Robot Navigation: Lecture 4

Lecture Notes

Victor Risager

December 28, 2023

1 Recap

Wheel odometry → diff. drive Visual odomemtry

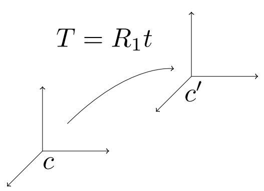

Figure 1: visualOdometry

We have the transform T:

$$T \begin{bmatrix} x \\ 1 \end{bmatrix} = y \begin{bmatrix} y \\ 1 \end{bmatrix}$$

When we compute a lot of tranformations T_i between C_1 and C_n, we need the intermediate transforms, to make sure that we keep track of the features.

1.1 Diff Drive

$$z = \begin{bmatrix} x \\ h \end{bmatrix}, \quad \frac{d}{dt} f = \begin{bmatrix} h \cdot \omega s \\ R_{\frac{\pi}{2}} \cdot h \omega d \end{bmatrix} \tag{1}$$

where

$$\omega s = \frac{\omega_r + \omega_l}{2} r$$

The hat is the command, while the actual input can be slightly different:

$$z_{n+1} = f(z_n, u_n) = f(\hat{z_n} + \Delta z_n, \hat{u_n} + \Delta u_n) \tag{2}$$

$$\hat{z_{n+1}} = f(\hat{z_n}, \hat{u_n}) \tag{3}$$

Then we take the first order taylor approximation to compute the actual input from the commanded input.

---

2 Probability

We have some probability distribution of our state:
Prior:

$$P(z)$$

The information on the probability distribution helps us:

$$0 \to P(z|o) = \frac{P(o|z)P(z)}{\sum_i P(o)}$$

Where $P(o|z)$ is the model of the sensor. o is the observation, and z is the state.
TODO: Add centering snippet with a raggedrigt term.

$$\boxed{\eta \cdot P(o|z)P(z) = P(z|o) = \frac{P(o|z) \cdot P(z)}{\sum_i P(o|z_i) \cdot P(z_i)}}$$

Note that we use the Total Probability theorem:

$$P(x) = \int_y ...$$

3 Localization

We can discretize our map by defining a grid in to cells or bins. Thus when assigning a system state, we assign a probability distribution to all of the different cells.

$$\sum_i P(z \in b_i) = 1$$

where b_i is bin (cell) i

$$P(z_{n+1} \in b_i) = \sum_j P(z_{n+1} \in b_i \wedge z_n \in b_j) \tag{4}$$

$\boxed{\wedge \text{ is called "wedge" and means "AND".}}$

4 Probabilistic Map-Based Localisation

It works in both known and unknown environments.
Instead of giving a single estimate of the pose of the robot, we give the distribution of all the robot poses.
Kalman always uses a gaussian distribution. If we are entirely certain of the position, we can imagine the distribution of the certainty is the *Dirac Delta* function, which is 100% probability, of a certain pose. i.e. the variance $\to 0$.

$$P(x) = \delta(x - a)$$

where $P(x) = 1$.
We improve belief by moving.
there are 3 different kinds of localisation problems:

- Position Tracking
- Global localisation
- Kidnapped robot problem

5 Filters

5.1 Bayes filters

for all x_t do:

$$\overline{x_t} = \int P(x_t|u_t, x_t - 1)... \tag{5}$$

5.2 Kidnapped robot problem

In this case there might go some time before the markov localisation problem can estimate the position.

5.3 Particle filters

We can use samples (particles) to represent the distribution.
This is done to increse computational effeciency, and we use them to simplify the number of states that the system can be in. In order to compute the validity of a particle, we use the $P(o|z)$ and plot in the state z and then it gives a distribution of the possible observations of the sensors. We can use the particles and just do a weighted average, and this would be the robot pose estimate.

$$P(x) = \int_0^y P(x)dx \int_0^y f(x)dx = F(x) \tag{6}$$

TODO: Add newlines to small hlines

Example:
we have a particle

$$z_{t+1} = f(z_t, u_t) = f(z_t, \hat{u_t} + \Delta u_t)$$

$$z_t^i$$

$$z_{t+1}^{i,j} = f(z_t^i, \hat{u_t} + \Delta^i u_t)$$

for $j \in 1..10$

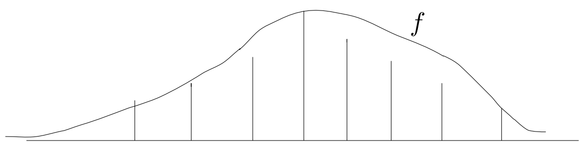

Figure 2: ParticleFilter

I have to add noise for the j variable. This means that for every particle, it forms new particles which have some noise in them, which is based on j.
Instead of just letting the number of particles grow, we only select the 10% of the particles, which are the best, this way we keep the number of particles the same across the itterations.
Two funcitions gives the new particle distribution $\rightarrow$ the particles, and the belief funciton.

Note:
Particle filters are best applied when we apply some new measurement.