EL3320 Applied Estimation: Projects

The project part of the course allows you to focus on a particular subject. The project is an important part of your final grade in the course and you should work in groups of two to solve a problem of your choosing. Groups of one or three are also allowed but two is preferred. The implementation is an important part of the project but please remember that what is not described in the project report will not be taken into account in the grading. That is, we will not look for things in the code that will give you a higher grade, you need to tell us about them in the report. The implementation is best presented as experimental results including plots in the report to document your achievements. These are then supported by the code and instruction for how to run it. You are free to include movie files and they can help show what you have done but they are not required.

The reports should be similar in format to published research articles. For example they might include abstract, introduction, background (including your reference citations), a description of your project, experimental results, conclusions (including reflections and insights you gained, very important). Maximum length is 10 pages. We refer you to the course PM for the formal requirements and grading criteria. So suggested estimators one could look at would include, EKF, PF, UKF, PHDfilter, Mixture of Gaussians, or MHT. SLAM specific estimators include Graphical SLAM, SEIF, FastSLAM. An overly ambitious project might tackle advanced methods such as MCMC or simulated annealing. You are free to chose any of these or some other but it is best to upload a project description to be sure that the choice is wise. There is definately a point of diminishing returns for complexity of the implmentation. A super presentation of a moderately complicated implementation with good experiemental evaluation is probably the most effective combination of efforts. If you are not going for an A you can chose a simple to implement method.

You are free to use software packages for handle matrix operations and similar but the implementation the filters and anything less that is core to the course should be your own. Remember that you are supposed to live up to the code of honor.

Extra Information for localization or SLAM projects

Some of the projects in EL3320 Applied Estimation deal with localization and mapping in mobile robotics. This page is intended to give some additional information and some pointers to help with those projects. We also provide a way to simulate data for those projects. EKF in the and particle

filter were introduced in the labs and one choice is to extend upon those. In the literature of mobile robot localization and mapping one can identify three problems

Localization given a known map Mapping given a known position Mapping and localization without a map (this is called Simultaneous Localization and Mapping or SLAM)

In localization one often distinguish between two problems

Tracking which refers to the situation where the initial position of the robot is known at least roughly. and global localization where the initial position is not known

Global localization is typically substantially harder because the search space is much larger. In practice the possible positions can be narrowed down to a floor or even a single room but even so symmetries in the environment can make the problem hard to solve. Normally the global localization cannot be solved in a single iteration because of these problems. It requires the integration of data collected while robot is moving to solve for the position. A special case of global localization is the so called kidnapped robot problem which refers to the situation where someone takes the robot and carried it away and puts it down somewhere else without letting the robot know about it. This is hard because the robot first has to figure out that it has been kidnapped and then it has to re-localize. In a symmetric environment this is especially hard.

WE describe here how the lab data was simulated to give you an idea of how to generate your own such simulations. The scenario that we model in the simulation of the labs is a differential drive mobile robot equipped with a sensor that can measure bearing and range to landmarks in the environment. The robot is equipped with one wheel encoder per wheel. These measure the rotation of the wheels. Given that the radius of the wheels (rR and rL) and the distance between the contact points of the wheels (the wheel base) (B) are known one can estimate how the robot moves. This is known as odometry. The odometry estimates of the pose (position + orientation) of the robot relative to the starting point of the robot or rather the position at which the odometry was zeroed. (This assumes that the wheels have been in contact with the floor at all times.)

A differential drive robot is actuated by two wheels. The rotation speed of each wheel (wR and wL) can be controlled individually. If we assume that the radius of the two wheels is the same (rR=rL) the robot will move straight if the set the same speed for both wheels. Different speeds on the wheels will make the platform turn. This is how the platform is controlled at the low level. Most application programmers prefer to control the robot using the translation speed (v) [m/s] and rotation speed (w) [rad/s]. Again,

knowing rR, rL and B the command (v,w) can be translated to the low level wheel speed commands wR and wL. The equation that relates the speed of the wheels (wR and wL) with the translation and rotation speeds (v and w) are:

v = (wR * rR + wL * rL) / 2 w = (wR * rR - wL * rL) / B which can be re-written as

wR = (v + B * w / 2) / rR wL = (v - B * w / 2) / rL A simple model for the motion of the robot is

x(k+1) = x(k) + v * dt * cos(theta(k)) y(k+1) = y(k) + v * dt * sin(theta(k)) theta(k+1) = theta(k) + w * dt where dt is the sampling interval.

The robot outputs the accumulated number of encoder "ticks" for each wheel in each iteration along with the accumulated odometric pose. To use the raw encoder reading you need to know the number of encoder "ticks" per revolution of the wheel. The default value for this is 2048. In addition to this you get the true robot pose and the measurements from the sensor. The true robot pose is helpful when testing you implementations and comparing to ground truth but remember that on a real robot you do not have access to this type of signal so you are not allowed to use it in your estimation. The measurements are of the landmarks in the map and are typically affected by sensor noise. Each measurement is tagged with the landmark it is a measurement of. This provides a way to sidestep one of the harder problems in robot localization and mapping, namely that of data association. That is, how do you know what part of the environment the measurements give information about? We can thus study the estimation problem in isolation from the problem of data association. It is however encouraged to perform experiments in the project without this information to see how this affects the problem.