Pavel Zhdanov

The purpose of this project is to implement Simultaneous Localization and Mapping using a Lego robot. Simultaneous Localization and Mapping is the process of building a map of an environment and simultaneously find the configuration of a robot within that map

**Algorithm**

SLAM can be viewed as a filtering problem with an enormous state space. That is we want to compute the joint posterior probability of robot configuration *xk* and map configuration *m*given a history of *k*observations *o*0:*k* and actions *a*0:*k*:

P (xk,m |o0:k,a0:k)


To estimate this distribution, we will use a Rao-Blackwellized particle filter. Particle filtering is a way of estimating a posterior distribution by using a large number of samples. Each particle, or sample represents a possible pose of the robot and an associated analytic representation of the map estimate.

Given a motion model

P (xk|xk−1,ak)


And an observation model

P(ok|xk,m )


Particle filtering generates samples at each time step drawn from the posterior distribution over poses.

**Models**

**Motion Model**

The motion model used to estimate the pose of the robot is computed from wheel odometer measurements with added gausian noise. The pose at time *k*is represented as *xk,yk* position on the plane and orientation *θk*.

The change in odometry Δ*lk* and Δ*rk* are sampled from a normal distribution centered on the actual measurements. Then the change in *x*, *y*, and *θ*are computed as:

      Δr  − Δl
Δ θ = --k-----k
      wheel base


Δx  = cos(θ+ Δ θ)Δrk-+Δlk-
                    2


               Δrk + Δlk
Δy = sin(θ+ Δ θ)----2----


**Ultrasound Map**

The ultrasound map is represented as an occupancy grid. Each element of the grid contains the log-odds ratio of the probability of that grid point being ultrasound reflective.

Formally, let *mx,y* be a binary random variable indicating whether position *x,y*is ultrasound reflective, and *mx,y* be the negation of that variable. Then we compute the probability of the square *x,y*be occupied as

P(m   |o  ) = P(ok|mx,y,xk)P(mx,y|o0:k−1)
    x,y 0:k           P(ok|o0:k−1)


We then define the log-odds ratio as:

log P(mx,y|o0:k)-= log P(ok|mx,y,xk)P(mx,y|o0:k−1)
   P(�mx,y|o0:k)      P(ok|m�x,y,xk)P(m�x,y|o0:k−1)


Note that this assumes independence of the observation from previous observations. This is not a realistic assumption, but works marginally well. An improvement would be to adopt an EM approach like described in

**Ultrasound Sensor Model**

To compute *P*(*ok*|*mx,y,xk*) adapted a model presented by Moravec

from 1988. Roughly speaking, the approach works by stepping through the possible ranges of the sonar sensor and computing for each value *o*∈{1,2,...,max } the value of *P*(*o*|*m,xk*). Let *R*be the set of sensor values and *M*(*r*) be the set of grid squares in the range of *r*∈ *R*.

S  =   ∑     P (m   )∗ P
  o  (x,y)∈M (o)    x,y    sensor error


                 ∑
P (o|m ) = So × (1−  P (r|m ))
                 r<o


### Software

The entire project is written in python. To run it, you will need a copy of python 3.5, as well as numpy, matplotlib

### Results



The SLAM process consists of a number of steps. The goal of the process is to use the environment to update the position of the robot. Since the odometry of the robot (which gives the robots position) is often erroneous we cannot rely directly on the odometry. We can use laser scans of the environment to correct the position of the robot. This is accomplished by extracting features from the environment and reobserving when the robot moves around. An EKF (Extended Kalman Filter) is the heart of the SLAM process. It is responsible for updating where the robot thinks it is based on these features. These features are commonly called landmarks and will be explained along with the EKF in the next couple of chapters. The EKF keeps track of an estimate of the uncertainty in the robots position and also the uncertainty in these landmarks it has seen in the environment. An outline of the SLAM process is given below.

