. For example, as illustrated in Figure 5, an inner circle was extracted with a few points. Whereas the outer circle should

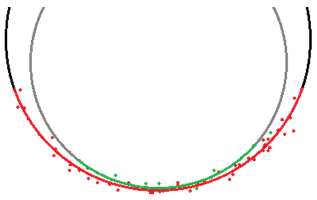


Fig. 5: Importance of adaptive thresholding for shape matching. The size of extracted circles varies according to the number of points used (red & green). Since estimation of a shape with a few points is not stable, it is important to adaptively set a threshold for shape matching.

ideally be extracted with these points if there is no error.

To solve this problem, a scale factor s based on linear regression is computed

$$s = -aN + b \tag{4}$$

where N is the number of points, a and b are positive coefficients. The lowest value of s is set as 1. In the thresholding, s is multiplied with each threshold to adaptively change it according to the shape size. As the error is relative to both the RTAB Map and the Efficient RANSAC outputs, it is difficult to theoretically derive the methodology to determine the coefficients. Instead, they are empirically optimized through experiments such as $a = -0.02$, and $b = 4$

VI. Shape update

The goal of the shape update is to determine the most probable primitive parameters based on all the information during vSLAM. The point cloud of a shape is updated and refined as well.

A. Updating shape parameters and point cloud

In shape parameter update, a weighting scheme is incorporated such as a weight corresponding to the number of times it has been updated. When a shape in the latest frame matches with the one in the shape map, the shape parameters are updated using

$$\boldsymbol{p_u} = \frac{w\boldsymbol{p_p} + \boldsymbol{p_l}}{w+1} \tag{5}$$

where w is the weight, $\boldsymbol{p_p}$, $\boldsymbol{p_l}$, $\boldsymbol{p_u}$ are parameters of a shape in the shape map, the parameters in the last frame and the updated parameters of the shape in the shape map, respectively. This gives an equal importance to the shape in the past frames that have been used for the update since the beginning.

Next, the point cloud of a shape is updated by fusing the one in the latest frame with the one in the shape map to generate a single point cloud. This process is required to estimate the length of a cylinder and the size of a plane because they are infinite in the parametric representation. Also, a hemisphere can be extracted from the arc length in the point cloud. As illustrated in Figure 6, the shape surface is first generated with the updated shape parameters. Then, point cloud fusion is simply done by projecting the two point clouds on the surface. As the capture continues, the amount of points in the point cloud accumulates. Therefore, the point cloud of the shape is downsampled after each point cloud fusion during a shape update process.

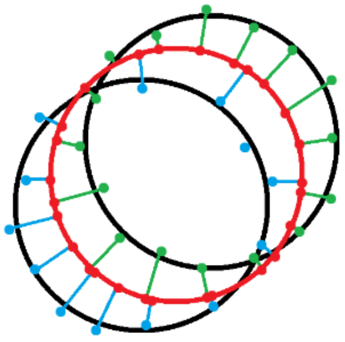


Fig. 6: Updating point cloud. Points in two matched shapes (blue & green) are projected on the updated shape (red).

In the next section, the detail of the point projection is described.

B. Point projection for point cloud fusion

A point projection method depends on object types. For planes, a target point $\boldsymbol{T}$ is projected onto a point $\boldsymbol{P}$ on the plane:

$$\boldsymbol{P} = \boldsymbol{T} - (\boldsymbol{V} \cdot \boldsymbol{N_p})\boldsymbol{N_p} \tag{6}$$

where $\boldsymbol{V}$ is the vector from an arbitrary point on the plane to the target point, $\boldsymbol{N_p}$ is the normalized normal vector of the plane, as illustrated in Figure 7a. For spheres, a target point $\boldsymbol{T}$ is projected onto a point $\boldsymbol{P}$ on the sphere:

$$\boldsymbol{P} = \boldsymbol{C} + R\frac{\boldsymbol{T} - \boldsymbol{C}}{|\boldsymbol{T} - \boldsymbol{C}|} \tag{7}$$

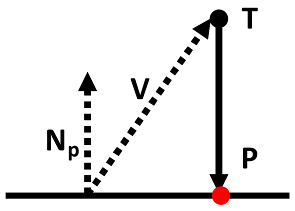
where $\boldsymbol{C}$ is the sphere center, and R is the radius, as illustrated in Figure 7b. For cylinders, the projection is done with the combination of the procedures for planes and spheres. First, the intersection $\boldsymbol{P'}$ between the rotation axis and the perpendicular vector from a target point $\boldsymbol{T}$ to the axis is computed:

$$\boldsymbol{P'} = \boldsymbol{X} + (\boldsymbol{D} \cdot \boldsymbol{V})\boldsymbol{D} \tag{8}$$

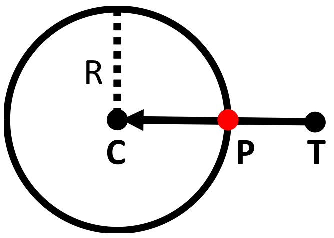
where $\boldsymbol{X}$ is an arbitrary point on the axis, $\boldsymbol{D}$ is the normalized axis vector, $\boldsymbol{V}$ is the vector from $\boldsymbol{X}$ to $\boldsymbol{T}$, as illustrated in Figure 7c. Then, $\boldsymbol{P}$ is computed with the same calculation with that for spheres, as illustrated in Figure 7b.

C. Removing unreliable shapes

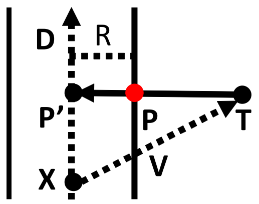
Given that Efficient RANSAC is based on random sampling, a shape that is a poor representation of the true shape can be detected. Such an erroneous shape would be rarely detected and have a lower weight than shapes that are detected many times and are a better representation. Consequently, we can eliminate such shapes by using a threshold for the weight such as removing shapes with a weight lower than average.



(a) Plane



(b) Sphere



(c) Cylinder

Fig. 7: Point projection. After updating shape parameters, point clouds are projected onto the surface of the updated shape.

As in [14], a bounding box of a shape is used to prevent some erroneous matching. For example, two planes at a distance can be matched because they are on the same plane, even though they are not related in 3D space. In this case, it is not enough to check shape parameters. Also, there is the case where a subset of the points is matched but the rest of the points are not. Comparing their bounding boxes simply solves these issues.

VII. Evaluation

To evaluate the performance of our method, we investigate the result of metric structural modeling using objects whose sizes are known. As a common dataset for structural modeling does not exist, we prepared our own dataset for the evaluation.

A. Setup

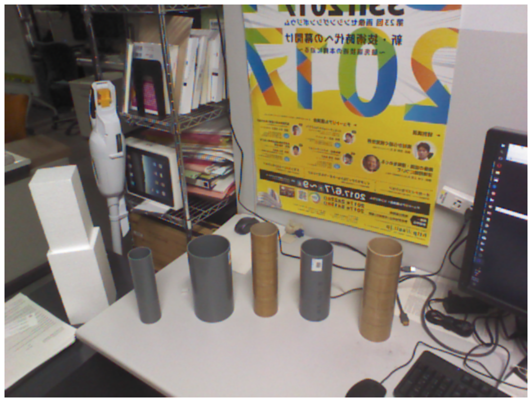
As illustrated in Figure 8, we prepared five cylinders, two hemispheres, and four spheres with different radii for the evaluation. The surface material of cylinders was polyvinyl chloride, and that of hemispheres and spheres was styrene foam. The cylinders are practically used for pipes in construction fields. As the radii of the objects were provided in their specification sheets by production companies, we used them as the ground truth.

For the data capture, we used a Kinect v1 sensor. We arbitrarily put objects on a desk and image sequences were captured with an arbitrary hand-held motion around the objects for a minute. We made the cylinder dataset containing cylinders only, and the sphere dataset containing hemispheres and spheres. For each dataset, we compared their estimated radii with their ground truth to quantitatively investigate the metric results.

B. Metric results

For the cylinder dataset, the modeling result is illustrated in Figure 9. and the accuracy of the result is described in Table II. In the table, the object number is determined from the left of the image. The accuracy was related to radius size such that the accuracy of smaller cylinders tended to be worse than larger ones. This is because fewer points can be captured from smaller objects.

For the sphere dataset, the modeling result is illustrated in Figure 10 and the accuracy of the result is described in Table III. Both the two hemispheres and the two spheres were accurately modeled. However, two spheres were not extracted owing to their small size. If the size of an object is small, the number of points in a point cloud is also low. With a Kinect v1 sensor, it was difficult to model any sphere which radius size was less than 30 mm in our experiment owing to the depth range of the sensor and the image resolution. Spheres tended to be more accurately modeled than cylinders


(a) Five cylinders practically used in construction fields

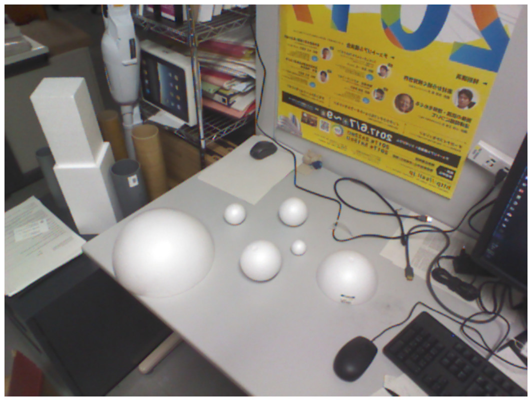

(b) Two hemispheres at both ends and four spheres between the hemispheres

Fig. 8: Evaluation setup. Objects with know sizes are prepared for the metric evaluation.

C. Examples of shape maps

Examples of shape maps are illustrated in Figure 11. We arbitrarily put different objects in the scene and generated the shape maps with our method. Given that these scenes

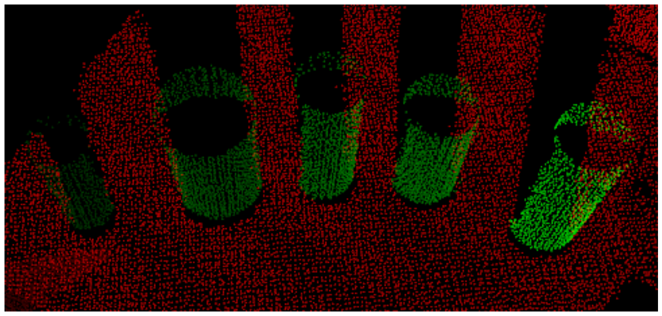


Fig. 9: Point clouds on cylinders. Cylinders(green) on the desk(red) were accurately modeled.

TABLE II: Results of measuring cylinders

Object	Ground Truth (mm)	Estimated (mm)	Error (mm)
1	27.0	31.5	4.5
2	53.0	54.3	1.3
3	32.5	37.1	4.6
4	40.0	41.0	1.0
5	39.0	40.0	1.0

comprised many planes including walls and cubes, most of the parts were modeled as planes. Cylinders and spheres in the scenes were correctly modeled.

During the incremental modeling, we measured computational costs with respect to the number of points in an incoming frame, as illustrated in Figure 12. The measurement was performed with Windows 10, Intel Xeon CPU E5-1620 v2 3.7 GHz, and 64GB RAM. The shape matching and update took less than 100-ms on average. The total cost was proportional to the number of points in a point cloud at a frame.

VIII. Conclusion

We proposed an incremental structural modeling using RGB-D SLAM. To generate a shape map containing primitive shapes, a primitive shape is incrementally modeled by fusing shapes estimated at previous frames. The history of the fusing process is used to avoid the influence of error accumulation in the SLAM. The point cloud of the shape is then updated by fusing the points in all the previous frames into a single point cloud. In the experimental results, we showed the performance of metric primitive modeling and its computational costs. In future, structural modeling using a voxel grid can be addressed, such that constraints of primitive shapes can be incorporated into surface extraction from the grid. Also, cube detection is important for structural modeling [22], and should be addressed.

Acknowledgment

A part of this work is supported by JSPS KAKENHI Grant Number JP17H01768.

References

[1] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

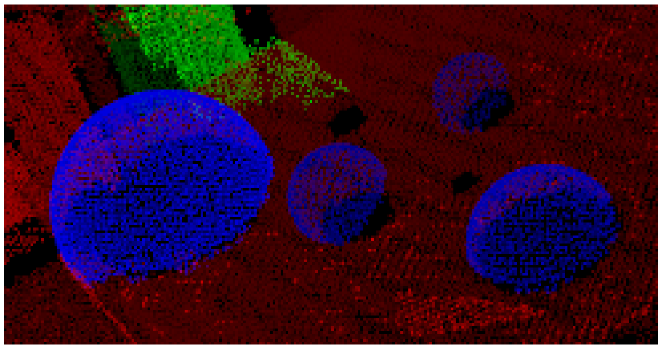


Fig. 10: Point clouds on hemispheres and spheres. Two hemispheres and two spheres(blue) on the desk(red) were modeled, but two small spheres were not because enough points were not extracted for the modeling.

TABLE III: Result of measuring hemispheres and spheres

Object	Ground Truth (mm)	Estimated (mm)	Error (mm)
Hemi1	125.0	121.9	3.1
Hemi2	75.0	73.3	1.7
Sphere1	30.0	-	-
Sphere2	50.0	49.9	0.1
Sphere3	40.0	41.0	1.0
Sphere4	20.0	-	-

[2] T. Taketomi, H. Uchiyama, and S. Ikeda, “Visual slam algorithms: a survey from 2010 to 2016,” *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, Jun 2017.
[3] M. W. Achtelik, S. Lynen, S. Weiss, L. Kneip, M. Chli, and R. Siegwart, “Visual-inertial slam for a small helicopter in large outdoor environments,” in *IROS*. IEEE, 2012, pp. 2651–2652.
[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “Monoslam: Real-time single camera slam,” *TPAMI*, vol. 29, no. 6, pp. 1052–1067, 2007.
[5] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *ISMAR*. IEEE, 2007, pp. 225–234.
[6] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *ECCV*. Springer, 2014, pp. 834–849.
[7] E. Eade and T. Drummond, “Edge landmarks in monocular slam.” in *BMVC*, 2006, pp. 7–16.
[8] W. Y. Jeong and K. M. Lee, “Visual slam with line and corner features,” in *IROS*. IEEE, 2006, pp. 2570–2575.
[9] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “Dtam: Dense tracking and mapping in real-time,” in *ICCV*. IEEE, 2011, pp. 2320–2327.
[10] S. A. Scherer and A. Zell, “Efficient onbard rgbd-slam for autonomous mavs,” in *IROS*. IEEE, 2013, pp. 1062–1068.
[11] C. Kerl, J. Sturm, and D. Cremers, “Dense visual slam for rgb-d cameras,” in *IROS*. IEEE, 2013, pp. 2100–2106.
[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinectfusion: Real-time dense surface mapping and tracking,” in *ISMAR*. IEEE, 2011, pp. 127–136.
[13] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic grasping of novel objects using vision,” *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.
[14] R. A. Roberto, H. Uchiyama, J. P. S. M. Lima, H. Nagahara, R.-i. Taniguchi, and V. Teichrieb, “Incremental structural modeling on sparse visual slam,” *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 5, Mar 2017.
[15] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. v. Gool, and W. Purgathofer, “A survey of urban reconstruction,” in *Computer graphics forum*, vol. 32, no. 6. Wiley Online Library, 2013, pp. 146–177.
[16] Y.-J. Liu, J.-B. Zhang, J.-C. Hou, J.-C. Ren, and W.-Q. Tang, “Cylinder detection in large-scale point cloud of pipeline plant,” *TVCG*, vol. 19, no. 10, pp. 1700–1707, 2013.

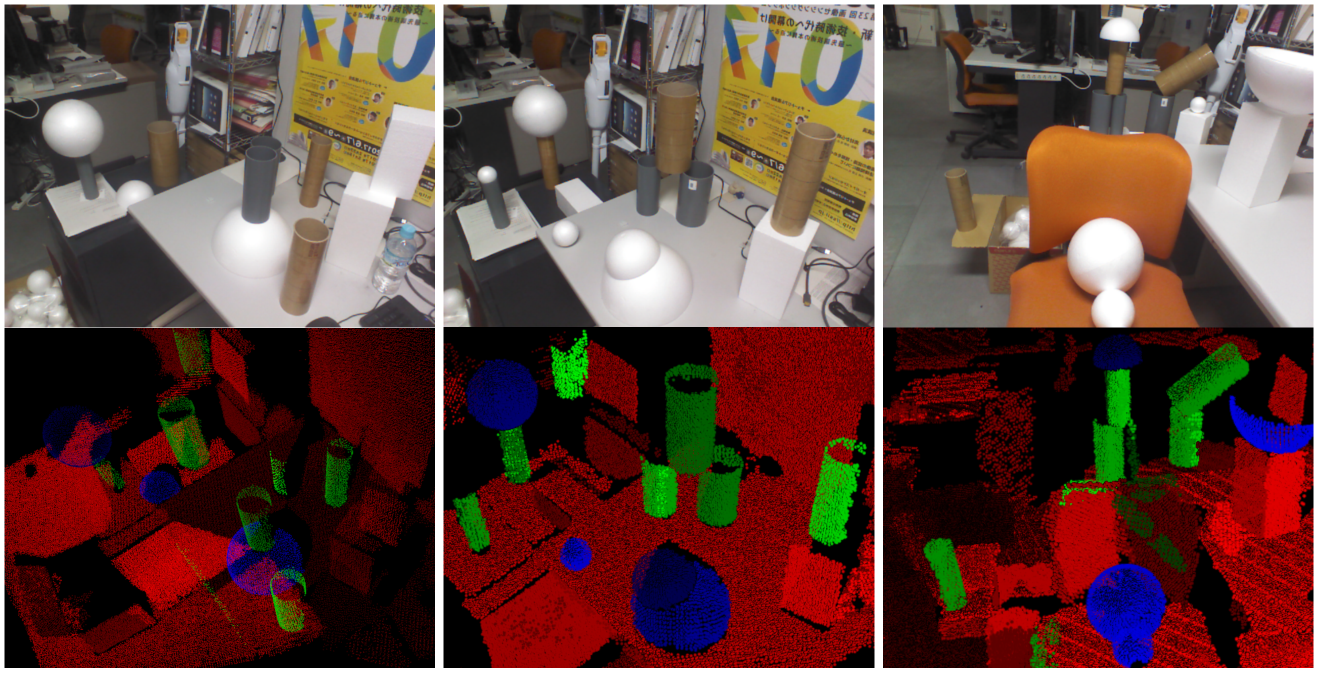


Fig. 11: Examples of a scene and its shape map. Planes(red), cylinders(green) and spheres(blue) are extracted.

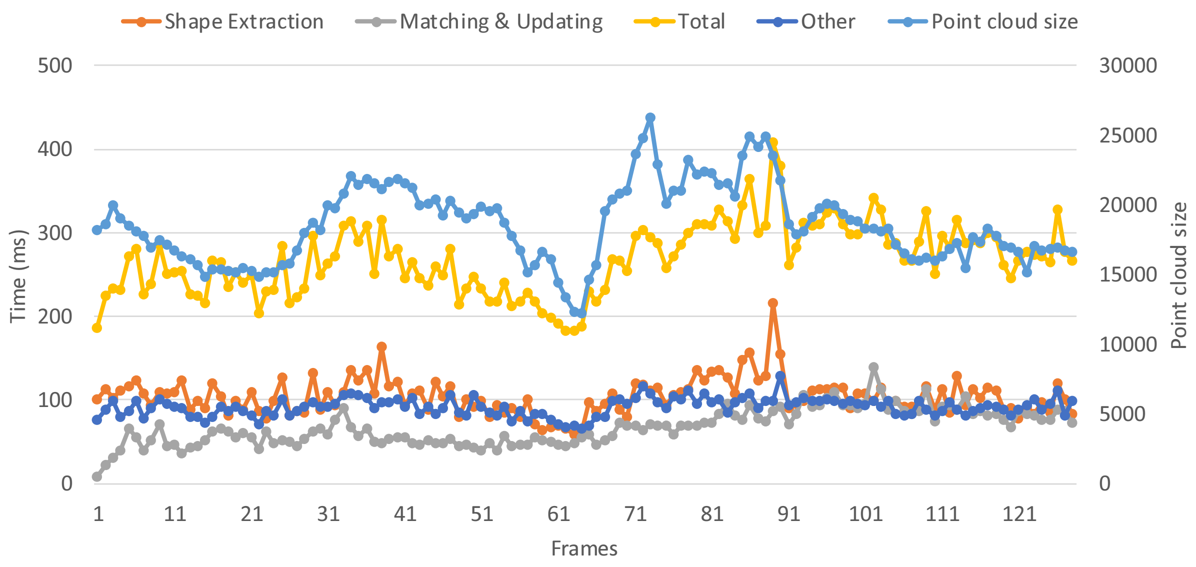


Fig. 12: Computational cost. Shape matching and updating need 100 ms in average. The cost is proportional to the number of points in a point cloud at a frame.

[17] R. Qiu, Q.-Y. Zhou, and U. Neumann, "Pipe-run extraction and reconstruction from point clouds," in *ECCV*. Springer, 2014, pp. 17–30.

[18] S. Ikehata, H. Yang, and Y. Furukawa, "Structured indoor modeling," in *ICCV*, 2015, pp. 1323–1331.

[19] D. Ramadasan, T. Chateau, and M. Chevaldonné, "Dcslam: A dynamically constrained real-time slam," in *ICIP*. IEEE, 2015, pp. 1130–1134.

[20] A. Concha and J. Civera, "Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence," in *IROS*. IEEE, 2015, pp. 5686–5693.

[21] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense planar slam," in *ISMAR*. IEEE, 2014, pp. 157–164.

[22] T. Nguyen, G. Reitmayr, and D. Schmalstieg, "Structural modeling from depth images," *TVCG*, vol. 21, no. 11, pp. 1230–1240, 2015.

[23] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *CVPR*, 2013, pp. 1352–1359.

[24] L. Yang, H. Uchiyama, J.-M. Normand, G. Moreau, H. Nagahara, and R.-i. Taniguchi, "Real-time surface of revolution reconstruction on dense slam," in *3DV*. IEEE, 2016, pp. 28–36.

[25] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *IROS*. IEEE, 2014, pp. 2661–2666.

[26] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, no. 2. Wiley Online Library, 2007, pp. 214–226.