RBE550: Transmission

Tanish A. Mishra

INTRODUCTION:

NOTE: DO NOT RUN rrt_planner.py DIRECTLY, FOLLOW THE STEPS BELOW

The simulation environment used for this assignment is WeBots R2023a (https://cyberbotics.com) due to its collision detection capabilities which can be used on .stl files directly. This allows for a direct import of the OpenSCAD files into the simulation environment.

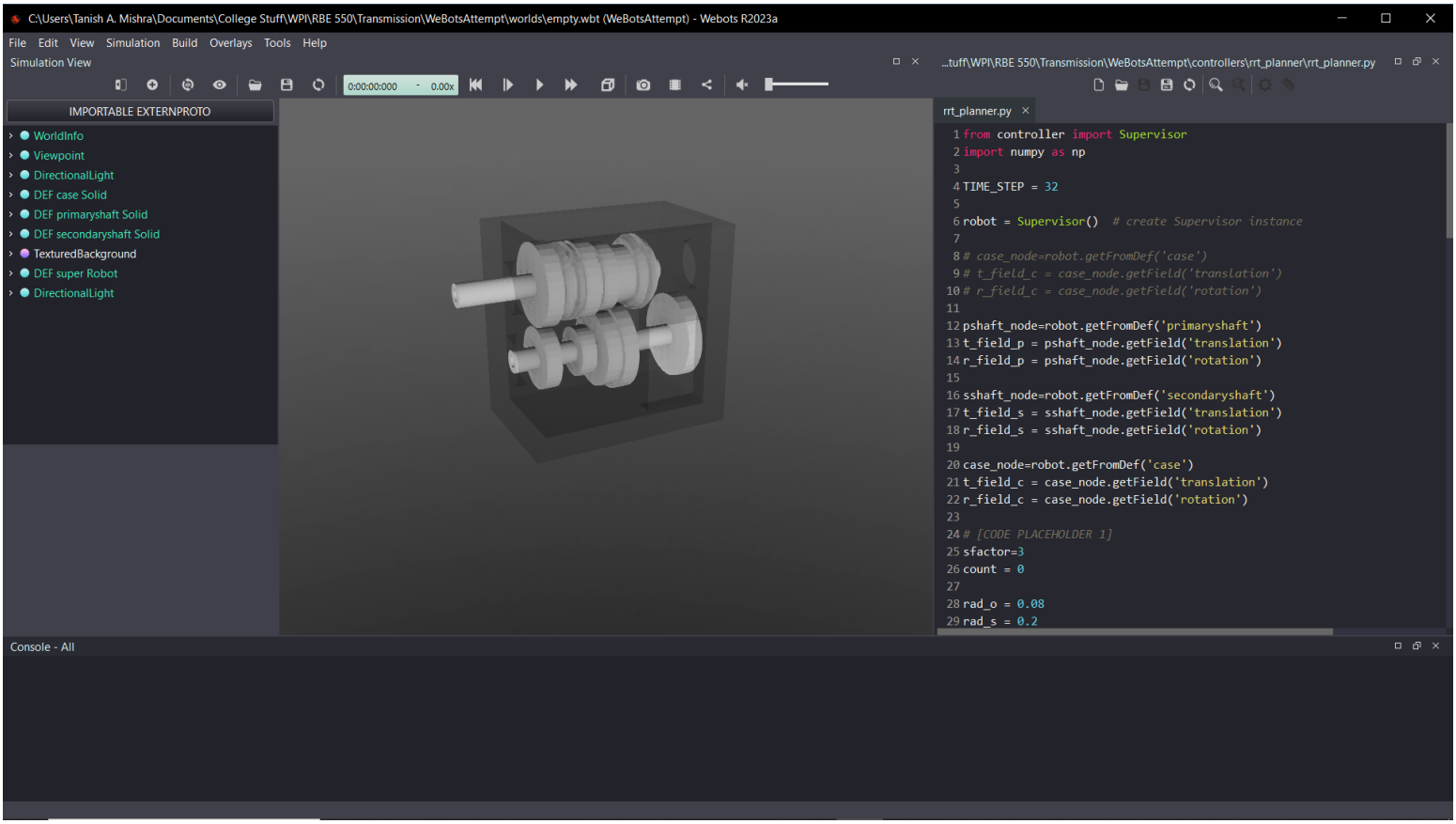

Figure 1: The simulation environment

To run this code, installation of WeBots R2023a is necessary, which can be done freely via https://cyberbotics.com. Once installed, unzip the source code folder and navigate to Transmission > WeBotsAttempt > worlds and double click on empty.wbt.

If the simulation screen displays “No Rendering” after the file opens, press Ctrl+4. Then press the play button on the toolbar on the top. To reset the simulation, press pause and then press rewind.

After pressing play, a few seconds (approx. 10-20) will be needed to generate the path, after which, the primary shaft should move on its own.

If any of the geometry is missing, it may need to be imported again (the necessary stl files are available in the Transmission folder). Instructions for the same are available in the documentation (https://cyberbotics.com/doc/guide/index).

For visualisation of the network and path, matplotlib was used after exporting all necessary coordinate values because WeBots does not possess strong plotting capabilities.

GRAPHS:

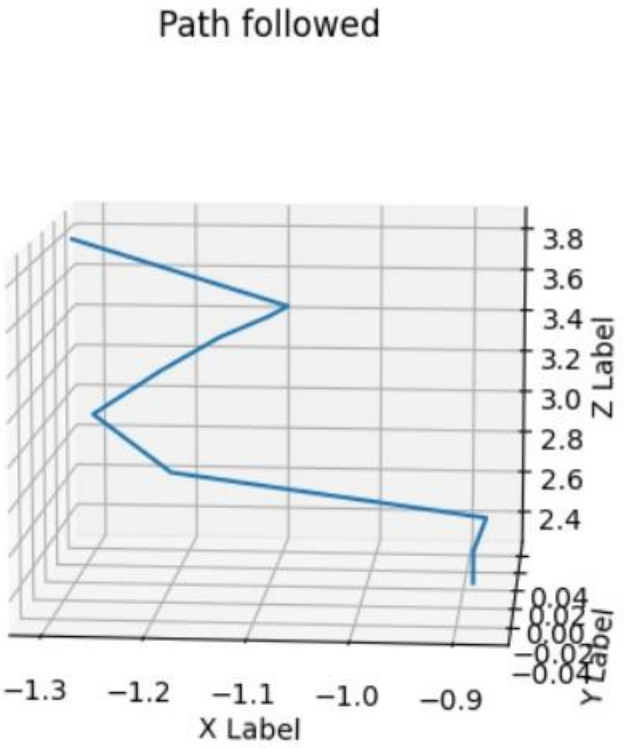

Figure 2: Actual path followed by a point on the shaft.

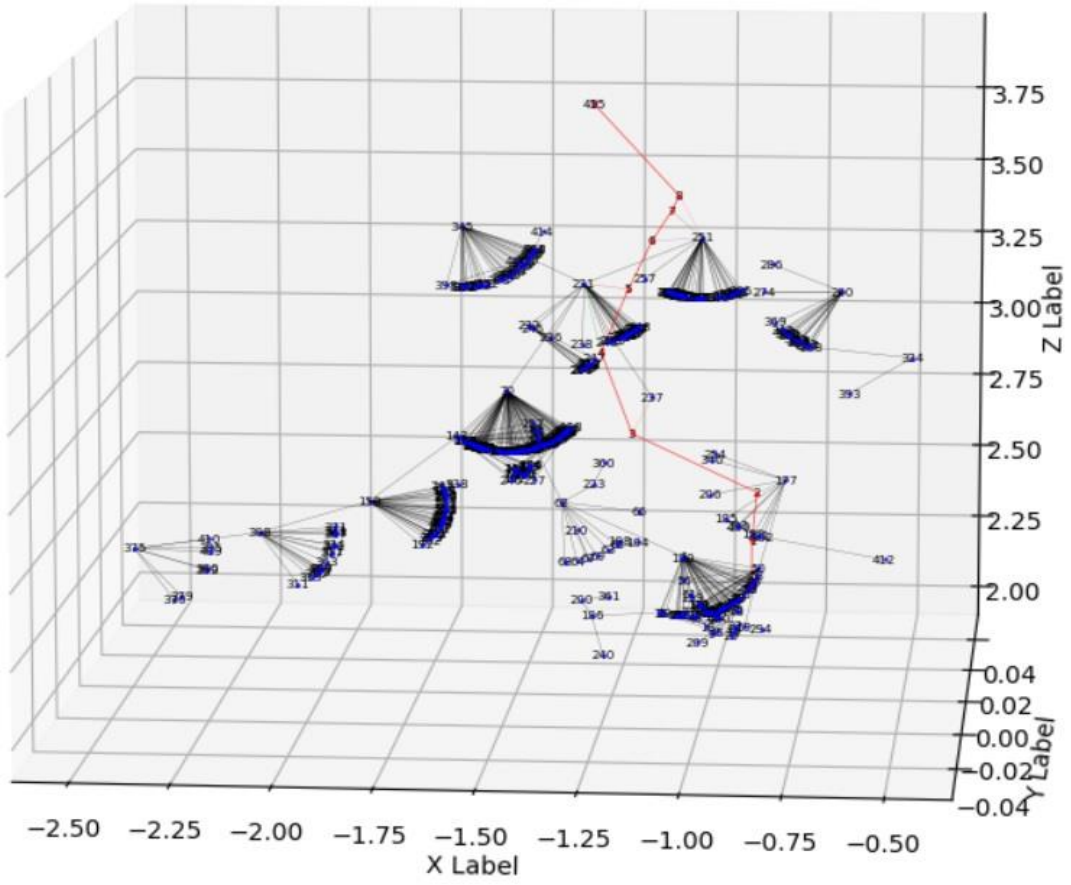

Figure 3: Complete tree graph generated using RRT.

CODE:

To implement RRT, the configuration space considered here is of cartesian coordinates and axis-angle representation of the orientation of the primary shaft. To speed up the computation, only rotations about one axis are considered (Y Axis in the simulation) since other rotations are not necessary for solving the problem. Additionally, only the x and z coordinates are considered for the same reason.

Axis angle representation was used because the functions in WeBots use the same.

For sampling, numpy.random.uniform is used. By specifying the range, we can sample a point in a particular reasonable region of the cartesian space. Angles can also be sampled in the same way independently.

Collision detection is carried out by first moving the primary shaft to a sampled position and orientation and then using the solidnode.getContactPoints() function which returns a list. If the length of this list is 0, it implies that the solid is not under any collisions and the sampled point is considered valid. If there is a collision, the shaft is returned to its previous state and a new point is sampled.

```
t_field_p.setSFVec3f(list(coord))
r_field_p.setSFRotation([0,1,0,angle])
pshaft_node.setVelocity([0,0,0,0,0,0])
robot.step(TIME_STEP)

if len(pshaft_node.getContactPoints())==0
    prevcoord=coord.copy()
    prevangle=angle.copy()
```

Figure 4:Collision detection