Manipulator Arm Motion Planning with Temperature Constraints

Joseph A. Starek, Marco Pavone

***Abstract*— Real-time guidance and control of a manipulator arm with embedded sensory networks is developed, with specific application to time-varying temperature obstacle avoidance. A real-time, closed-loop sampling-based anytime algorithm was written to simultaneously identify heated regions and construct a feasible motion plan in order to achieve the task of grasping and depositing a battery into a cup. Results from experimental trials are presented and the effectiveness of the modified Rapidly-exploring Random Tree (RRT*-Connect) approach is discussed. This work was done in support of the carbon nanotube and piezoelectric materials research of the Stanford Structures and Composites Laboratory.**

I. Introduction

This paper describes the motion planning algorithm selected for control of the Stanford Structures and Composites Laboratory (SACL) manipulator arm. The SACL manipulator arm, seen in Figure 1, was developed as an experimental testbed for cutting-edge embedded sensory networks. The approach-sensing skin is composed of a thick foam enveloped by a suite of resistance temperature detectors (RTD's), carbon nanotube (CNT) pressure sensors, strain gauges, and lead zirconate titanate (chemical formula $Pb\left[Zr_xTi_{1-x}\right]O_3$, abbreviated PZT [1]) piezoelectric sensors. Carbon nanotubes, a booming area of recent materials research, are currently being explored as pressure transducers due to their uniquely tailorable structure that enables them to act as either a metallic conductor or as a semiconductor, depending on their lattice orientation [2], [3]. PZT materials are ceramic compounds with piezoelectric, pyroelectric [4], and ferroelectric properties, meaning that they have the ability to develop a potential difference in response to physical deformation (piezoelectricity) and temperature change (pyroelectricity), and can spontaneously develop an electric polarity that can be altered by external magnetic fields (ferroelectricity). These remarkable properties, in addition to their low cost and light weight, have led to their widespread use as actuators and sensors in various "electroceramic" technological applications [1], including structural health monitoring of plates, beams, and aircraft wings [5].

Given this embedded sensor network, the task of the SACL robotic arm control system is to:

1) Develop a motion plan that will transfer an object from its initial location to a desired target location.
2) Measure the temperature of the target object and assess whether it is safe to grasp.

This work was supported by the Stanford University Department of Aeronautics and Astronautics.

Affiliation and e-mail: `Graduate student, jstarek@stanford.edu`

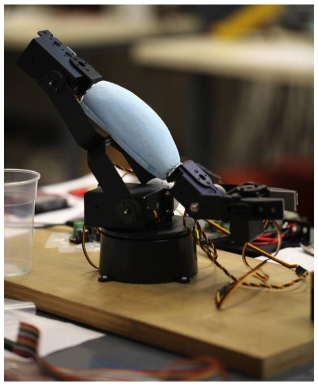


Fig. 1: Manipulator arm with prototype foam sensor material

3) Identify and avoid any "hot" temperature disturbances and choose an alternative path to the goal, if necessary.

Applications of these novel and unique capabilities include, for example, robotic assembly, bomb defusal, and search-and-rescue operations. As a simple experiment, the task imposed on the arm is to deliver a simple 9V battery to a standard-sized 8-ounce plastic cup. The experimental setup currently proposed is shown in Figure 2. Once initiated, the robotic arm must develop a motion plan that will position its end effector near the battery, attempt to grasp it, maneuver the battery above the cup, release it, and finally return to its original position, all while avoiding collisions with its base or knocking over the cup. A temperature disturbance, directed in the vicinity of the arm, will be introduced at some a priori unknown time during the experiment. The arm will be required to react to this disturbance based on the estimates it receives from the embedded sensors and determine an alternate feasible trajectory to the goal.

Due to the time-varying nature of the manipulator environment (imposed by the generation of the disturbance), a motion plan must be implemented in real-time and with constant re-evaluation along the path. Waypoint navigation, holding event identification, and temperature profile estimation of the local environment are all necessary to the success of the approach. The challenge of this work is to determine an effective on-line algorithm with anytime computation that can make these grasping and path-planning decisions in real-time while simultaneously avoiding the violation of any maximum temperature constraints on the sensors.

A. *Literature*

The concept of motion planning (MP) as a tool for controlling manipulator arm robots is not a new one – the two subjects have been tightly interconnected since the introduction

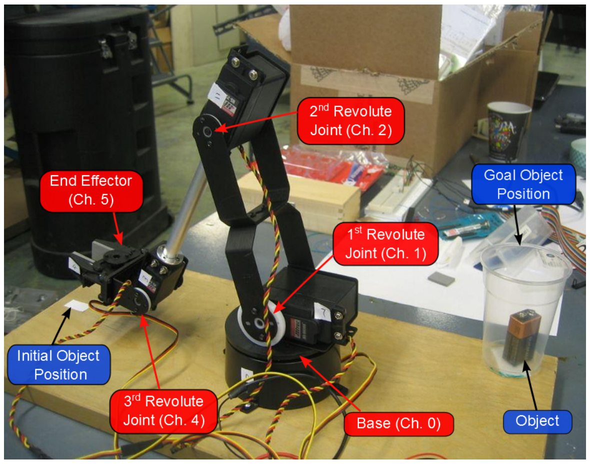


Fig. 2: Configuration of the Stanford SACL manipulator, with the target object (a 9V battery) positioned at the goal.

of planning algorithms in the 1960's. Particularly following the rise in popularity of MP algorithms in the early 1980's after the seminal work of Lozano-Perez et al. [6], motion planning has been an indispensable tool for robotic path planning in general [7]. One particularly strong explanation for this is their ability to verify and guarantee obstacle avoidance in a straightforward algorithmic framework. In fact, the primary advantage of motion planning over many other techniques is its elegant simplicity, whereby a motion planner converts a complex control problem into a geometric one. This not only makes for easier implementation but also facilitates the use of existing computationally efficient and well-established algorithms, such as graph-search techniques and geometric algorithms.

The study of the autonomous control of manipulators has been a longstanding practical application of the theory that has and still continues to motivate further research. The number of available techniques are vast, particularly for control of robots such as manipulators with purely kinematic constraints. A few representative examples are described here.

- In [8], Lozano-Pérez describes a simple algorithm for manipulator planning in stationary environments that effectively maps the locations of C-space obstacles by fixing joint angles one by one, progressively developing a collision-free "plan" for the first portion of the manipulator, then the second, and so on until a collision-free path is found for the entire manipulator. In high-dimensional problems, the approach relies on a conservative representation of the link geometry to reduce the dimensionality. The key advantage is that only subsets of the C-space are explored, allowing faster computation. Though easy to implement, the algorithm suffers by the need to map the configuration space and may miss feasible plans due to its over-approximation of the robot geometry.
- In [9], Chen and Hwang propose a sophisticated, two-level planner for robotic motion planning that employs dynamic graph-search. A global planner builds a coarse graph of points in the C-space and tracks their reachability, while a local planner tracks which sections of space can connect to the graph. To improve efficiency without sacrificing resolution-completeness, a plan is developed by first loosely searching for a path among promising areas of the graph (identified as "subgoals" if far from obstacles and easy to reach), only increasing resolution if no feasible path can be found. The algorithm scales well with problem complexity, has general applicability, and appears to work well for problems with under 13 DOF, as illustrated by several manipulator, rigid object, and multiple rigid body simulations. The major drawbacks of their approach include the assumption of polyhedral robots and obstacles, the neglect of dynamics, and the difficulty of implementation.
- Carriker et al. [10] consider a mathematical programming approach to the problem of planning a path for a mobile manipulator constrained to achieve a given sequence of end-effector tasks while limited by maximum and minimum joint angles and torques. Using a cost functional with mobility costs independent of arm costs, simulated annealing is used to identify near-optimal solutions. Despite a highly nonlinear global optimization problem with nonconvex constraints, most solutions appear to be within 15% of the optimal cost.
- A rather unique approach involving variational dynamic programming is used in [11], whereby constrained motion planning is solved progressively by replacing an equality constraint with inequality constraints of tighter and tighter tolerance. Essentially this implies that moveable objects are free to move without the aid of the grasping robots, and gradually the two are forced closer together until a physically realizable solution is obtained. Though limited by the need to manually specify "docking configurations" (stable and accessible states of moveable objects), Barraquand demonstrates manipulation problem solutions involving a non-serial manipulator, dual-arm manipulation, and multi-robot coordination.
- Finally, Khatib describes a practical real-time obstacle avoidance algorithm for manipulators in [12] based on time-varying artificial potential fields, demonstrating the technique using visual sensing on a PUMA 560 robot. Khatib defines the equations of motion in "operational space", which decouples end effector motion from link motions by treating it as a point mass.

For a more comprehensive review of robotic motion planning, see [13] for a survey of algorithmic approaches and their time complexity, [14] for a thorough overview of planning research conducted prior to the 1990's, and [7] for a succinct summary and comparison of the major classical and heuristic motion planning methods developed to date.

Though research in manipulator control is extensive, several aspects of the given control problem for the SACL robot appear to be relatively unexplored. The integration of a real-time motion planning algorithm and sensing in a hardware application is not tremendously common (though thoroughly studied theoretically), as the challenge of real-time planning by itself is still primarily unsolved. Several works have theoretically explored real-time robotic motion planning using visual sensing [12], [15], range sensing [16], and tactile sensing [17], but particularly few if any seem to have studied temperature-based sensing. Simultaneous Localization and Mapping (SLAM) algorithms, particularly those restricted to tactile or haptic mapping [18], [19], appear to be the most relevant, addressing the closely related issue of developing a motion plan that will converge to a map of the local environment. There are interesting motion planning control problems that address temperature constraints, but the majority focus on *internal* constraints due to joint actuator overheating in order to limit the wear on robotic parts. For example, Guilbert [20] solves a thermally constrained minimum-time manipulator optimal control problem by approximating its overall Joule heating, friction, and conduction heating with an empirical thermal model of the actuators. Pledel and Bestaoui [21] limit actuator temperatures by constraining the maximum DC motor currents and voltages, the motor current time derivatives, and the total RMS current over the trajectory. Though it does not consider a time-varying environment of thermal obstacles, a closely-related work is given in [22] in which the difficulties of proximity sensor-based motion planning are enumerated and discussed, based on a system of 16 infrared proximity sensors built into the skin of a manipulator arm. It is from these related works that we seek to leverage a practical implementation for the SACL robot.

B. *Motivation of Proposed Work*

Though manipulator motion planning is one of the oldest applications of planning algorithms, the advent of new ceramic and composite materials have yielded a host of new and challenging manipulation tasks that have not yet been fully studied. Constraints defined by a sensor network distributed over the robot are relatively unique and not typically seen in most robotic motion planning problems. Interestingly, for problems of this type, a configuration is invalidated not by the dynamics of the robot but purely by the nature of the environment experienced at that state. Furthermore, the thermal environment perceived by the robot changes over time as a direct result of the convective and radiative heat transfer from the temperature disturbance. Time-varying environments such as these are not commonly investigated in motion planning, particularly those imposed by fluid dynamics.

Furthermore, and intriguingly, the decision-making task required of the robot in this experiment emulates the same very natural decisions a human would make if he or she were placed in the unfortunate situation of needing to move his or her arm through an environment with unknown temperature obstacles. From an intellectual standpoint, this invites an exciting comparison between the results of our proposed algorithm and natural, intuitive arm movement.

In addition, the proposed problem requires the development of an effective, computationally inexpensive algorithmic solution to simultaneous motion planning and environmental estimation. This establishes a challenging test environment for the techniques employed in this paper, which incorporate ideas from multiple works in the literature. The main contribution of this work is in the rigorous validation of these algorithms in solving a complex task on real hardware, through which it is hoped that useful information can be gleaned regarding their practicality and effectiveness.

C. *Statement of Work*

The objectives of this paper are to describe the algorithm implemented on the SACL robotic arm and demonstrate its effectiveness in a temperature-constrained environment. The ability of the robotic arm to sense a hot obstacle and negotiate a new plan to the goal is illustrated during a simple experiment. Numerical results from experimental trials are provided, and comments are given regarding difficulties of implementation, lessons learned from testing, and strengths and weaknesses of the solution.

The paper is organized as follows. Section II gives a brief introduction to the motion planning problem. Section III then outlines the solution proposed to solving the specific motion planning problem discussed previously, followed by representative numerical results from simulation and experimental trials in Section IV. Finally, Section V closes with a brief summary of conclusions and future work.

II. Problem Formulation

The objective of the SACL manipulator arm experiment is to safely transfer a single potentially hot target object from its initial (stable) configuration to every one of a sequence of desired goal configurations, amidst external temperature obstacles. The manipulator is equipped with a dense network of piezoelectric pressure sensors, temperature sensors, and strain gauges that will determine online in near-real time the temperature and pressure distribution over its third link (see again Figure 1). A "safe" plan in this case is defined as a trajectory over which the manipulator does not collide with itself or the environment, and during which neither the object nor the manipulator arm sensor network experiences temperatures exceeding an imposed upper bound. In the event that temperature obstacles are encountered and "collision" with upcoming hazards becomes likely, the control system must develop a new plan to the goal, circumnavigating the disturbance(s) via an alternative route and updating the trajectory in real time.

This problem is posed as a specific application of a motion planning problem. In general, the motion planning problem can be stated formally as follows. Let $C \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ be the configuration space and the input space, respectively, and consider the autonomous dynamical system $\dot{x}(t) = f(x(t), u(t))$, $x(0) = x_I$, where $x(t) \in C$, $u(t) \in U$, and $f(\cdot,\cdot)$ models the system equations of motion. Let $C_{obs} \subset C$ represent the obstacle region, $C_{goal} \subset C$ denote the goal region, and $C_{free} := C \setminus C_{obs}$ denote the obstacle-free space. Finally, let $c(x(t),u(t)) \to \mathbb{R}_{\geq 0}$ be a cost function that maps state-control pairs into positive real numbers. Then, the traditional *robotic motion planning problem* is to find an action trajectory $u : [0,T] \to U$ yielding a feasible path $x(t) \in C_{free}$ over time horizon $t \in [0,T]$, which reaches the goal region $x(T) \in C_{goal}$ and minimizes the cost functional $J = \int_0^T c(x(t),u(t))\,dt$.

It can be shown that robotic motion planning is at least as difficult as the generalized mover's problem, which has been proven to be PSPACE-hard, implying NP-hard [23], [24]. Neglecting the dynamics, $f(\cdot,\cdot)$, and treating a manipulator as a kinematic chain of rigid bodies, manipulator planning is then simply a special, simplified case of the robotic motion planning problem. An introduction to the manipulator planning problem is given in [11], which shows that in general manipulation constraints are nonholonomic, as they depend on the first derivative along the path of the grasped object. For a more thorough description of the nature of the manipulation problem, including the definitions of part and robot configuration spaces, admissible, stable and grasped configurations, and manipulation graphs, please refer to [24].

With an immobile base, four revolute joints, the end effector, and the target object, the control and planning problem for the SACL robot consists of a minimum total of 11 degrees of freedom (DOF) in its full generality. This yields an 11-dimensional planning space consisting of the Cartesian product of the manipulator joint space $q \in \mathbb{R}^4$, gripper separation space $u \in \mathbb{R}_{\geq 0}$, the Euclidean space of target object positions $x_{obj} \in \mathbb{R}^3$, and the space of target object velocities $v_{obj} \in \mathbb{R}^3$, resulting in:

$$C = \mathbb{R}^4 \times \mathbb{R}_{\geq 0} \times \mathbb{R}^3 \times \mathbb{R}^3$$

In order to simplify the exposition and reduce the computational complexity of the problem, we make the following assumptions:

1) *Transition configurations* that transition the manipulator from transit mode (without the object) to transfer mode (with the object) [24, p. 334] are assumed to be known *a priori*
2) All links, the end effector, and the target object act as rigid bodies during the motion plan.
3) Reference joint angles are sufficient for manipulator control. That is, it is assumed that trajectory-following of the reference commands is accomplished automatically by the internal servo controllers, and hence additional feedback on the joint angles is unnecessary.
4) Joint angles can vary continuously from a given minimum to a given maximum without resulting in interlink collision. These bounds thus fully encode the restricted configurations that would result in the robot colliding with itself.
5) Link geometry can be conservatively approximated by Oriented Bounding Boxes (OBB's) [25]
6) The initial and goal target object positions are *reachable* and *stable*.

(1) and (2) imply that the motion plan is developed without the need to compute a transition plan, assuming one is known beforehand and given to the manipulator. This effectively allows the decoupling of the manipulation task from the motion plan, separating the manipulator joint space from the object phase space and enabling the motion plan between manipulator waypoints to be developed independently from the grasping task (implemented open-loop). By conducting the grasp separately, this reduces the dimensionality of the planning space to the dimensionality of the joint space, improving computation time. (2) also makes the good assumption that deformations of the robot and object can be neglected, ignoring their internal degrees of freedom. (3) suppresses the requirement for low-level control, eliminating the need for specification of classical control techniques that ensure the actual joint angle trajectory follows the commanded trajectory. (4) and (5) partially alleviate the burden on collision checking, assuming that joint angle bounds automatically account for intra-robot collisions and that robot-environment collisions can simply be determined for a given joint configuration by evaluating the feasibility of selected points on each link's OBB. Finally, (6) allows the planner to ignore the possible need of depositing the object in intermediate stable configurations (called "docked" positions) before transferring to the goal, as is required in much more difficult motion planning problems (e.g. [11]).

With this problem formulation in mind, we are now in better position to develop a real-time manipulator motion planner subject to temperature constraints. One potential solution, and the solution implemented on the SACL robot, is given in the following section.

III. Proposed Solution

In order to accomplish real-time motion planning of a manipulator arm subject to temperature constraints, we propose an incremental sampling-based motion planning approach. Though many alternative control strategies exist, e.g. optimal control, model predictive control, etc, motion planning offers several key advantages that are particularly useful for real-time autonomous control in this case:

- *Control in time-varying, cluttered environments.* Motion planners can be adapted to accommodate time-varying constraints or obstacles by simply appending the time t to the state vector. The direct incorporation of a collision-checking algorithm (either indirectly through representations of free space or externally) for most motion planners gives them a direct way to satisfy collision-avoidance constraints that are difficult to meet using other control techniques.
- *Ability to incorporate logic.* With the notable exception of adaptive control, most classical control assumes a fixed system that is unable to change modes, make system changes, or conduct upper-level decision making. The algorithmic framework of motion planning gives it the flexibility to apply logical decisions and respond to high-level specification of tasks. This greatly simplifies the decoupling of the manipulation and planning tasks for the SACL robot.

- *"Anytime" capability.* The incremental nature of some motion planning algorithms, primarily sampling-based methods, allow intermediate solutions to be reported as soon as they are found (if they exist) [26]. Time-permitting, subsequent iterations can be used to improve the initial trajectory and converge to the global optimal solution. This is important when working in real-time within environments hazardous to the robot.
- *Re-planning efficiency.* Particularly for roadmap and dense tree algorithms, previously-computed sets of feasible solutions can be updated or augmented periodically to reflect the changing environment (by updating roadmap edges, tree nodes, etc), allowing new paths to be determined efficiently in a so-called "lazy-search".

Current motion planning techniques rely on either exact combinatorial solutions or approximate algorithms. Though only exact algorithms are guaranteed to provide a correct solution in finite time (called *completeness*), approximate techniques, particularly randomized algorithms, have been shown to be more efficient in practice with simpler implementations as well as reduced expected running times. Furthermore, many exact algorithms rely on static time-invariant environments (e.g. cylindrical decomposition or Canny's roadmap algorithms [24]), which cannot be assumed for the scenario posed in Section II. Though the robot in this case is not subject to stringent dynamic constraints (meaning it is effectively able to freely move to any joint configuration within actuator limits) and despite the low problem dimensionality, the inability of an exact algorithm to accommodate the introduction of time-varying obstacles is too restrictive. Therefore, we attempt a motion planning solution that does not require explicit obstacle representation and accept the weaker notion of probabilistic or resolution completeness versus exact completeness.

Of the available approximate motion planning techniques, incremental sampling-based motion planning is chosen for this application due to, as noted previously, its anytime capability and re-planning efficiency through lazy-search. This approach discretizes the C-space into points called *samples*, converting the motion planning problem into a geometric one. This enables quick and uniform exploration of the available states around the robot. Sampling-based motion planning has been applied successfully to manipulators for several decades, including [8], [9], [27]. In recent years, several extensions have been made to more complex scenarios, including planning with sensing uncertainty [28], dynamic constraints [29], parallelization [30], and cost minimization [31], [26]. Mazzini [18] describes a particularly successful application of sampling-based MP for tactile-SLAM manipulators in oil well junctions, used to characterize a junction's topology in as few maneuvers as possible. The algorithm currently employed in this paper is based on Rapidly-Exploring Random Trees with re-wiring (RRT*) [32], [24], employing a simple approach adapted from Kuffner [33] that is particularly well-suited for robotic systems. According to LaValle [34] and Karaman et. al. [26], the RRT* algorithm provides a flexible, generic framework for kinodynamic planning that extends well to high-dimensional spaces, can probabilistically guarantee completeness, and, most importantly (as its name suggests), enable rapid exploration of the configuration space. See Ref. [32] for a more general description of RRT algorithms.

For the manipulation problem at hand, planning will be conducted in two stages. First, a global planner is used to identify a near-optimal open-loop solution from q_I to q_G while avoiding a set of "static" obstacles. In the initial stage, the world frame is taken to be absent of any temperature disturbances, therefore devoid of moving obstacles. As it is computationally inefficient to use inverse kinematics to precisely map obstacles from the world frame to the manipulator C-space, they are represented instead as sets of semi-algebraic primitives in the world frame and solution feasibility is tested by an a posteriori check. If at any time during motion execution along this path a temperature obstacle is encountered, then a second local planner is called to re-route the manipulator to the nearest available feasible path with as little compromise to the original cost function value as possible.

For the global-planning task, a modified form of the RRT-Connect algorithm is used (to be described below). Once a joint angle command history has been determined, closed-loop motion begins. Here, the initial near-optimal path is followed for as long as possible while repeatedly appealing to the sensing skin for temperature information. Should an imminent violation be detected by the introduction of a temperature obstacle, a local planner is called to navigate the system to new paths. In the case that local replanning is necessary, it is here that the use of an RRT-based global planner becomes particularly advantageous – the tree-like exploration of the ("statically") feasible space of C allows for efficient determination of new paths through heirarchical collision detection of the trees with new obstacles and expansion of the current state to the subset of pre-computed feasible paths that still remain feasible. If no pre-computed paths are feasible, the current state is set as a new node in the tree and attempts are made to connect it directly to the goal through the continued sampling and addition of safe nodes to the tree.

The RRT*-Connect algorithm, represented in pseudocode as Algorithm 1 and based on the planner proposed by Kuffner [33], is a simple variant of the RRT algorithm that uses bi-directional tree growth from initial and goal states. This limits the algorithm to single-query path planning, though by the same token also makes it particularly efficient at determining a path between any two points. Given that the SACL manipulator is required to maneuver in sequence through a series of fixed (and collision-free) waypoints, this property is directly sought, as motion between m waypoints can simply be partitioned into $m-1$ single-query motion plans. The primary differences between the framework employed here and that used by [33] are in the augmented tree data structures $\mathscr{T}_a$ and $\mathscr{T}_b$, improved for the purposes of rapid re-planning, and in the use of re-wiring (hence the distinguishing asterisk in the name of the algorithm). (Additionally, Algorithm 1 has been modified to avoid the incorporation of all intermediate CONNECT() nodes in order to temper the number of nodes added to each tree.)

Algorithm 1 Structure of the RRT*-Connect Algorithm

```
1: function RRT*–CONNECT($q_I$, $q_G$, $Q$)
2:     ADDNODE($q_I$, $\mathscr{T}_a$, $\varnothing$, 0)        ▷ Fwd. Tree root node
3:     ADDNODE($q_G$, $\mathscr{T}_b$, $\varnothing$, 0)        ▷ Rev. Tree root node
4:     $\mathscr{T}_a$.connections[$q_I$] = $\varnothing$
5:     $\mathscr{T}_b$.connections[$q_G$] = $\varnothing$
6:     iter = 1, $J^* = +\infty$
7:
8:     while iter < iter$_{\max}$
9:         $q_r = Q$[iter]
10:        [$q_{near,a}$, $q_{NN}$, $J_{NN}$] = NEAREST($\mathscr{T}_a$, $q_r$)
11:        [$q_{new,a}$, $J$, status] = EXTEND($q_{near,a}$, $q_r$, $\varepsilon$)
12:        if status != Trapped
13:            ADDNODE($q_{new,a}$, $\mathscr{T}_a$, $q_{near,a}$, $J$)
14:            REWIRE($\mathscr{T}_a$, $q_{new,a}$, $q_{NN}$, $J_{NN}$)
15:            $\mathscr{T}_a$.connections[$q_{new,a}$] = $\varnothing$
16:
17:            [$q_{near,b}$, $q_{NN}$, $J_{NN}$] = NEAREST($\mathscr{T}_b$, $q_{new,a}$)
18:            [$q_{new,b}$, $J$, status] = CONNECT($q_{near,b}$,$q_{new,a}$,$\varepsilon$)
19:            if status == Reached
20:                ADDNODE($q_{new,b}$, $\mathscr{T}_b$, $q_{near,b}$, $J$)
21:                REWIRE($\mathscr{T}_b$, $q_{new,b}$, $q_{NN}$, $J_{NN}$)
22:                $\mathscr{T}_b$.connections[$q_{new,b}$] = $q_{new,a}$
23:                $\mathscr{T}_a$.connections[$q_{new,a}$] = $q_{new,b}$
24:                feasible = 1        ▷ Feasible path found!
25:
26:        Swap $\mathscr{T}_a$ and $\mathscr{T}_b$
27:        iter = iter +1
28:
29:    if iter is even
30:        Swap $\mathscr{T}_a$ and $\mathscr{T}_b$
31:
32:    for each tree $\mathscr{T}_j$, $j \in a,b$
33:        for each $q_i \in \mathscr{T}_j$
34:            if $\mathscr{T}_j$.connections[$q_i$] != $\varnothing$
35:                ADDNODETOLEAFLISTS($\mathscr{T}_j$, $q_i$)
36:
37:    for each $q_i \in \mathscr{T}_a$
38:        $q_c$ = $\mathscr{T}_a$.connections[$q_i$]
39:        if $q_c$ != $\varnothing$ && $\mathscr{T}_a$.costs[$q_i$] + $\mathscr{T}_b$.costs[$q_c$] < $J^*$
40:            $q_a^* = q_i$
41:            $q_b^* = \mathscr{T}_b$.nodes[$q_c$]
42:            $J^*$ = $\mathscr{T}_a$.costs[$q_i$] + $\mathscr{T}_b$.costs[$q_c$]
```

The algorithm works as follows. Let Q denote the sequence of sample points, and $\mathscr{T}$ be a tree of feasible nodes and interconnecting trajectories called *branches*. The sample set Q is used to generate a deterministically random sequence of samples that direct the growth of each the forward tree $\mathscr{T}_a$ stemming from q_I and the reverse tree $\mathscr{T}_b$ stemming from q_G. The simulations discussed in Section IV employ the Halton

Algorithm 2 Adds node q to all ancestor leaf lists

```
1: function ADDNODETOLEAFLISTS($\mathscr{T}$, $q$)
2:     $q_{parent}$ = $\mathscr{T}$.parents[$q$]
3:     while $q_{parent}$ != $\varnothing$        ▷ Exits at root node
4:         ADDLISTELEMENT($q$,$\mathscr{T}$.leaf_lists[$q_{parent}$])
5:         $q_{parent}$ = $\mathscr{T}$.parents[$q_{parent}$]
```

Algorithm 3 Adds a node q to tree $\mathscr{T}$

```
1: function ADDNODE($q$, $\mathscr{T}$, $q_{parent}$, $J$)
2:     $|\mathscr{T}|$ = $|\mathscr{T}|$ + 1
3:     $\mathscr{T}$.nodes[$|\mathscr{T}|$] = $q$
4:     $\mathscr{T}$.parents[$q$] = $q_{parent}$
5:     $\mathscr{T}$.costs[$q$] = $\mathscr{T}$.costs[$q_{parent}$] + $J$
6:     $\mathscr{T}$.connections[$q$] = $\varnothing$
7:     $\mathscr{T}$.leaf_lists[$q$] = $\varnothing$
8:     $\mathscr{T}$.safety[$q$] = 1
```

Algorithm 4 Rewires suboptimal branches near q_{new}

```
1: function REWIRE($\mathscr{T}$, $q_{new}$, $q_{NN}$, $J_{NN}$)
2:     for e ach neighbor in $q_{NN}$
3:         if $\mathscr{T}$.costs[$q_{new}$] +$J_{NN}$ < $\mathscr{T}$.costs[$q_{NN}$]
4:             if Connect$q_{new}$, $q_{NN}$, $\varepsilon$ $\mathscr{T}$.parents[$q_{NN}$] = $q_{new}$ $\mathscr{T}$.costs[$q_{NN}$] = $\mathscr{T}$.costs[$q_{new}$] +$J_{NN}$
```

Algorithm 5 Grows the tree by a distance $\leq \varepsilon$ from q_{near} towards q

```
1: function EXTEND($q_{near}$, $q$, $\varepsilon$)
2:     [$q_{new}$, $J$] = STEER from $q_{near}$ towards $q$
3:     if CONSTRAINTVIOLATION($q_{new}$) == 0
4:         if $||q - q_{new}|| \leq \varepsilon$
5:             status = Reached
6:         else status = Advanced
7:     else status = Trapped
8:     return $q_{new}$, $J$, status
```

Algorithm 6 Grows the tree all the way from q_{near} to q, if possible

```
1: function CONNECT($q_{near}$, $q$, $\varepsilon$)
2:     status = Advanced, $J$ = 0
3:     while status == Advanced
4:         [$q_{new}$, cost, status] = EXTEND($q_{near}$, $q$, $\varepsilon$)
5:         $q_{near}$ = $q_{new}$
6:         $J$ = $J$ + cost
7:     return $q_{new}$, $c$, status
```

sequence, as described in Ref. [24]. This ensures a minimum-possible dispersion, while encouraging tree expansion to unexplored areas of the configuration space after just a few iterations. Then the NEAREST algorithm is called to identify points in the tree closest to the current sample point, typically through the use of an additional KD-tree data structure, which, once built with k nodes in $O(nk \lg k)$ time, can determine nearest-neighbors among a set of nodes n in $O(\lg k)$ time [24]. Once NEAREST returns the nearest neighbor q_{near} and other neighbors q_{NN} with their respective costs-to-go J_{NN} to q_r (sorted by optimality), EXPAND is used to call a *navigation function* STEER to determine a control law that guides the robot from each neighbor towards the sample point. In our case, as is typically chosen for kinematic systems, STEER simply directs the robot along straight-line paths between the nodes. If the expansion was done safely according the external CONSTRAINTVIOLATION algorithm, whereby *Reached* or *Advanced* is returned, then the newly-generated node is added to the tree and re-wiring is conducted. Subsequently, an attempt is made from the other tree to connect to the same node directly, where an advancement is made repeatedly until convergence or failure. If successful, its node is also added to its own tree, followed by a critical process called *re-wiring*.

Rewiring is a strategy used to update old branches based on the inclusion of new nodes to the tree, which may offer less costly paths to its original nodes. Put simply, if the cost to a new milestone plus the cost to an old node is less than the original cost to the old node, then we reconfigure the tree by deleting the branch between the old node and its parent, replace the parent with the new milestone, and store the branch between them. It can be shown that by allowing these updates at each iteration, as the number of iterations approaches infinity, the tree will converge to the globally optimal solution [26].

Finally, the trees are swapped in order to alternate between expansion and connection, and the entire process is repeated. Any successful connections are identified for each added node by saving the index of its corresponding leaf node to the augmented field entitled “connections”. At the very end, once the trees are complete, an additional routine is implemented that adds the indices of every connected leaf node to the leaf lists of each of its ancestor nodes using a call to the ADDNODETOLEAFLISTS function. The purpose of this augmented tree data structure and these additional functions is to expedite the replanning-process during closed-loop control, as explained in the following section (Section IV). A full pictorial representation of node memory is given in Figure 3.

Figure 4 illustrates this tree construction process. Note that, as described, the approach is heavily biased towards interconnection of the trees as opposed to exploration of the C-space, though this can be easily adjusted through the use of CONNECT() in place of EXPAND() on line 11. In this case, the trees would each attempt to expand entirely to the sample nodes q_r. This strategy more aggressively expands the trees but may result in fewer added nodes in the event that C is densely cluttered. It was determined through the simulation-based experiments described in the subsequent section that this was not a limiting factor due to the sparse population of obstacles in the manipulator space. Therefore, the current implementation of the algorithm uses the framework as shown in Algorithm 1 and the figure, but modified to use CONNECT in place of EXPAND.

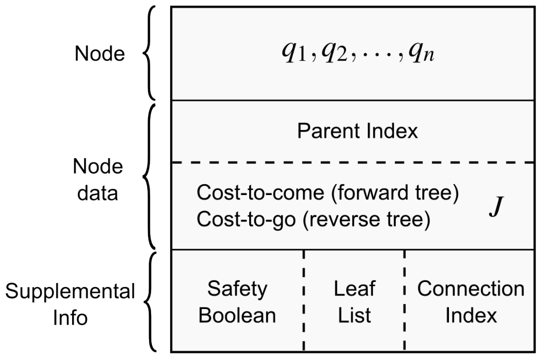


Fig. 3: Diagrammatic layout of augmented RRT node memory. In addition to tree node coordinates, parents, and costs, a singly-connected linked list of the leaf nodes in reverse-sorted order is maintained, a flag identifying the safety of each node, and the index of its companion tree paired node (if such a node exists) are all included. This enables efficient heirarchical collision detection and tree pruning (using “safety”), total path length minimization (using the “connection” index), and fast re-planning (using leaf nodes).

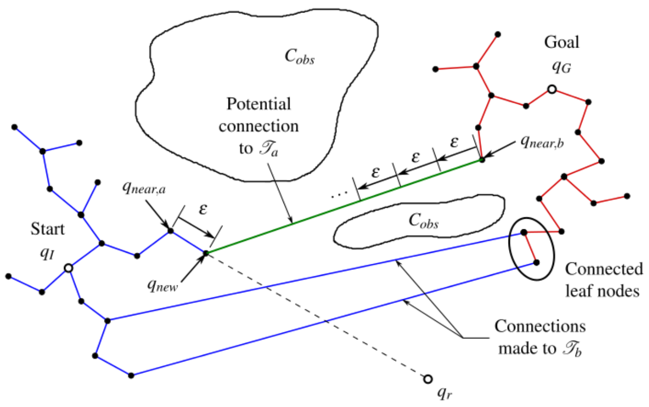


Fig. 4: Construction of forward $\mathscr{T}_a$ and reverse $\mathscr{T}_b$ trees using RRT-Connect. In this case, $\mathscr{T}_a$ is selected for expansion towards the random sample q_r and adds q_{new} to its set of nodes if $q_{new} \in C_{free} = C \setminus C_{obs}$. $\mathscr{T}_b$ subsequently attempts to connect to the newly added node through repeated incremental growth and collision-checking towards q_{new}. The behavior of the two trees is alternated between iterations.

IV. Simulation/Experiments

The following section numerically demonstrates the use of the Rapidly-exploring Random Tree Connect (RRT*-Connect) algorithm (shown as Algorithm 1) for manipulator motion planning in kinematically-constrained environments. First, a number of important implementation details for closed-loop control based on the RRT*-Connect algorithm are enumerated, including the specific method taken for adapting the motion planner for feedback. This is followed by illustrative results of the manipulator as it seeks to accomplish the experimental objectives described in Section II while avoiding a single, simulated temperature obstacle.

A. Temperature Obstacle Detection

The foremost critical process in the successful adaptation of the motion planning algorithm to closed-loop control is the temperature obstacle detection algorithm. The current strategy for detection, generation, and avoidance of temperature obstacles is heavily based on simplicity, relying primarily on – and therefore heavily restricted by – two key assumptions: (1) that heat sources remain fixed after they have been introduced, and (2) that heat sources act locally and do not change their environment significantly outside of a finite axially-directed region. This is meant to model such sources as blowtorches, convecting gas vents or tubes, or single low-temperature open flames, such as lit matches or candles.

As a preliminary experimental approach to handling these temperature constraints, a heuristic geometric scheme is employed. Currently, an estimation code is called periodically to determine, using a clustering algorithm, the region of maximum temperature on the surface of the sensing skin. Should the temperature within this region exceed a maximum allowable threshold, a flag is sent to the motion planner and the approximate radius, the skin surface normal vector, and the centroid's body-fixed coordinates of this violating region are each relayed as input. Once determined and a violation has been detected, the disturbance is modelled by a truncated cone (frustrum) with tunable half-angle β axis-aligned with the sensing-skin region centroid normal vector and cut off at heights h_1 and h_2 above the cone apex. The apex location is determined in such a way as to achieve a radius equal to the violating-sensing region radius. This cone, emanating from the sensing skin out towards (and hopefully enveloping) the heat source, is then assumed to remain fixed in the world frame, and is used to re-evaluate tree node safety and develop new plans around the temperature hazard. Put simply, once the skin has sensed an unsafe region, the arm assumes it lies within the general direction in which it experienced the greatest temperature threshold violation. The critical assumptions involved in this approach are that the trees of the current motion plan remain valid, that the goal point q_G is still safe, and that the heat source does not move once introduced.

This approach is intended to emulate the behavior of a human in reaction to a given temperature distribution across the surface of his or her own skin in an unknown environment, though this is purely conjecture and has no basis other than the personal experience of the author. The idea is somewhat related to the heuristic mapping technique used successfully by Mazzini in [18], which he calls "best-cone" search. However, it should be noted that learning-based schemes, artificial intelligence techniques, or otherwise tactile-SLAM methods are likely more suitable. It is with keen interest that we study these other techniques in future research.

Note that with a changing obstacle environment, nodes that were previously added to the tree and identified as safe can suddenly become invalid. To expedite the replanning process after a new obstacle has been introduced, an augmented form of the RRT data structure has been implemented (indicated in Figure 3), which allows nodes to be labelled as safe or unsafe as necessary without the need for deletion or re-insertion into the tree, thus saving valuable computational time and effort.

B. RRT Implementation

The following subsections describe how the RRT*-Connect algorithm is implemented for our particular manipulator motion planning problem.

a) Steering Law: Beginning from state q_I, the robot's task is to explore the local configuration space and determine a safe route towards a single goal state q_G. In order to develop a motion plan, the algorithm grows trees of trajectories from nodes currently residing in the tree, $q_i \in \mathscr{T}$ to sample states, $q \in Q$, via a steering law. As is typical throughout the literature, the steering law for this case directs the manipulator along the straight line segments between q_i to $q \in Q$ as discussed in Section III.

b) Sampling: Also as described in Section III, the robot configuration space ($n = 4$) is sampled according to the Halton sequence, which deterministically selects well-distributed points in an n-D hypercube, $[0,1]^n$. Given the set of samples H from $[0,1]^n$, the sample set Q can be formed by scaling H to span the manipulator joint space:

$$Q = H\left(q_{max} - q_{min}\right) + q_{min}$$

where $q_{min,i} \leq q_i \leq q_{max,i}, i = 1,\ldots,n$ is the range of motion of each arm link that will guarantee no inter-link collisions for every possible configuration of the robot. The resulting distribution can be seen in Figure 5.

c) Nearest Neighbor: A key component of the RRT*-Connect algorithm is the nearest neighbor selection algorithm, which selects from a source node the nodes in the tree that are "closest" to the source. Here we take this measure of closeness to be the weighted 2-norm Euclidean distance metric over the manipulator *joint* space, i.e.

$$\rho(q_1, q_2) = \left(\sum_{i=1}^{n} \left|w_i\left(q_{1,i} - q_{2,i}\right)\right|^2\right)^{\frac{1}{2}}$$

As required, this satisfies the axioms of a topological metric space. The benefit of defining closeness in joint space as opposed to the world space is that transformations back to the world frame are no longer necessary, which is fortunate as the need to do so for every iteration of RRT*-Connect between every candidate pair of neighbor nodes would be prohibitively expensive. The intent of minimizing the distance in joint space is to promote smooth motions of the

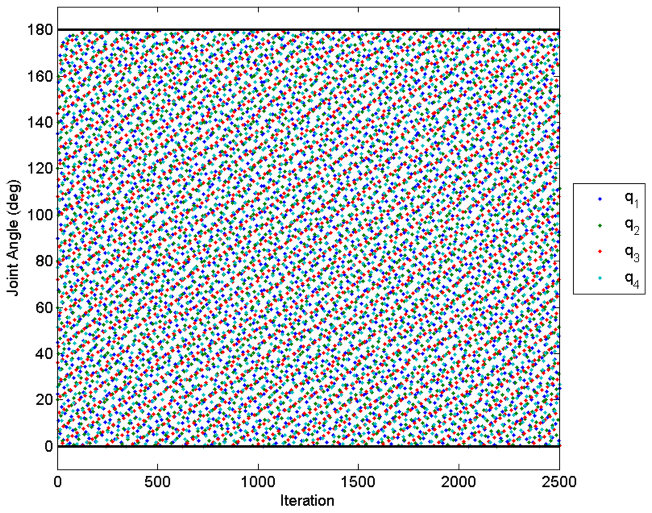


Fig. 5: Joint angle sample set produced by the Halton sequence after 2500 iterations

	L [in]	W [in]	H[in]	d_i [in]	a_i [in]	α_{i-1} [deg]
Link 1	3.25	2.4	1.4	0	0	0
Link 2	6.70	1.0	3.3	0	0	-90
Link 3	10.5	1.75	2.25	0	6.75	180
Link 4	2.2	4.0	2.75	0	8.25	180

TABLE I: Stanford manipulator arm geometry

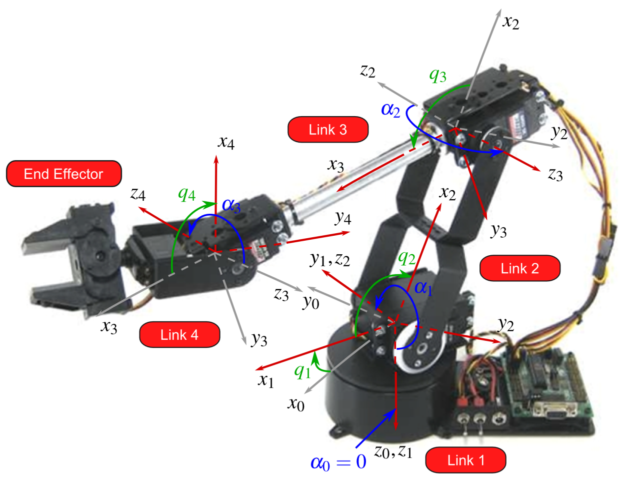


Fig. 6: Body-fixed frames of the manipulator arm and their associated Denavit-Hartenberg (DH) parameters

arm, in an attempt to encourage arm movements to be as natural as possible. The NEAREST routine of the RRT*-Connect algorithm uses this metric function to determine the priority of nodes closest to any given node (for re-wiring) or sample point (for tree growth).

d) Constraint Checking: In order to determine whether a particular configuration, q, of the robot is in violation of the obstacles in its environment, it is necessary to be able to transform efficiently from one coordinate basis to another. A typical approach for kinematically-constrained chains of rigid bodies, as is the case for a robotic manipulator, is the use of homogeneous transformation matrices, denoted by T [24]. Suppose vector $x \in \mathscr{R}^3$ resolved in frame F_2 are obtained from their components in frame F_1 by applying rotation matrix R and then translating them by vector v from origin 1 to origin 2. The coordinates of x can be related by the matrix equation:

$$\begin{pmatrix} x|_{F_2} \\ 1 \end{pmatrix} = \underbrace{\left[\begin{array}{c|c} R & v \\ \hline 0 & 1 \end{array}\right]}_{T} \begin{pmatrix} x|_{F_1} \\ 1 \end{pmatrix}$$

The inverse of this relation is:

$$\begin{aligned} \begin{pmatrix} x|_{F_1} \\ 1 \end{pmatrix} &= \left[\begin{array}{c|c} R^{-1} & -R^{-1}v \\ \hline 0 & 1 \end{array}\right] \begin{pmatrix} x|_{F_2} \\ 1 \end{pmatrix} \\ &= \underbrace{\left[\begin{array}{c|c} R^T & -R^T v \\ \hline 0 & 1 \end{array}\right]}_{T^{-1}} \begin{pmatrix} x|_{F_2} \\ 1 \end{pmatrix} \end{aligned}$$

A simple, structured way to define the rotation matrices and translation vectors that relate manipulator link body-fixed frames to one another (and ultimately to the world frame) is to use Denavit-Hartenberg (DH) parameters [24]. By representing the transformation from link frame i to frame $i-1$ as two screw operations defined by,

1) translation by d_i along the z_i axis
2) counter-clockwise rotation by $\theta_i = q_i$ about z_i
3) translation by a_{i-1} along the x_{i-1} axis
4) counter-clockwise rotation by α_{i-1} about x_{i-1}

any coordinate in body-fixed frame $i-1$ can be found from frame i as:

$$\begin{pmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} \cos q_i & -sin(q_i) & 0 & a_{i-1} \\ \sin q_i \cos \alpha_{i-1} & \cos q_i \cos \alpha_{i-1} & -\sin \alpha_{i-1} & -\sin \alpha_{i-1} d_i \\ \sin q_i \sin \alpha_{i-1} & \cos q_i \sin \alpha_{i-1} & \cos \alpha_{i-1} & \cos \alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{T_i} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

With this framework, any point in the body-fixed frame of link k can be resolved directly into the world frame using

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = T_1 T_2 T_3 \ldots T_k \begin{pmatrix} x_k \\ y_k \\ z_k \\ 1 \end{pmatrix}$$

or, conversely,

$$\begin{pmatrix} x_k \\ y_k \\ z_k \\ 1 \end{pmatrix} = T_k^{-1} \ldots T_3^{-1} T_2^{-1} T_1^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

The Denavit-Hartenberg parameters for the Stanford SACL manipulator arm are included in the geometric parameters listed in Table I and are illustrated in Figure 6.

With this representation of the manipulator, the problem now becomes how to best represent the geometry of the manipulator arm and its environment in a manner conservative enough to always guarantee collision detection yet

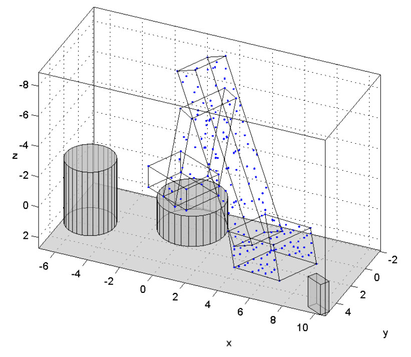


Fig. 7: Initial configuration of the manipulator arm and its perceived environment

relaxed enough to prevent occlusion of feasible paths. The representation must be simple and computationally efficient in order for the collision detection algorithm, often a bottleneck in computation, to be amenable to real-time planning. For this we employ a simple Oriented Bounding Box (OBB) architecture for the manipulator arm and represent the environment as a collection of obstacle primitives in the world space, $\mathscr{W}$, including planes, cylinders, cuboids, and the conical temperature obstacles. The complicated geometry of each link is instead replaced by an encasing box that simplifies the collision detection process. However, determining the intersection of a cuboidal link with even such simple objects as cones and cylinders is a non-trivial task. To reduce the computational challenge of exact intersection detection, we can approximate each box as a collection of its 8 corner points with a number of face points well-distributed across its planar boundaries. In the current implementation, due to its dispersion-minimization properties, the Halton sequence is used to generate well-placed coordinates on the unit square, which are then scaled to the dimensions of each face. The advantage of this description is that the complexity can be tailored to each link – for instance, the link on the rotating base, which is relatively immobile and unlikely to collide, can be represented by fewer facial points than the longer links further down the chain. This eliminates unnecessary computation where collisions are unlikely to occur, while giving more focus to higher risk areas. A diagram illustrating the world as seen by the manipulator from its starting configuration q_I, with a collection of these OBB face and corner points, is shown in Figure 7.

To be collision-free, all points on the manipulator must satisfy the following obstacle semi-algebraic relationships (derived from simple geometry):

- Planes (World frame):

  $$f(x,y,z) = a_p x + b_p y + c_p z + d_p \geq 0, \forall p = 1,\ldots,n_{planes}$$

  where $\hat{n} = (a,b,c)$ is the normal vector of each plane resolved in the world frame and d is the shortest distance to a point in the plane from the world origin
- Cuboids (World frame):

  $$f(x,y,z) = a_{c,i}x + b_{c,i}y + c_{c,i}z + d_{c,i} \geq 0$$
  $$\text{for at least one of } i = 1,\ldots,6, \forall c = 1,\ldots,n_{cuboids}$$

  (i.e. for intersection with a single cuboid, a manipulator point must lie below all 6 of its planar faces at once)
- Cylinders (Cylinder frame):

  $$f(x,y,z) = x^2 + y^2 - r_\ell^2 \geq 0$$
  $$|z| \geq H_\ell/2, \ \forall \ell = 1,\ldots,n_{cylinders}$$

  where r is the radius, H is the total height, and the cylinder frame is z-axis aligned with origin at the geometric center of the cylinder
- Temperature Obstacles (Obstacle frame):

  $$f(x,y,z) = x^2 + y^2 - (z\tan\beta_t)^2 \geq 0$$
  $$z \leq H_{t,1}$$
  $$z \geq H_{t,2}, \ \forall t = 1,\ldots,n_{tempobs}$$

  where β is the cone half-angle, H_1 is the frustrum floor, H_2 is the frustrum ceiling, and the obstacle frame is z-axis aligned and centered at the apex of this truncated cone.

Thus, testing of any manipulator configuration, $q = (q_1, q_2, \ldots, q_n)$, requires only the multiplication of transformation matrices $T_1(q_1), T_2(q_2)$, etc. with a manipulator body-fixed coordinate into the appropriate frame, followed by a subsequent check that the above obstacle primitive relations are satisfied or violated.

C. *Closed-loop Strategy*

With these sampling-based motion planning and temperature obstacle detection algorithms in place, the basic closed-loop motion planning and temperature obstacle avoidance strategy can be summarized as follows:

1) Pre-compute a motion plan by constructing trees of trajectories to sampled points, one growing forwards from initial state q_I and another growing backwards from goal state q_G
2) Once a plan is formed, commit to the near-optimal shortest path for a fixed time horizon of $t_{horizon}$ seconds
3) Unless violation of temperature constraints is detected, continue iterations of the RRT*-Connect algorithm to improve tree connectivity while committed to the motion plan
4) At the end of the fixed time horizon, re-evaluate the plan and update it to the new optimal path (if changed)

If at any time a new temperature disturbance is identified,

- Given an area of maximum temperature on the sensor skin, approximate the temperature obstacle as a truncated cone aligned with the skin surface normal
- Update the tree node safety properties to account for the new obstacle, search for feasible paths among nearest (safe) neighbors, store in sorted order the shortest paths that connect each tree to the other, and choose the

shortest-possible feasible path that does not intersect the temperature obstacle(s) as the new plan

- If no new path is found in sufficient time, stop the manipulator and conduct an exhaustive search until the first feasible plan is found. Continue motion if safe, or otherwise abort.

D. *Experimental Results*

The complete motion planner was demonstrated successfully during preliminary trials. When employed for closed-loop control in the trivial case of a static environment, the algorithm performs particularly well and often significantly improves the plan in real-time through repeated re-planning and tree growth. Provided a feasible plan can be developed initially, no difficulty was found in achieving its object relocation task. The real challenge arises when temperature obstacles are introduced. Thus far, a number of successful experiments have been conducted in which the manipulator was able to accomplish its mission following interruption by a simulated temperature disturbance, though as of this writing there has not been a physical demonstration involving real heat sources.

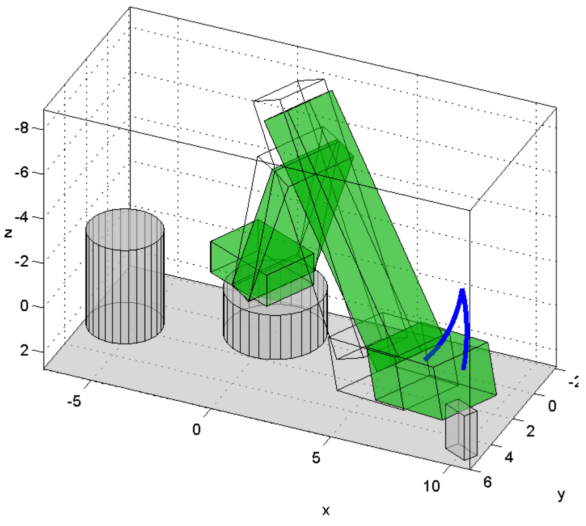
The performance of the manipulator is demonstrated in the open-loop disturbance-free case, Figure 8, and the closed-loop perturbed case, Figure 9, with a single virtual temperature obstacle introduced. These motions illustrate the typical path improvement and re-planning behavior of the motion planning algorithm. The joint angle motion plans corresponding to each scenario are given in Figures 10 and 11, respectively. Though not always successful, particularly when obstacles are introduced near the goal states, the arm appears to be reasonably robust to its temperature obstacle representations. Some difficulty still exists in the details of the upper-level control logic, such as when to reverse-course to previously-traversed safe configurations, how long to wait for temperature sensor acclimation, etc.; however, the planner itself appears to work well with the framework described in this paper. Further testing with real physical obstacles will be required before definitive conclusions can be formed.

V. Conclusions

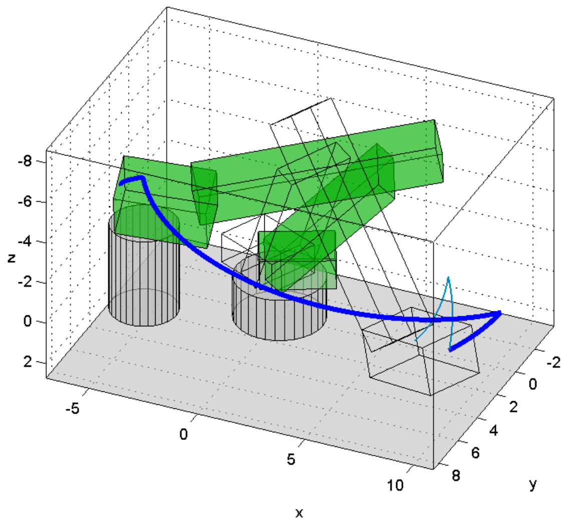
The motion planning strategy attempted in this paper has been demonstrated to work for a 4-DOF robotic manipulator arm. This reveals that sampling-based motion planning can be used to navigate a robotic system safely within a hazardous thermal environment. Though still in its formative stages, the heuristic approach taken thus far appears to be sufficient in practice. However, it must be noted that there are several clear opportunities for improvement, including:

- Improved Nearest-Neighbor search through the use of KD-tree data structures (currently using a brute force search)
- Extension to multiple-query planning, in order to improve robustness to temperature disturbances which may invalidate individual goal points
- Self-identification, exploration, and addition of new valid goal states in the event of goal region obstruction
- Incorporation of a temperature model that takes advantage of previous sensor readings, temperature spatial gradients, and time derivatives. Explicit thermal modelling should allow the planner to more precisely characterize and predict the nature of its environment.
- Inverse-kinematics for end-effector path planning in the world space, and for handling end-effector constraints such as keeping the grasped object level with the ground.

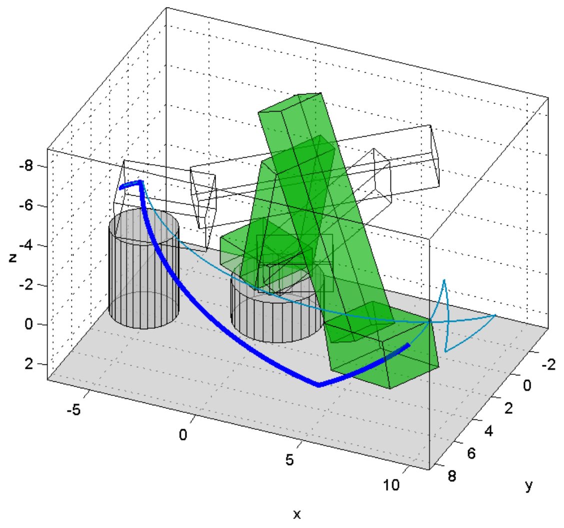
This yields significant room for further study. In the opinion of the author, though the current strategy works, a great deal can be done to increase plan robustness, improve robot safety, and provoke the arm to react and decide in a more intelligent manner. Real-time planning and control of dynamic systems while subject to the extreme time delays and uncertainties imposed by temperature sensing appears to be an unsolved and unexplored area of research, one in which we hope to pursue in coming work.



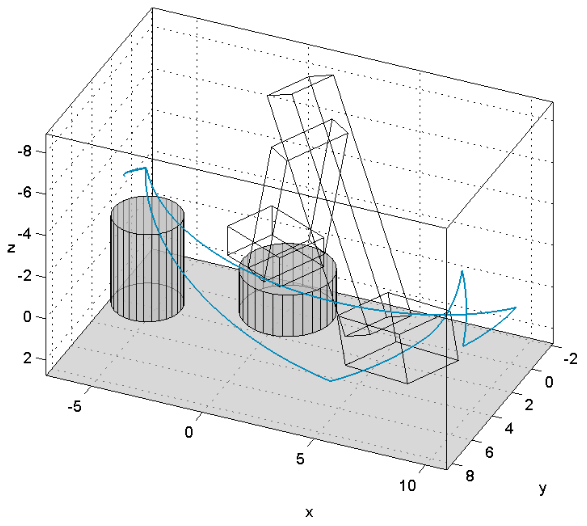
(a) Maneuver from initial position to battery



(b) Maneuver to deliver battery to cup

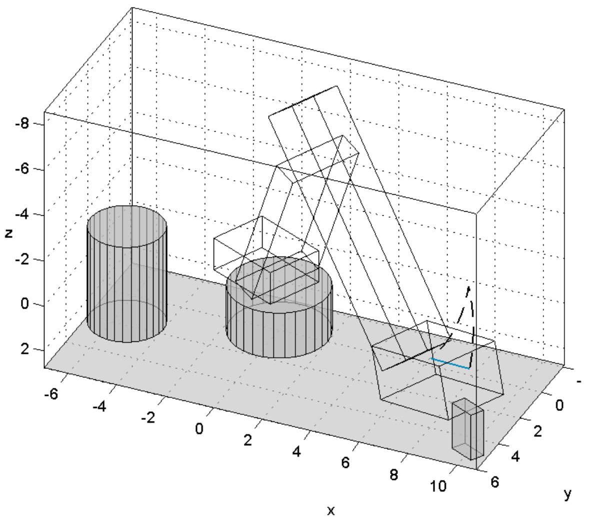


(c) Return maneuver to default position

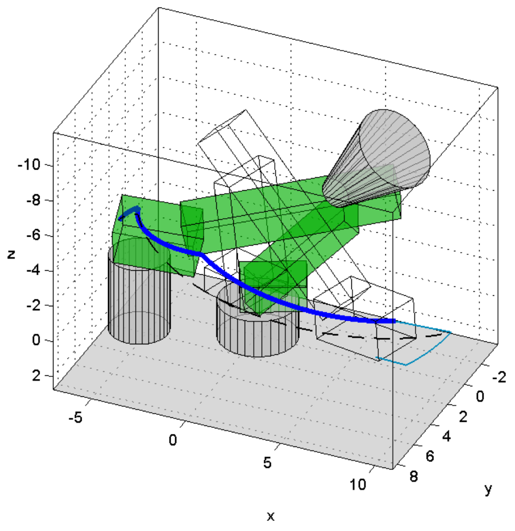


(d) Complete end-effector trajectory

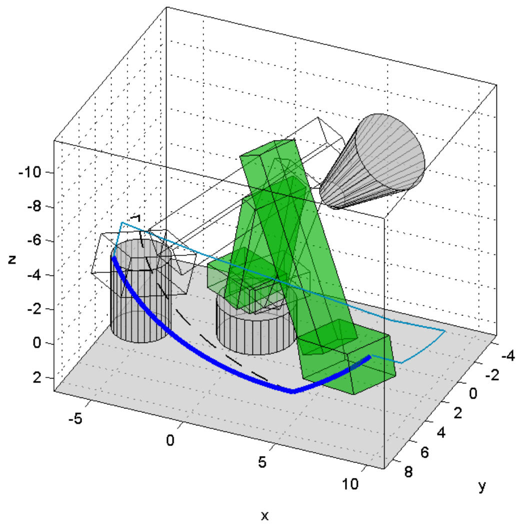
Fig. 8: Open-loop disturbance-free end-effector trajectory as viewed by an observer in the world frame. The dark blue curve represents the trajectory that would be traced out by the end effector along its new planned path, while the teal curve traces the already-traversed path. The transparent outline of the manipulator geometry illustrates its initial configuration at the beginning of the maneuver. The green-shaded outline indicates its targeted configuration.



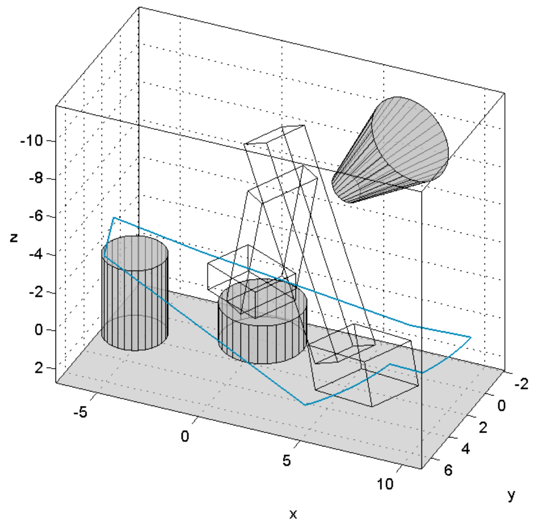
(a) Maneuver from initial position to battery



(b) Maneuver to deliver battery to cup

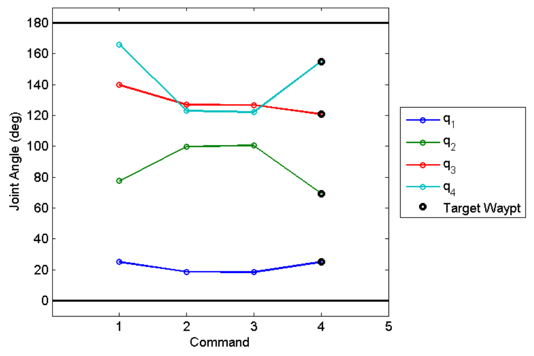


(c) Return maneuver to default position

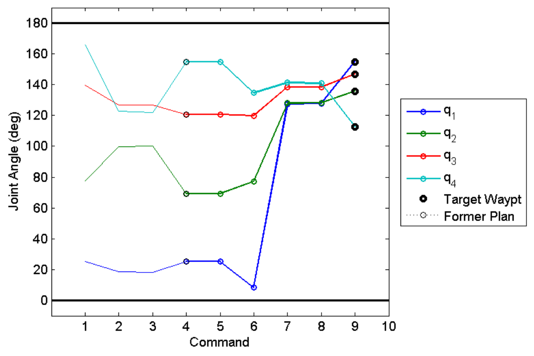


(d) Complete end-effector trajectory

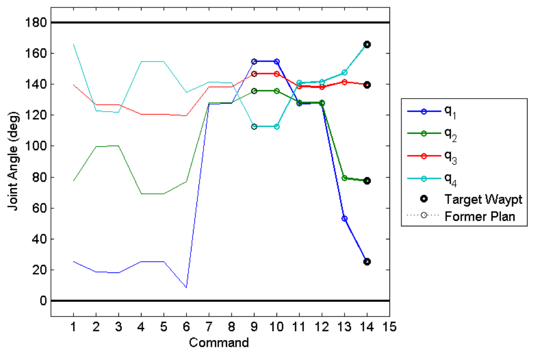
Fig. 9: Closed-loop end-effector trajectory as viewed by an observer in the world frame, with temperature obstacle included. Plans that have been revised during maneuver execution are indicated as dashed black lines (this indicates that after a horizon limit has been reached, the manipulator discovered a more efficient and/or safer trajectory to the goal). Note the cone that is introduced shortly after the initiation of the second maneuver (while holding the battery). This represents the introduction of the heat source, currently aimed towards the upper surface of the third manipulator link. The manipulator successfully "registers" its presence and develops a new plan that avoids the dangerous region and completes the desired task.



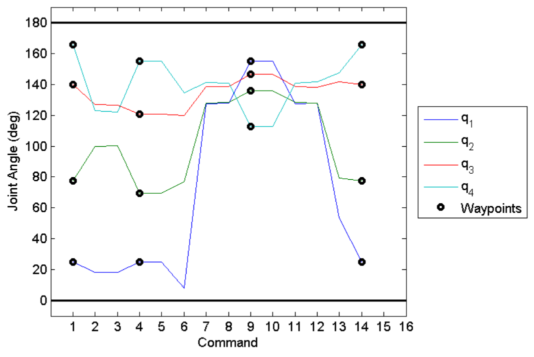
(a) Maneuver from initial position to battery



(b) Maneuver to deliver battery to cup

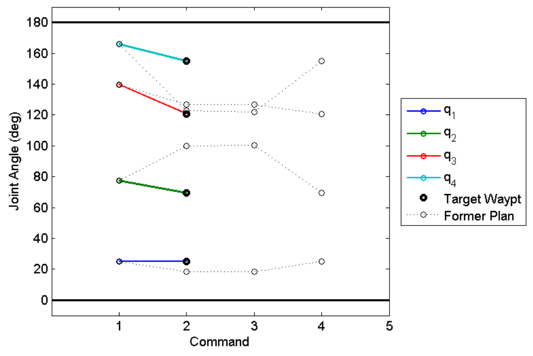


(c) Return maneuver to default position

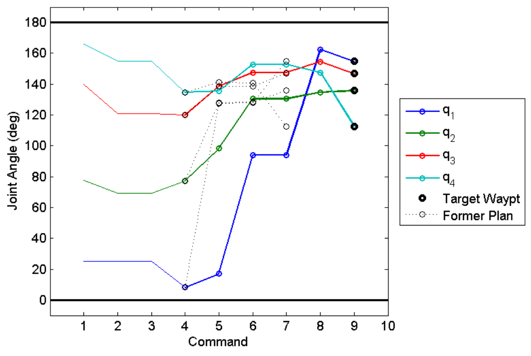


(d) Complete joint angle plan

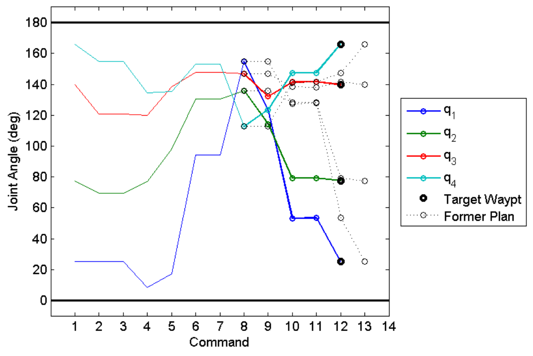
Fig. 10: Open-loop disturbance-free joint angle motion plan. This motion plan represents the nominal plan developed from the pre-computed RRT's after 2500 iterations. Note the system remains well within the imposed joint angle bounds.



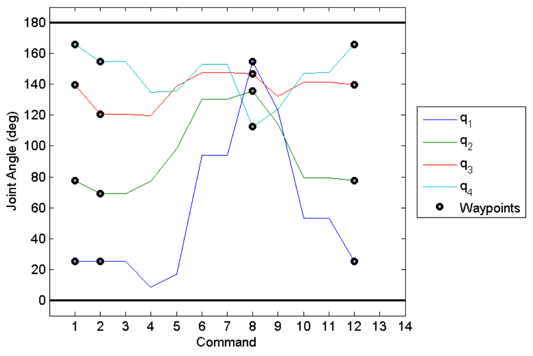
(a) Maneuver from initial position to battery



(b) Maneuver to deliver battery to cup



(c) Return maneuver to default position



(d) Complete joint angle plan

Fig. 11: Updates in the closed-loop joint angle motion plan with temperature obstacle included. This series of diagrams illustrates the revisions made to the joint angle command history as the RRT's become more mature and/or updates become necessary due to the introduction of the temperature obstacle.

References

[1] J. Rouquette, J. Haines, V. Bornand, M. Pintard, Ph. Papet, C. Bousquet, L. Konczewicz, F. A. Gorelli, and S. Hull. Pressure Tuning of the Morphotropic Phase Boundary in Piezoelectric Lead Zirconate Titanate. *Phys. Rev. B*, 70:014108, Jul 2004.

[2] Hu Chengguo and Hu Shengshui. Carbon Nanotube-Based Electrochemical Sensors: Principles and Applications in Biomedical Systems. *Journal of Sensors*, 2009:1–40.

[3] K. S. Karimov, M. Saleem, Z. M. Karieva, A. Khan, T. A. Qasuria, and A. Mateen. A Carbon Nanotube-Based Pressure Sensor. *Physica Scripta*, 83:065703, 2011.

[4] Weiguo Liu, Bin Jiang, and Weiguang Zhu. Self-Biased Dielectric Bolometer from Epitaxially Grown Pb(Zr,Ti)O[sub 3] and Lanthanum-doped Pb(Zr,Ti)O[sub 3] Multilayered Thin Films. *Applied Physics Letters*, 77(7):1047–1049, 2000.

[5] Xiaoliang Zhao, Tao Qian, Gang Mei, Chiman Kwan, Regan Zane, Christi Walsh, Thurein Paing, and Zoya Popovic. Active Health Monitoring of an Aircraft Wing with an Embedded Piezoelectric Sensor/Actuator Network: I. Defect Detection, Localization and Growth Monitoring. *Smart Materials and Structures*, 16(4):1218, 2007.

[6] Tomás Lozano-Pérez and Michael A. Wesley. An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[7] Ellips Masehian and Davoud Sedighizadeh. Classic and Heuristic Approaches in Robot Motion Planning - A Chronological Review. In *Proc. World Academy of Science, Engin. and Tech*, pages 101–106, 2007.

[8] Tomás Lozano-Pérez. A Simple Motion-Planning Algorithm for General Robot Manipulators. *Proc. IEEE Conf. on Robotics and Automation*, 3(3):224 –238, June 1987.

[9] Pang C. Chen and Yong K. Hwang. SANDROS: A Dynamic Graph Search Algorithm for Motion Planning. *Proc. IEEE Conf. on Robotics and Automation*, 14(3):390 –403, June 1998.

[10] Wayne F. Carriker, Pradeep K. Khosla, and Bruce H. Krogh. Path Planning for Mobile Manipulators for Multiple Task Execution. *Proc. IEEE Conf. on Robotics and Automation*, 7(3):403 –408, June 1991.

[11] Jérôme Barraquand and Pierre Ferbach. A Penalty Function Method for Constrained Motion Planning. In *Proc. IEEE Conf. on Robotics and Automation*, volume 2, pages 1235 –1242, May 1994.

[12] Oussama Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. of Robotics Research*, 5(1):90–98, 1986.

[13] Micha Sharir. Algorithmic Motion Planning, 1997.

[14] Yong K. Hwang and Narendra Ahuja. Gross Motion Planning — A Survey. *ACM Computing Surveys*, 24(3):219–291, Sept 1992.

[15] Koichi Hashimoto. *Visual Serving: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*, volume 7. World Scientific Pub. Co. Inc, 1993.

[16] Yong Yu and Kamal Gupta. On Sensor-based Roadmap: A Framework for Motion Planning for a Manipulator Arm in Unknown Environments. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 1919 –1924, Oct 1998.

[17] Allison M. Okamura, Neils Smaby, and Mark R. Cutkosky. An Overview of Dexterous Manipulation. In *Proc. IEEE Conf. on Robotics and Automation*, volume 1, pages 255 –262 vol.1, 2000.

[18] Francesco Mazzini. *Tactile Mapping of Harsh, Constrained Environments, with an Application to Oil Wells*. PhD thesis, Massachusetts Institute of Technology, Dept. of Mechanical Engineering, Boston, MA, 2011.

[19] Charles Fox, Mat Evans, Martin Pearson, and Tony Prescott. Tactile SLAM with a Biomimetic Whiskered Robot. In *Proc. IEEE Conf. on Robotics and Automation*, pages 4925 –4930, May 2012.

[20] M. Guilbert, L. Joly, and P.-B. Wieber. Optimization of Complex Robot Applications under Real Physical Limitations. *Int. J. of Robotics Research*, 27(5):629–644, 2008.

[21] Patrick Pledel and Yasmina Bestaoui. Actuator Constraints in Point to Point Motion Planning of Manipulators. In *Proc. IEEE Conf. on Decision and Control*, volume 2, pages 1009 –1010, Dec 1995.

[22] Edward Cheung and Vladimir J. Lumelsky. Proximity Sensing in Robot Manipulator Motion Planning: System and Implementation Issues. *IEEE Trans. on Robotics and Automation*, 5(6):740–751, 1989.

[23] J. H. Reif. Complexity of the Mover's Problem and Generalizations. In *20th Annual Symposium on Foundations of Computer Science*, pages 421 –427, October 1979.

[24] S. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

[25] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. In *Proc. 23rd Conf. on Comp. Graphics and Interactive Techniques*, SIGGRAPH '96, pages 171–180, New York, NY, USA, 1996. ACM.

[26] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *Int. J. of Robotics Research*, 30(7):846–894, June 2011.

[27] J. Cortés and T. Siméon. Sampling-Based Motion Planning Under Kinematic Loop-Closure Constraints. *Algorithmic Foundations of Robotics VI*, pages 75–90, 2005.

[28] B. Burns and O. Brock. Sampling-Based Motion Planning with Sensing Uncertainty. In *Proc. IEEE Conf. on Robotics and Automation*, pages 3313–3318. IEEE, 2007.

[29] Leonard Jaillet, Judy Hoffman, Jur Van den Berg, Pieter Abbeel, Josep M. Porta, and Ken Goldberg. EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2646–2652, Sept 2011.

[30] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-Based Roadmap of Trees for Parallel Motion Planning. *IEEE Trans. on Robotics and Automation*, 21(4), August 2005.

[31] Alejandro Perez, Robert Platt, George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. LQR-RRT*: Optimal Sampling-Based Motion Planning with Automatically Derived Extension Heuristics. In *Proc. IEEE Conf. on Robotics and Automation*, Boston, MA, 2012.

[32] S. M. LaValle and J. J. Kuffner. Randomized Kinodynamic Planning. *Int. J. of Robotics Research*, 20(5):378–400, 2001.

[33] J.J. Kuffner Jr and S.M. LaValle. RRT-Connect: An Efficient Approach to Single-query Path Planning. In *Proc. IEEE Conf. on Robotics and Automation*, volume 2, pages 995–1001. IEEE, 2000.

[34] S.M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998.