Localization with a Particle Filter

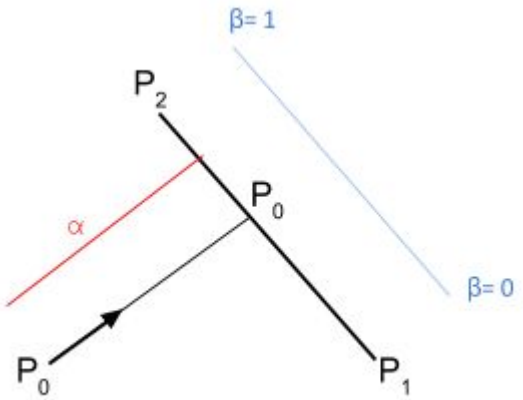
Tom Wilmots, John Larkin, and Ursula Monaghan

Particle Filter Review

- Represent possible robot positions as (x,y,theta)
- Start with n particles randomly distributed in maze
- Motion Update:
  - Given control u, update particles: $P' = \{x1, x2, ..., x_n\}$ with each x_i' sampled from $P(x_i' | x_i, u)$
- Measurement Update:
  - $W_i = \eta P(z | x_i')$
- W_i is the probability that particle i is a good particle

Ray Tracer

- Built a Ray Tracer to determine our expected measurements
- Ray Tracer takes a map of the world and sends out rays from a point and tests intersections with the world
- We record the length of the ray that hits the closest object



$$P_i = P_o + \alpha \mathbf{d}$$

$$P_i = P_1 + \beta(P_2 - P_1) = P_1 + \beta \mathbf{k}$$

$$\text{where } \mathbf{k} = (P_2 - P_1)$$

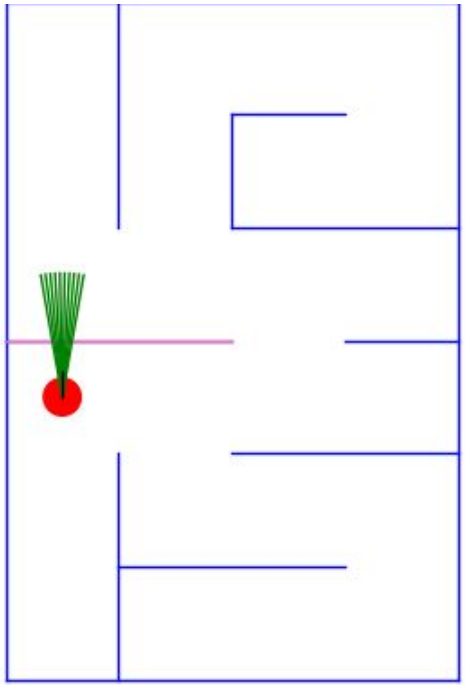
$$P_1 + \beta \mathbf{k} = P_o + \alpha \mathbf{d}$$

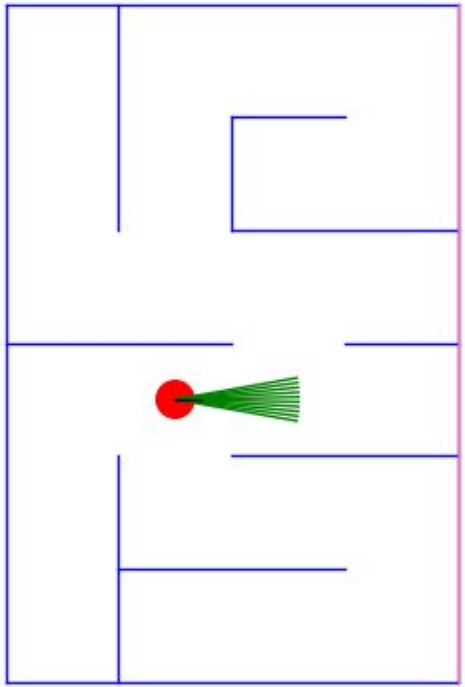
$$\alpha \mathbf{d} - \beta \mathbf{k} = P_1 - P_o$$

$$[\, \mathbf{d} \;\; \mathbf{k} \,]\, [\alpha \,;\, \beta] = [P_1 - P_o]$$

Solve for α to get distances, check to see if β is between (0,1)

Visualization of Ray Tracer





Basic Visualization Debugger - Suggested by Matt

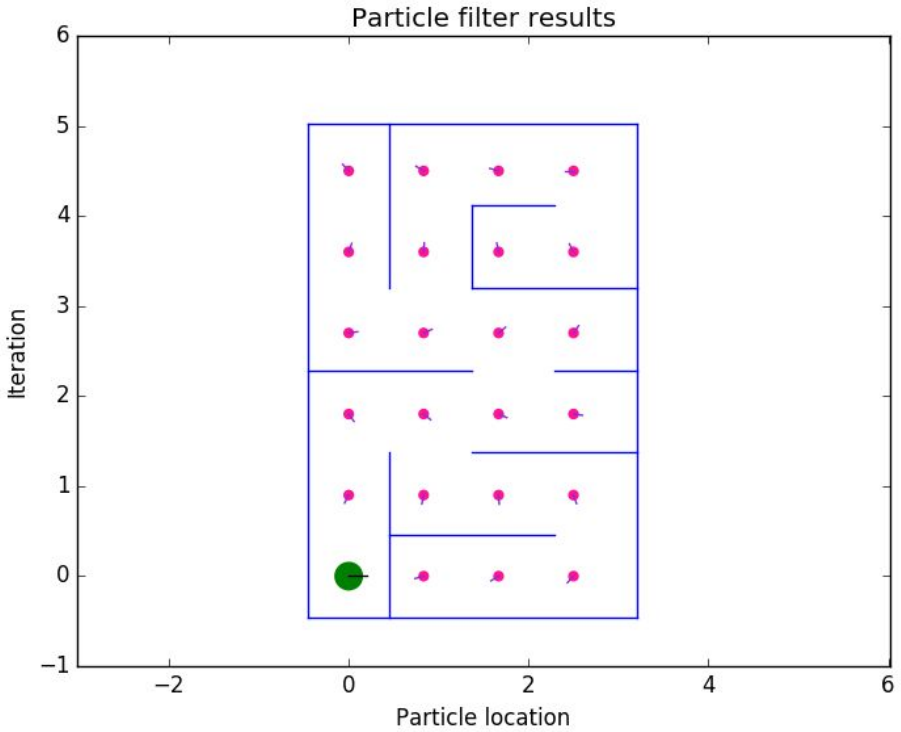
Real Measurement

- Subscribed to the SCAN topic of ROS to collect real measurements from Kinect
- Kinect was able to scan range of +/- 30˚ and output 640 rays, giving 640 distance measurements
- Did not use all 640 measurements, selected a subset of the rays to minimize computational runtime.
- Kinect is not at center of robot: Need to transform to account for the position of the scan

Updating Weights

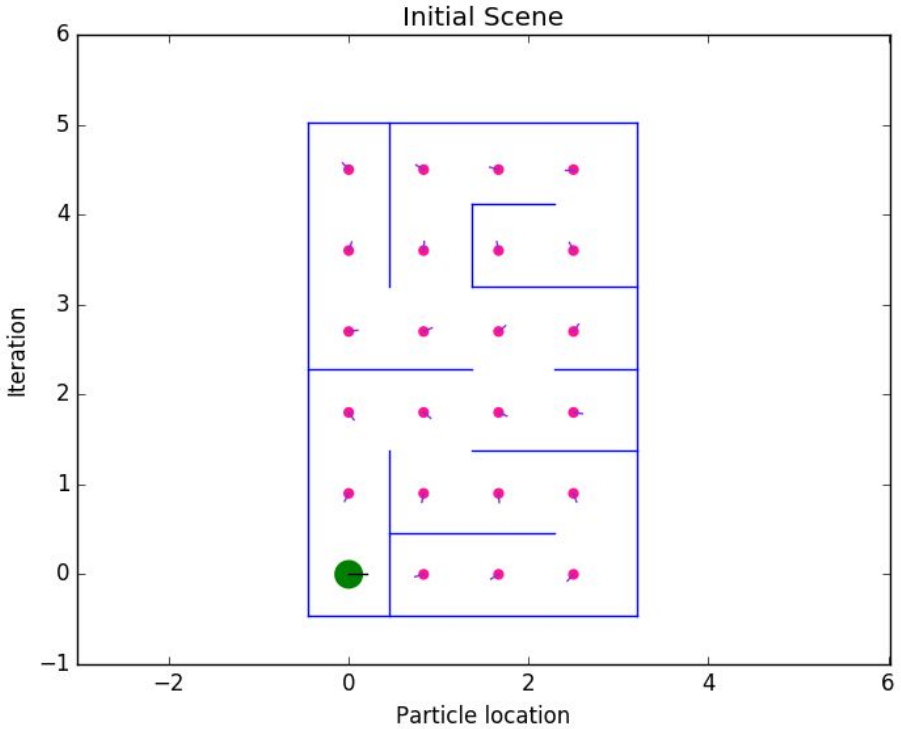
```
Particles = 3 by n array (n particles with x,y,θ)

weights = list of size n

z = measured rays from kinect

for i in particles:

     list_of_rays = expected measurements for this particle from Raytracer

     for j in list_of_rays:

          weight[i] = weight[i] * exp( (z[j] - list_of_rays[j] )^2 / (2*sigma^2) )

     weight[i] = weight[i] / sum( weight[i] ) #normalize weights

 #resample particles with replacement
```

Results (Trial 1)



- Measurement and motion step are plotted sequentially for a given iteration
- Pretty accurate path
  - During the run, Leela *actually* veered near cell (2,1)

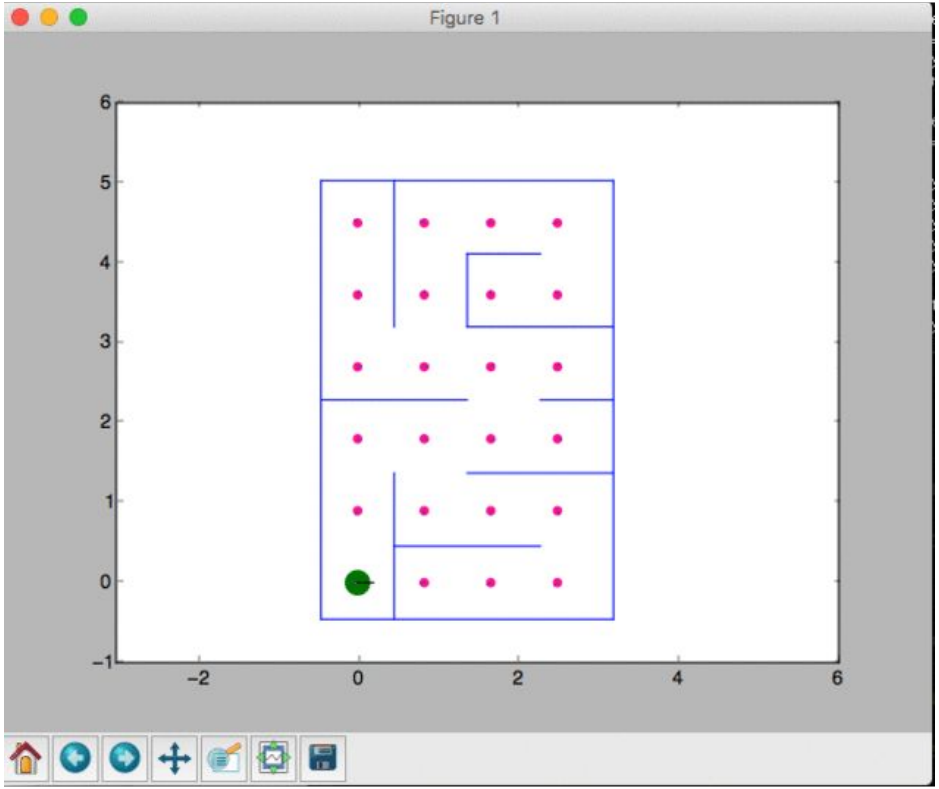
Results (trial 2)



- Same path as before

Problems

- Ran into a lot of different bugs
  - Motion Update
  - Underflow
  - Additive noise with motion update
    - Resampling
  - Real potential for non-convergence
  - Correct computation of weights / fundamental understanding of particle filter



Extensions

- Optimizing runtime - believe to be O(n*m*w) where *n* is the number of particles and *m* is the number of rays and *w* is the number of walls
  - More advanced data structures could reduce runtime
  - Could allow for more particles
- Degree of scan
  - Localization would be expedited with greater range of scan
    - Easy for raytracer
    - Harder for Leela