Perception-Constrained Robot Manipulator Planning for Satellite Servicing

Tariq Zahroof
Dept. of Mechanical Engineering
Stanford University
Stanford, CA 94305
281-940-9805
tzahroof@stanford.edu

Andrew Bylard
Dept. of Aerospace Engineering
Stanford University
Stanford, CA 94305
925-978-6146
bylard@stanford.edu

Hesham Shageer
Center of Excellence for Aeronautics and Astronautics
King Abdulaziz City for Science and Technology
Riyadh 12354, Saudi Arabia
904-679-1213
hshageer@kacst.edu.sa

Marco Pavone
Dept. of Aerospace Engineering
Stanford University
Stanford, CA 94305
650-723-4432
pavone@stanford.edu

***Abstract*— Satellite servicing is a rapidly developing industry which will require a number of advances in semi- or fully-automated space robotics capabilities. One upcoming mission example is the NASA Restore-L Robotic Servicing spacecraft, which is equipped with two 7-joint robotic manipulators for satellite to capture a satellite and perform a complex series of refueling tasks, including swapping between various end-effector tools stored on board. The manipulator must meet a number of difficult constraints, like collision avoidance and perception guarantees for object and end-effector tracking, which are time-consuming to design by hand. Thus, in this work we present an planning algorithm and software tool for automated, real-time computation of near-optimal, collision-free trajectories to abide by the constraints defined above. The tool framework generates trajectories from high-level objectives, and allows for problem-specific algorithm customization by hot-swapping sampling-based planner (SBP) and post-processing shortcut pairs. Additionally, we added collision-checking constraints to guarantee field-of-vision and line-of-sight of targets (eye-in-hand camera configuration) and the end-effector (eye-to-hand camera configuration). The tool is integrated with the ROS MoveIt! framework to in order to simulate, plan, and visualize the resulting trajectories. Although the framework is applied to the Restore-L mission in this paper, the tool can be applied to any fixed-base manipulator planning scenario with a similar class of constraints.**

TABLE OF CONTENTS

1. INTRODUCTION

Cumulatively, the space industry has lost billions of dollars due to satellite system failures [1] and decommissioning of functional satellites due to depletion of fuel [2]. This presents an open opportunity for on-orbit satellite servicing, a broad field encompassing rendezvous, docking, and a variety of satellite repair and maintenance tasks. Currently, due to the large costs of human labor in space, satellite servicing operations often rely on robotic tools to complete tasks [3]. Thus a robust and streamlined workflow between operators and the robot is necessary to make full use of available robotic capabilities, reduce operator workload, and ultimately reduce costs. In particular, adding more autonomous or semi-autonomous functionality to these robotic tools is crucial to ensuring a streamlined concept of operations can be built around them.

To perform tooling and servicing tasks, space servicing technologies have often incorporated manipulator systems for complex object and environment interaction. Early 6-degree-of-freedom (DoF) systems, such as the Canadarm and the European Robotic Arm (ERA) [4], [5], provided basic SE(3) end-effector interaction using the minimum number of manipulator joints. These led to later systems, such as the 7-DoF Canadarm2 and the 15-DoF Special Purpose Dexterous Manipulator (SPDM) for the International Space Station, which increased the number of DoFs to improve singularity avoidance and collision avoidance capabilities [6], [7]. Unfortunately, these newer manipulator systems still made use of only astronaut teleoperated control or pre-designed trajectories, forcing operators to personally direct task-specific end-effector movements around obstacles [8]. Designing these trajectories is very difficult and tedious, particularly when operating in cluttered environments and given redundancy in the manipulator, and currently a new trajectory must be crafted manually for each task.

In some areas of on-orbit satellite servicing, autonomous capabilities have been studied and applied with success. For example, research in Autonomous Rendezvous and Capture (AR&C) has tackled the entire process of localization, navigation, capture, and stabilization of target satellites with minimal human intervention [9]. Example work includes Jewison et al.'s integration of satellite and spacecraft and target properties [10] and Steinberg's motion synchronization with tumbling objects during docking procedures with uncooperative targets [11]. However, once a target object has been captured, there is limited work on implementing autonomy on robot manipulators to complete the satellite servicing tasks.

978-1-5386-6854-2/19/$31.00

Fortunately, within the broader robotics community, such automated algorithms for guiding robotic manipulators are more readily available. In early work, Khatib introduced artificial potential fields to model obstacles as repulsive surfaces within the classical manipulator torque control framework [12]. This approach had some success for simple obstacle avoidance, but in more cluttered environments, the method would get stuck in unsuccessful local minima, halting manipulator movement or leading it into dead ends. Later work includes the broad field of trajectory optimization, which makes use of advances in convex and nonconvex solvers. Well-known algorithms of this class include CHOMP [13], STOMP [14], and as TrajOpt [15]. These approaches have been quite successful in providing better manipulator trajectories for more cluttered environments, sometimes even given very simple, naive initializations. However, their dependence on initialization is quite severe, and when planning in very cluttered environments or on complex dynamical systems, the task of finding a seed trajectory that leads to a successful or desirable solution is a difficult one on its own [16]. Often this results in a great deal of wasted time spend using the algorithm to refine unsuitable initializations without success.

Another available option is sampling-based planning (SBP), which performs a global search for near-optimal, collision-free robot trajectories. SBP algorithms sample collision-free robot configuration states (called nodes or waypoints interchangeably within this paper) from the configuration space of the robot and connects them with robot subpaths (or edges) once these subpaths are approved by a collision-checker. The scheme for choosing nodes and edges is often designed to result in full trajectories that optimize some cost function, and since the collision-checker can be considered as a black-box, SBP approaches are amenable to robot planning problems having a wide variety of obstacles and other constraints. SBP has been particularly successful for high-dimensional robots requiring fast computation of collision-free geometric paths [17]. Within space robotics, SBP has historically been used for tasks such as attitude control [18], docking [19], multi-robot control [20], and manipulator control in the pre-capture phase [21], but not for fine robotic satellite servicing tasks. However, SBP approaches alone are not always sufficient for this purpose, as although they are adept at quickly generating coarse near-globally-optimal paths, fully refining trajectories to optimality using SBP algorithms can be very computationally intensive.

This gap can be closed by various post-processing techniques, used to improve path quality. This can be accomplished using the class of previously discussed trajectory optimization approaches. However, quick quality improvements can also be achieved for a SBP-generated path via simple approaches such as path pruning and shortcutting by removing unnecessary waypoints [22].

When designing trajectories for spacecraft manipulation tasks, there are often special requirements which may be desired by mission designers and operators. One such example which we address in this paper is the ability to guarantee particularly desired camera visibility throughout a manipulator trajectory. For example, it may be desired that a supervising camera can always keep the manipulator end-effector in view or that an end-effector-mounted camera keep a particular target in view while repositioning. Though previous work has incorporated perception information into manipulator control (e.g. visual-servo control, using perception data within a manipulator position control loop [23], [24], [25]), this particular perception consideration applied as a hard constraint to generated manipulator trajectories has not yet been addressed. In particular, it would be desirable to incorporate this constraint smoothly into an existing planning pipeline without sacrificing much planner speed.

Statement of Contributions

To meet these demands, this paper makes two key contributions. First, we devise a line-of-sight perception constraint for a SBP algorithm to generate trajectories which guarantee a servicing robot manipulator will retain view of a target throughout a maneuver. Second, we develop a SBP and trajectory refinement pipeline and tool to rapidly generate collision-free trajectories of a satellite-servicing robotic manipulator, applying it to the example of the Restore-L satellite servicer performing a refueling task. Detailed simulation data is provided to ensure that the best combination of SBP algorithm and trajectory refinement algorithm is chosen.

Paper Organization

The paper is organized as follows: In Sec. 2 we review the motion planning problem solved by the tool, and detail SBP and the refinement algorithms evaluated for use in the tool. Then, in Sec. 3, we discuss the perception constraint and its integration into the planning tool. Next, in Sec. 4, we provide a high-level overview of the manipulator trajectory generation tool, which is the main contribution of the paper, as well as further implementation details of the tool. Finally, Sec. 5 presents simulations evaluating SBP and refinement algorithm combinations in various environments and Sec. 6 discusses the results.

2. Problem Formulation and Algorithms

A. The Motion Planning Problem

SBPs generate plans by connecting feasible states of the robot in the configuration space. The configuration space, or C-space, represents all possible configurations of the manipulator (e.g. joint angles and velocities). The C-space is further divided into two regions: the obstacle-space, C_{obs}, representing the infeasible robot states due to collisions, and its complement, the free-space, C_{free}. To clarify, collision states do not only include unwanted robot-object contact, but also violation of problem constraints (for instance, breaking visual line-of-sight).

The motion planning problem is defined as the determination of a robot action trajectory $u(t)$ yielding a feasible C-space path $x(t)$ such that $x(t) \in C_{\text{free}}$ within a specified time horizon $t \in [0, T]$, which satisfies $x(0) \in X_{\text{start}}$, $x(T) \in X_{\text{goal}}$, and minimizes the cost $J = \int_0^T g(x, u)dt$ [26]. In other words, a successful motion planner will provide a feasible path ($x(t) \in C_{\text{free}}$) to link the starting configuration, $x \in X_{\text{start}}$ to the final configuration, $x \in X_{\text{goal}}$, within a specified amount of time, T.

A motion planner may additionally attempt to find an optimal solution. Let $g(x, u)$ represent the cost to move from one configuration to another, i.e. penalize motions by some criteria (for this paper, path length in the joint space). Thus, an optimal motion planner will provide a set of controls, $u(t)$ that minimizes the total cost along the path.

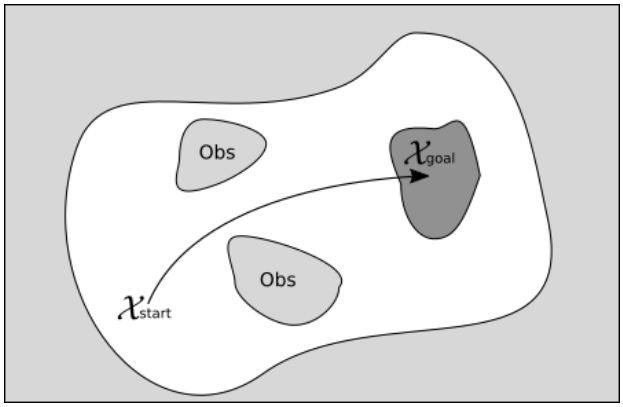


Figure 1: The motion planning problem

B. Sampling-Based Motion Planning Algorithms

There exists a wide variety of sampling-based motion planners. For this work, we chose three of the most commonly used classes of SBP approaches, described at a high level below. Of these five total variants were chosen for simulation comparisons, as described in Sec. 5.

Rapidly-Exploring Random Trees (RRT)— RRT is an SBP algorithm that performs simultaneous graph construction and search, incrementally building a tree of feasible paths through the space. At each step, RRT randomly samples nodes within the C_{free}, and then attempts to connect the node to the closest, feasible node in the tree [27].

We use two variants of RRT for benchmarking: The first variant is RRTConnect, which is a bi-directional version of RRT, growsing two trees from the start and goal towards one another e plannto decreasing time [28]. Second, we use RRT*, an asymptotically optimal version of RRT that converges towards the globally optimal solution as the number of samples increases by rewiring connections between nodes to maintain a tree of optimal paths [29].

Probabilistic Roadmaps (PRM)—PRM takes a different approach from RRT, first representing the free space by sampling a batch of collision-free nodes and connecting nodes to all their k-nearest neighbors with collision-free edges where possible. This creates a graph or "roadmap," which can be queried for optimal paths using standard graph-search algorithms (e.g. Djikstra's) [30]. Furthermore, PRM*, an asymptotically optimal variant, improves path quality and uses fewer collision checks by re-wiring and scaling the number of nearest neighbors considered. In this case, the main computational bottleneck is the full creation of a collision-free roadmap before beginning a path search. A exhaustive roadmap is not required for most planning problems, and if obstacles or constraints change, the roadmap must be regenerated.

Fast Marching Tree (FMT)*— Like RRT, FMT* performs graph construction and search simultaneously. However, FMT* makes use of "lazy" collision, checking, reducing the total number of computationally expensive edge collision-checks needed to solve a planning problem, compared to RRT and PRM variants. In addition, by operating on a fixed batch of collision-free nodes sampled in advance, still able to guarantee asymptotic optimality using dynamic programming [31].

We used FMT* and a bi-directional variant of FMT* called BFMT*. Like RRTConnect, BFMT* grows two trees, one from the start and one from the goal, towards each other [32]. By alternating between two trees, the number of nodes on the search frontier is reduced, reducing computation compared to FMT*. The algorithm can use different strategies for alternating between the two trees, such as by taking turns ("Alternating Trees") or by operating on the tree with the lowest-cost frontier node from the tree's root ("Balanced Trees"). Additionally, the algorithm termination criteria can either be the first connection between the two trees returned ("First Path"), or when the trees have expanded sufficiently far into one another ("Best Path"). We used the Balanced Trees and Best Path combination for improved path quality.

C. Shortcutting Refinement Algorithms

Additionally, a variety of shortcutting methods are available for refining coarse paths generated by SBP algorithms. The simplest shorcutting methods are corner cutting and path pruning techniques, which only remove single nodes in a path and attempt to reconnect the remaining nodes. However, there are a number of more complex variants, of which we chose four for comparison.

Shortcut— The first shortcutting technique, Shortcut, improves upon simple corner cutting and path pruning techniques by attempting to optimally connect two random nodes along a trajectory via straight line. If the path is feasible, then the former intermediate nodes are discarded, and the shortcut is cemented with interpolated points [22].

Adaptive Shortcut— The Adaptive Shortcut method is the same algorithm as Shortcut, but it also uses an oracle to improve obstacle hugging [33]. The oracle inserts new nodes between trajectory waypoints to provide new start and end locations for straight-line replacements. As a result, Adaptive Shortcut runs Shortcut numerous times, increasing computational burden.

Partial Shortcut—The Partial Shortcut method provides better path optimization by employing Shortcut on a specific manipulator joint. After selecting two random nodes n_i and n_j along a trajectory and selecting a specific joint x_i to shortcut, Partial Shortcut replaces the x_i joint values of the nodes along the trajectory from n_i to n_j with the interpolation of x_i from n_i to n_j. Due to more incremental shortcutting along a single manipulator joint, Partial Shortcut provides higher quality paths compared to regular Shortcut, at the cost of more iterations.

Adaptive Partial Shortcut—The final shortcut method, Adaptive Partial Shortcut, uses a weighted distribution to prioritize joints during Partial Shortcut [34]. Specifically, the technique builds a distribution by subtracting the cost of the current trajectory from the optimal, obstacle-free cost from start state to goal state. Using these weights, a random manipulator joint is chosen to optimize. Consequently, Adaptive Partial Shortcut offers a lower iteration count by prioritizing large cost-inducing joints. Unfortunately, the algorithm may get stuck repeatedly prioritizing already fully-optimized joints that significantly contribute towards cost.

3. PERCEPTION CONSTRAINTS

The objective of the perception constraints is to guarantee manipulator motions which meet desired constraints on camera visibility. One such constraint is an "eye-in-hand"

perception constraint, which provides continual visibility of a desired target from a camera mounted on the end-effector. This would allow operators to perform an additional visual check of trajectory accuracy and gain better insight of target-manipulation interactions during complex task sequencing. Another perception constraint is an "eye-to-hand" constraints, where the camera is attached somewhere else in the environment, such as on a second static arm or on the body of a spacecraft. Here, the goal is to restrict end-effector movement to within the field-of-view of the camera. Similarly, this limitation provides increased supervision and understanding for an observing operator. Both constraints require constant field-of-view (Figure 2) and line-of-sight (Figure 3) for the target.

To enforce these constraints within a SBP pipeline, we simply add them to the collision-checking process. As such, samples are considered in collision if either of the constraints are violated in addition to obstacle contact. The two types of constraints are formulated and applied as follows:

A. Field-of-View Constraint

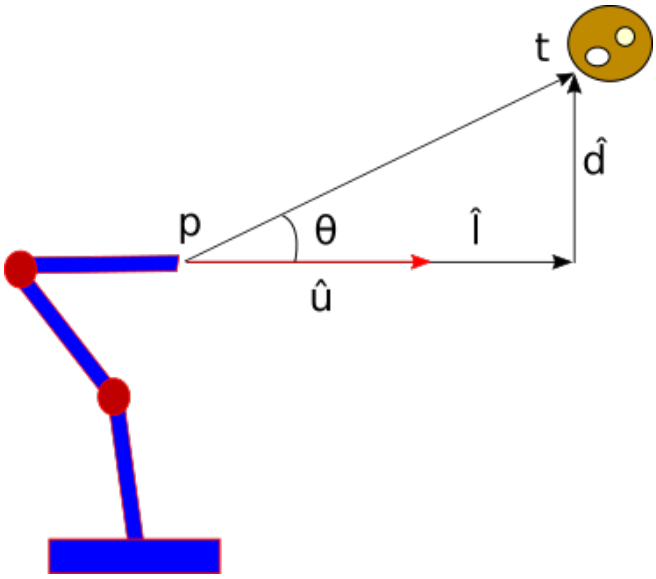


Figure 2: Diagram of a field-of-view constraint from end-effector camera to target

An example diagram of a field-of-view cone constraint can be seen in Figure 2. Let p represent the camera's location and t represent the target location. If $\hat{u}$ represents the unit vector along the direction the camera is facing, then the vector l along the camera's normal vector to the target is

$$l = (t - p) \cdot \hat{u} \tag{1}$$

The vector d to the target point within the tangent plane to the camera's direction is determined by

$$d = t - (p + \hat{l}) \tag{2}$$

Thus, the cone constraint is satisfied if d is less than the field-of-vision dictated by the camera angle, θ.

$$d \leq l \times \tan(\theta) \tag{3}$$

B. Line-of-Sight Vision Constraint

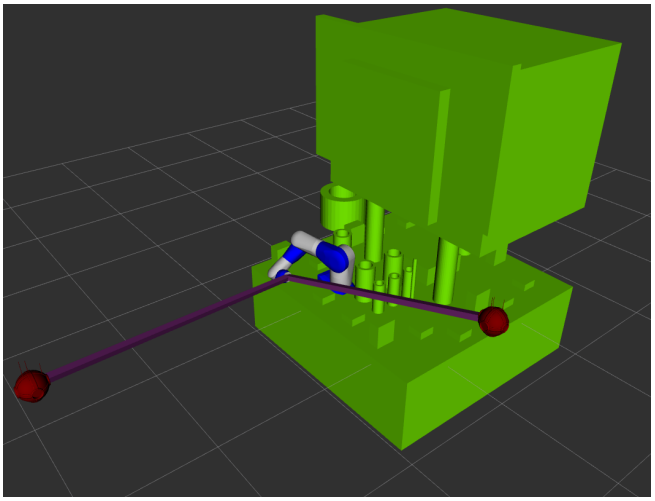


Figure 3: Purple rectangular prisms show the line-of-sight constraint from the end-effector to the red target and environmental camera

To accomplish the line-of-sight criterion, a thin rectangular prism is added to the end-effector to ensure no obstacles are present between the target and the camera. To determine the rotation of the prism, let $\hat{z}$ represent the unit direction of the long prism axis. The axis of rotation $\hat{a}$ and the angle of rotation ϕ are then found by:

$$\hat{a} = \hat{z} \times \frac{t - p}{\|t - p\|} \tag{4}$$

$$\phi = \cos^{-1}(\hat{z} \cdot \frac{t - p}{\|t - p\|}) \tag{5}$$

Finally, the quaternion q rotating the prism into place is calculated by

$$q = \begin{bmatrix} \frac{\hat{a}}{\|\hat{a}\|} \times \sin(\frac{\phi}{2}) \\ \frac{\hat{a}}{\|\hat{a}\|} \times \cos(\frac{\phi}{2}) \end{bmatrix} \tag{6}$$

4. The Developed Tool

The motion planning pipeline we developed was built on the ROS MoveIt! tool in order to easily integrate shortcut post-processing techniques with existing Open Motion Planning Library (OMPL) SBP algorithms. Furthermore, we modified the core MoveIt! collision-detection system to add optional perception constraints for target and/or end-effector tracking during motion execution.

The tool's modularity allows for the integration of personal planners and the customized pairings of SBPs and shortcuts. As a black-box solver, the tool "works" for any robotic application, regardless of the robot's number of joints or the obstacle difficulty, as long as the robot specifications, obstacle environment, and the start and desired end states are provided in a fixed-base planning scenario. The tool may be repeatedly queried by a system for real-time planning applications, assuming the aforementioned information is updated. Additionally, the generated trajectory may then be viewed through RViz for final operator verification.

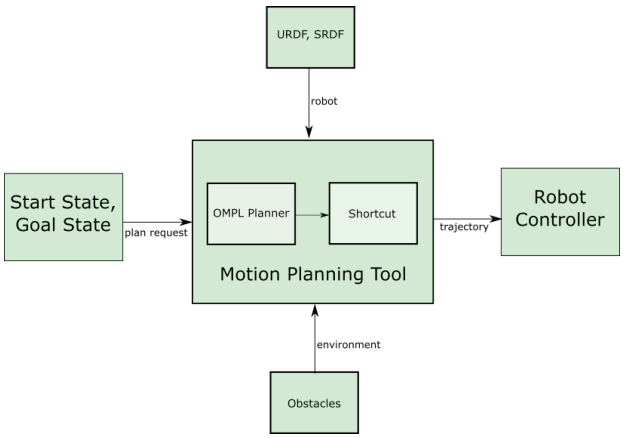


Figure 4: Diagram showing the tool's motion planning workflow

To quickly generate trajectories, MoveIt! divides the kinodynamic problem into two stages. During the first stage, MoveIt! determines kinematic trajectory waypoints (manipulator joint configurations) by geometric planning. Then dynamic constraints are satisfied to generate a full trajectory by using time parameterization. To ensure finer velocity and acceleration control, the tool injects waypoints into the trajectory by interpolation before time parameterization.

Through the simulation experiments of Section 6, we concluded that BFMT* and Partial shortcut generate the optimal combination of fast yet high quality paths for planning scenarios with one, two, and no perception constraints. However the framework permits operators devise their own SBP and shortcut pairing for problem-specific objectives. Figure 4 demonstrates the problem-solving workflow of the tool.

5. EXPERIMENTS

To test the planning pipeline and evaluate different SBP and shortcutting combinations, we simulated a 7-DoF Motoman SIA20D manipulator arm navigating five different problem setups: three simple environments without perception constraints applied, a maze of rubble with an eye-in-hand perception constraint applied, and a mock-up of a satellite with both an eye-in-hand and hand-to-eye perception constraint applied. The three simple environment (a sphere environment, a box environment, and a cluttered environment as shown in Fig. 5), served as isolated test cases to provide base statistics on the planners in simple navigation tasks. The sphere environment offered minimal obstruction to demonstrate small path perturbations, the box environment forced the manipulator to stretch while moving around the box's extremities, and the cluttered environment offered numerous different pathways for the manipulator to reach the goal. The environment involving a maze of space rubble, shown in Fig. 6, forced the robot arm to navigate to a new vantage point to observe a target while navigating through an obstacle-dense scenario. This case was particularly challenging due to the requirement that the end-effector must always be facing the target. Finally, the satellite environment shown in Fig. 7) reflected the proper application environment for the planner, where an operator would want to track the end-effector's location via an environmental camera while the end-effector provides a moveable camera viewpoint for a specified target. In this task, the robot performed a cork-screwing motion to move forwards to settle near a tooling location in order to continuously guarantee end-effector and target visibility.

Environment	Max Time (s)
Sphere	1
Box	1
Cluttered	3
Maze	11
Satellite	22

Table 1: Max Planning Time for Obstacle Courses

As mentioned in Sec. 2, five SBP algorithms were selected for comparison: BFMT*, FMT*, RRTConnect, RRT*, and PRM*. FMT* was naturally chosen to benchmark the improvements provided by the bidirectional trees of its variant. The second planner, RRTConnect, is popular for rapid planning scenarios and is also MoveIt!'s endorsed planner. The third algorithm, RRT*, is a popular, asymptotically-optimal variant of RRT used in scenarios where path quality is essential. However, due to the number of costly collision checks, RRT* failed to provide plans in adequate time when vision constraints was added; thus, PRM*, another asymptotically-optimal algorithm was used to benchmark path quality.

The SBP solutions were then subsequently fed into one of four different shortcut methods for post-processing. For a negligible increase in time, post-processing algorithms offered further path quality improvements to the trajectory. However, as shown by the results, the path quality improvements decreased as the number of constraints increased, due to the reduction of the number of feasible intermediate states between trajectory waypoints.

Each algorithm and shortcut pair was run 100 times and averaged. Furthermore, a maximum time limit was imposed to prevent excessive planning times given poor sample draws. The time limit particularly affected RRT* and PRM*, which continuously attempt to improve a discovered plan until the time limit expires. The time limits are detailed in Table 1.

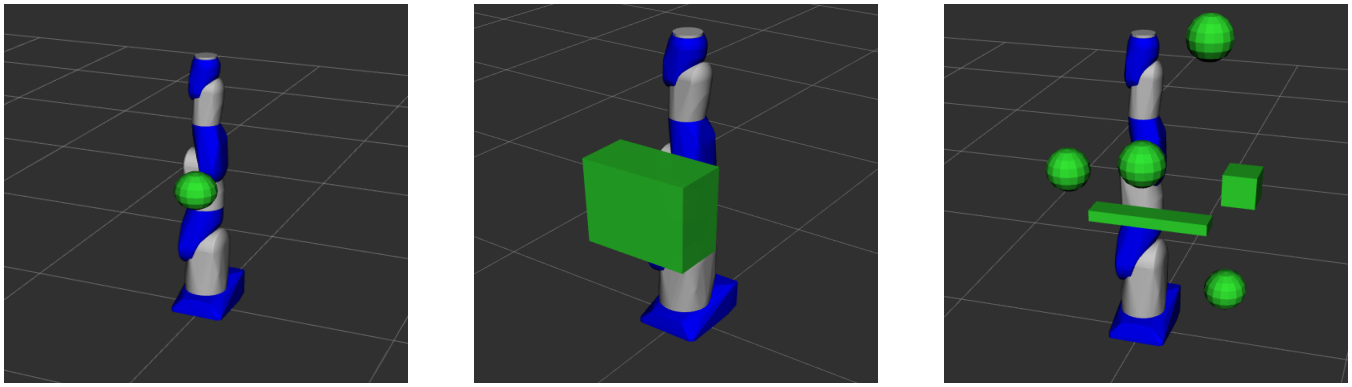
6. RESULTS AND ANALYSIS

SBP and shortcutting pairs were evaluated according to computation time, rate of success in finding a solution, and path quality according to some cost metric. For the manipulator planning, the cost metric was taken to be total distance in the joint space.

A. Three Simple Environments (No Vision Constraint)

The algorithms performed as expected in the three simple scenarios. Tables 8 and 9 show the average SBP time and cost, respectively, for the simple environment. Cost is shown as a comparison against the no-obstacle, optimal straight line path.

BFMT* yielded high-quality solutions using less computation time than its counterpart, FMT*. As the number of nodes in FMT*'s single tree increased, the dynamic programming algorithm had to satisfy a larger number of low-cost frontier nodes while searching for the goal location. On the other hand, BFMT* reduced the explored node count and thus planning time, by expanding from the goal and start region.



(a) Sphere environment (b) Box environment (c) Cluttered environment

Figure 5: Simple environments with no perception constraints

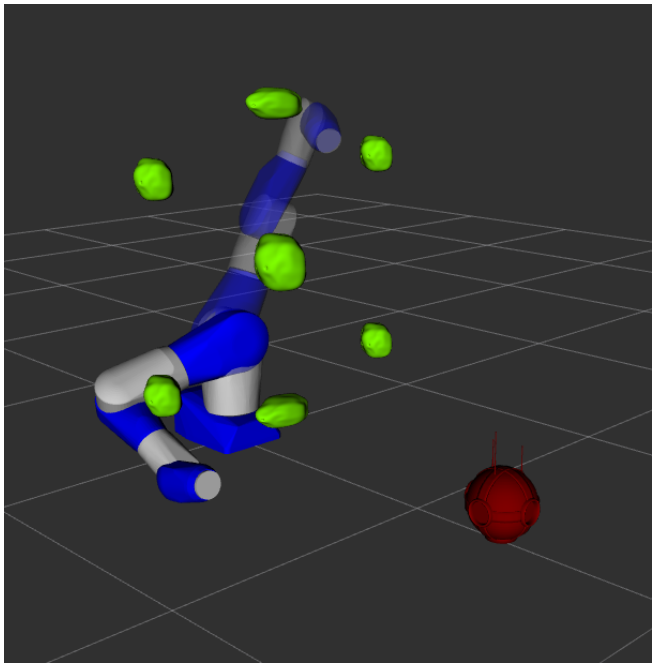


Figure 6: Maze environment with target perception constraint

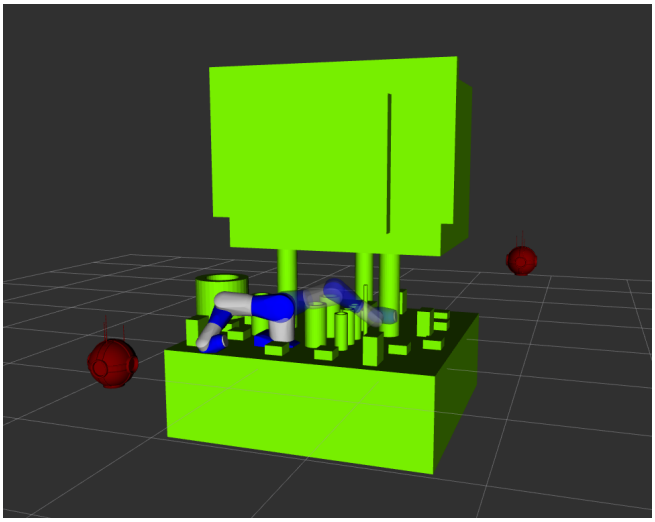


Figure 7: Satellite environment with target and environmental camera constraint

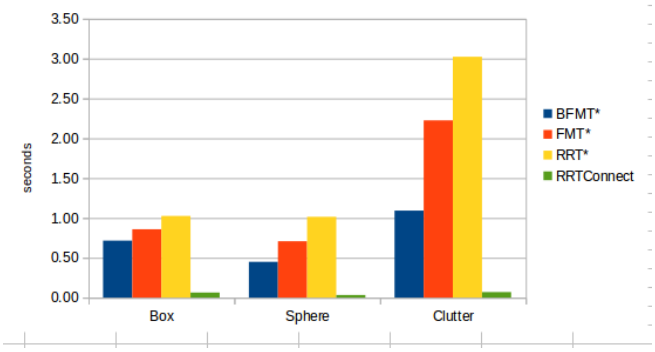


Figure 8: Average times of SBP algorithms for the simple environments

RRTConnect created paths in less than 10% of the computation time of its closest competitor, BFMT*. However, RRT-Connect paths suffered in quality, providing greatly increased path costs due to the algorithm's emphasis of *finding* a any feasible path over necessarily an optimal path. In practice this would lead to much higher energy expense and motion durations when executing trajectories, which is undesirable. On the opposite end of the spectrum, RRT* provided the highest quality paths at the cost of planning time. However, RRT* paths took significantly longer than BFMT* to perform rewiring operations and further develop the tree. Unfortunately, the rewiring operations and extra node expansions resulted in more collision checks, one of the biggest contributors to planning computation times. As such, even with the generous maximum allotted planning time, RRT* failed for simple planning scenarios (11%, 1%, and 4.63% for box, sphere, and cluttered environments, respectively).

As for post-processing algorithms, the average time and cost results for the simple environments are show in table 2. Shortcut and Adaptive Shortcut provided the weakest results for cost reduction at 5% and 8%, respectively. The poor performance was due to the algorithms' consideration

	Box		Sphere		Clutter	
Shortcut	Time (s)	Cost Red (%)	Time (s)	Cost Red (%)	Time (s)	Cost Red (%)
Shortcut	0.04	5	0.05	5	0.06	5
Adaptive	0.17	8	0.22	8	0.30	10
Partial	0.02	19	0.01	15	0.03	24
Adaptive Partial	0.01	19	0.01	13	0.03	23

Table 2: Post-processing times and cost for the simple environment

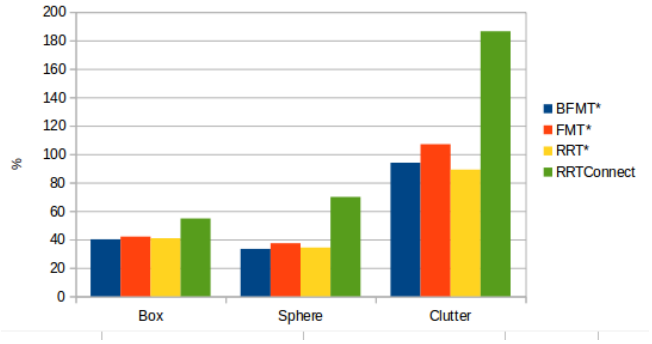


Figure 9: Increase in SBP cost (%) vs no-obstacle, optimal cost for the simple environments

of all manipulator joints when pruning. Adaptive Shortcut's oracle slightly improved cost by increasing the number of opportunities to make straight-line connections and therefore hug obstacles, but the resulting improvement was marginal compared to the drastic relative increase in computation time. Both techniques ultimately offered severely diminishing results as problem dimension size increased. Alternatively, partial shortcut and adaptive partial shortcut provided incredible cost improvements in cost, at around 15-25%, with minimal time increase by skipping collision-checking over already optimized path sections.

B. Maze Environment (Target Vision Constraint)

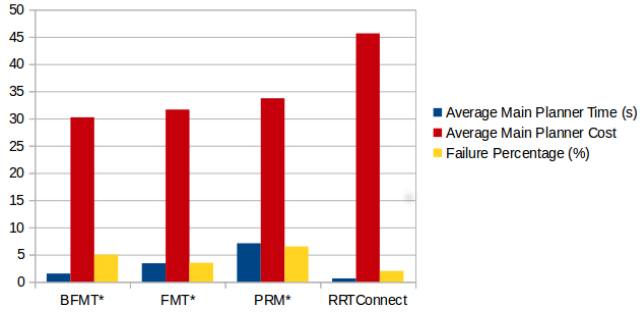


Figure 10: SBP times (s) , cost, and failure rates (%) for the maze

For the Maze environment, the SBP results are available in Fig. 10. It is worth noting that, RRT* was replaced in the comparison by PRM* due to performance issues. RRT*'s problems with collision-checking were heightened by the additional vision constraints, causing it to rarely succeed within the enforced time limits. Again, BFMT* outperformed FMT*, cutting planning time by more than half while also

Shortcut	**Avg Time (s)**	**Avg Cost Red (%)**
Shortcut	0.05	2
Adaptive	0.17	5
Partial	0.05	14
Adaptive Partial	0.04	13

Table 3: Post-processing times and cost for the maze

offering marginal cost improvements. This was also due to the increased cost of collision-checking from FMT*'s larger frontier. However, BFMT* incurred a slightly higher failure rate (5% vs 3.5%) after path validation, most likely due to the precision in collision-checking when transitioning between trees. In fact, all of the algorithms suffered some failure rate, with PRM* suffering the highest (6%) Also, despite PRM*'s high maximum planning time of 7s, it produced marginally worse paths compared to BFMT* and FMT*. Like RRT*, PRM* suffered from heavy collision-checking loads from its costly re-wiring procedure (i.e. many re-wiring procedures were ineffective or did not contribute to improving path quality). RRTConnect, the final benchmarking algorithm, offered the expected results of significantly worse path quality (around a 50% increase compared to BFMT*) at a fraction of the time. However, the time improvements were less appreciable, as it only performed at a third of the speed. This reduction in speed was due to the ultimate need of a larger number of nodes to determine path (in contrast, the FMT* algorithms draw a fixed number of nodes at the start of the algorithm).

In this environment the shortcut methods suffered reduced effectiveness, as straight-line connections between waypoints became infeasible due to vision constraint violation, as illustrated in Table 3. The Shortcut and Adaptive Shortcut approaches offered little improvement over the initial trajectory (2% and 5%, respectively), while Partial and Adaptive Partial shortcuts offered 13% and 14%. Adaptive Partial performed marginally worse, due to its optimization biasing towards manipulator joints that generate the greatest variation from the no-obstacle, optimal cost. This potentially biases the algorithm towards fully optimized manipulator joints that are inevitable sources of high cost. As such, increasing the number of loop iterations has slightly lesser effects on Adaptive Partial.

The full SBP and shortcut results are available in Figure 11.

C. Satellite Environment (Global and Target Vision Constraint)

The satellite environment induced the heaviest strain on the collision checking, effectively adding four more constraints (two field-of-view and two line-of-sight). The algorithms suffered increased planning time due to the reduced number

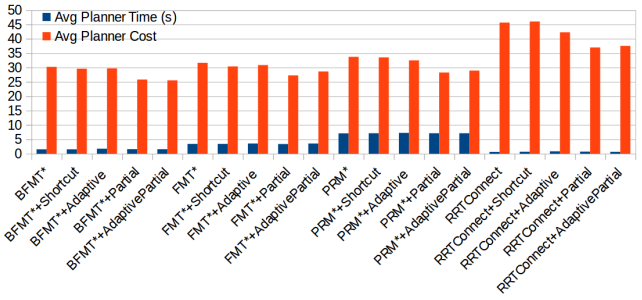


Figure 11: SBP with shortcut times and cost for the maze

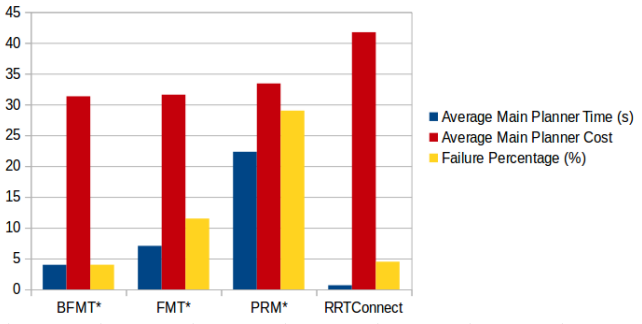


Figure 12: SBP times (s), cost, and failure rates (%) for the satellite

of viable states and the large number of obstacles on-board the satellite and chaser. Their results are available in Table 12. Since SBP's indirectly characterize obstacle spaces, all obstacles had to be checked against the manipulator. RRT-Connect again generated the plans the fastest. Interestingly, due to the reduced number of viable states, the path quality improved when compared to BFMT* (around 33% increase). FMT* performed worse than BFMT*, suffering a failure rate of 12% due to the number of available, expanding roots in the frontier. The bi-directional tree methods, BFMT* and RRTConnect, proved apt for these circumstances. PRM*, however, performed terribly, failing 30% of the time with a 22 second planning time. The sheer number of obstacles and limited path options were not suitable for the roadmap and re-wiring technique.

The shortcutting techniques yielded similar cost results to their maze counterparts, down only a couple more percentage points. However, their time contribution more than tripled. While still negligible (with the exception of Adaptive, which rose to 0.65 s) when compared to SBP times, it shows that the number of obstacles plays a major factor in planner efficiency. The shortcut results are tabulated in Table 4.

Shortcut	Avg Time (s)	Avg Cost Red (%)
Shortcut	0.16	2
Adaptive	0.65	4
Partial	0.14	11
Adaptive Partial	0.11	10

Table 4: Post-processing times and cost for the satellite

To further improve efficiency for both SBP and shortcut techniques, a simpler collision mesh was created for the satellite. The satellite was decomposed into various simple convex objects, allowing for faster and simpler obstacle collision-checking. Additionally, the representations were inflated in size for manipulator safety guarantees. The full SBP and shortcut results are available in Figure 13.

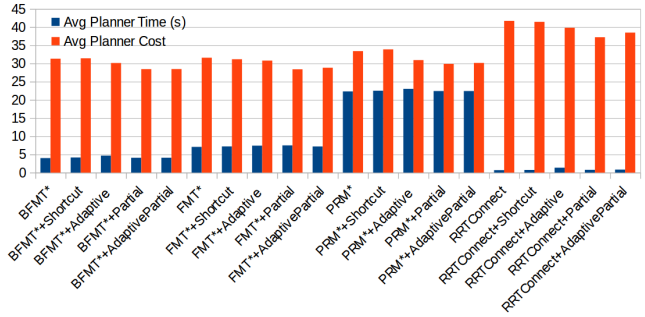


Figure 13: SBP with shortcut times and cost for the satellite

D. SBP and Shortcut Selection

From the results obtained, we determined that the BFMT*+Partial Shortcut approach offered the best combination of speed and path quality. In particular, BFMT*'s lazy dynamic programming approach combined with bi-directional trees helps deal with the heavy collision-checking loads induced by the perception constraints and the connection process for difficult start-to-goal paths. Additionally, the tool's modularity allows for easy SBP and shortcut hotswapping for any specific needs, as well as the robot and environment. Any robot dimensionality (within reason) is accommodated by SBP, although settings (particularly sampling, edge length discretization for collision-checking, and allowed planning time) will have to be adapted to the specific circumstances.

7. SUMMARY

In conclusion, we have developed a motion planning tool to generate real-time, collision-free trajectories for the satellite servicing manipulators such as the 7-DoF Restore-L robotic manipulators with perception constraints for target and end-effector visibility. This tool autonomously creates complex paths, replacing the previous procedure of hand-designing end-effector paths and/or trajectories during satellite-servicing procedures. Furthermore, the modularity of the framework allows for custom SBP and shortcut pairings to tailor algorithms to specific problem scenarios. Finally, the addition of perception constraints assists robot-operator interfacing, providing detailed visual information for nuanced task execution and unforeseen maneuvers.

We determined that BFMT* and Partial shortcut are the best combination of planning and post-processing strategies, respectively, provide thing optimal combinations of fast planning times, high success rate, and high path quality. Given the robot, the environment, and the desired start and goal positions, the tool will generate a collision-free trajectory that also respects the robot limits and constraints. As such, the tool is easily integrable into existing robotic operations.

There are several remaining challenges in the area of SBP to improve planner performance. Currently, the planner evenly

samples over a given workspace. Biasing the sampling to a focus on regions of interest to reduce the number of useless samples and increase the number of path-improving samples would both improve path cost and decrease computational loads of unnecessary collision checking. Future work could include the incorporation on learning sampling distributions for repeated environments interactions [35]. Another option is the representation of the joint space in lower dimensions to improve sampling coverage. Although 7 dimensions is well within reason for SBP, dimensionality becomes an issue as the number of DoFs increases over 15 (like the SPDM), as the samples density becomes scarce to cover the C-space. Through learning obstacle environments in latent spaces, the planner could discover non-intuitive trends within the joint space for lower-DoF representations of the C-space.

ACKNOWLEDGMENTS

We thank the team at the Satellite Servicing Projects Division at NASA Goddard for their helpful discussions and insight on satellite servicing operations and spacecraft manipulator considerations.

REFERENCES

[1] G. A. Landis, S. G. Bailey, and R. Tischler, "Causes of power-related satellite failures," in *Photovoltaic Energy Conversion, Conference Record of the 2006 IEEE 4th World Conference on*, vol. 2. IEEE, 2006, pp. 1943–1945.

[2] D. H. Martin, *Communication satellites*. AIAA, 2000.

[3] T. B. Sheridan, "Space teleoperation through time delay: Review and prognosis," *IEEE Transactions on robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.

[4] B. A. Aikenhead, R. G. Daniell, and F. M. Davis, "Canadarm and the space shuttle," vol. 1, no. 2, pp. 126–132. [Online]. Available: https://avs.scitation.org/doi/abs/10.1116/1.572085

[5] R. Boumans and C. Heemskerk, "The european robotic arm for the international space station," vol. 23, no. 1, pp. 17–27.

[6] R. Mcgregor and L. Oshinowo, "Flight 6a: Deployment and checkout of the space station remote manipulator system (SSRMS)."

[7] R. Mukherji, D. A Mdr, C. M Rey, C. J Stieber, L. , and M. , "Special purpose dexterous manipulator (SPDM) advanced control features and development test results."

[8] D. King, "Space servicing: past, present and future," in *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS*, 2001, pp. 18–22.

[9] D. Zimpfer, P. Kachmar, and S. Tuohy, "Autonomous rendezvous, capture and in-space assembly: past, present and future," in *1st Space exploration conference: continuing the voyage of discovery*, 2005, p. 2523.

[10] C. M. Jewison, B. McCarthy, D. C. Sternberg, D. Strawser, and C. Fang, "Resource aggregated reconfigurable control and risk-allocative path planning for on-orbit servicing and assembly of satellites," in *AIAA Guidance, Navigation, and Control Conference*, 2014, p. 1289.

[11] D. C. Sternberg *et al.*, "Optimal docking to tumbling objects with uncertain properties," Ph.D. dissertation, Massachusetts Institute of Technology, 2017.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Conf. on Robotics and Automation*, 1985.

[13] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: gradient optimization techniques for efficient motion planning," in *Proc. IEEE Conf. on Robotics and Automation*, 2009.

[14] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Conf. on Robotics and Automation*, 2011.

[15] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems*, 2013.

[16] J. Pan, Z. Chen, and P. Abbeel, "Predicting initialization effectiveness for trajectory optimization," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5183–5190.

[17] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[18] E. Frazzoli, M. A. Dahleh, E. Feron, and R. Kornfeld, "A randomized attitude slew planning algorithm for autonomous spacecraft," in *AIAA Conf. on Guidance, Navigation and Control*, 2001.

[19] R. I. Zappulla, J. Virgili-Llop, and M. Romano, "Near-optimal real-time spacecraft guidance and control using harmonic potential functions and a modified RRT." [Online]. Available: https://calhoun.nps.edu/handle/10945/51981

[20] J. Cortes and T. Simeon, "Sampling-based motion planning under kinematic loop-closure constraints," in *Algorithmic Foundations of Robotics VI*, M. Erdmann, M. Overmars, D. Hsu, and F. van der Stappen, Eds. Springer Berlin Heidelberg, vol. 17, pp. 75–90.

[21] F. James, S. V. Shah, A. K. Singh, K. M. Krishna, and A. K. Misra, "Reactionless maneuvering of a space robot in precapture phase," vol. 39, no. 10, pp. 2419–2425. [Online]. Available: http://arc.aiaa.org/doi/10.2514/1.G001828

[22] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007. [Online]. Available: https://doi.org/10.1177/0278364907079280

[23] N. P. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision," *IEEE transactions on robotics and automation*, vol. 9, no. 1, pp. 14–35, 1993.

[24] S. Morikawa, T. Senoo, A. Namiki, and M. Ishikawa, "Realtime collision avoidance using a robot manipulator with light-weight small high-speed vision systems," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 794–799.

[25] K. Hosoda, K. Sakamoto, and M. Asada, "Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3d reconstruction," *Journal of*

the Robotics Society of Japan, vol. 15, no. 2, pp. 290–295, 1997.

[26] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

[27] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Workshop on Algorithmic Foundations of Robotics*, 2000.

[28] J. J. Kuffner and S. M. LaValle, "RRT-connect: an efficient approach to single-query path planning," in *Proc. IEEE Conf. on Robotics and Automation*, 2000.

[29] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *CoRR*, vol. abs/1005.0416, 2010. [Online]. Available: http://arxiv.org/abs/1005.0416

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

[31] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast Marching Tree: a fast marching sampling-based method for optimal motion planning in many dimensions," *Int. Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[32] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2015.

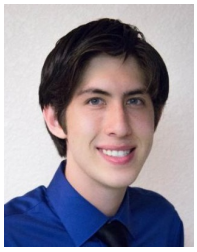
[33] D. Hsu, "Randomized Single-query Motion Planning in Expansive Spaces," *PhD thesis, Dept. of Computer Science, Stanford University, Stanford, CA*, 2000.

[34] J. Polden, Z. Pan, N. Larkin, and S. van Duin, "Adaptive partial shortcuts: Path optimization for industrial robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 1, pp. 35–47, Apr 2017. [Online]. Available: https://doi.org/10.1007/s10846-016-0437-x

[35] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. IEEE Conf. on Robotics and Automation*, 2018.

BIOGRAPHY

Tariq Zahroof *is a graduate student in Mechanical Engineering at Stanford University. He received his B.S. in Mechanical Engineering from UT Austin in 2017. His current research interests include real-time motion and task planning for space manipulator arms in uncertain environments, design of robotic systems, and dynamic grasping and object manipulation.*



Andrew Bylard *is a Ph.D. student in Aeronautics and Astronautics at Stanford University. He received his B.Eng. with a dual concentration of Mechanical Engineering and Electrical Engineering from Walla Walla University in 2014, and a M.Sc. in Aeronautics and Astronautics from Stanford University in 2016. He is currently a NASA Space Technology Research Fellow. Andrew's research interests include real-time spacecraft motion-planning under uncertainty and use of gecko-inspired controllable dry adhesives for in-space grasping and manipulation. He is currently a lead developer at the Stanford Space Robotics Facility, which houses test beds used to perform spacecraft contact dynamics experiments and demonstrate autonomous spacecraft robotics and proximity operations under simulated frictionless, microgravity conditions.*

Hesham Shageer *is an Assistant Research Professor at the King Abdulaziz City for Science and Technology (KACST). He is one of the Principal Investigators (PI) with the collaborative Center of Excellence for Aeronautics and Astronautics (CEAA) a collaborative research agreement between KACST and Stanford University. His research experience includes Adaptive Control Theory and Design. Mathematical System Modeling and Simulation, Product Design and Development as well as Experimental and Computational Methods. Hesham received his B.S. (2004), M.S. (2006) and Ph.D. (2013) degrees from the University of Virginia (UVA) with a major in Electrical Engineering and a specialization in Control Systems. While at UVA, he participated on a novel aircraft control design project funded by the National Aeronautics and Space Administration (NASA) as well as the Solar Decathlon Competition sponsored by the U.S. Department of Energy. His research interests span multiple domains including Machine Learning Based Adaptive Control Synthesis, Autonomous Aircraft System Design, Green Engineering and Bio-mimicking Design Concepts.*

Dr. Marco Pavone *is an Assistant Professor of Aeronautics and Astronautics at Stanford University, where he is the Director of the Autonomous Systems Laboratory. Before joining Stanford, he was a Research Technologist within the Robotics Section at the NASA Jet Propulsion Laboratory. He received a Ph.D. degree in Aeronautics and Astronautics from the Massachusetts Institute of Technology in 2010. Dr. Pavone's areas of expertise lie in the fields of controls and robotics. His main research interests are in the development of methodologies for the analysis, design, and control of autonomous systems, with an emphasis on autonomous aerospace vehicles and large-scale robotic networks. He is a recipient of an NSF CAREER Award, a NASA Early Career Faculty Award, a Hellman Faculty Scholar Award, and was named NASA NIAC Fellow in 2011. He is currently serving as an Associate Editor for the IEEE Control Systems Magazine.*