Videre SLAM & The EKF

Andy Sayler

December 15, 2010

Abstract

This paper provides an explanation of the steps my partner, Constantin Berzan, and I undertook to implement a SLAM system on the Videre robotic base running the ADE architecture. This project was undertaken as part of the Tufts University Behavior Based Robotics course led by Prof. Matthias Scheutz in the fall of 2010. My work on the project, and thus this paper, will focus on the EKF components of the SLAM system and the Videre modeling necessary to build the EKF. As of the writing of this paper, the EKF is fully functioning, but is unstable and can not be relied on to provide accurate state estimation for the larger SLAM system. Efforts to stabilize our implementation are beyond the scope of this project.

The code for this project is available at `https://github.com/cberzan/bbr-mapper`.

Contents

1 Introduction

As robots become more and more part of our daily lives, and as we intrust them with increasingly human-like responsibilities, the seemingly simple abilites that most humans take for granted become increasingly important. Take, for instance, a simple human concept: knowing where one is located. While most humans might not be able to rattle off their exact world coordinates on command, we can all tell you where we're standing with respect to the room we're occupying, and where that room is relative to the point at which we entered the building. For a robot, however, these tasks are far from simple.

We have been trying to solve the problem of giving robots awareness of their location and surroundings for as long as we have had mobile robots. Through the 1970s and 1980s many attempts were made to solve this problem. It wasn't until 1986 that the first hints of the present approach to solving the localization and mapping problem began to take shape and the concept of SLAM was born [6].

SLAM is an acronym for **S**imultanous **L**ocalization **A**nd **M**apping. It describes a set of probabilistically based solutions to the localization and mapping problem. The key to the SLAM approach is that it acknowledges and leverages the fact that knowing where you are (locational awareness) and knowing what your surroundings are (environmental awareness) tend to be tightly coupled problems.

Since one often utilizes one's surroundings to aid in determining one's location, having a sound map of your environment is very helpful to determining where you are located. Likewise, in order to construct a map of your environment, one must have some idea of where they are located so as to provide context for the world they are observing around them. It is this inherent coupling of localization and mapping problems that has made the SLAM approach to solving these problems so appealing.

In this paper, I will discuss the components of a standard SLAM system, our SLAM implementation using an Extended Kalman Filter, and the performance of our SLAM system.

2 Problem Description

2.1 SLAM

As mentioned in the introduction, "SLAM is the process by which a mobile robot can build a map of the environment and at the same time use this map to compute its location" [7]. There are a multitude of ways to implement SLAM systems, and SLAM is far from being a "solved" problem. Each approach to SLAM has it pros and cons, with difficulties often involving striking a good balance between performance and computational complexity. It's a field of active research and development. Good sources for more detailed information SLAM concepts include [6], [7], and [5].

In this project, we took a fairly traditional approach to SLAM, using an Extended Kalman Filter based SLAM implementation. Figure 1 shows the basic system level flow of the EKF SLAM process. The SLAM process involves two major parts: landmark extraction and state estimation. Landmark extraction aims to provide the robot with a set of re-observable landmarks that can be used as reference points for state estimation. State estimation aims to use the observed landmarks and additional sensor and system model data to estimate the "true" state of the system at a given moment in time.

We utilized a RANSAC line-detection based landmark extraction system for our SLAM implementation. This system is discussed in [1] and the implementation was primarily undertaken by my partner, Constantin Berzan. The details of the landmark extraction implementation will not be discussed here. For more information on this component, please consult Constantin's paper

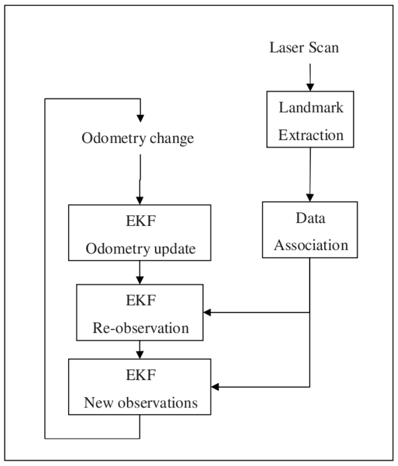


Figure 1: EKF Based SLAM Overview [1]

regarding this project.

2.2 The EKF

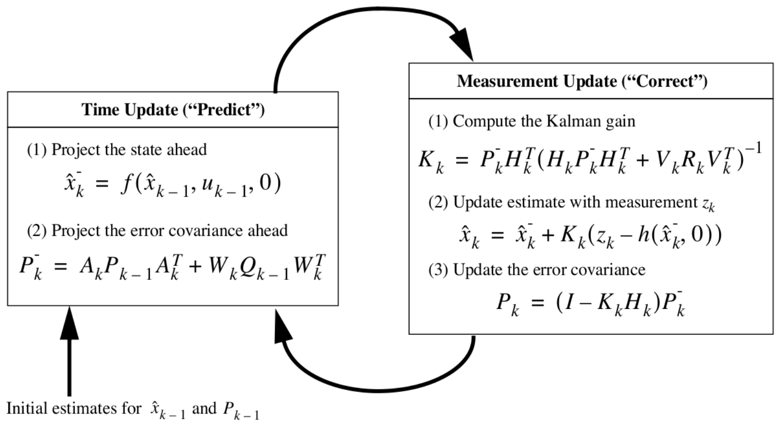
The second major component in any SLAM system is some form of state estimation algorithm. In this project, we used an Extended Kalman Filter to provide state estimation. The Extended Kalman Filter is a linearized version of the Kalman Filter. More information on both can be found in [4].

The purpose on an EKF is to deduce an accurate estimation of the state of a system given a set of noisy measurement and imperfect system models. In robotics, this is enormously helpful, as the system tends to be too complex for highly accurate system models and the environments in which robots operate tend to be very noisy. An EKF provides the tools necessary to take a set of noisy measurements and state predictions from a variety of sensors or models and estimate the "true" state of the system more accurately than any model or sensor measurement would be able to provide independently.

The EKF algorithm involves two primary steps: "prediction" and "correction". Figure 2 shows the relationship between these two steps and the equations involved in each. The EKF is a cyclic filter that runs in a continuous loop providing a new state estimate each loop iteration based on the most recent sensor measurements and predictions.

The "prediction" step of the EKF uses either a system state model or direct system state measurement to predict the current state of the system. The "correction" step uses information about the position of a set of landmarks relative to the robot to correct the predicted state and converge toward the "true" system state. The EKF calculates a running covariance for each landmark and state measurement and uses these to calculate how much each measurement or state model can be "trusted", in effect weighting each input to the EKF by the likelihood that the given input reflects the "true" state of the system. More accurate measurements (lasers, etc) end up getting weighted more heavily than less accurate measurements (system odometry, etc).

When properly tuned, an EKF is capable of converging over time on the "true" state of the system even when the individual inputs and landmarks are noisy.



Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$
(2) Project the error covariance ahead
$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$
(3) Update the error covariance
$P_k = (I - K_k H_k) P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure 2: The EKF Equations [4]

3 Implementing the Extended Kalman Filter

3.1 Videre Data

Since the EKF relies on having some idea about the accuracy of the various input sensors it relies on, we began by collecting a set of odometry test data for the Videre base. This was done by instructing the Videre base to drive a given distance and than measuring the actual distance the robot drove. We repeated our tests for a variety of distances and speeds. The results are summarized in table 1. In general the Videre odometry tends to increase in accuracy at higher speeds.

Speed	**Distance**	**Normalized Error**
.1 m/s	2 m	11 cm/m
.25 m/s	2 m	7.6 cm/m
.5 m/s	2 m	1.1 cm/m

Table 1: Videre odometry test data [8]

3.2 EKF Code

The EKF code for this project is implemented in its own ADE server: EKFServer. This was done to allow the EKF computations to complete asynchronously from the other portions of the code base. Since the EKF calculations can often become computationally complex, isolating it in its own server helps to insulate the rest of the project code from delays due to EKF computations.

The EKF implementation closely follows that described in [1], with the missing gaps filled in from information in [4] and [5]. The EKF uses the robot's odometry data to "predict" the current positions and then uses the RANSAC landmarks (derived from the laser sensor) to "correct" its position estimate. The Jama package [9] provides the necessary matrix manipulation and computation functions.

The EKFServer provides a single function to its clients: Pose getPose(). This function provides the most recent robot Cartesian state estimate to the caller. The EKF relies on both the Landmark-

Server and either a Videre or simulator ActionServer to provide it with updated landmark and odometry data.

3.3 Results

Unfortunately, as of the writing of this paper, the EKF is not functioning correctly. The EKF is fully implemented, and all steps and equations have been coded and verified, but the state estimate provided by the EKF is unstable. Thus, instead of converging as an EKF should, the EKF state estimate diverges, quickly running away and providing highly inaccurate results. For the functioning version of the code-base, the EKF has been short circuited and the rest of the SLAM system relies solely on the Videre's odometry to provide a running state estimate.

Even without a working EKF, we were able to construct rudimentary occupancy grid maps of the robot's surroundings and detect and re-associate landmarks. More details on these results are available in Constantin Berzan's paper. These results, while an acceptable proof-of-concept for a SLAM system, do suffer from inpercise state estimates. These errors introduce a "smudging" evident in the output occupancy grid maps and occasionally cause what should be a consistent landmark to be detected as multiple landmarks.

3.4 Analysis

While we did not have time to complete a thorough analysis of the instabilities evident in our EKF, we do have some theories as to the origin of these irregularities. First, most EKF system rely on landmarks that are detected independently from the current state estimate. For example, point landmarks derived directly from laser sweeps provide a direct measure of the relative angle and distance to the landmark from the robot. When these measurements are incorporated into the EKF, they provide an independent measurement off which to correct the state estimate.

RANSAC landmarks, however, rely on the current state prediction to formulate which point on the detected line should be used as the landmark. This creates a coupled dependency between the EKF estimate and the EKF inputs that might destabilize the EKF if not accounted for correctly. We did not have time to dive into the analysis of this coupling, but it would be a good place to start hunting for possible causes of the instability.

Furthermore, the RANSAC algorithm requires a far more complex error and accuracy model than we was available to us. Thus, the EKF simply estimates a RANSAC landmark's inaccuracies as being linear in distance and constant in orientation. While this error model is appropriate for the base RANSAC laser readings, it might not be appropriate for the full RANSAC landmark algorithm, leading the EKF to misinterpret that accuracy of the landmark readings and destabilizing the state estimate as a result.

It should be noted that the authors of [1] allude to these difficulties on there personal comments regarding their tutorial. These comment are available in [2] and [3].

4 Conclusion

While we were unable to complete a fully functioning EKF, we were able to build a full SLAM system and demonstrate the rudimentary mapping output and landmark detection that such a system can provide. The EKF, while unstable, is functionally complete, and could probably be stabilized with further tuning, experimentation, and error analysis.

Even without the aid of the EKF, our system is capable of randomly exploring it's surroundings and providing a usable two dimensional graphical map of its surroundings. The EKF, once stabilized, can be easily reinserted into the system to provide even more accurate mapping and landmark identification capabilities. Another option to increase the performance of our SLAM system would be to abandon the EKF altogether in favor of another form of state estimation. The code base is constructed in such a manner as to make such a replacement fairly trivial.

This project revealed the difficulties associated with troubleshooting an algorithm as complex as an EKF. Since the EKF algorithm maintains a number of large matrices, few of which have any intuitive relationship to each other, trying to debug an EKF by analyzing its intermediate or final outputs is nearly impossible. Instead, the EKF must be analyzed mathematically to arrive at an appropriate solution, and then this solution must be implemented in the code base. This is a tedious and error prone process. I imagine that an entire paper could be written on EKF troubleshooting techniques.

This project was successful in demonstrating the power of a SLAM system to detect a robot's location and provide a map of the robot's surroundings. We feel we have gained a valuable understanding of the SLAM process and the various pitfalls associated with building SLAM systems.

References

[1] Riisgaard & Blas. MIT OpenCourseWare. *"SLAM For Dummies"*. 2004.

[2] Blas, Morten Rufus. MIT OpenCourseWare. *"Personal Insights in Course Project"*. May 12, 2004.

[3] Riisgaard, Soren. MIT OpenCourseWare. *"SLAM For Dummies: Personal Comments"*. 2004.

[4] Welch & Bishop. UNC Chapel Hill. *"An Introduction to the Kalman Filter"*. July 24, 2006.

[5] Nebot, Eduardi. Australian Centre for Field Robotics, The University of Sydney. *"Simultaneous Localization and Mapping 2002 Summer School"*. July 31, 2002.

[6] Durrant-Whyte & Bailey. IEEE Robotics & Automation Magazine. *"Simultaneous Localization and Mapping: Part 1"*. June 2006.

[7] Durrant-Whyte & Bailey. IEEE Robotics & Automation Magazine. *"Simultaneous Localization and Mapping: Part 2"*. 2006.

[8] Berzan & Sayler. "Videre Odometry Data". `https://spreadsheets.google.com/pub?key=0AimqizebBXDSdDhVUU5BOUJnaFQ2S2VjQ2NiMWZKYVE&hl=en&single=true&gid=1&output=html`. November, 9th, 2010.

[9] *Jama: A Java Matrix Package*. The MathWorks and the National Institute of Standards and Technology. `http://math.nist.gov/javanumerics/jama/`. July 13, 2005.