ASEN 5519 Final Project: Wire Routing in 2D and 3D Environments Using Multi-level Path Planning

1st Scott Scheraga
Department of Mechanical Engineering
University of Colorado Boulder
Boulder, CO, USA 80309
Scott.Scheraga@colorado.edu

I. Introduction

The IROS 2020 Robotic Grasping and Manipulation Competition is a event organized by NIST in which teams use robotic manipulator arms to perform various assembly tasks including driving screws, putting various pegs in holes, wrapping a belt around two pulleys and installing multiple electrical connectors, as seen in Fig. 1. In Fall 2020, University of Colorado Boulder's Correll Lab competed in this competition, and had the most difficulty with a task involving routing a USB cable around two posts, through two "gates" and vertically plugging the USB cable into a port.

Points in the competition's USB cable routing task are earned from successfully routing the cable in the required path through/between each of the obstacles in correct directions. An optimal trajectory is not required. CU Boulder's team was able to accomplish the wire-routing task only in testing, by hard-coding a sequence of locations that the arm would reach through inverse kinematics, and interpolating between poses.

Integrating a non-optimizing multi-level planner to create a path between multiple general zones would greatly improve future performance at this task. In this paper, I detail my approach to this problem and include goals that were and were not accomplished in the time-frame of this semester.

II. Problem Statement(Project Goals)

The overall project goal was to create a path planning methodology for wire routing from start to goal locations, with mandatory subgoal locations/zones in the environment in both 2D and 3D environments.

a) Subproblem 1: Implement GoalBias-RRT or a PRM Planner in a 2D environment to guide a point-robot to a sequence of subgoals to approximate the intended USB cable path, and go in the correct direction through each set of obstacles, without any collisions.

b) Subproblem 2: Implement Syclop or an LTL high level planner in order to direct the low-level planner between local goals or local zones.

c) Subproblem 3: Add directionality to the point-robot poses,and also restrict the turning radius of paths created by the low level planner

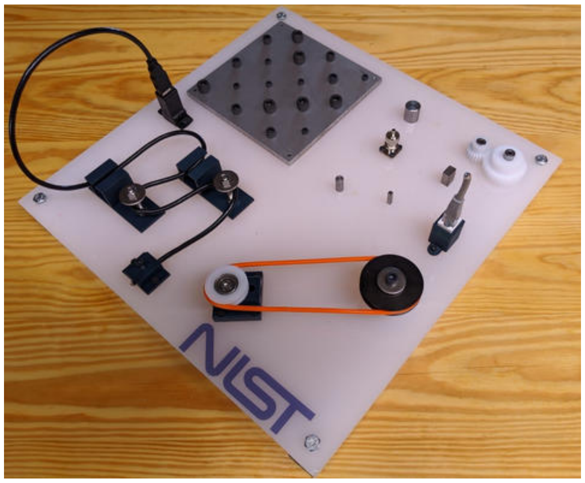

Fig. 1. The IROS 2019 Robotic Grasping and Manipulation Competition task board, with USB Cable routing task on the left side. (Image does not match competition-provided sample CAD files I used for motion planning)

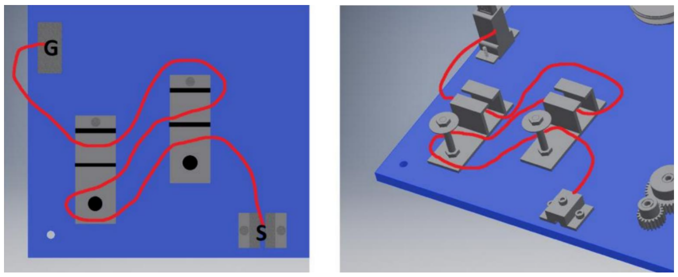

Fig. 2. Conceptual Routing Paths overlaid on 2D (Left Image) and 3D (Right Image) competition-provided sample CAD files

d) Subproblem 4: Simulate the cable to follow the generated path, and also modify all high and low-level path planning algorithms such that they work in a 3D environment.

e) Additional Subproblems: To develop realistic paths for the USB cable, I needed to implement a way of collision-checking locations around any sampled point within a given radius (depending on the environment). This is in order to represent the cross-sectional volume of the wire. Additionally, I needed to determine an upper limit to the resolution of the boundary zone such that the algorithms run in a reasonable amount of time.

III. Methodology

Contrary to the initially conceived subproblems, development of the wire-routing multi-level path planner developed in a significantly different order, with Subgoal 3 and Subgoal 4's cable simulator not being achieved in the timeframe of the semester. It was determined that development of the 3D environment needed to occur fairly early in order to make implementation easier.

Development occurred in roughly the following order: Low-level 2D planner with subgoals, Low-level 3D planner with subgoals, 2D/3D planners with subgoals with wire "volume", High-level 2D/3D planners, Integration of 2D/3D High-level and Low-Level Planner.

A. Low-Level 2D Planner with Subgoals

I chose to use a goal-biased RRT planner [1] as a foundation for low-level planning as opposed to a PRM planner due to the smaller required number of critical variables to tune, my level of prior experience in the algorithm's reliability in reaching goals, and perceived ease of future modification. The parameters used for RRT were ρ=0.25 inches minimum distance between nodes, a 5% bias to the goal, and a minimum acceptable distance to goal of 0.25 inches. Obstacles are represented with an intersection of half-planes, using defined minimum and maximum values for X and Y. As seen in Fig. 3, I was able to link multiple runs of the RRT algorithm with local start and local goal coordinates. The intended path is in green while the generated path is depicted in red. In Fig. 3, it is notable that while the algorithm was able to successfully reach each of the subgoal "waypoints" in proper sequence, the planned segments intersect each other at multiple locations. Additionally, the plots frequently come into very close contact with the obstacles. Both characteristics of the path would be unacceptable for actual USB cable routing.

B. Low-level 3D Planner With Subgoals

Next, a 3D waypoint-constrained GoalBiased RRT Algorithm was implemented, along with 3D representations of the obstacles, with dimensions taken from the competitions CAD data. Similar to my implementation of the 2D planner, I defined obstacles with an intersection of half-planes, using defined minimum and maximum values for X, Y and Z. For the planner, I set local start points, local goal points and minimum Z values that could be added to the tree to be 0.5 inches, as a rough estimate for the cutout location on a robot's gripper the USB cable would usually fall into during grasps in the real life environment. For the 3D planner, the same RRT parameters were used as the 2D planner. Similar to the path generated in Fig. 3, the 3D path in Fig. 4, often comes extremely close to contacting obstacles. While not depicted in Fig. 4, the generated paths also occasionally come close to intersecting.

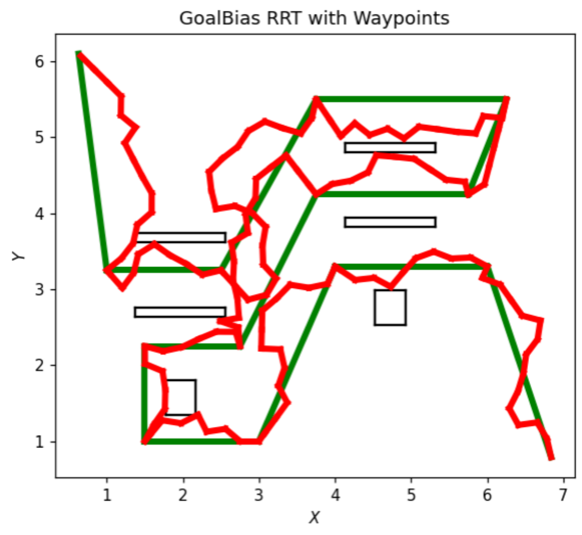

Fig. 3. 2D Path Generated by Goal-Biased RRT (red line) with a sequence of waypoints (green line)

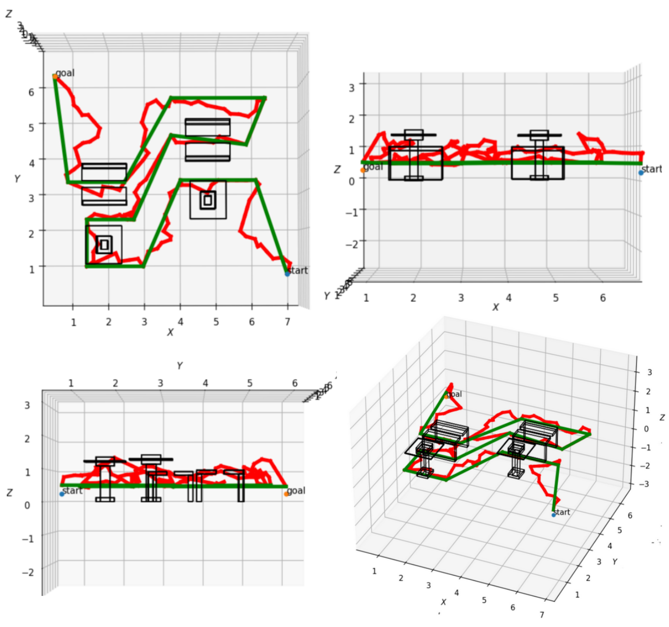

Fig. 4. Four views of the 3D Path Generated by Goal-Biased RRT (red line) with a sequence of waypoints (green line)

C. Planning With "Volume" Collision Detection

In order to more accurately portray the real-life planning problem, next major feature to be implemented (which was not part of the original subproblems) was path generation given a wire cross-section. The real-life USB cable has a cross-sectional radius of 0.07 inches. In order to only add configurations and edges to the tree that did not intersect with obstacles given a wire volume, the RRT algorithm was modified. Once the q_{rand} random sample is generated, q_{near}, the nearest configuration from the tree to it within distance ρ

is calculated. A quantity of points of a set radius around q_{near} is then checked for collisions. If there are no collisions around q_{near}, a modified IsSubpathCollisionFree function is run. This function checks not only if there are any obstacle collisions of points on the subppath, but also obstacle collisions of circles (2D environment)/spheres(3D environment) of points of the incremental points along that path.

In the 2D implementation seen in Fig. 5, the circles of points used 2 layers of points in addition to the circle's center, and 15 points per layer. In the 2D implementation, the circles of points used a radius of 0.07 inches, 2 nested layers of points in addition to the circle's center, and 15 points per layer, resulting in 31 points total.

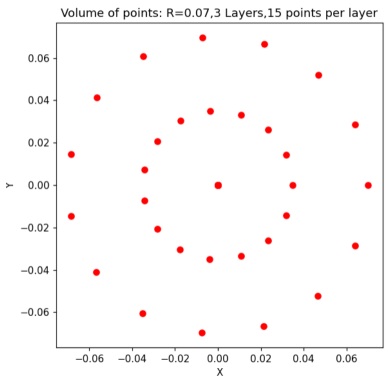

Fig. 5. Example circle of points used for 2D-environment collision checking with volume

In the 3D implementation seen in Fig. 6, the spheres used a radius of 0.07 inches, only a single nested spherical layer of points, 8 rows of points in the Z axis, and 5 points per row in the Z axis. Including the center point, this resulted in 41 collision-tested points per q_{near} and points along each tested subpath. The number of points tested for each "volume sphere" resulted in significant impacts in algorithm run time. While the selection of the final quantity of points per sphere was somewhat arbitrary, the final numbers result from paring down the number of points such that runs of the 3D planner with volumes took under 4 minutes per run. Additionally, through intuition, with the 40-point configuration, the points remain well distributed enough to prevent any obstacles from easily falling in between vertical "slices" of each tested sphere, resulting in false-negative collision checks.

After volume collision checks were sucessfully implemented in the 2D environment (Fig. 7) and 3D environment (Fig. 8), the issue of paths being generated exceedingly close to obstacles was improved. However, similar to before, generated paths often collide with each other. Additionally, the planners occasionally take unexpected routes from odd directions that would result in a loss of points in the real-life IROS competition.

D. High-level 2D/3D planners

In order to better constrain the generated paths in the 2D and 3D environments, a higher level planner would need to be implemented. Inspired by SyCLoP (Synergistic Combination of Discrete and Continuous Search) [2], I aimed to first create an abstraction of the environment, within which I would later be able to create a high-level discrete plan. Then, to synergize with the low level planner, the high level planner would bias the sampling done by the low level planner. As the majority of the issues I was having with the 3D Goal-Biased RRT planner took place on the XY (top down) plane, a 2D high level planner could drive sampling in the 3D planner to great effect.

As seen in Fig. 9, I abstracted the space into zones that only exist in areas that do not intersect with the 2D obstacles. These zones use similar definitions to obstacles, with minimum and maximum values for X and Y.

To drive sampling of the low-level planner, I utilized Di-

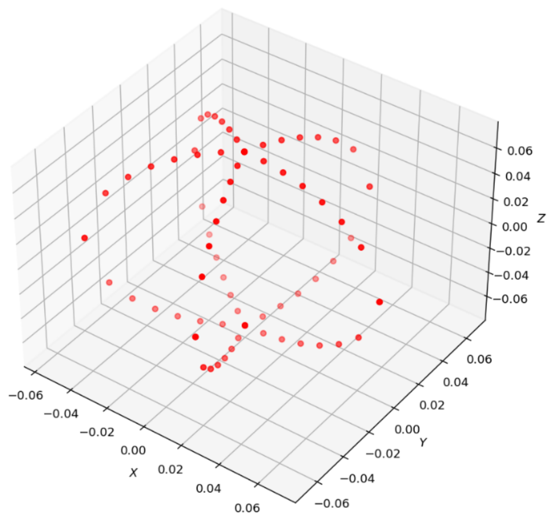

Fig. 6. Example sphere of points used for 3D-environment collision checking with volume

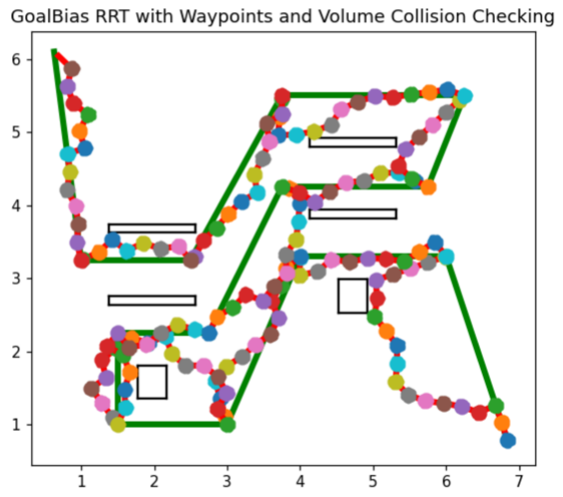

Fig. 7. 2D Goal-Biased RRT with Volume Collision Checks. Tested volumes are represented only at nodes

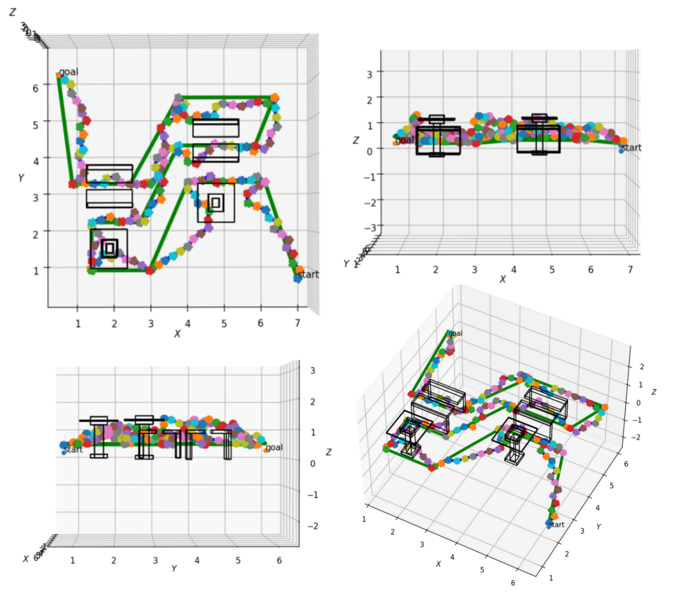
Fig. 8. 3D Goal-Biased RRT with Volume Collision Checks. Tested volumes are represented only at nodes

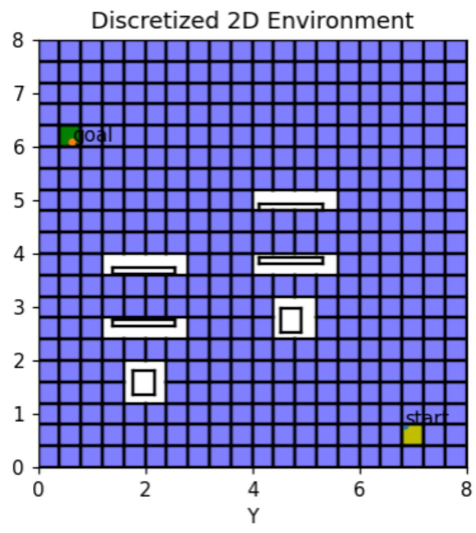
Fig. 9. Discretized grid comprising the abstracted environment of the high level planner.

jkstra's Algorithm to generate Manhattan-type paths between sets of waypoints, as seen in Fig. 10. The paths created did not need to be exhaustively searched, as optimal paths are not required for the wire-routing task. Manhattan-type-only edges/movements were deemed acceptable as opposed to Manhattan-type plus diagonal movements due to the abstracted nature of the planner, and the fact that the lower level planner also is biased to the goal, potentially smoothing out sharp path edges. (A reliable A^* algorithm implementation and diagonal path edges/movements were not able to be successfully debugged within the project's timeframe). An initial implementation of a 0.5x0.5 inch grid size revealed that even in the discretized environment, the center of the obstacles resulted in overlapping paths. In order to alleviate this issue while still maintaining a high degree of environment abstraction, a 0.4x0.4 inch gridsize was used, and an additional waypoint (labeled "3" in Fig. 10)

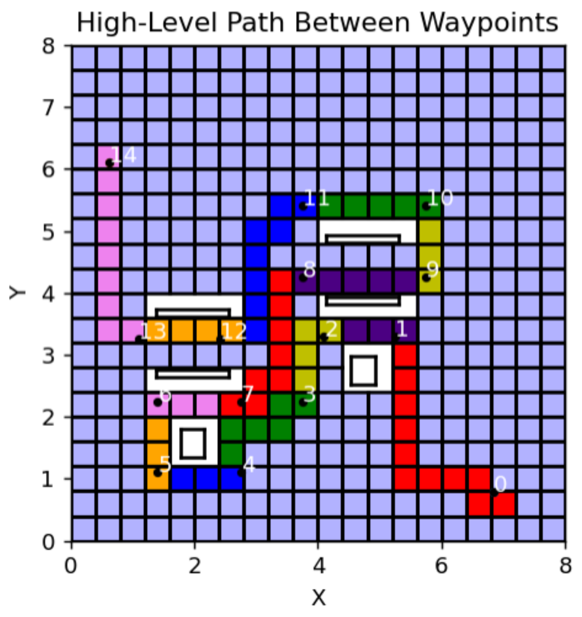
Fig. 10. Subpaths denoted by different colors in the discretized environment generated by Dijkstra's Algorithm between start ("0") and goal ("14"), with local start/goal nodes at each of the other numbers

E. Integration of 2D/3D High-level Planner and 2D/3D Low-Level Planners

While the 2D and 3D RRT planners previously randomly sampled configurations anywhere in their environments, to integrate the high level planner's paths between abstracted zones in the environment and the low level RRT planners, I biased sampling. Using the same local start and local goal configurations as the high level planner, I segmented the overall high-level plan to local segments. When the 2D and 3D Goal-Biased RRT planners would aim to travel from one local start to a local goal, a large percentage of configuration sampling would be from that segment's high level plan. When the low level planner randomly decides to sample from the local high-level plan, it randomly selects one of the zones from the high level plan, and randomly samples from within it.

After some tuning, I found that a 5% goal bias, 20% random sampling and a 75% to effectively balance over- and under-biasing the 2D and 3D RRT planners' paths to the high level plan. As seen in Fig. 11, the 2D generated path neither sticks too close to the green line between waypoints, or too close to the occasional sharp corners of the Manhattan-type edges (pink squares) of the high level planner. Additionally, the overall planned path does not collide with itself at any point.

Additionally, the 2D high level planner was able to be successfully integrated with the 3D Goal-biased RRT planner with the same biasing parameters as the 2D RRT planner, as seen in Fig. 12. Due to time constraints, 3D sampling of configurations was only done within a range of 0.2-0.75 inches in Z, using the same ”generate_nodezone” function used to synergize the high level planner’s list of zones from each local start and goal with RRT sampling. The integrated 2D high level + 3D low level planner had similar positive performance to the 2D high level + 2D low level planner.

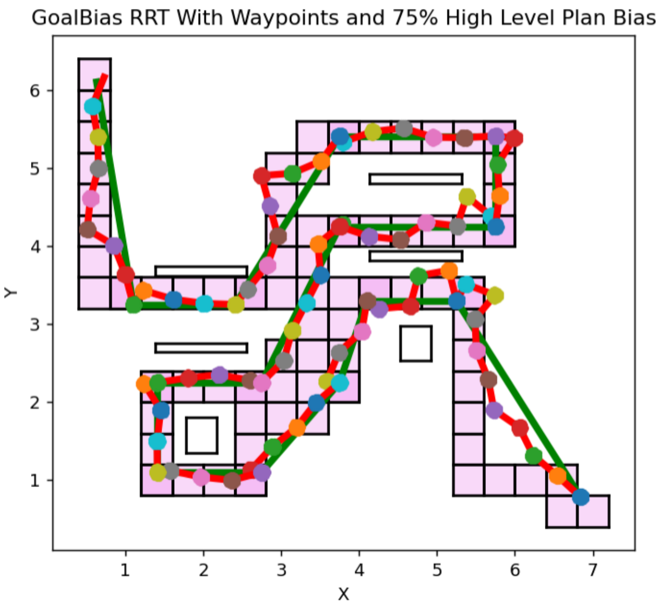

Fig. 11. Integrated High-level 2D Dijkstra’s Algorithm and Low-level 2D Goal-Biased RRT planner with waypoints and volume collision checks

IV. Results

A high-level planner consisting of a Dijkstra’s Algorithm-driven set of paths in a discretized environment was able to be integrated with both a 2D and 3D Goal-Biased RRT planners with volume collision checking. The integration of a high-level planner resulted in significant improvements in removing the tendency of local paths to intersect with each other, especially in the center area of the environment- notably local paths between waypoints 2-3 and 7-8 in Fig. 10. Paths are still occasionally generated very close to obstacles, and sharp corners between pairs of path edges are still present in nearly every run of both the 2D and 3D multi-level planners.

V. Discussion

The integrated high level and low level planner resulted in significantly improved paths compared to the low-level-only 2D and 3D GoalBias RRT with waypoints planners. The integration of a high-level planner removed path intersections and greatly reduced the likelihood that paths were created in incorrect directions around obstacles, as seen in Fig. 7.

A while the integrated planner greatly improved path quality, a few issues still remain. The integrated planner generates

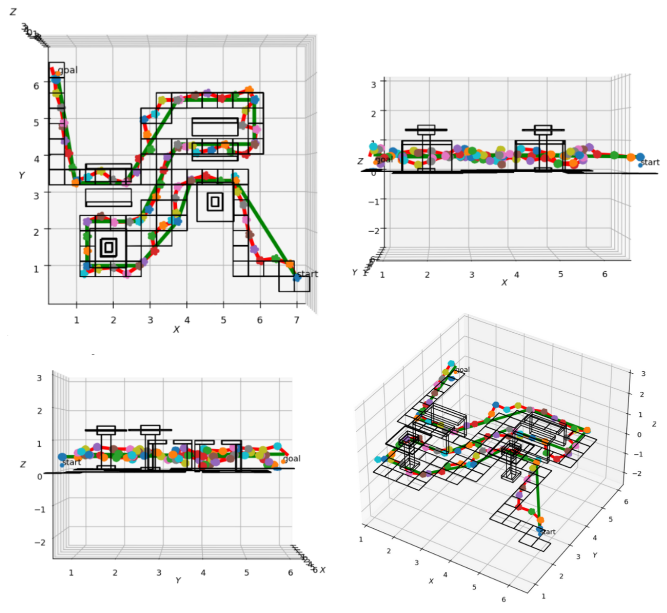

Fig. 12. Integrated High-level 2D Dijkstra’s Algorithm and Low-level 3D Goal-Biased RRT planner with waypoints and volume collision checks

paths with unrealistically sharp corners between sets of edges. Path smoothing through the removal of nodes and connection of neighboring ones was considered as an option in the original proposal to alleviate this. However, with enough smoothing, an inevitable result for each local path would be between the removal of all points between the path’s local start and local goal. A superior solution would likely be the addition of a function that only adds edges to the lower-level planner’s trees that satisfy a minimum angle condition with the parent function of that node. This would result in more realistic rounded paths.

The integrated planners still occasionally develop paths very close to obstacles, especially when the high level planner’s grid is directly adjacent to an obstacle. To develop safer paths, a nonuniform grid could be implemented for a high-level planner’s that utilizes grid sizes that increase in size the further one is from obstacles and tends to center grid cells and the centerline between obstacles.

To further improve the integrated multi-level 3D planner, a true 3D abstracted grid with a planner would need to be implemented. The current implementation may more accurately be considered a 2.5D high-level planner, as it only biases X and Y axis sampling to the high-level plan while Z axis sampling is constrained between values of 0.2 and .75 inches. While this is sufficient for planning in the majority of the environment, in the real life wire-routing problem, the goal state requires the USB cable to inserted vertically downward at more than two times the height of the rest of the environment.

Lastly, in order to truly make this planner contribute to future CU Boulder efforts in the IROS Robotic Grasping and Manipulation Competitions, a wire should be simulated with an (abstracted) multi-joint beam, in order to properly demonstrate that while the tip of the wire follows a generated motion plan, the rest of the wire may bind against the obstacles, leading to the need for implementing a way to plan for slack in the wire's motion plan.

VI. Conclusion

The integration of a SyCLoP-inspired Dijkstra's Algorithm-driven high level planner in a highly-abstracted environment resulted in significant qualitative performance improvements to 2D and 3D RRT lower-level planners that routed between subgoals and had collision detection for volumes around the paths. The higher level planner virtually removed the potential for local paths to intersect.

In order to further improve the applicability of this multi-level planner to a robotic wire-routing task, a minimum angle-limiting function could be used to eliminate unrealistically tight corners in the paths. Additionally, modifying the high-level planner's discretized grid such that it tends to creating paths along centerlines between objects would increase the safety for the robot manipulator arm accidentally colliding with obstacles.

Acknowledgment

Thank you to Professor Morteza Lahijanian, Ofer Dagan and University of Colorado Boulder for their support and resources for this project

References

[1] S. LaValle., "Rapidly-exploring random trees: A new tool for path planning," Technical Report. Computer Science Department, Iowa State University , vol. 26.3, (TR 98–11), 1998.

[2] E. Plaku, L. Kavraki, and M.Vardi., "Motion planning with dynamics by a synergistic combination of layers of planning," IEEE Transactions on Robotics, vol. 26.3, pp. 469, 2010.

VII. Appendix

A. Detailed views of the output of the Integrated High-level 2D Dijkstra's Algorithm + Low-level 3D Goal-Biased RRT planner with waypoints and Volume collision checks

:

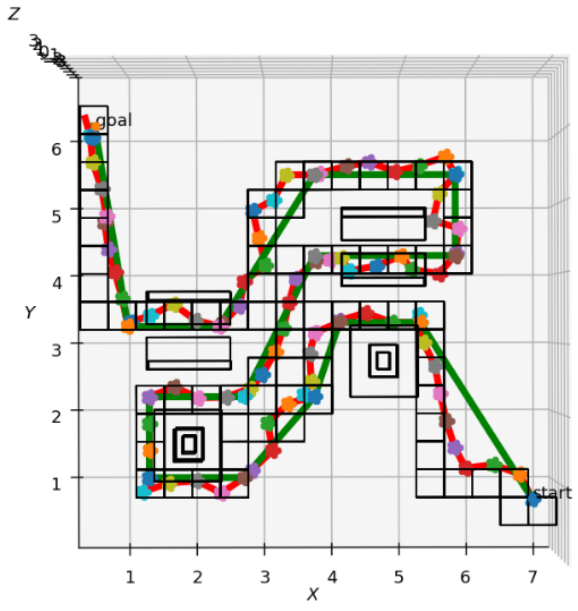

Fig. 13.

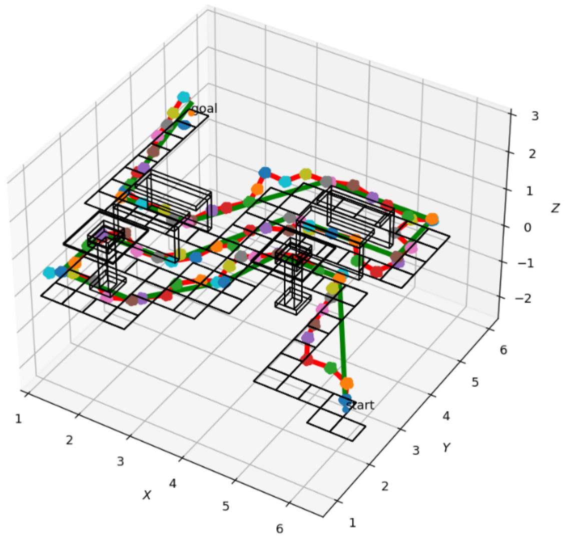

Fig. 14.

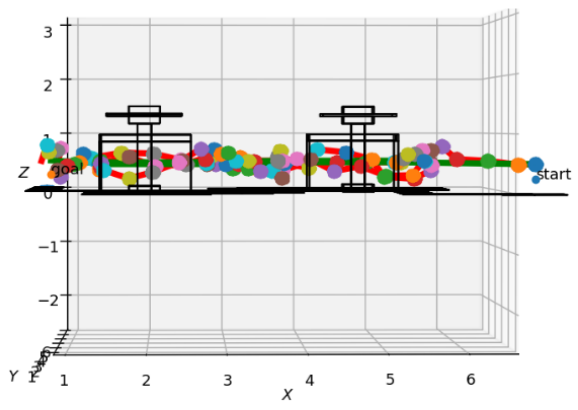

Fig. 15.

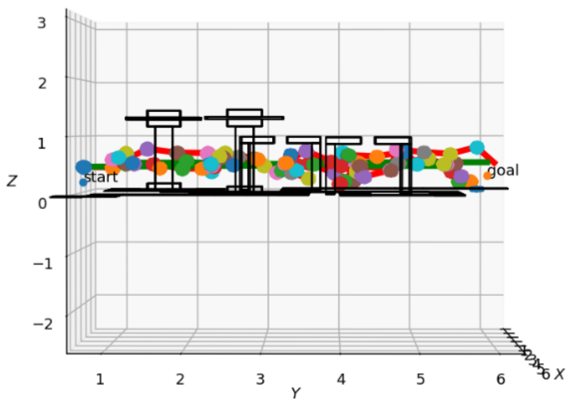

Fig. 16.