Autonomous Generation, Segmentation, and Categorization of Point Clouds

David Skiff, Computer Science in the College of Engineering, 2012, des259@cornell.edu

***Abstract*— Microsoft Kinect dynamically provides texture mapped 3-dimensional point clouds. Combining this with new Simultaneous Localization and Mapping techniques, fast generation of detailed, fully textured, 3-dimensional point clouds of an environment, and the Kinect's location in it, can be found on the fly. One application of the this new ability is to the segmentation and categorization of an environment, and the generation of a simple floor plan. The projects goal is to generate a simple, reasonably fast set of tools that segments simple Point Clouds into seperate objects, assigns them basic categories (movable and unmovable), and generates a simple blueprint for the environment. The results can be used for navigation, or tasks such as moving around and sorting objects in a room. Another possible use is the detection of doorways. In the future, more work on the segmentation algorithms would be desired.**

I. INTRODUCTION

The recent introduction of Microsoft's Kinect has created a significant amount of research into non-uniform 3-dimensional point cloud manipulation and analysis. This project applies several known robotics methods for both 2-dimensional and 3-dimensional problems to the task of Simultaneous Localization and Mapping (SLAM), Mesh Segmentation, and 3-Dimensional Object Classification.

In robotics, its often valuable to be able to dynamically discover an environment, produce a map of the environment, and locate objects with-in it. Navigation and Control are two of the most important aspects of robotics. Having a map of the environment, the robots location in the environment, and the location and class of the objects with-in it, provide the robot with invaluable information for both of the tasks. Indeed, the first localization and mapping are the focus of SLAM, an important and heavily researched field of robotics.

As a result, this project seeks to generate a 2-dimensional blueprint of a room dynamically. Ideal it would contain immovable objects, as well as walls, and doorways. This differs Occupancy grid method used by most SLAM projects. The project seeks to map walls, and specific points where it cannot walk, but allows for moving of objects.

II. RELATED WORK

There's a lot of prior work in SLAM. The process of performing SLAM is usually broken up into five separate problems: State Estimation, State Update, Landmark Extraction, Landmark Update, and Data Association. There are several solutions already available, most notably ones based on extended Kalman Filters [1] (or EKF-SLAM) and based on particle filters [1] (or fastSLAM). Both methods provide a means for estimating the location of the robot based on transformations in the landmarks after movement. The EKF solution is by far the most popular. However, with the added information provided by the Kinect, we can locate landmarks in a 3-dimensional space accurately, and then use RANSAC to estimate transformations. This allows us to skip the use of EKF, since mapping can be effectively performed without knowing the location or transformation ahead of time. This is used by RGBD-SLAM [2], and has provided impressive results with the Kinect. However, the method is in its early stages, and no paper is available on it, yet. SLAM methods often are used to generate Occupancy Maps, gradient maps indicating the perceived cost of traveling over a region. [9]

III. APPROACH

The project is entirely implemented in the Robot Operating System (ROS), which can be found at *www.ros.org*. The project is broken up into seperate steps, and the results from each are piped between the them (see Fig. 2).

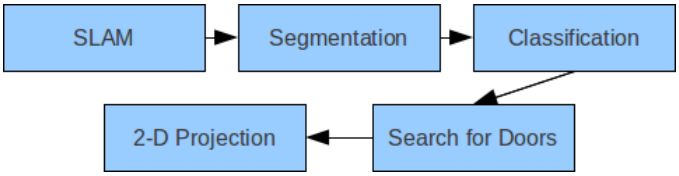

Fig. 1. Data Path for Project

A. RGBD-SLAM

SLAM is usually difficult do to the classic chick or the egg problem, where you need to the position and transformation to map, and you need the map in order to locate yourself. However, by using the method from [2], the mapping is performed by estimating the transformation using RANdom SAmple Consensus (RANSAC) [4]. This removes the mutual requirement, and significantly simplifies the problem. The solution from [2] involves the use of Speeded Up Rubust Features (SURF) [3] to generate landmarks which are then projected into 3-dimensional space. For each frame received from the kinect, the transformation from the previous states is estimated using RANSAC. This allows the creation of a graph connecting camera viewpoints between frames. Location is easy to derive, since we have depth information from the camera. [2] Each point cloud is combined into a cumulative Point Cloud.

B. Segmentation

In order to achieve all of the goals (identify movable objects, identify doors, and create a blueprint) we segment the

object into walls and seperate objects. There's a significant amount of noise and excessive clustering of pixels, so a Gaussian Filter is used to remove noise, and a Voxel Grid is used to down sample the data. While most research for segmentation has been done for 2-dimensions, there is some prior work for 3-dimensional, or mesh, segmentation. However, most of it requires uniform, non-noisy, Point Clouds. As a result, we use Planar and k-Means Clustering Segmentation [6]. We assume the floor is a large, flat plan, and that the walls are large planes perpendicular to the floor. Thus, we fit planes to the points at the specific orientations using PCL's provided parallel plane consensus segmentation which uses RANSAC [4]. Once the floors and walls have been removed, k-Means Clustering is used to segment the objects with Euclidean Distance is used as the distance function. The result is a set of expected wall, floor, and object Point Clouds.

C. Categorization

The walls and floors are already categorized for us, so what remains is to classify the objects as either movable or immovable. While a number of methods could potentially be used for this, this project decided to use a Bag of Keypoints approach [7]. First, a suitable feature extractor must be chosen. We need something that's scale, position, and rotation invariant, such that we can merely compare features any point in an image and calculate distance. While there are quite a few methods available for 2-dimensional image, 3-dimensional feature extractors are more scarse, NARF was used by this project. Normal Aligned Radial Feature NARF (NARF) [8] is a rotation invariant feature extractor. NARF first extracts interest points, and then generates a feature descriptor at that point. It does so by:

- calculating a normal aligned range value patch in the point, which is a small range image with the observer looking at the point along the normal,
- overlay a star pattern onto this patch, where each beam corresponds to a value in the nal descriptor, that captures how much the pixels under the beam change,
- extract a unique orientation from the descriptor,
- and shift the descriptor according to this value to make it invariant to the rotation.

(from [8]). Now that we have a feature extractor, we simply create test sets and extract features from them. Identify the most

D. Map

We know have a collection of wall point clouds, a collection of object point clouds, and a floor and roof point cloud. In order to generate the blueprint, we project all walls and immovable objects down to 2-dimensions, and then use linear regression to fit lines to the clouds. Then, collision points between the lines are taken as corners. The resulting map is output as a collection of endpoints for the walls, and the convex hulls of the projection of each individual object cloud.

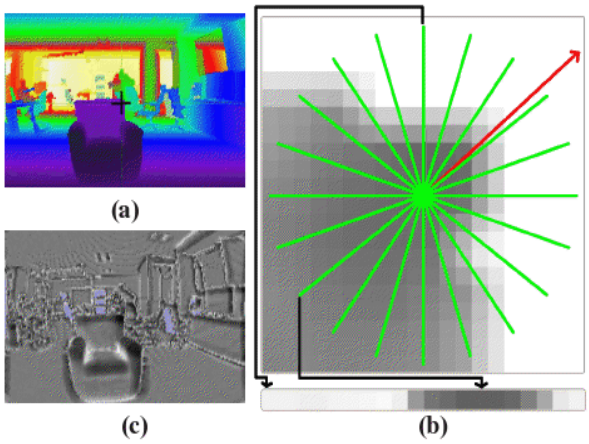

Fig. 2. From [8]. (a) A range image of an example scene with an armchair in the front. The black cross marks the position of an interest point. (b): Visualization how the descriptor is calculated. The top shows a range value patch of the top right corner of the armchair. The actual descriptor is visualized on the bottom. Each of the 20 cells of the descriptor corresponds to one of the beams (green) visualized in the patch, with two of the correspondences marked with arrows. The additional (red) arrow pointing to the top right shows the extracted dominant orientation. (c): The descriptor distances to every other point in the scene (the brighter the higher the value). Note that mainly top right rectangular corners get low values.

IV. Results

RGBD-SLAM is produces incredible results. However, it's incredibly buggy, and combined with rviz's crash susceptability, long runs were somewhat restricted with on the fly viewing (or even without). Regardless, full point clouds of several rooms were generated successfully, and basic blueprints were successfully generated. However, object classification failed to perform anywhere near accurately. This is likely due to too small of training set being generated. Excluding the objects, the blueprints themselves were qualitatively good.

V. Conclusions and Future Works

A. Conclusions

RGBD-SLAM provides an excellent, albeit noisy and non-uniform, Point Cloud, which can be suitably filtered and segmented to generate basic blueprints of a room. While object classification isn't working correctly, yet. The generation of a larger training set would likely elleviate this issue.

B. Future Works

Generate a larger training set for the object classification. At the time of writing this, the object classification fails to perform well, and is likely due to limited training data.

The current method of object segmentation, k-Means, allows for simple non-contiguous segmentation. However, if objects touch, or overlap, then the k-Means identifies the the seperate elements as a single object. There are alternative Mesh Segmentation algorithms available, like that suggested by [5], but are significantly more arduous to implement.

It was an initial goal of this project to identify doorways from the resulting blueprint. However, time was too much of a constraint. However, since we'd only be looking at the wall blueprint, We can search around the endpoints of lines, identify gaps, and match them against the size restrictions for a door.

REFERENCES

[1] Durrant-Whyte, H., and T. Bailey. "Simultaneous Localization and Mapping: Part I." *IEEE Robotics & Automation Magazine* 13.2 (2006): 99-110. Print.

[2] Ruchti, Philipp, Felix Endres, Juergen Hess, Nikolas Engelhard, Juergen Sturm, Wolfram Burgard, and Daniel Kuhner. "RGBD-6D-SLAM." *ROS Wiki*. Web. 03 Apr. 2011. ¡http://www.ros.org/wiki/openni/ Contests/ROS 3D/RGBD-6D-SLAM¿

[3] Bay, H., A. Ess, T. Tuytelaars, and L. Vangool. "Speeded-Up Robust Features (SURF)." Computer Vision and Image Understanding 110.3 (2008): 346-59. Print.

[4] Bolles, Robert C., and Martin A. Fischler. "Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." *Communications of the ACM* 24.6 (1981). Print.

[5] Jagannathan, Anupama. "Segmentation and Recognition of 3D Point Clouds within Graph-theoretic and Thermodynamic Frameworks." Thesis. Northeastern University, 2005. Print.

[6] Lloyd, S. "Least Squares Quantization in PCM." *IEEE Transactions on Information Theory* 28.2 (1982): 129-37. Print.

[7] Csurka, Gabriella, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual Categorization with Bags of Keypoints. Xerox Research Centre Europe. Print.

[8] Bastion Steder,Radu Bogdan Rusu, Kurt Konolige, Wolfram Burgard "NARF: 3D Range Image Features for Object Recognition" (2010)

[9] Mark Whitty, Stephen Cossell, Kim Son Dang, Jose Guivant and Jayantha Katupitiya "Autonomous Navigation using a Real-Time 3D Point Cloud"