PLANAR MOBILE ROBOT MOTION

"I certify that the following work is my own and completed in accordance with the academic integrity policy as described in the Robotics I course syllabus."

Kevin Morrison
MANE 4560
9/13/2021
Code can be found at https://github.com/KevinMorrison-629/Robotics_I

Executive Summary

The goal of this project was to accurately model an omni-directional robot and its movement throughout an environment. This was to require the calculation of the various robot poses relative to world space, along with its position and rotation as it moved throughout this environment.

An environment to display the robot was developed in python, along with methods to convert between various object frames. In order to make the objects and environment modular, classes were created that could store a robot's pose, along with other parameters.

A parameterization was developed in order to map the position and rotation to a length parameter λ for multiple straight-line paths. This parameterization was then used to find the position and angle of the robot with respect to time.

A note on Omni-directional Vehicles

Throughout this project, it was assumed that the robot is an omni-directional vehicle. That is, the vehicle can move in any direction without moving in in the plane. With a typical vehicle, like a car, the rotation of the vehicle is accomplished through the rotation of two leading two wheels. For this type of vehicle, in order to rotate it must also move forward or backwards. In contrast, omni-directional vehicles do not need to move forward or backwards to rotate. Instead, they use rotated wheels to give a component of acceleration in multiple planar axes. By rotating multiple wheels, it is possible for the components of velocity to sum to various directions. In order to allow the wheels to freely move in any direction, small discs around the circumference of the wheel are added perpendicular to the wheel turning direction. This allows the vehicle to slide in any direction with respect to the wheel, with the components of acceleration from all the wheels summing to give a total vehicle acceleration.

Section 1: Object Coordinate Frames

Problem Statement

This section required the calculation of coordinate frames of multiple objects in a room and their poses in the room frame. The following objects are shown below:

(from left to right: robot, person, table, shelf, room)

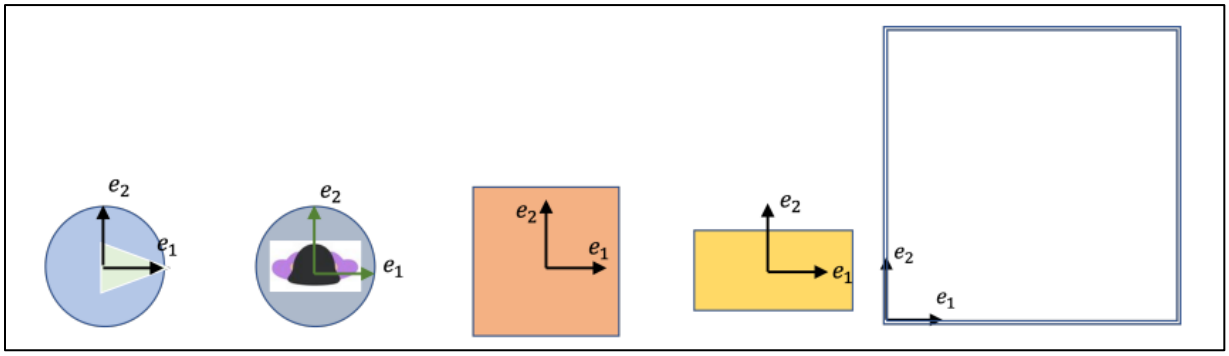

Figure 1: Objects in the Environment

The orientation of the objects with respect to each other or the world frame were given, such that the orientation and position for each of the objects could be calculated. The following information was given:

- The room frame with respect to the robot frame is:
  - $R_{10} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad p_{01} = \begin{bmatrix} -0.5 \\ 4 \end{bmatrix}$
- The person is facing north, and the robot is at the distance vector $\begin{bmatrix} 3 \\ 0 \end{bmatrix}$ away in the person frame
- The table is at center of the room with its first coordinate axis $\vec{e}_1$ pointing in the northeast direction
- The shelf is facing east and is located 3.5m to the north of the person

Approach

The position and rotation of each of the objects with respect to the room frame were calculated using these governing equations:

$$R_{01} = R_{10}^T$$

$$\vec{p}_{01} = R_{10}\vec{p}_{10}$$

$$\vec{p}_{02} = R_{02}\vec{p}_{01} + \vec{p}_{21}$$

$$\vec{p}_{02} = \vec{p}_{01} + \vec{p}_{12}$$

where:

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad \text{and} \qquad \vec{p} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Results

Using these equations, the frames of each of the objects were calculated. The robot frame is denoted as *frame 1*, the person frame as *frame 2*, the table frame as *frame 3*, and the shelf as *frame 4*. The room frame is considered *frame 0.*

$$R_{01} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \qquad p_{01} = \begin{bmatrix} 4 \\ 0.5 \end{bmatrix}$$

$$R_{02} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad p_{02} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$$R_{03} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \qquad p_{03} = \begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix}$$

$$R_{04} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad p_{04} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

A visual of the objects in the room can be seen below. The coordinate axis for each of the objects follows the right-hand rule, with the frame's local x-coordinate represented by $\vec{e}_1$ and y-coordinate by $\vec{e}_2$. In the figure, each grid section corresponds to a 0.5mx0.5m square. These conventions are held constant throughout all the sections of this report.

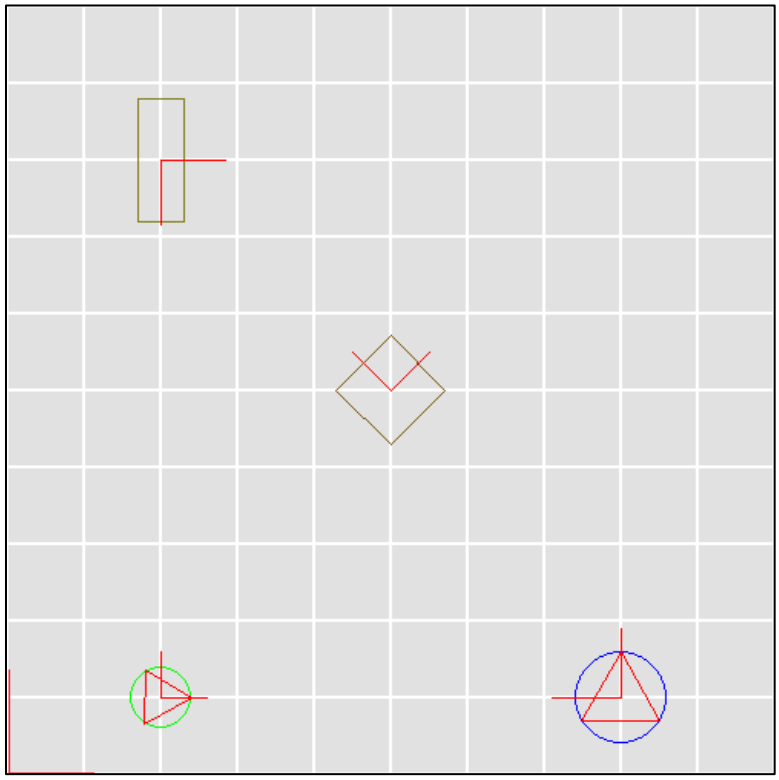

Figure 2: Mapped Object Frames

Section 2: Calculation of Robot Poses

Problem Statement

This section required the calculation of robot target poses in the room frame. The following information regarding these poses was given:

- The robot faces the front of the shelf (where the arrow of $\vec{e}_2$ for the shelf is) and the edge of the robot is 0.1m away from the shelf. Denote this pose by (O_a^*, p_a^*).
- The robot faces the front of the person (where the arrow of $\vec{e}_2$ for the person is) and the edge of the robot is 0.1m away from the cylinder enclosing the person. Denote this pose by (O_b^*, p_b^*).

The approach to calculating these poses was the same as in **Section 1: Object Coordinate Frames** and utilized the same equations.

Results

The poses of the robot for each of these cases (denoted by *a* and *b*) was calculated and shown below:

$$(\theta_a^*, p_a^*) = \left(\pi \middle| \begin{matrix} 1.55 \\ 4 \end{matrix}\right)$$

$$(\theta_b^*, p_b^*) = \left(\frac{3\pi}{2} \middle| \begin{matrix} 1 \\ 1.1 \end{matrix}\right)$$

Additionally, another figure was developed to show these positions and orientations relative to the room frame. They keep the same conventions as were used in *Figure 2: Mapped Object Frames.*

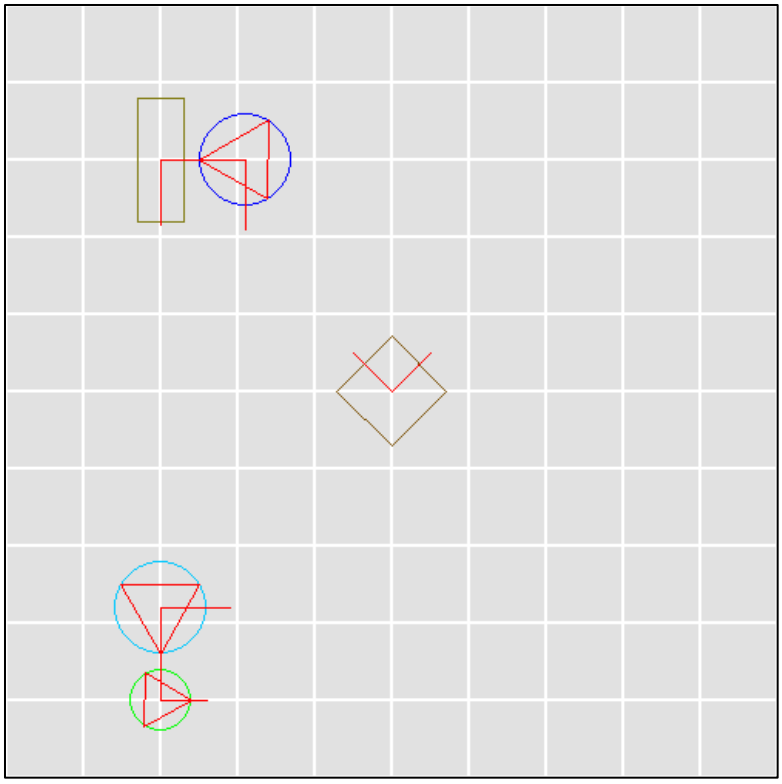

Figure 3: Mapped Robot Poses

Section 3: Calculation of Poses given relative orientations

Problem Statement

This section supposed that the robot was omni-directional. It required that a path be constructed, consisting of the least number of straight-line segments for the robot to go from its starting position to (θ_a^*, p_a^*) and then to (θ_b^*, p_b^*) without colliding with any of the objects in the room. This path should be parameterized by a path length, λ. After this, a trajectory that was indexed with time was to be generated given some constraints to the speed and angular velocity.

The speed and angular velocity constraints were as follows:

- $|u_i| \leq 2\ m/s$
- $|\omega| \leq 1\ rad/s$

Approach

In order to develop a path that would not collide with any obstacles, intermediary points were strategically calculated to avoid these barriers. The method through which these points were chosen is elaborated upon in the comments within the attached code.

Straight-line paths were generated and parameterized by the path length, λ, using the equation:

$$path_i = p_0 * \left(1 - \frac{\lambda_i}{l_{slp}}\right) + p_1 * \left(\frac{\lambda_i}{l_{slp}}\right)$$

where l_{slp} is the length of the straight-line path from p_0 to p_1 and λ_i is the parameterized length across the total straight-line path. The rotation of the robot was also parameterized using λ, though the rotation was not parameterized by each of the paths, but rather through the concatenation of the paths from the initial position to the first pose (O_a^*, p_a^*), and the first pose to the final pose (O_b^*, p_b^*).

Because the rotation to each of the poses can be done either clockwise or counterclockwise, the robot has the option of rotating in either of these directions when changing its pose. In this project, it was assumed that the robot would rotate in the direction which would result in the least change in angle over the path.

The path was indexed with time such that for a given path from $\vec{p}_0$ to $\vec{p}_1$:

$$\frac{d\lambda}{dt} = min\left(abs\left(\frac{u_{max}}{\frac{\vec{p}_1 - \vec{p}_0}{\|\vec{p}_1 - \vec{p}_0\|}}\right), abs\left(\frac{\omega_{max}}{\frac{\vec{\theta}_1 - \vec{\theta}_0}{\|\vec{p}_1 - \vec{p}_0\|}}\right)\right)$$

$$\vec{u}_{01} = \frac{d\lambda}{dt} \cdot \frac{\vec{p}_1 - \vec{p}_0}{\|\vec{p}_1 - \vec{p}_0\|}$$

$$\vec{\omega}_{01} = \frac{d\lambda}{dt} \cdot \frac{\vec{\theta}_1 - \vec{\theta}_0}{\|\vec{p}_1 - \vec{p}_0\|}$$

The lambda rate $\frac{d\lambda}{dt}$, was calculated by finding the minimum lambda rate for the positional and rotational parameterizations. These rates were then used to calculate the position and rotation of the robot at a given point in time using the following equation:

$$\vec{p}_{01}(t) = \vec{p}_{01}(t-1) + dt * \vec{u}_{01}$$

where $\vec{p}_{01}$ is the position of the robot with respect to the room frame at a given point in time (along path from the initial pose to a given *point 1*), dt is the timestep size (in seconds), and $\vec{u}_{01}$ is the velocity along the path from the initial pose to a given *point 1*. The rotation is governed by a similar equation:

$$\vec{\theta}_{01}(t) = \vec{\theta}_{01}(t-1) + dt * \vec{\omega}_{01}$$

The total path was divided into three main paths as was done in **Section 2**, and the path positions and rotations were calculated using the same methods.

Results

The path consisting of the least number of straight-line segments for the robot to go to (θ_a^*, p_a^*) and then to (θ_b^*, p_b^*) is shown below:

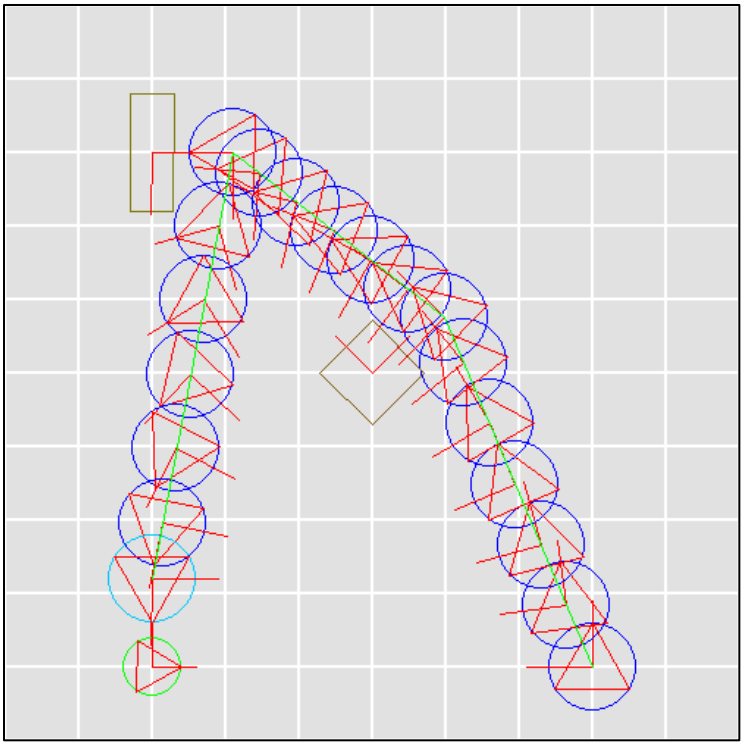

Figure 4: Mapped Robot Path

As shown in *Figure 4*, the robot passes beside both the table and the shelf without collision, with straight line paths resulting in the shortest length, 7.362m. The following figure shows the positions and rotation with respect to lambda.

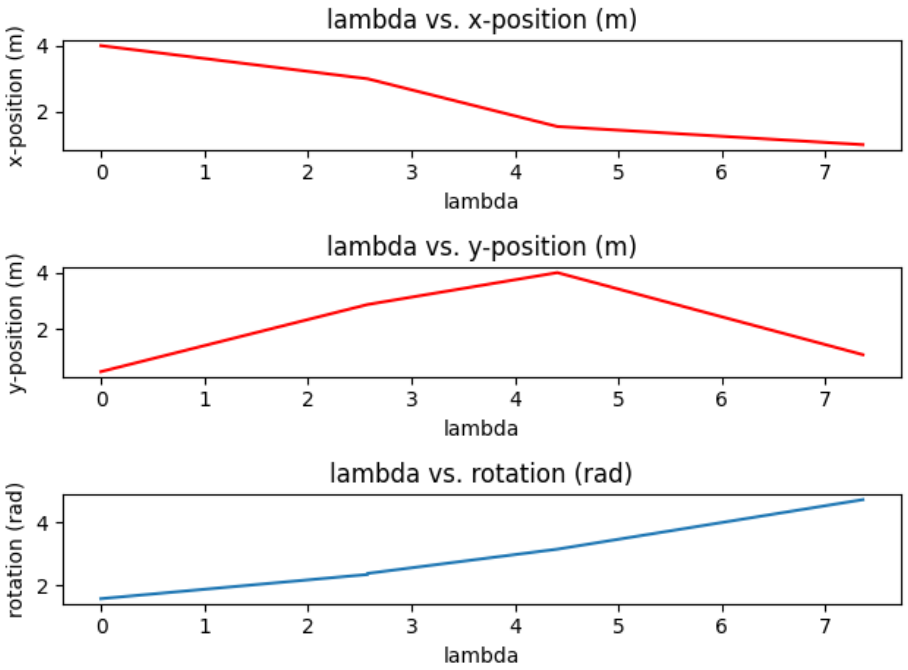

Figure 5: Robot Path as a Function of Lambda

After this, the robot trajectory was calculated, $\lambda(t)$. The maximum positional and rotational speeds were used as constraints for this trajectory. The total travel time for the robot along the path was 3.53s. The figure below shows the poses of the robot with respect to time (with respect to the room frame).

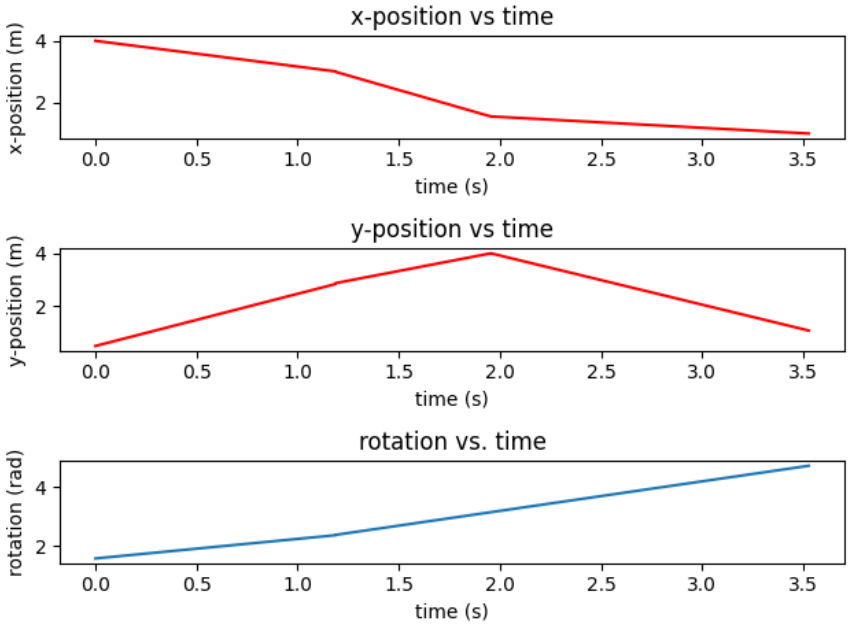

Figure 6: Robot Path as a Function of Time

In addition to *Figure 6*, a video of the robot in motion was developed. This can be found in the main folder for this project.

Conclusion

In this project, python was used in order to develop methods in which object frames could be calculated based on known relative poses and paths were generated and parameterized by a path length, then indexed with time.

The intermediary point was hand calculated, with the given path start and end points, and the object in the way being known constants. Implementation of a process to eliminate this hand calculation would be an ideal first improvement to the project.

Though this project only dealt with straight-line paths, most trajectories in the real world would not include sharp corners, and the robot velocity would not instantaneously change during movement. Creation of a gradual path using splines, or using the potential field method, could help alleviate this issue. The inclusion of a maximum acceleration could also be used to further remove the drastic velocity changes that occurred in this project.