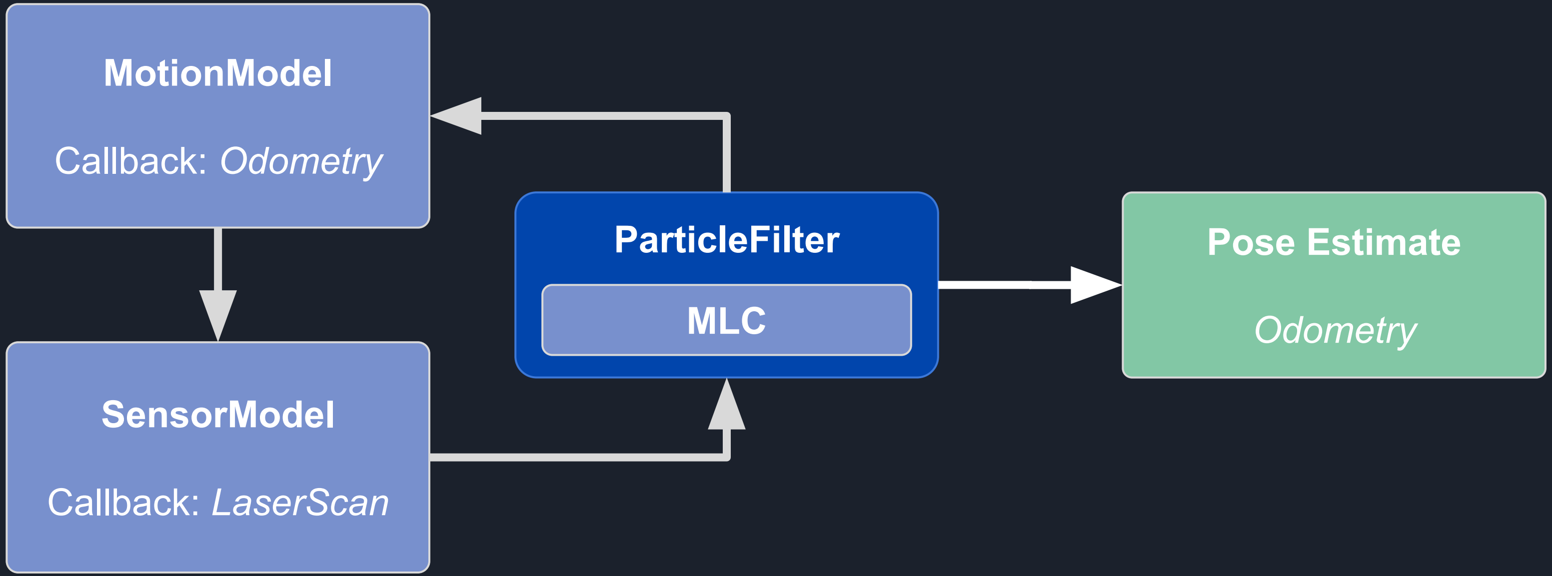
RSS Lab 5

Team Bird-Planes

Joshua, Isaac, Lilly, Mario

Components of Localization



Module 1: Motion and Sensor Models

Motion Model

$$X_k = R\Delta X + X_{k-1}$$

$$R = \begin{bmatrix} \cos(\theta_{k-1}) & -\sin(\theta_{k-1}) & 0 \\ \sin(\theta_{k-1}) & \cos(\theta_{k-1}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X_{k-1} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix}$$

Rotation Matrix (r W)

Particle (W)

Motion Model **evaluates** a series of N particles at a time

$$X_k = R\Delta X + X_{k-1}$$

We receive...

$$\begin{bmatrix} X_{k-1}^{(0)} \\ X_{k-1}^{(1)} \\ \cdots \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & \theta_0 \\ x_1 & y_1 & \theta_1 \\ & \cdots & \end{bmatrix} \qquad \Delta X = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$

Nx3 Matrix of Particles (W)

Odometry (r)

Motion Model **generates** a different Rotation Matrix for each particle

$$X_k = R\Delta X + X_{k-1}$$

To reflect probable future states, we calculate...

$$R_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrix for Each Particle

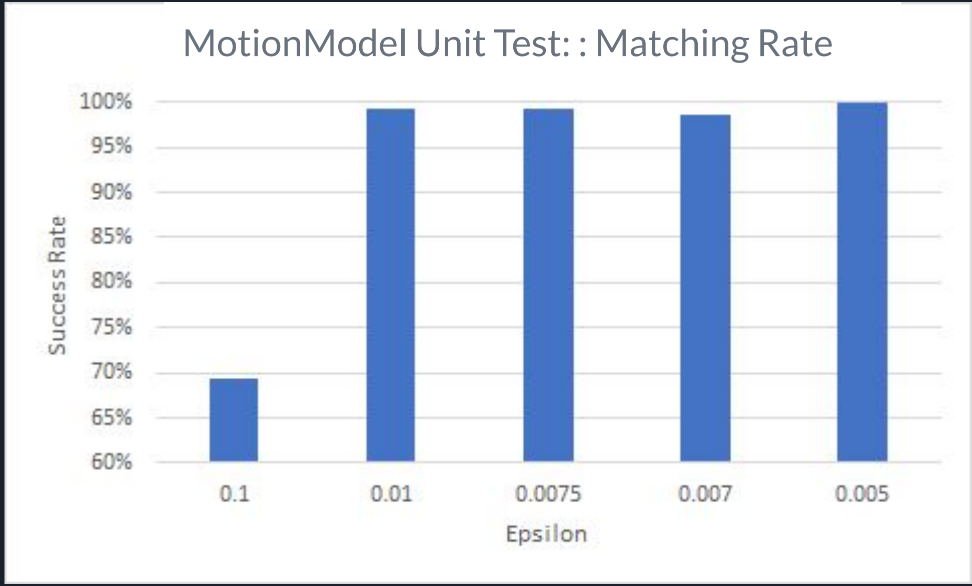
$$\begin{bmatrix} X_k^{(0)} \\ X_k^{(1)} \\ \cdots \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & \theta_0 \\ x_1 & y_1 & \theta_1 \\ & \cdots & \end{bmatrix}$$

Nx3 matrix of updated particles given the transformed odometry

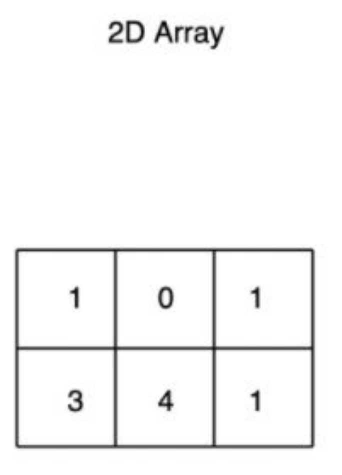
When Motion Model **injects noise** into Odometry, the response is **stable** for SD below 0.01

For standard deviation of noise ε:

$X_k = R\,[\,\Delta X \pm \mathbf{norm(0, \varepsilon)}\,] + X_{k-1}$



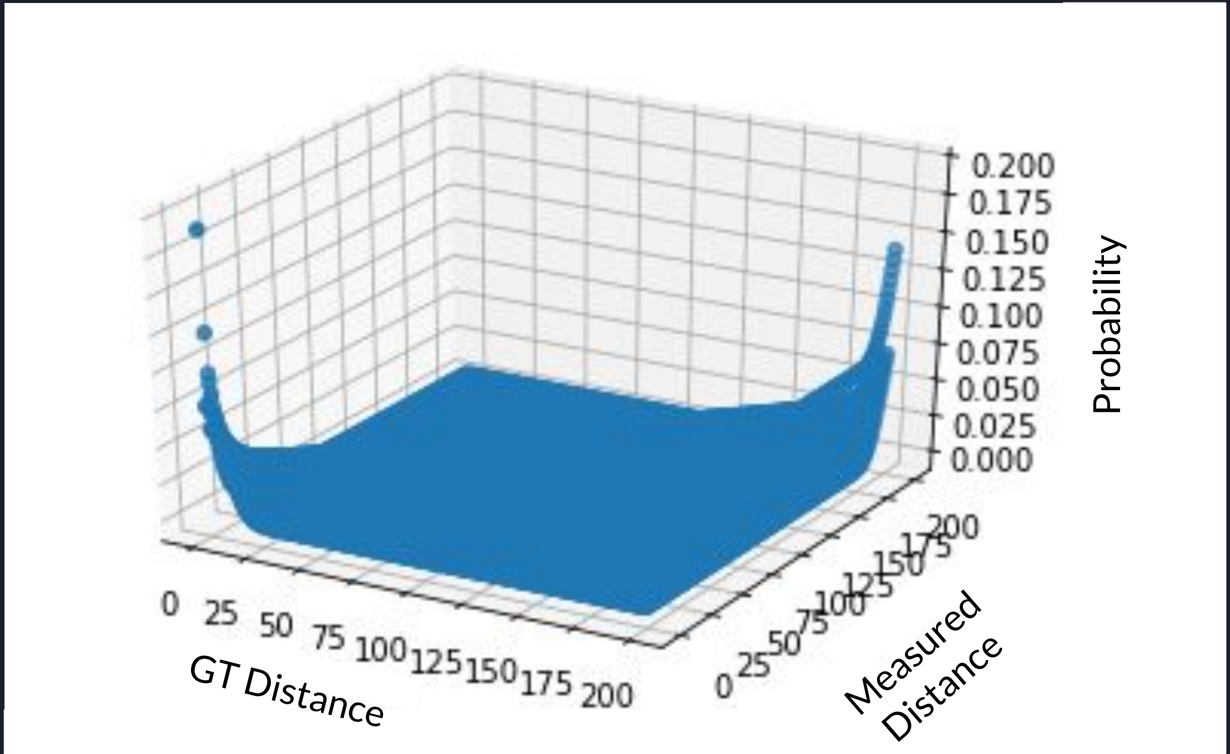
Sensor Model is **probabilistic** with weighted contributions



1. *P* (Detecting known obstacle)
2. *P* (Short measurement)
3. *P* (Missed measurement)
4. *P* (Random measurement)

$$p(z_k|x_k, m) = \alpha p|_{hit} + \alpha p|_{short} + \alpha p|_{max} + \alpha p|_{rand}$$

Sensor Model **precomputes** probability over the discrete range $0 \le z \le z_{max}$



Sensor Model

1. Ray Tracing
2. Convert meters to pixels
3. Clip scans
4. Evaluate probabilities

$$p(z_k|x_k, m) = p(z_k^{(1)}, \ldots, z_k^{(n)}|x_k, m) = \prod_{i=1}^{n} p(z_k^{(i)}|x_k, m)$$

Module 2:
Particle Filter (via MCL)

Particle Filter **receives** callbacks for Odometry and Observations, and **returns** a Pose Estimate

1a. Odometry Callback

`odom ->` **MotionModel** `-> parts`

1b. Laser Callback

`scan ->` **SensorModel**`-> probs`

2. Monte Carlo (MCL)

`Rate: 20 Hz`

Sample and Average

`particles(200) -> sample(20)`
`sample ->` **`pose_est`**

3a. Visualize Particles

`sample -> PoseArray`

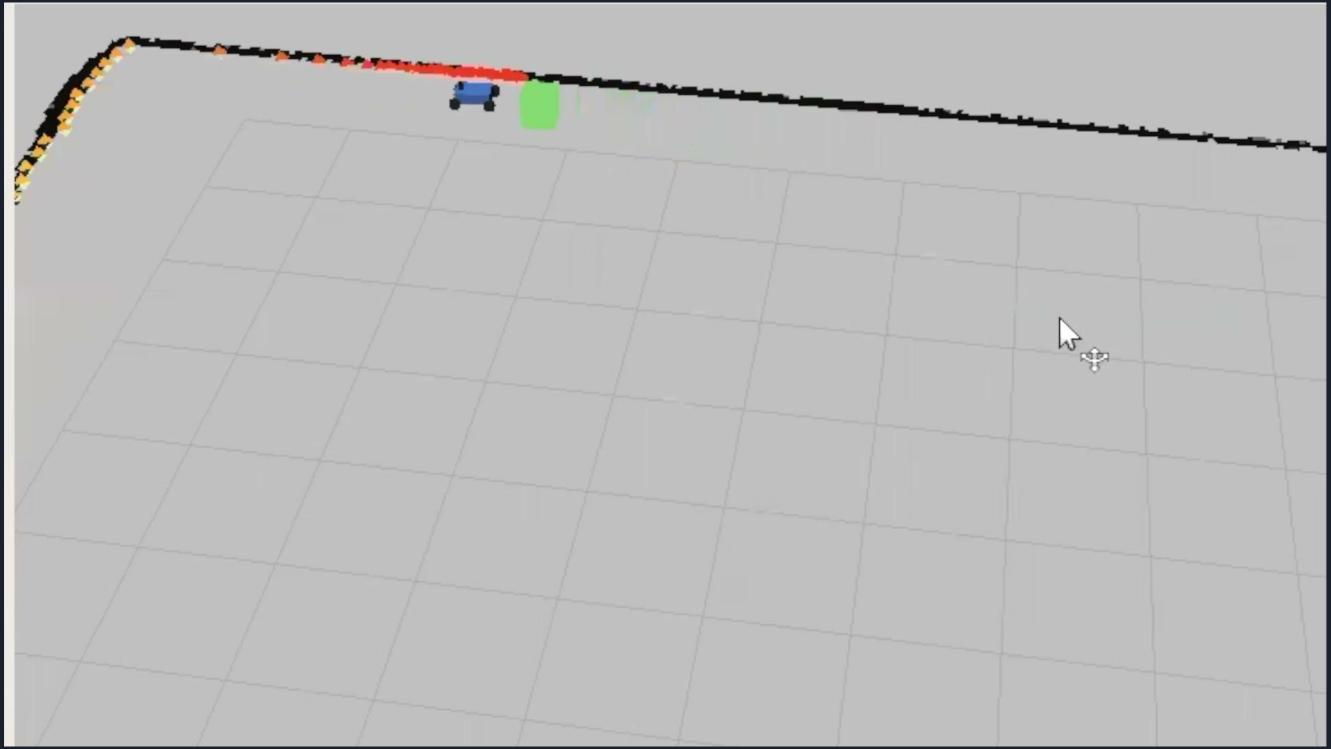
3b. Visualize Estimate

`post_est -> Marker`

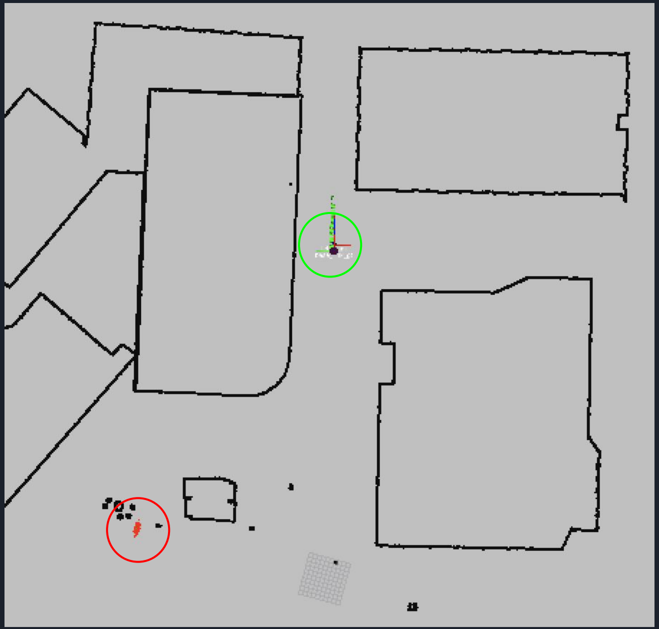
3c. Publish Estimate

`pose_est -> Odometry`

rviz: Particles and Pose Estimate

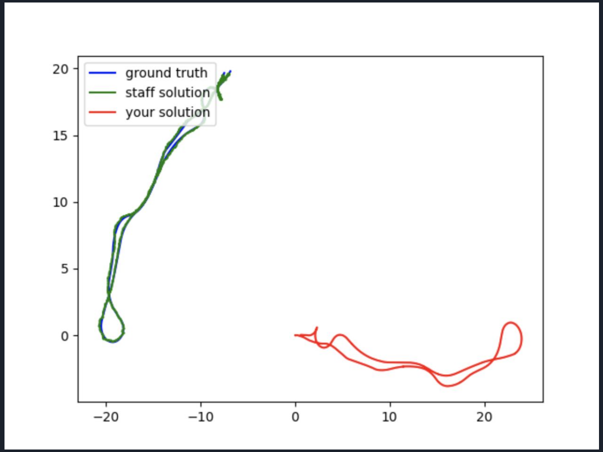


Next Steps for Particle Filter



Tesse implementation

1. Record rosbags
2. Create error graphs
3. Optimization
4. Troubleshoot autograder
5. Troubleshoot tesse implementation



Conclusion

Technical

- **Submodules:** MotionModel and SensorModel were validated more efficiently than submodules in Lab 4
- **Integration:** correct implementation, poorly tuned.

Communications

- **Working in pairs >> Working alone**
  - Github: version safety with "active" branches (e.g. `dev_sensor`, `dev_motion`)
- **Limited communication between pairs**

Thank You

Questions?

Particle Filter **receives** callbacks for Odometry and Observations, and **returns** a Pose Estimate

1a. Odometry Callback

1b. Laser Callback

2. Monte Carlo (MLC)

`Rate: 20 Hz`

Particle Filter **receives** callbacks for Odometry and Observations, and **returns** a Pose Estimate

1a. Odometry Callback

`odom ->` **MotionModel** `-> parts`

1b. Laser Callback

`scan ->` **SensorModel**`-> probs`

2. Monte Carlo (MLC)

`Rate: 20 Hz`

Sample and Average

`particles(200) -> sample(20)`
`sample ->` **`pose_est`**

Particle Filter **receives** callbacks for Odometry and Observations, and **returns** a Pose Estimate

1a. Odometry Callback

`odom ->` **MotionModel** `-> parts`

1b. Laser Callback

`scan ->` **SensorModel**`-> probs`

2. Monte Carlo (MLC)

`Rate: 20 Hz`

Sample and Average

`particles(200) -> sample(20)`
`sample ->` **`pose_est`**

3a. Visualize Particles

`sample -> PoseArray`

3b. Visualize Estimate

`post_est -> Marker`

3c. Publish Estimate

`pose_est -> Odometry`