Dead Reckoning:

1. The goal is to find the next x and y coordinate and theta direction based on the current location, distance traveled, and robot rotation. First, we have the definition:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{matrix} v\cos\theta \\ v\sin\theta \\ \omega \end{matrix}$$

We have the pose of the robot, which is the current x, y, and theta. First, we calculate the change in x based on the integration of vcos(theta) from the current time(t_n) to the next time(t_n+1). We performed several calculations and substitutions as described below:

$$\frac{dx}{dt} = v\cos\theta$$

$$\int_{x_n}^{x_{n+1}} dx = \int_{t_n}^{t_{n+1}} v\cos\theta\, dt \quad (\text{time} = n+1) \quad \Big\downarrow \ v \text{ is constant}$$

$$x_{n+1} - x_n = v\int_{t_n}^{t_n+1} \cos\theta\, dt \quad \Big\downarrow \ \frac{d\theta}{dt} = \omega \Rightarrow dt = \frac{d\theta}{\omega}$$

$$= \frac{v}{\omega}\int_{t_n}^{t_{n+1}} \cos\theta\, d\theta$$

$$x_{n+1} - x_n = \frac{v}{\omega}\left(\sin(\theta+\phi) - \sin(\theta)\right) \quad \Big\downarrow \ d = vt,\ \phi = \omega t,\ v = d/t,\ \omega = \phi/t$$

$$= \frac{\text{distance}/t}{\phi/t}\left(\sin(\theta+\phi) - \sin(\theta)\right)$$

$$x_{n+1} = x_n + \frac{\text{distance}/t}{\phi/t}\left(\sin(\theta+\phi) - \sin(\theta)\right)$$

One of the substitution we did above was that the d_theta/d_t = angular velocity. We used that definition to help solve the equation.

Based on the calculation above, we are able to calculate the next x coordinate based on the current x, distance travelled, rotation, and current theta pose. We also performed a similar calculation for y:

$$\frac{dy}{dt} = v \sin\theta$$

$$y_{n+1} - y_n = \frac{v}{\omega} \int_{t_n}^{t_{n+1}} \sin\theta \, d\theta$$

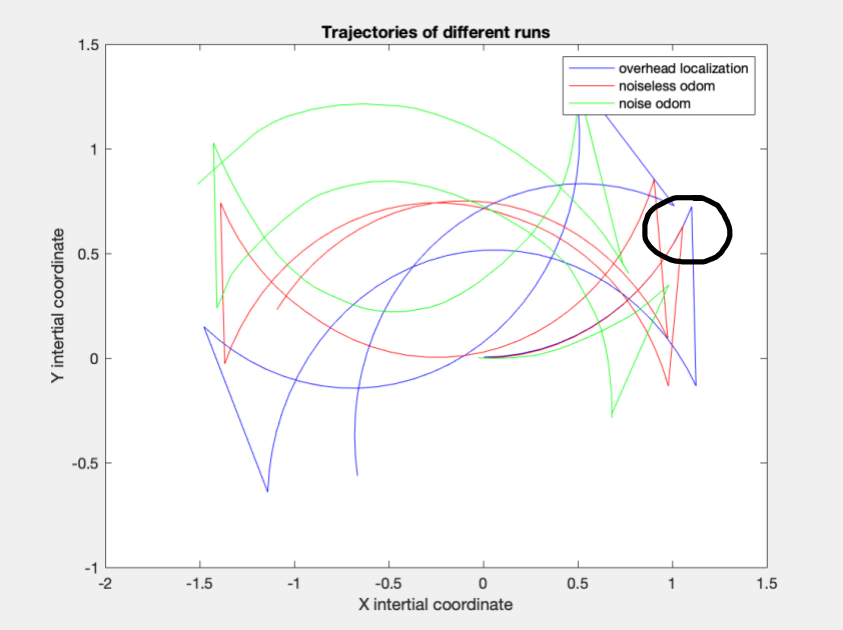
$$= \frac{v}{\omega} \left(-\cos(\theta+\phi) + \cos(\theta) \right)$$

$$= -\frac{v}{\omega} \left(\cos(\theta+\phi) - \cos(\theta) \right)$$

$$y_{n+1} = y_n - \frac{v}{\omega} \left(\cos(\theta+\phi) - \cos(\theta) \right)$$

However, there is also a special case where phi or rotation is zero. In that case, the next x and y will just be current x and y. The only thing that changes is the theta, in which new theta = old theta + rotation.

4. For the noise, a gaussian noise with mean = 0 and std = 0.5 is used.

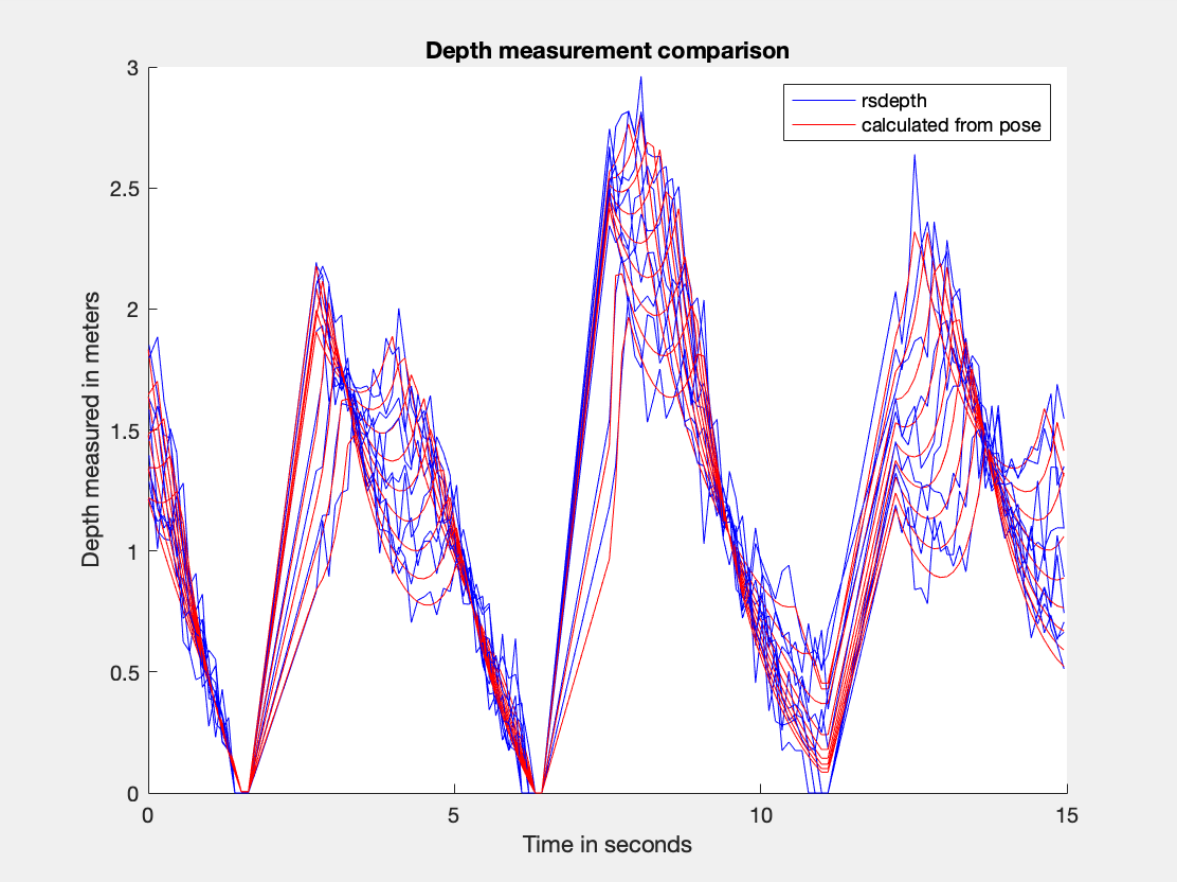


5. Integrated trajectory from first run did not match the overhead localization. At first, I expected it to be the same because the simulator should be a near perfect scenario. However, turns out they are not similar. So it turns out that there are odometry measurement errors during scenarios like bumping unto a wall, which may be caused by time delays. This can be noticed on the “circle” portion of the graph as the robot hits a wall. Additionally, since we are integrating the odometry result to find the trajectory, the error become even more significant and it will keep on compounding. As a result, the calculated trajectory kept on diverging away from the real trajectory.

6. It has a negative impact on the localization. We can see that the odometry result from the first run already causes a diversion from the true trajectory. When noise is added to it, the odometry measurement will even be more off. As a result, we can see that the difference between the calculated location of the robot from the second run to the true location is even greater. This shows that sensor noise would cause an even greater miscalculation of the robot location.

7. The errors may be from the robot itself, such as a difference in wheel diameters between the two wheels, or maybe inaccuracies on the encoder of the wheels. The errors may also be from the environment, such as if the robot is going through uneven terrain or there is wheel slippage through slippery terrain or bumping unto walls. These errors may somewhat be difficult to prevent.

Expected Depth Measurement



4. On the graph above, we can see the blue line plot showing the depth measurement using the realsense sensor, with added noise of 0.1std. We can see that it is very jagged and not identical to the red plot of calculated depth using depthPredict.m function. In fact, I did not expect them to be identical. This is because since the realsense sensor is added with noise, similar to real-life scenario, and so there will be some errors to the measurement. As a result, it will not be identical to the expected depth that we calculated using the pose of the robot.