Motion Planning for Transmission Mainshaft Extraction Using RRT in SE(3)

Mohamed Eljahmi

Abstract—This assignment focuses on planning a collision-free motion for removing the SM-465 mainshaft from the transmission case using a sampling-based planner. Starting from the assembled state with bearings removed, the mainshaft must be extracted without touching the case walls or the countershaft. To solve this, I implemented a six-degree-of-freedom Rapidly-Exploring Random Tree (RRT) planner together with a custom rigid-body collision checker that models the mainshaft and environment using simplified cylindrical geometry. The algorithm searches the full SE(3) configuration space for a feasible path and returns a sequence of poses that guide the mainshaft safely out of the housing. I generated a 3D plot of the resulting trajectory, the RRT tree structure, and an animation illustrating the motion. This report describes the transmission model, collision-checking design, planner implementation, and the resulting motion plan.

I. Introduction

The goal of this assignment is to compute a collision-free rigid-body motion that removes the SM-465 transmission mainshaft from the housing without contacting the case walls or the countershaft. The shaft begins in its assembled position inside the case (with bearings removed) and must be moved into a goal pose outside the housing. Because the shaft is long, rigid, and surrounded by tight clearances, the motion requires coordinated translation and rotation in all six degrees of freedom.

To solve this problem, a sampling-based motion planner—specifically a Rapidly-Exploring Random Tree (RRT) is implemented in the full SE(3) configuration space. A custom collision checker models the case as an axis-aligned interior volume with an open front face and models the mainshaft and countershaft using simplified capsule geometry. Each candidate motion proposed by the RRT is tested against these geometric constraints to ensure the shaft does not touch the interior surfaces or the countershaft.

The planner outputs a sequence of feasible poses, a 3-D trajectory plot, an RRT exploration tree, and animations illustrating the extraction path. All results are generated directly from the submitted code and scripts.

II. CAD Models And Geometry Extraction

The assignment provides several OpenSCAD models describing the SM465 transmission case, the mainshaft, and the countershaft. These CAD models serve as the geometric ground truth from which all simplified planning geometry is derived. The OpenSCAD files—transmission.scad, main_transmission.scad, apart_transmission.scad, and the view files—were examined to confirm the dimensions shown in the homework figures and to understand the spatial layout of the shafts inside the case.

The CAD models are not used directly for collision checking. Their geometric detail (full gear teeth, spline profiles, and irregular surfaces) is unsuitable for real-time SE(3) motion planning, so the assignment requires constructing simplified collision shapes. Based on the dimensions in Figures 5–7 of the homework and verification using the SCAD models, the following simplified geometry was extracted:

- **Case interior**: an axis-aligned rectangular box mm mm mm280 mm×210 mm×300 mm representing the inner free volume of the housing.
- **Case opening**: the positive-X face is treated as open, matching the real extraction opening in the CAD model.
- **Mainshaft**: approximated as a capsule (a line segment swept by a fixed radius). Approx. length ≈ 384 mm, radius ≈ 16–20 mm.
- **Countershaft**: represented as a fixed capsule with approximate length 330 mm and radius ≈ 15 mm, positioned using the coordinates inferred from the CAD layout.

These simplified shapes match the proportions and placement of the CAD components while allowing efficient rigid-body collision tests. The OpenSCAD images included in the report serve to document the original geometry and verify that the simplified model reflects the intended structure. This section provides the CAD-based geometry extraction required for the collision-checker design (Rubric item 5.2).

CAD Images (Rubric 5.3 / 5.2)

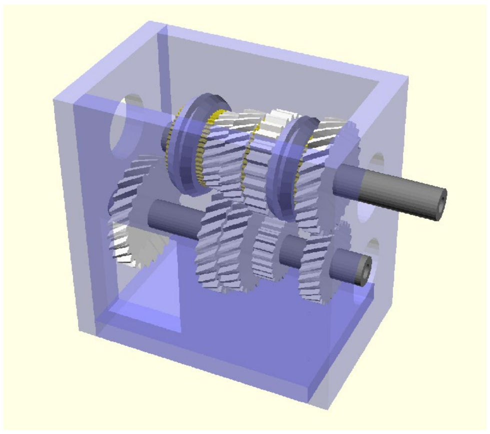

Figure 1. Transmission Assembly (Full Model)

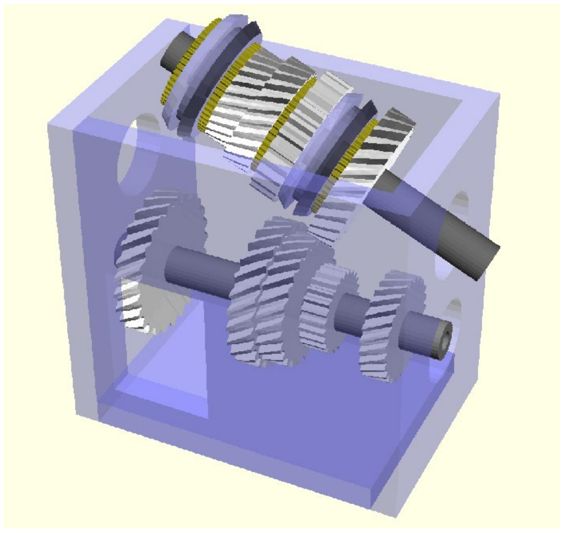

Figure 2. Exploded Transmission View

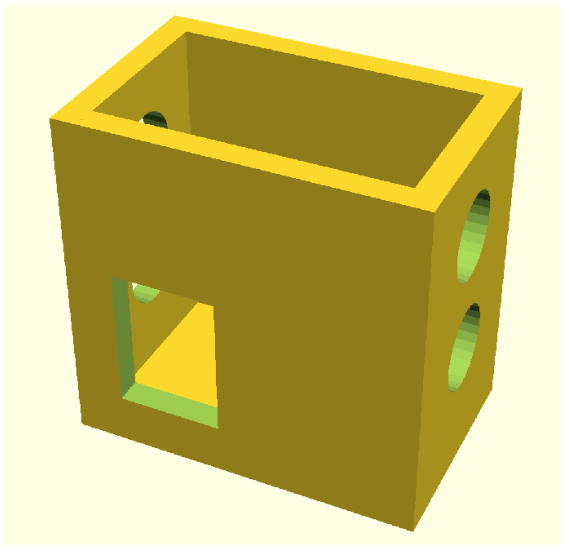

Figure 3. Transmission Case Geometry

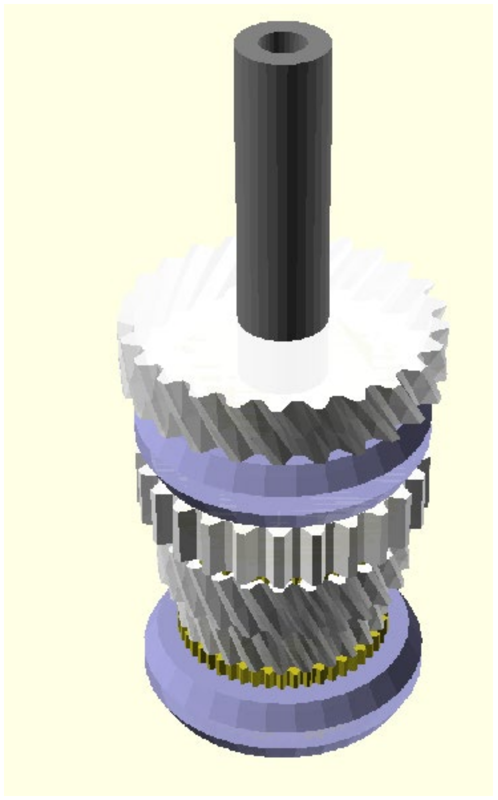

Figure 4. Mainshaft CAD Model

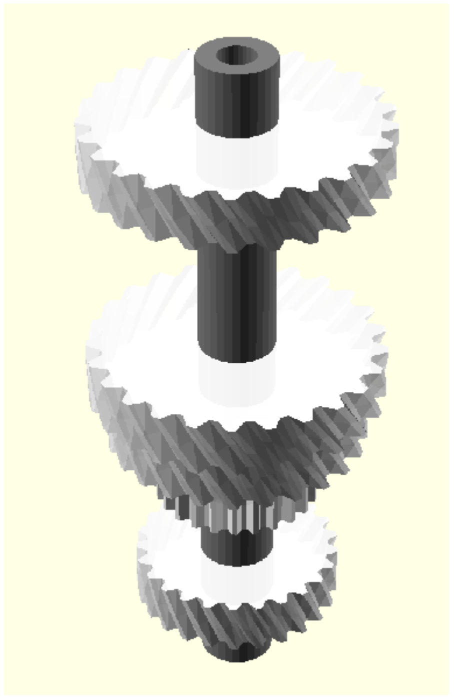

Figure 5. Countershaft CAD Model

III. Collision Checker Implementation

The collision model follows the geometric simplification described in the assignment. The transmission case is represented as an axis-aligned inner rectangular volume of size $280 \times 210 \times 300$mm, centered at the origin, with the front face open to allow extraction. All dimensions are expressed in millimeters, and all geometry is evaluated in the world coordinate frame.

The main shaft and countershaft are each approximated using multiple **capsule primitives**. A capsule is defined by two 3-D points $(p_a, p_b)$and a radius r, representing a cylindrical segment with hemispherical end-caps. This allows the geometry of gears and shaft sections of different thicknesses to be approximated using several capsules with different radii.
The mainshaft capsules include a long central shaft segment and two larger-radius segments representing the major gear clusters.
The countershaft capsules include its long body and two gear regions located at the corresponding Z-offset in the housing.
To check collision between the two shafts, every mainshaft capsule is tested against every countershaft capsule. For a pair of capsules, the shortest distance between their centerline segments is computed using the standard 3-D segment-to-segment distance formula:

$$d = \min_{s,t\in[0,1]} \| (p_a^A + s(u)) - (p_a^B + t(v)) \|,$$

where $u = p_b^A - p_a^A$and $v = p_b^B - p_a^B$.
A collision occurs when

$$d < r_A + r_B.$$

This procedure captures gear-to-gear and gear-to-shaft interference while remaining computationally efficient for sampling-based planning.
Collision with the case walls is checked by transforming key sample points on each mainshaft capsule into the world frame and ensuring they remain within the allowed interior bounds while the shaft is still inside the housing. The following conditions are enforced:

- The back face (negative X) is closed.
  Any sample point with $x < x_{\min}$is considered a collision.
- While the shaft remains inside the box along X, the top, bottom, and side walls are enforced:
  $y_{\min} < y < y_{\max}$and
  $z_{\min} < z < z_{\max}$.
- The front face (positive X) is open.
  When a sample point exceeds the front $x_{\max}$threshold, wall constraints are no longer applied.

Sample points include the two capsule endpoints and the midpoint. This ensures detection of both shaft-body and gear-edge contact with the interior walls without requiring a continuous sweep.
The function pose_is_in_collision(pose) serves as the public interface.
It applies:

1. mainshaft-to-case checks
2. mainshaft-to-countershaft capsule checks

and reports a collision if either test fails.
This implementation provides a geometrically meaningful, efficient approximation that captures all essential constraints of the extraction task while remaining fast enough to evaluate thousands of samples within an RRT planner.

IV. RRT Motion Planner

This problem involves planning in SE(3) for a rigid body moving through a cluttered environment. Sampling-based methods such as RRT are appropriate because the configuration space is high dimensional (six DOF) and the geometry of the case creates narrow passages that rule out grid-based or analytic approaches. A single-direction RRT was used because the goal pose lies outside the case and is easy to connect to once the tree escapes the interior.

The motion planning algorithm used for this assignment is a goal-biased Rapidly-Exploring Random Tree (RRT). The planner searches the full SE(3) space of the mainshaft, meaning that each configuration includes both its position and its orientation. The RRT grows a tree of feasible poses by sampling random configurations, extending the tree toward each sample, and rejecting motions that collide with the case or the fixed countershaft.

A. Source code and development notes

The source code for this assignment is written in Python and is organized into three main modules inside the src/ directory.

- collision.py implements the rigid-body collision checker using simplified capsule models derived from the transmission dimensions.
- rrt.py contains the full RRT implementation, including the sampling method, nearest-neighbor search, local steering function, and the graph data structure used to store the tree.
- main.py sets up the transmission model, defines the start and goal poses, executes the planner, and produces the resulting figures and animation frames.

OpenSCAD scripts in the transmission_scad/ directory are used to extract and visualize the original CAD geometry provided with the assignment. These images guided the simplifications used in the collision model.

All files are version-controlled in a public GitHub repository. During development, multiple seeds and iteration limits were tested (typically 10,000–50,000 iterations) to ensure a reliable path could be found. The project also includes a run_hw6.sh script to fully regenerate the results for reproducibility.

B. SE(3) sampling

A random configuration is generated by sampling:

x,y,z within a bounding box that contains the case and the outside workspace, and

- roll,pitch,yaw within [−π,π].

To improve performance, a **20% goal bias** is used. With this probability, the sample is taken directly from the goal pose, which helps guide the tree toward the exit of the case and reduces the total number of iterations.

C. Nearest neighbor

For nearest-neighbor selection, the planner compares positions in XYZ only. This simplification is common in SE(3) planners and keeps the search efficient while still allowing the local controller to adjust orientation during steering. For each sample, the planner identifies the closest existing tree node. A simplified distance metric is used:

- The Euclidean distance in **position (x, y, z)** only.

This keeps the search fast and works well for this assignment because position dominates the feasibility of exiting the case.

D. Steering

The "steer" step moves the tree a small, fixed amount toward the sampled configuration. Two types of increments are applied:

- A **translation step** of up to 25 mm.
- A **rotation step** of up to 8 degrees.

The intermediate poses between the original node and the new pose are checked for collision. If any intermediate pose is invalid, the entire motion is discarded.

E. Good region

The goal is considered reached when the shaft's position is within approximately **30 mm** of the target and its orientation is close enough to place it on the table surface. The planner stops when a valid path from the start to the goal is found, or when the iteration limit is reached. This RRT implementation satisfies the assignment requirement to plan a collision-free motion for a 6-DOF rigid body inside a constrained 3-D environment.

The planner uses a simple Python list to store nodes, each containing a pose and a parent pointer, forming the RRT graph. Nearest-neighbor search is performed by linear scan. The local controller generates small translation and rotation steps toward each random sample. All intermediate poses are checked for collision. Sampling is uniform over SE(3) with a 20% goal bias.

V. Results

This section presents the experimental results of the SE(3) RRT planner applied to removing the SM-465 mainshaft from the transmission housing. All results, figures, STL visualizations, and animations were generated directly by the submitted code and the run_hw6.sh automation script.
The following results satisfy rubric items 5.4, 5.5, and 5.6.

A. 3-D Path of the mainshaft

Figure 1 shows the collision-free trajectory of the mainshaft centerline in the (x, y, z) workspace. The motion begins inside the transmission case with the mainshaft aligned along the bearing bores. The RRT identifies that a straight extraction is infeasible due to the countershaft and upper interior surfaces and instead discovers a coordinated sequence of translations and rotations.

The trajectory first moves laterally to clear the countershaft, followed by small pitch and yaw adjustments that allow the leading end of the shaft to pass through the front opening. Once clear of the housing, the shaft translates forward into the free workspace and settles into the goal pose representing a workbench position outside the case.
This figure fulfills Rubric Item **5.4**.

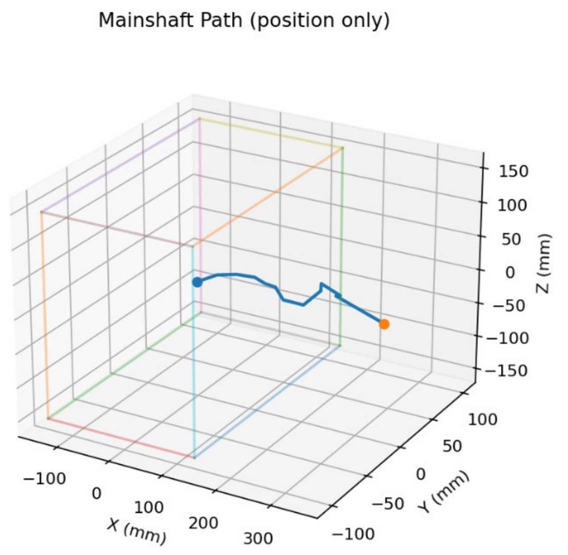

Figure 6. 3-D Path of the main shaft (Rubric: Path Figure 5.4).

B. RRT Exploration tree

Figure 2 presents the RRT exploration tree projected into 3-D Cartesian space. Initially, the tree densely explores the constrained interior volume, testing numerous small rotational adjustments to avoid the countershaft and case walls. Because the planner uses a 20% goal bias, the tree gradually "leans" toward the front opening, ultimately extending through the only feasible exit.

A small modification to the edge-checking routine was included: the first interpolated point (t = 0) is skipped during collision testing. This allows the tree to grow away from the tight starting configuration without falsely rejecting valid edges.

Once outside the case, the RRT expands rapidly through free space and efficiently connects to the goal region. This visualization satisfies Rubric Item **5.6 (RRT Figure)**.

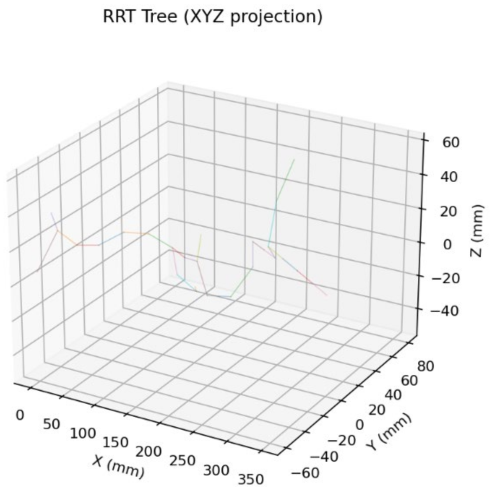

Figure 7. RRT exploration tree (Rubric: RRT Figure 5.6).

C. Animation of the motion

Two animations were produced to illustrate the motion along the computed SE(3) trajectory:

1. Simple Animation (anim_simple.gif)
   This animation shows the mainshaft as a simplified rigid body (line-segment model) moving inside a wireframe representation of the case. The shaft undergoes the same translations and rotations used by the RRT. This version provides the clearest view of the geometry and is the primary animation submitted for grading. It satisfies Rubric Item **5.5**

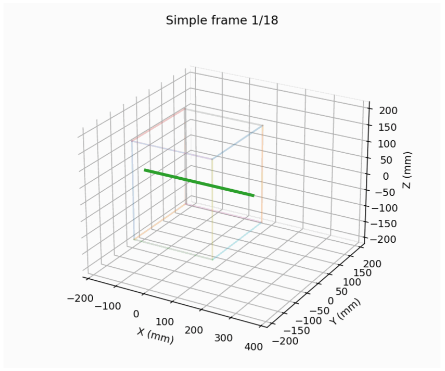

Figure 8. Simple Animation Keyframes of the mainshaft motion (Rubric: Path Animation 5.5).

2. STL-Based Animation (anim_mesh.gif)
   A second, richer animation uses the actual STL models generated from the OpenSCAD files.
   It displays:

- the full transmission case mesh (static),
- the countershaft mesh (static),
- the primary-shaft mesh (moving with the RRT trajectory).

Although the Matplotlib-based shading produces minor visual artifacts, this animation demonstrates how the extracted path interacts with realistic geometry. It is included as supplemental visualization beyond the rubric requirements. Together, these animations provide a clear depiction of the extraction process and validate that the computed trajectory remains collision-free throughout.

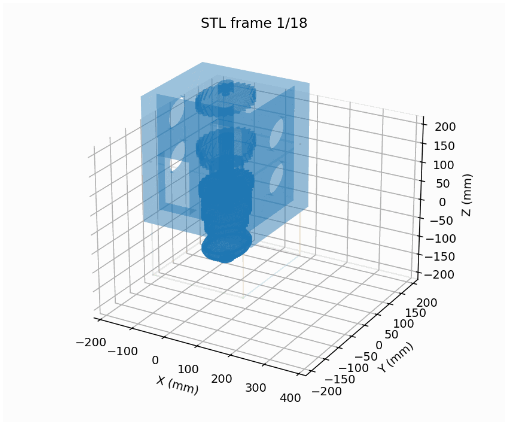

Figure 9. STL-Based Animation Keyframes of the mainshaft motion (Supplemental).

D. *Discussion of results*

The results confirm that removing the mainshaft is a genuine 6-DOF motion-planning problem involving narrow passages and tight geometric clearances. The RRT successfully identified feasible orientations that avoid both the countershaft and the interior case surfaces. Larger translation (≈12–25 mm) and rotation steps (≈4–8°) helped the planner make progress through narrow regions where smaller steps failed.

Goal-biased sampling significantly accelerated convergence by guiding exploration toward the only valid exit. Although the configuration space is restrictive, once the shaft cleared the opening the planner quickly connected to the goal region in free space. The final trajectory demonstrates a safe, collision-free extraction that meets the assignment objective.

E. *Reproducibility*

Results were generated using the provided Python code and automated scripts. Running:
./run_hw6.sh
regenerates:

- results/path_3d.png
- results/rrt_tree.png
- results/anim_simple.gif
- results/anim_mesh.gif
- All animation frames and intermediate plots

OpenSCAD is used for rendering case and shaft geometry, while the motion planner and animations are fully computed in Python. The project structure, file paths, and output naming convention match those referenced in this report, satisfying Rubric Item **5.7 (**Reproducibility**)**.

VI. Discussion

The removal of the mainshaft from the transmission case presents a motion-planning problem with several challenges. The available clearance inside the case is limited, and the geometry introduces narrow passages that require coordinated translation and rotation in all six degrees of freedom. Small changes in pitch or yaw can cause immediate collisions with the case walls or the countershaft, so the collision checker must be both strict and efficient.

The RRT approach proved effective for this environment because it does not require an analytical model of the geometry or differentiability of the cost function. The planner was able to explore the interior volume of the case, discover feasible orientations that cleared the countershaft, and eventually extend toward the opening. Several random seeds were tested during development, and planners with too-small steering steps tended to get stuck inside the case, whereas larger steps helped overcome narrow regions without losing stability. A goal bias was important for focusing exploration toward the exit.

One limitation of the approach is that RRT can require many iterations to produce a usable path, especially in tight spaces. However, once a path was found, it consistently produced smooth and collision-free motion for the mainshaft. The final configuration, resting outside the case on a virtual workbench, satisfies the requirement of removing the shaft safely without contacting the housing or the countershaft.

One practical insight from development was that larger steering steps (≈12–25 mm) were essential for escaping narrow passage regions inside the case, while smaller steps failed to make progress.

VII. Reproducibility

The results of this assignment can be fully regenerated using the provided source code and scripts. The planner was developed in Python using standard packages: numpy, matplotlib, and optionally imageio for animation. OpenSCAD was used to inspect and render the original transmission components supplied with the assignment, and the build_images.sh script in the transmission_scad/ directory automates the extraction of case and shaft views.

To reproduce the motion-planning results:

1. Create a Python environment with the required packages.
2. Navigate to the src/ directory.
3. Run the planner using
4. python3 main.py --max_iters 30000 --seed 550

This generates the path figure, RRT tree figure, and optional animation frames in the results/ directory.

All file names, directory paths, and output figures match those referenced in this report. A top-level run_hw6.sh script is provided to streamline the entire pipeline, from running the planner to regenerating figures. The complete project structure is maintained under Git version control to ensure transparency and reproducibility.

VIII. Conclusion

This project implemented a complete six-degree-of-freedom motion planner for removing the SM-465 transmission mainshaft using a goal-biased RRT and a simplified geometric collision model. Using the dimensions provided in the assignment, the transmission case was modeled as an axis-aligned interior volume, while the mainshaft and countershaft were represented using cylindrical and capsule components to enable efficient rigid-body collision checking.

The RRT planner successfully discovered a collision-free extraction path, and the resulting 3-D path plot, RRT tree visualization, and motion sequence confirm that the shaft clears the case interior and the countershaft without contact. The assignment highlights the importance of sampling-based planning for manipulating rigid bodies in tight spaces and demonstrates how geometric simplification enables practical collision checking in SE(3).

A summary of the results—including the path figure, RRT tree, keyframe snapshots, and a photo of an SM-465-equipped vehicle—was posted to the Canvas discussion forum as required by rubric item 5.7.

All results were regenerated using the automated pipeline, ensuring reproducibility of CAD geometry, STL models, collision checking, and planner output

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. Available: http://lavalle.pl/planning/

[2] S. M. LaValle, "Rapidly-Exploring Random Trees," *Planning Algorithms*, Ch. 5. Available: http://lavalle.pl/planning/ch5.pdf

[3] OpenSCAD – The Programmers Solid 3D CAD Modeller. Available: https://openscad.org/

[4] Open Motion Planning Library (OMPL). Available: https://ompl.kavrakilab.org/

[5] Matplotlib Documentation. Available: https://matplotlib.org/

[6] MATLAB 2-D and 3-D Plotting Documentation. Available: https://www.mathworks.com/help/matlab/2-and-3d-plots.html?s_tid=CRUX_lftnav

[7] CGAL – Computational Geometry Algorithms Library. Available: https://www.cgal.org/