PROBABILISTIC NAVIGATION TECHNIQUES FOR SWIMMING ROBOTS

Anwar Ahmad Quraishi
Supervisor: Yannick Morel
Introduction

Project Envirobot: Autonomous swimming robot for locating pollution sources in water bodies

Objective: Track regions of high concentration and possible sources

Goal of my project: Compare different probabilistic navigation techniques for the swimming robot
Outline

- Envirobot project
- Probabilistic navigation
- Bayes filter
- Extended Kalman Filter
- Particle filter
- Comparison
- Progress and outlook
Envirobot project

- Measure concentration as the robot moves
- Assess local concentration trends
- Estimate and move in the direction of highest gradient

Trajectories of the robot navigating a diffusion plume, moving towards areas of greater concentration
Probabilistic navigation

- Noisy sensors and actuators, hence motion not deterministic
- Accurate estimate of trajectory important
- Shape of real and estimated trajectory must be similar
Bayes filter

- Computes beliefs over robot’s state
- Models noise in sensors and actuators
- Recursive algorithm consisting of two steps
  - Predict new state from the prior state
  - Compute the new state by correcting prediction using sensor measurements

\[
\bar{p}(x_t) = \int p(x_t \mid x_{t-1}, u_t) \text{bel}(x_{t-1}) \, dx_{t-1}
\]

Motion model

\[
p(x_t) = \eta p(z_t \mid x_t) \text{bel}(x_t)
\]

Measurement model
Implementations of Bayes filter

- Measurement and motion model are defined differently in different implementations
- **Extended Kalman Filter** and **Particle filter** are compared
Extended Kalman filter

- Represent beliefs by a Gaussian distribution
- Linearize state transition and measurement functions

- Motion model

\[
x_k = g(x_{k-1}, u_k) + \varepsilon_k
\]

\[
p(x_k | u_k, x_{k-1}) \sim \mathcal{N}(g(x_{k-1}, u_k), R_k)
\]

- Measurement model

\[
z_k = h(x_k) + \delta_k
\]

\[
p(z_k | x_k) \sim \mathcal{N}(h(x_k), Q_k)
\]
Extended Kalman Filter

- At each time step $k$,

\[
\begin{align*}
\bar{\mu}_k &= g(\mu_{k-1}, u_k) \\
\bar{\Sigma}_k &= G_k \Sigma_{k-1} G_k^T + R_k \\
K_k &= \bar{\Sigma}_k H_k^T (H_k \bar{\Sigma}_k H_k^T + Q_k)^{-1} \\
\mu_k &= \bar{\mu}_k + K_k (z_k - h(\bar{\mu}_k)) \\
\Sigma_t &= (I - K_k H_k) \bar{\Sigma}_k
\end{align*}
\]
Extended Kalman Filter

Mean Navigation error: 3.70 ± 1.22m
Mean estimation error: 3.29 ± 1.70m
Extended Kalman Filter

Mean Navigation error: 3.15 ± 1.40m
Mean estimation error: 3.07 ± 1.50m
Extended Kalman Filter

- Impact of parameters

High motion error covariance
Low measurement error covariance

Low motion error covariance
High measurement error covariance
Particle filter

- Belief represented by a set of $m$ particles
- Particles are random state samples drawn from the belief distribution
- Higher density where probability is higher
- Can model arbitrary distributions, not just linear Gaussian
Particle filter

• Algorithm
  • At each time step $k$,
    • Propagate each particle $x_k^{[m]}$ forward using the motion model, $p(x_k | u_k, x_{k-1}^{[m]})$
    • Compute importance of each particle, $w_k^{[m]} = p(z_k | x_k^{[m]})$
    • Include particle $x_k^{[m]}$ in the new set with probability $\propto w_k^{[m]}$
Particle filter

- Two variations
- PF 1
  - Propagate the particles forward
  - Compute importance weights
  - Retain particles with higher weights and discard others (Resample)
- PF 2
  - Propagate the particles forward and incorporate new measurement
  - Compute importance weights
  - Resample
- Tested both variations, similar performance
Particle filter 2

Mean Navigation error: 3.70 ± 1.22m
Mean estimation error: 3.14 ± 1.35m
Particle filter 2

Mean Navigation error: 3.15 ± 1.40m
Mean estimation error: 2.22 ± 1.00m
Performance comparison

Mean Navigation error: 3.70 ± 1.22m
Mean estimation error of EKF: 3.29 ± 1.70m
Mean estimation error of PF: 3.14 ± 1.35m
Performance comparison

Mean Navigation error: $3.15 \pm 1.40m$
Mean estimation error of EKF: $3.07 \pm 1.50m$
Mean estimation error of PF: $2.22 \pm 1.00m$
## Progress and Outlook

<table>
<thead>
<tr>
<th>Task</th>
<th>EKF</th>
<th>PF 1</th>
<th>PF 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read literature, study different localization Methods</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation and test in post processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparative result study</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tune parameters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge into closed loop control code in C++</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test on the real robot in the pool (time allowing)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance comparison

• For each sampled position $x_k$,
  • Shift trajectory to superimpose $x_{k,real}$ and $x_{k,est}$
  $$\sum_{n=1}^{w} d(x_{k+i,real}, x_{k+i,est})$$
  • Calculate $d(.)$ is the Euclidean distance

• Average errors:
  • Kalman Filter:
  • FastSLAM 2: