Intelligent Robotics

Report

Assignment 1 - Particle Filter Localization

Oct 2021

Group 02

2026786 Eunji Kwak

2100013 Sungin Hwang

1906868 Woohyeon Jo

2018963 Xuemeng Li

1940922 Yujia Hu

Introduction

Over the past two weeks, we worked to implement a particle filter algorithm for localization. The aim of the task is to understand the working of a particle filter localization algorithm and analyse the node within our understanding.

We collaborated as a team and we started by developing the node to implement the algorithm. Then, we try to figure out the difference between our implementation and AMCL implementation. Furthermore, we found the advantages and limitations of the particle filter algorithm.

Particle Filter

Before we start the task, the most important thing we have to do is understand the concepts of the particle filter fully. Basically, the particle filter is the main algorithm which computes importance sampling approximations recursively. Particle filters, also known as sequential Monte Carlo, generate n new samples that approximate the robot's position after the motion command when the robot moves. And each sample is generated by randomly drawing a sample from the previously computed sample set, with likelihood determined by their p-values.

With the particle filter algorithm, we have to focus on finding an approximate representation of a complex model. In other words, the particle filter helps us to find any arbitrary pdf rather than an exact representation of a simplified model like Gaussians.

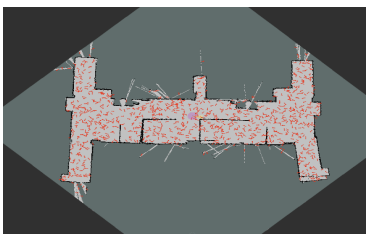
- Experiment

  In order to complete the task, we divided our attempt into three steps, prediction, innovation and resampling.

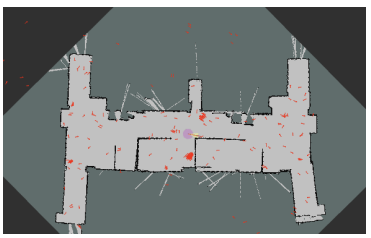
  For the first method, prediction, we have tried to combine a Gaussian random distribution to generate the random particles and predict the position of the robot. All the pose arrays are the possible location of the robot. We added weights w to each pose array and all weights remain the same and the result of this movement is added to the state. Assuming we know the value of the noise, we add a random number to the result. An initial set of pose arrays is first obtained that obeys a uniform distribution, and the initial pose arrays arrive at another set of pose arrays under the action of motion and noise.

  Next method is the innovation step. For the innovation, an array was created to determine the accuracy of the pose array by comparing the magnitude of the weights. The second set of pose arrays still obeys a uniform distribution. At this point, we first append weight to each particle with the sensor model and we obtain a measurement pose array and assign probability values to each group of the predicted pose array. Then we re-arranged them in a descending order by indexes. The probability values are assigned by the weights.

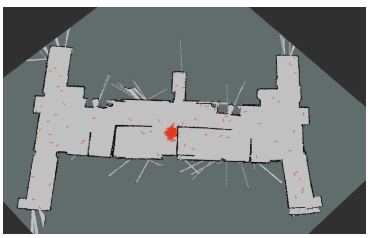
  The last method is the resampling step. In the resampling step, the pose arrays are re-selected from the set of predicted pose arrays by probability. We calculated the mean in four directions, and a further mean is divided by all particle numbers. Then set an array with current values in four directions. At this point the mean value of the points is the final result.



<Figure1 - Prediction>

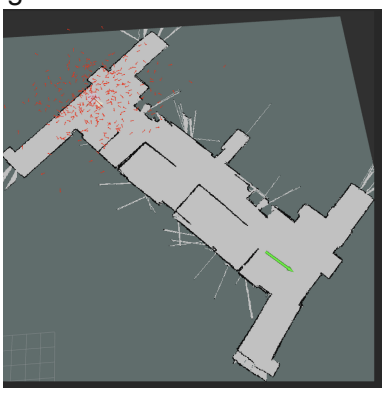


<Figure2 - Innovation>

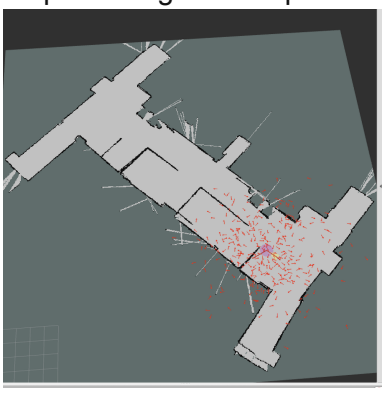


<Figure3 - Resampling>

Figures 1,2 and 3 show steps how we predicted the position of the robot. These figures are estimated with the simpath1.bag and simpath2.bag files.



<Figure4 - Estimating poses manually>



<Figure5 - Random poses of particles around the estimation>

Especially, in figures 4 and 5, it shows the random poses of particles around what we estimated based on the Gaussian random distribution.

- Limitations

  Generally, Monte Carlo Localization is used to solve the global localization problem. However, it is hard to recover from specific situations - for example, kidnapped robot problem. In terms of robotics, the kidnapped robot problem is defined as the situation when the robot is moved to an arbitrary position without update. This is because the particles are only “survived” with one “pose”. In other words, when the robot’s location is predicted, the particles in places where the robot is less likely to be present gradually disappear. After all, if this pose is incorrect, the algorithm cannot recover.

  Furthermore, we realized that the prediction sample propagated by particle filter does not consider the last state value, so compared the sample by this algorithm with the true posterior distribution the deviation is greater. Especially when prognostic data appear at the coda of transferred probability distribution or the likelihood function is very tight (the peak value), particle filters may expire.

- Advantages

  Even though the particle filter has some limitations, the algorithm has advantages based on its features. It estimates the distribution of one potential variable at a time, rather than all at once. Also, a set of samples with certain weights is produced, rather than samples with no weights.

  Besides, the particle filtering algorithm has excellent accuracy and immunity to interference, and is able to parallel computing and achieve great real-time performance.

  Moreover, because the particle filter uses a non-parametric Bayes filter, it can represent not only a normal distribution but also a distribution that does not follow any particular distribution.

Discussions

After writing our own node for localization, we have wrestled with the question of how MCL can be augmented for AMCL and we have discussed the relationship between MCL and AMCL. We found out how PF works in MCL and AMCL, and we also found out what algorithm difference there is between the two.

The most important difference between them would be the process of resampling. For MCL, if the robot is moved to a different location at this point, all the particles that were correct are no longer correct, and there are no particles elsewhere. Since there are no particles to approximate the true position of the robot, the relocation will fail. However, AMCL will keep some of the random particles, so there will still be particles that slowly approximate the real position of the robot. In other words, AMCL tracks the average of likelihood short-term and long-term. And then, if short-term likelihood is better or equal to long-term likelihood, random samples are not generated. But, if short-term likelihood is worse than long-term likelihood, random samples are added in proportion to that value to correct the current location.

Therefore, we executed the command "rosrun amcl amcl scan:=base_scan" in the IntelligentRobotics_SoftwareSetup_Notes_2021.pdf. The simulated robot in Rviz was made to handle the kidnapped robot problem by using the amcl algorithm library.

Nevertheless, we think that the AMCL algorithm does not completely solve the kidnapped robot problem. This is because it is difficult to correct accumulated errors. In particular, errors are likely to accumulate in an experimental environment with few characteristics, making it almost impossible to solve this problem if the robot deviates from the expected position.

Conclusion

In conclusion, we can say that we worked cohesively as a team. Looking back over the task, this is a challenge for us. It tested not only the understanding of what we had learnt in the last few weeks but also the ability of teamwork. But, all of the members have worked hard over the two weeks within both online and face-to-face meetings. And finally we can implement the particle filter algorithm for localization.

During the task, we faced lots of errors related to code and time schedule, but we actively communicated with each team member and sought useful key points ground in our discussions. Therefores, we have completed this task successfully.

Code repository: https://github.com/ezi-sab/IR_Group02/tree/master