-A Bead Maze is not a kids' game-
Algorithmic Motion Planning - Final Project

Nir Manor 305229627, Ortal Cohen 308524875

06/2024

1 Introduction

This project aims to develop a solution enabling a UR5 robotic arm to accurately follow a predefined path, specifically modeled after a bead maze game. The challenge lies in translating a physical trajectory, which exists in the task space, into a series of coordinated movements in the robot's configuration space, ensuring the robot's end effector follows the bead maze path as closely as possible. The **task space** refers to the physical environment in which the robot operates, including objects, obstacles, and trajectories. For this project, the task space trajectory is the path of the bead maze prototype. Conversely, the **configuration space** is a mathematical representation of the robot's possible positions, defined by its joint parameters.

Addressing Local Minima

A significant challenge in trajectory optimization is the potential for the solution to converge to local minima, resulting in suboptimal paths. To mitigate this we used **splitting** which means that the trajectory is divided into smaller segments, each optimized individually. This approach reduces the complexity of the optimization problem and helps avoid local minima.

Distance Metrics

Two key distance metrics are considered for evaluating the path fidelity:

- **Hausdorff Distance**
  This metric assesses the maximum distance from a point on one path to the closest point on the other path, providing a measure of the overall deviation between the two paths.

- **Fréchet Distance**
  Unlike the Hausdorff distance, the Fréchet distance considers the sequence of points along the paths, offering a measure of similarity between two curves. Like a person walking a dog, where the person and the dog each follow a different path, the Fréchet Distance considers the most optimal way to walk both paths, looking at the shortest maximum "leash" needed to keep the person and the dog connected as they each move along their paths.

  These metrics serve as the foundation for the optimization problem, where the goal is to find a configuration space trajectory that minimizes the chosen distance metric relative to the task space trajectory.

  In our project we tried to minimize the Fréchet Distance.

2 The Process, Challenges and Solutions

Our project was divided into two main parts:

1. **Modeling the bead maze prototype**
   We have built a prototype of a bead maze path (a 3D path with a straight line and a curve) and therefore we needed to model the path in order to get the task space trajectory.

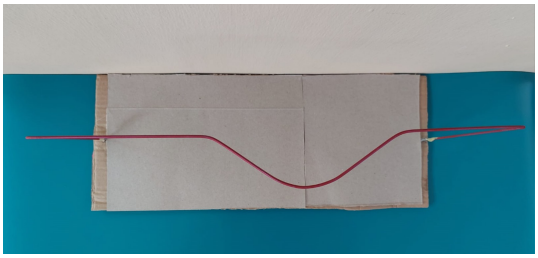


Figure 1: Top view of the 3D path

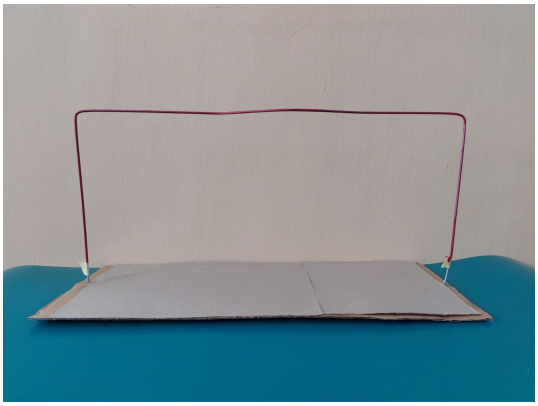


Figure 2: Front view of the 3D path

2. **Solving the optimization problem**
   The project's core involves formulating and solving an optimization problem where the objective is to minimize the distance between the desired task space path (the bead maze trajectory) and the actual path taken by the robot's end effector. This requires selecting appropriate distance metrics to quantify the "closeness" of the two paths.

Modeling the bead maze prototype

- **Methods**:
  - **Creating a 3D model from an Image**
    We created a script which takes as an input an image and process it to generate a 2D spline path. The script involved contour detection, spline fitting, scaling and trimming based on user input. For that we created one more model, as shown below:

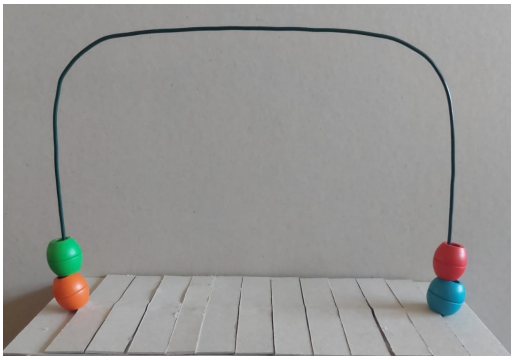


Figure 3: 2D path

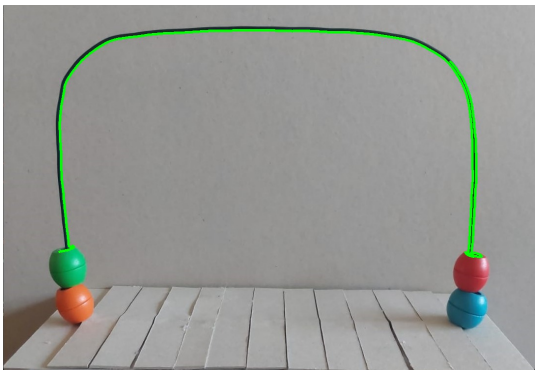


Figure 4: 2D path with the spline detection

We tried to create such a script to generate a 3D spline path but encountered lots of difficulties and the result wasn't good enough. Therefore we tried to create the model by sampling configurations in the lab.

- **Sampling configurations in the lab**
  We placed the robot on the prototype path and manually moved it along this path. Using an interface of UR5 we created a script to sample configurations, taking additional samples at critical points such as the big curve. To obtain the task space trajectory, we converted these configurations into coordinates using forward kinematics. Despite these efforts, we encountered issues, such as the robot disconnecting during remote control mode. This required us to sample multiple times and to process and clean the collected configurations.

  In the end we used the model which we created by sampling configurations in the lab.

Solving the optimization problem

- **Method**:
  Once we had a trajectory of the prototype path, the trajectory was divided into smaller segments (waypoints). And following that, we performed the following steps:

  - **Calculating possible configurations by inverse kinematics (IK)**
    For each waypoint and its tangent and with various normal options we computed possible IK configurations that are valid. A valid configuration means:
    * configuration is within angle limits $(-\pi, \pi)$
    * has no collisions between the links of the robot, no collisions with the prototype path which we defined as an obstacle, no collision with the floor and with the window.
    * the configuration is at most 1 cm from the path.

    The number of configurations per waypoint was checked against a threshold to limit the number of solutions.

    The end effector's orientation was crucial for ensuring it could effectively manipulate the bead along the maze path. The primary orientation requirement was for the Z-axis of the end effector to point perpendicular to the path's tangent vector while also pointing in the negative Y and negative Z directions to minimize collisions with the floor.

To achieve this, several arbitrary normal vectors were tested for each waypoint along the path. These vectors were used to calculate a cross-product with the tangent vector, yielding candidate normal vectors. The chosen normal vector, tangent, and binormal vectors were used to construct a rotation matrix that ensured the desired orientation. Additionally, different roll angles around the tangent vector were tested to provide further valid configurations, reducing the risk of collisions and maintaining the required orientation. This method allowed for the effective following of the path while avoiding obstacles and ensuring the end effector remains correctly oriented.

In the figures below you can see all the options of the normals we have considered:

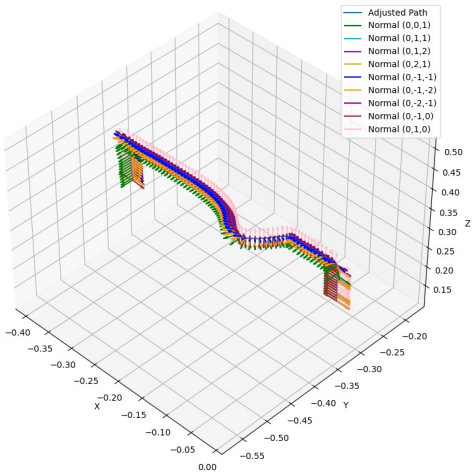


Figure 5: the 3D path with all the normals

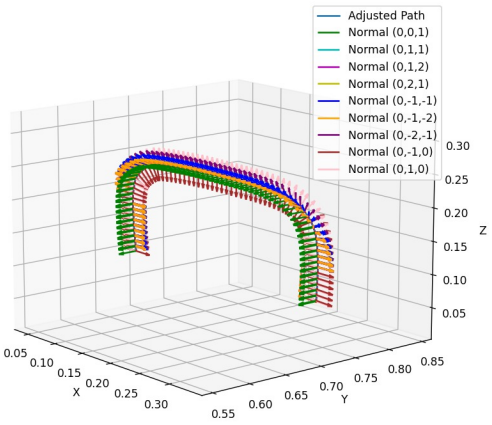
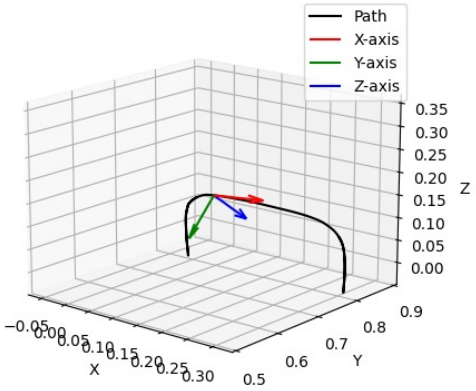
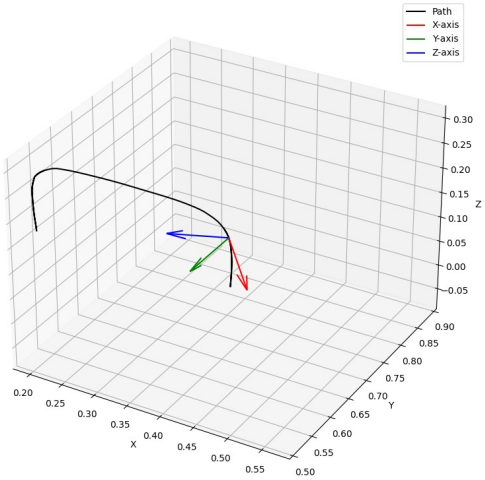
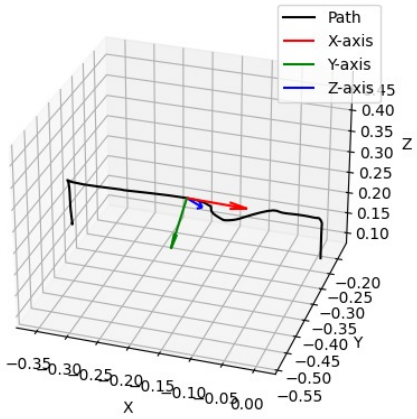


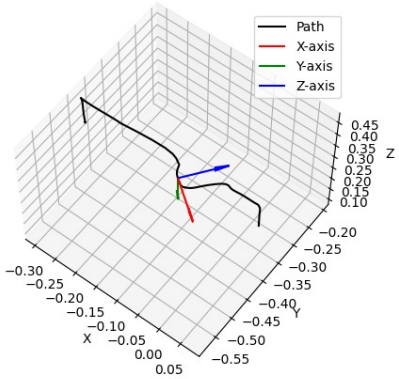
Figure 6: the 2D path with all the normals

In the figures below, you can see the tangents, normals, binormals, and the orientations of the end effector along the path:









– **Constructing the cross product graph**

  * Layer: all possible IK configurations for a given waypoint. Meaning layer i represents waypoint i.
  * Vertex: (layer i, a possible IK configuration for waypoint$_i$)
  * Edge: an edge was created between all vertices within the same layer and between each vertex in layer i to each vertex in layer $i+1$, while checking there is no collision.
  * Cost of an edge: the cost of an edge represents how far we deviate from the desired path by moving from one configuration to another.

– **Computing the bottleneck shortest path**

We implemented a Dijkstra-like algorithm, the bottleneck shortest path algorithm, and computed it on our cross-product graph. For each vertex in

the first layer, we ran this algorithm, and once it reached any vertex in the last layer, we saved the path and returned the path that had the minimum bottleneck value from all paths.

Computing the bottleneck shortest path corresponds to the path that minimizes the Fréchet distance between our reference path in the task space and the path which the robot takes in the C-space. This means finding the path that has a minimal maximal edge weight. In the bottleneck shortest path graph search, we calculated the cost of each vertex by taking the maximum of all edges that lead to each vertex. This ensures that the vertex cost reflects the maximum deviation from the desired path when transitioning between configurations.

We attached 4 GIFs of the robot moving along the 3D path which demonstrate a bit of the process and the improvements we did.

3 Results

As demonstrated in the attached video, the robot's end effector maintained an appropriate distance from the path for part of the time and if there was a bead on the maze it would have pushed it, but it occasionally deviated, being either too close or too far. We hypothesize that this may be due to issues within the path model itself. This problem indicates the need for further work on modeling the path, perhaps by using tools like vision.

Additionally, during lab tests of the robot's movement along the path, we were surprised to observe the robot's last link rotating 360 degrees, as shown in the video. Upon investigation, we found this occurs when the last value of the sixth link in two consecutive configurations is 0.*** in the first and 360.*** in the second. Consequently, the robot rotated nearly a full circle instead of making a small adjustment. Despite exploring various solutions, we have yet to resolve this issue. Throughout the process, we utilized a GIF animation to visualize the robot's movement along the path, however this visualization did not capture the rotation of the last link. Incorporating a GIF that also accounts for this rotational movement could have preemptively addressed this unexpected behavior. Additionally, employing a live simulator of the robot in the lab would be a valuable tool for testing and refining our path design and the robot's movements.

Aside from this, the robot moved smoothly without jumps or unnecessary movements along the path, except towards the end, where the robot did some jumps which we haven't resolved yet.

And of course we still need to create a better model of the path and better path for the robot in order to move a bead along the path :)

Note: we haven't attached a video of the robot moving along the 2D path because we had issues with finding a path when we were at the lab and we solved it only afterwards, therefore only a GIF is attached.