Preliminary PDM Project Report: Group 3

Danish Ansari (5094755) Jasper van Brakel (5323215) Tyler Olson (5946182) Gijs Zijderveld (5306728)

***Abstract*— We test an adaptation of the RRT* algorithm a modified collsision checker as a global planner with using a holonomic high-five robot in a dynamic environment.**

I. INTRODUCTION

In this project, we implement a novel motion planner for a simulated holonomic mobile manipulator in the `gym_envs_urdf` environment[1]. Our robot has been tasked with navigating through a scene and giving a human a high-five. The workspace $\mathcal{W} \in \mathbb{R}^3$ consists of a few randomly generated static (and dynamic?) obstacles and (at least?) one goal region that the end effector (the robot's hand) must reach to start the high-five action. The robot is composed of a holonomic point base (provided by `gym_envs_urdf`) with a simple arm consisting of three cylindrical links with a hand on the end.

We have elected to simulate this toy robot to demonstrate our planning algorithm because it has a complex configuration space with an under constrained system. To facilitate development and testing within the given time constraints, we simplify the problem by using a holonomic base.

We will compare our collision checker with a classical one by testing the simulated robot in the same environment with each. This comparison will be done in three different test scenarios with varying degrees of complexity.

The first scenario involves a robot being positioned on one side of a wall that has a castle-like hole in the form of an arch. The objective is for the robot to find this hole and navigate through it to reach the goal. Placement of the wall and hole will be randomized over several trials. The primary goal of this test is to demonstrate the effectiveness of the modified RRT* algorithm and also to compare the performance of both algorithms in a simple environment. Navigating through narrow corridors is a difficult task for RRT algorithms so this should be a fair comparison.

In the second scenario, the environment will be slightly more complex. This can be achieved by introducing additional walls or creating smaller holes that the robot must maneuver through. The purpose of this scenario is to observe how the performance of both algorithms changes in a more intricate environment.

Lastly, the third scenario will feature multiple missions in the same (possibly dynamic?) environment. The intention here is to evaluate the benefit our method can provide in scenarios where there is already partial knowledge of obstacles in configuration space. Performance with the classical algorithm will also be recorded as a benchmark.

II. ROBOT MODEL

Figure 1 below shows the composition of the robot schematically in two different configurations. Additionally, the joints and joints frames are visualised.

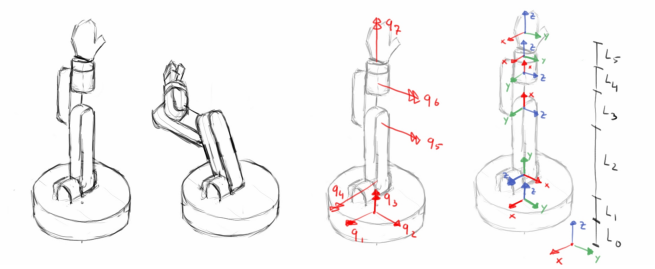

Fig. 1. Schematic drawings of the test robot.

The base is modelled as a planar joint which is always fixed to the floor and the arm has four joints, three of which have a limited range of motion. Thus, the configuration space $\mathcal{C}$ is isomorphic to $SO(2)\times\mathbb{S}^1\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\mathbb{S}^1$. This configuration of joints allows the end effector to freely move in all six degrees of freedom of 3D space.

To simplify the problem even further, the assumption is made that the robot will be able to perfectly follow the provided path. Therefore, only the forward and inverse kinematics are needed to control the robot, no additional controller will be implemented.

The kinematic equations are derived using the Denavit-Hartenberg (DH) parameters [2]. Table I provides the DH parameters for each link, which can be used to derive the homogeneous transformation matrices from the base of the robot to each other frame on the robot. The transformation between the frames connected to Link 4 could not be done in a single step using the DH parameters, which results in two sets of DH parameters for Link 4 as denoted in Table I.

TABLE I
DENAVIT-HARTENBERG MOBILE MANIPULATOR

Link #	d	θ	r	α
1	l_1	$\pi/2 + q_3$	0	$\pi/2$
2	0	$\pi/2 + q_4$	l_2	$\pi/2$
3	0	q_5	l_3	0
4-I	0	q_6	l_4	0
4-II	0	$\pi/2$	0	$\pi/2$
5	l_5	q_7	0	0

With seven joints and an arm with four linkages, providing the symbolic derivations of the homogeneous transformation

matrices becomes quite lengthy. Instead, the method used to retrieve the transformation matrices using the DH parameters in Table I is provided. (1) is used to determine the homogeneous transform matrix between two consecutive frames.

$$ {}^{i-1}_{i}\mathrm{T} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & r_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & r_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

The homogeneous transformation matrix between the world base frame and robot base frame is derived by hand: (See (2)).

$$ {}^{0}_{1}\mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & q_1 \\ 0 & 1 & 0 & q_2 \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

The homogeneous transformation matrix between any two frames can then be determined using the following property of homogeneous transformation matrices: (See (3)).

$$ {}^{0}_{3}\mathrm{T} = {}^{0}_{1}\mathrm{T}\,{}^{1}_{2}\mathrm{T}\,{}^{2}_{3}\mathrm{T} \tag{3} $$

The forward kinematics of the robot are now fully defined and can be used to retrieve the position of the robot base and the joint positions in Cartesian space using the robot configuration.

III. MOTION PLANNING

Algorithms which stem from RRT are fit for solving planning problems for mobile manipulators because they can easily sample in unstructured, high dimensional configuration spaces and do not rely on complete knowledge of obstacle locations in the workspace. The anytime property of RRT* can make such planning problems tractable, even if the solutions are sub-optimal. However, in practice, because RRT* is single-query using it alone is often too slow to respond to dynamic obstacles and requires use of a local planner to make corrections while the robot is moving. We implement a modified RRT* algorithm with the goal of speeding up collision checking enough to use a global planner in real time even for a robot with a high dimensional configuration space. This is achieved by lazily removing regions from the sampling space $\mathcal{S} \subseteq \mathcal{C}$ that are coincident with obstacles $\mathcal{C}_{obs}$.

The implementation of our collision checker is given in algorithm 1. Whenever a collision is detected, add a constraint $f : \mathcal{S} \mapsto \{True, False\}$ where $f(\mathcal{C}_{obs}) = True$ and *False* otherwise. All configurations of the robot which collide with the selected sample $s \in \mathcal{S}$ can be described by calculating the null space for the robot with a prismatic joint replacing the colliding link. This fictitious prismatic joint is limited to the length of the link it replaced. *The null space can then be described using parameterized vector addition so linear inequality constraints can be formulated in $\mathcal{C}$, making them quick to evaluate.* We can additionally perform the same evaluation for previous links in the chain, until s is out of reach of the reduced chain. *For now this is left out intentionally for future work.* Collision checking can be short-circuited by first testing if sample points conflict with any of the previously discovered obstacles $\mathcal{S}_{obs} \subset \mathcal{C}_{obs}$. Note that $\mathcal{S}_{obs} \cup \mathcal{S} = \mathcal{C}$.

Algorithm 1 Our proposed collision detection algorithm. Returns *True* if a collision is detected and *False* otherwise.

```
function CollisionChecker(sample s ∈ S, S_obs ⊂ C)
    discovered_obstacle_constraints = generate_constraints(S_obs)
    for all constraints f in discovered_obstacle_constraints do
        if f(s) then
            return True
    if classical_collision_checker(s) then
        S_obs ← formulate_nullspace(s)
        return True
    return False
```

By removing newly detected collision regions $\mathcal{S}_{obs}$ from $\mathcal{S}$, our implementation does not waste time evaluating sample points that are already known to cause collisions. Because the sampling space is shrunk with every collision, the probability of a future sample causing a collision monotonically decreases. Furthermore, if we have perfect knowledge of obstacle trajectories $\mathcal{S}$ can be carried between time steps, making it faster to generate new paths in the future. *It may be possible to transform each sample space obstacle forward in time if collision regions can be represented invariant to obstacle velocity.*

IV. RESULTS

V. DISCUSSION

REFERENCES

[1] M. Spahn, *Urdf-Environment*, version 1.0.0. DOI: 10.4121/19362017. [Online]. Available: https://github.com/maxspahn/gym_envs_urdf.

[2] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 22, no. 2, pp. 215–221, Jun. 1, 1955. DOI: 10.1115/1.4011045.