[Robotique mobile]Particle Filter localization

yu-tianchi

January 2021

1 Introduction

In this practical work, we will work on robot localization using a Particle Filter. We need to complete the code to make it possible to simulate a robot moving on a given trajectory in an environment made up of point landmarks. It implements a simple particle filtering method using the perception of the direction and distance of these point landmarks.

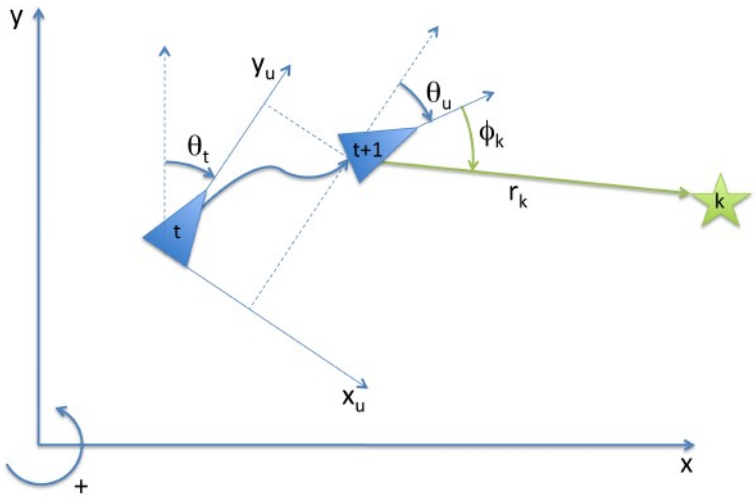


Figure 1: Model:Notations for the motion and observation models used in the code

2 Questions

2.1 Question 1

Parameters *Change the different parameters of the particle filter to evaluate their influence on the performance of the filter (speed, precision, stability, performance of recovery after sensor failure at time 400 ....).*

Answer:

1) The number of particles (nParticles)

Here we test several experiments with nParticles = 10, 100, 500 and 1000, the results are as follows:

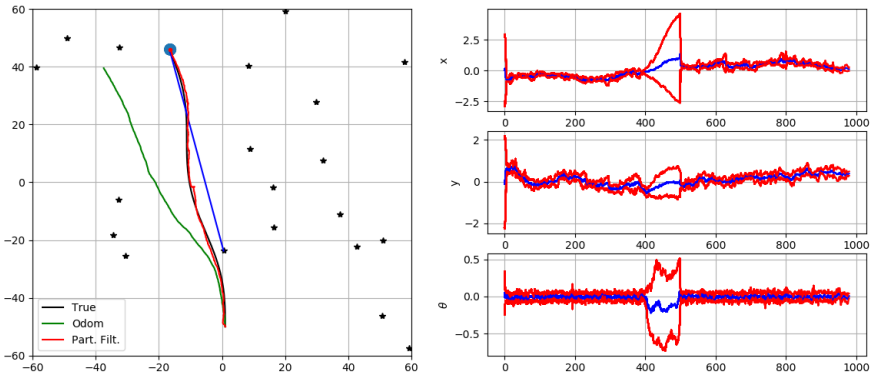


Figure 2: Particle filter with 10 particles

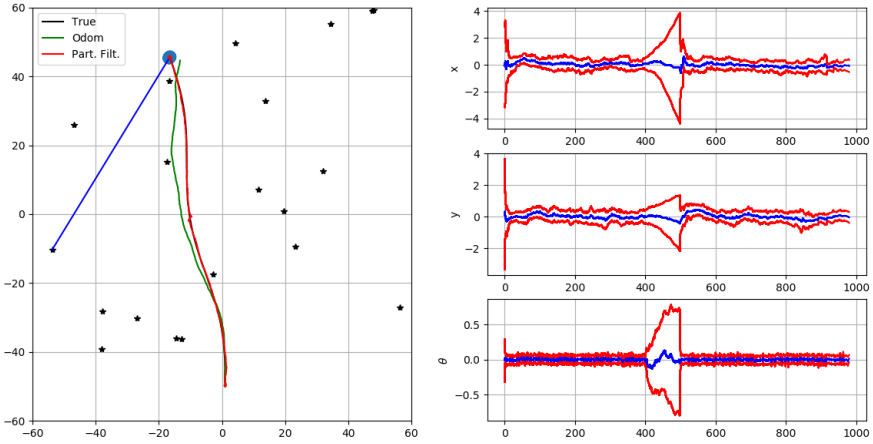


Figure 3: Particle filter with 100 particles

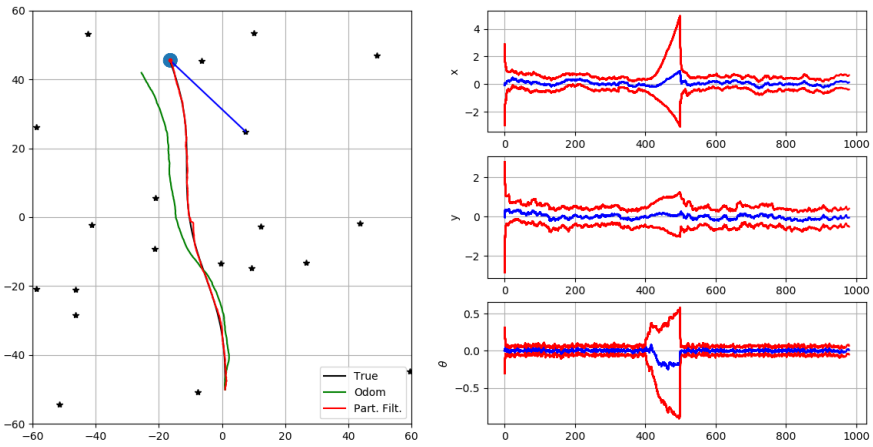


Figure 4: Particle filter with 500 particles

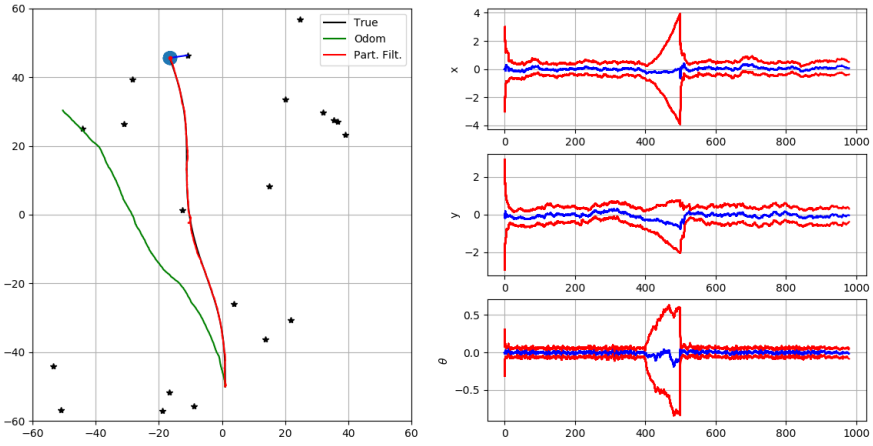


Figure 5: Particle filter with 1000 particles

Speed

10 particles: Running time is 13.217349529266357

100 particles: Running time is 15.584003925323486

500 particles: Running time is 35.75776410102844

1000 particles: Running time is 63.99188232421875

The large number of particles will effect the efficiency of the calculation, because particle filter need to take the measurements from every particles. The result is that, more particles means slower speed.

Precision & Stability

Apparently, with more particles, the result will be preciser and more correct. If the filter uses more particles to estimate the localization, the estimated trajectory is more like the true one, and the variances of position and orientation are more stable. Because if there are only several particles, the observation by the landmarks and the estimated result are easily be effected by imprecise/unimportant particles.

Recovery

During the filter processing, there is a failure of sensors, and the robot loses the observations from the sensors. No matter how many particles used for the particle filter, the influence of the failure are similar.

Actually the recovery ability depends on the landmarks after the failure of sensors. If the robot can collect the information of the landmarks soon, it will get back to the normal situation and diverge. In this point of view, more particles may have a better effect.

2) The initial variance around the true position for particle creation (np.diag([1,1,0.1]) around line 160)

In the np.diag([1,1,0.1]), the first two arguments are corresponding to x and y position, the last argument is for the orientation.

Change the initial variance around the true position to the 2 times and 10 times than the original one. We have the results:

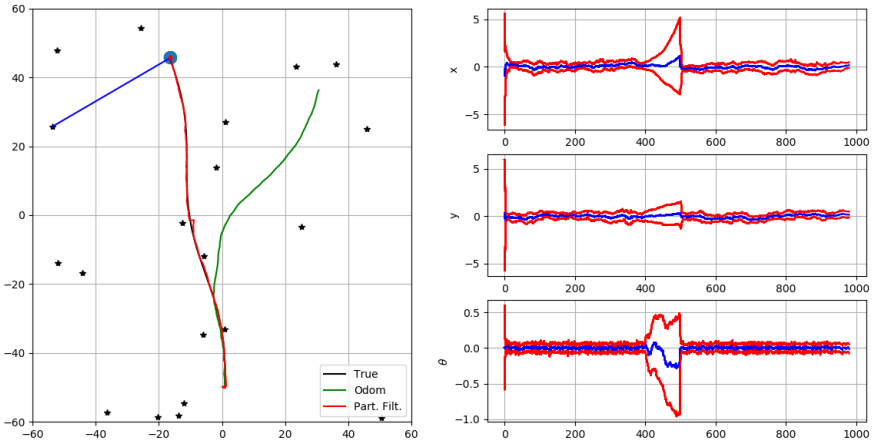


Figure 6: Particle filter with 2 times initial variance

Speed 1(in previous part): Running time is 15.584003925323486

2 : Running time is 15.374964237213135

10 : Running time is 14.727152585983276

There is no big difference about the speed.

Precision Stability

Apparently, the initial variance around the true position for particle creation effects the precision and stability. By the trajectory, if the initial variance is too large, the robot is harder to find the correct trajectory at first. But is will catch the line finally, with lower precision. Also the stability is worse than the small value.

Recovery

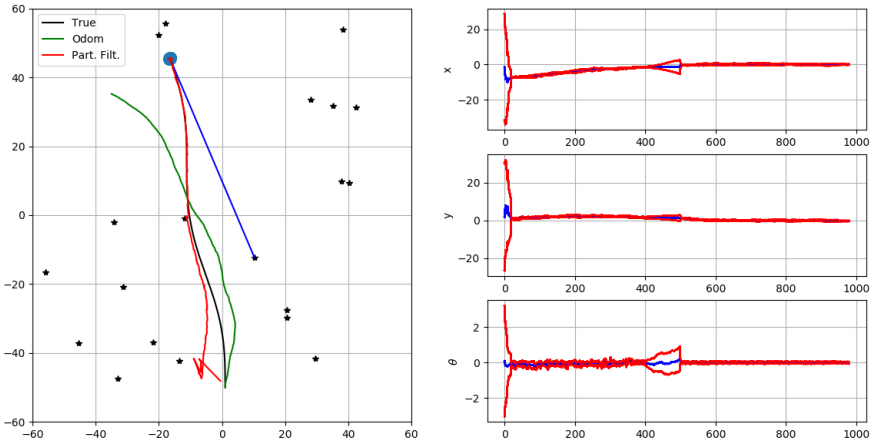
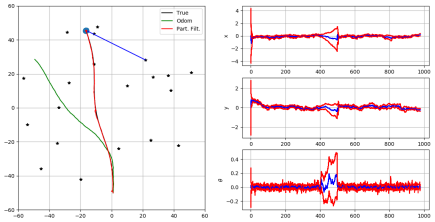


Figure 7: Particle filter with 10 times initial variance

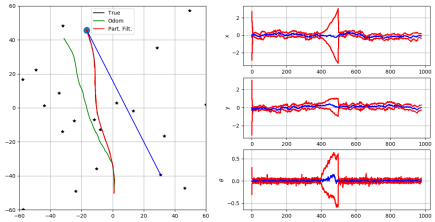
It is hard to say which one performs better when the failure of sensor arrives. If the initial variance is too large, the robot may even not find the correct trajectory after the recovery. In general, there is no big difference in this ability.

3) The noise of the perception and motion models used for the particle filter (UEst and REst) with respect to the simulation noise (smaller, equal or bigger than UTrue and RTrue).

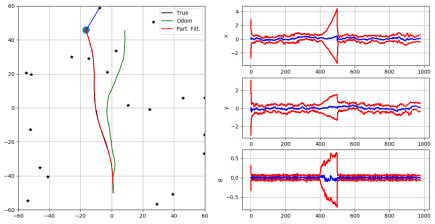
By now, the estimated results are based on the truth that the noise of the perception and motion models are bigger (2 times bigger) than the simulation noise. Now, we simulate the process by the smaller noise and the same noise.



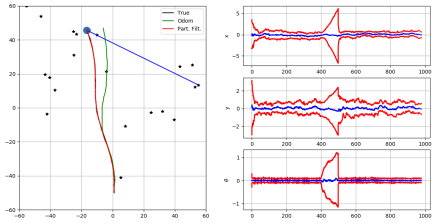
(a) pic1.smaller noise



(b) pic2.same noise



(c) pic2.(2 times)bigger noise



(d) pic2.(5 times)bigger noise

Figure 8: Particle filter localization with different noise

Speed

The noises have no influence on speed.

Precision & Stability

With a smaller noise, the variance is much smaller, however it seems to be harder to find the trajectory at the beginning, and it is not accurate according to the trajectory. Also, the stability is worse, if the noise is too small, because the variation of variance is more obvious.

Recovery

During the failure of sensor, the robot with a big noise performs better.

In my opinion, it is better to have the same noise with the simulation noise. The big noise means good tolerance and bad variance, the small noise means smaller variance but doesn't mean the better performance.

2.2 Question 2

Global Localization *Use the particles initialization without any knowledge of the initial position (uncomment xParticles = 120 * np.random.rand(3, nParticles)-60 around line 163). Choose the filter parameters to reach a correct performance. How do you compare the performances with the situation in the previous question ?*

Answer:

Imaging that there is no knowledge of the initial position, the problem would be more difficult to realize, so apparently, we need more particles for this particle filter. We test for 100 particles, 1000 particles, 2000 particles and 5000 particles. (In the following test, we always keep the noise and variance as the original settings)

For the first two conditions, the robot loses its mind and the estimated trajectory has a huge distance with the correct one. Even though, 1000 particles help to optimise the variance and stability, the accuracy is still bad.

With 2000 particles, the situation has an obvious improvement. At first, the robot are not at the right position. However, after several steps of simulation, it finds the right localization. And the performance of variance is much better. 5000 particles give us a much more correct and preciser result, of course, with the cost of time.

Point of view in speed:

100: Running time is 14.569238424301147

1000: Running time is 71.05322122573853

2000: Running time is 141.49342155456543

5000: Running time is 255.23547995623123

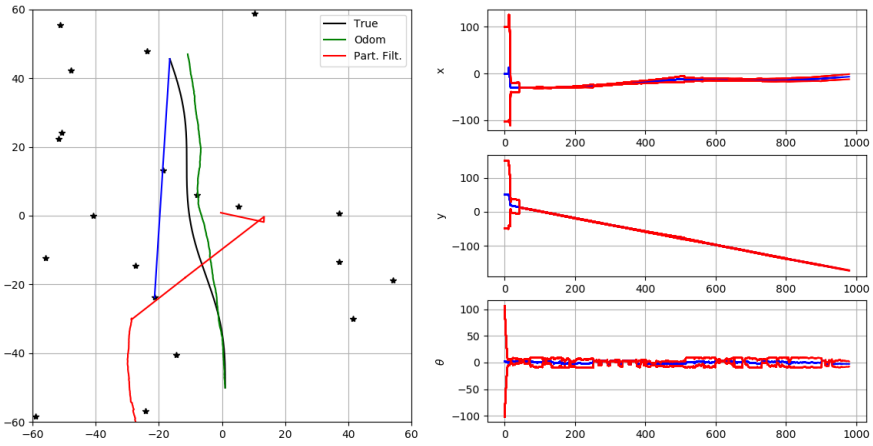


Figure 9: Global Localization with 100 particles

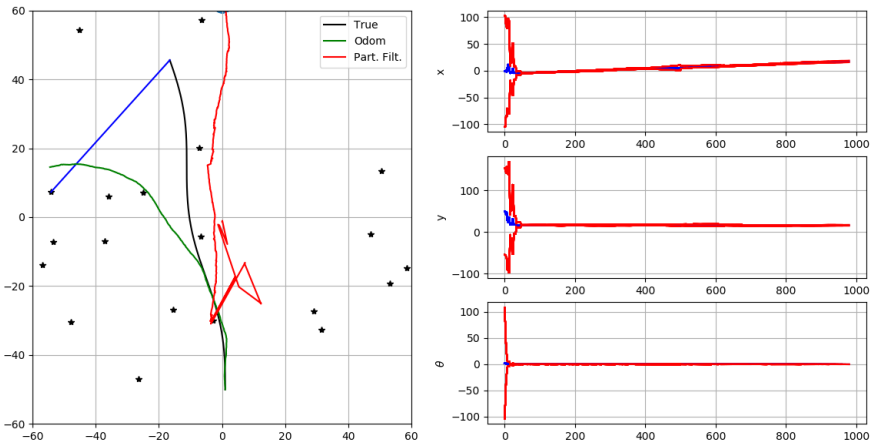


Figure 10: Global Localization with 1000 particles

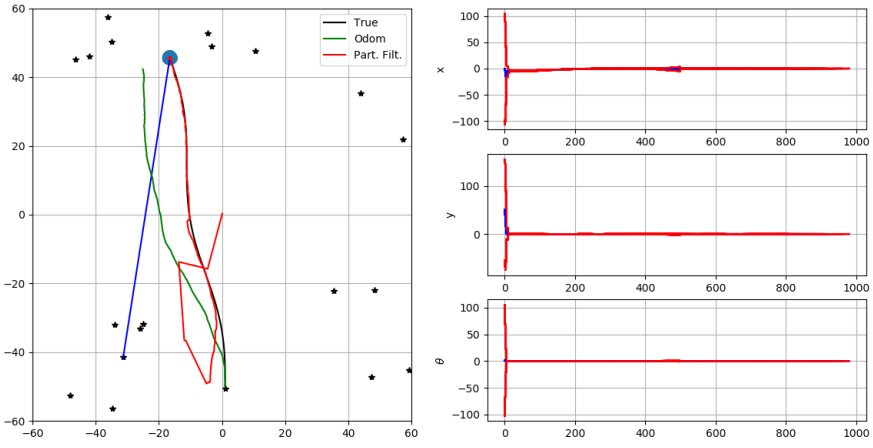


Figure 11: Global Localization with 2000 particles

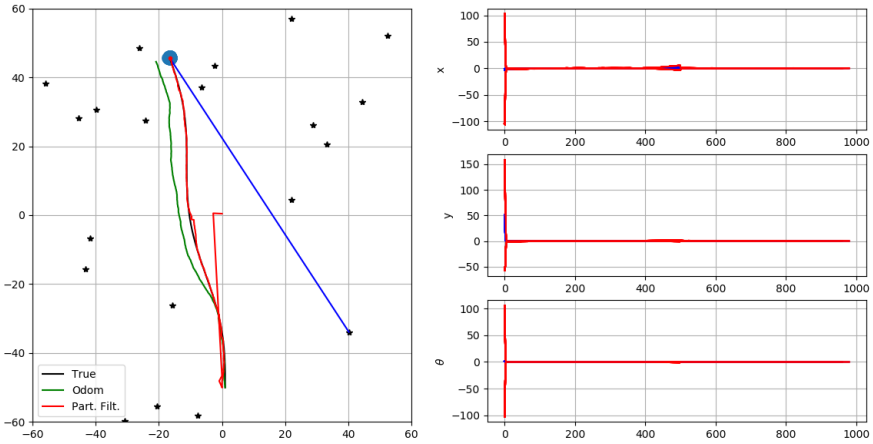


Figure 12: Global Localization with 5000 particles

As we can see, the number of particles is the most important parameter for the global localization. Even we have no knowledge of initial position, we can still get a good estimation with many particles by scarifying the computation resource. Also, we can optimise the result by adjust other arguments, for example the noise of perception and motion models.

2.3 Question 3

Sensor Resetting Localization *Modify the re-sampling function to use the "Sensor Resetting Localization" method. You must replace a certain number of particles with particles drawn according to the likelihood of the observations.*

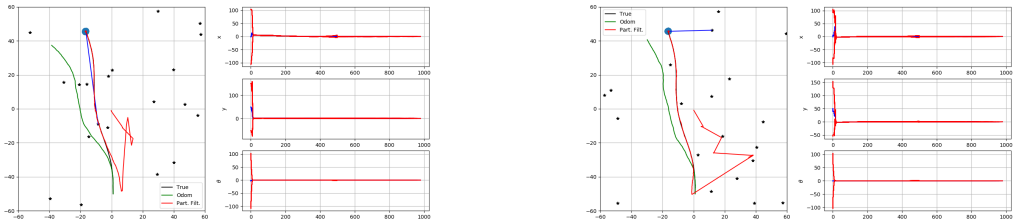
The observation model of the direction and the distance of a point landmark allows to generate new particles on a circle around the perceived landmark. Note that this is not necessarily possible or practical with all perception models.

These new particles should only be generated in the map area (-60 $<$ x $<$ 60;-60 $<$ y $<$ 60) and only when the localization is poorly estimated, i.e. when the sum of the likelihoods of the particles is below a threshold.

Test this method on the global localization problem and show how it can improve the performances.

Answer(The code is implemented in a new file: ParticleFilter_Localization_SRL.py):

Here we use the Sensor Resetting Localization method to optimize our model. The process of generate/substitute particles should be completed during the re-sampling process. The number of new samples is calculated by $ns = (1 - \frac{Averagesampleprobability}{threshold}) * NumberofallParticles$, where the threshold I set as 0.05. When the ns is positive, we should operate the substitution as following steps: Choose ns particles randomly, create new particles according to the position of robot, which should be formed in the circle around the landmark and keep the orientation and distance. Here are the results for 800 particles and 1000 particles:



(a) pic1.SRL 800 Particles

(b) pic2.SRL 1000 Particles

Figure 13: Particle Filter with SRL algorithm(with threshold = 0.05)

Point of view in speed:

800: Running time is 57.72269916534424

1000 : Running time is 72.38915252685547

Compared with the global localization in the Question 2, the performance of SRL algorithm is obvious. For example, Particle Filter with global localisation need 2000 particles (at least more than 1000 particles) to get good trajectory after several mussy steps, however SRL helps the global localisation to be easier to realised. With only 800 particles, the performance is similar with 2000 particles before, according to the trajectory and variance.

Of course, at first the error of localisation is large, with the motion and perception of robot, the error decreases faster and faster. It can use fewer samples and handle larger errors in modelling. One of the advantages of SRL is that fewer samples can be used without sacrificing much accuracy. This is possible in part because it is more efficient when globally localizing.

SRL can handle larger systematic errors in movement because once enough error has accumulated, SRL will replace the current estimate of the robot's location with one based on the sensor readings, effectively re- setting the localization.