Simultaneous Localization and Mapping

Kenny Kang

Abstract—The experiments documented in this paper aimed to implement Simultaneous Localization and Mapping(SLAM) algorithms in a simulated environment. In addition to functionality, the robustness of the algorithm was also tested on a variety of gazebo worlds, analyzing the shortcomings of performing SLAM in a simulated setting. The Real Time Appearance Based Mapping(RTAB-Map) algorithm was the SLAM technique of choice due to its utilization of visual sensors and time efficiency.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Localization.

1 Introduction

THe rise of autonomous systems such as self driving cars has been integrating robots into everyday life. However, this evolution comes new problems that must be solved, one being the issue of mapping and localizing in an unknown environment. Individually, mapping and localization depend on being given data in advance, a known map for localization and a known pose for mapping. Unfortunately when a robot is placed in the real world, environments are dynamic and constantly change, making localization difficult. In addition, an exact pose cannot be given for an environment that has not yet been defined. This is the dilemma. One algorithm depends on the output of the other, but if neither produce accurate outputs, error builds off one another. SLAM algorithms are a proposed solution to this issue.

2 Background

The problem space of SLAM covers many variables. As a result, there are different characteristics which lead of various directions of potential solutions. SLAM algorithms come in two forms: Online and Full SLAM. Online SLAM algorithms estimate the current pose using current measurements while Full SLAM estimates trajectories using a history of measurements. In addition, the very nature of SLAM problems deal with multiple types of data which add complexity and additional challenges

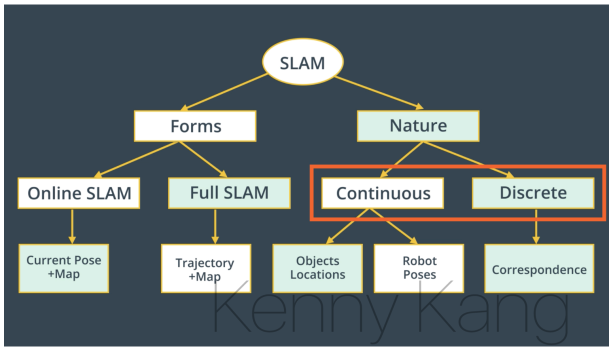

Fig. 1. Characteristics of SLAM Problems

2.1 Challenges

The main challenges arise from the problem's continuous and discrete nature. The robot's potential poses and object locations within an environment make up the continuous aspect. As the algorithm progresses, the number of objects that the robot must keep track of increases which lead to a high dimensional parameter space. The discrete nature of the problem is composed of the correspondence values for each image frame that is recorded. These correspondence values refer to a relationship between two images and different time steps. As the number of observed frames grow, the number of comparisons that need to be made grow exponentially.

2.2 Algorithms

SLAM algorithms can fall into five categories:

1) Extended Kalman Filter SLAM
2) Sparse Extended Information Filter (SEIF)
3) Extended Information Form (EIF)
4) FastSLAM
5) GraphSLAM

2.2.1 *FastSLAM*

FastSLAM is a natural extension of existing localization and mapping algorithms. It utilizes trajectory estimation from Monte Carlo Localization with a custom particle filter. Unlike MCL, the complexity of keeping track of multiple dependent landmarks in SLAM make in infeasible. Fortunately, these landmarks can be treated as independent variables assuming poses are known. This makes it possible to find the joint probability distribution of the state space with particles.

$$p(x_{0:t} m_{1:M} | z_{1:t} u_{1:t}) = p(x_{0:t} | z_{1:t} u_{1:t}) p(m_{1:M} | x_{0:t} z_{1:t})$$

where x is the robot pose, m is the map, z is the measurement, and u is the inputted control. In order to implement map estimation with known particles, either an EKF or Occupancy Grid Mapping algorithm are used.

2.3 GraphSLAM

GraphSLAM is an algorithm which solves the full SLAM problem, outputting the entire robot path in addition to its environment rather than just the next pose. The algorithm consists of two parts: graph building and graph optimization. The graph itself is made of up nodes which either represent motion constraints, which are created after a robot's movements, or measurement constraints, which are recorded from features sensed by robot. Once the graph is updated with each time-step, it is optimized to minimize the error found throughout the map.

The GraphSLAM method used in this paper is the Real Time Appearance Based Mapping(RTAB-MAP) algorithm. This method tracks features using visual bag of words and searches for loop closures. These loop closures and the odometry measurements are the constraints that are ultimately being tracked.

2.4 Comparison / Contrast

The issue with FastSLAM is its precision. When the robot's pose is purely based on sampling, the odds of the pose being exact are very unlikely. In contrast, GraphSLAM as it considers the dependencies of each state with the rest of the path history. However one disadvantage of GraphSLAM is its heavy computational needs as it is constantly making comparisons to an ever growing database in comparison to the uniform resources a particle filter implementation needs.

3 Scene and Robot Configuration

3.1 Kitchen and Dining

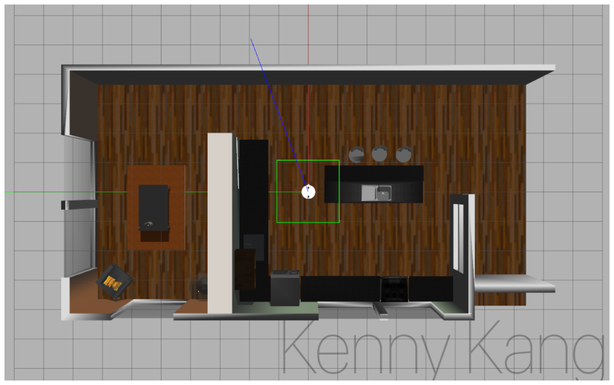

Fig. 2. Top-down view of the Kitchen and Dining world

3.1.1 Robot

The robot itself consists of two 2D LiDAR sensors, which enables a 360 view of obstacles, and an RGB-D camera, to capture depth info for 3D mapping and RGB images for feature detection.

TABLE 1
Kitchen and Dining Robot Parameters

Component	Geometry	Size
Chassis	Box	0.4 x 0.2 x 0.15
	Link Origin	[0,0,0.05,0,0,0]
Chassis Head	Cylinder	0.23(radius), 0.1(length)
	Link Origin	[0,0,0.15,0,0,0]
Back and Front Caster	Sphere	0.0499(radius)
Left and Right Wheels	Cylinder	0.1(radius) x 0.05(length)
Kinect (RGB-D)	Link Origin	[0,0,0,0,0,0]
	Shape Size	0.4 x 0.2 x 0.1 (Box)
	Joint Origin	[0.23,0,0.15,0,0,0]
	Parent Link	Chassis
	Child Link	Kinect
Hokuyo (2D LiDAR)	Link Origin	[0,0,0,0,0,0]
	Shape Size	0.1 x 0.1 x 0.1 (Box)
	Joint Origin	[0.15,0,0.25,0,0,0]
	Parent Link	Chassis
	Child Link	Hokuyo
Hokuyo2 (2D LiDAR)	Link Origin	[0,0,0,0,0,0]
	Shape Size	0.1 x 0.1 x 0.1 (Box)
	Joint Origin	[0.15,0,0.25,0,0,0]
	Parent Link	Chassis
	Child Link	Hokuyo2

3.1.2 RTAB-Map

The parameters for the RTAB-Map package did not require any changes from the default values for using the Speeded-up Robust Features(SURF) detector. The choice of using surf as opposed to another feature detector such as Scale Invariant Feature Transform (SIFT) was due to its speed advantage. While SURF is not as robust as SIFT, speed is more of a priority in real-time mobile robots when computational resources are limited.

TABLE 2
Kitchen and Dining RTAB-Map Parameters

Parameter	Value	Description
Kp/DetectorStrategy	0	Speeded-up Robust Features (SURF)
Kp/MaxFeatures	400	Max visual words per image (bag-of-words)
SURF/Hessian Threshold	100	Sensitivity to features detection
SURF/Octaves	4	Size of detected features
SURF/OctaveLayers	2	Determines scale invariance of features
SURF/GpuVersion	false	Determines utilization of GPU
Reg/Strategy	0	Performs loop closures with a visual transform(RANSAC)
Vis/MinInliners	15	Determines utilization of GPU

3.2 Dirty Cafe

3.2.1 World

When considering the features of a newly created environment, a primary goal was to model a full world rather than simply scattering objects in a blank one. A feature-rich environment encourages a more robust algorithm which is the ultimate goal of this experiment. This influenced the choice of using the pre-made cafe environment as the foundation of the custom environment. In early experiments, repetitive patterns seemed to become an issue, causing incorrect loop closures, so additional props were added in order to create a more distinctive environment with additional features.

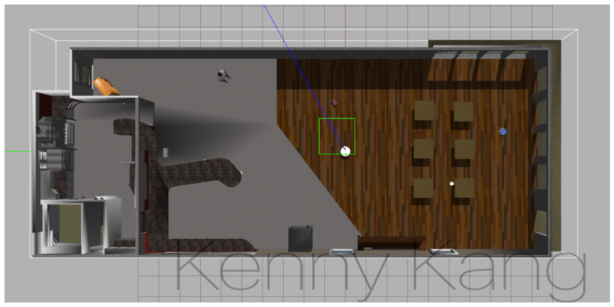

Fig. 3. Top-down view of the Dirty Cafe world

3.2.2 *Robot*

The only adjustment made from the previous robot was raising the pitch of the camera joint on the robot by 15 degrees. The repetitive patterns on the ground largely influenced this change as the features it captured of the floor were unnecessary for accurate loop closures and diverted attention from more unique features found on the walls of the cafe. This is illustrated in Figure 4. by clusters of features. The color difference is not significant, simply different software color schemes for the same data.

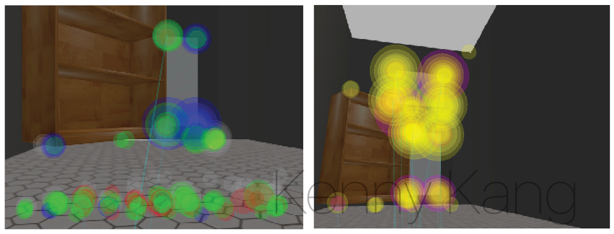

Fig. 4. Feature clusters before(left) and after(right) pitch adjustment

3.2.3 *RTAB-Map*

In comparison to the Kitchen and Dining world, accurate mapping and localizations were not possible with the default settings. While adjusting the pitch of the camera allowed for more useful features to be detected, there were still issues with the density of features in problematic areas as seen in figure 5. The left image of a brick wall detected many features, but repeats across the world which leads to inaccurate loop closures. The image of the right is a unique object to the environment, but captured significantly less features. Since loop closures are determined by a correspondence threshold, the more features that are detected, the more likely a loop closure is to occur. Therefore the maximum number of visual features of a frame being considered was reduced. In addition, SURF parameters were altered to encourage the detection of larger features to balance out the dominance of small patterns on walls that were causing incorrect loop closures.

4 Results

4.1 Kitchen and Dining

The SLAM algorithm on the kitchen and dining world produced a very accurate map with only 1 to 2 passthrough of. This was achieved using only the default settings of the RTAB-Map ROS package.

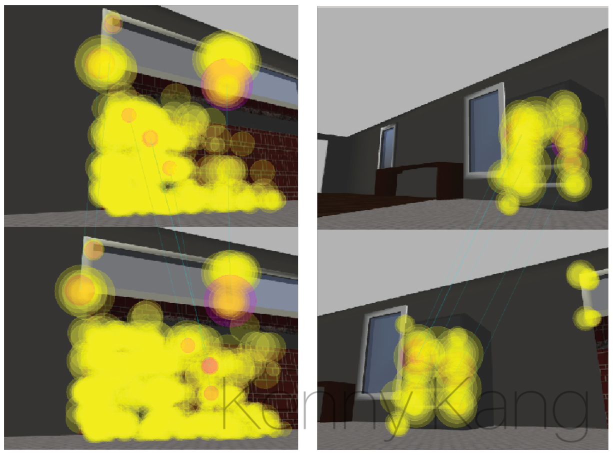

Fig. 5. Feature Density and Loop Closure Accuracy

TABLE 3
Dirty Cafe RTAB-Map Parameters

Parameter	Value	Description
Kp/DetectorStrategy	0	Speeded-up Robust Features (SURF)
Kp/MaxFeatures	100	Max visual words per image (bag-of-words)
SURF/Hessian Threshold	400	Sensitivity to features detection
SURF/Octaves	10	Size of detected features
SURF/OctaveLayers	5	Determines scale invariance of features
SURF/GpuVersion	true	Determines utilization of GPU
Reg/Strategy	2	Performs loop closures with a visual transform(RANSAC) followed by a geometric transformation (ICP)
Vis/MinInliners	15	Determines utilization of GPU

4.2 Dirty Cafe

Unlike the previous world, the cafe world produced many incorrect loop closures, particularly in one area as a wall produced a repeating pattern. While tuning parameters did not completely eliminate incorrect loop closures, they were being correct shortly after.

The most common issue with the benchmark model was its tendency to ”look back”. This is caused by its uncertainty of it’s environment and the possibility of a quicker path in this direction. While this did not affect it’s overall performance, this hindered the overall time in which the robot reached its goal.

5 Discussion

The effectiveness of the RTAB-Map algorithm was extremely dependent on the environment as seen from these experiments. The single biggest factor for this difference seemed to have been the presence of patterns throughout the world as the detected features were interpreted as corresponding features when they differ in location. This seems to be an issue difficult to find a solution for, even more so in simulations where consistency of patterns become a hindrance. The methods used to counter this involved restricting feature detection which is counter-productive as more unique features adds robustness to the algorithm.

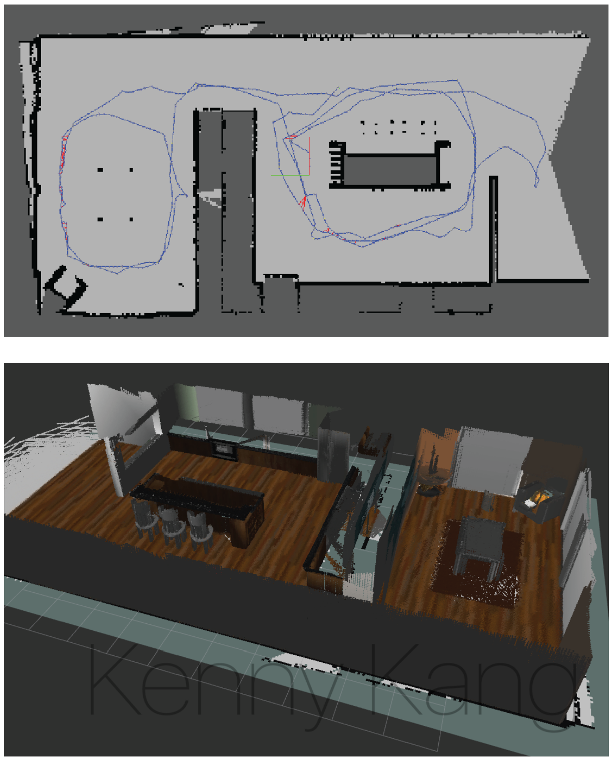

Fig. 6. Kitchen and Dining 2D(top) and 3D result(Bottom)

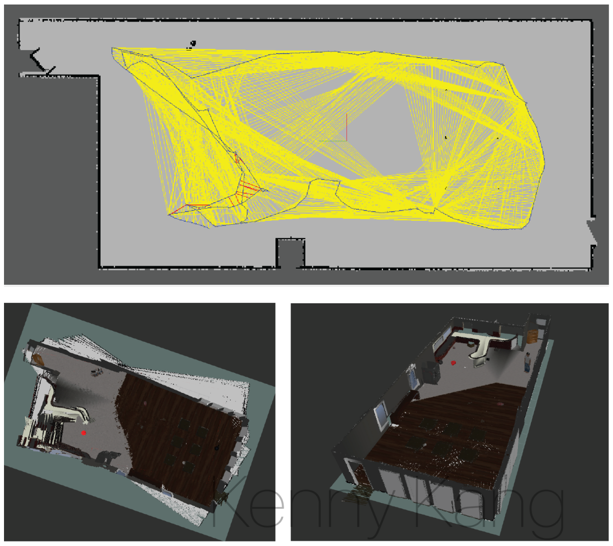

Fig. 7. Cafe 2D result(top). Before(bottom left) and after(bottom right) altering parameters.

6 Conclusion / Future work

SLAM algorithms are extremely useful for robots that are being implemented in unmapped environments, however with the versatility comes a need for a more robust algorithm. From this experiment alone, it can be observed that RTAB-Map is not robust to repetitive features which are prevalent in the real world. The SURF feature detector is the bottleneck and can potentially be replaced.

Convolutional Neural Networks are capable of abstracting features at different levels according to each layer of the network. Therefore, being able to describe regions of an image according to higher level features as opposed to lower level ones can be beneficial for ignoring patterns. This is based on the assumption that many of these problematic patterns are made up of simple shapes.