Particle Filter Implementation

Nare Karapetyan

18 Feb 2017

1 Introduction

The following report discusses the implementation details of a Particle filter localization algorithm. The task is to implement initialization, propagation and update steps. The bag file provided from an actual run of a Turtlebot 2 robot moving through the environment was used for experiments and debugging. A known map of the environment is considered with an initial unknown pose of the robot (global localization).

2 Proposed Solution

2.1 Initialization

N particle filters were randomly distributed over the free space. ROS *RandomNumberGenerator* class was used to generate random values for all particles state parameters, e.g. x, y, θ.

```
x = Ran.uniformReal(x_min_, x_max_);
y = Ran.uniformReal(y_min_, y_max_);
theta = Ran.uniformReal(0, 2*M_PI);
```

2.2 Propogation

After a new odometry information was received, e.g. a motion was performed, all particles are updated according to the motion model. The linear and angular velocities are defined by calculating the distance that robot has travelled and heading change. The time stamp at each moment gives information how long the motion will take.

```
ros::Duration dur = stamp - old_stamp_;
double dt = dur.toSec();
double rot = (old_yaw_ - yaw);
double dist = sqrt(pow((old_x_ - x), 2) + pow((old_y_ - y), 2));

particle_filter_->move(dist/dt, rot/dt, dt);
```

2.3 Update

The update step was implemented additionally introducing *sense()* function, that estimates the sensor output for every particle. sense functions takes as an input the particle and the angle of the current sensor beam. Function incrementally builds a laser line until it heats the wall or the distance is greater than maximum distance. Additionally instead of processed sensors percentage sensors number is used to allow easier testing for different number of sensor rays. The average weight over the number of processed sensor readings is set as particles' new weight.

Weight calculation:

```
w += Sam_[i].w() * gaussianWeight(sens_read_value, \
                       sensor_accuracy_, sense(Sam_[i], angle, sensor_range));
```

Sensor reading for particles:

```
 while(!isOutsideGridFreespace(px) || dist <= sensor_range){
       ++step;
       next_point = PointGivenStartingPointAngleDistance(starting_point,\
              theta, step*resolution_);
       px = convertWorldPointToPixel(next_point.x, next_point.y);
       dist = EuclideanDistance(starting_point, next_point);
}
```

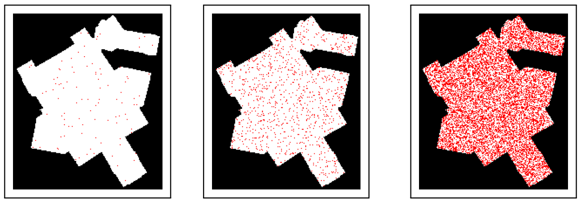
3 Experimental Results

The number of particles selected was 100, 1000 and 10000. The number of processed sensor rays were 1, 3 and 10.

For following set of experiments no significant change was observed in both performance.

Next sections will demonstrate initialization step and some excerpts from propagation and update phases.

3.1 Initialization



3.2 Propagation and Update in Action

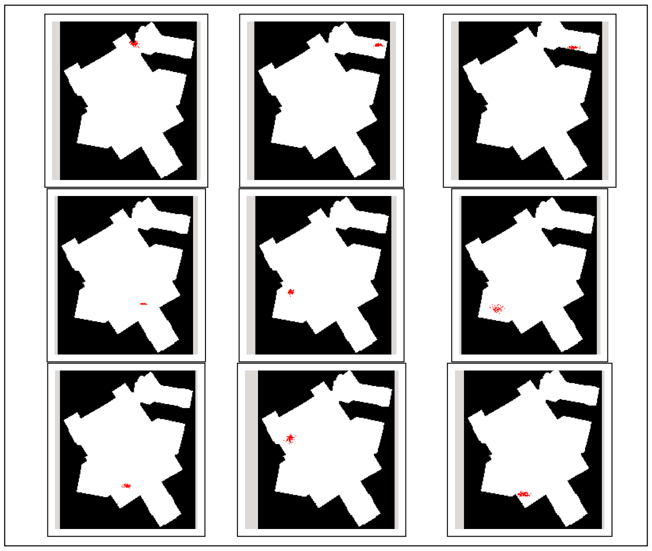


Figure 1: The following results are recorded for 1000 particles. First row are results from 1 sensor reading, second row for 3 sensor reading and finally the last one is for 10 rays.