

Motion Planning for Collision-resilient Mobile Robots in Obstacle-cluttered Unknown Environments with Risk Reward Trade-offs

Zhouyu Lu, Zhichao Liu, Gustavo J. Correa, and Konstantinos Karydis

Abstract— Collision avoidance in unknown obstacle-cluttered environments may not always be feasible. This paper focuses on an emerging paradigm shift in which potential collisions with the environment can be harnessed instead of being avoided altogether. To this end, we introduce a new sampling-based online planning algorithm that can explicitly handle the risk of colliding with the environment and can switch between collision avoidance and collision exploitation. Central to the planner’s capabilities is a novel joint optimization function that evaluates the effect of possible collisions using a reflection model. This way, the planner can make deliberate decisions to collide with the environment if such collision is expected to help the robot make progress toward its goal. To make the algorithm online, we present a state expansion pruning technique that significantly reduces the search space while ensuring completeness. The proposed algorithm is evaluated experimentally with a built-in-house holonomic wheeled robot that can withstand collisions. We perform an extensive parametric study to investigate trade-offs between (user-tuned) levels of risk, deliberate collision decision making, and trajectory statistics such as time to reach the goal and path length.

I. INTRODUCTION

Achieving high mission performance in face of risk is a common trade-off in robot motion planning [1]. The trade-off becomes increasingly important for planning and navigation of robots with noisy actuation and sensing (such as small and inexpensive robots), especially when there is need to deploy them rapidly into dangerous or otherwise not easily accessible areas. A key source of risk includes possible collisions when navigating in an unknown environment.

Autonomous navigation in unknown environments can impact several applications ranging from surveying and inspection to search and rescue [2], and has thus received significant attention. Conventional planning and navigation strategies have used geometric representations of boundaries to define collision free paths with convergence guarantees [3]. In partially-known environments, planning with local information based on instantaneous sensor data for collision avoidance is appealing for computationally-constrained platforms requiring high-rate collision avoidance [4]. Three main categories exist for local collision avoidance planning algorithms [5], which include planning by reacting to sensor data [6], map-based planning using sensors to build maps or

The authors are with the Dept. of Electrical and Computer Engineering, University of California, Riverside. Email: {zlu044, zliu157, gcorr003, karydis}@ucr.edu. We gratefully acknowledge the support of NSF under grant #IIS-1910087, ONR under grants #N00014-18-1-2252 and #N00014-19-1-2264, ARL under grant #W911NF-18-1-0266, and UCR Office of Research and Economic Development under a Collaborative Seed Grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

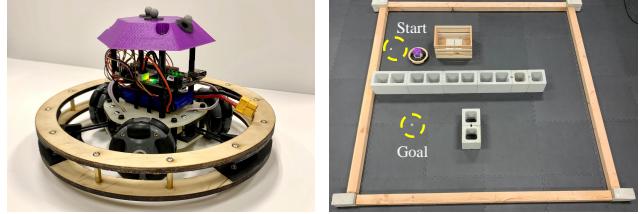


Fig. 1: This study’s holonomic wheeled robot (left) navigating in a confined corridor environment populated with obstacles (right) while harnessing collisions. A supplementary video showing instances of the experiments can be found in <https://www.youtube.com/watch?v=S3oYebJRFaO>.

using a priori known global maps [7], [8], and planning to maximize exploration coverage [9], [10].

However, as robots increasingly venture outside the protected lab environment and into the real—uncertain—world, guaranteeing collision avoidance becomes an even more challenging task [11], [12]. If a conservative local planner is employed to avoid collisions in cluttered environments, the robot may not be able to find a feasible path to the goal even if one exists. At the same time, recent advances in material science and mechanical design have helped introduce robots that can safely withstand collisions (e.g., small legged robots with exoskeletons [13], aerial robots with protective cages [14], and soft robots [15]).

Taken together, these observations may explain an emerging paradigm shift: *Collisions with the environment could be harnessed instead of being avoided*. In fact, it has been demonstrated how allowing for collisions can benefit motion planning and control [16], localization [17], sensing [18], and robot agility in terms of rapidly changing direction of motion [19]. Collisions can also be useful for exploration of cluttered unknown environments [20]. Our work in this paper focuses on online motion planning that balances collision avoidance and harnessing collisions for mobile robot navigation in unknown environments.

Understanding the trade-offs and balancing between performance and risk has been a research focus for a while. One approach is to utilize the concept of inevitable collision states (ICS) for safe path planning [21], where the states that cannot avoid future collision must be prohibited. However, identifying appropriate ICS can be challenging for robots with noisy actuation and sensing. A different approach is to generate an explicit risk function based only on partial information of the robot state i.e. its location with respect to obstacles, while ignoring its speed and heading [22]–[24]. However, ignoring the kinematic and dynamic properties of

the robot may yield imprecise results. Merging these two types of approaches is possible through a continuous risk function based on the concept of collision time, which is the time in which the vehicle can hit the obstacle along its heading, if it were to lose control [25].

While current risk-aware approaches can describe trade-offs between risk and reward of collision avoidance in mobile robot navigation, they cannot help determine *how to best utilize those collisions in unknown environments*. In related yet distinct efforts, models for reflection on obstacles have been developed to capture velocity, position, and uncertainties that result to collisions [20], [26], [27]. These works can be leveraged to explicitly include boundary interactions within planning algorithms. When the environment is known, collisions can serve as a practical means to improve the effectiveness of trajectories; through dissipation of energy or redirection of momentum, colliding agents can thus be endowed with greater maneuverability [28]. Moreover, accepting collisions can help relax some of the safety conditions when generating a (conservative) path based on ICS planning; the latter may in cases make reaching a desired goal position slower [2]. In previous work [29], we showed how wall-following maneuvers can benefit a stochastically-moving robot to reach its goal in ingress/egress tasks.

In this paper, we design a reactive sampling-based planner for collision-resilient mobile robots navigating in unknown obstacle-cluttered environments. The planner relies only on local information about the environment gathered through its sensors. We first propose a reward function to evaluate the effect of possible collisions based on a reflection model for the robot. We define a risk function based on the concept of collision time. The reward function is integrated with the risk function and a distance metric cost into a joint optimization function for motion planning.

The main contributions of the paper are as follows.

- We present a planner capable of evaluating the trade-offs between harnessing and avoiding collisions in unknown environments.
- We propose a novel formulation of reward function based on a reflection model to help the robot utilize collisions in motion planning.
- We present a pruning technique that can significantly reduce the search space while maintaining the solution quality and ensuring completeness.

The proposed algorithm is tested with the Omnipuck holonomic wheeled robot [26], which we built in-house. The robot includes a reflection ring to help withstand collisions and rapidly change direction of motion after a collision by passively redirecting impact energy. The algorithm is tunable by the user, and thus able to produce less risky paths at the expense of increasing overall path length and lowering the chance of utilizing collisions when appropriate.

II. SAMPLING-BASED PATH PLANNING ALGORITHM

Consider a global (unknown but bounded) map populated with an unknown number of static obstacles. The state $\mathbf{q} = (x, y, \theta, v)$ includes the robot's center of mass position

(x, y) , as well as the direction θ and magnitude v of its instantaneous velocity. We assume that 1) the robot is capable of building a local map of the environment from different sensors (such as a LIDAR) as it navigates, 2) the obstacles in the environment are closed convex sets, and 3) the robot can receive bearing and range information about the goal.

A. Sampling on Local Configuration Space

Let $\mathcal{M} \subset \mathbb{R}^2$ be a sliding local map to represent the environment. The map moves with the robot; its size is a hyper parameter selected by the user. After computing a visibility polygon (e.g., created through ray casting [30]), we extract the local free space \mathcal{F} , obstacle-extending frontiers $\mathcal{B}_i, i = 1, \dots, N$ due to M identified obstacles, and the observable obstacle boundaries $\partial\mathcal{O}_1, \dots, \partial\mathcal{O}_M$. Then, the local map can be expressed as $\mathcal{M} = \mathcal{F} \cup (\bigcup_{i=1}^N \mathcal{B}_i) \cup (\bigcup_{i=1}^M \partial\mathcal{O}_i) \cup \mathcal{U}$, where \mathcal{U} is the unobservable part of the local map (Fig. 2).¹

If the goal is not within \mathcal{F} , $\bigcup_{i=1}^N \mathcal{B}_i$ is sampled to generate a set of candidate states to be explored at future iterations. Let $\mathcal{S} = \{s_\alpha = (x_\alpha, y_\alpha) : s_\alpha \in \bigcup_{i=1}^N \mathcal{B}_i\}$ be the set of positions of all sampled points on the boundary. Positions on each \mathcal{B}_i can be sampled according to some user-defined distribution, e.g., uniformly. Let $\Theta = \{\frac{2\pi n_l}{N_l} : n_l = 0, \dots, N_l\}$ be the set of $N_l \in \mathbb{N}^+$ allowable velocity directions for state expansion. This paper adopts the 8-orientation state expansion, that is $N_l = 8$. Lastly, let $\mathcal{V} = \{\frac{v_{max} n_v}{N_v} : n_v = 0, \dots, N_v\}$ be the set of $N_v \in \mathbb{N}^+$ allowable velocity magnitudes for state expansion. We set $N_v = \lfloor \frac{\text{length}(\mathcal{B}_i)}{\Delta l} \rfloor$, where Δl is a hyper parameter that determines sampling on a frontier \mathcal{B}_i . Then, the sampling configuration space is

$$\mathcal{C} = \mathcal{S} \times \Theta \times \mathcal{V} .$$

The size of \mathcal{C} depends on the size of $\bigcup_{i=1}^N \mathcal{B}_i$ and how it is sampled, and on the values of N_l and N_v . Sampling the velocity vector space (Θ, \mathcal{V}) presents an interesting trade-off. Larger N_l and N_v are expected to yield better results overall as they implicitly constrain the position vector space \mathcal{S} . However, doing so will increase the size of \mathcal{C} and lead to higher computational complexity for planning. Exploring this trade-off is outside the focus of the present paper. While we utilize frontiers, our method is different in that we do not set bounds on velocities based on inevitable collision states but instead evaluate and predict the effect of potential collisions, in unknown environments. Our method can also be tuned to switch between collision exploitation and safety (in the latter case by recovering upper velocity bounds as those used in current collision-avoidance frontier-based methods).

The mechanism to switch between collision exploitation and safety (as well as predict the effect of collisions) is based on constructing a special set \mathcal{E} . This set contains the unit vectors tangent to each $\partial\mathcal{O}_j, j = 1, \dots, M$ where $\partial\mathcal{O}_j$ and any

¹Note that \mathcal{U} contains all partially-observed obstacles with the exception of their observable boundaries.

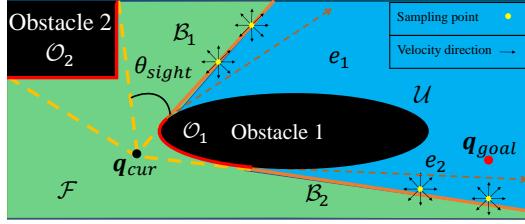


Fig. 2: Instance of sampling on a local map with two obstacles.

B_i intersect. For example in Fig. 2, $\mathcal{E} = \{e_1, e_2\}$. Elements in set \mathcal{E} represent the direction of the *predicted boundaries* of partially-observable obstacles. For convex obstacles, set \mathcal{E} is typically different from the frontier. In the limit that \mathcal{E} and the frontiers coincide, our algorithm will produce a safe collision avoidance behavior as there will be no collisions that can be potentially exploited. (Colliding in that case may be counterproductive.) As we discuss shortly, predicted boundaries are critical in order to be able to estimate possible future collisions, and serve as an integral part in defining this work's risk (in terms of obstacle collision) and collision reward cost functions.

B. Formulating the Planner Optimization Function

The planner seeks to select a sequence of intermediate states $\mathbf{q}_s \in \mathcal{C}$ that take the robot from a current state \mathbf{q}_{cur} (in the local map) to the goal state \mathbf{q}_{goal} . Each intermediate state $\mathbf{q}_s \in \mathcal{C}$ is calculated by solving the (local) unconstrained optimization problem

$$\mathbf{q}_s = \arg \min_{\mathbf{q}_s \in \mathcal{C}} [w_p J_{pos} + w_r J_{risk} + w_v J_{vel}] . \quad (1)$$

The cost function in (1) is a weighted sum of individual costs $\{J_{pos}, J_{risk}, J_{vel}\}$, each representing distinct objectives:

- Minimizing J_{pos} finds the shortest path between \mathbf{q}_{cur} and \mathbf{q}_{goal} . (See Section II-B.1 and Algorithm 1.)
- Minimizing J_{risk} chooses the least risky path in terms of avoiding collisions. (See Section II-B.2 and Algorithm 2.)
- Minimizing J_{vel} picks an as high as possible number of beneficial collisions while maintaining as high speed as possible; *this is where the novelty of our cost function arises from*. Instead of designing a stop maneuver to ensure the robot always stays in free known space, this cost will generate a potential to make the robot go and exploit the unknown area, colliding with obstacles and using collisions as a means to steer toward the goal. (See Section II-B.3.)

Values for weights $\{w_p, w_r, w_v\}$ are hyper parameters selected by the user, and allow the algorithm to be tuned for balancing safety, risk, and exploration.

1) *Generating J_{pos} :* To determine an intermediate state \mathbf{q}_s , we begin by computing J_{pos} via Algorithm 1. In the inputs to Algorithm 1, \mathbf{p}_{cur} and \mathbf{p}_{goal} are the current and goal positions, $\mathbf{p}_s \in \mathcal{S}$ is the position of the candidate point in the sampling configuration space, \mathbf{p}_{pre} is the previous position, and \mathbf{p}_{ini} is the initial starting point. A penalty factor

$f_a \in [1, \infty]$ is applied to force the robot not to vary its orientation beyond a prescribed threshold θ_{thres} .

We first approximate the length of the path from the current position \mathbf{p}_{cur} to the goal position \mathbf{p}_{goal} which consists of the actual cost $PathLength(\mathbf{p}_{ini}, \mathbf{p}_{pre}, \mathbf{p}_{cur}, \mathbf{p}_s)$ and the heuristic cost $PathLength(\mathbf{p}_s, \mathbf{p}_{goal})$. We then assign the penalty factor f_a to regulate direction of motion, and normalize the output to yield $J_{pos} = normalize(P_{pose})$.

Algorithm 1: Generate P_{pose}

```

input :  $\mathbf{p}_{cur}, \mathbf{p}_{goal}, \mathbf{p}_s, \mathbf{p}_{pre}, \mathbf{p}_{ini}, \theta_{thres}, f_a$ 
output:  $P_{pose}$ 
1  $PL_{actual} \leftarrow PathLength(\mathbf{p}_{ini}, \mathbf{p}_{pre}, \mathbf{p}_{cur}, \mathbf{p}_s)$ 
2  $PL_{heuristic} \leftarrow PathLength(\mathbf{p}_s, \mathbf{p}_{goal})$ 
3  $D_{pos} \leftarrow PL_{actual} + PL_{heuristic}$ 
4 if  $\angle(\mathbf{p}_{pre}, \mathbf{p}_{current}, \mathbf{p}_s) \geq \theta_{thres}$  then
5   |  $P_{pose} \leftarrow f_a \times D_{pos}$ 
6 else
7   |  $P_{pose} \leftarrow D_{pos}$ 
8 end
9 return  $P_{pose}$ 

```

2) *Generating J_{risk} for a point:* We define the risk as the probability of colliding with obstacles. To generate J_{risk} , which is an explicit evaluation of probability of collision, we follow Algorithm 2. In the inputs to Algorithm 2, \mathbf{v}_s is the velocity of the candidate point, \mathcal{E} is the set of unit vectors e_i of predicted obstacle boundaries, $\mathbf{p}_p = (x_p, y_p)$ is the potential collision point based on \mathcal{E} , and δv is a positive infinitesimal value used to evaluate J_{risk} based on the distance between the robot and the obstacles when the robot is stationary.

Algorithm 2: Generate J_{risk}

```

input :  $\mathbf{p}_s, \mathbf{v}_s, \mathbf{p}_p, \mathcal{E}, \delta v$ 
output:  $J_{risk}$ 
1 if  $\|\mathbf{v}_s\| > 0$  then
2   |  $\mathbf{s}_p \leftarrow \mathbf{p}_p - \mathbf{p}_s$ 
3   |  $d_p \leftarrow \|\mathbf{s}_p\|$ 
4   |  $v_c \leftarrow \|\text{Proj}(\mathbf{v}_s, \mathbf{s}_p)\|$ 
5 else
6   |  $v_c \leftarrow \delta v$ 
7   |  $d_p \leftarrow \arg \min_{\mathbf{e}_i \in \mathcal{E}} \text{distance}(\mathbf{p}_s, \mathbf{e}_i)$ 
8 end
9  $t_c \leftarrow \frac{d_p}{v_c}$ 
10  $J_{risk,h} \leftarrow normalize(\frac{1}{t_c})$ 
11  $J_{risk,g} \leftarrow -f_g \theta_{sight}$ 
12  $J_{risk} \leftarrow J_{risk,g} + J_{risk,h}$ 
13 return  $J_{risk}$ 

```

We determine the heuristic part of J_{risk} by calculating the collision time t_c [25] based on the current state and the predicted boundary of the obstacle in \mathcal{U} if the candidate velocity $\|\mathbf{v}_s\| > 0$. Otherwise, we generate J_{risk} given δv and d_p where d_p is the minimal distance between the robot

and the obstacles in the environment. We use $J_{risk,g} = -f_\theta \theta_{sight}$ (line 11), where θ_{sight} is the angle of the narrow region as illustrated in Fig. 2. Weighting θ_{sight} with a constant penalizing factor $f_\theta \in [0, 1]$ helps prevent the robot from going into narrow regions in \mathcal{F} .

3) *Generating J_{vel}* : This cost forces the robot to exploit the unknown area, colliding with obstacles and using collisions as a means to steer toward the goal. It is the distinctive element with respect to other related algorithms (e.g., [2], [4], [5]) in the sense that it relaxes the condition of calculating and enforcing inevitable collision states (ICS). In [26], a reflection model is generated to fit the behavior of a holonomic robot after colliding with the environment. Works [26] and [20] show that bouncing motion primitives can be leveraged in fast planning algorithms to generate minimum-time trajectories. We simplify the complex problem of vehicle-surface interactions using Garwinski model [31]. Then we can generate an explicit reward function to evaluate whether the robot can benefit from the reflection-like bouncing behavior.

$$\mathbf{R}_{ref}(t) = \begin{cases} \langle \mathbf{v}_{ref}, \mathbf{unit}_{goal} \rangle & \mathbf{p}_{ref} \notin \mathcal{F} \\ \text{penalty} & \text{otherwise} . \end{cases} \quad (2)$$

The reflection velocity \mathbf{v}_{ref} in (2) is the instantaneous velocity of the robot after impact. It is calculated by taking the norm of \mathbf{e}_p which lies along the predicted boundary and is directed toward \mathcal{U} . We apply a rotation to \mathbf{e}_p and denote the normal vector as $\mathbf{n}_p = \text{Rot}(\mathbf{e}_p, \frac{\pi}{2})$. \mathbf{v}_{ref} can then be derived from \mathbf{v}_s and \mathbf{n}_p based on the reflection model. \mathbf{p}_{ref} is an estimation of the robot's position after a user-defined time interval. *penalty* is a penalty value in $[-\infty, 0]$ to prevent the robot bouncing back to the currently known space.

After we compute the explicit heuristic function to evaluate risk and reflection in unknown space, we then generate J_{vel} which represents how the impact velocity on the boundary will affect future behavior of the robot. The effect of velocity in \mathcal{C} can then be approximated by

$$J_{vel} = -\langle \mathbf{v}_s, \mathbf{p}_{goal} - \mathbf{p}_{cur} \rangle (1 - J_{risk,h}) - R_{ref} J_{risk,h} . \quad (3)$$

C. Candidate State Pruning for Online Implementation

To make the algorithm online, we develop a two-step pruning technique for reducing the number of candidate states $\mathbf{q}_s \in \mathcal{C}$. States are removed based on either their position and/or their velocity vector (see Fig. 3).

Position-Based Pruning: In position-based pruning, candidate states with positions that are less likely to be selected are dynamically identified and removed from \mathcal{C} . The chance for a candidate state to be selected based on its position alone depends on the number of neighboring candidate states and their relative distance to the goal, and whether there is separation by free or unobservable space. For example, in the illustrations shown in Fig. 3a and Fig. 3b, candidate states lying on \mathcal{B}_2 are pruned because they are close to candidate states lying on \mathcal{B}_3 which in turn are closer to the goal, and they are separated from them by free space \mathcal{F} .

Position-based pruning does not remove candidate states that may be far from the goal (such as those lying on

\mathcal{B}_1 and \mathcal{B}_4) to promote exploration and to handle potential uncertainties (e.g., robot drift or under-estimation of collision impact). Instead, it removes states that are expected to have a similar effect, if selected, toward exploration. The completeness can be guaranteed assuming that the obstacle is convex, since the algorithm always enforces exploration of the unknown space \mathcal{U} . However, the pruning technique cannot guarantee completeness in maze-like environments.

Velocity-Direction-Based Pruning: In velocity-based pruning, candidate states with velocities directed toward the free space \mathcal{F} are pruned. As in position-based pruning, the objective here is to retain the candidate states with velocities that encourage exploration. For example, in the illustrations shown in Fig. 3b and Fig. 3c, all candidate states remaining after position-based pruning are evaluated once more, and only those that lead toward unobservable space \mathcal{U} are kept.

III. TRAJECTORY GENERATION

We are now ready to introduce our trajectory generation algorithm for collision-resilient robots navigating in unknown obstacle-cluttered environments. Recall that no prior knowledge of the map is required. The algorithm consists of two components: 1) generating appropriate waypoints via B-spline interpolation, and 2) selecting appropriate low level collision-free or collision-harnessing maneuvers.

A. B-spline Generation

Once the intermediate state \mathbf{q}_s is found, we use the cubic B-spline [32] to interpolate a spline with control points as WP . The Python B-spline interpolation function in the `scipy` package is used to generate a cubic B-spline Spl is the set of points in the B-spline. A trajectory is then generated from Spl using a time interval T defined as

$$T = \max\{T_v, T_{map}\} , \quad (4)$$

where T_{map} is the time needed to update the sliding map and T_v is the maximum velocity of the robot. We can approximate T_v by

$$T_v = P_{safe} \times \frac{\sum_{i=2}^N \|WP[i] - WP[i-1]\|}{v_{max}} \quad (5)$$

where P_{safe} is the parameter which restricts the reference velocity to stay within the range $[0, v_{max}]$. In our case, we choose $N = 4$ to keep the minimal number of control points for cubic spline.

B. Maneuver Selection

We utilize a low-level switching controller to ensure that the robot follows the trajectory and reaches the desired waypoint. At each instance when the sliding local map is updated, the controller switches between the following strategies: free-space, boundary following [29], [33], and flow-through [27] as described in Algorithm 3. From the switching strategy, the robot will determine whether the B-spline trajectory is suitable or it needs to generate a new reference trajectory as illustrated in Fig. 4. In all strategies, the robot follows reference trajectories using a PID controller.

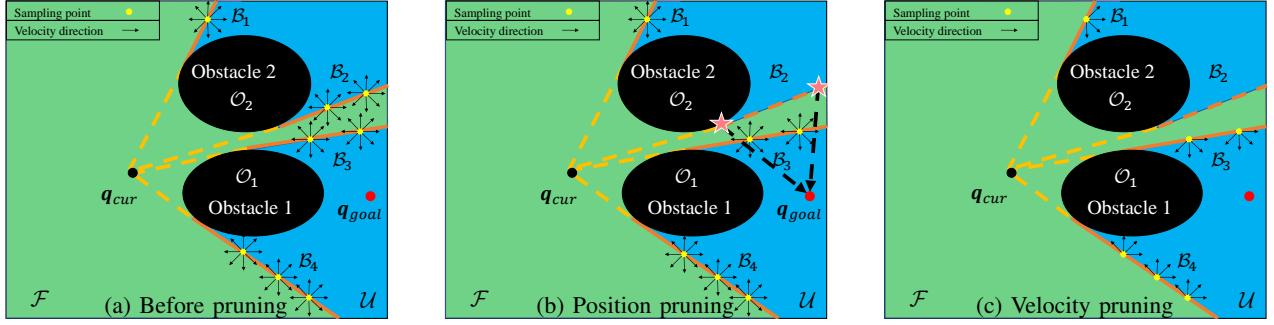


Fig. 3: Example of pruning on the sampling configuration space \mathcal{C} .

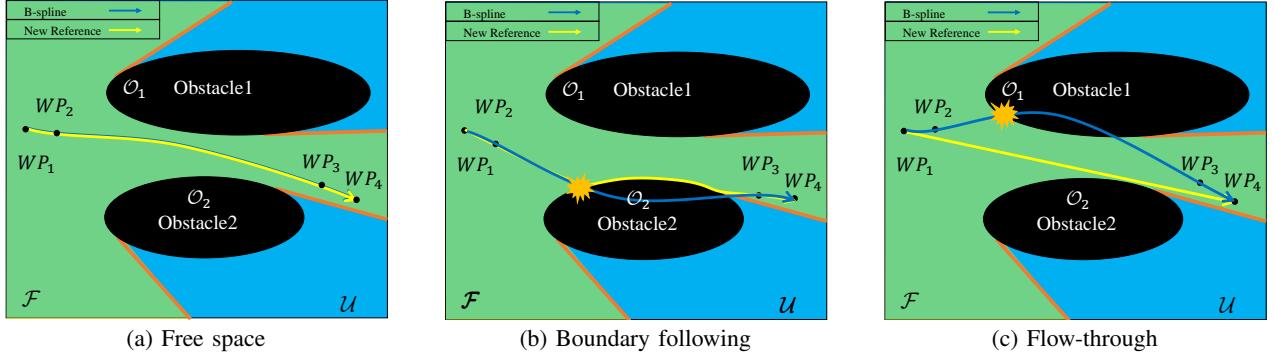


Fig. 4: The switching controller deals with potential collisions that may occur in future time.

Algorithm 3: Maneuver Selection

```

input : WP,  $\mathbf{p}_c$ 
output: controller_class
1 if  $\mathbf{p}_c = \emptyset$  then
2   | controller_class = free_space
3 else
4   |  $\mathbf{v} \leftarrow \frac{\mathbf{WP}_4 - \mathbf{WP}_1}{\|\mathbf{WP}_4 - \mathbf{WP}_1\|}$ 
5   |  $\mathbf{v} \leftarrow \text{Rot}(\mathbf{v}, \frac{\pi}{2})$ 
6   |  $\mathbf{v}_c \leftarrow \mathbf{p}_c - \mathbf{WP}_1$ 
7   | if  $\langle \mathbf{v}, \mathbf{v}_c \rangle \geq 0$  then
8     | | controller_class = flow_through
9   | else
10    | | controller_class = boundary_following
11  end
12 end
13 return controller_class
  
```

When the robot does not face a collision, it utilizes the free-space strategy and follows the trajectory generated from the B-spline \mathbf{q}_{spl} as depicted in Fig. 4a. If colliding at position \mathbf{p}_c , the robot will utilize either the flow-through or the boundary following strategy. Let \mathbf{q}_{bound} be the trajectory projection on the boundary. If $\langle \mathbf{v}, \mathbf{v}_c \rangle < 0$, then the robot will use the boundary following strategy. The collision point \mathbf{p}_c , where the B-spline \mathbf{q}_{spl} first intersects the obstacle boundary $\mathbf{b}_{\mathcal{O}}$, is referred to as the engaging point \mathbf{p}_d . At this point, the robot will follow a newly generated reference trajectory \mathbf{q}_{bound} , generated by projecting \mathbf{q}_{spl} to the boundary $\mathbf{b}_{\mathcal{O}}$ until the disengaging point. The boundary following

maneuver is depicted in Fig. 4b.

$$\mathbf{q}_{ref}(t) = \begin{cases} \mathbf{q}_{spl}(t) & s = off \\ \mathbf{q}_{bound}(t) & s = on \end{cases}$$

$$\phi(s, x) = \begin{cases} off & s = off, \|\mathbf{p}(t) - \mathbf{p}_c\| \geq \delta \\ on & s = off, \|\mathbf{p}(t) - \mathbf{p}_c\| \leq \delta \\ off & s = on, \|\mathbf{p}(t) - \mathbf{p}_d\| \leq \delta \end{cases}.$$

If $\langle \mathbf{v}, \mathbf{v}_c \rangle \geq 0$ and $\mathbf{q}_{spl} \cap \mathcal{O} \neq \emptyset$, then the robot will use the flow-through maneuver as depicted in Fig. 4c. \mathbf{q}_{flow} is generated by projecting \mathbf{q}_{spl} to vector $WP[4] - WP[1]$.

$$\mathbf{q}_{ref}(t) = \begin{cases} \mathbf{q}_{spl}(t) & s = off \\ \mathbf{q}_{flow}(t) & s = on \end{cases}$$

$$\phi(s, x) = \begin{cases} off & s = off, \|\mathbf{p}(t) - \mathbf{p}_c\| \geq \delta \\ on & s = off, \|\mathbf{p}(t) - \mathbf{p}_c\| \leq \delta \end{cases}.$$

IV. EXPERIMENTAL RESULTS

A. Robot and Environment Setup

We implement our proposed algorithm on the collision-resilient Omnipuck platform [26] which we built in-house (Fig. 1). The body of the Omnipuck robot is surrounded by a reflection ring that enables it to collide safely with the environment and rebound from it. The robot operates in a $2.4m \times 2.4m$ confined corridor environment with rectangular pillars serving as static convex obstacles as shown in Fig. 1. The position of Omnipuck is captured using a 12-camera VICON motion capture system. A laptop with a 2.3 GHz i7 CPU and 12 GB RAM processes position data and sends controller commands to the robot at a frequency of 20 Hz.

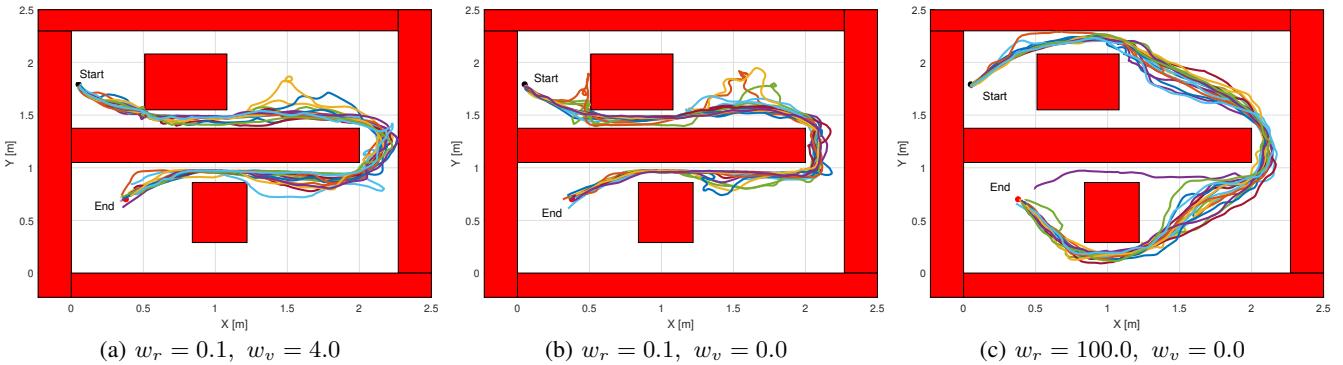


Fig. 5: Experimental trajectories with $T_{map} = 0.2s$. (a) High risk path with intentional collision reflection. (b) High risk path without intentional collision reflection. (c) Low risk path without intentional collision reflection.

We utilize the ray casting algorithm [30] to simulate LIDAR behavior and generate a current sliding map. See <https://www.youtube.com/watch?v=S3oYebJRFa0> for instances of our experiments.

B. Effect of Harnessing Collisions

To examine the effect of different weight factors $\mathbf{w} = [w_p, w_r, w_v]$ in (1) on planning the motion of a collision-resilient robot in unknown environments, we select weight factors that emphasize components of the objective function (1) to generate either a low/high risk path or a path that harnesses collisions. We consider three case studies as follows.

- A high risk trajectory that intentionally collides with the environment to harness collisions is generated when (1) is optimized with w_v set to a large value.
- A high risk trajectory that may unintentionally collide with the environment without harnessing collisions is generated when (1) is optimized with w_r set to a small value and w_v set to zero.
- A low risk ('safe') trajectory that seeks to avoid collisions altogether is generated when (1) is optimized with w_r set to a large value and w_v set to zero.

A higher w_v is expected to produce a risky trajectory which can utilize the reflection after colliding to continue making progress toward the goal. We anticipate that in this way the robot can reach the goal with a shorter arrival time and path length. To test this hypothesis, we run 20 closed-loop experimental trials with map update time $T_{map} = 0.2$ sec with each of the three strategies listed above. The top speed in all cases is $v_{max} = 1.2$ m/s.

Experimental results are shown in Fig. 5. Individual trials for each case are overlaid on an augmented-obstacle representation of the environment map. We notice that when the planner can tolerate more risk (i.e. lower w_r), then the robot will try go through the two narrow corridors to decrease arrival time and total path length. However, if no mechanism to harness collisions is in place, and unintentional collisions do happen, some of them may have a negative effect on the robot trajectory. For example, in Fig. 5(b) some trials collide with the left side of the top obstacle, effectively increasing both the arrival time and the total path length. On the contrary, if the planner deliberately seeks to collide when needed (Fig. 5(a)) we observe that no such negative

collisions occur. As expected, a risk-averse (i.e. higher w_r) planner will seek for longer trajectories through the wider parts of the environment (Fig. 5(c)).

The improvement (%) when utilizing collisions in terms of arrival times and total path length can be clearly seen in Table I. Results reveal that our proposed planning algorithm to harness collisions has a major impact in terms of mean arrival time. Mean total path length is similar to the case of high-risk planning without explicitly harnessing collisions, but significantly better than in the case of low-risk planning.

TABLE I: Improvements of collision-harnessing planning in terms of arrival time and total path length when compared to high-risk and low-risk planning without intentional collisions.

Improvement (%)	High risk path	Low risk path
Arrival time mean	7.60	8.53
Arrival time STD	3.48	2.39
Path length mean	2.32	16.01
Path length STD	0.99	0.47

C. Parametric Study on Different Mapping Times

Finally, we investigate the effect of map update time T_{map} . We collect 20 experimental trials for each of the three strategies, and for $T_{map} \in [0.2, 0.4, 0.6, 0.8, 1.0]$ sec in the same environment, thus giving rise to a total of 300 experimental paths. Results are shown in Fig. 6. As T_{map} increases, both mean arrival times and mean total path lengths for all three strategies increase. This is because uncertainty in the controller increases as the duration to update the map increases. In all cases, harnessing collisions yields better results; however, the rates of increase in arrival times and total path lengths do not appear to depend on the employed planning strategy.

V. CONCLUSIONS

We present a new reactive planning algorithm suitable for collision-resilient robots in unknown environments. The planner allows for potential collisions to be harnessed to improve robot navigation in obstacle-cluttered environments. Experiments show that integrating collision harnessing in online planning can decrease both the arrival time and path length of generated trajectories when compared to high-risk trajectories without utilizing collisions ($> 7\%$, $> 2\%$), and risk-averse safe trajectories that avoid collisions altogether

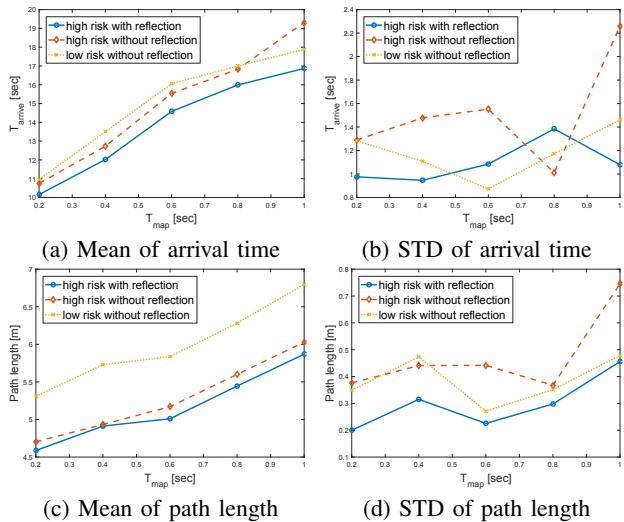


Fig. 6: Arrival time and path length with different T_{map} .

($> 8\%$, $> 16\%$). At each planning interval, the pruning technique can save about 3.1ms of computational time.

The work herein contributes foundational algorithmic tools to support the emerging paradigm shift where robots might deliberately choose to collide with the environment, should that help them make progress toward an assigned goal they are tasked to reach. In addition, the pruning technique introduced in this work can apply to other planning algorithms as well, to help reduce the computational time in support of online operation. Future work will focus on analyzing in detail the effect of the various quantities that determine the degree of pruning. We will also focus on experimental evaluation in more complex environments.

REFERENCES

- [1] X. Xiao, J. Dufek, and R. Murphy, “Explicit-risk-aware path planning with reward maximization,” *arXiv preprint arXiv:1903.03187*, 2019.
- [2] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, “Real-time planning with multi-fidelity models for agile flights in unknown environments,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 725–731.
- [3] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
- [4] B. T. Lopez and J. P. How, “Aggressive collision avoidance with limited field-of-view sensing,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1358–1365.
- [5] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, “Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1474–1481, 2018.
- [6] V. Vasilopoulos and D. E. Koditschek, “Reactive navigation in partially known non-convex environments,” 2018.
- [7] M. Pivtoraiko, D. Mellinger, and V. Kumar, “Incremental micro-uav motion replanning for exploring unknown environments,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2452–2458.
- [8] A. Dhawale, X. Yang, and N. Michael, “Reactive collision avoidance using real-time local gaussian mixture model maps,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3545–3550.
- [9] L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeyns, “Autonomous visual mapping and exploration with a micro aerial vehicle,” *Journal of Field Robotics*, vol. 31, no. 4, pp. 654–675, 2014.
- [10] B. Davis, I. Karamouzas, and S. J. Guy, “C-opt: Coverage-aware trajectory optimization under uncertainty,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1020–1027, 2016.
- [11] M. Hoy, A. S. Matveev, and A. V. Savkin, “Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey,” *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.
- [12] S. Campbell, W. Naeem, and G. W. Irwin, “A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres,” *Annual Reviews in Control*, vol. 36, no. 2, pp. 267–283, 2012.
- [13] D. W. Haldane, C. S. Casarez, J. T. Karras, J. Lee, C. Li, A. O. Pullin, E. W. Schaler, D. Yun, H. Ota, A. Javey *et al.*, “Integrated manufacture of exoskeletons and sensing structures for folded millirobots,” *Journal of Mechanisms and Robotics*, vol. 7, no. 2, p. 021011, 2015.
- [14] A. Briod, P. Kornatowski, J.-C. Zufferey, and D. Floreano, “A collision-resilient flying robot,” *Journal of Field Robotics*, vol. 31, no. 4, pp. 496–509, 2014.
- [15] T. Li, Z. Zou, G. Mao, X. Yang, Y. Liang, C. Li, S. Qu, Z. Suo, and W. Yang, “Agile and resilient insect-scale robot,” *Soft robotics*, vol. 6, no. 1, pp. 133–141, 2019.
- [16] K. Karydis, D. Zarrouk, I. Poulopoulos, R. S. Fearing, and H. G. Tanner, “Planning with the star (s),” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 3033–3038.
- [17] S. Mayya, P. Pierpaoli, G. Nair, and M. Egerstedt, “Localization in densely packed swarms using interrobot collisions as a sensing modality,” *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 21–34, 2018.
- [18] T. Schmickl, R. Thenius, C. Moeslinger, G. Radspieler, S. Kernbach, M. Szymanski, and K. Crailsheim, “Get in touch: cooperative decision making based on robot-to-robot collisions,” *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 1, pp. 133–155, 2009.
- [19] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, “Robotic vertical jumping agility via series-elastic power modulation,” *Science Robotics*, vol. 1, no. 1, 2016.
- [20] Y. Mulgaonkar, A. Makineni, L. Guerrero-Bonilla, and V. Kumar, “Robust aerial robot swarms without collision avoidance,” *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 596–603, 2017.
- [21] T. Fraichard and H. Asama, “Inevitable collision states step towards safer robots?” *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.
- [22] B. Pfeiffer, R. Batta, K. Klamroth, and R. Nagi, “Probabilistic modeling for uav path planning in the presence of threat zones,” in *Handbook of military industrial engineering*. CRC Press, 2009, pp. 5–1.
- [23] D. Liu, M. Cong, and Y. Du, “Episodic memory-based robotic planning under uncertainty,” *IEEE Transactions on Industrial Electronics*, vol. 64, no. 2, pp. 1762–1772, 2016.
- [24] H. Huang and A. V. Savkin, “Viable path planning for data collection robots in a sensing field with obstacles,” *Computer Communications*, vol. 111, pp. 84–96, 2017.
- [25] J. Song, S. Gupta, and T. A. Wettergren, “T*: Time-optimal risk-aware motion planning for curvature-constrained vehicles,” *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 33–40, 2018.
- [26] A. Stager and H. G. Tanner, “Mathematical models for physical interactions of robots in planar environments,” in *International Symposium on Experimental Robotics*. Springer, 2018, in press.
- [27] ———, “Composition of local potential functions with reflection,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5558–5564.
- [28] M. Mote, M. Egerstedt, E. Feron, A. Bylard, and M. Pavone, “Collision-inclusive trajectory optimization for free-flying spacecraft,” *Journal of Guidance, Control, and Dynamics*, pp. 1–12, 2020.
- [29] Z. Lu and K. Karydis, “Optimal steering of stochastic mobile robots that undergo collisions with their environment,” in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 668–675.
- [30] F. Bungiu, M. Hemmer, J. Hershberger, K. Huang, and A. Kröller, “Efficient computation of visibility polygons,” *arXiv preprint arXiv:1403.3905*, 2014.
- [31] R. L. Garwin, “Kinematics of an ultraelastic rough ball,” *American Journal of Physics*, vol. 37, no. 1, pp. 88–92, 1969.
- [32] M. Elbanhawi, M. Simic, and R. N. Jazar, “Continuous path smoothing for car-like robots using b-spline curves,” *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 23–56, 2015.
- [33] A. Stager and H. G. Tanner, “Stochastic behavior of robots that navigate by interacting with their environment,” in *IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 6871–6876.