SLAM for Dummies

A Tutorial Approach to Simultaneous Localization and Mapping

Personal Comments

by

Søren Riisgaard

Introduction

This document describes the insights I have gained during implementation and documentation of the SLAM process. I have, with Morten Rufus Blas, implemented SLAM and written a tutorial on the subject. During this work I have learned a lot of the key elements of the SLAM process and the problems one will encounter when working in this area. I will describe the major issues here, the issues that I find most important.

As the SLAM process is, roughly speaking, divided into a number of steps I will follow this chronological order in this document, starting with the landmarks and landmark extraction as this is the first task one faces when doing SLAM.

Landmarks

Landmarks are geometric objects (when using a laser scanner), that can be recognized over and over again. Specifically some properties of landmarks are important:

They should be re-observable, distinguishable from each other and stationary. Furthermore we need a critical number of landmarks to be able to localize ourselves. Landmarks can be extracted in a number of ways. We initially started with spike landmarks, which seemed as a fine solution in the beginning. We later found out that they were not precise enough. The spike landmarks gave lots of landmarks, but that does not help if they are not precise. Instead of the spike landmarks we decided to go for the RANSAC landmarks. They proved to be much better and a lot more precise. The increased precision was probably also because we now use a number of laser scanner samples and gather these into one landmark. So the lesson learned is that you are much better off using landmarks that are more accurate than just mere points.

In the EKF we have used the range to a landmark as a measurement of the error on the measurement. E.g. the error can be 1 cm/m and with a range to a landmark of 2 m the error is 2 cm. This posed a problem when using RANSAC as the range sent to the EKF is not also a measure of the length of the laser scanner readings, since these are only used to calculate the landmark position. Remember the landmark is the point on the line found, which is closest to (0,0). If e.g. the robot is 100 meters from (0,0) the landmark may be close to (0,0) meaning that the landmark may be up to 100 meters

from the robot, even though the laser scanner can only measure just around 8 meters! So we decided to use the point closest on the line, to the robot as the error range and the error bearing. The fact that most landmarks will tend to accumulate around (0,0) also means that there may be a lot of landmarks just around this point. When doing data association it can be hard to distinguish the landmarks from each other if they are all accumulated around (0,0). This means that our RANSAC landmark extraction strategy will not really scale very well with increasing number of landmarks.

We also used RANSAC in a, to us, novel way. We extract multiple lines instead of just a single line. Whenever some points have been associated to a line they are removed for this round, so they will not also be a part of other lines. This proved to be very useful. It gave us very exact landmarks and also few landmarks, which is preferable. Having less landmarks gives less problems with data association and more precise landmarks is of course the best.

I have learned that the data association and the way landmarks are added to the database should be tuned to the landmark extraction used. For example using the spike landmarks there was much need for removing landmarks, since there was very many spike landmarks. An effective data association algorithm was also needed since the landmarks tended to lie close to each other. When adding landmarks we decided that they had to be observed a certain number of times before they were actually added as landmarks. Also this number of times depends on the quality of the landmarks and on the odometry performance. If the odometry is really bad, it can on one hand be a good thing to hurry up and add a landmark, so more landmarks are available for use in navigation. But on the other hand we do not want to include bad landmarks in the state. So a balance has to be found. Maybe it would be an idea to couple the robot covariance and the addition of landmarks, such that a landmark measured two times from a position with very low covariance gets added, while a landmark observed ten times from a position with high covariance should not be added. This way a dynamic addition of landmarks could be used.

We also found an optimization in adding new landmarks after updating the old landmarks. This way there is less data to process. It is not a very complicated optimization, but nonetheless it is effective and easy to implement. It is just a question of rearranging the order in which the functions are called. Summarizing it is important to really think of a desired data association policy and a policy for adding landmarks.

In general there is a lot of tuning of parameters to find the right balance in the EKF. The measurement error and the process error is an example of this. They should resemble the error probabilities of the measurement device and the robot odometry performance. But in reality what you want to do is to tune the errors to obtain the right learning rate. If for example the odometry is very precise and the measurement device is not precise it will mean that we only trust the odometry. So if we want to use the measurements more we need to either decrease the measurement error or increase process error. Also if this means that the values do not resemble the actual physical properties they resemble. As a last thing, regarding the errors we also found that having too little error yielded a computation error. This is due to the innovation covariance in the Kalman gain. If the innovation covariance results in a zero matrix we will not be able to get the inverse of the matrix. So we always need to maintain some degree of uncertainty.

The SLAM process is a quite interesting field because lots of the steps performed can be performed in different ways. There are an infinite number of permutations that may qualify as SLAM, each with different properties and advantages. During the creation of the tutorial I have only touched on a few of these possible solutions, but during the literature review I found an amazing “add-ons” to SLAM. This is what makes it so interesting because SLAM can be tuned and tweaked to fit your needs, if not you may invent a new way so it fits your needs.

SLAM should not be so hard; it is after all just a number of matrices and operations on these. And it is not. The problem is that it can be hard to make sense of the intermediate results in all the matrices. The only real result there is to check if SLAM has succeeded is if the robot moves correctly. If the final result is not correct it can almost impossible to find the error.

Final notes

Unfortunately we did not manage to make our SLAM application completely bug-free, which is a little ironic since we have written a tutorial on the subject. This is why we have not included the source code. I feel that it does still make sense to write a tutorial, since we have the whole process right. This also underlines the fact that it is very hard to debug a SLAM application, since many of the intermediate results are now known, but one can mostly only rely on checking the final output. An extended

version of the tutorial would make sense, which should include information on debugging SLAM. This would of course require that our application works.

Conclusion

Overall I gained a lot of insight and a deeper understanding of the SLAM process, how data flows in the system and what effect the different operations have on the overall output. It has been really helpful to write this tutorial and proves that you really learn a subject by teaching it. Not only the SLAM process has been challenging but also all the other stuff and hurdles have let me get a feeling on the problems one faces when working with robots. I feel that I am ready to begin thinking of extensions to SLAM, ideas that would make the process better, contributing to the SLAM community.

The downside of this project was that our SLAM did not work completely correct, thus disabling us of including the complete source code.