Bicycle tracking in a village

Yaxi Xie

1 Bicycle tracking in a village

1.1 Task1

We can use the information from the map like the positions of some landmarks or houses. In this task, we have already known the accurate position of each house and the boundary of the map. Thus we can update the weights by testing if the particle is on road or not. If the particle is not on road, its weight will be updated as 0, which means that the probability of this particle existing should be 0, i.e. this particle should not exist. In this way, we can use the map information to control which state is the most probable one. My idea of using the information from map in details are as follows:

Firstly, we get the predicted weights and states.We know the state vector is:

$$X_s = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{1}$$

The state is predicted by the CV motion model:

$$x_k^{(i)} = Ax_{k-1}^{(i)} + q_{k-1} \tag{2}$$

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

And the weights are predicted by measurement model:

$$W_k^{(i)} \propto W_{k-1}^{(i)} p(y_k|x_k^{(i)}) \tag{4}$$

$p(y_k|x_k^{(i)})$ is the likelihood of this particle and follows the equation as follows:

$$y_k = Hx_k + r_k \tag{5}$$

$$y_k = \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 & \sigma_r^2 \\ \sigma_r^2 & 0 \end{bmatrix} \tag{6}$$

Then for each particle, we do a judgement of whether the position of the particle is on road or not by introducing the coding part of *isOnroad.m* which contain the information of map. If the particle is not on road, its weights will be updated as 0. Finally, we normalized the total weight. In conclusion, we first predict the weight and states by using the motion and measurement model then update it using the information of map.

And we can alternatively using the information of map by incorporating it into the measurement or motion model. For example, we can introduce it into the measurement model as designing the new one as :

$$y_k = \begin{bmatrix} v_x \\ v_y \\ u = isOnroad(p_x, p_y) \end{bmatrix} + R_k \tag{7}$$

$u = isOnroad(p_x, p_y)$ is the critical result: $0or1$.In this way, we can also assign the weights of the outliers as 0. From my perspective, we can also include the map information as part of the motion model but checking the position of the particle states is much easier.

Figure 1.1: The estimated PF-track and true track of the bicycle - Known prior

1.2 Task2

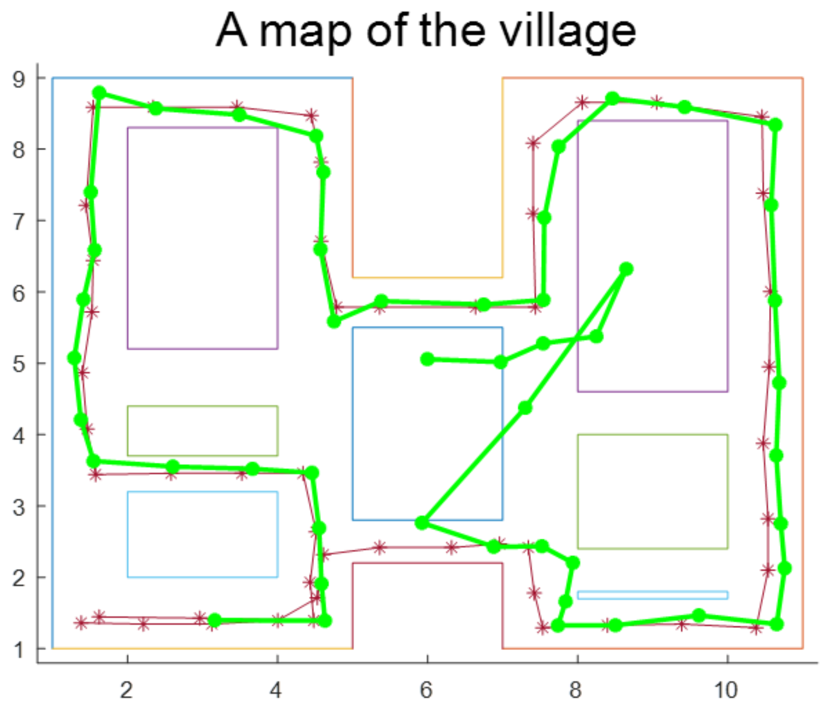

Figure 1.2: The estimated PF-track and true track of the bicycle - Unknown prior

After tuning the measurement and motion noise, the good estimated track is obtained. In this case, the tuned noise covariance matrix-es are as follows:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \tag{8}$$

And for each time step k, the number of the particles is $N = 20000$.The prior velocity is 0. Since the SIR (Sequential Importance Resampling) algorithm performs better than SIS, we use the SIR method to do filtering. Figure 1.1 and 1.2 illustrates the result when the initial position of the bicycle is known or unknown.

In the first scenario, it is assumed that the true initial position of the bicycle is known. From figure 1.1, we can see that the PF filter can always locate the

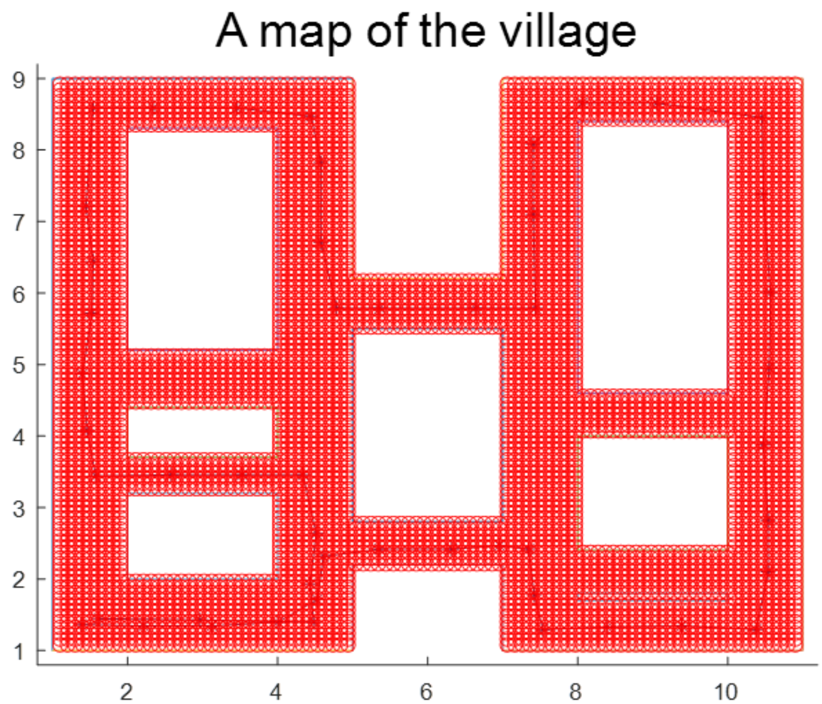

Figure 1.3: The prior state of the particles- generated by uniform distribution around the map (is On road)

accurate position of the bicycle and the averaged Mean Squared error is about $MSE_x = 0.449$ and $MSE_y = 0.467$. So PF filter can locate the position as soon as it starts to work when the prior is known.

In the second scenario, it is assumed that the prior state is unknown. From figure 1.2, we can see that the PF filter can still locate the accurate position of the bicycle and the averaged Mean Squared error is about $MSE_x = 1.449$ and $MSE_y = 1.467$. But compared with the first scenario, PF filter needs several time steps to track the trail. This is reasonable because we generate uniform distributed particles on the map as the prior in this case, which is shown in the figure 1.3. So PF filter requires time to update the weights and focus on the most likely particles. The velocity measurements y_k and the information of the map enable the PF filter to track the true position even though the prior is unknown. As mentioned before, we use the map information to check if the particles are on road or not. If it's not on road, its weight will be update as 0 which helps PF filter to rule out those large chunks of unused particles as time goes by. Additionally, the resampling step also helps to delete those particles with a smaller weightv(probability), which has been interpreted before.