Using Machine Learning Techniques for Speeding up Manipulator Path Planning to Find High Quality Paths in Cluttered Environments

Pradeep Rajendran†, Shantanu Thakar†, Prahar M. Bhatt†, Ariyan M. Kabir†,and Satyandra K. Gupta† Center for Advanced Manufacturing, University of Southern California
Los Angeles, California 90089-1453.

Robotic manipulators are widely used in the factory floor for various tasks such as automotive assembly, part transport, packing and machining. A large fraction of these manipulators follows pre-programmed sequence of actions to complete tasks. Before any task, human operators program the manipulator by manually teaching waypoints to accomplish a given task. In high-mix, low-volume manufacturing settings, robots cannot be manually programmed by humans as robot-tasks change frequently. In such applications, we seek a planning method that quickly produces high-quality paths given a finite time budget. High-quality paths typically have minimal end-effector motion for the following reasons. Many manufacturing applications require a task to be completed with a minimal execution time. And, the task constraints typically dictate the joint speeds required and thus, the required joint speeds are much lower than the what the robot maximum limits are. So, minimizing joint motion is not going to make task execution faster. In fact, the workspace is cluttered and minimizing the end-effector motion instead reduces the risk of collision with other objects. In high-mix, low-volume applications, if the trajectory planning takes a long time, the robot is idle and not productive during that time. In such applications, robot needs to be productive for as long as possible and planning time needs to be minimal.

We focus on a general point-to-point path planning problem where the start and goal end-effector poses are given. Such under-specified start/goal configurations arising out of end-effector poses frequently occur in manipulator path planning (e.g., task feasibility assessment in task and motion planning). The objective is to find a configuration-space path that minimizes tool-path length. Tool-path length refers to the distance traveled by the tool at the end-effector and in this work, it is approximated by the distance traveled by the end-effector. A configuration-space path with minimum path-length in configuration-space does not necessarily correspond to having minimum tool-path length. Finding paths with minimum tool-path length is challenging as the workspace is often cluttered with obstacles with complex geometries.

Deterministic search-based methods can be used to solve the path planning problem for manipulators. But without specialized heuristics and techniques to control graph sparsity, these methods take a longer time to solve the problem than sampling-based methods.

Sampling-based motion planning algorithms have been studied for more than two decades and they have been extremely successful in solving a wide variety of motion planning problems. Popular sampling-based methods include RRT [1] and PRM [2]. These sampling-based planning methods are able to quickly generate paths by randomly sampling the configuration space and therefore cannot produce high-quality paths. They may produce feasible paths involving large end-effector motion. On the other hand methods that aim for asymptotic optimality take a long time to converge to an optimal solution. With a limited amount of time, neither of these methods can produce the required paths in high-mix, low-volume applications.

Local optimization can be performed using methods can be to smooth trajectories. But, these methods are limited to the homotopy class of the seed path and may not produce high-quality trajectories. Even after local optimization, we are likely to observe locally-optimal paths belonging to a variety of homotopy classes.

In the past, we worked on search-based [3], and sampling-based [4], [5] approaches that find a compromise between solution quality and planning time. They produce solutions with low-tool path length quickly by exploiting workspace cues. Workspace cues refer to focusing hints in the form of configuration space regions. These regions are derived from workspace paths of a reference point on the end-effector (Figure 1).

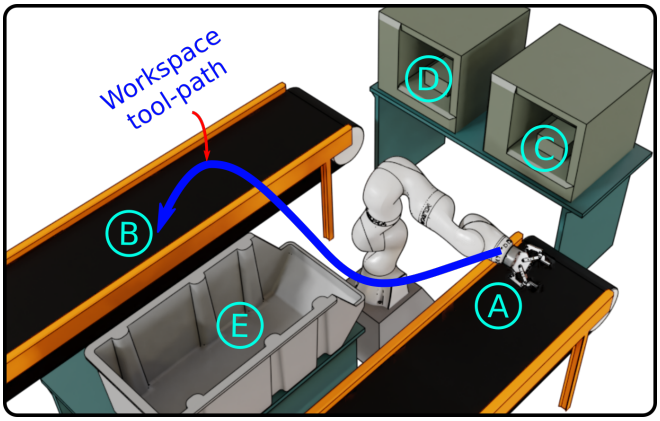
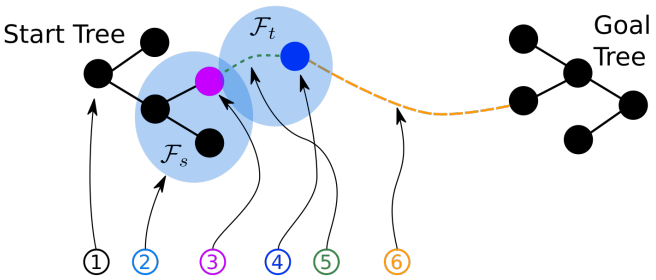


Fig. 1: Point-to-point planning between any two points in (A, B, C, D, E) can benefit from workspace-cues generated from workspace tool-paths. An example tool-path is shown for A to B.

In our prior work [4], we recognized that a general bi-directional tree search algorithm can be decomposed into the following primitive modules: tree selection, focus selection, node selection, target selection, extend strategy selection, and connection strategy selection (Figure 2). We refer to a specific implementation of a module as a *strategy*. In the literature, many of these modules have been imple-

mented in different ways. This has resulted in a rich set of alternative strategies. Each strategy is designed with a specific application in mind. For example, there are a set of alternative strategies for (1) handling the narrow passage problem, (2) focused sampling and extend strategy. In [4], as part of a human/machine cooperative-planning strategy, we exploited human assistance through an intuitive GUI to help dynamically refine workspace focus regions. These regions were simultaneously used by the planner for focusing the search efforts. As a continuation of [4], in [5], we removed the human component and improved our framework by reducing failure rate and reducing sub-optimality in many complex scenarios.

Through [4], [5], we have shown that intelligently switching between these strategies (Figure 2) (1) improves the robustness of the search method, (2) encodes user preferences (e.g., planning speed vs. sub-optimality trade-off), and (3) produces high-quality solutions in many challenging scenarios quickly. In this work, we are interested in using machine learning for search strategies. We use framework shown Figure 2) to solve the path planning problem. Conceptually, our framework is composed of six modules as in Figure 2.



#	Module	Strategy
1	Tree Selection	TS_A, TS_B, TS_C
2	Focus Selection	FS_A, FS_B
3	Node Selection	NS_A, NS_B
4	Target Selection	TaS_A, TaS_B, TaS_C
5	Extend Strategy Selection	ES_A, ES_B
6	Connect Strategy Selection	CS_A, CS_B, CS_C

Fig. 2: Bi-Directional Tree Search Framework

For each module, we dynamically select one of the strategies from the table in Figure 2 during each iteration. Each module operates on inputs and provides outputs. Thus, all strategies corresponding to a particular module follow the same input/output relationship (see Figure 2). For instance, consider the node selection module. No matter what strategy is used within the node selection module, the output has to be a node belonging to the tree that was given as an input.

We tested our method in a diverse set of 30 scenarios. These scenarios were constructed by taking inspiration from manufacturing tasks. Figure 3, 4 shows a representative subset of the test scenarios. This set contains scenarios of varying difficulty including those scenarios for which workspace cues could be misguiding.

Results shown in Figure 5 were computed using the data collected over 50 planning queries for each scenario (total 30), for each planner. In Figure 5, each colored data point (x, y) is the performance of a particular planner. Here x refers to the average failure rate of a particular planner averaged over all scenarios and all trials. And, y refers to the average suboptimality factor attained for each planner-scenario combination. The star symbol denotes the expected performance of a particular planner.

To study the impact of strategies, test scenarios shown in Figure 3, 4 are used. A brief description of some of the test scenarios is given below.

S03 features a table-like structure where the tool has to reach the other side of the central partition. In *S04*, a sanding tool needs to be moved from one side of the mold to the middle slot. There is a narrow constriction blocking the way, and the tool cannot be slid through. In *S07*, the manipulator is reaching into the void formed by the cross-bars supporting the cylindrical structure. In *S12*, the manipulator is in a pillar problem where it has to move from one side of the rods to the other side. In *S14*, the manipulator needs to move from one mold to another mold. But, its motion highly restricted by the long vertical rod in the middle. In *S23*, the manipulator arm is reaching out into a circular cut and it needs to pull the arm out of the cut and reach the goal pose. In *S26*, a sanding tool is to be transferred from one circular slot of a mold to another slot next to it. The mold is structured such that simply sliding the tool will not work (due to constriction). The tool has to be carefully maneuvered out of the mold first before inserting it again into the next slot. In *S30*, the manipulator needs to move the tool from a confined pose between two bars. And, the tool has to be taken out into a new pose where it is relatively less confined.

In Figure 5, RRTC and RRTC* have the highest failure rates. Compared to RRTC, RRTC* yields only a slightly better solution quality and incurs an increased failure rate. This is due to time spent rewiring nodes that were never going to be part of a high-quality solution. EETC dominates RRTC in terms of failure rate and path quality. The blending of EETC and RRTC as in A-RRTC+A-EETC yields lower failure rates and demonstrates the synergistic interaction between the strategies. This positive interaction is particularly noticeable in the workspace-misguiding scenarios (S12, S26, S19, S31) where EETC is likely to fail often. However, it does little to increase path quality. Context-awareness and synergistic interaction of strategies help drive the failure rate down and increase path quality in CODES3. Specifically, random sampling in $\mathcal{W}$-space helps to easily identify narrow passages in $\mathcal{W}$-space that are close to the end-effector; thus, it effectively identifies the corresponding $\mathcal{C}$-space narrow passages (e.g. scenarios S23, S21). Although the path quality of alternative methods are occasionally better in some tough scenarios, CODES3 has a considerably lower failure rate and suffers only slightly in terms of path quality. Sub-optimal feasible solutions obtained at a low failure-rate are often more valuable than optimal solutions with high failure-rates.

We have shown that by switching between strategies, we are able to strike a balance between path quality and planning

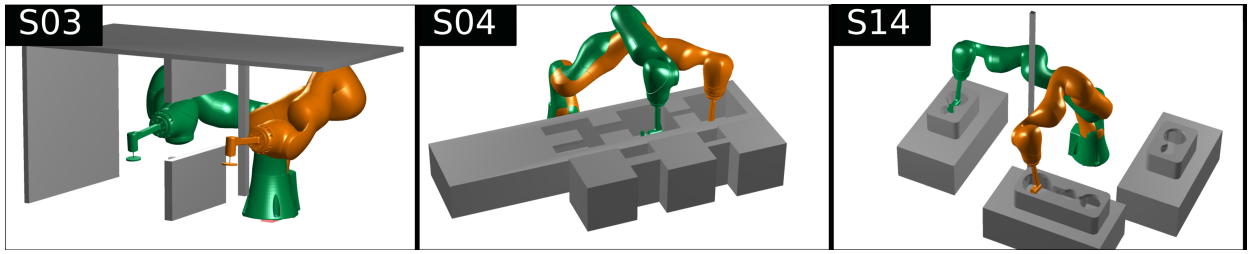


Fig. 3: A subset of the test scenarios. Start configuration is shown in orange and the goal configuration is shown in green. The robot is holding a rotary sanding tool in all scenarios.

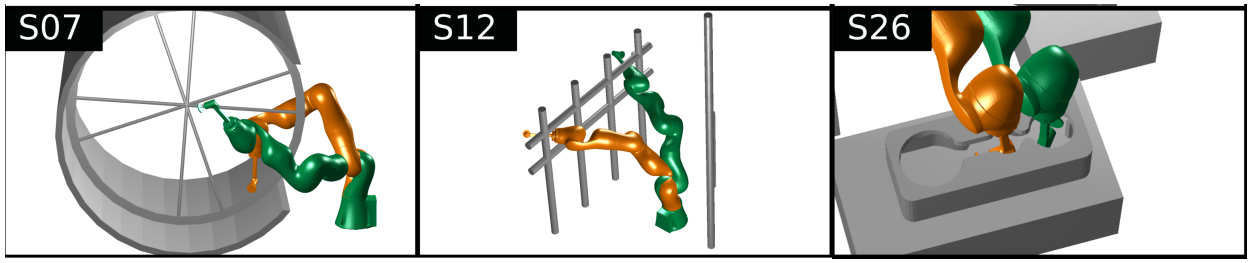


Fig. 4: A subset of the test scenarios. Start configuration is shown in orange and the goal configuration is shown in green. The robot is holding a rotary sanding tool in all scenarios.

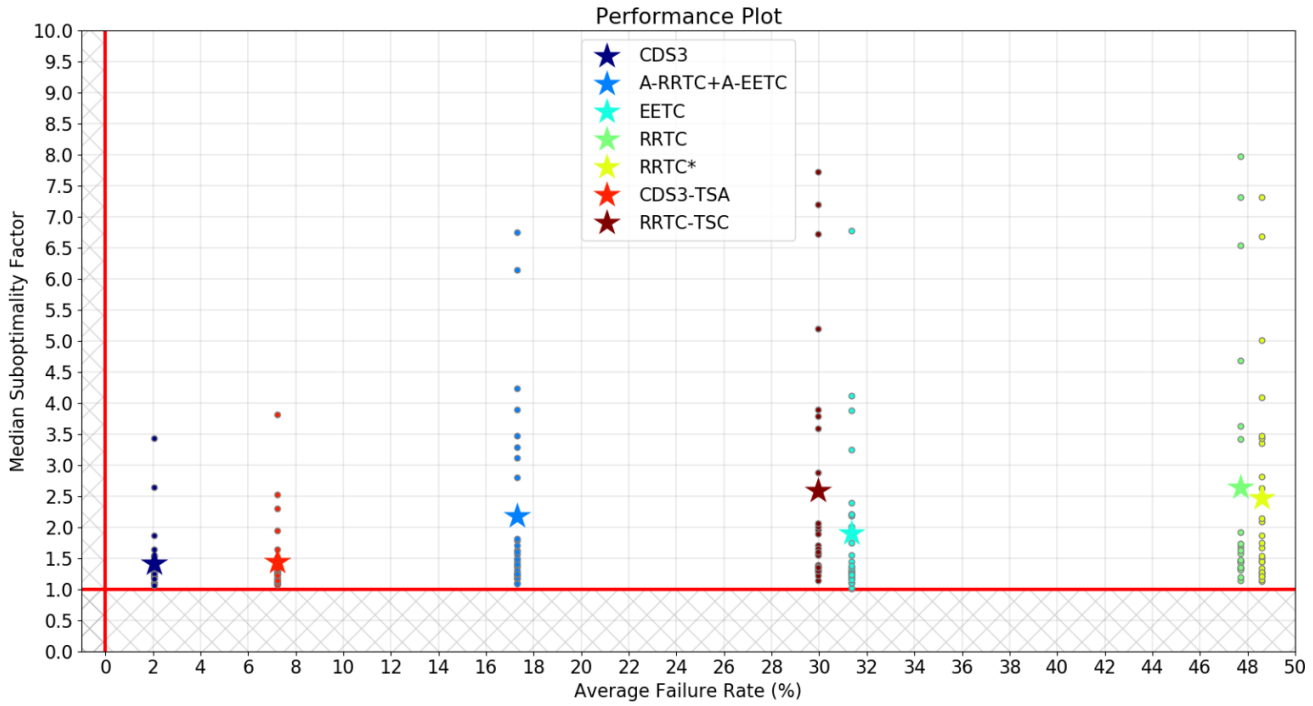


Fig. 5: Performance of CODES3 against alternative planners recorded over 50 (trials) planning queries of each of the 30 scenarios. A total of 1500 planning queries were made to each planner. For each planner, failure rate is measured by the percentage of failed planning queries over all planning queries made. A failed planning query happens when no plan is produced within 7.5 s (timeout)

times. Our experiments show that CODES3, an instance of the proposed framework, yields high-quality paths quickly in challenging test scenarios.

References

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug 1996.

[3] A. M. Kabir, B. C. Shah, and S. K. Gupta, "Trajectory planning for manipulators operating in confined workspaces," in *2018 IEEE International Conference on Automation Science and Engineering (CASE)*, Munich, Germany, Aug 2018.

[4] P. Rajendran, S. Thakar, and S. K. Gupta, "User-guided path planning for redundant manipulators in highly constrained work environments," in *IEEE International Conference on Automation Science and Engineering (CASE)*, Vancouver, Canada, August 2019.

[5] P. Rajendran, S. Thakar, A. Kabir, B. Shah, and S. K. Gupta, "Context-dependent search for generating paths for redundant manipulators in cluttered environments," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, November 2019.