Names: Andrew Lan, Mateo Munoz, Kieran Stone

Lab 3 Report

1.1:

Solving forward differential equations:

$$x_R = \frac{r\phi_l}{2} + \frac{r\phi_r}{2}$$

$$\theta_R = \frac{\phi_l r}{d} - \frac{\phi_r r}{d}$$

$$\frac{2}{r}x_R = \phi_l + \phi_r$$

$$\frac{d}{r}\theta_R = \phi_l - \phi_r$$

$$\frac{d}{r}\theta_R + \frac{2}{r}x_R = (\phi_l + \phi_r) + (\phi_r - \phi_l)$$

$$\frac{d}{2r}\theta_R + \frac{1}{r}x_R = \phi_r$$

$$\frac{2}{r}x_R - \frac{d}{r}\theta_R = (\phi_l + \phi_r) - (\phi_r - \phi_l)$$

$$\frac{1}{r}x_R - \frac{d}{2r}\theta_R = \phi_l$$

So therefore:

$$\phi_l = \frac{x_R}{r} - \frac{d\theta_R}{2r}$$

$$\phi_r = \frac{d\theta_R}{2r} + \frac{x_R}{r}$$

1.2:

Ways to generate a trajectory:
Define a set of waypoints that the robot must drive through and use linear interpolation to connect the waypoints. You can also use other types of interpolation such as polynomial or cubic splines.

Defined list of Waypoints:

$(-0.313756, -0.413751)$ first corner.

$(0.320071, -0.416924)$ second corner.

$(0.324869, -0.256266)$ turn left.

$(0.0497542, -0.0275227)$ curve right.

$(0.344084, 0.249577)$ curve left.

$(0.122365, 0.419171)$ slight turn left.

$(-0.303882, 0.416781)$ third corner.

$(-0.211567, 0.146211)$ going around lookers right side of the box.

$(-0.305326, -0.0442482)$ getting back to the line.

$(-0.310298, -0.253309)$ start line.

2.5: Since atan2 only covers range [-pi, pi], which will cause sudden jump from negative to positive when the value is out of the range. To prevent this, we check the alpha value every time we are using it, if the value is out of range (less than -pi), we add 2pi to the value to make sure every value is in the same range.

3.2: We are setting the left velocity to be a negative constant * alpha + positive constant * rho and left velocity to be a positive constant * alpha + positive constant * rho, in this way, when the alpha is negative (need to turn right), we will have a positive left velocity and negative right velocity to turn right, vice versa. When we are reaching the target, the velocity keeps decreasing to prevent passing the target.

2. The role of the position error is to know how far away the robot is from the target.

3. The role of the heading error is to adjust the robot to head the same direction as the target.

4. The bearing error allows us to rotate the robot towards the general direction of the target and then use position error to get the robot to the target.

5. We couldn't find the radius of the robot wheel to plug in our equation for adjusting the speed. When using the odometry from lab2, the robot will go the opposite direction as it should be.

6. Final controller:

Using the equation, $|\alpha| > 0.1$ we rotate the robot at $\frac{MaxSpeed}{4}$ until we point to the direction of the target.

Once the bearing error α is small, we use the equation $\rho > 0.5$ to drive the robot at $\frac{MaxSpeed}{2}$ forward to reach the target.

After the position error ρ is below 0.5, we use the equation $|\eta| > 0.1$ to rotate the robot at $\frac{MaxSpeed}{4}$ until we head to the direction desired. Finally, when η is below 0.1 we stop and record the waypoint.

Feedback controller:

$$v_l = -8 \cdot \alpha + 12.56 \cdot \rho$$

$$v_r = 8 \cdot \alpha + 12.56 \cdot \rho$$

7. It changes the priority of how the robot moves to the target, whether it is rotated and adjusting the angle to the target first or moving to a certain position and then adjusting the angle.

8. If we increase the gain constant, it will prioritize minimize certain error and ignore other error. If they become too large, the robot does not care about any other error and just focuses on one error, so it would not reach the desired position.

9. When the obstacle is detected, we can rotate the robot either left or right and follow the obstacle until we can move around the obstacle.

10. We spent about 5-6 hours programming.