CAT-RRT: Motion Planning that Admits Contact One Link at a Time

Nataliya Nechyporenko*, Caleb Escobedo, Shreyas Kadekodi, and Alessandro Roncone

***Abstract*— Current motion planning approaches rely on binary collision checking to evaluate the validity of a state and thereby dictate where the robot is allowed to move. This approach leaves little room for robots to engage in contact with an object, as is often necessary when operating in densely cluttered spaces. In this work, we propose an alternative method that considers contact states as high-cost states that the robot should avoid but can traverse if necessary to complete a task. More specifically, we introduce *Contact Admissible Transition-based Rapidly exploring Random Trees* (CAT-RRT)[1], a planner that uses a novel per-link cost heuristic to find a path by traversing high-cost obstacle regions. Through extensive testing, we find that state-of-the-art optimization planners tend to over-explore low-cost states, which leads to slow and inefficient convergence to contact regions. Conversely, CAT-RRT searches both low and high-cost regions simultaneously with an adaptive thresholding mechanism carried out at each robot link. This leads to paths with a balance between efficiency, path length, and contact cost.**

I. Introduction

Robot behaviors are designed around the fundamental safety constraint of collision-free paths, as it ensures minimal physical interaction with the environment that could lead to robot error states or damage. However, this principle is oftentimes too limiting, as environmental constraints (e.g. tight spaces, areas with occlusion), perceptual constraints (e.g. narrow field of view, sensor inaccuracies), and operational constraints (e.g. maintaining a vertical cup orientation to avoid spilling) must also be accounted for while guaranteeing a collision-free path. As a result, a robot manipulator will likely fail to reach into a cluttered space due to the minimal clearance between the arm and the objects required to meet collision-free guarantees (see Fig. 1). Because motion planning is a fundamental component of a robot operating in the real world, having it restricted means significantly hindering robot capabilities; this limits the potential for robots to complete real-world tasks in unstructured or semi-structured environments such as harvesting fruit on a farm or picking items in a cluttered warehouse.

In this work, we are motivated by the idea that a binary collision test with a measure of whether the robot is in collision with the environment is insufficient to delineate the boundary between a valid or an invalid motion plan. Collision checkers provide the motion planner with a query function to test whether two geometric models overlap [2, 3]. Rather than invalidating any interactions between the robot and the environment, it is possible to evaluate them based on a continuous scale of object contact. This allows a robot to consider paths that would be discarded by traditional motion planning techniques while increasing success rate and enabling the robot to explore the environment through contact.

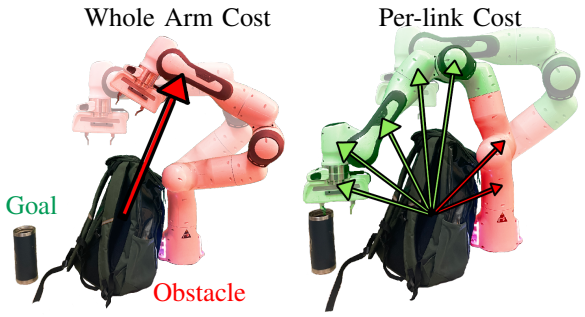

Fig. 1: CAT-RRT is an optimization planner which uses a per-link cost heuristic to generate a path in clutter by allowing contact to occur if it is necessary to succeed at the task. Rather than invalidating contact states or restricting motion for the entire arm (left), we propose a method that generates a path by prioritizing the least impacted links (right).

More specifically, in this paper we introduce a motion planner, Contact Admissible Transition-based Rapidly exploring Random Trees (*CAT-RRT*), that can generate paths in cluttered and unstructured environments by guiding the robot through states of *admissible contact*, which we define as contact necessary to reach the goal configuration. We are inspired by the literature in optimization-based motion planning [4, 5], which differs from traditional search-based motion planning in that it seeks to find a path that optimizes over a cost function. In particular, Transition-based Rapidly Exploring Random Tree (T-RRT, [6, 7]) uses the output of a cost function to increment or decrement a single global variable, called *temperature*, which is proportional to the likelihood of accepting high-cost states. CAT-RRT differs from T-RRT by not only using a set of temperatures, but also having each branch within the search tree adjust their own temperatures. This allows the planner to simultaneously propagate paths into low and high-cost regions based on multiple variables. We define cost with respect to proximity or contact with an obstacle, as shown in Fig. 1.

Additionally, in prior work cost functions are often defined to minimize travel distance [4], as short paths are a desired property [5]. In our work, we define a novel per-link cost heuristic which computes artificial repulsive and attractive

*Authors are with the Human Interaction and Robotics [HIRO] Group, Computer Science Department, University of Colorado Boulder, Boulder, CO USA. This work is partially supported by NSF FW-HTF grant #2222952/2953. NN is supported by NSF DGE #2040434. name.surname@colorado.edu

[1]Supplementary video and open source code [1].

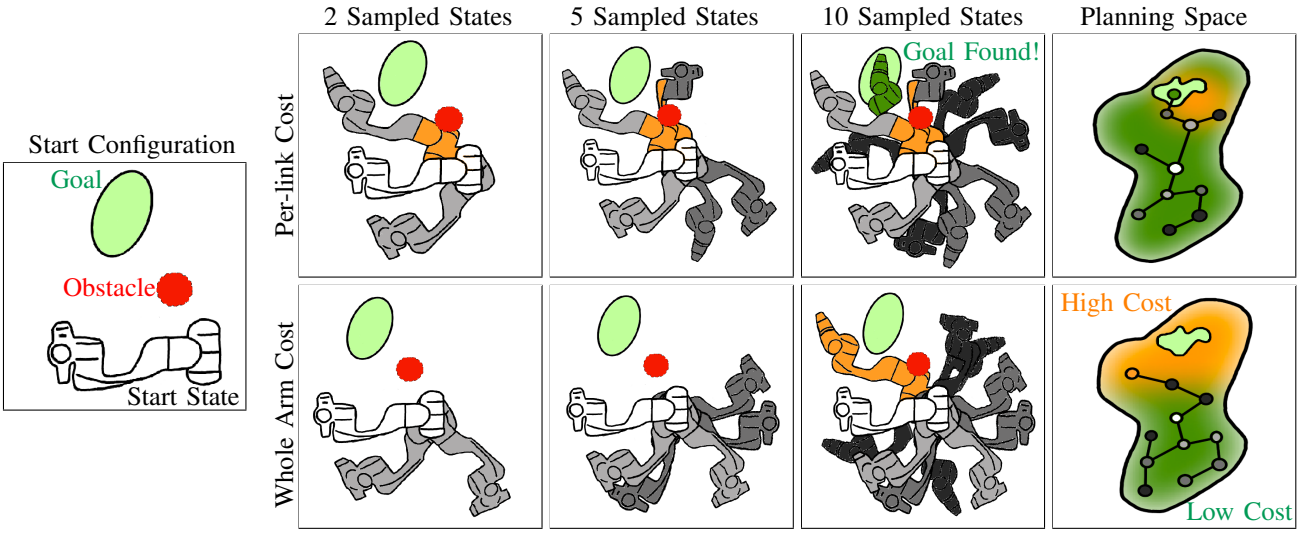

Fig. 2: Example scenario for *per-link* (top row) and *whole-arm* cost (bottom row) with a common start configuration (leftmost vignette). Object–robot contact is shown in orange and sampled states are grouped in gray scale depending on when the state was sampled, darker states are sampled later in time. The rightmost column depicts the planning space of both cost heuristics with green being a low cost area and orange being high cost. The per-link cost planner is able to find the goal location due to a reduced high cost area surrounding the goal location even though some links are in contact with the object.

vectors, together forming an artificial potential field (APF) [8], for every arm link. This allows the planner to assign a different temperature to every link and prioritize motion of the links that are least impacted by vector repulsion, as shown in Fig. 2.

In summary, our contributions are: 1) a novel optimization planner which successfully generates feasible trajectories even when the robot may need to come in contact with an obstacle; 2) an APF-based per-link cost heuristic which prioritizes motion with links that are unrestricted by contact. We performed an extensive quantitative evaluation in simulation and a qualitative demonstration in the real world. Collectively, our results demonstrate that, while relevant literature struggles to generate any path into high-cost regions, CAT-RRT can consistently find feasible trajectories by gradually admitting contact one link at a time.

II. Related Work

In this section, we analyze three major approaches branching from optimization-based Rapidly Exploring Random Tree (RRT): informed approaches, stochastic approaches, and node-changing approaches. We give a brief overview of the representative planners we choose from each category to use as baselines for our work. We also summarize several works that explore contact admissible motion planning without a focus on optimization.

In the wake of success of sampling-based planners, RRT has been widely adopted due to its simplicity and efficiency [9–11]. However, because any feasible path is accepted without regard for path quality, it generally produces suboptimal solutions [12]. To improve upon RRT, other works propose a method of prioritizing nodes that converge toward an optimal solution [12, 13]. Often, this is achieved with an informed heuristic during node creation or a modified acceptance test that uses path quality to bias nodes toward low cost regions. The most prominent of these is RRT*, which is used as one of the baselines in our evaluation. RRT* is an incremental sampling-based planning algorithm that maintains a tree without any "redundant" edges—edges that are not within the lowest cost path from the start to current node in the tree [13]. RRT*, like other tree refinement methods, has optimality guarantees. There are several existing modifications of RRT* as well, including using potential field-guided RRT* sampling, but these have not been tested on high-dimensional robot systems [14–16].

A new wave of batch-informed trees have been proposed, which focus on both efficiency and path quality [17–19]. One such planner is Batch-Informed Tree* (BIT*), used in our evaluation, which leverages a local optimization module to improve an initial path toward a local optimum. BIT* is probabilistically complete and has been shown to find solutions more often than other almost-surely asymptotically optimal planners.

Other optimization methods include stochastic planning algorithms. One such example is Transition-based RRT (T-RRT), which propagates a tree search based on a stochastic optimization method with transition tests to accept or reject new states, but offers no optimality guarantees [6, 7]. Other works build on T-RRT to enable anytime behavior, bi-directional tree growth, and applicability to multi-agent systems [20–23]. CAT-RRT shares a T-RRT-like optimization approach but with a unique transition test (detailed in Section III-B). To demonstrate this distinction, we use T-RRT as a baseline in our evaluation.

Several planners attempt to improve optimization effi-

ciency by biasing sampled nodes based on a chosen direction [24, 25]. This strategy is desirable because the search can be moved in the direction of low-cost regions especially when guided by potential fields. Vector Field RRT (VF-RRT) does this through the Upstream Criterion, as defined in Eq. (6), which is used to bias sampling toward nodes that minimize the extent to which a path goes against a given vector field [26]. We chose VF-RRT to evaluate this strategy since it is highly applicable to potential field-based cost functions which our work relies on.

The following two papers are the closest to our work. [25] develops a potential field guided RRT* algorithm for the problem of fruit harvesting [25]. It defines leaves as permeable obstacles, which the robot is allowed to come into contact with after incurring a cost. The authors use a combination of an RRT*-like approach with tree refinement and a VF-RRT-like approach with node biasing—both of which are evaluated in our experimental framework. Finally, [27] compares a potential field cost function as applied to T-RRT and other sampling approaches [27]. The authors do not consider contact behaviors and rely on a simplified cost calculation between a single point on the robotic arm and an arbitrary obstacle point.

Finally, several works address contact admissibility and motion planning in the context of perception. Instead of using optimization, these works replace a binary collision-check function with a binary cost-based function [28–30]. They rely on a threshold that reflects how much contact a robot can make with an object. Such a threshold is challenging to define ahead of time for all environments. In contrast, our work uses an adaptive threshold mechanism.

III. Methods

In this section, we first outline the problem of path finding. Next, we describe how CAT-RRT plans a path while optimizing over a cost function using a set of temperatures and a transition test. Then, we describe how the temperatures are generated based on a separate cost for each link of the arm. Finally, we define additional cost heuristics from existing literature, which are used to evaluate CAT-RRT.

A. Problem description

We use a similar definition of the planning problem as [18]. Let $Q \subseteq \mathbb{R}^n$ be the state space of the planning problem. Let $\mathbf{q}_{\text{start}} \in Q_{\text{free}}$ be the initial state of joint angles and $\mathbf{q}_{\text{goal}} \subset Q_{\text{free}}$ be the set of the desired goal states. Let $\sigma : [0, 1] \rightarrow Q_{\text{free}}$ be a continuous map to a sequence of states through a space of bounded variation that can be executed by the robot (i.e. self-collision free, feasible path) and Σ be the set of all such nontrivial paths. The optimal planning problem is then formally defined as the search for a path, $\sigma^* \in \Sigma$, that minimizes a given cost function, $c : \Sigma \rightarrow \mathbb{R}^n_{\geq 0}$, while connecting $\mathbf{q}_{\text{start}}$ to $\mathbf{q}_{\text{goal}} \in Q_{\text{goal}}$ where $\mathbb{R}^n_{\geq 0}$ is the set of non-negative real numbers.

B. Motion planning with CAT-RRT

CAT-RRT benefits from the exploratory strength of RRT-like algorithms that quickly expand toward large regions of unexplored space. Additionally, it integrates features of stochastic optimization methods from T-RRT-like planners, which use transition tests to accept or reject potential states. The main algorithm runs as follows: a random state, $\mathbf{q}_{rand}$, is selected from the configuration space, which is a minimum distance away from $\mathbf{q}_{near}$. A transition test function is used to evaluate $\mathbf{q}_{rand}$. If it passes the transition test, then it is added to the tree, and the process repeats until a path to $\mathbf{q}_{goal}$ is found. The main tree construction algorithm of CAT-RRT is defined in [6] and will not be reintroduced here for brevity. However, the transition test is unique to our approach and defined in Algorithm 1. First, we evaluate a vector of costs, $\mathbf{C}$, for $\mathbf{q}_{rand}$, with each cost corresponding to a link on the arm. Next, we obtain a vector of temperatures, $\mathbf{T}$, stored in the nearest node of the tree. One link at a time, we evaluate and update the tree node based on a transition test. A transition test is passed if the link's cost, $\mathbf{C}[i]$, is lower than its allowed temperature, $\mathbf{T}[i]$, and the temperature is reduced unless it reaches a user-defined minimum value, t_{min}. If all the links pass the test, then the temperature vector is stored in the child node of the added state. A failed transition test increases the temperature for the given link, thereby increasing the chance of a state sampled in that region to be accepted in the next iteration. Although previous works use an intermediate exponential function based on the Metropolis criterion to relate cost and temperature [6], we did not find this beneficial and opted for a direct relationship. In our algorithm, the temperature is synonymous to a dynamic cost threshold. Both ω and γ are user-defined values that control the rate of temperature decrement and increment. Our source code provides more specifics on parameter tuning [1].

Algorithm 1 CAT-RRT Transition Test

$\mathbf{C} \leftarrow$ `GetPerLinkCost`$(\mathbf{q}_{near}, \mathbf{q}_{rand}, \mathbf{q}_{goal})$ ▷ Eq. 4
$\mathbf{T} \leftarrow$ `GetTemperature`$(Node_{parent})$
for $i = 0...L$ **do**
  if $\mathbf{C}[i] < \mathbf{T}[i]$ and $\mathbf{T}[i] > t_{min}$ **then**
    $\mathbf{T}[i] -= \omega$
  else if $\mathbf{C}[i] > \mathbf{T}[i]$ **then**
    $\mathbf{T}[i] += \gamma$
    return $False$
  end if
end for
`StoreTemperature`$(\mathbf{T}, Node_{child})$
return $True$

CAT-RRT differs from T-RRT in that, rather than having a global temperature parameter for all nodes, the temperature is stored at the parent node and inherited by the child node. Furthermore, rather than storing a temperature as a scalar for the entire robot body, we create a temperature vector where each link is independently represented. Consequently, the tree accepts or rejects nodes based on the temperature at every link. This results in CAT-RRT's two distinct properties: 1) each branch of the tree regulates its own temperature, and 2) each link on the robotic arm enters high cost regions

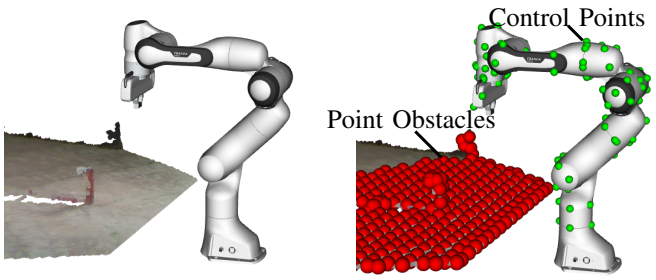

Fig. 3: The image on the left shows the robot in front of a point cloud of an object sitting on top of a table. The image on the right shows the same scene with an overlay of point obstacles in red and robot control points in green.

independent from the rest of the kinematic chain. When one link is in a high-cost region, it will stay in this position while the other links maintain low-cost region positioning. In the real world, this equates to one link of the robotic arm maintaining contact with an object while the other links continue to traverse contact-free spaces. This is in contrast to planners that attempt to always maintain low costs throughout the arm which may lead to scattered contact along a path. Fig. 2 summarizes how CAT-RRT converges to a goal state using discrete costs along the robotic arm. In the absence of obstacles, CAT-RRT's transition test is not invoked and the planner operates as RRT. Next, we describe how the robot's perception of the environment is converted to a cost for each link.

In this work, we try to step away from the dependence on high-resolution collision models for motion planning, as these are often unavailable for a robot operating in unstructured settings. However, each planner does require a basic understanding of the environment and the robot's position in space. To acquire this understanding, as detailed in Fig. 3, we assume the robot is equipped with a camera that relays depth perception information, as a point cloud, to the motion planning algorithm. The point cloud is converted to *point obstacles*, which are used as the basis for the planner's obstacle representation. Fig. 3 shows the original point cloud and point obstacles, $p_k \in \Lambda$. Similarly, to ease reliance on 3D mesh models, the planner uses a set of *control points* to represent the robot and its position. The control points, $p_q \in \Gamma$, are represented by green spheres on Fig. 3.

C. Defining the cost heuristics

1) Controlling cost magnitude: Repulsive vector costs and unit magnitude costs serve as the basis of the cost heuristics defined in this paper. These costs are generated from the distance between point obstacles and robot control points. The vector magnitudes, $\vec{\mathbf{v}}$, are scaled to be inversely proportional to the distance. Rather than opting for the traditional potential field equation introduced by Khatib et al. [8], which increases the repulsive force to infinity as the distance to obstacles becomes zero, we use a scaling function, $\boldsymbol{S}$, shown in Eq. (1). Both a and b are scalar parameters, which allow us to control the magnitude of cost associated with contact.

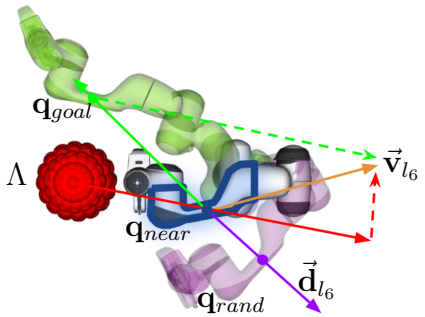

Fig. 4: Robot's initial configuration ($\mathbf{q}_{near}$) in white, goal configuration ($\mathbf{q}_{goal}$) in green, random sampled state ($\mathbf{q}_{rand}$) in purple, and a set of point obstacles (Λ) in red. The directional vector for link number six, $\vec{\mathbf{d}}_{l_6}$, points from $\mathbf{q}_{near}$ to $\mathbf{q}_{rand}$. The desired directional vector for the link, $\vec{\mathbf{v}}_{l_6}$, is a weighted sum between the vector from Λ to $\mathbf{q}_{near}$ and the vector from $\mathbf{q}_{near}$ to $\mathbf{q}_{goal}$.

Here, a controls the maximum scaling value of $\vec{\mathbf{v}}$ and b controls how fast the function converges to the maximum as $||\vec{\mathbf{v}}||$ goes to zero.

$$\boldsymbol{S}(\vec{\mathbf{v}}) = \frac{a * \vec{\mathbf{v}}}{b * ||\vec{\mathbf{v}}|| + 1} \tag{1}$$

2) Per-link cost heuristic used with CAT-RRT: The per-link cost heuristic is an essential component of CAT-RRT as it guides the search tree. The desired vector at link l, $\vec{\mathbf{v}}$, defines the desired direction of motion in Cartesian space for every link of the arm. K is the number of point obstacles, N is the number of control points, L is the number of links, p_k is the kth obstacle point, $p_{q_{near},i}$ is the ith control point on the robot's $\mathbf{q}_{near}$ state, and link number l is $l \in [1, ..., L]$, $p_{q_{goal},i}$ is the ith control point on the robot's goal state. α and β are scalar parameters.

$$\vec{\mathbf{v}} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N}\sum_{i=1}^{N}\alpha * \mathbf{S}(p_{q_{near},i} - p_k) + \beta * (p_{q_{goal},i} - p_{q_{near},i}) \tag{2}$$

The random directional vector $\vec{\mathbf{d}}$ from $\mathbf{q}_{near}$ to the uniformly sampled state $\mathbf{q}_{rand}$ at every link is obtained as follows;

$$\vec{\mathbf{d}} = \frac{1}{N}\sum_{i=1}^{N}(p_{q_{rand},i} - p_{q_{near},i}) \tag{3}$$

The cost at every link $\mathbf{c}$ is defined by Eq. (4), with lower costs indicating an alignment between the directional vector and the desired vector.

$$\mathbf{c} = (-\vec{\mathbf{v}}) \cdot \vec{\mathbf{d}} \tag{4}$$

Fig. 4 shows the components of the per-link vector field alignment cost heuristic, which guides the robot away from obstacles and towards goal locations using randomly sampled states. Next, we define the cost heuristics from previous work and how we implement them. These methods are used by state-of-the-art planners for comparison against CAT-RRT and the per-link cost heuristic.

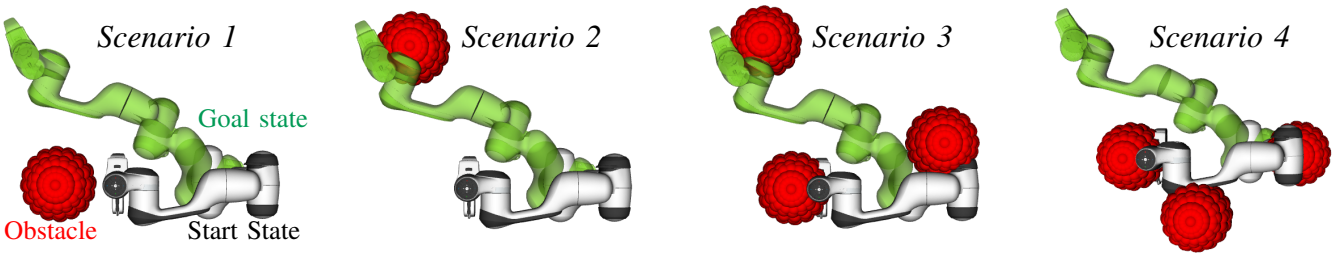

Fig. 5: In our evaluation, the robot is tasked with finding a path from the start state (white) to the goal state (green) while moving through obstacle regions (red) in four experimental scenarios of increasing complexity. The scenarios from left to right are increasingly more complex with obstacles overlapping with the start and goal states.

D. Comparison with state of the art

1) Obstacle overlap heuristic used with T-RRT, RRT, and BIT*:* The "permissible contact" planners [28–30] detailed in Section II evaluate the cost of a path based on the amount of overlap between the robot and the potential obstacles in the environment. In this work, the amount of obstacle–robot overlap is equivalent to adding up the vector magnitudes given from Eq. (1), which is implemented as the cost function $\mathbf{C}$:

$$\mathbf{C} = \sum_{l=1}^{L} \left\| \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N} \sum_{i=1}^{N} \mathbf{S}(p_k - p_{q_{near},i}) \right\| \tag{5}$$

This baseline cost heuristic is used to generate low-cost paths by T-RRT, RRT*, and BIT*.

2) Upstream Criterion used with VF-RRT: As discussed in Section II, one approach to improve the convergence rate of sampling-based planners is to adjust the newly sampled nodes in the direction of a vector field [24–26]. To test this approach, we use VF-RRT with the Upstream Criterion [26]. The Upstream Criterion is defined as:

$$\int_0^L (||f(q(s))|| - \langle f(q(s)), q^{'}(s) \rangle) ds \tag{6}$$

where $f(q(s))$ is a piecewise continuous vector field and $||f(q(s))||$ represents the norm of $\langle f(q(s)), q^{'}(s) \rangle$. The function $f(q(s))$ is not explicitly defined in the original paper and is left for the user to define based on a specific application. Since we are planning in robot configuration space, the output of $f(q(s))$ must be a vector of joint angles. However, our robot and the environment are defined in Cartesian space by point obstacles and control points. To obtain a set of joint angles from a set of points in Cartesian space, we apply the inverse of the Jacobian $\mathbf{J}$ to Eq. (1). $\mathbf{J}_l^{\dagger} \in \mathbb{R}^{3xl}$ is the Moore-Penrose pseudo-inverse of $\mathbf{J}$ at link l.

$$f(q(s)) = \sum_{l=1}^{L} \left(\mathbf{J}_l^{\dagger} \times \left(\frac{1}{K} \sum_{k=1}^{K} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{S}(p_k - p_{q_{near},i}) \right) \right) \tag{7}$$

IV. Experiments

The experimental evaluation is performed in both simulation (Section IV-B) and real-world (Section V-B). The former allows for repeatable and reproducible experiments, while the latter shows the applicability of planning with contact in the real world.

A. Specifications of the experimental testbed

Based on the discussion in Section II and the implementation in Section III, we evaluate: T-RRT, RRT*, and BIT*, which use the obstacle-robot overlap cost heuristic, VF-RRT, which uses the upstream criterion, and CAT-RRT, which uses the per-link cost heuristic. Each planner was allotted a maximum of 60 seconds to compute and refine a path. We believe this to be a reasonable amount of evaluation time and comparable to prior work—e.g. [18] used a 20 second limit for a similar 7 degree-of-freedom (DOF) problem to evaluate BIT* with limited compute power.

Each planner relies on a set of control points and point obstacles referred to in Section III-B. To obtain the control points, we extract 115 vertices from the robot's 3D mesh, openly available on the Franka Emika repository. To obtain point obstacles, we downsample a point cloud using the Point Cloud Library Voxel Grid [31] filter with a leaf size of 0.05m. The point cloud is then converted to the robot's frame of reference. Each of the resulting voxels, or values on a regular grid in 3D space, is considered as a point obstacle. In simulation, the point obstacles are added artificially to create example objects, represented by red orbs on Scenarios 1-4 in Fig. 5.

The planning algorithms are implemented in C++ with the the ROS (Noetic) framework [32]. We use TRRT, VF-RRT, RRT*, and BIT* within the Open Motion Planning Library [33] and integrate CAT-RRT within the library as well. We use Moveit! for simulation [34] and Rviz for visualization. Franka Emika Panda is used as the robot platform and an OAK-D Pro [35] camera to capture the point cloud. All experiments were performed on a computer with an Intel i9 processor and 16GB RAM.

B. Simulated experimental scenarios

For the simulated experiments, four scenarios of increasing complexity are designed—see Fig. 5. Scenario 1 evaluates if each planner can succeed in finding a path from a free low-cost start state to a free low-cost goal state in the presence of a single obstacle. This is a baseline scenario used to check fundamental path finding capabilities. In Scenario 2, the planner is asked to compute a path in which the robot's goal state is in contact with an obstacle. This scenario tests the planner's ability to plan into a high-cost region. Scenario

3 includes two obstacles at the start state and one in the goal state, which tests the planner's ability to traverse between two high-cost regions. Finally, Scenario 4 is set up similarly to Scenario 3, but with an additional obstacle blocking the path away from the other obstacles, meaning the robot cannot break contact with the high-cost regions as in Scenario 3. This tests how the planner is able to modulate high-cost regions across the robotic arm.

C. Metrics for evaluation

We evaluate each planner based on its ability to successfully generate a path within the allotted time. For the resulting trajectories, we measure the distribution of contact along the arm and the overall path length. These metrics represent the planner's ability to minimize contact cost while moving toward the goal. We run fifty trials for each planner in each scenario and average the metrics across the successful trials. Path length is calculated as the sum of the L^2-norm between the end-effector Cartesian points of the trajectory. For the simulated experiments, we measure the amount of contact along the trajectory by calculating the overlap between the 3D mesh of the arm and the obstacles. This is done through a collision post-processing step. First, we place spherical collision objects of the same size as the red point obstacle orbs into the robot scenario. Then, we run collision detection based on the Bullet Physics Engine on every state along the trajectory. The collision checker returns the number of states in collision and the contact depth, or penetration depth, between each overlapping robot-obstacle pair.

V. Results and Discussion

In this section, we summarize the results obtained after running the experiments outlined in Section IV. We demonstrate that T-RRT and VF-RRT struggle to navigate into high-cost regions with contact. While RRT* tends to prioritize shorter paths by incurring more contact, BIT* generates longer paths with less contact. In contrast, CAT-RRT tends to find a better balance between path length and contact depth.

A. Simulation experiments

1) Scenarios 1 & 2: All planners are able to find a path with no obstacle overlap for Scenario 1. However, the results from Scenario 2 demonstrate a significant rift in the capabilities of the planners, in that T-RRT and VF-RRT were able to compute a successful path 0/50 times while the other planners were able to find such a path 50/50 times. Table I summarizes the binary results of the planners in their ability to plan into high-cost regions.

The reason T-RRT struggles to find a path in Scenario 2 is because the global temperature variable is prohibitive in allowing the tree to explore high-cost regions. The temperature parameter is proportional to the probability of having a state accepted as a node in the tree. Fig. 6, right shows that the temperature drops early in the iteration phase because of the large number of samples generated in the low-cost space of the robotic arm. This prevents the states in the high-cost regions near obstacles from being accepted.

	T-RRT	VF-RRT	RRT*	BIT*	CAT-RRT
Scenario 1	✓	✓	✓	✓	✓
Scenario 2	✗	✗	✓	✓	✓

TABLE I: Here, ✓represents the planner successfully finding a path 50/50 times in under 60 seconds and ✗represents a failure or a 0/50 success rate of finding a path to the goal region.

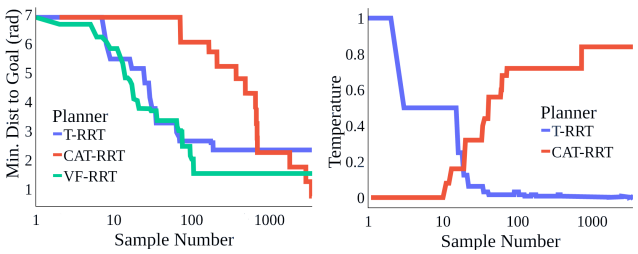

Fig. 6: Scenario 2 analysis of T-RRT and VF-RRT failures. (Left) The minimum distance to goal of new nodes as T-RRT builds its tree. (Right) The average temperature and distance to goal of CAT-RRT. Unlike T-RRT and VF-RRT, CAT-RRT overcomes the high-cost threshold and converges to a solution.

VF-RRT faces a similar issue: newly sampled states will always be directed away from the high-cost regions, which in Scenario 2 is where the goal state is located. Whereas a sample in a high-cost region of T-RRT would be rejected, with VF-RRT it would be adjusted downstream from the high-cost region and ultimately end up further from the goal state. Because T-RRT and VF-RRT cannot compute plans into high-cost regions within a reasonable amount of time given our testbed specification, we excluded them from our comparison chart in the subsequent tests in Scenario 3-4. In this sense, we use Scenario 2 to filter out methods which struggle to generate paths in complex cost spaces with high DOF robots.

2) Scenario 3: Table II details simulation results from Scenario 3. BIT* finds a solution path that is double the length of RRT* and almost triple the length of CAT-RRT. However, BIT* does maintain lowest minimum contact across the links. RRT* has a shorter path length than BIT*, but at the expense of high contact depth. CAT-RRT tends to generate a short path length without moving into the obstacles region as much as RRT*. CAT-RRT also produces its paths much faster computationally. Fig. 7 shows a sample end-effector trajectory taken by each planner in Scenario 3. RRT* chooses to move through the obstacle, BIT* takes a longer path but with low contact impact, and CAT-RRT chooses to navigate between the obstacles while optimizing for the cost at every link. Although we do not apply smoothing to any of the generated trajectories, CAT-RRT tends to suffer the most from this as it does not go through

Method	Scenario 3										Scenario 4								
	Path Metric			Total Contact Depth for Link (mm)								Path Metric			Total Contact Depth for Link (mm)							
	S (50)	T (s)	PL (m)	1	2	3	4	5	6	7	8	S (50)	T (s)	PL (m)	1	2	3	4	5	6	7	8
RRT*	36	62.9	1.7	0.	0.	0.1	15.4	16.1	12.8	23.9	28.4	31	63.2	1.1	0.	18.1	41.5	6.3	28.9	22.	32.6	35.8
BIT*	50	60.0	3.9	0.	0.	0.2	20.6	3.2	8.1	18.3	21.5	50	60.0	1.8	0.	23.2	43.8	10.7	30.1	29.7	28.7	24.5
CAT-RRT	50	18.7	1.2	0.	0.	0.	18.3	0.2	5.5	13.9	25.4	50	15.9	1.2	0.	24.6	51.2	5.6	10.2	13.	20.8	22.9

TABLE II: Experimental results of each planning algorithm for Scenarios 3 and 4. The path metrics include the number of successes out of 50 trials (S), the average time to compute the path within the allotted time budget of 60s (T), and the total path length of the end-effector (PL). All the metrics, including contact depth for each link, are averaged across the successful trials. The highlighted colors in Scenario 4 correspond to the maximum contact depth peaks in Figure 8.

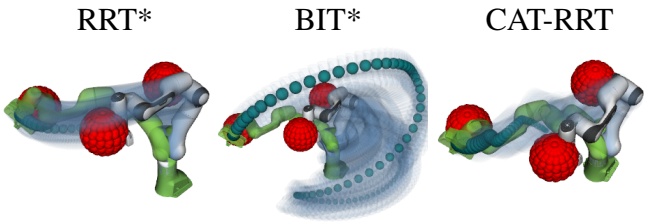

Fig. 7: Example trajectories for Scenario 3 generated by each planner. RRT* chooses to traverse the obstacle in front, BIT* first moves away from all obstacles before returning in the direction of the goal state, and CAT-RRT finds a low-cost path in between the two obstacles.

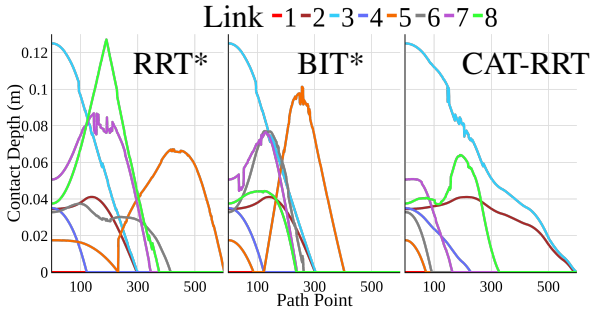

Fig. 8: Contact depth at each link along one generated sample trajectory in Scenario 4. The peaks correspond to a high level of overlap between the link and the obstacle. Whereas RRT* and BIT* have three and two peaks each, CAT-RRT maintains one prolonged contact at a single link, achieving a faster convergence to goal with a shorter path length.

any rewiring steps as the other planners. A post-processing trajectory optimization step can smooth out the trajectory and reduce the higher average contact depth values for CAT-RRT in Scenario 3.

3) Scenario 4: The results from Scenario 4 are also detailed on Table II. In this scenario, CAT-RRT outperforms the other planners in its ability to generate a path of shortest length, with the least impact, and faster computationally. On Fig. 8, the contact penetration depth at every link is plotted across a sample trajectory generated by each planner. With only one peak, as opposed to two and three for BIT* and RRT* respectively, CAT-RRT demonstrates its ability to keep one link in contact while moving other links through free space. This concept is highlighted in Fig. 1. This is another reason for which the path length of CAT-RRT is shorter, as it can maintain contact with the obstacle at the base while moving perpendicular to the obstacle at the end-effector. In contrast, the other planners tend to produce contact more randomly along the links while searching for a minimum-cost path to the goal. This results in longer high-cost trajectories for the arm and each link.

B. Real-world demonstration

To validate our simulation findings, we set up a real-world experiment that demonstrates a situation in which contact is harmless. More specifically, we show how the robot can reach for an object while making contact with a soft obstacle which overlaps with the goal state. Our supplementary video showcases the results [1]. Although the planning time of CAT-RRT remains a challenge for real-world operation, we believe this can be addressed with parallel computing and algorithm optimization.

VI. Conclusion and Future Work

This work is guided by the idea that planners can enhance their operational capabilities by reducing reliance on collision checking and increasing tolerance to contact with objects. We present a method that allows robots to intelligently plan for contact given a limited understanding of the environment. We show that our planner can successfully generate a path into high-cost regions with obstacles. Compared to other planners, which use a single cost for the entire arm, CAT-RRT can create shorter paths in less time using a per-link cost heuristic. In our future research, we aim to demonstrate how robots can help leverage more "action" in the "sense-perceive-act" paradigm [36]. To do so, we aim to tightly couple CAT-RRT with control ([37, 38]) to track contact during trajectory execution and perception to adjust planning costs based on object properties (e.g. hard or soft material). We believe that a robot that can plan and adjust for contact is better equipped to handle manipulation tasks in unstructured environments in the agricultural, industrial, and retail sectors.

References

[1] URL: https://nataliya.dev/cat-rrt.

[2] Jia Pan, Sachin Chitta, and Dinesh Manocha. "FCL: A general purpose library for collision and proximity queries". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3859–3866.

[3] P Jiménez, Federico Thomas, and Carme Torras. "Collision detection algorithms for motion planning". In: *Robot motion planning and control* (2005), pp. 305–343.

[4] Nancy M Amato et al. "Choosing good distance metrics and local planners for probabilistic roadmap methods". In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation*. Vol. 1. IEEE. 1998, pp. 630–637.

[5] Ron Wein, Jur Van Den Berg, and Dan Halperin. "Planning high-quality paths and corridors amidst obstacles". In: *The International Journal of Robotics Research* 27.11-12 (2008).

[6] Léonard Jaillet, Juan Cortés, and Thierry Siméon. "Transition-based RRT for path planning in continuous cost spaces". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 2145–2150.

[7] Léonard Jaillet, Juan Cortés, and Thierry Siméon. "Sampling-based path planning on configuration-space costmaps". In: *IEEE Transactions on Robotics* 26.4 (2010).

[8] Oussama Khatib. "The potential field approach and operational space formulation in robot control". In: *Adaptive and Learning Systems: Theory and Applications* (1986).

[9] Mohamed Elbanhawi and Milan Simic. "Sampling-Based Robot Motion Planning: A Review". In: *IEEE Access* (2014).

[10] Steven M. LaValle. *Rapidly-exploring random trees: A new tool for path planning*. 1998.

[11] Steven M. LaValle and Jr. James J. Kuffner. "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400.

[12] Jonathan D Gammell and Marlin P Strub. "Asymptotically optimal sampling-based motion planning methods". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4 (2021), pp. 295–318.

[13] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[14] Iram Noreen et al. "Optimal path planning in cluttered environment using RRT*-AB". In: *Intelligent Service Robotics* 11 (2018), pp. 41–52.

[15] Wei Wang et al. "An improved RRT path planning algorithm for service robot". In: *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. Vol. 1. IEEE. 2020, pp. 1824–1828.

[16] Hussein Mohammed, Lotfi Romdhane, and Mohammad A Jaradat. "RRT* N: An efficient approach to path planning in 3D for Static and Dynamic Environments". In: *Advanced Robotics* 35.3-4 (2021), pp. 168–180.

[17] Alexander C Holston, Deok-Hwa Kim, and Jong-Hwan Kim. "Fast-BIT: Modified heuristic for sampling-based optimal planning with a faster first solution and convergence in implicit random geometric graphs". In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2017, pp. 1892–1899.

[18] Jonathan D Gammell, Timothy D Barfoot, and Siddhartha S Srinivasa. "Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search". In: *The International Journal of Robotics Research* 39.5 (2020), pp. 543–567.

[19] Marlin P Strub and Jonathan D Gammell. "Advanced BIT (ABIT): Sampling-Based Planning with Advanced Graph-Search Techniques". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020.

[20] Didier Devaurs, Thierry Siméon, and Juan Cortés. "Enhancing the transition-based RRT to deal with complex cost spaces". In: *2013 IEEE international conference on robotics and automation*. IEEE. 2013, pp. 4120–4125.

[21] Didier Devaurs, Thierry Siméon, and Juan Cortés. "A multi-tree extension of the transition-based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 2991–2996.

[22] Didier Devaurs, Thierry Siméon, and Juan Cortés. "Optimal path planning in complex cost spaces with sampling-based algorithms". In: *IEEE Transactions on Automation Science and Engineering* 13.2 (2015), pp. 415–424.

[23] Romain Iehl, Juan Cortés, and Thierry Siméon. "Costmap planning in high dimensional configuration spaces". In: *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2012, pp. 166–172.

[24] Dmitry Berenson, Thierry Siméon, and Siddhartha S Srinivasa. "Addressing cost-space chasms in manipulation planning". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 4561–4568.

[25] Heramb Nemlekar et al. "Robotic Lime Picking by Considering Leaves as Permeable Obstacles". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 3278–3284.

[26] Inyoung Ko, Beobkyoon Kim, and Frank Chongwoo Park. "Randomized path planning on vector fields". In: *The International Journal of Robotics Research* 33.13 (2014).

[27] Ryo Kabutan and Takeshi Nishida. "Motion Planning by T-RRT with Potential Function for Vertical Articulated Robots". In: *Electrical Engineering in Japan* 204.2 (2018).

[28] Tapomayukh Bhattacharjee et al. "A robotic system for reaching in dense clutter that integrates model predictive control, learning, haptic mapping, and planning". In: Georgia Institute of Technology. 2014.

[29] Daehyung Park et al. "Interleaving planning and control for efficient haptically-guided reaching in unknown environments". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2014, pp. 809–816.

[30] Brad Saund and Dmitry Berenson. "Motion planning for manipulators in unknown environments with contact sensing uncertainty". In: *Proceedings of the 2018 International Symposium on Experimental Robotics*. Springer. 2020.

[31] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1–4.

[32] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[33] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. "The open motion planning library". In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[34] Sachin Chitta. "MoveIt!: an introduction". In: *Robot Operating System (ROS) The Complete Reference (Volume 1)* (2016).

[35] Luxonis. *OAK-D: Stereo camera with Edge AI*. 2020. URL: https://luxonis.com/.

[36] Jeannette Bohg et al. "Interactive perception: Leveraging action in perception and perception in action". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291.

[37] Caleb Escobedo et al. "Contact Anticipation for Physical Human–Robot Interaction with Robotic Manipulators using Onboard Proximity Sensors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021.

[38] Caleb Escobedo et al. "A Framework for the Systematic Evaluation of Obstacle Avoidance and Object-Aware Controllers". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022.