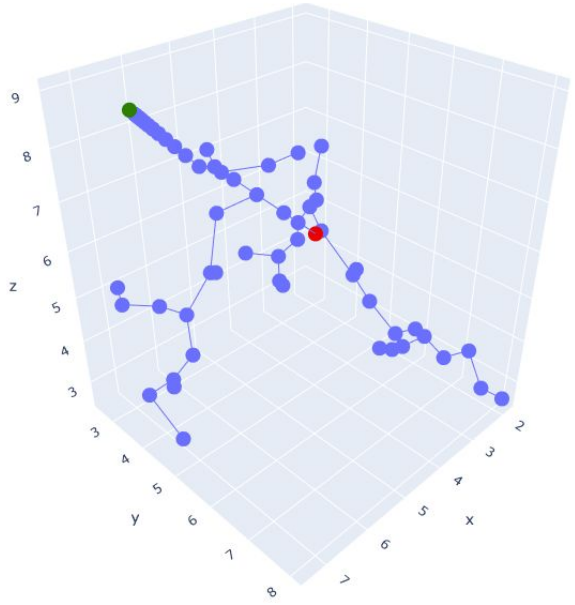
Project 6 - Report

CS 3630, Spring 22

Anthony Wong

Q1.1 Visualization of the RRT tree (1 point)

Paste a screenshot of the visualization of the RRT tree in your report.



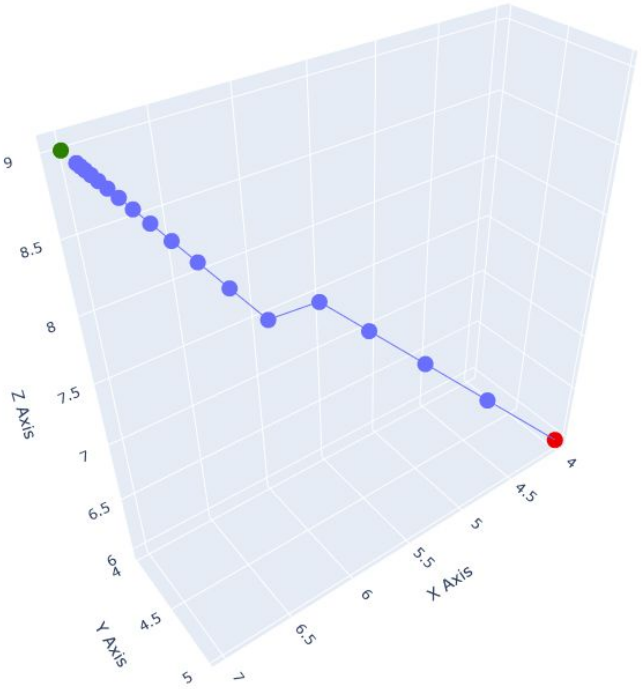
Q1.2 RRT Implementation (1 point)

Paste a screenshot of your implementation of the run_rrt function.

```python
def run_rrt(start_node, target_node, generate_random_node, steer, distance, find_nearest_node, threshold):
  '''
  Arguments:
   - start_node: the start node, it could be gtsam.Point3 or gtsam.Pose3.
   - target_node: the destination node, it could be gtsam.Point3 or gtsam.Pose3.
   - generate_random_node: this function helps us randomly sample a node
   - steer: this function finds the steer node, which takes us closer to our destination
   - distance: this function computes the distance between the two nodes in the tree
   - find_nearest_node: this function finds the nearest node to the randomly sampled node in the tree
   - threshold: float, this is used for the terminating the algorithm

  Returns:
   - rrt: List[gtsam.Point3] or List[gtsam.Pose3], contains the entire tree
   - parent_index_list: List[int], contains the index of the parent for each node in the tree
  '''

  rrt = []
  parent_index_list = []
  max_iterations = 2000
  rrt.append(start_node)
  parent_index_list.append(-1)

  for i in range(max_iterations):
    ######## Student code here ########

    sample = generate_random_node(target_node)
    near, nearidx = find_nearest_node(rrt, sample)
    steernode = steer(near, sample)
    rrt.append(steernode)
    parent_index_list.append(nearidx)
    if distance(target_node, steernode) < threshold:
      break

    ######## End student code  ########

  return rrt, parent_index_list
```

Q1.3 Visualization of the final path (1 point)

Paste a screenshot of the visualization of the final path obtained in your report.



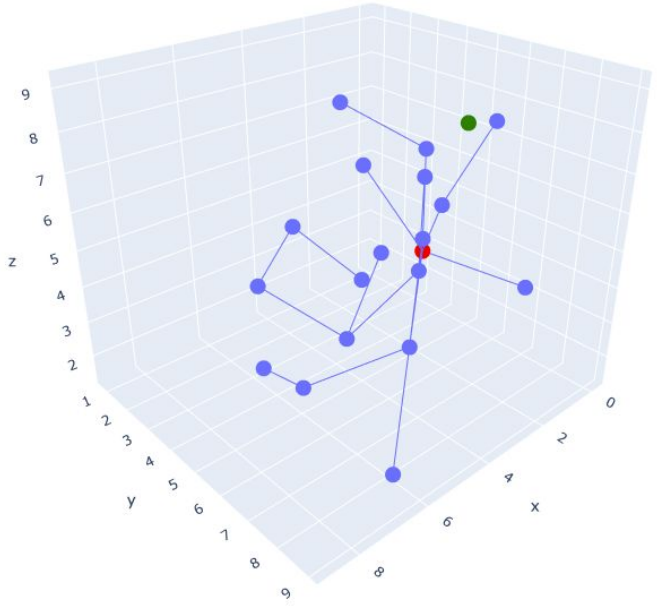
Q1.4 Nodes are close to one another! (2 points)

If you look closely at the tree and the path, you'll notice that as we reach close to the goal (the green node), the nodes are extremely close to one another! We begin to take really small steps and we converge very slowly towards it. Can you explain why this is happening?

This happens because we have a 20% chance of sampling the goal node. This means we will initially make significant progress toward the target. However, once we get closer, if we sample the goal node the distance we progress towards it gets increasingly small. This is why we see small steps.

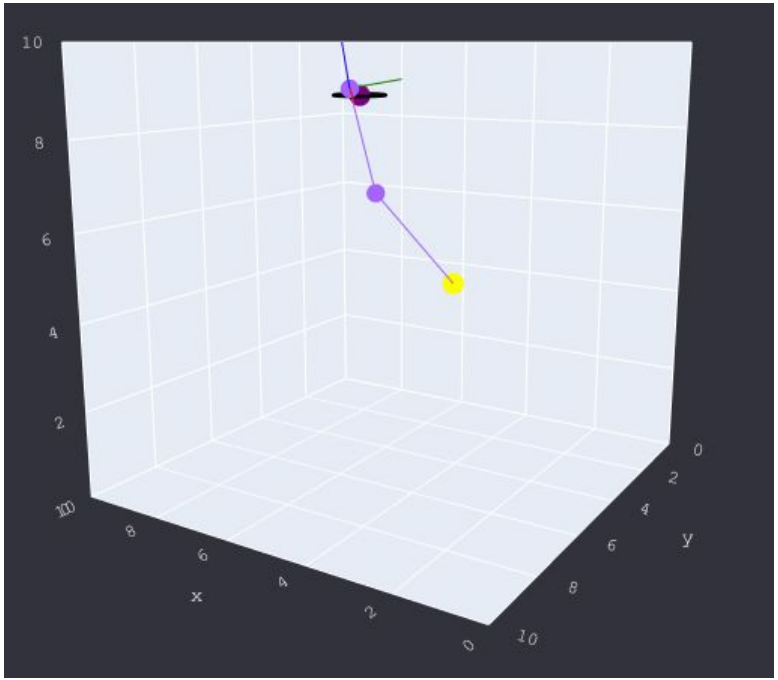
Q2.1 Visualization of the RRT tree (2 points)

Paste a screenshot of your RRT tree visualization obtained after steering with a terminal velocity.



Q2.2 Visualization of the final path (2 points)

Paste a screenshot of your final path obtained, after steering with a terminal velocity.



Q2.3 Is steer() realistic? (2 points)

The “steer” function we coded up does not seem realistic in nature. What do you think are the two major limitations of our approach?

One limitation is that it is able to change direction and altitude instantaneously without changing its velocity, which is not realistic. Additionally, the steer function does not incorporate gravity into its approach. This is also not realistic as there is no way of getting around it.

Q3.1 Configuration 1 (5 points)

Configuration:

- Yaw: 90 degrees
- Pitch: 0 degrees
- Roll: 0 degrees
- Thrust: 10 Newtons

1. In which direction is the drone flying in the navigation frame? [0 0 0]
2. Is the drone flying upwards, downwards or maintaining level flight? Maintaining
3. In what direction is the thrust applied in the navigation frame? [0 0 1]
4. What is the speed of the drone? 0 m/s
5. What direction is the front of the drone facing in the navigation frame? [0 1 0]

Q3.2 Configuration 2 (5 points)

Configuration:

- Yaw: 0 degrees
- Pitch: 45 degrees
- Roll: 45 degrees
- Thrust: 20 Newtons

1. In which direction is the drone flying in the navigation frame? [0.644 -0.765 0]
2. Is the drone flying upwards, downwards or maintaining level flight? Maintaining
3. In what direction is the thrust applied in the navigation frame? [0.5 -0.707 0.5]
4. What is the speed of the drone? 23.83 m/s
5. What direction is the front of the drone facing in the navigation frame? [0.707 0. -0.707]

Q3.3 Configuration 3 (5 points)

Configuration:

- Yaw: 25 degrees
- Pitch: 0 degrees
- Roll: 45 degrees
- Thrust: 15 Newtons

1. In which direction is the drone flying in the navigation frame? [0.552 -0.809 0.203]

2. Is the drone flying upwards, downwards or maintaining level flight? Upwards

3. In what direction is the thrust applied in the navigation frame? [0.299 -0.641 0.707]

4. What is the speed of the drone? 18.6 m/s

5. What direction is the front of the drone facing in the navigation frame? [0.906 0.423 0.]

Q3.4 Configuration 4 (5 points)

Configuration:

- Yaw: 10 degrees
- Pitch: 45 degrees
- Roll: 30 degrees
- Thrust: 0 Newtons

1. In which direction is the drone flying in the navigation frame? [0. 0. -1.]

2. Is the drone flying upwards, downwards or maintaining level flight? Downwards

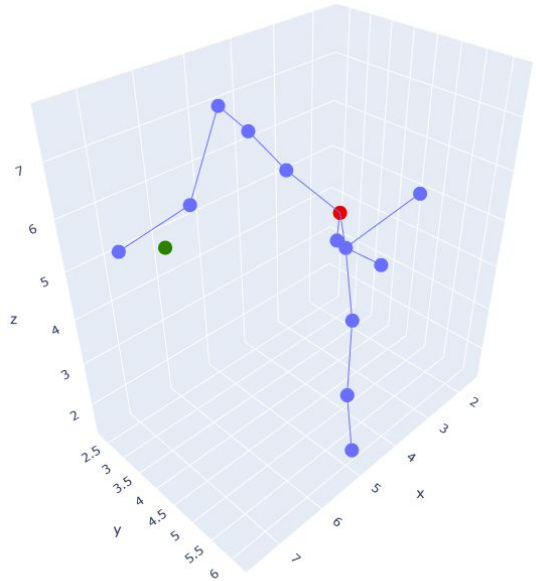
3. In what direction is the thrust applied in the navigation frame? [0.69 -0.386 0.612]

4. What is the speed of the drone? 15.339 m/s

5. What direction is the front of the drone facing in the navigation frame? [0.696 0.123 -0.707]

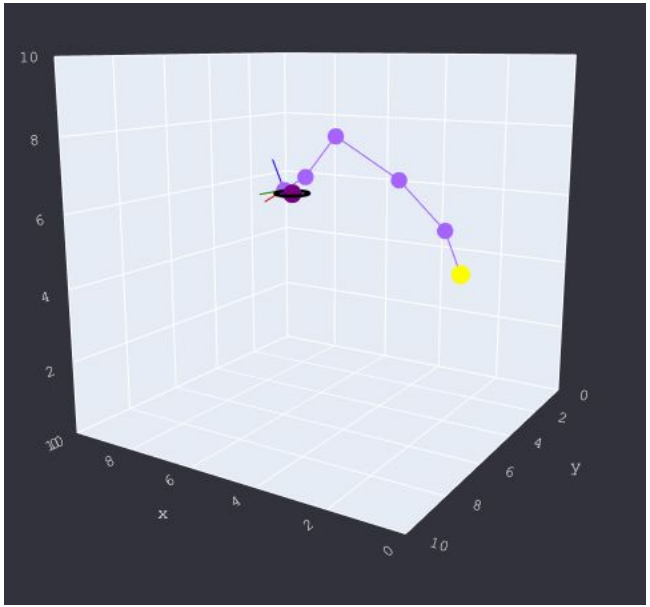
Q4.1 Visualization of the RRT tree (3 points)

Paste a screenshot of your RRT tree visualization obtained after implementing a more realistic version of the steer function.



Q4.2 Visualization of the final path (3 points)

Paste a screenshot of your final path obtained after implementing a more realistic version of the steer function.



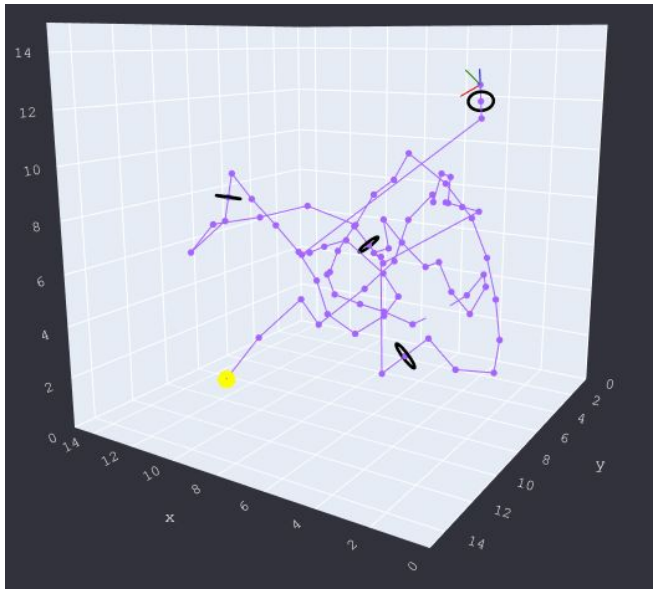
Q4.3 Fly faster? (3 points)

Open-ended: Do you think the drone can fly faster than our current implementation using these realistic dynamics? If yes, how? If no, why?

In certain cases it is possible to fly faster. If we use realistic dynamics we avoid making significant changes in progression towards the goal when sampling. This allows for the 20% that we sample the target node to take over, meaning we are likely heading is a progressive direction when it is chosen. If we do not use realistic dynamics, there’s a higher likelihood that we undo progress we may have which will take more extra steps to correct.

Q5.1 Drone trajectory visualization (3 points)

Paste a screenshot of your drone's trajectory at the GT 3630 Drone Racing Challenge, passing through all the hoops in the circuit.



Q5.2 Drone racing implementation (3 points)

Paste a screenshot of your implementation of the drone_racing_rrt function.

```python
def drone_racing_rrt(start_pose: gtsam.Pose3, target_poses: List[gtsam.Pose3]) -> List[gtsam.Pose3]:
  '''
  Arguments:
   - start_pose: gtsam.Pose3, initial pose of the drone
   - target_poses: List[gtsam.Pose3], list of hoop poses (RRT targets)

  Returns:
   - drone_path: List[gtsam.Pose3], entire path from start to last hoop
  '''

  drone_path = []

  ######## Student code here ########

  currpose = start_pose
  for tpose in target_poses:
    rrt, ids = run_rrt(currpose, tpose, generate_random_pose, steer, helpers.distance_between_poses, find_nearest_pose, threshold=1)
    path = get_rrt_path(rrt, ids)
    helpers.pass_through_the_hoop(tpose, path)
    drone_path += path
    currpose = drone_path[-1]

  ######## End student code  ########

  return drone_path
```

Q5.3 Drone trajectory optimization (3 points)

Open-ended: What do you think can be some ways to optimize the trajectory taken by the drone? Keep in mind that instantaneous rotation of the drone is limited to -10 to 10 degrees in yaw, pitch and roll.

Just like in previous projects, the concept of path smoothing couple help optimizing the trajectory. This can help minimize unnecessary poses in our path. Though we would need to ensure we fit within the rules of instantaneous rotation, we would be able to have at least some impact on performance.

Q6 Feedback (1 point)

Please provide feedback on the coding portion of the project. How did it help your understanding of the material? Is there anything that you think could have been made more clear?

I would’ve liked more examples and explanation for the yaw, pitch, and roll matrix problem and as to how that translates to the rotation matrix. Additionally, I think gtsam documentation can still be improved.