Incremental Structural Modeling Based on Geometric and Statistical Analyses

Rafael Roberto[1], João Paulo Lima[1,2], Hideaki Uchiyama[3], Clemens Arth[4],
Veronica Teichrieb[1], Rin-ichiro Taniguchi[3], Dieter Schmalstieg[4]

[1]Voxar Labs, Centro de Informática, Universidade Federal de Pernambuco, Brazil
[2]Departamento de Informática e Estatística, Universidade Federal Rural de Pernambuco, Brazil
[3]Laboratory for Image and Media Understanding, Kyushu University, Japan
[4]Institute of Computer Graphics and Vision, Technische Universität Graz, Austria

rar3@cin.ufpe.br, jpsml@cin.ufpe.br, uchiyama@limu.ait.kyushu-u.ac.jp,
arth@icg.tugraz.at, vt@cin.ufpe.br, rin@kyudai.jp, dieter@icg.tugraz.at

Abstract

Finding high-level semantic information from a point cloud is a challenging task, and it can be used in various applications. For instance, it is useful to compactly represent the scene structure and efficiently understand the scene context. This task is even more challenging when using a hand-held monocular visual SLAM system that outputs a noisy sparse point cloud. In order to tackle this issue, we propose an incremental primitive modeling method using both geometric and statistical analyses for such point cloud. The main idea is to select only reliably-modeled shapes by analyzing the geometric relationship between the point cloud and the estimated shapes. Besides that, a statistical evaluation is incorporated to filter wrongly-detected primitives in a noisy point cloud. As a result of this processing, our approach largely improved precision when compared with state of the art methods. We also show the impact of segmenting and representing a scene using primitives instead of a point cloud.

1. Introduction

Automatic reconstruction of 3D object shapes is useful for several application, such as blueprint generation for architecture. Since it is a difficult task to accomplish, it has been a relevant research topic for years. Several methods have been proposed to achieve it by using laser scanners [21] and cameras [9]. Due to low-cost RGB-D sensors, such as Microsoft Kinect and Google Tango, 3D data acquisition became more common, and runs in real time even on mobile devices. Such sensors acquire the depth of an object on a pixel-by-pixel basis and, then, describe the obtained shapes as a point cloud. Although they are very useful for 3D measurement and visualization, there are some aspects to be improved. For instance, usually the scene is represented using a point cloud or a mesh computed from it. The latter simply consists of connected points with little information about the semantic structure. Also, both the point cloud and the mesh are frequently over-represented because a dense point cloud would be redundant to represent a 3D structure.

Inferring object semantics such as structural or object type information from a point cloud is an important research issue. This process is specifically referred to as semantic modeling, and it is useful in several applications for scene understanding and reverse engineering. The shape parameters of geometric primitives such as scale and pose are valuable knowledge to be estimated especially in human-made environments. Such semantics are useful for controlling picking robots [10], and they can also replace redundant point clouds with data structures, leading to savings in memory.

Most of the existing methods on semantic modeling estimate geometric primitives in a dense point cloud, which is commonly acquired using laser scanners offline [1]. Since the acquired point cloud contains more information available to rely on and it is relatively noiseless, a most points can stably fit primitive shapes. However, it is more challenging to extract primitive shapes in a noisy and sparse point cloud of a partially-observed object computed from image-based approaches with mobile devices. Some studies have tackled this issue by limiting specific situations, such as detecting only one shape class [11, 15]

These methods still require further generality and improvements of precision and accuracy for various applications. In particular, one uses these primitives as

constraints for visual SLAM systems [16], in which the error of incorrect primitive estimates can propagate and increase during tracking. In our preliminary experiments, a single mis-estimated shape had a negative influence on the whole tracking process.

In this paper, we propose a high-precision semantic modeling approach that leverages geometric and statistical analyses for a noisy and sparse point cloud. We build it on top of the general primitive estimation approach from Efficient Random Sample Consensus (RANSAC) [18] and the improvements from Incremental Structural Modeling (ISM) [17]. In summary, our method uses some useful information based on the geometry of estimated shape to compute a weight that indicates the reliability of the detected primitive. When this estimation is not reliable enough, we perform a statistical evaluation using the detection history and we eliminate those shapes with random detection. The main contributions of this paper are summarized as follows:

- A new approach considering the randomness and geometry of the estimated shape for ISM (Section 3);
- Evaluations of the proposed method in comparison with existing ones, showing that it improves semantic modeling precision (Section 4).

2. Related Work

Several methods have been proposed in the literature to detect shapes in point clouds. One conventional approach is to use reverse engineering techniques to estimate geometric primitives, such as region growing [3]. It can efficiently deal with large amounts of data because it makes simple comparisons using the normals to determine if a set of points belongs to the same group. However, this approach is not robust to noisy point clouds because it can lead to a wrong classification. The robustness has been improved using both Hough Transform [2] and RANSAC [8]. The advantage of RANSAC based techniques is that they can work even for a sparse point cloud because they only need a minimal set of points to estimate a primitive. Another approach is based on machine learning techniques, which combine local features and AdaBoost to detect complex objects [13]. Support Vector Machine (SVM), Fast Point Feature Histogram (FPFH) descriptors and RANSAC are also used to extract semantics from a point cloud [4].

Most of these semantic modeling methods were designed to use dense point clouds as input data. One example is [5], which uses a reverse engineering approach to estimate the floor plan of houses from a dense 3D point cloud generated using the Kinect sensor. Some methods detect only specific primitives, such as planes [11, 12] or cylinders [7, 15]. While other methods deal with different classes of shapes at once. For instance, GlobFit [6] can estimate planes, cylinders, cones and spheres. Some even detect pre-modeled complex shapes along with planes and cylinders in industrial scenarios [14].

Primitives can also be detected in scenes with a sparse point cloud. However, most methods were designed to detect planes only. One example is [20], which estimated planes based on its reconstruction to create textured models. The Incremental Structural Modeling (ISM) method estimated multiple types of primitives in a point cloud acquired using a visual SLAM system [17]. This solution is based on the Efficient RANSAC approach from [18] as a shape detector and uses the incremental construction of the SLAM map to detect primitives and refine the detection result. Although this method achieves good results for different types of primitives, it still needs to improve precision in challenging scenarios.

3. Geometric and Statistical Incremental Semantic Modeling

Similar to ISM [17], our method uses the generating process available in visual SLAM systems to evaluate the primitives estimated using Efficient RANSAC [18] incrementally. Figure 1 illustrates the flow of our method. The main difference with ISM is to perform two additional analyses to improve precision. First, we propose to use the geometric information of the detected primitive to measure its similarity to the input point cloud. This information is later used to assess the shape correctness. After that, we perform a statistical analysis through the detection history to identify and eliminate random primitives.

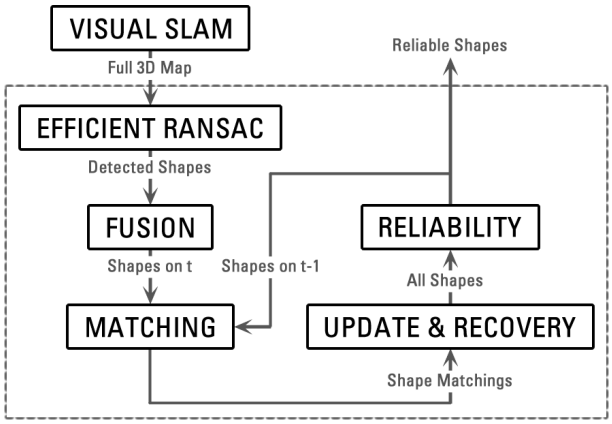

Figure 1. Flow of our Geometrical and Statistical Incremental Semantic Modeling (GS-ISM) approach.

When a visual SLAM system reaches a keyframe, it updates the map, which adds new points to the map. In our method, we use **Efficient RANSAC** to detect the shapes every time we have a new map. However, some of the detected shapes may be wrong and are not reliable due to a noisy input data or the small number of points in the

map. The remaining steps of our Geometric and Statistical Incremental Semantic Modeling, namely GS-ISM, are, thus, enforced to identify which of these detected shapes are reliable. We detail this process in the following subsections.

3.1. Shape Fusion

In keypoint based visual SLAM systems, most of the characteristics appear at highly textured areas. Thus, Efficient RANSAC may detect one primitive as multiple ones because these textured areas may be distant from each other. Therefore, it is important to combine all the shapes belonging to an object into one to improve their representativeness and reliability [17].

As in ISM, we use the similarity of the shape parameters to decide whether to fuse two shapes or not. To combine planes, it is necessary that their normals point to the same direction and the distance between them is smaller than a threshold based on the point cloud size. For spheres, the decision is based on the distance between their centers and the difference between their radii, which also should be smaller than a certain threshold. Finally, cylinders are fused if their axis point to the same direction and the distance between them, as well as the difference between their radii, are smaller than a value.

Additionally, we consider the proximity between the primitives to restrict or widen the similarity thresholds. The principle is that two distant shapes have a smaller possibility to be the same than closer ones. Experimentally, we widen the thresholds by 25% when evaluating the fusion primitives that have an intersection. Otherwise, we restrict them by 25%. This means that distant shapes have to be more alike to be merged. On the other hand, closer primitives are more likely to be the same and the threshold can be less restricted.

3.1.1 Parameter Computation

Similar shapes according to the aforementioned criteria are fused. We set the parameters of the resulting fused shape as a weighted average between the parameters of both primitives. This is more reliable than when using the parameters of the shape with more number of points, as in ISM. The weight is based on the geometric analysis of the points in the detected shape. The main idea is to use the average Euclidean distance of each input point that was used to estimate the primitive to the resulting shape. Even for noisy data, this distance will be smaller on correct estimations than on wrong ones. For instance, considering a globe being tracked that was correctly modeled as a sphere or incorrectly detected as a plane. The average distance of the 3D points in the globe to their projection in the sphere will be smaller when compared to the distance of the same input points to their projection in the wrong plane. Figure 2 illustrates this idea.

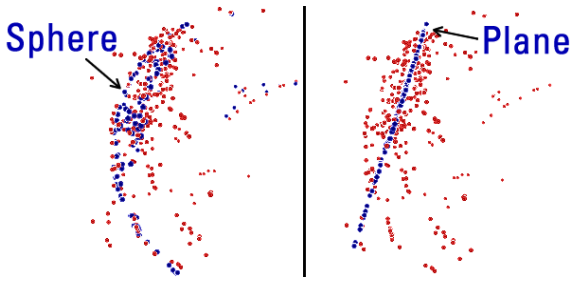

Figure 2. Difference between the input points to their correspondent projected points on a shape estimated correctly (left) and incorrectly (right).

Thus, the parameters P_f of the fused shape will be:

$$P_f = \sum_{i=1}^{n} w_i P_i, \tag{1}$$

where n is the number of similar shapes to be fused and P_i are their parameters. The weight w_i is:

$$w_i = \frac{np}{\sum_{j=1}^{np} d_j}, \tag{2}$$

where d_j is the Euclidean distance between the input points to its projection in the estimated shape and np is the number of points in the estimated primitive. The weights are normalized and $\sum_{i=1}^{n} w_i = 1$.

Using Equation 1, every fused shape will influence the resulting primitive. However, it will be closer to the one with the smaller error. This average can be applied to every parameter except the plane and the cylinder position. For the plane position, it is only valid if they are all projections on the other planes. Thus, we project the position of one point p_1 in the others, which in the case illustrated in the top row of Figure 3 will be p'_1. Then, we compute the weighted average between p_1 and p'_1 instead of p_2. This will also work in case of parallel planes. As for the cylinder position, this parameter will be the axis intersection. In case of concurrent or parallel axes, we fuse the cylinders in pairs if there is more than one. First, we get the points on each axis that is closer to the other and the resulting position will be their weight average.

3.1.2 Inclusion Criteria

In case a shape is not fused with any other, we make an initial reliability evaluation to decide if we keep this detected primitive or not. ISM eliminated shapes with a small number of points. We, on the other hand, use four geometric characteristics of the primitive to make this assessment, as described below:

- **Number of Points:** shapes with more points are usually more reliable because the estimation is based

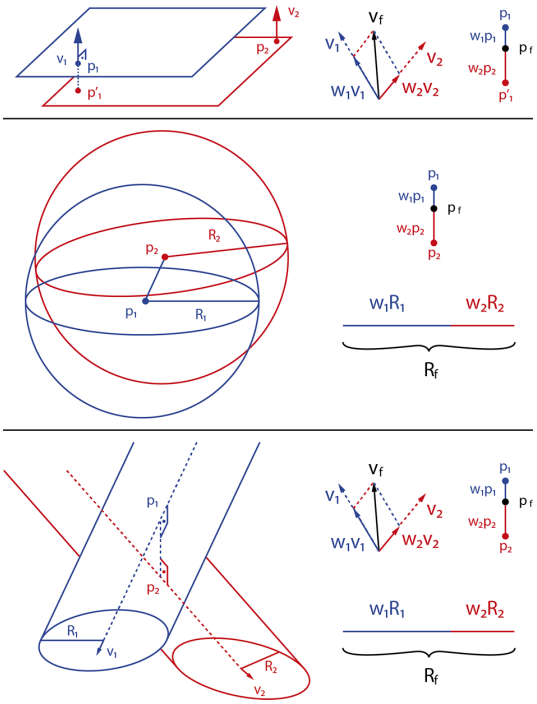

Figure 3. Fusion of parameters for different classes of shapes.

on a large amount of data. Good primitives are *larger than 2%* of the whole point cloud;

- **Dispersion:** it measures how spread are the points in the primitive. Since the keypoints are clustered around highly textured areas, small regions tend to concentrate most of the shape points. The dispersion of reliable shapes is *smaller than 20%* of this value for the whole point cloud;

- **Distance:** it is the same Euclidean distance mentioned previously. Reliable shapes have an average distance *smaller than 5%* of the largest size of the entire point cloud bounding box;

- **Radius:** the sphere and cylinder radius can also provide a hint regarding the shape's reliability. A noisy plane can be estimated as one of these two primitives with a considerable radius. Therefore, good sphere and cylinders have radius *smaller than* the largest size of the entire point cloud bounding box. It should be noted that this criterion is not applied to planes.

We only keep shapes that pass in all of these criteria. These values were determined experimentally and they are based on the dimension of the input point cloud because it puts all thresholds in proportional to the scene scale.

3.2. Shape Matching

As in ISM, we use the intersection of the 3D bounding box and the distance between the center of mass to match the detected shapes with those previously found. In summary, we compute a s_c score for each class of primitive that a given shape on current detection intersects on the previous estimation. This score is proportional to the intersection volume and inversely proportional to the distance between the center of mass:

$$s_c = \frac{\sum_i^{ns_c} |\psi_i| p_i}{\sum_i^{ns_c} |\psi_i| d_i} \frac{|\psi_s|}{\sum_i^{ns_c} |\psi_i|}, \tag{3}$$

where ns_c are the indexes of the shapes of the given class with intersection on previous keyframes, $|\psi_i|$ is the number of points of that shape, p_i and d_i are the intersection ratio and distance between centers of mass, respectively, and $|\psi_s|$ is the number of points in the primitive on the current keyframe. We select the one with the maximum score as correspondence.

3.3. Shape Update and Recovery

The shape detected on current frame inherits the history data of the one it matched on previous detections. These data contain the primitive class that was detected on every keyframe, as well as the average distance to the original point cloud at that detection. With that information, we can verify if the current estimation is following the historical data.

We check the class that this primitive was detected as over time. If the current shape has the same type of the one that appears in more than half of them, including the current detection, its parameter is updated. This new parameter will be the weighted average of each detection over time. With this update, every previous detection will influence the final shape, which results in primitives that are more stable through time than using just the parameters from the last one, as in ISM.

We use a similar process as explained in Section 3.1 to update the new parameter P_u:

$$P_u = \sum_{i=1}^{n-1} w_i P_{u-1} + w_n P_f, \tag{4}$$

where n is the number of shapes in the past, including current detection and w_i are their respective weights, which are normalized. P_{u-1} and P_f are, respectively, the parameters in previous detections and the current parameter after fusion.

On the other hand, if the current shape has a type that is different from the one that appears most of the time, it is changed to that class of primitive. As for the parameters, it will be the same as the previous P_u from that type.

In this step, we also evaluate shapes that were not detected on the current keyframe but appeared previously. We recover these shapes with the same parameters from the last appearance. Its history data will be updated using the

average distance of the recovered shape, but it will not have a primitive class associated at this particular moment. This shape will eventually disappear when not detected anymore because there will be no class of primitive that appear in more than 50% of the time.

3.4. Reliability Computation

At this point, we have all the detected shapes and their correspondent history information since their first appearance. In ISM, all primitives that appeared in more than 50% of the time were considered reliable. GS-ISM, on the other hand, compute the reliability based on a geometric and statistical evaluation.

3.4.1 Geometric Analysis

For each class of primitive that appears in the history data, we compute the weight w_c:

$$w_c = \frac{1}{\sum_{i=1}^{h} d_i}, \tag{5}$$

where h is the number of times that each class of primitive appears in the history data and d_i is the distance of the points in that shape to the original point cloud.

The weights are normalized and the one with the maximum value is the dominant class. We judge shapes whose dominant primitives have a weight higher than 0.75 as reliable. On the other hand, we consider unreliable those in which all weights are smaller than 0.5. When the weight of the dominant shape is between these two values, its classification will be determined by the statistical analysis. If this evaluation shows that the detection class through history is random, the primitive will be unreliable. Otherwise, it will be set as reliable.

3.4.2 Statistical Analysis

We perform a runs test for randomness to determine if the estimation history is random. Basically, this non-parametric test uses the expected value and standard deviation to estimate the minimum number of runs that a sample can have to be considered random [19]. A run means a sequence of consecutive estimates of one particular class of primitive. In our case, we have three types of shapes. However, we decided to look at our history of classification as binary data because the convergence is faster. Therefore, we denote a $+$ for the first primitive detected. Then, we repeat the sign if the shape class is the same as the previous one. Otherwise, we invert it . For a 5% level of significance, the sample is random if the number of runs is greater than:

$$N(R) = \mu - 1.65\sigma, \tag{6}$$

where the expected value μ and the standard deviation σ for the total number of samples n are:

$$\mu = \frac{2n-1}{3}, \tag{7}$$

$$\sigma = \sqrt{\frac{16n-29}{90}}. \tag{8}$$

Table 1 shows the example of a history information with a sequence of four spheres, followed by one cylinder, one plane and then by two other spheres. There are $R = 4$ runs and $n = 8$ samples. In this case, the minimum number of runs for a random sample is $N(R) = 3.269$, which indicates a random sample.

Table 1. History of the estimated shape from a primitive. For each sample that represents a keyframe K_i, it was classified as plane (P), sphere (S) or cylinder (C).

	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
Primitive	S	S	S	S	C	P	S	S
Label	$+$	$+$	$+$	$+$	$-$	$+$	$-$	$-$

4. Evaluation

We implemented our approach in C++ using OpenCV[1] and Efficient RANSAC[2] as libraries. We compared GS-ISM with Efficient RANSAC and ISM regarding precision, recall and $F_{0.5}$-Score. The choice of this metric instead of F_1-Score was because it highlights the precision, which is the focus of our method. We used a dataset[3] that has five scenes targeting distinct types of primitives and different numbers of keyframes, as seen in Table 2. Besides that, we modified Efficient RANSAC to use the same initial seed for random number generation. Therefore, it is not necessary to run the system several times during evaluation because it will always return the same result.

Table 2. Details of the dataset used for the evaluation, in which P, S and C stands for Plane, Sphere and Cylinder, respectively.

Test Case	Number of Frames	Number of Keyframes	Primitives in Scene
Case 1	1,660	31	P, S, C
Case 2	1,346	24	C
Case 3	849	20	S, C
Case 4	405	7	P
Case 5	499	17	P

It is possible to see in Figure 4 that our method obtained 100% precision in all cases while ISM achieved it only once. In turn, Efficient RANSAC never results in more

[1]Available at: `http://opencv.org/`
[2]Available at: `https://goo.gl/XINs6N`
[3]Available at: `https://goo.gl/c6mtBF`

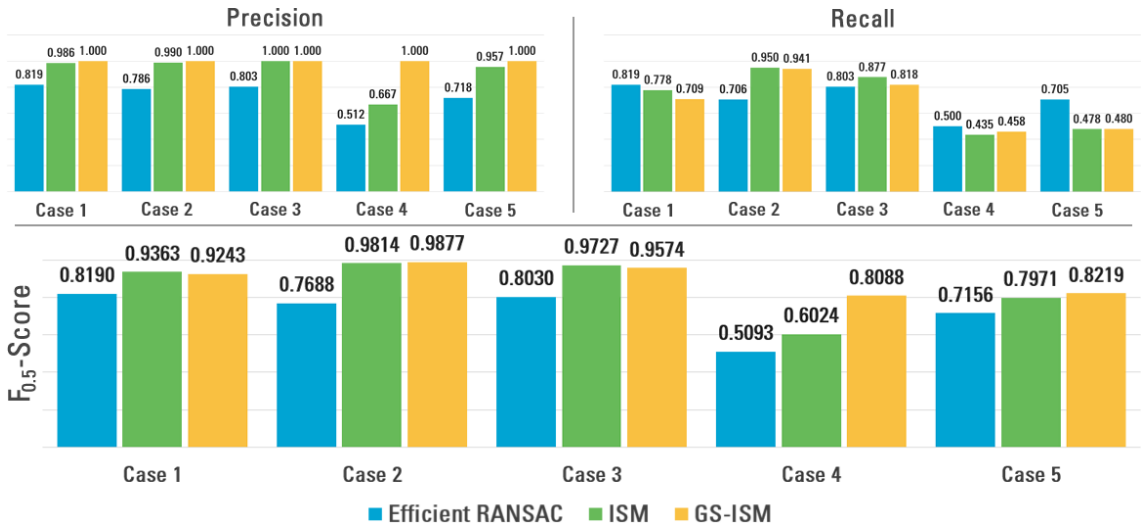

Figure 4. Comparison of precision, recall (top) and $F_{0.5}$-Score (bottom) between Efficient RANSAC [18], ISM [17] and our method.

than 82%. When comparing GS-ISM precision with ISM, we noted that in every case that ISM estimated a shape wrong, it happened in the initial keyframes. This is an expected behavior since the point cloud has few points in the initial reconstructions that the primitive detectors can rely on. The geometric and statistical analyses were able to identify these early incorrect detections. For instance, in the first three keyframes of Case 2, the bottle in the right side is assigned as a sphere, then as a cylinder and later as a sphere again because of the small number of noisy points. For the ISM, the bottle appears as a sphere in two-thirds of the time. Therefore, in the third keyframe, this bottle is incorrectly estimated as a sphere. In our case, the bottle is assigned to the same primitives in the first three keyframes but, each one has a weight based on the geometric analysis. After normalization, the weights of detection history are: 0.186 (sphere in the first keyframe), 0.537 (cylinder in the second keyframe) and 0.277 (sphere in the third keyframe). Thus, for GS-ISM, the bottle will be assigned as a cylinder because its weight is higher than the 0.463 of the sphere.

Regarding the recall, Figure 4 displays that our method is worse than the other two approaches. GS-ISM compromises recall in order to be entirely sure that the most reliable shapes are selected. Using the same bottle as an example, in the third keyframe the cylinder weight is 0.537, which is below the reliability threshold of 0.75. However, since it is above the 0.5 unreliability mark, we perform a statistical analysis to verify the randomness of this detection. According to Equation 6, this estimation history is random and the shape is assigned as unreliable.

Concerns to $F_{0.5}$-Score, we can see in Figure 4 that our method presented a better result in three of the five cases. The most significant improvement is in Case 4, which is very challenging because the reconstruction is very noisy. This evaluation indicates that the restriction imposed improved the precision, but with the cost of having few shapes detected. Therefore, it is possible to adjust the parameters to have more primitives and decrease the precision. These changes will depend on the target application. Table 3 provides some examples of possible modifications to make and the outcome for Case 1.

Table 3. The influence of modifications in GS-ISM on the final precision and recall in Case 1.

Condition Changed	Precision	Recall
Remove geometrical analysis	-1.409%	+1.846%
Remove statistical analysis	-1.409%	+3.139%
Double elimination thresholds	-0.704%	+2.602%

The first and third rows of Figure 5 present one keyframe from each test case and the estimated shapes using our method. They are represented by the projection of the input points used to compute the primitive. The second and fourth rows display one view of the input point cloud in red and some of the estimated shapes in blue. From the last row, it is possible to see how challenging is *Case 4*. Although the books are aligned in real life, the points from the left one are not aligned with the other two.

4.1. Runtime Evaluation

Concerning the computational cost, Efficient RANSAC takes on average 25.335 $\pm$ 9.597 ms to estimate the primitives in a computer with a Core i7-6820 (2.70 GHz) and 16GB of RAM. The other steps combined run in 13.528 $\pm$ 5.496 ms on average. ISM is 7.78% faster to perform the same steps. The bottleneck is the *shape fusion* step, which takes 12.699 $\pm$ 5.474 ms of that time. It is worth mentioning that the execution time is related to the number of input points. Therefore, the measurements, which are the

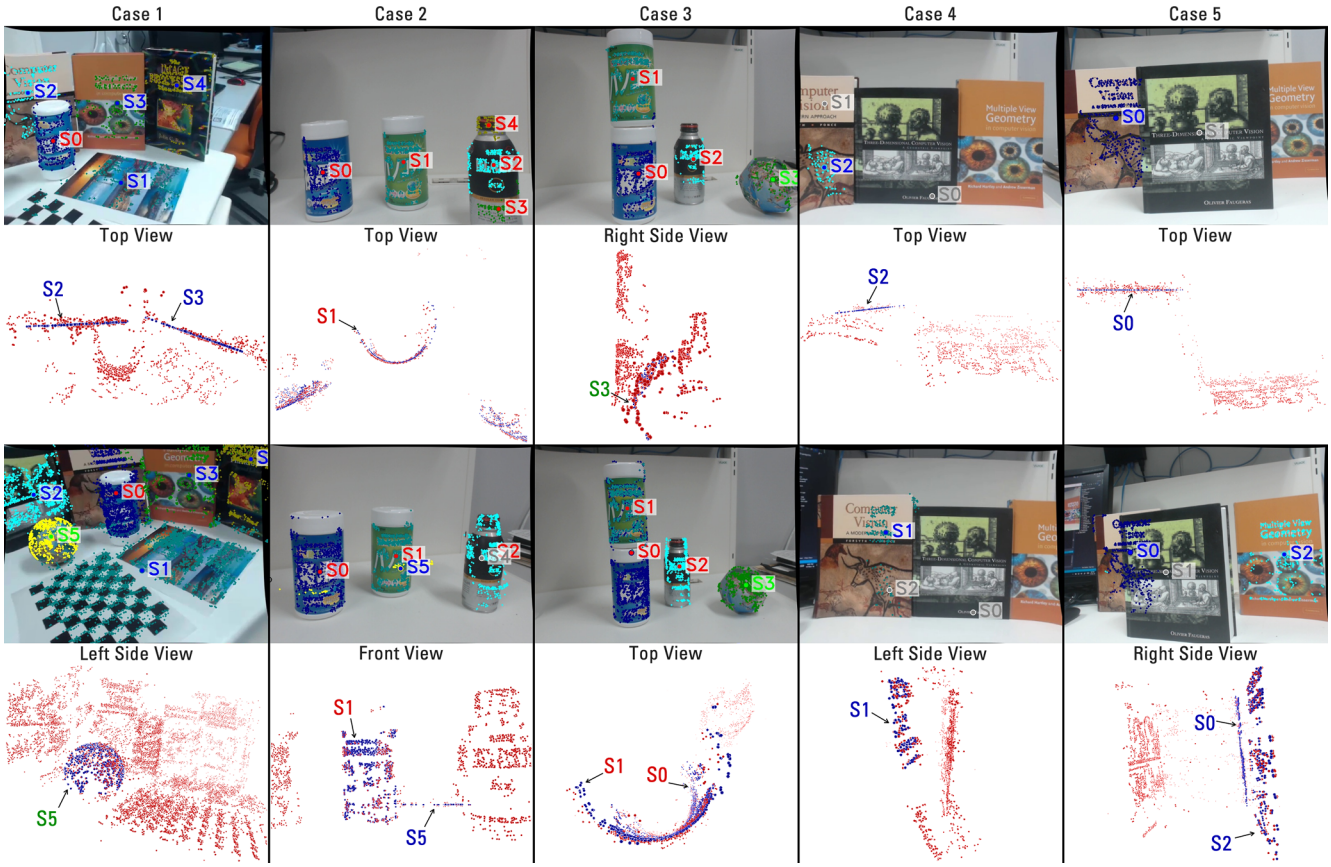

Figure 5. The first and third rows show the result of GS-ISM on each test case. Blue labels represent planes, green ones are for spheres and red for cylinders. The second and fourth rows show one particular view of the input point cloud (in red) and some of the estimated primitives (in blue).

averages of all five test cases, were normalized to a group of thousand points.

4.2. Segmentation Evaluation

We also evaluate how our approach segments the point cloud, which is a natural outcome of semantic modeling. Several objects have the form of the basic primitives we estimate with GS-ISM. Looking at an average from all five test cases, 70.85% of the points can be assigned to a plane, sphere or cylinder. Even though we are dealing with scenes designed with this type of primitives, the number is similar to a study that claims that 78% of all elements in an industrial scenario can be modeled using these three shapes [4]. Moreover, the chart on Figure 6 shows that only 6.30% of the remaining points were not labeled as any primitive. The other 22.85% points come from primitives that were discarded because they were unreliable.

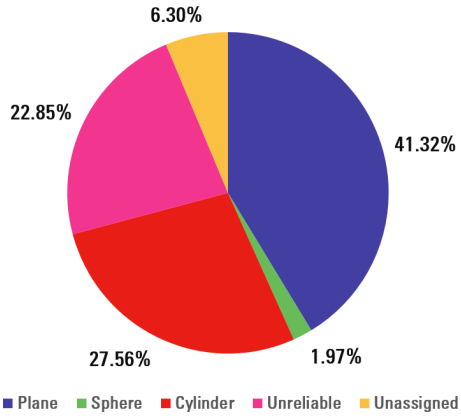

Figure 6. Percentage of points that were labeled to each primitive.

4.3. Point Cloud Representation Evaluation

Finally, we compared the scene representation using the point cloud and the modeled primitives. The scene is usually overrepresented when it is described using the points because there are many redundant points. We measure the memory necessary to represent the reconstruction of each test case using the point cloud and we compared it with the description of the same map using the data structure of the primitives modeled with GS-ISM. Figure 7 shows this difference in KB between then, ordered

by the number of points from the reconstructed map. The most significant difference is in the last keyframe of Case 1, which has 16,302 points. Using the point cloud requires 8.69 times more memory than describing the same map using the six detected primitives plus the 3,915 unreliable and unassigned points.

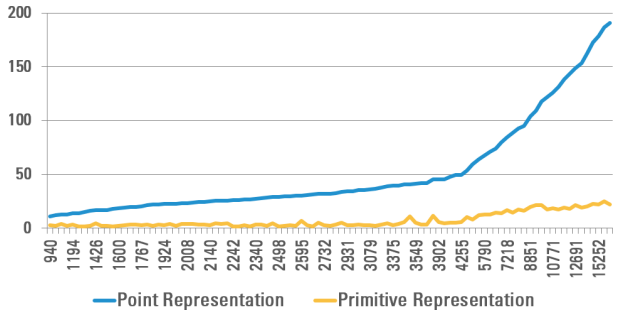

Figure 7. Memory (in KB) required to describe a scene using the point cloud and the data structure of the detected primitives.

Moreover, describing a scene using points commonly results in over and underrepresentation at the same time. For instance, if we consider only the cylinder detected in the last keyframe of Case 1, we can notice that there are more points than necessary to describe the textured front side but none to represent the back side. Thus, it is possible to use much less information to define this shape while filling the missing parts. Using this cylinder as an example, it is necessary 24 times more memory to describe it using the points than using the data structure of the detected primitive.

4.4. Proof of Concept

We intended to see how GS-ISM responds to an application that benefits from having semantic knowledge of the environment. We developed Shape Hunt, a system to help children to identify some of the primitives they are learning in school. The idea of this proof of concept is that it draws a shape and he/she has to find real objects with the same geometric form. Since we have the scene map, we can identify all selected shapes and the child always have to find a new one. Figure 8 shows this proof of concept.

We used the images from the dataset as input for this application. The scenes are limited to a small workspace, which was sufficient for this test. The application behaves as expected. Each primitive was detected only once and none was misclassified.

5. Conclusions

We presented a method that performs geometrical and statistical analyses to improve the incremental estimation of shapes in sparse point clouds. Our approach uses the geometric characteristic of the primitives to assess their reliability. This information is related to the distance between the input points used to estimate the shapes and their projection on the primitive. Depending on the geometric analysis, we also run a randomness test to keep the most reliable primitives.

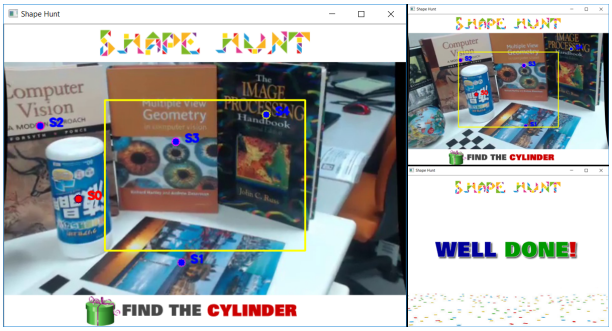

Figure 8. The application indicates the shape the user have to find (left image). When the correspondent shape is centered (top-right), it gives a positive feedback (bottom-right) and moves to the next primitive.

The experiment indicates that our method improved precision in all test cases, which outperforms existing methods in this criteria. Our method focuses on precision and for that, we are compromising recall to assure we have the correct shapes. However, we can modify the parameters to increase recall when necessary. The evaluation also suggests that our method can segment more than 70% of the point cloud from the test cases, which is compatible with the literature. Finally, we showed that representing a shape using primitives requires almost nine times less memory to describe the scene than using the point cloud.

As future works, we plan to integrate our system with a SLAM system such that we can evaluate it in more complex scenes. Besides that, we aim to port this semantic modeling method to mobile devices in order to measure the impact of using primitives on memory consumption and processing time. We also intend to use semantic knowledge of the scene to improve the tracking results of the visual SLAM system. The hypothesis is that by replacing the points in the original map by their projection on the estimated shapes we would denoise the point cloud, which can reduce errors. This can lead to a faster convergence of the bundle adjustment algorithm that can be very useful for mobile devices. In this case, precision is crucial because one wrong estimate can cause an error that would propagate to the entire tracking process.

Acknowledgments

The authors would like to thank CNPq (process 140898/2014-0), CAPES (process 88881.134246/2016-01) and the Austrian FFG under the Matahari project nr. 859208 for partially funding this research.

References

[1] J. Chen and B. Chen. Architectural modeling from sparsely scanned range data. *Int. J. Comput. Vision*, 78(2-3):223–236, July 2008.

[2] B. Drost and S. Ilic. Local hough transform for 3d primitive detection. In *2015 International Conference on 3D Vision*, pages 398–406, Oct 2015.

[3] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. *Real-Time Plane Segmentation Using RGB-D Cameras*, pages 306–317. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[4] J. Huang and S. You. Detecting objects in scene point cloud: A combinational approach. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 175–182, June 2013.

[5] Y. M. Kim, J. Dolson, M. Sokolsky, V. Koltun, and S. Thrun. Interactive acquisition of residential floor plans. In *2012 IEEE International Conference on Robotics and Automation*, pages 3055–3062, May 2012.

[6] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4):52:1–52:12, July 2011.

[7] 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. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1700–1707, Oct 2013.

[8] D. Lopez-Escogido and L. G. de la Fraga. Automatic extraction of geometric models from 3d point cloud datasets. In *2014 11th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pages 1–5, Sept 2014.

[9] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Interdisciplinary Applied Mathematics. Springer New York, 2005.

[10] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE, 2003.

[11] T. Nguyen, G. Reitmayr, and D. Schmalstieg. Structural modeling from depth images. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1230–1240, Nov 2015.

[12] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke. *Efficient Multi-resolution Plane Segmentation of 3D Point Clouds*, pages 145–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[13] G. Pang and U. Neumann. Training-based object recognition in cluttered 3d point clouds. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 87–94, June 2013.

[14] G. Pang, R. Qiu, J. Huang, S. You, and U. Neumann. Automatic 3d industrial point cloud modeling and recognition. In *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, pages 22–25, May 2015.

[15] R. Qiu, Q.-Y. Zhou, and U. Neumann. *Pipe-Run Extraction and Reconstruction from Point Clouds*, pages 17–30. Springer International Publishing, Cham, 2014.

[16] D. Ramadasan, T. Chateau, and M. Chevaldonn. Dcslam: A dynamically constrained real-time slam. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1130–1134, Sept 2015.

[17] 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*, 9(1):5, Mar 2017.

[18] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

[19] D. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures, Fifth Edition*. Taylor & Francis, 2011.

[20] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3D architectural modeling from unordered photo collections. *ACM Trans. Graph.*, 27(5):159:1–159:10, Dec. 2008.

[21] L. Zhu, J. Hyypp, A. Kukko, H. Kaartinen, and R. Chen. Photorealistic building reconstruction from mobile laser scanning data. *Remote Sensing*, 3(7):1406–1426, 2011.