Lab 5: Localization

Team 20

We want to give our robot the ability to know where it is in a larger, real-world context

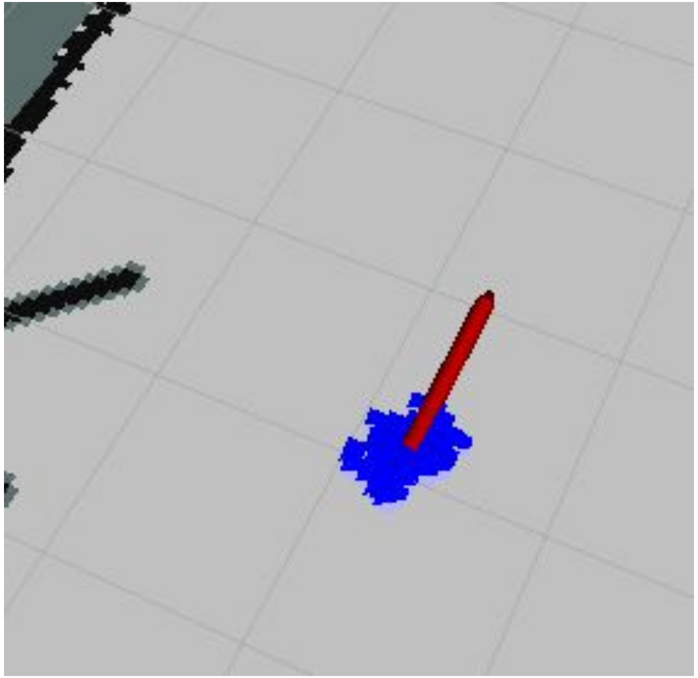
Location Estimation Via Monte Carlo Localization

Update estimate via dead reckoning

Tune motion estimate using LIDAR values

Creates high error after long periods of time

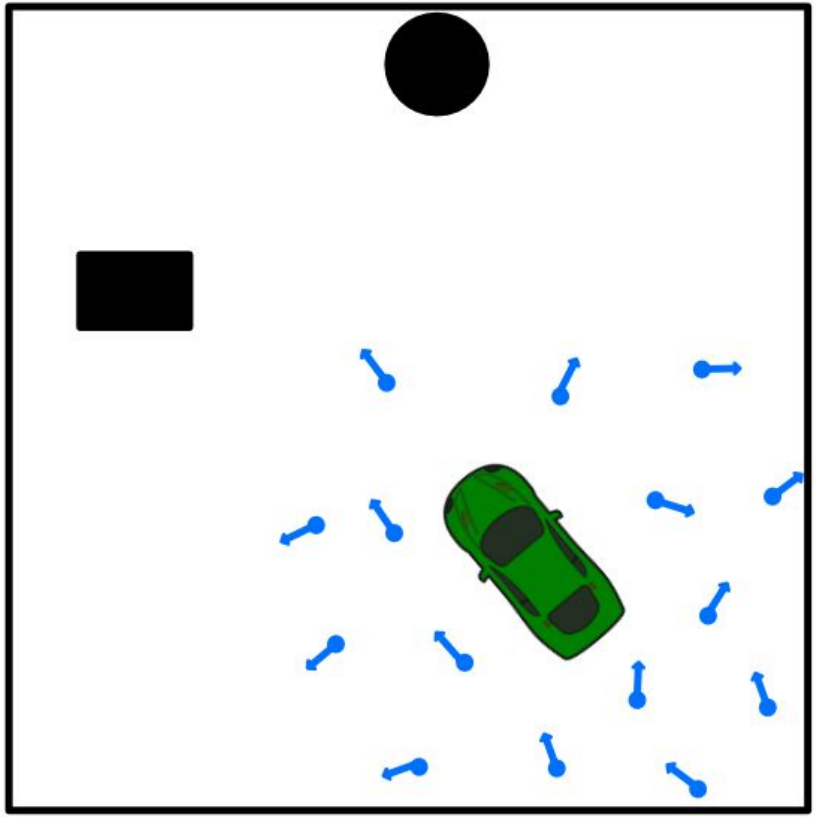
Motion Model



200 particles initialized with Gaussian noise(std=0.15) in x and y, and Gaussian noise(std=0.01) in theta

- Update the location of each particle according to odometry data + noise
- Noise sampled randomly from a Gaussian distribution with mean 0 and standard deviation of 0.01

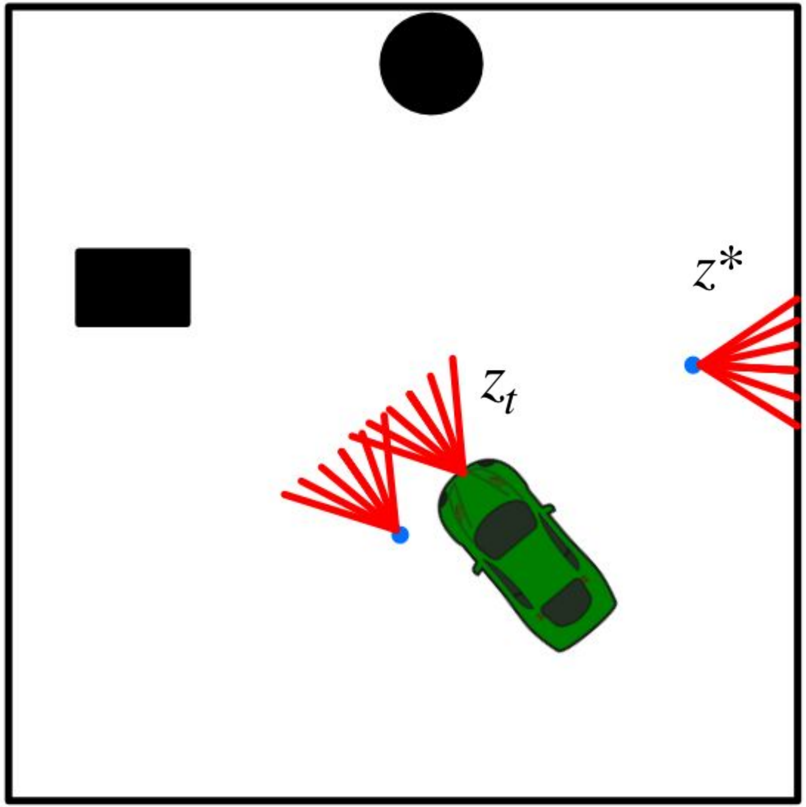
Sensor Model



- Motion model returns a set of possible spread out particles

- Sensor model aims to choose the most likely subset based on the robots sensor measurements

Sensor Model

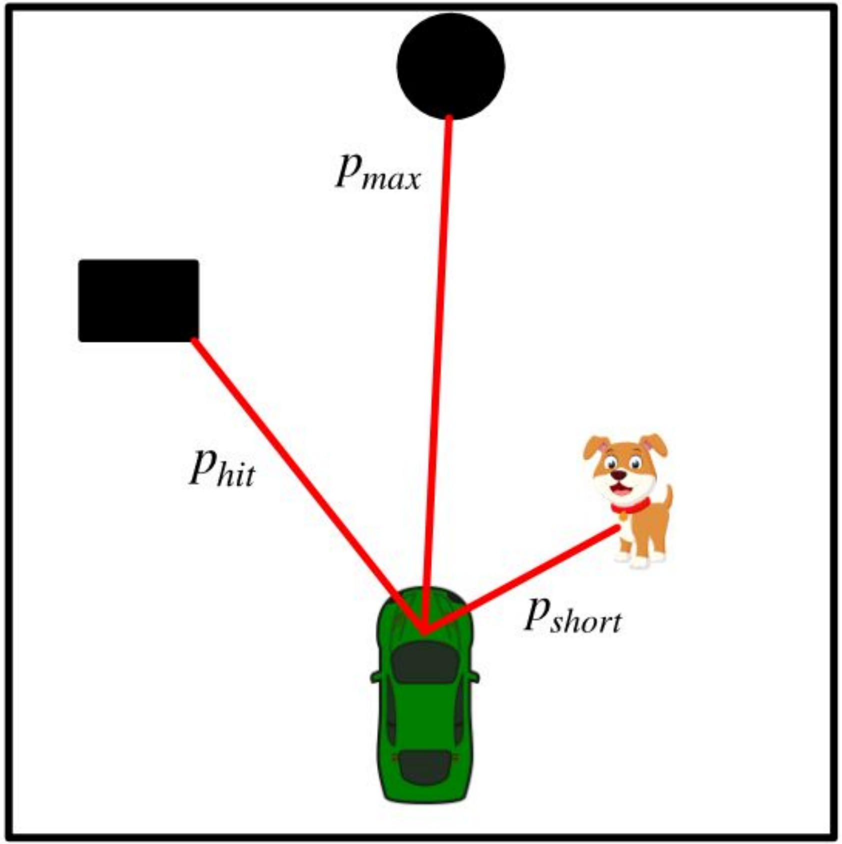


1) Generates a LIDAR scan for each particle using ray casting

2) Compares the actual scan from the car to each particles ray casted scan

3) Assigns a likelihood probability to each particle

Sensor Model Probability Components



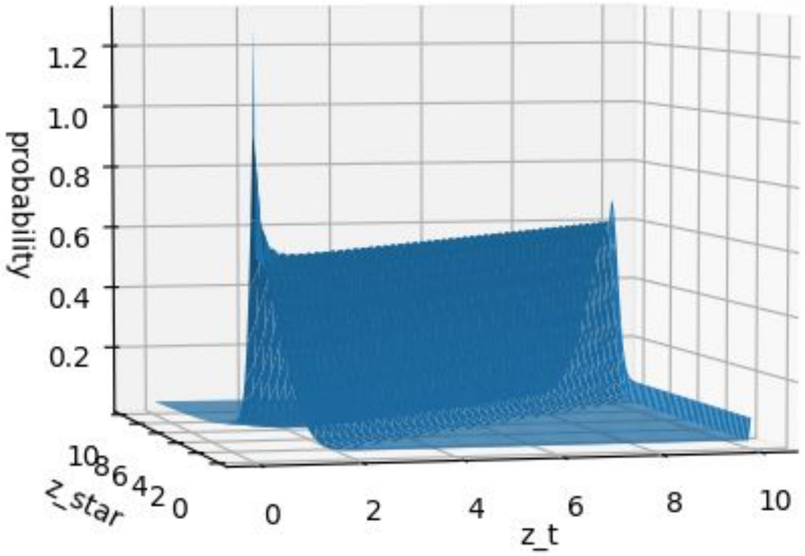
1) P(hit) : models expected measurements as a gaussian

2) P(max) : models unexpected large measurements as a probability spike

3) P(short) : models unexpected small measurements as a decreasing line

4) P(rand) : takes into account totally random effects in the form of a small flat constant

Sensor Model Probability Distribution



Pre-computed probability values with a_hit = 0.74, a_short = 0.07, a_max = 0.07, a_rand = 0.12, z_max = 10, std = 0.5

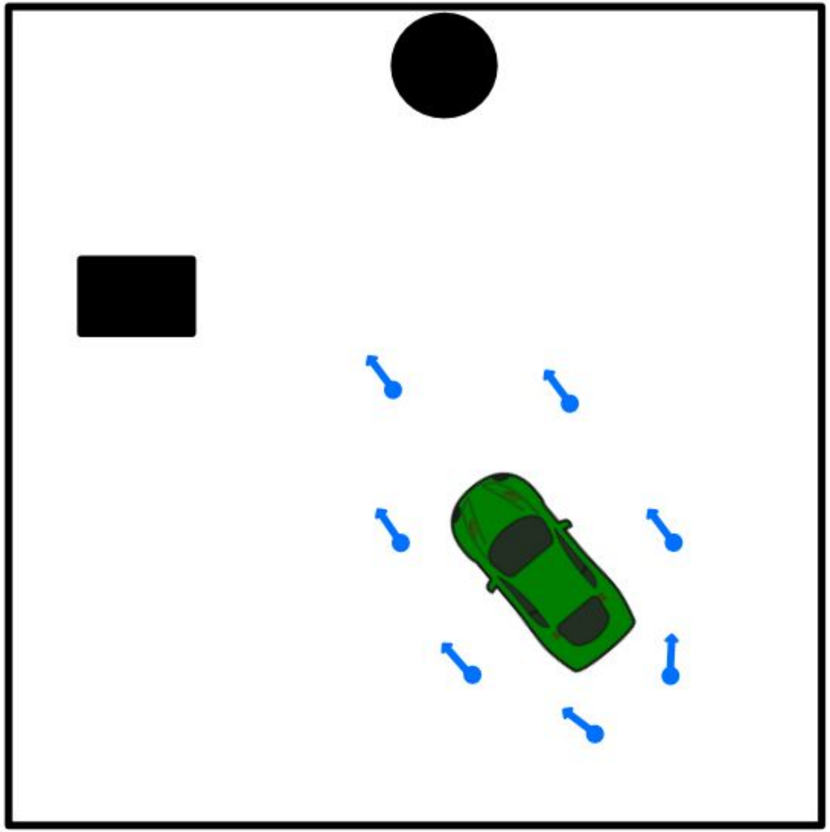
1) Determine ground truth LIDAR scans for each particle location via raycasting

2) Downsample observed LIDAR scan to same number of beams as raycasting

3) Look up particle probabilities for z_t = observed LIDAR and z* = raycasting scan

4) Resample based on above probabilities

Sensor Model Result



1) Resampled particles more closely match the robots actual pose

2) Both high probability and some lower probability particles are sampled

Particle Filter

Integration of Motion and Sensor Model

Editing conflict resolved by gating and using a stack for Motion model.

Motion Model Rate: 50 Hz

Sensor Model Rate: 7-25Hz (depending on number of particles)

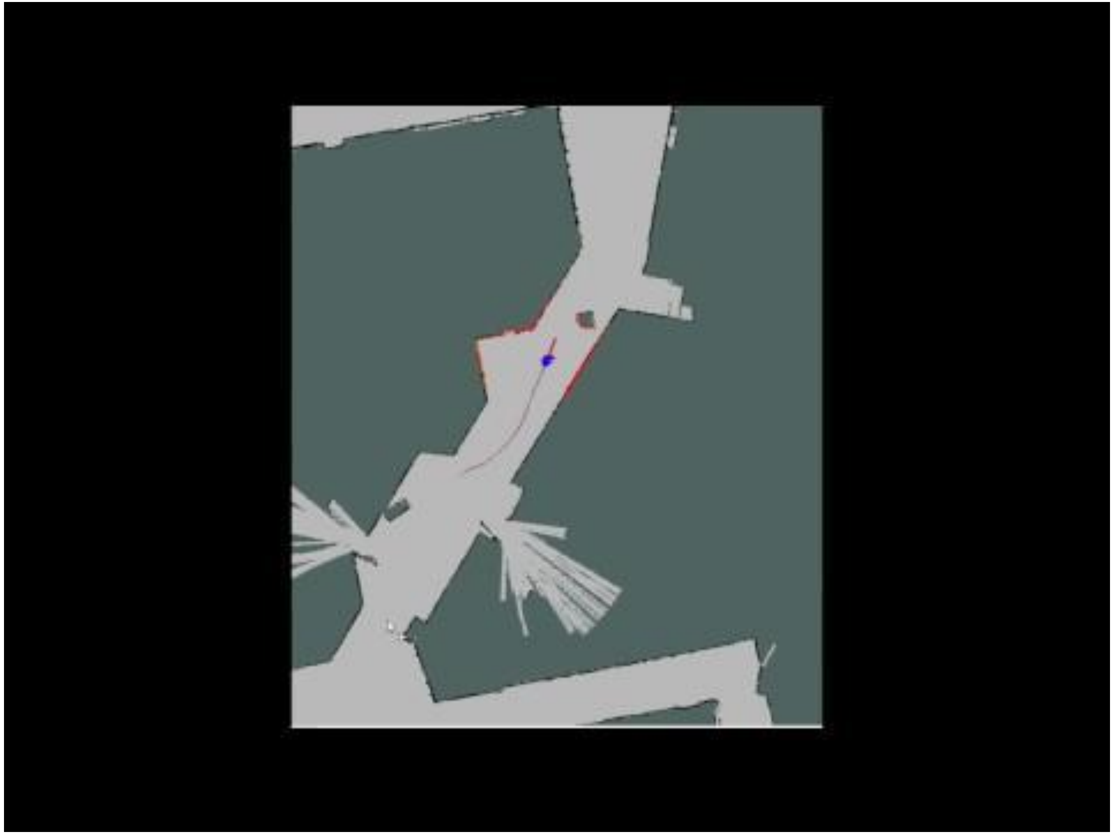
Calculation of Average Pose

Average X and Y: weighted average based on particle probabilities from sensor model

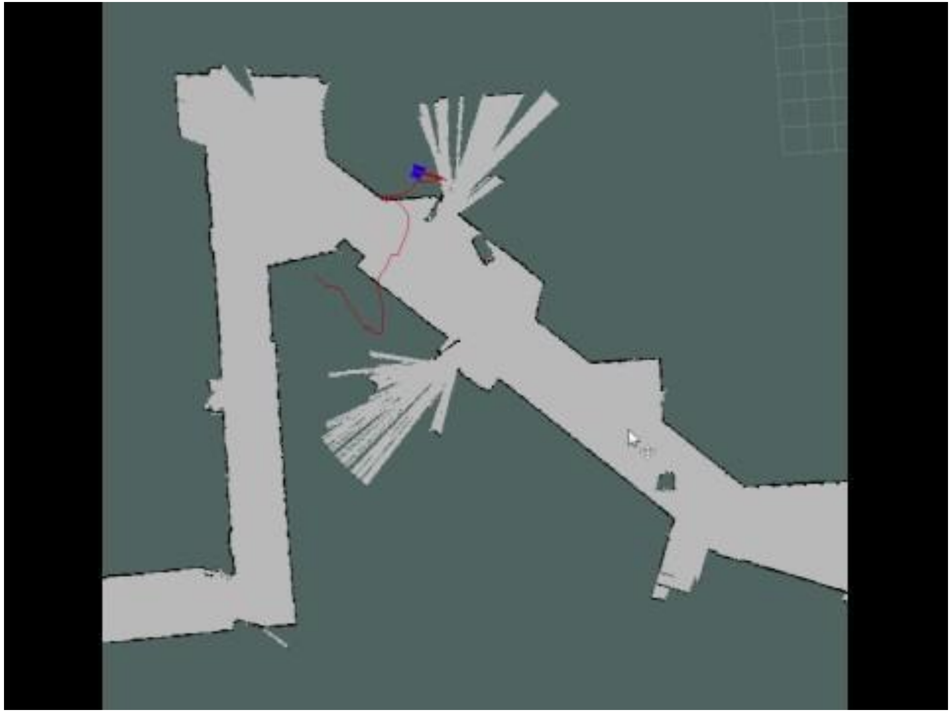
Average Theta: determination using mean of circular quantities

$$\theta = atan2(\Sigma_{i=1}^{n} p_i sin(\theta_i), \Sigma_{i=1}^{n} p_i cos(\theta_i))$$

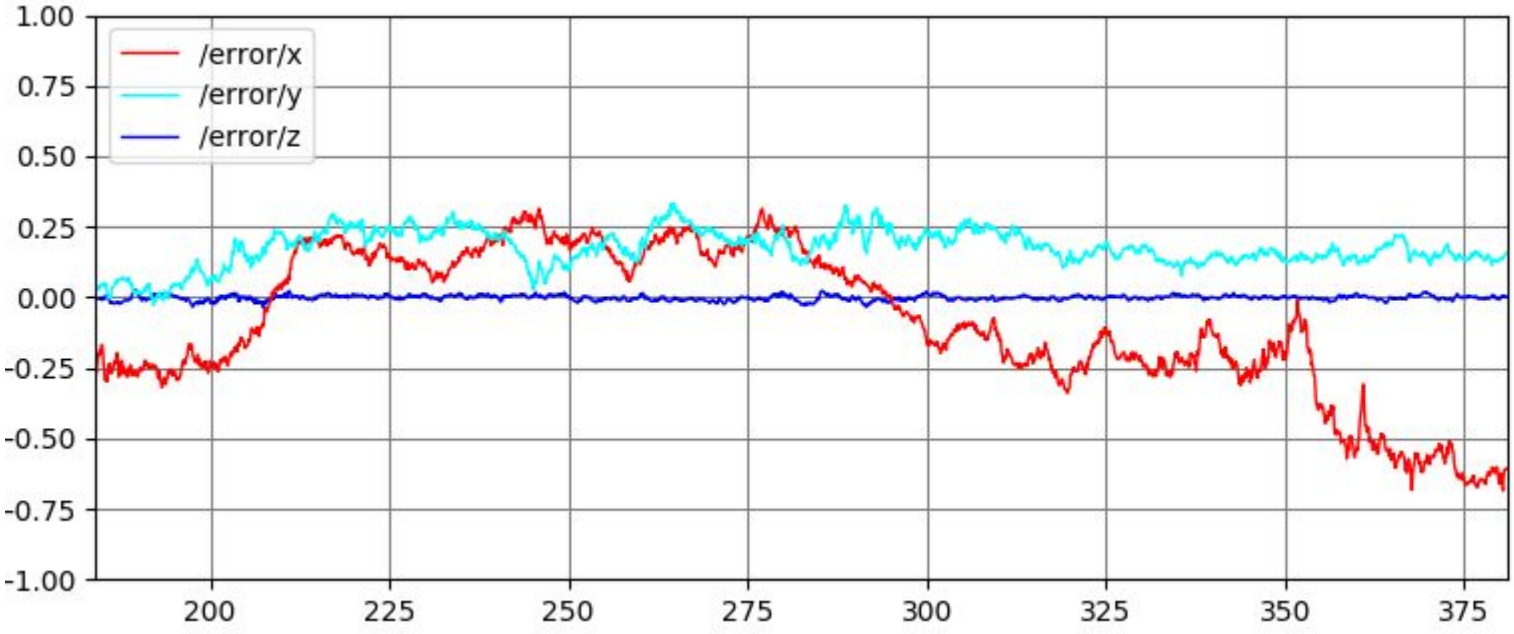
Video Demonstration in Simulation



Video Demonstration in Stata Basement



We compare our estimated pose with the ground truth pose in simulation



Error in x, y, and theta over time

Thanks!