Comp 4180 - Assignment 4

Judah Zammit
zammitj3@myumanitoba.ca
7839359

Anamelechi Stanley
stanleya@myumanitoba.ca
7819079

December 15, 2020

1 Localization

1.1 Particle Filtering Method

We begin by initilizing all particles to either random locations and orientations or the starting location. Then, whenever a command is sent to the robot, we apply the appropiate motion to each of the particles. After the motion is applied, we add noise by sampling from a unit normal distribution and adding this sample to the location and orientation of the particle. Each particle and each element of the location and orientation uses a seperate sample.

After this, we use the camera to make observations. We then calculate the probability of observing these features given the location and orientation of the particle. We use this probability as the unormalized weight of each particle. We then resample the particles by taking a weighted random sample from the particles using the normalized weights.

We ensured that all of our calculations were vectorized, not iterative, enabling us to use two thousand particles.

1.2 Observations

This section will elaborate on the observations that we used.

We noted whether the goal was in view and geometrically calculated whether each particle should be able to see the goal. If the observation was the same as what the particle expected to see, we made the probabilty 1. Otherwise, we made it 0.2.

If the goal was not occluded by anything, we calculated where, exacty, between the goal post the robot's trajectory was pointing towards. For example, whether it is heading towards the left or right most side of the goal. Once again, we calculated the same information from the particles and set the probability equal to 1 if these two values were within 5 pixel of eachother and 0.2 otherwise. Though the goal was very rarely not occluded, when it wasn't, this feature was very effective for the localization of the robot.

We calculated the distance of the robot from the goal (if the goal was in sight) using the triangle similarity. We then set the probability to be the likelihood of the y position of the particle under a gaussian distribution with the mean the previously calculated distance and variance 10. Note that all of these were done for both goals.

We found whether the robot's trajectory was pointing off the right or left edge of the field and calculated the same for each particle. If these two values agreed, we set the probability to 1. We set the probability to 0.2 otherwise.

Finally, we simple set the probability to 0 if the particle has gone off the edge of the map. To combine these observations, we simply mul-

tipled the probabilites. Note that, when the robot was obstructed, we ignored all of these features, except for the off the edge of the map feature.

1.3 Localization Results

We tested our algorithm on world 1-1,1-2,3-1 and 3-2. Using initial notification, the algorithm works nearly perfectly on all the worlds except when it is very close to the goal, as there are very few features to detect.

Without initial notification the performs identicly on maps where the goal is visible very early on–i.e all but 3-1–and takes longer but still finds the robot when the goal comes into sight on maps where the goal remains out of view for longer.

All experiments were fairly consistent accros multiple runs.

2 Mapping

2.1 Framework

We represent each particles' hypothesis about the location of a obstacle with a two dimensional gaussian distribution with diagonal covariance.

To generate new and update old hypothesis, we first observe the bounding boxes of the obstacles from the camera of the robot. With these bounding boxes, we estimate the obstacle's distance away from the robot as well as the angle away from the robot's trajectory. Using the tringle similarity and adjusting for a narrow field of vision, we were able to estimate the obstacles relative position from the robot with reasonably high accuracy.

We then chose whether the observation belonged to one of our previous hypothesis or if it was a new observation (more details on this in the next section). If it is a new hypothesis, we initilize it with a mean at the location of the observation and a covariance matrix with 200 along the diagonal.

If it belongs to an old hypothesis, we calculate the likelihood of the observation under the hypothesis gaussian distribution and use this as the particles weight. This weight is added to the weights calculated in the localization section to prevent the mapping weight from dominating the localization weight as it tends to be significantly smaller.

To update the old hypothesis, we simply multiply the old hypothesis' gaussian distribution with a gaussian distribution with mean at the location of the observation and a covariance matrix with 200 along the diagonal.

2.2 Obstacle Assignment

This section will go into detail on how we choose to generate a new hypothesis or update an old one, given some observation.

We first check whether there are any hypothesis with a mean within 75 pixels from our observation. If so, we automaticly assign it to that hypothesis. If there are more then one, we chose the one closest to the observation.

If none our within 75 pixels, we calculate the likelihood of the observation under all the old hypothesis' gaussian distributions. We check whether there are any that resulted in a likelihood more then 1e-20 and assign the observation to it if there are. If there are more then one that accomplish this, we choose the one with the highest likelhood.

If none of these conditions hold, we generate a new hypothesis and set the weight to 1e-10 to discourage the generation of new obstacle hypothesis. If the observation falls outside of the map, we ignore the observation and set the particles weight to 0.

2.3 Mapping Results

- **World 1-1** The mapping performed very well with and without initial notification and was fairly consistent accros multiple experiments.
- **World 1-2** The mapping performed well with and without initial notification but frequently double detected the same obstacle. In addition, without initial notification, the obstacles are drawn correct relative to eachother but are often shifted. This was fairly consistent accross runs.
- **World 3-1** Due to the high volume of obstacles the mapping performed poorly with and without initial notification.
- **World 3-2** The mapping performed well (most of the times) with initial notification and it performed alright, but not great, without initial notification.