COMP-150-BBR: Final Project Report

Constantin Berzan

December 15, 2010

Abstract

In Simultaneous Localization and Mapping (SLAM), a robot builds a map of an unknown environment, while at the same time keeping track of its location within the map. The goal of our project has been to implement a SLAM system based on the Extended Kalman Filter. This document describes our progress towards that goal, focusing on landmark extraction, data association, and mapping.

All code for this project can be found at http://github.com/cberzan/bbr-mapper.

1 Introduction

A robot navigating an unknown environment is faced with two related tasks: mapping and localization. Mapping is the problem of building a representation of the environment, or answering the question, "What does the world look like?" Localization is the problem of determining the robot's pose (position and orientation) on the map, or answering the question, "Where am I?" [2] In practice, these problems cannot be solved independently of each other. Simultaneous Localization and Mapping (SLAM) attempts to solve both problems in parallel, iteratively using feedback from one to improve the answer to the other. SLAM has been seen as a 'holy grail' of mobile robotics, because its solution would allow robots to be truly autonomous in novel environments [3].

Our goal for this project has been to implement a simple SLAM system. We followed an Extended Kalman Filter (EKF) tutorial by Riisgaard and Blas [1]. The work was divided as follows: Andy worked on the EKF, and I worked on extracting landmarks and building a persistent map. We had initially planned on mapping an entire building floor, but difficulties with the EKF have forced us to restrict our scope to landmark re-observation and simple map-making. The rest of this document is organized as follows: Section 2 provides a more detailed description of the SLAM problem, and its EKF-based solution. Section 3 gives details about our implementation, focusing on landmark extraction, and presents some experimental results. Section 4 concludes.

2 The SLAM Problem

The SLAM problem is very general: it applies to any kind of robot, with any kind of sensors, in any kind of environment. We focused on a relatively simple case: building a 2D map indoors using odometry (how much the wheels have turned) and a laser range finder. We used the Videre ERA

base, the Hokuyo URG LRF, and ADE. We tested our robot on the first floor of Halligan hall, which has several hallways connected by 90 degrees turns, and no cyclical paths.

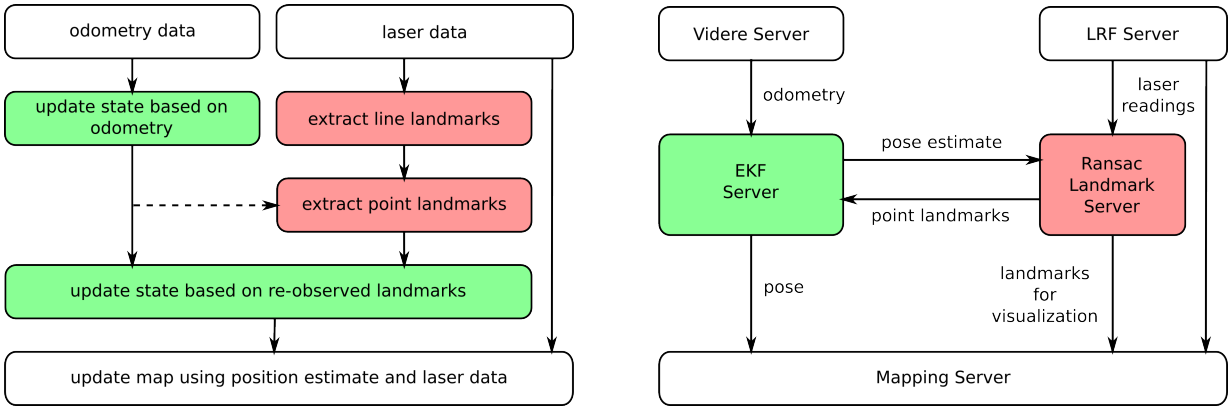
Because of the uncertainty inherent in the robot's sensors, probability plays a central role in any SLAM system. The core idea of SLAM is to get an estimate of the robot's position from odometry, and then use features in the environment to improve that estimate. This raises three questions: First, what features in the environment to detect, and how to detect them reliably (the landmark extraction problem). Second, how to recognize when the same landmark is seen repeatedly in subsequent scans (the data association problem). Third, how to use the odometry data and re-observed landmarks to estimate the robot's position (the integration problem).

Several solutions to each of these problems have been proposed. In order to be useful, landmarks have to be stationary and easily identifiable. The type of sensors also restricts the choice of landmarks. Data association can be done by comparing observed positions with estimated positions, or by comparing landmark features regardless of their expected position. The most common approach to the integration problem is the Extended Kalman Filter. This statistical method uses several sources of data (odometry, landmarks) to bound the error in its pose estimate. A newer and more complicated approach to integration is Monte Carlo sampling, or particle filtering.

Following Riisgaard and Blas [1], we use lines and points for landmarks, position comparison for data association, and an EKF for integration. The next section describes our solution in detail.

3 Our SLAM Implementation

Figure 1 displays a conceptual diagram of our solution. The EKF updates its pose estimate from odometry. Line landmarks are extracted from laser data, and converted to point landmarks using the estimated robot pose. The EKF gets the point landmarks, and re-estimates the robot pose. Finally, the map is updated using the latest pose estimate and laser data. Simple navigation is implemented as schema-based random exploration with obstacle avoidance (not pictured).



(a) Conceptual design

(b) Code organization

Figure 1: Conceptual design (a) and code layout (b) of our SLAM system. Arrows indicate flow of data. Green boxes indicate EKF components, and red boxes indicate landmark extraction components. Each server in (b) runs in a separate process. Communication is achieved through ADE calls, made either by one of the servers pictured, or by the architecture controller (ArchImpl).

The rest of this section describes our techniques for landmark extraction, data association, and mapping. We also discuss the strength and weaknesses of the methods used, and show some experimental results.

3.1 Landmark Extraction

Landmark extraction is critical for a viable SLAM system. If not enough landmarks are detected, or if they are only re-detected for short periods of time, or if a landmark is mistakenly associated with a different landmark from a previous scan, then the system's performance will suffer.

We have adopted and extended the idea of using lines as landmarks from Riisgaard and Blas [1]. Lines have the advantage of being easy to detect consistently. Their main disadvantage is that they do not directly tell the robot's position (for example, the two walls of a long hallway will look the same regardless of where the robot is along the hallway). Moreover, our EKF expects the landmarks to be points relative to the robot, so the lines have to be converted to points somehow.

Figure 2 shows the three coordinate systems involved in landmark extraction. The origin of the World coordinate system is fixed at the robot's initial position. The origin of the Robot coordinate system is the robot's center of rotation, and its x axis points towards the forward direction of motion. The origin of the Laser coordinate system is the laser range finder, and its x axis corresponds to the 180-degree range of the LRF.

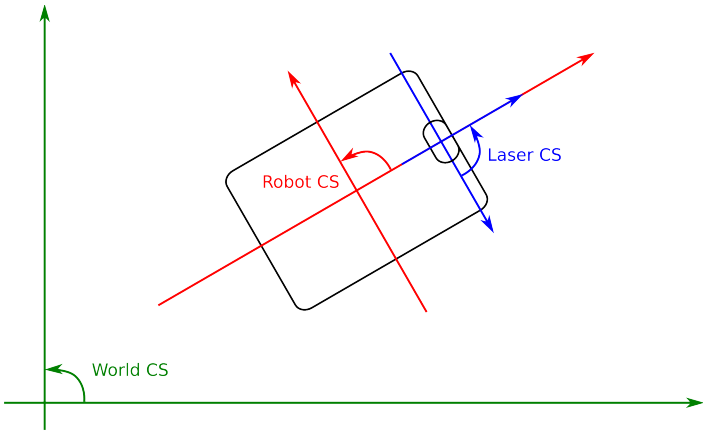


Figure 2: The three coordinate systems involved: Laser, Robot, and World. The goal of localization is to determine the position of the RCS relative to the WCS, in other words, the robot's pose.

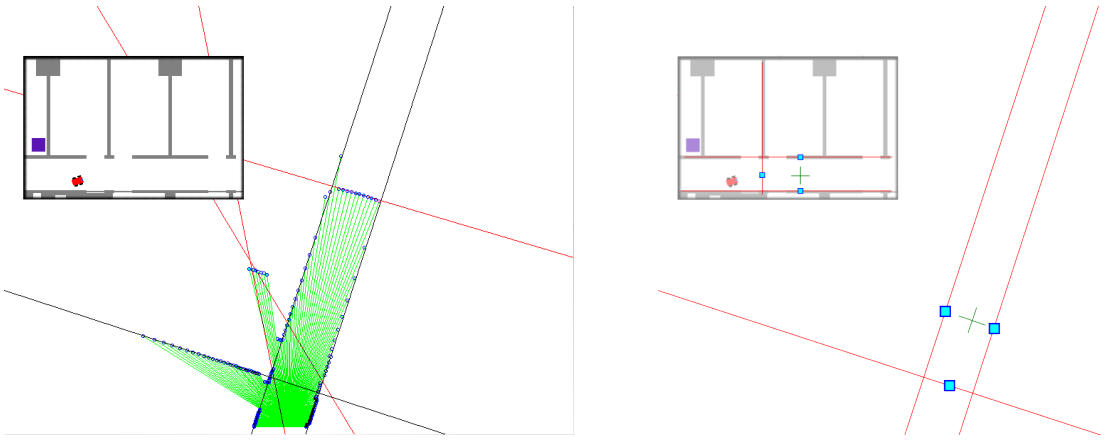
We first convert our polar laser readings into Cartesian coordinates. Then, we use a Random Sample Consensus (RANSAC) algorithm to fit straight lines through these points. Table 1 outlines the basic algorithm. We made several changes to improve the algorithm's speed and reliability. Since our points are ordered by angle, we sample from consecutive laser readings. This finds lines very quickly, usually in under 10 iterations. We compute the R2 coefficient for the *consensus line*, and if it is below a certain threshold, we save the original *sample line* instead. This deals with the uneven point density (because there is one point per degree, there are more points nearby than far away, and they can skew the consensus line). We run the entire greedy algorithm in 10 meta-iterations, choosing a fitted line set that (1) matches as many points as possible, and (2) does not have many confound points (points that fit more than one line).

Input: set of points $\mathcal{P}$
Output: set of fitted lines $\mathcal{L}$
Parameters: sample size m, minimum number of consensus points N, maximum number of iterations k, distance threshold t
Repeat while $\|\mathcal{P}\| \geq N$ and k iterations not reached:
Let $\mathcal{P}_{sample} = \{m$ random points from $\mathcal{P}\}$
Fit line ℓ_{sample} through points in $\mathcal{P}_{sample}$
Let $\mathcal{P}_{consensus} = \{$ all points in $\mathcal{P}$ within t distance of $\ell_{sample}\ \}$
If $\|\mathcal{P}_{consensus}\| \geq N$:
Fit a new line $\ell_{consensus}$ through points in $\mathcal{P}_{consensus}$
Add $\ell_{consensus}$ to the output set $\mathcal{L}$
Remove all points in $\mathcal{P}_{consensus}$ from $\mathcal{P}$

Table 1: The basic RANSAC line-detection algorithm.

3.2 Data Association

We first convert the detected lines to the Robot coordinate system. This is a constant transformation that can easily be measured or guessed. We then use the pose estimate from the EKF to convert the lines to the World coordinate system. We project the origin of the WCS onto each line, and use the resulting point as a landmark. Figure 3 shows an example.



(a) Line detection (b) Landmark extraction

Figure 3: Landmark extraction. **(a)** RANSAC line detection showing the fitted lines (black) and other lines that were tried (red). Note that the far line at the edge of the LRF's "visual field" is correctly rejected, because there is insufficient consensus. **(b)** Landmark extraction showing the WCS origin (green cross), and the projection of the origin onto each detected line.

We compare the new point landmarks to previously seen landmarks. An association is made when the position difference is within a small error margin. Otherwise, the landmark is stored as new. When the EKF asks for a list of landmarks, we only send the landmarks which (1) were observed in the last scan, and (2) were seen at least a minimum number of times.

3.3 Mapping

Riisgaard and Blas [1] seem to be satisfied with a system that localizes itself within the map it has built, without creating a human-readable representation of the map. One of our goals has been to create an image representation of the map, one that would be meaningful to a human.

For this purpose, we use a simple occupancy grid. The world is represented as a grid of cells, with each cell holding a probability that it is occupied. On every laser scan, all cells where a laser beam ends get their occupancy increased. The visualizer simply paints each cell, using a shade of gray determined by the occupancy value. We have considered making the occupancy of cells decay with time, but this was unnecessary for the short runs we subjected our robot to.

3.4 Analysis

We chose the EKF-based approach because it was the older, more established method, and we had a novice tutorial describing it [1]. Unfortunately, we found out that the tutorial was incomplete, and Riisgaard and Blas never got their SLAM system to work all the way. With all the right components in place, our EKF would fail to converge. At this point, we decided to restrict the scope of our project to building a simple map while re-observing landmarks. We accomplished this by short-circuiting the EKF Server in Figure 1b, and making it pass through the unmodified odometry data.

Even though we did not get to test our landmarks with a working EKF, we can talk about the advantages and disadvantages of the method we used. The main advantages are the simple design, and the ease with which lines can be re-observed. The main drawbacks are that the estimated pose is required for determining the point landmarks, and that our method for converting lines to points makes the point landmarks cluster around the origin of the World coordinate system. While this is fine for a proof-of-concept SLAM system, more advanced landmark extraction would be required for building a larger map.

3.5 Experiments and Results

Without a functional EKF, we had to rely on odometry data to obtain the robot pose. During earlier odometry testing on the Videre base, we determined that the odometry was more accurate at higher speeds. We therefore set the maximum translational speed of the robot to 0.5 m/s, and the maximum rotational speed to 0.2 rad/s. Obstacle avoidance worked fine at these speeds.

We tested our system in several trials. Figure 4 shows a typical result. The odometry served as a good estimate of robot pose for the most part, with one notable exception: Whenever the robot made a large turn (more than 90°), the angle in the odometry became inaccurate. We relaxed the distance threshold t in RANSAC a little bit, because we found that the real-world LRF was more noisy than the simulated one. While going through a straight hallway, the two landmarks representing the walls were re-observed consistently.

Figure 4: Sample generated map. The robot started at the green cross, went right, and then left. The highlighted area on the right is the point where we put an obstacle in front of the robot to force it to turn. This large turn skewed the theta component of the odometry. The highlighted area on the left is a point where people repeatedly stepped in front of the robot, to test its response. The robot had to stop and start repeatedly, but it did not make any major turns, so the odometry remained accurate.

4 Conclusion

We started with a plan to do SLAM using an EKF. We did not manage to get the Kalman Filter to work, so instead we focused on basic mapping and landmark re-observation. Our landmarks were obtained by projecting a point onto lines extracted from laser data. We did mapping using a simple occupancy grid. We tested our robot in the real world, and got acceptable results on short runs.

One of the main lessons of this project has been the difficulty of implementing and debugging a probability-heavy algorithm such as the EKF. We would not recommend attempting to get one working in two months, unless one has solid background in control theory or statistics. We also familiarized ourselves with a line detection algorithm (RANSAC), and a mapping strategy (occupancy grid). Additionally, we improved our knowledge of the ADE framework, by learning how to design a multi-process system, and how to write visualizations.

As section 3.4 suggests, it would be interesting to look at other kinds of landmarks, such as doors, corners, edges, etc. It would also be interesting to detect landmarks using vision, rather than just a laser range finder. Finally, many applications probably require a solution that works in 3D rather than 2D. SLAM is a problem that can be worked on indefinitely, since there are improvements that can be made no matter how advanced a system becomes.

5 References

1. Riisgaard, S. and Blas, M. R. (2005). *SLAM for Dummies*

2. *OpenSlam.org*, a public repository of SLAM code. http://www.openslam.org

3. Durrant-Whyte, H., Bailey, T. (2006). Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms. *Robotics and Automation Magazine* 13: pp. 99–110