RBE 550: Transmission Assignment

Azzam Shaikh
April 22, 2024
GitHub repo: github.com/azzamshaikh/rrt_planner

I. Introduction

The purpose of this assignment is to gain experience developing an RRT-based motion planner. The implementation of the planner will focus on planning a 6-DOF path that removes the mainshaft of a SM-465 transmission out of its housing without colliding with the main case or the installed countershaft. Figure 1 shows an image of the SM-465 transmission.

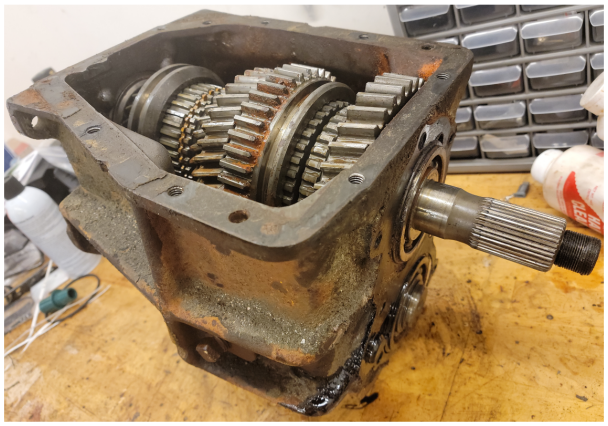


Fig. 1. SM-465 transmission [1].

The implementation of the simulation environment was developed using MATLAB 2023 and heavily utilized the Robotics Systems Toolbox due to its out-of-the-box 3D collision detection capabilities.

II. Methods

The methods used to model the transmission components - such as the case, the countershaft, and the mainshaft - and the implementation of the RRT-based planner will be discussed.

A. Modeling the Components

For the overall modeling of the assignment, since 3D collision detection was required, MATLAB and the Robotics Systems Toolbox was utilized. The Robotics Systems Toolbox contains the capability to model robotic systems and generate collision bodies via primitive, out-of-the-box shapes or via custom meshes [2]. These collision bodies can be used with various functions to check if collisions exist between world objects and the robot itself [2]. Thus, in order to achieve appropriate collision detection, the transmission itself was modeled as a robot, or in this case, as a `rigidBodyTree` [2]. This object allows the creation of a robot that reflects the connections of robot rigid bodies and joints. In this case, the transmission can be modeled as a series of rigid, fixed joints and created as a `rigidBodyTree`. Further discussion on the movement and manipulation of the `rigidBodyTree` will occur in later sections. After determining the modeling of the transmission, the remaining portions - i.e. the transmission case and countershaft - could be modeled as collision objects in the world environment. With these models implemented in this manner, the main `checkCollision(robot,config,worldObjects)` function can be utilized to check collisions between the robot in a specific configuration against the world objects [2].

A simplified model of the transmission was provided, as shown in Figure 2, where the outer shell is the case, the bottom shaft is the countershaft, and the top shaft is the mainshaft that will be removed. This simplified model will be recreated in MATLAB in order to modeling of the different components. This will be discussed further below.

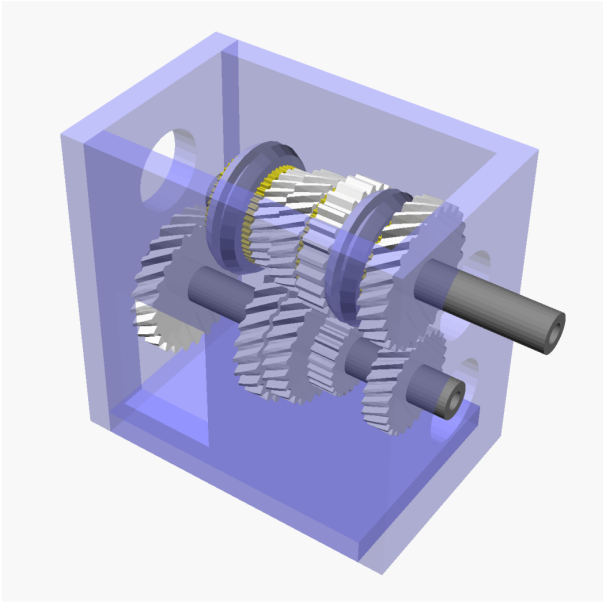


Fig. 2. Simplified SM-465 transmission [1].

1) World Objects: In order to model the world objects in MATLAB, it was initially attempted to import the CAD models via STL files and create a collision mesh. However, the output of the meshes were not entirely sufficient to represent the models. Thus, the models were recreated with the primitive collision objects available in the toolbox, the `collisionBox` and `collisionCylinder` objects.

For the case, each of the walls were recreated using the

dimensions provided in the assignment description. Based on Figure 8 from the assignment, the case walls which contained the shaft bearings were simplified from having circular cutouts to square cutouts, based on the available primitive shapes. The walls which contained the power take off drive mounting cutouts were modeled as basic rectangles. No cutout was applied in order to simplify the overall model. The overall housing can be seen in Figure 3.

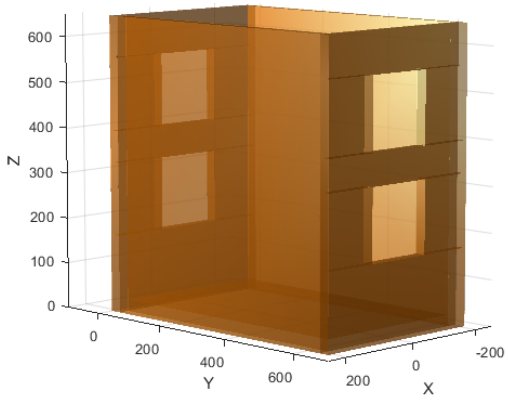


Fig. 3. Modeled case.

For the countershaft, the gears and shafts were all modeled as cylinders using the dimensions provided in the assignment description via Figure 7. The overall countershaft model can be seen in Figure 4.

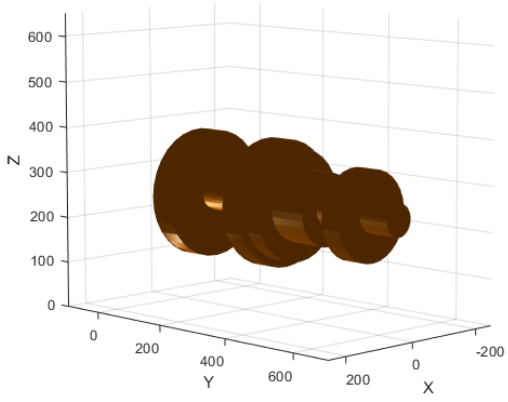


Fig. 4. Modeled countershaft.

With these two models completed, the resulting world object could be created, which is shown in Figure 5.

2) Mainshaft: As mentioned previously, the transmission can be modeled as a series of rigid, fixed joints and created as a `rigidBodyTree`. Thus, in order to create the model, the base of the robot model was selected to be the end of the mainshaft, i.e. the right most end of the mainshaft, as shown in Figure 2. Using this as the base, the remaining portion of the

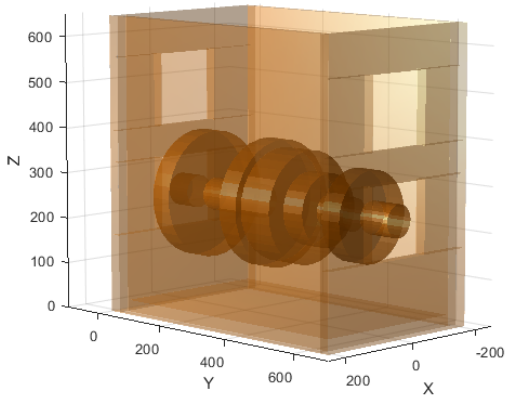


Fig. 5. Modeled case and countershaft world objects.

gears were modeled as fixed joints with `collisionCylinder` objects as its collision body. The placement of the fixed joints was accomplished using Denavit-Hartenberg (DH) parameters. The parameters were specified as initial angular offset to align the model with the world environments and linear translations, based on the dimensions provided in the assignment description via Figure 6. The overall mainshaft model can be seen in Figure 6.

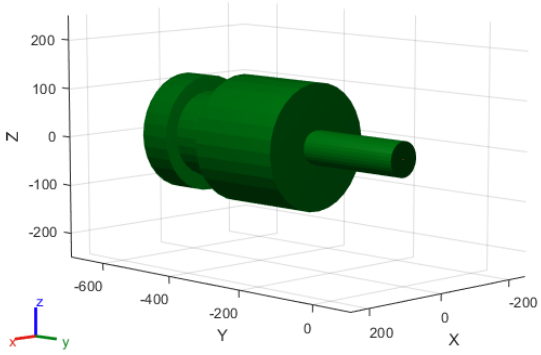


Fig. 6. Modeled mainshaft.

In order to move and position the transmission in the context of the world objects, the method described via a MATLAB Answers post was followed [3].

With this approach, the transmission can be moved as needed. The final model assembly can be seen in Figure 7.

As a sanity check, it was tested to see if the collision detection worked as required. When placing the transmission in the position shown via Figure 8, when running the `checkCollision(robot,config,worldObjects)` function, a true value was output. This is expected as the transmission is in collision with the case.

Once the modeling was complete, the RRT planner could

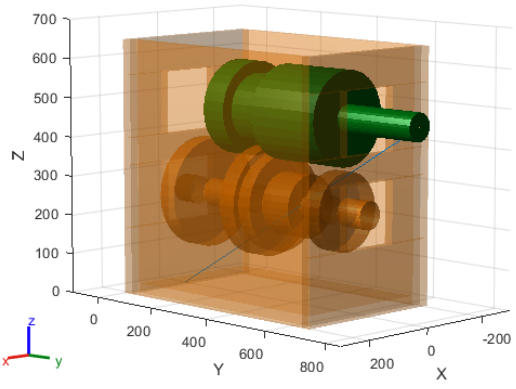


Fig. 7. Final transmission modeling.

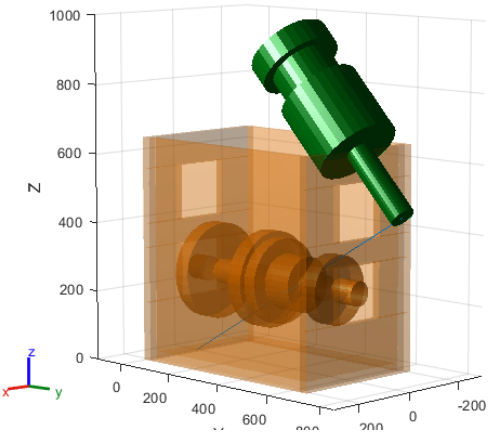
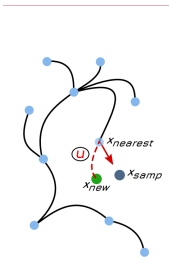


Fig. 8. Sample collision check position.

be developed to find a path to remove the transmission from the case without any collisions.

B. RRT Implementation

In order to implement the RRT planner, the basic RRT algorithm described in the Week 11 lecture was followed. Figure 9 shows the algorithm.




```
BUILD_RRT(X_init, X_goal)
    T.INITIALIZE(X_init)
    while T < maximum tree size do
        X_sample ← SAMPLE(X)
        X_nearest ← T.FIND_NEAREST(X_sample)
        X_new ← LOCAL_PLANNER(X_nearest, X_sample)
        if COLLISION_CHECK() then
            T.ADD(X_new) with an edge from X_nearest to X_new
            if X_new ∈ X_goal then
                return SUCCESS, motion to X_new
            end
        end
    end
    return FAILURE
```

Algorithm 1: Basic RRT algorithm

Fig. 9. RRT algorithm from Week 11 lecture [4].

Based on Figure 9, the implementations of the various functions will be discussed.

1) Initialize: In the initialization of the RRT, the initial start position will be taken as an input and added to a list of nodes. This list will be used to store the nodes added via the RRT algorithm.

2) Sampling: The output of the sampling function should be a node that gets selected from the free space. To accomplish this, given a range of x, y, z values (in the free space), and rotation limits, the function will randomly pick a 3D position from the free space and output it as the sample.

3) Find Nearest: The output of the find nearest function should be a node from the initialized node list that is closest to the sampled point. To find the closest node, for each node in the node list, the Euclidean distance between the sampled point and the node will get calculated and the node with the smallest distance will be selected and output as the nearest node.

4) Local Planner: The output of the local planner will be a new node that is closest to the sampled point. Generally, this would require a motion model of the robot to generate a path that can best reach that node, to create the new node. However, since this transmission does not have any controls and can move freely within its 6-DOF, a linear motion was selected. Effectively, a vector was created from the sample pose to the nearest node. Using this vector, the motion toward the sample point would be in a linear trajectory at some ratio of the vector. This can be visualized in Figure 10. This linear motion approach ensures that the smaller portions of the free space are generated as nodes, relative to the nearest node, as opposed to the full distance. These smaller ratios ensure that sufficient collision checking can occur to find the appropriate path. Once the new position is found, it can be created as a new node.

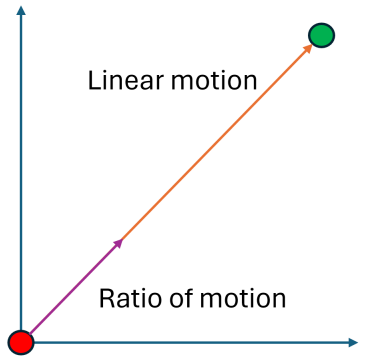


Fig. 10. Implementation of local planner motion.

5) Collision Checking: The collision checking for the 3D model was discussed in the mainshaft model section (II-A2). Effectively, within the RRT algorithm, the position of the transmission will be updated and a collision check will occur against the world obstacles. If there are no collisions, the new node will be added to the node list. If collisions are present, it will not add the node to the node list. This ensures only valid nodes are added to the models.

If the new node is within some threshold distance of the goal node, the planner will output the final path; otherwise, the planner will fail.

C. Simple RRT Check

After creating the supporting functions, a simple test was completed to check the validity of the planner. In a simple 3D space where the x, y, and z limits range from 0 to 5 and the start and goal positions are (0,0,0) and (5,5,5), respectively, the RRT algorithm output the following results.

Figure 11 visualizes the resulting tree structure that was searched and its associated nodes. The green x-marks indicate nodes that were sampled while the red x-marks indicate the nodes that were added to the tree. The blue lines represent the overall tree structure.

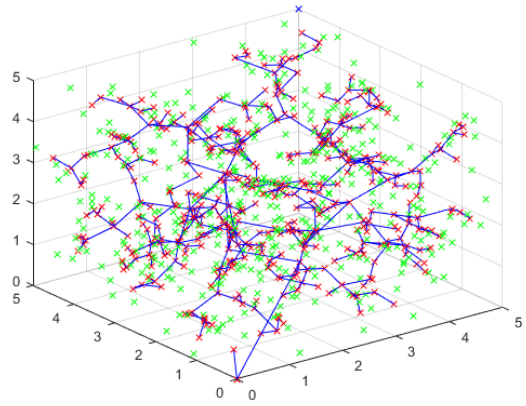


Fig. 11. Simple RRT tree.

Figure 12 visualizes the resulting path found in the resulting tree structure.

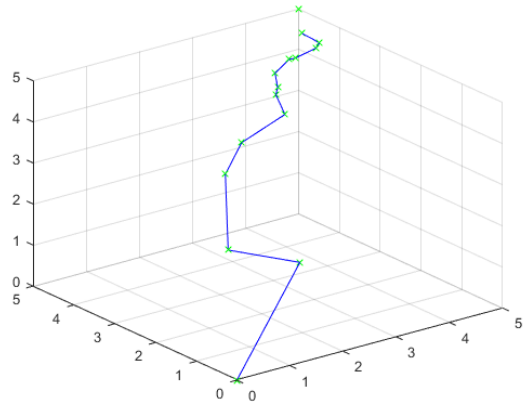


Fig. 12. Simple RRT tree.

Overall, the planner shows successful results in finding the solution. This simple approach will be expanded to include the transmission objects.

III. Results

After combining the RRT planner and the models of the transmission, a path could be generated to find a collision free path between the start pose and the goal pose.

Figure 13 and 14 visualizes an isometric view and YZ view, respectively, of the overall RRT graph structure that was searched and generated. The green x-marks indicate nodes that were sampled while the red x-marks indicate the nodes that were added to the tree. The blue lines represent the overall tree structure.

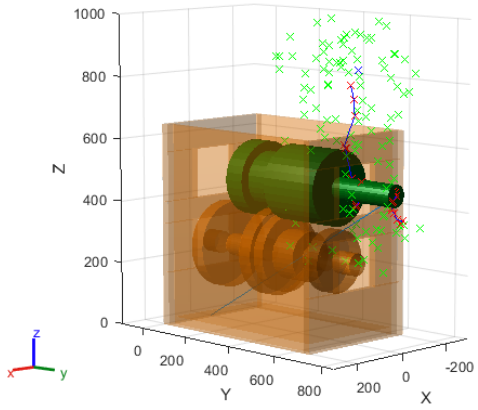


Fig. 13. Isometric view of RRT graph tree.

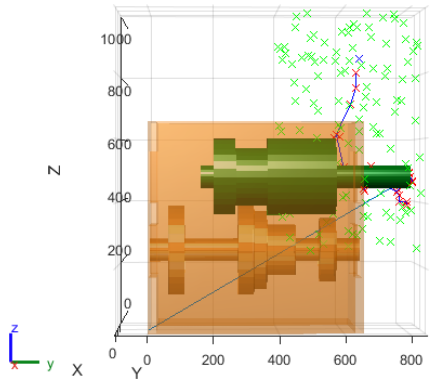


Fig. 14. YZ view of RRT graph tree.

Figure 15 and 16 visualizes an isometric view and YZ view, respectively, of the final path found by the planner.

An animation is available for viewing.

IV. Discussion

Overall, the RRT planner was successful in finding a collision free path to remove the mainshaft from the housing. The RRT planner provides a fast method of sampling the free space without having full knowledge of the environment, leading to effective tree generations. It was noted that, when generating

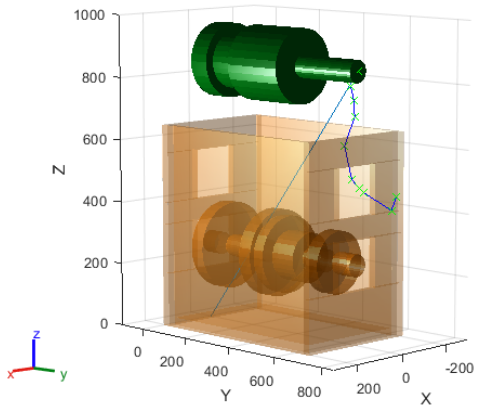


Fig. 15. Isometric view of the final collision free path found by the planner.

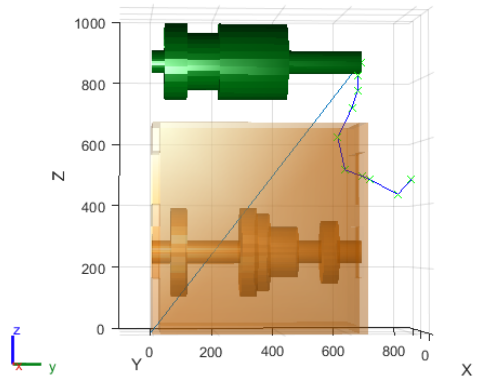


Fig. 16. YZ view of final collision free path found by the planner.

samples in the free space, implementing limits on the search field increased the efficiency in finding a solution. Knowing certain regions where points shouldn't be sampled, such as on the opposite side of the case or inside the countershaft, helped provide meaningful sampling. In general cases, such as a 2D application for mobile robots, the use of an occupancy grid would provide this same benefit in having meaningful sampling.

For future approaches, it would be interesting to dive into how different motion models can impact the generation of the RRT graph.

REFERENCES

[1] D. M. Flickinger. Transmission Assignment. (2024, Spring). RBE 550. Worcester, MA, USA: Worcester Polytechnic Institute

[2] MATLAB, "Collision Detection - MATLAB & Simulink," www.mathworks.com. https://www.mathworks.com/help/robotics/collision-detection.html?s_tid=CRUX_lftnav (accessed Mar. 31, 2024).

[3] MATLAB, "change base of rigidBodyTree like in SerialLink," www.mathworks.com, Dec. 17, 2020. https://www.mathworks.com/matlabcentral/answers/572548-change-base-body-of-rigid-tree-robot (accessed Mar. 31, 2024).

[4] D. M. Flickinger. Week 11 Lecture: Rapid-exploring Random Trees. (2024, Spring). RBE 550. Worcester, MA, USA: Worcester Polytechnic Institute