Sawyer Path-planning with Rapidly-Expanding Random Trees

Justin Sadler

***Abstract*— In robotics, inverse kinematics is a well-researched topic. Rethink Robotics, the company that developed the 7-jointed Sawyer robot, provided an inverse kinematics service that can analytically determine the necessary angles needed to move the end-effector to a desired position. However, while inverse kinematics is a useful tool for determining the configuration necessary to move an end-effector to a specified Cartesian coordinate, it is difficult to take obstacle avoidance into account. Furthermore, analytical and numerical methods for the Sawyer robot must take into account that the robot has 7 links that could collide with obstacles if the path planning is careless.**

In this paper, I introduce a path planner that utilizes inverse kinematics to determine the goal configuration and utilizes Rapidly-exploring Random Trees (RRT) to compute the intermediate configurations that would avoid obstacles. This method has been implemented and tested on a Sawyer robot provided by Professor Eshed Ohn-Bar and Professor Wenchao Li at Boston University.

I. Introduction

Path planning with a jointed robotic arm through an environment with obstacles poses a unique set of challenges. Many researchers have investigated path planning with the Sawyer robot.

Existing work has utilized A* as a deterministic method of path-planning [3]. However, in a large configuration space, using A* could become unrealistic as a continuous configuration space would need to be discretized for path planning. A potential-based method that combines inverse kinematics and potential fields has shown to be a promising candidate. However, this method cannot be generalized as certain obstacles because of the Local Minima Problem [4]

Additionally, sampling-based methods, which are usually probabilistically complete, have been explored in theoretical work and practical applications [4]. However, there is sparse research on using a sampling-based method on a jointed robotic arm such as the Sawyer robot. The paper explores combining inverse kinematics with a classical path-planning method, but instead of using a potential-based algorithm, I utilized RRT, a sampling-based method.

II. Approach

At a high level, my approach was to represent the Sawyer robot as seven connected cylinders, each given a body frame. Although imperfect, these cylindrical polygons give the functionality of checking if any link of the Sawyer robot is in collision with an obstacle. Given a 3D Cartesian point for the final end-effector position, I use the inverse kinematics service provided by Rethink Robotics to get the final configuration for the joint angles. Finally, I use RRT to find an obstacle-free path between the start and goal configuration. A similar approach was done by Min Cheol Lee [6]. However, instead of RRT, they use a potential-based algorithm for path planning. I used a sampling-based method that is more suited to kinodynamic systems. In the following section, I discuss the reasoning and mathematics needed for path planning.

Note Unless otherwise specified, the unit of measurement is meters.

A. Sawyer Robot Representation

As mentioned before, I used 7 cylinders to represent the 7 links of Sawyer. Although the links are not perfectly cylindrical, they provide an overestimation of the boundaries of Sawyer to assist with obstacle avoidance. Each cylinder has a body frame, meaning a 3x3 rotation matrix and a 3x1 translation vector are associated with each cylinder. For the rest of the paper, we will refer to the 3 rotation matrices, $R_x(\theta)R_y(\theta), R_z(\theta)$ as the matrix representing a rotation about the x,y, and z-axis respectively. 1. Rotation around the x-axis (R_x):

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

2. Rotation around the y-axis (R_y):

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

3. Rotation around the z-axis (R_z):

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The "origin" of a cylinder is defined to be the center of the bottom circular face. Within the cylinder's body frame, the cylinder extends in the z-direction.

Below is an example of a cylinder used in the system. In the figure, the cylinder has been rotated about the x-axis by 45 degrees and translated by 1 unit in the z-direction

Figures 2 and 3 are views of Sawyer when all joint angles are zero.

Below are the tables for the link length and joint angle ranges [2]. The radius of each link was estimated by a tape measure.

[1]Weichao Zhou is with the Department of Electrical and Computer Engineering, Boston University, 8 Saint Mary's Street, MA 02215, United States zwc662@bu.edu

[2]Sabbhir Ahmad is with the Department of Electrical and Computer Engineering, Boston University, 8 Saint Mary's Street, MA 02215, United States sabbhir92@bu.edu

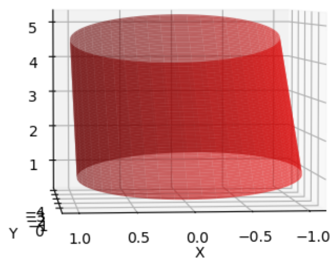

Fig. 1: Cylinder Example

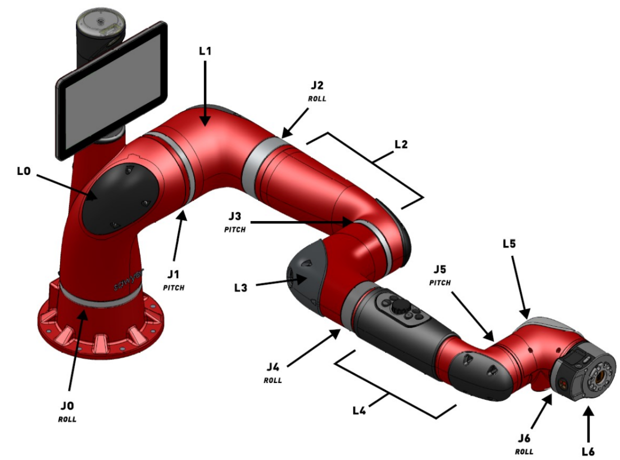

Fig. 2: Sawyer View 1

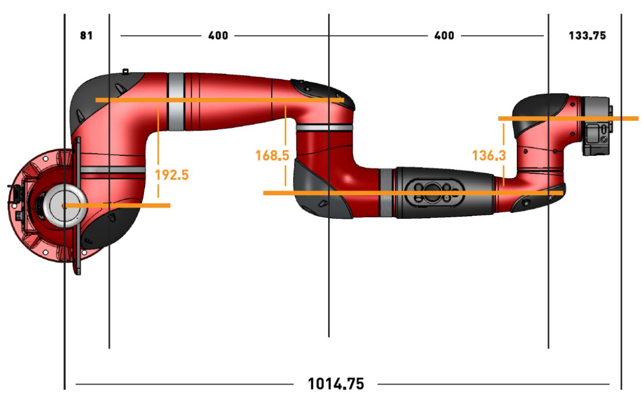

Fig. 3: Sawyer View 2

Link Name	Link Length	Link Radius
L0	81mm	80mm
L1	192.5mm	80mm
L2	400mm	50mm
L3	16.5mm	50mm
L4	400mm	30mm
L5	136.3mm	50mm
L6	133.75mm	30mm

TABLE I: Link Lengths

Joint Name	Angle Range (Degrees)
J0	350
J1	350
J2	350
J3	350
J4	341
J5	341
J6	540

TABLE II: Link Lengths

In the positioning system of the robot, the world frame is defined to be the bottom of L0 where the z-axis is pointed toward the ceiling, the x-axis is pointing in the same direction as the arm when all the joint angles are 0, and the y-axis is left to be determined by the right-hand rule. I defined 7 body frames that are each at the base of the links. such that the z-axis is always faced in the direction of the link. Below are the rotation matrices for the transitions between the body frames of each link. Let θ_i be the i'th joint angle.

$$^{W}R_0 = R_z(\theta_0)$$
$$^{0}R_1 = R_x(-\pi/2)R_z(\theta_1)$$
$$^{1}R_2 = R_y(\pi/2)R_z(\theta_2)$$
$$^{2}R_3 = R_y(\pi/2)R_z(-\theta_3)$$
$$^{3}R_4 = R_y(-\pi/2)R_z(\theta_4)$$
$$^{4}R_5 = R_y(-\pi/2)R_z(\theta_5)$$
$$^{5}R_6 = R_y(\pi/2)R_z(\theta_6)$$

For the translation vectors, they can be expressed as the following:

$$^{i}T_{i+1} = \begin{bmatrix} 0 \\ 0 \\ L_i \end{bmatrix}$$

for $0 \leq i \leq 5$ where L_i is the length of the i'th link. $^{W}T_0 = 0$

Using the above equations, the rotation matrix between the world frame and the i'th body frame can be computed recursively as follows:

$$^{W}R_i = \begin{cases} ^{W}R_{i-1}\ ^{i-1}R_i & \text{if } i > 0 \\ R_z(\theta_0) & \text{if } i = 0 \end{cases}$$

The translation vector between the world frame and the i'th body frame can also be computed recursively.

$$^{W}T_i = \begin{cases} ^{W}T_{i-1} + {^{W}R_{i-1}}\ ^{i-1}T_i & \text{if } i > 0 \\ 0 & \text{if } i = 0 \end{cases}$$

Figure 4 is a 3D plot for the Sawyer robot when all the angles are 0.

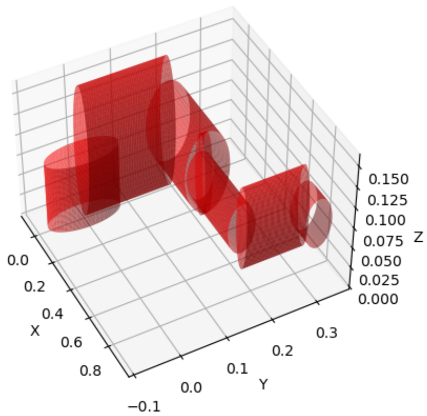

Fig. 4: Sawyer Plot

B. Obstacles

For simplicity's sake, I used spherical obstacles in the path planner. The spheres will be defined by a 3D Cartesian point in the world frame and a positive radius. Figure 5 shows an example.

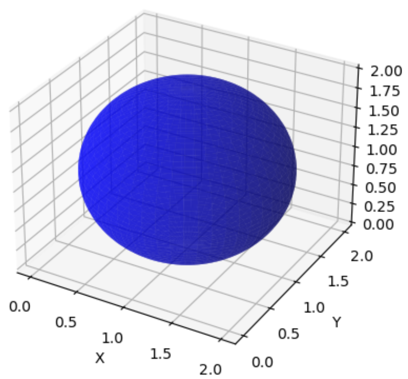

Fig. 5: Sphere Plot

C. Collision Detection

My method of detecting a collision between one of the Sawyer robots and an obstacle is based on the article written by Michael Sunker [7]. It goes as follows

1) Given a set of joint angles, compute the rotation matrices and translation vectors for every body frame.
2) For each cylinder:
   a) Perform a rigid body transform on the sphere's center so that the center is expressed in the cylinder's body frame. If p_{center} is the sphere's center, this could be done with the equation
   $$^{W}R_i^T(p_{center} - {}^{W}T_i)$$
   b) Determine which point on the body frame's z-axis (the cylinder's center line) is the closest point to the sphere. Denote this point p_{close}.
   c) Check if the distance between the sphere's center and p_{close} is less than the sum of the cylinder's radius and the sphere's radius.
3) It is assumed that the Sawyer robot will not self-intersect. Collision check is done.

D. Algorithm

The goal of the algorithm is to move the end-effector to a specified 3D coordinate. We can use Rethink Robotics's inverse kinematics service to compute the joint angles necessary to move the I then run RRT on the starting and ending joint angles [1]. The cost between two joint configurations is defined as

$$||\boldsymbol{\theta_1} - \boldsymbol{\theta_2}||^2$$

.

The steps for the RRT algorithm were taken from the original paper by Steven LaValle and the *Principles of Robot Motion* book [4] [5]. The high-level overview of the RRT algorithm is

1) Initialize a tree with the starting configuration as the root.
2) Pick a node q in the tree with some probability
3) Sample a configuration q_{rand} near q
4) Try to connect q_rand to q by moving as far up the line between q_rand and q as possible.
5) If it is possible to add q_rand to tree, add it.
6) Restart from 2 until the maximum number of iterations has occurred or a path to the goal has been found.

III. Experiment & Results

I ran the RRT planner on Cartesian coordinates (0.1, 0.5, 0.3). The maximum number of iterations was set to 10,000, and the step size between configurations was set to π. The threshold that decides whether a configuration is close enough to the goal was 2.25. The threshold and step size were determined through trial and error. On occasion, the planner would move too quickly or too slowly to produce meaningful results. If the step size is too large, the planner won't produce a smooth path. If the step size is too large, we could have incomplete exploration. If the step size is too small, a path will not be found in the maximum number of iterations. Similarly, a large threshold would end the planner too quickly, while a small threshold may never be reached in the number of iterations I specified. A spherical obstacle was placed in between the start configuration, denoted q_{start}, and the goal configuration, denoted q_{goal}. This is to test if my planner would be careless and plan a path through an obstacle.

Figure 6 shows a plot of the setup. The configuration in red is the q_{start} and the configuration in gold is the q_{goal}.

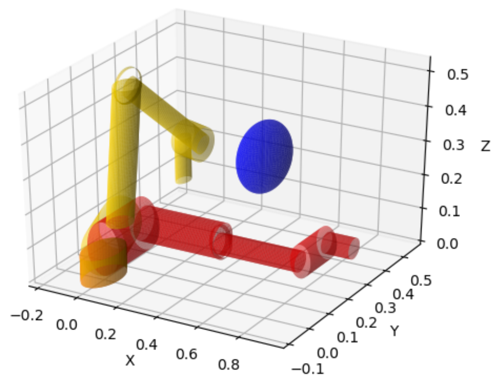

Fig. 6: World Plot

Though it takes some computational time, the RRT planner consistently finds a path between the q_{start} and the q_{goal}. A video demonstration can be found on the GitHub repository associated with this project.

IV. Discussion

Although RRT is not an optimal planner, it is probabilistically complete, has consistent behavior, and is relatively simple to code [5]. This sampling-based method is an excellent way to provide obstacle avoidance to Sawyer's inverse kinematics capabilities. This project also provides a baseline for optimization. For example, One could use RRT* to optimize the path between q_{start} and q_{goal} [4]. Another example could be using two RRTs with one being rooted at the q_{start} and the other could be rooted to q_{goal} and merge the two trees when they are close enough [4].

V. Conclusion

Sampling-based methods are proven to be viable options for implementing obstacle avoidance on the Sawyer robot. Continued development should be focused on optimizing the baseline RRT planner, either by reducing computational time or decreasing the total cost of the path. Overall, the project was successful in path planning with obstacle avoidance on Sawyer.

References

[1] Ik service example. `https://support.rethinkrobotics.com/support/solutions/articles/80000980360-ik-service-example`. Accessed: 2023-12-01.

[2] Sawyer hardware. `https://support.rethinkrobotics.com/support/solutions/articles/80000976455-sawyer-hardware`. Accessed: 2023-12-01.

[3] Pedro Goncalves da Costa. Motion planning in cartesian space for the collaborative redundant robot sawyer. *Masters Abstracts International*, 2019.

[4] K. M. Lynch Howie Choset and S. Hutchinson. *Principles of robot motion: theory, algorithms, and implementations*. Cambridge, Mass. Bradford, 2005.

[5] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. *The annual research report*, 1998.

[6] Min Cheol Lee Sun-Oh Park and Jaehyung Kim. Trajectory planning with collision avoidance for redundant robots using jacobian and artificial potential field-based real-time inverse kinematics. *International Journal of Control, Automation and Systems*, 2020.

[7] Michael Sunkel. Collision detection for cylinder-shaped rigid bodies. 2010.