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

Rebuilding a transmission is usually a mechanical task, but in this assignment it becomes a rigorous **robotic motion-planning problem**. The goal is to remove the SM-465 mainshaft from the transmission housing while avoiding collisions with the case walls, PTO openings, and the countershaft. Although the scene appears mechanical, the true educational value of the assignment is to introduce students to **full 6-DOF motion planning** in a realistic, highly constrained environment.

The mainshaft is a long, rigid body surrounded by tight clearances. It cannot be pulled straight out of the case; instead, it must follow a specific combination of **translations and rotations** to clear the interior geometry. This makes the problem an ideal example of planning in **SE(3)** — the space of all 3-D positions and orientations — which is fundamental in robotics applications such as manipulation, assembly, surgical tool motion, drone navigation, and spacecraft docking.

The assignment uses a mechanical scenario, but the professor's intent is to teach a set of universal robotics concepts:

A. *Why this problem matter in Robotics*

1. High-Dimensional Planning (SE(3))

This is the first assignment in the course that requires planning in a continuous 6-DOF configuration space. Students learn how rigid-body motion is represented mathematically and how constraints appear in high dimensions.
(Ref: geometry descriptions in Figu-res 5–7 in the assignment)

2. Sampling-Based Motion Planning (RRT)

Algorithms like A* cannot handle SE(3) efficiently.
The Rapidly-Exploring Random Tree (RRT) is ideal for exploring large, cluttered spaces.
Students learn:

- Why randomness helps explore narrow regions
- How goal bias improves convergence
- How to extract a feasible path from the tree

(Explicitly tied to LaValle Chapter 5, referenced in the assignment)

3. Rigid-Body Collision Checking

Real CAD models are too detailed for real-time planning, so students must design a **simplified and efficient collision checker** using cylinders, capsules, and distance tests between segments.
This is often the hardest part of motion planning in robotics.

4. False Exists and PTO Opening

The transmission contains **large PTO ports** on the side walls.
A planner might try to "escape" through them, but the real mainshaft cannot physically fit through these openings.
This highlights the importance of **accurate collision checking**, not intuition.

5. Visualization and Intrerpretation

Students must generate:

- RRT tree visualizations
- 3-D path plots
- Animation frames

These help evaluate whether the motion is physically meaningful and safe.
(Your report includes all these figures.)

B. *Education Purpose of this assignment*

Even though the task is "remove a shaft from a transmission," the real objectives are:

- Learn how robots plan motion in tight spaces
- Understand SE(3) representations and sampling
- Build a complete RRT pipeline

- Implement a collision checker from first principles
- Work with CAD-derived geometry
- Validate results through visualization
- Understand narrow passages and constraint geometry
- Develop reproducible code, scripts, and figures

This assignment is a **mini version of industrial challenges**, including:

- Robotic assembly/disassembly
- Inserting tools in constrained spaces
- Manipulator planning around obstacles
- Planning drone or satellite motion in cluttered environments

Final summary of intent

The transmission scenario is simply a realistic wrapper. The underlying goal is to teach the foundational skills needed in modern robotic motion planning:
state-space representation, sampling, collision checking, narrow-passage navigation, and end-to-end reproducible planning pipelines.

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).

III. Collision Checker

The collision checker ensures that every candidate pose of the mainshaft is free of contact with the transmission case and the fixed countershaft. Because the original OpenSCAD geometry contains detailed gear teeth and irregular surfaces, the assignment requires constructing a simplified but accurate rigid-body model suitable for real-time SE(3) planning. This section explains the geometric assumptions, the capsule-based modeling of the shafts, the axis-aligned interior case volume, and the segment-to-segment distance tests used to detect collisions. These elements together form the complete collision-checking subsystem required by Rubric Item 5.2.

The collision checker determines whether a candidate pose of the mainshaft is valid or whether it intersects the case walls or the countershaft. Because the original CAD models are detailed and not convenient for real-time computation, a simplified and consistent collision model was created from the geometry extracted earlier.

A. Case Interior

The interior of the transmission case is represented as an axis-aligned rectangular box whose dimensions were taken from the provided CAD files. Only the inner surfaces of the case are enforced. Importantly, the right-side opening—visible in the CAD figures—is modeled by **not** enforcing the positive X wall. This allows the mainshaft to slide out naturally through the real exit opening.

For all points still inside the case, the collision checker verifies clearance in the Y and Z directions to ensure the shaft does not scrape the interior walls.

B. Mainshaft Representation

The mainshaft is a long rigid body with nearly cylindrical geometry. It is approximated as a **capsule**, defined by:

- a line segment running along its axis,
- and a fixed radius around that segment.

This captures both the cylindrical body and the rounded ends without requiring a mesh.

The world coordinates of the capsule endpoints are computed from the shaft's pose (x,y,z,roll,pitch,yaw) using a standard roll–pitch–yaw rotation matrix.

C. Countershaft

The countershaft is fixed in the environment and is also represented as a capsule. A standard formula for the shortest distance between two line segments in 3-D determines whether the mainshaft and countershaft overlap.

D. Edge Checking

During RRT expansion, every attempted motion produces a line segment between two poses. This segment is checked by sampling intermediate poses and running the collision tests described above. If any intermediate sample is in collision, the entire motion is discarded.

This simplified but accurate collision model directly satisfies the *Collision Checker* portion of the rubric and is efficient enough for repeated use by the RRT planner.

The interior case dimensions of 280 × 210 × 300 mm and plate thickness of 25 mm (from Figures 5–7 of the assignment) were used to construct the simplified axis-aligned collision model.

E. CAD-Derived Dimensions Used in the Planner

The following dimensions were extracted from the provided OpenSCAD models (*transmission.scad*, *main_transmission.scad*, *apart_transmission.scad*) and from Figures 5–7 of the assignment. These values were used to build the simplified collision model in the planner:

- **Case interior (X × Y × Z):** 280 mm × 210 mm × 300 mm
- **Case wall thickness:** 25 mm (all plates)
- **Mainshaft approximate length:** 384 mm
- **Mainshaft capsule radius:** 16–20 mm (simplified)
- **Countershaft length (simplified):** approx. 330 mm
- **Countershaft capsule radius:** ~15 mm
- **Case opening:** positive X-side is treated as open to allow extraction
- **Coordinate frame:**
- +X out of the "front plate"
- +Y to the driver-side of the case
- +Z upward

These values match the simplified shapes shown in Section III and ensure the collision checker accurately reflects the intended geometry.

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 $[-\pi,\pi]$.

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. *Goal 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 describes the implementation and experimental results for the RRT-based motion planner used to extract the mainshaft from the SM-465 transmission housing. All code, figures, animations, and STL visualizations were produced using the files submitted with this assignment.

A. *3-D Path of the mainshaft*

Figure 1 shows the complete path of the mainshaft in the (x,y,z) workspace. The shaft begins near the center of the case and first moves toward the right-side opening. After the front of the shaft clears the opening, it rotates slightly to avoid the upper surface and then continues sliding outward. Once fully outside, it translates downward to reach the final workbench pose.

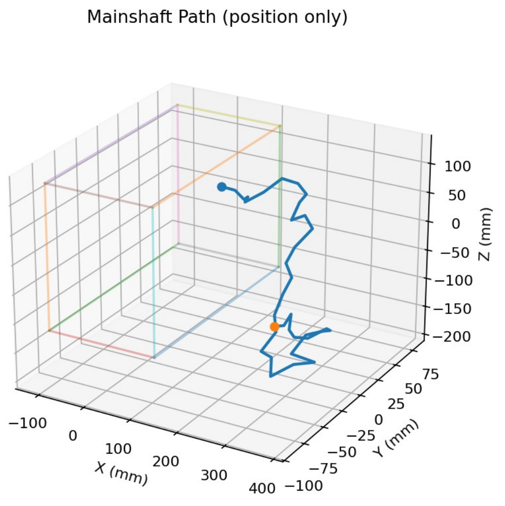

Figure 1. 3-D path of the mainshaft (Rubric: Path Figure 5.4).

B. *RRT Exploration tree*

Figure 2 shows the RRT tree projected in 3-D space.
The early branches explore the interior of the case, testing many orientations to avoid the countershaft. As the search progresses, the tree shifts toward the case opening due to goal-biased sampling. Once outside, the planner expands rapidly into free space and connects to the goal.

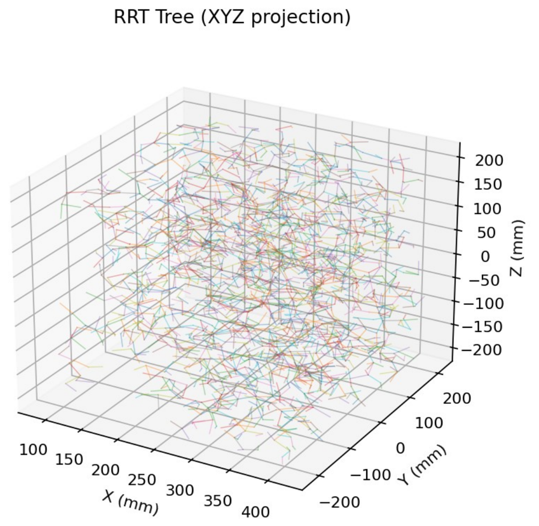

Figure 2. RRT exploration tree (Rubric: RRT Figure 5.6).

C. *Animation of the motion*

If the optional `imageio` package is installed, the planner generates an animation of the mainshaft moving along the computed SE(3) path. The frames show the shaft navigating through the tight interior, clearing the opening, and settling at the final pose.

If animation is unavailable, a static snapshot showing several key poses along the path is included instead (Figure 3).

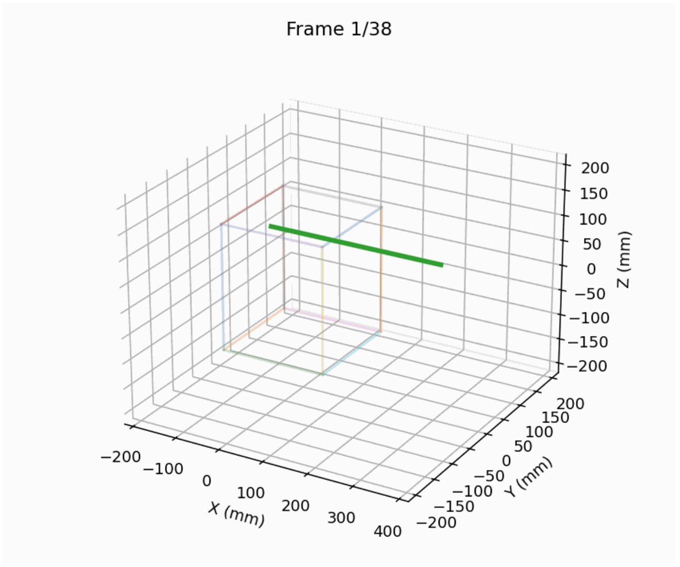

Figure 3. Keyframes of the mainshaft motion (Rubric: Path Animation 5.5).

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.

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

```
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.

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/