Cooperative Multi-agent Localization

Richard Lin
Department of Computer Science
Stanford University
yifengl@stanford.edu

Jerry Meng
Department of Computer Science
Stanford University
zmeng90@stanford.edu

Abstract

We tackle the problem of localizing several agents on a known world map. Agents would have simulated LIDAR to detect environmental obstacles as well as simulated radio to cooperate with other agents. We implemented a variant of particle filtering, and modified it so that it can leverage the additional information provided by other agents to localize more accurately and more quickly.

1 Introduction

The problem of localization is a central one in practical computing and robotics. Modern robots already do well with the development of technologies such as advanced LIDAR, and computer vision methods like SLAM. However, there are several scenarios where robot localization may fail. For example, robots navigation in large open spaces is difficult because LIDAR scans may not be able to detect walls that are far away and the robot would not have any observations to update its position. If the environment has many similar locations, given the observation, the robot cannot distinguish between the many possible locations.

We attempt to tackle these localization challenges in an environment where there are other agents also trying to localize themselves. When there are multiple agents in the environment, the agents can help each other with localization. This is useful in scenarios like warehouse robotics where multiple robots operate independently but still need to localize in the same environment.

In our approach, we combine particle filter and triangulation to localize the agents. We assume each agent has a LIDAR with limited range and also a radio that can communicate with other agents within a certain range. Using the radio, an agent can broadcast its current belief position as well as determine its distance relative to other agents. The directions of other agents remain unknown. Using the distance and belief positions of other agents, we are able to update the agents' belief positions more accurately and quickly converge to the correct location.

2 Related Work

Localization problems can be separated to two categories; finding the global position of agents given a-priori information and without a-priori information [1]. Our localization problem is in the latter category, where at the beginning of the localization, there is no prior belief of where the agents are on the map.

One efficient localization technique for multi-agents with a-priori is triangulation. Triangulation helps determine an agent's position when there are at least three landmarks with known locations on the map. The agent can estimate its relative position on the map using the landmark's position as well as the agent's current angle relative to each of the landmarks. While triangulation can be used without a-priori, it may have problems with correctly identifying the landmarks[2]. In outdoor scenarios, triangulation is used when the agent has a GPS and the GPS signal is temporarily lost. So given the last known location, the agent can re-localize using known landmarks on the map. There

was research in this area where two landmarks were used and the third landmark is the distance to another agent[3]. In our approach, we use all agents within range as potential landmarks.

Common localization methods for multi-agent without a-priori include Extended Kalman Filter and Particle Filter. Particle Filter has less computational complexity since it can work with any particle distribution. In [4], agents receive other agents' distance and bearing, so the relative positions of nearby agents are known, and the belief states are updated for all agents together. This reduces the number of particles required to localize all agents, but it also may converge to a steady-state error position. Since the agents need to receive each other's direction, it may be necessary that there are no obstacles between the agents. This is not a requirement for our approach since we do not care about the bearing. The approach that we propose also does not converge to a steady-state error.

3 Model

Our model is a Unity simulated environment. Unity is a 3D game engine with much built-in physics simulation, and we built our environment, agents, and particles on the platform. The world is a 2D continuous plane, enclosed by walls which together consist of the terrain. Each agent is represented by a circle with a diameter of 1 unit and is spawned within this enclosing. The agent would move within this environment with a random force applied every simulation update.

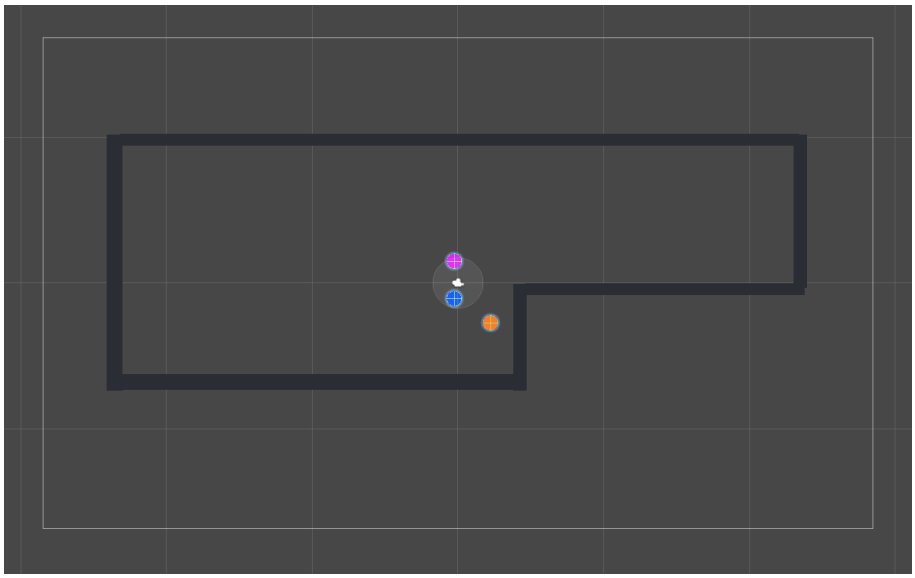

Figure 1:
Unity simulated environment with three agents.

LIDAR was simulated using Unity's physics raycast. From the agent, it would emanate rays, which are vectors that detect physics collisions, and we programmed the LIDAR simulation such that if there is a collision within distance d, the LIDAR's reading would return the distance to collision for that one ray, and if not, there would be no reading. The LIDAR system projects rays at 20-degree intervals, a hyperparameter that worked well, which meant that radial raycasts would happen 18 times every LIDAR update.

Particles for Particle Filter are simulated as tiny blocks with known global positions. Each particle has the functionality of a LIDAR so we can simulate the known observation of each particle.

The radio used to estimate the agent-to-agent distance is simulated without noise. If two agents are within range r, then they can obtain the actual distance between each other. The distance estimation does not contain noise because the estimation in reality would be accurate enough that it shouldn't affect the algorithm [5]. The added noise in particle sampling should cover any noise in the radio estimation. Through the radio, agents also receive where neighboring agents' belief positions.

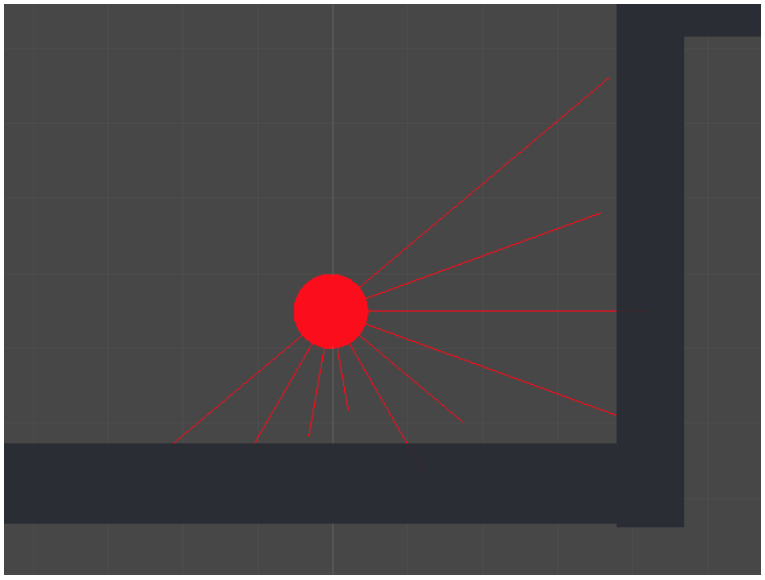

Figure 2:
Red agent receiving LIDAR feedback.

4 Approach

4.1 Naive Particle Filter with Puffing

We implemented particle filtering as the primary way for agents to localize. Initially, the number of particles are set to the world size as defined by the enclosing walls of the environment and the particles are uniformly distributed. As LIDAR and radio information came in for each particle on simulation update, the particles are weighted based on how likely it is for the agent to be near the particle's position. The weight is equivalent to:

$$P(s'|o,a,b) \propto O(o|s',a,b)P(s'|a,b) \propto O(o|s',a)$$

The particle's position represent s, the agent's previous position. Action a is the displacement of the agent since last update. s' is computed by adding s with a. Because the particle weights are relative to each other, the actual weight computation is simplified to be the percentage of particle rays that match with the agent LIDAR rays.

Once we have the weights set, we resample the particles according to their weight. Some Gaussian noise are added to the new particles to potentially get better position estimates of the agent's current state.

To help with recovery from incorrect localization, we introduce a technique called Puffing. When the agent has a lot of observations of the environment, but none of the particles are close to matching this observation, the noise is increased drastically during resampling so that particles will spread out more and search for the correct location. A cluster of particles would appear to puff up, hence the naming. This method is most obvious when the agent is travelling along an edge in the world toward a corner while most particles are near a different edge in the world. Once the agent enters the corner it would receive more observations than it would on the edge, so the particles would spread out and look for that corner. The amount of noise amplification is a hyperparameter that should not be too large. If the noise is so large that all the particles suddenly span the entire world, it would get out of the incorrect localization but then the previous localization efforts would also be in vain. It would be equivalent to restarting the localization. The noise increase only happens when the agent receives more observations, since an agent travelling in open space may already be localized, we do not want to spread out the particles.

4.2 Multi-agent Circular Search

After this initial resampling, we added in the multi-agent aspect. An agent can receive other nearby agents' distance from itself without knowing the direction through the radio. The radio can also inform the agent of other agents' belief positions. The belief position of an agent is defined as a weighted position sum of all the particles for that agent. When the particles are spread out at multiple locations, the belief position representation is not accurate. We have tried to make the belief position to be the location of the largest particle cluster, but it was too expensive to compute these clusters.

If there were other agents nearby our agent, our agent would first pick out a set of low-weight particles – the ones that will likely be resampled out of existence on the next simulation update – to be re-located based on the radio information from other agents nearby. Assume there are n neighboring agents to our agent, the low-weight particles would be evenly distributed into n arrays.

For each array, we would use the neighbor agent's belief position as the center and its distance as the radius to evenly distribute the particles on a circle, with the hope that the neighbor agent is localized enough that some particles on the circle will grow to have high weight and repopulate near the actual position of our agent. When multiple circles intersect with each other, the intersecting areas would have more particles, serving as a proxy for raising the weights of that intersection.

The benefit of creating such circles is that if two agents are physically close to each other but one of them is localized incorrectly and their belief positions are far apart, the incorrectly localized agent will have a circle of particles around the correctly localized agent, and therefore moving particles closer to it's actual location. Given observations, the particle circle can repopulate and correct a far incorrect localization belief.

4.3 Multi-agent Triangulation

When there are multiple neighboring agents, the particle circles may intersect with each other, and assuming the neighboring agents are correctly localized, our agent's actual position most likely lies on one of the intersections. When there are two neighbor agents, we may have at most two intersections. When we have three or more agents, there's a chance that we can triangulate the agent's position and get one intersection, or we may end up with multiple intersections if the other agents' belief positions are not accurate. We analytically compute the intersections based the belief positions and distances. Once we have the intersections, we then evenly re-locate the low-weight particles to the intersecting areas.

This approach reduces the number of particles wasted on low probability positions of the particle circle. It also allows an agent with no observations to be localized with help of other agents. If our agent is in open space with no observation to help localize, but there are three correctly localized neighbors with observations, the three neighbors can help triangulate our agent's position in open space.

5 Results and Discussion

5.1 Evaluation Metric

The approaches are evaluated based on the time that is required for all agents to correctly localize on the map. An agent is correctly localized if 70% of its particles are within 2 units of the agent, see Figure[3].

This metric is derived via visual inspection. We do not need exact localization of the agents to consider them localized since precise localization still relies on having good observations of the environment. For our approach we care more about how quickly the particles converge on the agents' approximate positions so that once observations are available, the agent can quickly localize to the correct position. The belief position is not used as a part of evaluation because as discussed earlier, the belief position doesn't represent the particles' beliefs as closely as it should when the particles are spread out. The time is recorded once all the agents are localized.

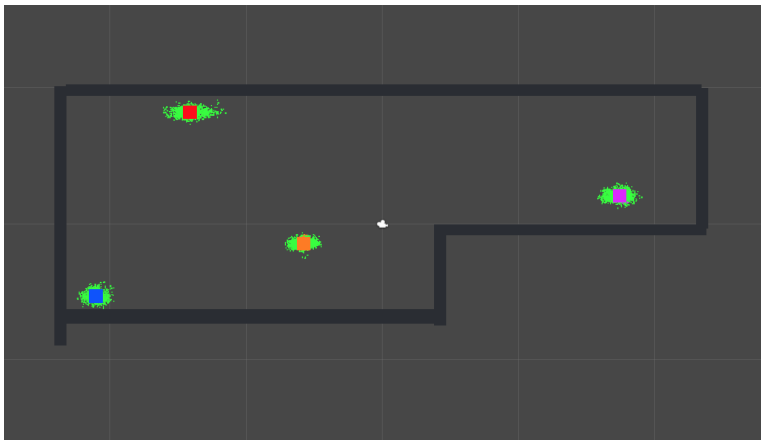

Figure 3:
All agents localized.

5.2 Results

We ran our tests with 4 agents on all three approaches. In the figures, particles are green, agents are colored circles and belief positions are colored squares matching the color of their agents. The times to localize all 4 agents on the given map is the following:

Naive approach: 81.68 seconds to localize.
Figure[4] shows that particles are spread out during the first few updates and for agents in open area (red and purple agent), the particles took a long time to converge since there were no observations.

Circular search: 19.2 seconds to localize.
Figure[5] shows that when agents are near each other, the particles are not as spread out and all the low weight particles are located on the circles where the agents are likely to be. The particles that remain spread out belong to the red agent where there are no observations or other nearby agents.

Triangulation: 11.47 seconds to localize.
Figure[6] shows that there are roughly three large clusters of particles near the purple, orange and blue agents. The cluster of points more accurately estimate the the agents' position compared to circular search since circular search still relies on observations to redistribute the circle. The particles belonging to the red agent are still spread out.

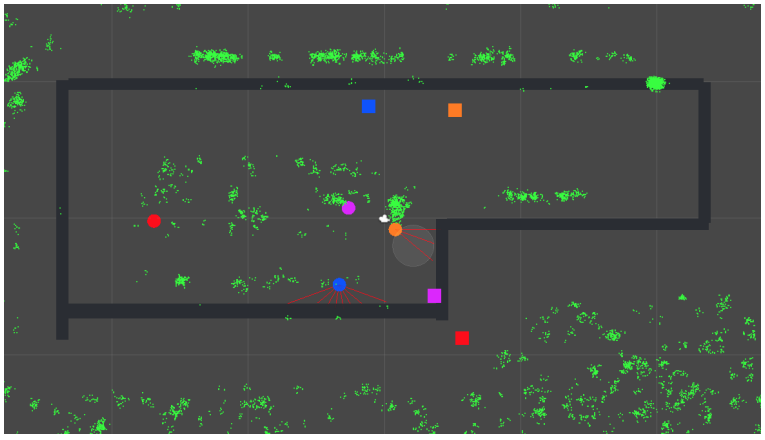

Figure 4:
Naive approach on first few updates.

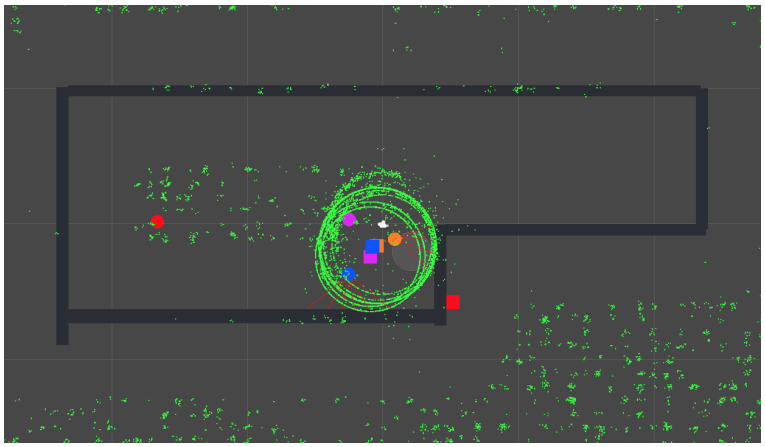

Figure 5:
Circular search on first few updates.

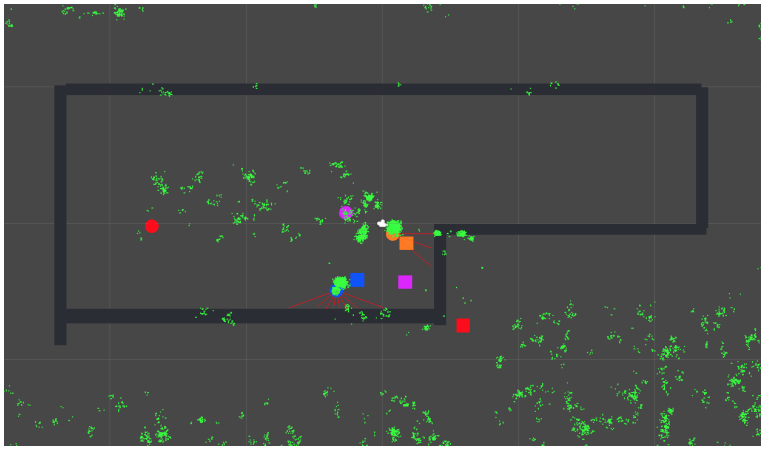

Figure 6:
Triangulation method on first few updates.

6 Future Directions

Future directions include trying different particle filter optimizations, such as different resampling mechanisms, different ways of of dealing with low-weight particles. In addition, we've found that although agents could help each other, it usually took at least one of them to be somewhat localized for the interaction to help. If both were unlocalized, sharing information did not help either. We could include how strongly an agent believes it's localized in these interactions to reduce sharing incorrect information. In the case that an agent is near a wall and its particles are spread out along the wall. If another correctly localized agent is nearby, the agent by the wall should be able to use the other agent's information to reduce the particle spread along the wall. This is something that our current approach does not deal with.

References

[1] Muhammad Saim, Khalid Munawar, and Ubaid M. Al-Saggaf. An overview of localization methods for multi-agent systems. 2017.

[2] M. Betke and L. Gurvits. Mobile robot localization using landmarks. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, volume 1, pages 135–142 vol.1, Sep. 1994.

[3] I. Shames, B. Fidan, B. D. O. Anderson, and H. Hmam. Self-localization of mobile agents in the plane. In *2008 3rd International Symposium on Wireless Pervasive Computing*, pages 116–120, May 2008.

[4] A. Prorok and A. Martinoli. A reciprocal sampling algorithm for lightweight distributed multi-robot localization. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3241–3247, Sep. 2011.

[5] M. Pelka, C. Bollmeyer, and H. Hellbrück. Accurate radio distance estimation by phase measurements with multiple frequencies. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 142–151, Oct 2014.