Motion Planning for Manipulators in Unknown Environments with Contact Sensing Uncertainty

Brad Saund and Dmitry Berenson
University of Michigan

Abstract. Localization error, sensor noise, and occlusions can lead to an imperfect model of the environment, which can result in collisions between a robot arm and unobserved obstacles when manipulating. The robot must navigate around these obstructions despite not knowing their shape or location. Without tactile sensors, the robot only observes that a contact occurred somewhere on its surface, a measurement containing very little information. We present the Collision Hypothesis Sets representation for computing a belief of occupancy from these observations, and we introduce a planning and control architecture that uses this representation to navigate through unknown environments. Despite the dearth of information, we demonstrate through experiments that our algorithms can navigate around unseen obstacles and into narrow passages. We test in multiple environments in simulation and on a physical robot arm both with and without the aid of a 2.5D depth sensor. Compared to a baseline representation Collision Hypothesis Sets produce an approximately 1.5-3x speed-up and improve the success rate from 40-60% to 100% in tested scenarios with narrow passages.

1 Introduction

Robots rely on sensors to construct models of the world for use in motion planning, but in many practical scenarios sensing limitations result in an incomplete or inaccurate model, resulting in plans that can collide with unobserved obstacles. Sensing limitations occur in manipulation tasks due to limited range and field of view, invalid measurements caused by glare, and insufficient accuracy for motion in tight areas. Furthermore, robots may reach into occluded areas during maintenance and assembly tasks (e.g. reaching into a car engine) and household tasks (e.g. reaching deep into a cabinet or behind a box). In the scenarios examined in this paper collisions between the arm and environment can be sensed without damage to the robot and used to inform future plans, but a lack of tactile sensing generates large non-Gaussian uncertainty over the belief of the occupancy of the environment.

The task we consider is to move from a given start configuration through free space to a goal configuration as quickly as possible, however the free space is not known *a priori* and the occupancy must be sensed through contact. We do not require tactile sensing to detect contact since most robots do not possess touch-sensitive skin. Even robots with such skin may pick up objects, effectively extending their kinematic chain with unsensorized geometry. We assume a robot is able to detect contact using joint torque feedback as is available on many

This research was funded in part by Toyota Research Institute (TRI). This article solely reflects the opinions of its authors and not TRI or any other Toyota entity.

robot arms. We further use that joint torque to determine which links may be in contact.

The task of moving to a goal in an unknown environment can be framed as a Partially Observable Markov Decision Process (POMDP), where the belief over occupancy is obtained through noisy collision measurements. In our implementation the workspace, observed collisions, and known obstacles are all stored in voxel grids of size N, thus the size of a belief state is 2^N. A measurement either indicates a configuration along a path did not collide and thus the voxels occupied by the robot do not contain obstacles, or that a collision occurred and thus at least one voxel blocked movement to the new configuration. Measurement uncertainty does not primarily come from sensor noise, but because a measurement only provides a set of voxels where at least one is occupied. The size of the belief space and the measurement uncertainty make this problem intractable for a standard POMDP solver, thus we propose an approach which is specialized to our domain.

Our key contribution is a representation for contact uncertainty we call *Collision Hypothesis Sets (CHS)*, which enables planners to more accurately reason about potential collisions. This paper first reviews prior work on planning in uncertain environments (Sec. 2). The task is then defined (Sec. 3), the CHS representation is motivated and presented (Sec. 4), and the planning and control architecture is then described (Sec. 5). Results are reported for experiments performed in simulation and on a physical robot, comparing the CHS representation to a baseline unified cost grid (Sec. 6). With enough open free space the two methods perform similarly. However, when the robot must enter narrow passages to reach a goal the CHS representation reduced the total execution time by approximately a factor of 1.5 to 3. Furthermore, in narrow passage scenarios where the baseline method only reached the goal in approximately 40-60% of trials, using CHSs produced a success rate of 100%.

2 Related Work

A core component of motion planning is the ability to check the validity of a path. Many motion planning algorithms assume deterministic collision checking but with uncertainty in the robot or environment it is only possible to calculate the probability of collision. With an arbitrary belief Monte Carlo simulations (MCS) can be used to estimate collision probabilities [14], although this approach is computationally expensive and generally not practical for large state spaces, so many approaches apply only to specific belief distributions.

Assuming Gaussian uncertainty over the robot and specific objects enables computation of collision probability for a single configuration [4, 20, 21]. To estimate the collision probability along a full path, rather than at an individual configuration, [17] propagates only the portion of a Gaussian robot belief that are not in collision. Other forms of uncertainty, such as point cloud measurements from range sensors, may be better modeled with specifically-tailored distributions dependent on distance and incidence angle [1]. However, none of these distributions accurately model the information obtained when a robot contacts the environment.

Localization methods exist that are specifically tailored to model the unusual contact sensing uncertainty. Using a particle filter, a belief consistent with a contact measurement can be updated using rejection sampling [18], or sampling from the contact manifold [11]. Both of these methods require models of objects in the world, which we do not assume are known. Other methods use joint position and torques to estimate the contact location on the robot surface [3,12], but require accurate torque measurements to produce accurate estimates. Since we wish to sense contact just above the noise threshold the torque measurements will contain significant noise, thus we use a comparatively simple method that more reliably estimates which links may be in contact.

By treating the probability of collision as a cost, the problem of planning under uncertainty can be framed as an optimal path planning problem. While there exist numerous optimal planners through continuous space, such as RRT* [9] and its many variants, these methods rely on a quick computation of path cost to run efficiently and since computing collision probability is far more expensive than a typical path length cost these methods ultimately explore too few nodes for our problem in a reasonable time.

Rather than attempting to minimize the probability of collision, many algorithms search for a path with some acceptably-low probability of collision. This can be done conservatively by inflating the robot [15], iteratively considering simplified dynamics [5], or in a Probabilistic Roadmap where each edge collision cost is bounded [6]. These methods approximate the full path probability of collision from the probability of collision of the individual states, thereby assuming that probabilities of collision are independent. In our work the occupancy uncertainty is heavily coupled across space, thus this independence assumption does not hold (see Sec. 5 for further discussion).

While executing a plan a robot may learn new information that causes the plan to become invalid or suboptimal, and the robot can replan a new path given this new information. Most prior work on replanning assumes collisions can be sensed at a distance and a primary goal is to avoid collisions [8]. Some methods do allow for collisions and store contact locations sensed using tactile skin on a robotic arm [2, 10]. Using a simulated skin a fast planning architecture was demonstrated with a fast local controller that falls back to a slower sampling-based planner when stuck in a local minimum [16]. We use a similar approach for mixing planning and control, but our method does not require tactile skin to sense precise contact points.

3 Problem Statement

Given a robot with configuration space $\mathcal{C}$, define a workspace voxel grid of size $N \times N \times N$ as $\mathcal{W}$ with static workspace occupied voxels $\mathcal{W}_\mathcal{O} \subseteq \mathcal{W}$ and free space $\mathcal{W}_\mathcal{F} = \mathcal{W} \setminus \mathcal{W}_\mathcal{O}$. For any configuration $q \in \mathcal{C}$ the robot occupies a subspace of workspace, defined by the mapping $\mathcal{R}(q) : \mathcal{C} \to \mathbb{P}(\mathcal{W})$ where $\mathbb{P}$ denotes the powerset. A collision occurs when $\mathcal{R}(q) \cap \mathcal{W}_\mathcal{O} \neq \emptyset$, and is detected by joint torque feedback (discussed in Sec. 4). While the planner does not have access to $\mathcal{W}_\mathcal{O}$ directly, the swept volume of the previously visited configurations $q^i_{visited}$ is known to be free and is stored in $\mathcal{W}_{SV} = \cup_i \mathcal{R}(q^i_{visited}) \subseteq \mathcal{W}_\mathcal{F}$.

A discretized path ξ consists of configurations q_i such that $||q_{i+1} - q_i|| < \delta$ where δ is set based on the desired resolution. A robot at q_i attempting to follow the ith path ξ_i^{plan} takes execution time $T_{exec}(\xi_i^{plan}) > 0$ to arrive at

$$M(q_i, \xi_i^{plan}) = q_{reached} \tag{1}$$

If ξ_i^{plan} collides then $q_{reached} \in \xi_i^{plan}$ is the configuration directly before the first $q \in \xi_i^{plan}$ in collision. After a collision a new ξ_{i+1}^{plan} may be executed.

Define an algorithm $\mathcal{P}$ that takes time t_i^{plan} to produce ξ_i^{plan}, a path that must begin at q_i. $\mathcal{P}$ is aware of past planned paths and visited configurations $Q_i^{visited} = \{q_0, q_1, ..., q_i\}$. We refer to $\mathcal{P}$ as a "planner" if it plans ξ^{plan} reaching the goal, and a "controller" if it computes a short ξ^{plan} towards the goal. A single $\mathcal{P}$ may choose to act as either a planner or controller depending on context.

$$\mathcal{P}(Q_i^{visited}, \xi_1^{plan}, \ldots, \xi_{i-1}^{plan}) = (\xi_i^{plan}, t_i^{plan}) \tag{2}$$

Given a robot in configuration q_{init} and a set of goal configurations Q_{goal} our objective is to choose a $\mathcal{P}$ to reach a goal configuration in the least time.

$$\underset{\mathcal{P},n}{\text{minimize}} \quad \sum_{i=0}^{n-1} t_i^{plan} + T_{exec}(\xi_i^{plan}) \tag{3}$$

$$\text{subject to:} \quad q_0 = q_{init}, q_n \in Q_{goal} \tag{4}$$

$$(\xi_i^{plan}, t_i^{plan}) = \mathcal{P}(Q_i^{visited}, \xi_1^{plan}, \ldots, \xi_{i-1}^{plan}) \tag{5}$$

$$q_{i+1} = M(q_i, \xi_i^{plan}) \tag{6}$$

Since planning time is part of the objective, $\mathcal{P}$ must choose a tradeoff between analyzing all information to choose the best ξ_i^{plan}, and minimizing planning time t_i^{plan}. A controller typically achieves a small t_i^{plan}, while a planner spends more time to produce better ξ_i^{plan}. As ξ^{plan} are executed, $\mathcal{P}$ has access to more knowledge about $\mathcal{W}$, and effectively using this information is key to minimizing total time. We will not solve Eq. 3 computationally, but instead design a $\mathcal{P}$ with desirable qualities and justify our choices with experimental trials.

4 Representing Uncertain Contact Information

When a collision is detected during execution the swept volume of a set of configurations $q_{collision}^i$ along the robot path within some small distance $d_{\mathcal{K}}$ after collision is assumed to contain the point of contact. While in theory the set of points on the robot surface contains the contact point, in practice joint measurement uncertainties, robot geometry uncertainties and approximations, and material compliance require a larger set to guarantee encapsulation of the contact point. A set containing the contact point is constructed $\mathcal{K}_i = \cup_i \mathcal{R}(q_{collision}^i) \setminus \mathcal{W}_{SV}$, the total volume of the robot in the possible collision configurations with the known free space removed (Fig. 1).

In our problem collisions are detected using measured joint torque $\tau_{meas} \in \mathbb{R}^J$, where J is the number of robot joints. Using a mass model of the robot the

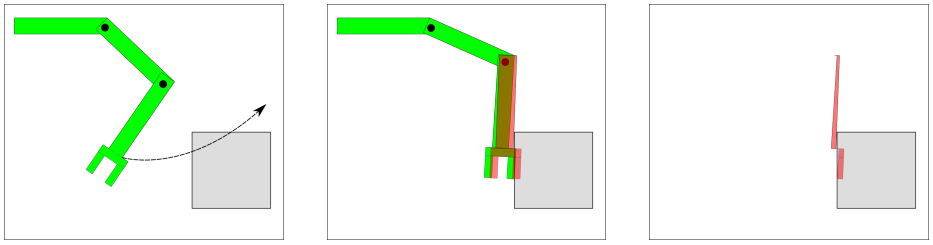


Fig. 1: A plan for the green robot (left) results in a collision with the grey unknown obstacle. A red collision hypothesis set is added using links possibly in collision based on measured joint torques (center). The known free space is removed (right).

expected joint torque due to gravity and dynamics τ^{exp} is calculated and used to estimate the external joint torque $\tau^{ext} = \tau^{meas} - \tau^{exp}$. A noise threshold τ^{th} is set for each joint and τ^{ext} triggers a collision detection whenever any joint exceeds its threshold. Joint i exceeding τ_i^{th} implies an external (contact) force on a link after joint i on the kinematic chain. A set of links that must contain a contact $\mathcal{L}_{contact}$ is constructed by first finding the highest i where $\tau_i^{ext} > \tau_i^{th}$, then adding all links downstream from joint i to $\mathcal{L}_{contact}$. Only the links in $\mathcal{L}_{contact}$ are used to create $\mathcal{K}_i$.

4.1 Baseline: Unified Cost Grid

When provided with noisy measurements a common approach to modelling uncertainty extends a binary occupancy map to a Unified Cost Grid (UCG), a voxel grid where each voxel stores its likelihood of occupancy. We compute this likelihood as the count of how many observed collisions could be explained if that voxel were occupied. The UCG representation computes a path cost by summing the likelihood of the voxels in the swept volume of the path.

Unfortunately, by combining all measurements into a single grid the UCG representation loses the information that each collision was caused by at least one occupied voxel. As we show in Section 6, this approach fails when entering narrow passages as collisions near the entrance create a high cost for feasible paths. Therefore, we construct a representation that maintains this information.

4.2 Collision Hypothesis Sets

Define a *Collision Hypothesis Set* (CHS) as a set of points in the robot workspace containing at least one point in collision. The $\mathcal{K}_i$ constructed after a collision are CHSs, since $\mathcal{K}_i \cap \mathcal{W}_\mathcal{O} \neq \emptyset$. However, unlike in UCG, the CHS representation never combines $\mathcal{K}_i$s into a unified grid, and instead maintains a set $\mathcal{K}$ of all generated $\mathcal{K}_i$.

Planning will require evaluating the probability of collision for a path using $\mathcal{K}$. Let the swept volume of a path ξ on the robot be $\mathcal{W}_\xi \subseteq \mathcal{W}$. We define the probability of collision of ξ with a single $\mathcal{K}_i$ as

$$p_{collision}(\mathcal{W}_\xi, \mathcal{K}_i) = \frac{|\mathcal{W}_\xi \cap \mathcal{K}_i|}{|\mathcal{K}_i|} \tag{7}$$

Assuming there exists a single occupied voxel uniform randomly selected from $\mathcal{K}_i$, this is precisely the probability the path collides. While this will likely be an underestimate of the true probability, it encourages exploration and further collision measurements will help localize the contact. The probability of collision for a full path is computed assuming independence between $\mathcal{K}_i$s, thus

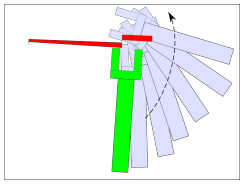
$$p_{collision}(\mathcal{W}_\xi, \mathcal{K}) = 1 - \prod_i (1 - p_{collision}(\mathcal{W}_\xi, \mathcal{K}_i)) \tag{8}$$

This definition for collision probability captures several key features that the UCG representation lacks. A $\mathcal{K}_i$ with fewer voxels represents a more precise knowledge of where the contact occurred and thus a more precise estimate of workspace occupancy. In addition, a path that moves the robot through an entire CHS is guaranteed to collide, since $\frac{|\mathcal{W}_\xi \cap \mathcal{K}_i|}{|\mathcal{K}_i|} = 1$, representing the information that a CHS contains at least one point in collision. A path ξ that is attempted but blocked due to a detected collision creates a $\mathcal{K}_i$ that lies entirely within $\mathcal{W}_\xi$, so the updated collision probability for ξ will now be 1.

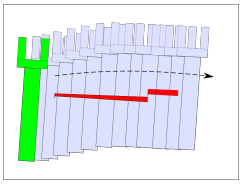
5 Interleaving Planning and Control

To achieve our objective of reaching a goal configuration in minimal time (Eq. 3), we must choose an algorithm $\mathcal{P}$ that compute good motions ξ_i^{plan} in low planning times t_i^{plan}. Local controllers quickly compute locally good ξ_i^{plan} but may get stuck in local minima. Global planning can escape these minima by planning a full path, but requires significant computation time. Our environments include many undetected obstacles and a local controller may produce many collisions before getting stuck, thereby providing more information to the global planner without adding much total time. Thus we use a planner initially and when stuck in cul-de-sacs, and local control otherwise. Our full architecture is presented in Algorithm 1.

Planning: The objective of the planner is to find a path to the goal with the minimal probability of collision. Unfortunately, as discussed in Section 2, previous methods that plan over obstacle uncertainty are not applicable when using CHSs, as these planners typically rely on a path cost definition that is purely the sum [9] or maximum [20] of costs of states/edges. Figure 2 illustrates two problems with inferring path collision probability using the costs of only individual states along the path. In Fig. 2a the swept volume of adjacent states overlap significantly, thus summing costs of states could significantly overestimate the probability of collision. In Fig. 2b multiple states collectively intersect the entirety of a CHS thus guaranteeing a collision, but each state individually only intersects a fraction of that CHS, thus taking the maximum over all state costs would significantly underestimate the probability of collision.



(a)



(b)

Fig. 2: Plans for the green arm sweep through the blue region. Red: a CHS

Algorithm 1
MainLoop(q_{cur}, q_{goal})

1: $\mathcal{K} \leftarrow \emptyset$; $\mathcal{W}_{SV} \leftarrow \emptyset$
2: **while** $q_{cur} \neq q_{goal}$ **do**
3: $\xi \leftarrow$ Planner($q_{cur}, q_{goal}, \mathcal{K}$)
4: $q_{cur} \leftarrow$AttemptPath($\xi, \mathcal{K}, \mathcal{W}_{SV}$)
5: **while** $q_{cur} \neq q_{goal}$ **and** ($\xi \leftarrow$ Controller($q_{cur}, q_{goal}, \mathcal{K}$)) $\neq \emptyset$ **do**
6: $q_{cur} \leftarrow$ AttemptPath($\xi, \mathcal{K}, \mathcal{W}_{SV}$)

Algorithm 2
AttemptPath($\xi, \mathcal{K}, \mathcal{W}_{SV}$)

1: **for** q_i in ξ **do**
2: **if** q_i causes collision **then**
3: $\mathcal{K}$.addNew($q_i, \xi, d_{\mathcal{K}}$)
4: **break**
5: $q_{cur} \leftarrow q_i$
6: $\mathcal{W}_{SV} \leftarrow \mathcal{W}_{SV} \cup \mathcal{R}(q_{cur})$
7: $\mathcal{K}$.subtract($\mathcal{W}_{SV}$)
8: **return** q_{cur}

To plan a path we create PathBiRRT (Alg. 3), a planner based on bi-directional RRT [13] that ensures the cost of the path generated is below a specified threshold p_{thr}. For the CHS representation the `Cost` function is given by Eq. 8. For the baseline UCG representation, which we compare to in the results, the `Cost` sums the cost of all voxels in $\mathcal{W}_\xi$, i.e. $\sum_i |\mathcal{W}_\xi \cap \mathcal{K}_i|$.

Ideally we desire a `Connect` function that ensures the cost of the entire path to any new node is below p_{thr}. In practice, computing this cost for every explored node is prohibitively expensive, thus we approximate this cost as an accumulation of branch costs computed in the `Connect` function (Alg. 4). When extending towards q_{target} the full path cost of the branch from q_{near} is calculated (Line 6). The cost from the root to the q_{new} is approximated by accumulating the approximate cost to q_{near} (computed previously) and the cost of the new branch (Line 8). When using CHS the `Accumulate` function is the combination of independent probabilities: $1-(1-p_1)(1-p_2)$. UCG accumulates by adding the costs: $c_1 + c_2$. By computing the cost over full branches this approximation is significantly better than accumulating cost purely based on states, however, this approximation may still over or underestimate the true cost, as separate branches along a path may overlap in $\mathcal{W}$.

PathBiRRT (Alg. 3) repeatedly calls `Connect` to build a tree from the start and a tree from the goal, generating a potential path ξ when the two trees meet. ξ may exceed p_{thr} due to the approximation error within `Connect` and because ξ is the combination of paths from two trees. The cost of ξ is checked (Line 9) and if it exceeds p_{thr} then the highest cost edge from ξ is pruned along with all child edges, and planning continues.

For planning within fixed time t_{plan}, we provide two methods for setting p_{thr}. The anytime APathBiRRT (Alg. 5) begins with $p_{thr} = \infty$ and continues searching for lower-cost paths until time runs out. In contrast, IPathBiRRT (Alg. 6) begins with an optimistic $p_{thr} = c_{init}$. In each iteration IPathBiRRT allocates a fraction ψ_f of the remaining time t_ψ for planning with the current p_{thr}. If a plan is not found within t_ψ, p_{thr} is increased for the next iteration, approaching p_{max} for CHS, or v_{max} for UCG. Once a path is found, IPathBiRRT then invokes APathBiRRT for the remainder of the planning time.

As a benchmark, we also implemented ABiRRT, which iteratively decreases a cost threshold (as in Alg. 5), but when checking to add a new node (as in Alg.

4, Line 5-9) only the configuration cost is considered ($\texttt{Cost}(\mathcal{R}(q_{new}), \mathcal{K}) < p_{thr}$) and there is no full path check (Alg. 3, Line 9). To support the claim that asymptotically optimal planners are not practical for this problem we also compare against RRT* [9], which always computes the full path cost when considering new connections.

Local Control: The local controller samples a specified number n_c of straight-line motions of length d_c uniformly from the half-sphere in $\mathcal{C}$-space that reduce the robot's distance to the goal. From these samples, the controller greedily selects the motion with lowest probability of collision, using Eq. 8. If no motion is found with probability of collision $< p_c$ for CHS, or cost $<$ nvox$_c$ for UCG, the controller assumes it is stuck (Alg. 1 Line. 5) in a cul-de-sac and invokes the planner.

Algorithm 3
PathBiRRT($q_{init}, q_{goal}, \mathcal{K}, p_{thr}, t$)

1: $\mathcal{T}_A$.init(q_{init}); $\mathcal{T}_A[q_{init}]$.approxCost $\leftarrow 0$
2: $\mathcal{T}_B$.init(q_{goal}); $\mathcal{T}_B[q_{goal}]$.approxCost $\leftarrow 0$
3: **while** timeElapsed() $< t$ **do**
4: $q_r \leftarrow$ sampleConfig()
5: status, q_{new} = Connect($\mathcal{T}_A, q_r, p_{thr}$)
6: **if** status $\neq$ *Trapped* **then**
7: **if** Connect($\mathcal{T}_B, q_{new}, p_{thr}$) = *Reached* **then**
8: $\xi \leftarrow$ path($\mathcal{T}_A, \mathcal{T}_B$)
9: **if** Cost($\mathcal{W}_\xi, \mathcal{K}$) $< p_{thr}$ **then**
10: **return** ξ
11: **else**
12: e $\leftarrow$ highestCostEdge(ξ)
13: **if** e in $\mathcal{T}_A$ **then**
14: $\mathcal{T}_A$.prune(e)
15: **else**
16: $\mathcal{T}_B$.prune(e)
17: swap($\mathcal{T}_A, \mathcal{T}_B$)
18: **return** $\emptyset$

Algorithm 4
Connect($\mathcal{T}, q_{target}, \mathcal{K}, p_{thr}$)

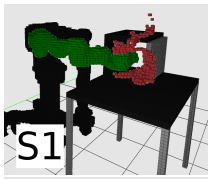
1: $q_{near} \leftarrow$ nearest($\mathcal{T}, q_{target}$)
2: $\mathcal{W}_{seg} \leftarrow \{\}$
3: $\xi \leftarrow$interpolate($q_{near}, q_{target}, \delta$)
4: **for** q_{new} in ξ **do**
5: $\mathcal{W}_{seg} \leftarrow \mathcal{W}_{seg} \cup \mathcal{R}(q_{new})$
6: $c_{seg} \leftarrow$ Cost($\mathcal{W}_{seg}, \mathcal{K}$)
7: $c_{near} \leftarrow \mathcal{T}[q_{near}]$.approxCost
8: $c_{approx} \leftarrow$ Accum(c_{seg}, c_{near})
9: **if** collides(q_{new}) **or** $c_{approx} \geq p_{thr}$ **then**
10: **if** $q_{new} = q_{near}$ **then**
11: **return** {Trapped, q_{new}}
12: **return** {Advanced, q_{new}}
13: $\mathcal{T}$.add(q_{new})
14: $\mathcal{T}[q_{new}]$.approxCost $\leftarrow c_{approx}$
15: **return** {Reached, q_{new}}

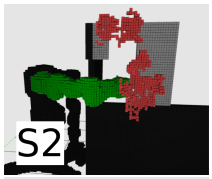
Algorithm 5 APathBiRRT
($q_{init}, q_{goal}, \mathcal{K}, p_{thr} = \infty$)

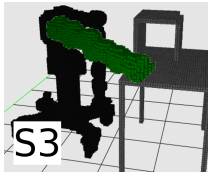
1: $\xi_{best} \leftarrow \emptyset$
2: **while** timeRemaining()>0 **do**
3: $\xi \leftarrow$ PathBiRRT(q_{init}, q_{goal}, p_{thr}, timeRemaining())
4: **if** $\xi \neq \emptyset$ **then**
5: **if** Cost($\mathcal{W}_\xi, \mathcal{K}$) = 0 **then**
6: **return** ξ
7: $p_{thr} \leftarrow$ Cost($\mathcal{W}_\xi, \mathcal{K}$) - ϵ
8: $\xi_{best} \leftarrow \xi$
9: **return** ξ_{best}

Algorithm 6 IPathBiRRT
($q_{init}, q_{goal}, \mathcal{K}, c_{init}, c_{max}, \psi_f$)

1: $p_{thr} \leftarrow c_{init}$
2: **while** timeRemaining()>0 **do**
3: $t_\psi \leftarrow$ timeRemaining() $\cdot \psi_f$
4: $\xi \leftarrow$ PathBiRRT($q_s, q_{goal}, p_{thr}, t_\psi$)
5: **if** $\xi \neq \emptyset$ **then**
6: **return** APathBiRRT(q_s, q_{goal}, p_{thr} =Cost($\mathcal{W}_\xi, \mathcal{K}$))
7: $\alpha \leftarrow$ timeElapsed()/totalTime()
8: $p_{thr} \leftarrow \alpha \cdot c_{max} + c_{init}$
9: **return** $\emptyset$







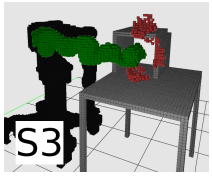


Fig. 3: Simulated Scenarios

Fig. 4: Physical Robot environments

6 Experiments and Results

To demonstrate the advantages of our representation we compared Collision Hypothesis Sets (CHS) to the baseline Unified Cost Grid (UCG) in multiple environments in simulation and on a physical robot using multiple planning approaches[1]. Parameters used in experiments are given in Table 5. Voxel grids were implemented on the GPU using GpuVoxles [7]. Planners were implemented using OMPL [19] with modification. Code was run on a computer with i7-7700 processor and a NVidia 1080 Ti GPU. For all planners each path was smoothed using 100 iterations of shortcut smoothing.

Simulation Experiments simulated a 7DOF Kuka iiwa arm in the environments in Fig. 3, shown with red $\mathcal{K}_i$s, grey unobserved obstacles, and black observed obstacles. Rather than simulating joint torque, collisions were determined when the simulated robot moved into an obstacle in the workspace. All links downstream of the true link in collision were used to generate a CHS. In scenario S1, the simplest environment, the robot's goal was to reach inside a box located on a table with some occupancy known from a simulated depth sensor. Scenario S2 was harder as the robot needed to move the entire arm through a narrow slot occluded from the sensor. Scenario S3 was identical to S1 except the robot used no depth sensor information. Each simulation trial allowed 15 minutes for the robot to reach the goal.

Physical Robot Experiments were conducted on a physical 7DOF Kuka iiwa arm capable of sensing joint torque. A Kinect depth sensor created known obstacle occupancy. Figure 4 shows the physical robot experimental setups. Each physical trial allowed 5 minutes for the robot to reach the goal. In physical robot scenario R1 the robot placed a pitcher inside a box with the lid occluding the side and back walls from the Kinect. R2 involved placing a cylindrical can on a short shelf. Glare and occlusions resulted in a sparse and noisy occupancy map from the Kinect, both blocking a feasible path to the goal and missing portions of the top and bottom of the shelf. To accommodate noise up 30 intersections were allowed between the robot and the Kinect occupancy map.

[1] Videos of experiments available at https://www.youtube.com/watch?v=EjCq1Q4nNUc

	Description	Value
p_c	controller max collision probability	0.9
nvox_c	controller max collision voxels	50
d_c	controller motion length	0.3 radians
n_c	controller number of samples	20
δ	path discretization size	0.14 radians
$d_{\mathcal{K}}$	dist. for $\mathcal{K}_i$ creation	0.05 radians
τ^{th}	torque threshold	[20, 20, 15, 5, 4, 3, 1] Nm
t_{plan}	allowed planning time per iteration	30s
N	voxel grid size	200x200x200
ψ_f	IPathBiRRT phase fraction	1/4
c_{init}	IPathBiRRT initial cost	0.3
p_{max}	IPathBiRRT CHS c_{max}	1
v_{max}	IPathBiRRT UCG c_{max}	400
ϵ	APathBiRRT improvement factor	0.0001

Fig. 5: Experimental parameters

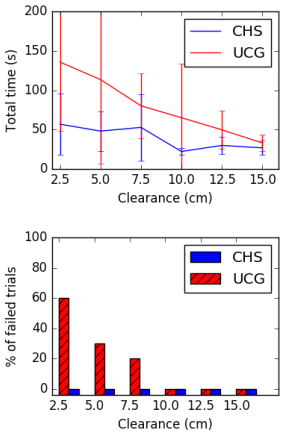


Fig. 6: Total time and failure % for R4, averaged over 10 trials for each clearance with a 5 min. timeout

In scenario R3 the robot arm moved from below to above a table. The sensed table was artificially shifted 5 cm away from the robot, simulating localization or sensor error. R4 tested the behavior moving through a narrow passage between two tables. This gap was adjusted between 15.5cm to 28cm, corresponding to a clearance of 2.5cm to 15cm for the 13cm wide robot hand. In R4 Kinect data was not used, thus the robot only sensed obstacles through contact.

Results: Table 1 reports the results for each simulated scenario, comparing CHS and UCG using the proposed APathBiRRT and IPathBiRRT planners as well as RRT* and ABiRRT. RRT* did not reach the goal within the time limit in any scenario. ABiRRT performed far worse than our proposed methods, indicating that considering full path collision probability is superior to only per-configuration collision probability. Table 2 reports the results for each physical scenario, comparing CHS and UCG using IPathBiRRT only, as this planner performed the best in simulation trials. In both simulation and physical trials we observed our CHS formulation outperforms the UCG approach in terms of computation time.

In simpler scenarios (S1, S2, R1) early collisions were navigated better by the local controller, causing fewer lengthy planning iterations. For example in R1 our method required invoking the global planner only once in all ten trials, leading to an average success time of 18.6s compared to 76.6s when using a unified cost grid. In harder scenarios (S3, R2) collisions occurred on multiple sides of narrow passages. Using a unified cost grid, the planner and controller avoided the center of the passage as this area has cost accumulated from multiple collisions, thus only 40% of the trials successfully reached the goal for R2. Paths through the center do not intersect with all voxels from any CHS, thus our approach correctly identified possible paths and succeeded in 100% of trials in R2. Even

	Planner	S1			S2			S3		
		Succ.	Time	P.Calls	Succ.	Time	P.Calls	Succ.	Time	P.Calls
CHS	APathBiRRT	100%	110	4.1	100%	100	3.4	100%	220	6.5
	IPathBiRRT	85%	69	3.0	100%	64	2.6	100%	180	5.3
	RRT*	0%	-	-	0%	-	-	0%	-	-
	ABiRRT	100%	78	3.5	30%	230	20	60%	200	15
UCG	APathBiRRT	100%	110	4.2	95%	120	3.9	40%	440	18
	IPathBiRRT	55%	71	2.9	90%	130	3.9	60%	280	15
	RRT*	0%	-	-	0%	-	-	0%	-	-
	ABiRRT	100%	113	4.4	80%	250	11	65%	500	22

Table 1: Simulated Scenarios: Successes within 15 min, total time (s), and number of planner calls averaged over 20 trials for each entry. Blue: Proposed methods.

	Planner	R1					R2				
		Succ.	Time	P.Calls	Plan	Ctrl	Succ.	Time	P.Calls	Plan	Ctrl
CHS	IPathBiRRT	100%	19	1.1	3.4	15	100%	62	2.1	38	24
UCG	IPathBiRRT	100%	77	2.3	57	19	40%	180	4.8	130	43
	Planner	R3					R4: clearance=2.5cm				
		Succ.	Time	P.Calls	Plan	Ctrl	Succ.	Time	P.Calls	Plan	Ctrl
CHS	IPathBiRRT	100%	100	1.0	32	65	100%	57	1.2	1.5	50
UCG	IPathBiRRT	100%	100	1.0	34	65	40%	140	2.5	50	82

Table 2: Physical Robot Experiments: Successes within 5 min., total time (s), number of planner calls, planning time (s), and controller time (s) averaged over 10 trials.

considering only trials where UCG succeeded our approach still reached the goal in approximately one third of the time on average.

In R3 the table obstacle created a cul-de-sac for the local controller, though there was significant free space for the planner to conservatively avoid the table. Since plans were able to avoid much of the CHSs, accurately modeling collision probability was less important, thus the baseline method and proposed method performed similarly.

In R4 with a large gap between tables both methods found a path quickly. For both methods, successful trials primarily invoked the local controller and rarely needed the global planner. As the gap narrowed both methods required more time to find the opening, however UCG took longer on average and in 60% of trials did not find the opening within the allowed time of 5 minutes (Fig. 6).

7 Conclusions and Future Work

Most robots do not have touch-sensitive skin, and those that do may manipulate unsensorized objects. The proposed Collision Hypothesis Sets allow reasoning about the knowledge gained when these robots and objects come into contact with the environment. We showed how collision hypothesis sets can be used in controllers and planners to search for paths with minimum probability of collision. We performed simulated and physical robot experiments and found for simpler environments our approach takes less time to reach the goal while for more complex environments our methods succeeds where other approaches fail.

We explained why many existing methods for planning under uncertainty cannot be applied to contact observations and presented two planners that first

approximate, then compute the full path cost. However, the planners implemented could be made more efficient and the resulting paths are often significantly suboptimal. Our future work seeks to improve these results.

References

1. K. Bae, D. Belton, and D. D. Lichti. A closed-form expression of the positional uncertainty for 3D point clouds. *TPAMI*, 2009.
2. T. Bhattacharjee, P. Grice, A. Kapusta, M. Killpack, D. Park, and C. Kemp. A robotic system for reaching in dense clutter that integrates model predictive control, learning, haptic mapping, and planning. *IROS*, 2014.
3. A. Bicchi, J. K. Salisbury, and D. Brock. Contact sensing from force measurements. *IJRR*, 12(3):249–262, 1993.
4. L. Blackmore. A probabilistic particle control approach to optimal , robust predictive control. In *AIAA*, 2006.
5. A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *ICRA*, 2011.
6. L. Guibas, D. Hsu, H. Kurniawati, and E. Rehman. Bounded uncertainty roadmaps for path planning. In *WAFR*, 2008.
7. A. Hermann, F. Drews, J. Bauer, S. Klemm, A. Roennau, and R. Dillmann. Unified GPU voxel collision detection for mobile manipulation planning. *IROS*, 2014.
8. Lucas Janson, Tommy Hu, and Marco Pavone. Safe motion planning in unknown environments: Optimality benchmarks and tractable policies. *CoRR*, 2018.
9. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
10. M. Killpack, A. Kapusta, and C. Kemp. Model predictive control for fast reaching in clutter. *Autonomous Robots*, 40(3):537–560, Mar 2016.
11. M. Klingensmith, M. Koval, S. Srinivasa, N. Pollard, and M. Kaess. The manifold particle filter for state estimation on high-dimensional manifolds. *CoRR*, 2016.
12. G. Koonjul, G. Zeglin, and N. Pollard. Measuring contact points from displacements with a compliant, articulated robot hand. In *ICRA*, 2011.
13. J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. *ICRA*, 2000.
14. A. Lambert, D. Gruyer, and G. Saint Pierre. A fast monte carlo algorithm for collision probability estimation. In *ICARCV*, Dec 2008.
15. A. Lee, Y. Duan, S. Patil, J. Schulman, Z. McCarthy, J. van den Berg, K. Goldberg, and P. Abbeel. Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. *IROS*, 2013.
16. D. Park, A. Kapusta, J. Hawke, and C. Kemp. Interleaving planning and control for efficient haptically-guided reaching in unknown environments. *Humanoids*, 2014.
17. S. Patil, J. van den Berg, and R. Alterovitz. Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In *ICRA*, 2012.
18. B. Saund, S. Chen, and R. Simmons. Touch based localization of parts for high precision manufacturing. *ICRA*, 2017.
19. I. Şucan, M. Moll, and L. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19, 2012. http://ompl.kavrakilab.org.
20. N. E. Du Toit and J. W. Burdick. Probabilistic collision checking with chance constraints. *IEEE T-RO*, 27(4):809–815, Aug 2011.
21. J. van den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha. LQG-Obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty. *ICRA*, 2012.