Project 1 Report

Joshua Cook
Cyber Physical Design - CS-7639-O01
Spring 2020

Modeling the Kinematics of the Robots

1a: Find out the equation describing the relationship between the rpm of the wheels, ω_{wheel}, and the forward velocity of the robot, v, with the given parameters. The radius of the wheels is 1.5cm.

$$\begin{aligned} 1 \text{ revolution per minute} &= 2\pi \cdot \text{ radius per minute} \\ &= 2\pi \cdot 1.5 \frac{\text{cm}}{\text{minute}} \\ &= 5\pi \times 10^{-4} \frac{\text{m}}{\text{second}} \end{aligned}$$

1b: Find out the relationship between the translational velocity of the wheels, v_{wheel}, and the angular velocity of the robot, ω, assuming the wheels are rotating in opposite directions and with the same speed.

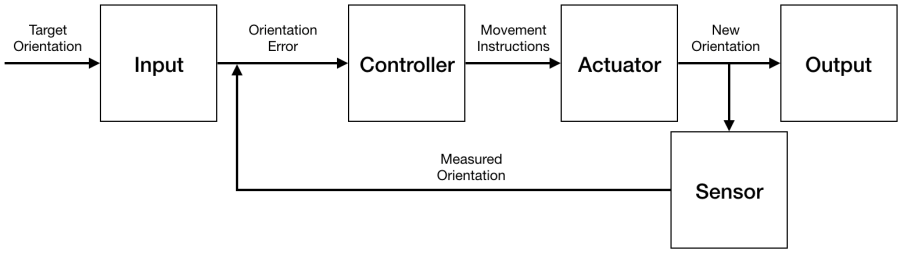
Per Robotarium Website:

> The robots are 11 cm wide, 10 cm long, and 7cm tall. However, the robots are much taller when considering the antenna and tracking markers.

Then, the radius is 5.5 cm.

$$\begin{aligned} \omega_{robot} = \frac{d\theta}{dt} &= \frac{v_{\perp}}{r} \\ &= \frac{v_{wheel} m/s}{5.5 \text{ cm}} = 18.2 \cdot v_{wheel} \; rads/s \end{aligned}$$

2: Describe the feedback control loop for controlling the orientation, θ, of the robots along a path used in the Robotarium using a block diagram. That is, a desired orientation is commanded and thus should be your input. Make sure to start with the input and end with the output in order to have a complete diagram.



3: Give brief descriptions of the sensor, actuator and controller. What do they do? Relate these components to the actual relevant functions in the Matlab code.

- **controller**, takes the *orientation error*, that is the difference between the *measured orientation* and the *target orientation* and uses this to prepare *movement instructions*, a unicycle dynamics vector, $u = (v, \omega)$
- **actuator**, take the *movement instructions* and uses them to drive the motion of the robot. This results in the robot being in a new position, including for this cycle, a *new orientation*
- **sensor**, reads the *new orientation* of the robot and combines this with the input, the *target orientation* to generate the *orientation error*

Simulation

Indicate the number of iterations needed in Simulation Task 3.

Python and Pandas were used to analyze the data.

Identify the $u = \vec{0}$ Transitions

x	y	θ	v	ω	time	transition
-0.504	0.009	-1.223	0.0	0.0	330	After Task 2
-0.500	-0.002	-0.010	0.0	0.0	374	After Orienting for Task 3
0.520	0.113	0.091	0.0	0.0	755	After Task 3
0.548	0.117	3.133	0.0	0.0	819	After Orienting for Task 4
-0.493	-0.000	-3.082	0.0	0.0	1226	After Task 4
-0.507	0.001	-0.007	0.0	0.0	1296	After Orienting for Task 5
0.490	0.002	-0.037	0.0	0.0	1688	After Task 5

Based on this, the agent begins task 3 at time index 375 and completes task 3 at time index 754.

Total Iterations to complete Task 3: 379

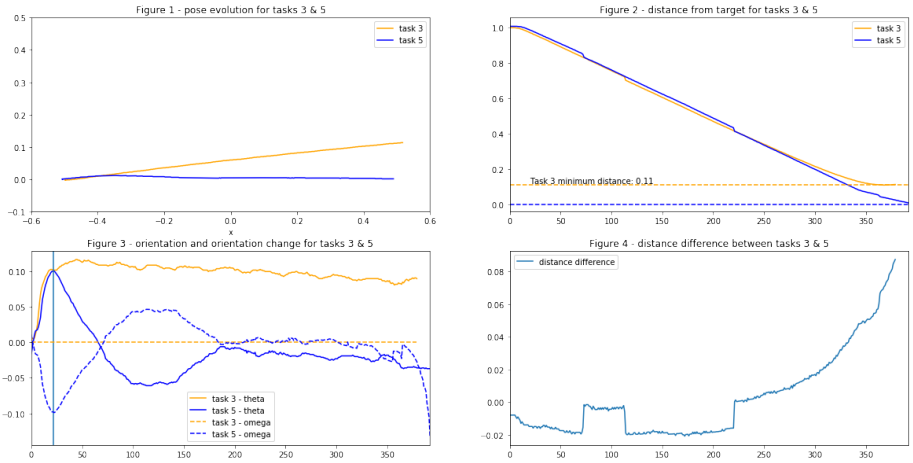
This is reasonable as the agent has a velocity of 0.08 m/s, each iteration is 0.033 seconds, and the agent must travel one meter.

```
>>> 379*0.033*.08
1.001
```

Data Analysis

1: Plot the experiment position data of Task 3 and Task 5 on the same Y vs X plot.

2: Plot the distance to the target position of Task 3 and Task 5 against the number of iterations (normalized from 0) on the same plot.



3: Discuss and explain what you see in the plots.

Task 5 Achieves The Target In Figure 2, we can see that the final distance from the target is very nearly 0 for the agent during task 5.

Task 3 Misses The Target In Figure 1, we can see that the agent misses the target in task 3. In Figure 2, we can see that the distance from the target never drops below 0.11 in Task 3. In Figure 1, we note that the agent appears to travel very nearly in a straight line throughout the task 3. Based on this we can attribute the missing of the target to a divergent orientation at the outset.

Divergent Orientation At Outset In Figure 3, we note the despite beginning with an orientation of $\theta = 0$, the agent rapidly diverges to $\theta = 0.1$ while attempting both Task 3 and Task 5. The closed loop control approach fixes this issue as can be seen in Figures 1 and 3, where the positions (Figure 1) and orientation, θ (Figure 3) of the agent during tasks 3 and 5 line up for this beginning period, but are repaired by the agent.

Task 5 Takes Slightly Longer Than Task 3 In both Figures 2 and 3, we can see that Task 5 uses slightly more time steps than does Task 3. It is

important to note that Task 5 uses a modification to its velocity, cutting it's velocity to one third when the distance drops below a certain threshold.

This was implemented in Matlab as follows, where `reasonable_speed` is set to 0.08 m/s and `turn_angle` is computed by the closed loop control algorithm:

```
if distance < 0.1
    u = [reasonable_speed/3; turn_angle];
else
    u = [reasonable_speed; turn_angle];
end
```

Distance Differential is Stable Until End of Task In Figure 2, we can see that distance trajectories are very similar during both tasks until near the end. In Figure 4 we compare the differential between Tasks 3 and 5. We see that this differential is largely stable until approximately time index 220, when it rapidly shoots up.

The Agent Leads To The Right In Figure 3, we can see that the agent leads slightly to the right. We can note the downward trend of θ in task 3 from time index 25 and in task 5 from approx time index 180.