Names: Andrew Lan, Mateo Munoz, Kieran Stone

Lab 4

Part 0: Transformation Matrix

w_x = cos(θ)-sin(θ)X_r * r_x

w_y = sin(θ) cos(θ)Y_r * r_y

1 = 0 0 1 * 1

2. We roughly spend 5 hours programming

3. Since the robot is doing a rotation, the standard rotation matrix for robot is:

R = cos(θ) -sin(θ)

sin (θ) cos(θ)

To perform a translation, we need to multiply by the position vectors $[x,\ y]^T$, and since there is no z coordinate change, z = 1

$T\ =\ \cos(\theta)\quad -\sin(\theta)\ \ x$

$\sin(\theta)\ \ \cos(\theta)\ \ y$

0 0 1

Thus, any point $[r_x, r_y]$ in the robot's coordinate is transformed into world coordinate as:

w_x = T r_x

$w_y\ =\ T\ r_y$

$1\ =\ T\, 1$

4. 1/300 = 0.003 meters per pixel

5. If the odometry goes wrong, it will affect the lidar sensor readings and cause incorrect world coordinates transformation, and further cause misaligned mapping.

6. The resolution should fit with the robot’s movements to prevent the robot moving through multiple pixels at the same time or taking too long to move to the next pixel.

If the resolution is too low, the details for the map and the routine of the robot may not be accurate and cause collisions. If the resolution is too high, it will take more time to generate the map and computational heavy.

7. We can store the first Lidar sensor reading as the starting line and keep comparing it with the Lidar reading throughout the whole routine, once the reading gets similar to the starting line, we can say that the robot goes back to the starting line.