DETC2014/MESA-35561

COLLISION PREDICTION: CONSERVATIVE ADVANCEMENT AMONG OBSTACLES WITH UNKNOWN MOTION*

Yanyan Lu

MASC Group

Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: ylu@gmu.edu

Zhonghua Xi

MASC Group

Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: zxi@gmu.edu

Jyh-Ming Lien

MASC Group

Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: jmlien@cs.gmu.edu

ABSTRACT

Collision detection is a fundamental geometric tool for sampling-based motion planners. On the contrary, *collision prediction* for the scenarios that obstacle's motion is unknown is still in its infancy. This paper proposes a new approach to predict collision by assuming that obstacles are *adversarial*. Our new tool advances collision prediction beyond the translational and disc robots; arbitrary polygons with rotation can be used to better represent obstacles and provide tighter bound on predicted collision time. Comparing to an online motion planner that replans periodically at *fixed time interval*, our experimental results provide strong evidences that our method significantly reduces the number of replannings while maintaining higher success rate of finding a valid path.

1 INTRODUCTION

During the past three decades, there have been extensive work on planning motion in dynamic environments. One of the first ideas was to construct visibility graph(s) in configuration-time space (CT-space) [1]. In late 1990s, probabilistic methods such as PRM [2] and RRT [3] greatly enhanced the ability of planners by sampling and connecting configurations in CT-space.

The idea of temporal coherence is later exploited to gain better efficiency by repairing the invalid portion of the (tree-based or

graph-based) roadmaps or path since the changes in the configuration space is usually small from frame to frame [4, 5, 6, 7]. These planning strategies are often known as *replanning methods* [8, 9, 10, 11, 12]. Although these replanning methods are efficient, almost all existing frameworks update the environmental map and then replan periodically at *fixed time interval*. That is, even if there are no changes in the configuration space, motion planner will still be invoked to replan. The situation is even worse when replanning is not done frequently enough. Paths that are believed to be valid may become unsafe. Ideally, the repair interval should be determined adaptively based on the motion of the obstacles.

Motivated by this observation, we consider the problem of *collision prediction* under the framework of online motion planning in the workspace consisting of dynamic obstacles that moves along some unknown trajectories with bounded velocities. More specifically, we are interested in determining the time that the robot **R** collides with an obstacle **O** whose motion is unknown when **R** travels on a path Π .

Main Contribution In this paper, we propose a new geometric tool called *collision prediction* that allows the robot to determine the critical moments that the robot and obstacles can collide. Then, only at critical moment, the robot will update its belief of the environmental configuration and re-plan if necessary. The main challenge in predicting collision steams from the assumption that obstacle's motion is unknown. To provide conservative estimation, the basic framework introduced in this paper models the obstacles as *adversarial agents* that will minimize the time that the robot remains collision free. As a result, a robot can

*This work is supported in part by NSF IIS-096053, NSF EFRI-1240459, NSF CNS-1205260, FHWA DTFH61-10-R-00036, and AFOSR FA9550-12-1-0238 and USGS.

actively determine its next replanning time by conservatively estimating the amount of time (i.e., *earliest collision time*) that it can stay on the planned path without colliding with the obstacles. The idea of earliest collision time and conservative advancement are detailed in Section 3.

Overall, our new tool advances collision prediction beyond the translational and disc robots [13, 14, 15, 16, 11]. Arbitrary (even non-simple) polygons with rotation can be used to better represent obstacles and provide tighter bound on predicted collision time. This prediction is determined only based on the last known positions of the obstacles and their maximum linear and angular velocities. In our experimental results (Section 6), we demonstrate that our method significantly reduces the number of replannings while maintaining higher success rate of finding a valid path.

2 RELATED WORK

Motion planning problems involving dynamic environments can be roughly classified into two categories: (1) The trajectory of every moving obstacle is fully known in advance, and (2) the trajectory of a moving obstacle is partially or completely unpredictable. Since our work falls into the second category, we will focus on reviewing recent works considering unknown environments.

2.1 Collision Avoidance

Due to little knowledge of the environment, safety becomes very important and challenging in path planning in unknown environments [17, 18, 15, 19, 20, 21, 22, 23, 24, 25]. Fraichard and Asama [21] provided the formal definitions of two new concepts: inevitable collision state (ICS) and inevitable collision obstacle (ICO). If the robot is in an ICS, no matter what its future trajectory is, a collision eventually occurs with an obstacle in the environment. ICO is a set of ICS yielding a collision with a particular obstacle. Shiller et al. [19] proposed a motion planner based on Velocity Obstacles (VO) for static or dynamic environments. The time horizon for a velocity obstacle is computed based on the current positions of robot and the obstacle as well as control constraints. With this adaptive time horizon strategy, the velocity obstacle tightly approximates the set of ICS. Gomez and Fraichard [22] proposed another ICS-based collision avoidance strategy called ICS-AVOID. ICS-AVOID aims at taking the robot from one non-ICS state to another. The concept of Safe Control Kernel is introduced and it guarantees ICS-AVOID can find a collision-free trajectory if one exists. Recently, Bautin et al. [26] proposed two ICS-checking algorithms. Both algorithms take a probabilistic model of the future as input which assigns a probability measure to the obstacles' future trajectories. Instead

of answering whether a given state is an ICS or not, it returns the *probability* of a state being an ICS. Wu and How [27] extended VO to moving obstacles with constrained dynamics but move unpredictably. To compute the velocity obstacle of an obstacle o at some time t , it first predicts its reachable region considering all possibly feasible trajectories and then maps this reachable region into velocity space by dividing it by time. To the best of our knowledge, all ICS or VO (even [26, 27]) variants require certain level of information about the future in the environment. When it comes to environments whose future is largely unpredictable, ICS or VO may still fail to avoid approaching collisions, while our new method can guarantee safety only based on the maximum velocities of obstacles.

The work closes to the spirit of our new method is by van den Berg and Overmars [11]. Their work assumes that the robot and all obstacles are discs and it conservatively models the swept volume of an obstacle over time as a cone with the slope being its maximum velocity. In this way, no matter how the obstacle moves, it is always contained inside this cone. Therefore, the computed path is guaranteed to be collision free. However, these assumptions can be unrealistic for many applications. For obstacles with arbitrary shapes or rotation, computing their swept volumes is nontrivial.

2.2 Collision Prediction

Since the robot has partial or no information about the environment, it is very difficult to plan a collision free path for it to move through a field of static or dynamic obstacles to a goal. One of the biggest challenge is to predict possible collisions with dynamic obstacles whose trajectories are unknown. There exists a lot of work which checks collisions at a sequence of fixed time steps [10, 28, 29, 30, 31]. For example, van den Berg et al. [10] performed collision detections at fixed time intervals (every 0.1 seconds in their experiments). Both the robot and dynamic obstacles were modeled as discs moving in the plane. Moreover, the future motions of a moving obstacle were assumed to be the same as its current motions. In order not to miss any collisions, they either increased the number of time steps or assumed the objects move slowly.

There are also works which adaptively changed the frequency of collision checks: collisions are more frequently checked for two objects which are more likely to collide. Hayward et al. [13], Kim et al. [16] and Hubbard [14] assumed that the maximum magnitude of the acceleration is provided for each object. Hayward et al. calculated the amount of time within which two moving spheres are guaranteed not to collide with each other. Then more attention was adaptively paid to objects which are very likely to collide. Hubbard first detected collisions between the bounding spheres of two objects. Then the pairs of objects

whose bounding spheres collide are checked for collisions using sphere trees that represent the objects. Kim et al. [16] first computed the *time-varying bound volume* for each moving sphere with its initial position, velocity and the maximum magnitude of its acceleration. As time goes by, the radius of this *time-varying bound volume* increases and it is guaranteed to contain the sphere at any time in the future. For two moving spheres, whenever their *time-varying bound volumes* intersect, they are checked for actual collision. Chakravarthy and Ghose [15] proposed *collision cone* approach (similar as velocity obstacle) for predicting collisions between any two irregularly shaped polygons translating on unknown trajectories. All these methods are limited to discs, spheres or translational objects. Our new tool allows polygons with arbitrary shape (even non-simple polygons) with rotation.

Almost all existing works collect sensory data and update its environmental information at fixed times. As a result, either updating is redundant or the situation is even worse if update is performed not frequently. The robot may be at some state which leads it to be in unavoidable collisions. To address this, we propose to update environmental belief when necessary by exploring temporal coherence of obstacles and predict a critical time t such that the robot is guaranteed to move safely along its current path until t .

3 Overview: Conservative Advancement

Planning a path in environments populated with obstacles with unknown trajectories usually involves two steps: (1) find an initial path Π based on known information and then (2) modify Π as the robot receives new information from its onboard sensors at *fixed times*. To provide a more concrete framework for our discussion, we assume that the robot $\mathbf{R}$ still plans a path Π based on its current belief of the state of the workspace. However, instead of determining if Π is still safe to traverse at fixed time, $\mathbf{R}$ determines the critical moment t that Π may become invalid. The robot budgets a certain amount of time Δt before this critical moment t to update its belief and replan if necessary. To make our discussion more concrete, let's emphasize again that this setting is merely a framework among other applications of collision prediction.

Because the trajectory of the obstacles in workspace is unknown, the critical moment t can only be approximated. To ensure the safety of the robot, our goal is to obtain conservative estimation $t' \leq t$ of the unknown value t . Follow the naming tradition in collision detection, we call such an estimation *conservative advancement* on Π and denote it as CA_Π . To compute CA_Π , the robot assumes that all obstacles are adversarial. That is, these adversarial obstacles will move in order to minimize the time that Π remains valid.

Contrary to traditional motion planning methods [32,33], the calculation of CA_Π (performed by the robot) in some sense reverses the roles of robot and obstacles. The robot $\mathbf{R}$ is now fixed to the path Π , thus the configuration of $\mathbf{R}$ at any given time is known. On the other hand, the obstacles' trajectories are unknown but will be planned to collide with $\mathbf{R}$ in the shortest possible time. As we will see later, the motion strategy for an obstacle $\mathbf{O}_i$ will only depend on the maximum translational velocity v_i and a maximum angular velocity ω_i around a given reference point o .

3.1 Estimate Conservative Advancement on Path Π

Without loss of generality, the problem of estimating CA_Π can be greatly simplified if we focus on only a single obstacle and a segment of path Π . Let Π be a sequence of free configurations $\Pi = \{c_1, c_2, \dots, c_n\}$ with $c_1 = \mathbf{S}$ and $c_n = \mathbf{G}$, where the $\mathbf{S}$ and $\mathbf{G}$ are start and goal configurations, respectively. Given a segment $\overline{c_j c_{j+1}} \subset \Pi$, we let $ECT_{i,j}$ be the earliest collision time (ECT) that $\mathbf{O}_i$ takes to collide with the robot at a point $c \in \overline{c_j c_{j+1}}$. Then we have $CA_\Pi = \min_i (\min_j (ECT_{i,j}))$, where $1 \leq i \leq |\mathbf{O}|$ and $1 \leq j < n$. Note that $ECT_{i,j}$ is infinitely large, if $\mathbf{O}_i$ cannot collide with $\mathbf{R}$ before $\mathbf{R}$ leaves $\overline{c_j c_{j+1}}$.

Lemma 3.1. *If $ECT_{i,j} \neq \infty$, then $ECT_{i,j} \leq ECT_{i,k}, \forall k > j$*

That is once an earliest collision time is detected for a path segment $\overline{c_j c_{j+1}}$, it is not necessary to check its subsequent segments including $\overline{c_k c_{k+1}}$ with $j < k < n$. In Section 3.2, we will provide a brief overview on how $ECT_{i,j}$ can be computed.

Before we proceed our discussion, we would like to point out that our method does not consider collisions between the obstacles. Although this makes our estimate more conservative, the obstacle with the earliest collision time rarely collides with other obstacles.

3.2 Earliest Collision Time (ECT)

Given a segment $\overline{c_j c_{j+1}} \subset \Pi$ of path in C-space, our goal is to compute the earliest collision time $ECT_{i,j}$ when obstacle $\mathbf{O}_i$ hits robot $\mathbf{R}$ somewhere on $\overline{c_j c_{j+1}}$. Assume $\mathbf{R}$ starts to execute on Π at time 0.

Since the robot $\mathbf{R}$ moves along a known path Π , $\mathbf{R}$ knows when it reaches a given configuration $c \in \Pi$. Let t be the time that $\mathbf{R}$ takes to reach a configuration $c(t) \in \overline{c_j c_{j+1}}$ and let T be the time when $\mathbf{O}_i$ reaches this $c(t)$. Because $\mathbf{O}_i$ is constrained by its maximum linear and angular velocities v_i and ω_i , there must exist an earliest time $\hat{T}$ for $\mathbf{O}_i$ to reach $c(t)$ without violating these constraints. Since every configuration on $\overline{c_j c_{j+1}}$ is parameterized by t , this $\hat{T}$ can also be expressed as a function of t . Let this function be $f(t)$. Furthermore, when the robot $\mathbf{R}$ and $\mathbf{O}_i$ collide, they must reach a configuration c at the same time. Therefore, we

FIGURE 1. The red (thicker) curves in both figures are plots of the earliest arrival time $f(t)$ for an obstacle. Black straight lines are plots of $g(t) : t = T$. (a) When there is at least one intersection (blue dot) between $f(t)$ and $g(t)$, **collision region** is not empty. (b) Otherwise, the **collision region** is empty.

also consider the relationship between t and T modeled by the function $g(t) : t = T$.

In both figures Fig. 1(a) and Fig. 1(b), a bold (red) curve represents $f(t)$ and a black straight line represents $g(t)$. These two curves subdivide the space into interesting regions.

For a point $p = (t, T > t)$, indicates situations that $\mathbf{O}_i$ reaches $c(t)$ later than t . No collisions will happen because when $\mathbf{O}_i$ reaches $c(t)$, the robot $\mathbf{R}$ already passes $c(t)$.

The points $p = (t, T < f(t))$ indicates impossible situations that $\mathbf{O}_i$ needs to move faster than its maximum velocities in order to reach $c(t)$ at T .

For a point $p = (t, f(t) < T < t)$ from the region above curve $f(t)$ but below curve $t = T$, $\mathbf{O}_i$ has the ability to reach $c(t)$ earlier than $\mathbf{R}$. In order to collide with $\mathbf{R}$, $\mathbf{O}_i$ can slow down or wait at $c(t)$ until $\mathbf{R}$ arrives. We call this region the **collision region**.

Given that the robot $\mathbf{R}$ enters the path segment $\overline{c_j c_{j+1}}$ through one end point c_j at time t_j and leaves $\overline{c_j c_{j+1}}$ from the other end-point c_{j+1} at time t_{j+1} , the earliest collision time ECT_{ij} is the t coordinate of left most point of the **collision region** between t_j and t_{j+1} . Therefore if this collision region is empty, $\mathbf{R}$ and $\mathbf{O}_i$ will not collide on $\overline{c_j c_{j+1}}$.

Based on what has been discussed so far, the most important step of estimating critical moment is to compute $f(t)$, the earliest moment when $\mathbf{O}_i$ reaches $c(t)$. The shape of function $f(t)$ depends on the type and the degrees of freedom of the robot and obstacles.

In the following sections, we will discuss two examples of how $f(t)$ can be formulated when: (1) both $\mathbf{R}$ and $\mathbf{O}_i$ are points, and (2) $\mathbf{R}$ is a point and $\mathbf{O}_i$ is a polygon. From these examples, we

can build up $f(t)$ for complex polygons using the $f(t)$ of points and line segments even when rotation is considered.

4 Point-Point Case

FIGURE 2. When both $\mathbf{O}_i$ and $\mathbf{R}$ are points, their closest distance can be computed using Law of cosines in $\triangle pcj$.

To warm up our discussion, we start with a point robot $\mathbf{R}$ and a point obstacle $\mathbf{O}_i$ without rotation. Let obstacle $\mathbf{O}_i$'s current pose p coincide with its reference point o and $c(t)$ is the pose of the robot at time t . The function $f(t)$ can be simply defined as

$$f(t) = |\overline{pc(t)}|/v_i. \quad (1)$$

Since $\mathbf{R}$ moves with a given velocity, $\overline{c_j c_{j+1}} \subset \Pi$ can be linearly interpolated and every point on $\overline{c_j c_{j+1}}$ is parameterized by $0 \leq \lambda \leq 1$. So the distance L between c_j and $c(t)$ is $L = |\overline{c_j c(t)}| = \lambda |\overline{c_j c_{j+1}}|$ and, the function f can be simply written as:

$$f(t) = \sqrt{L^2 + d^2 - 2dL \cos \theta} / v_i \quad (2)$$

where $d = |\overline{pc_j}|$ and θ is the angle $\angle pc_j c_{j+1}$. This is illustrated in Fig. 2.

In order to compute the **collision region**, we need to find out the intersections of functions $f(t)$ and $g(t) = t = T$. By replacing $f(t)$ with t , we got a quadratic equation with only one variable $t = \sqrt{L^2 + d^2 - 2dL \cos \theta} / v_i$. If there exists at least one root in $[t_j, t_{j+1}]$ for the above equation (Fig. 1(a)), the **collision region** is not empty. Otherwise there will be no collision between $\mathbf{O}_i$ and $\mathbf{R}$ on path segment $\overline{c_j c_{j+1}}$ (Fig. 1(b)).

5 Point-Polygon Case

Now, we move on to the case where robot $\mathbf{R}$ is a point and obstacle $\mathbf{O}_i$ is a polygon that can translate and rotate around a given

FIGURE 3. (a) If A is fixed and B translates towards A along the closest direction, the closest features between B and A remain the same until they collide. (b) A polygon rotates around a point o . The area it sweeps over is bounded by the trajectory of every vertex (p_1 through p_4) and edges from both original and destination shapes.

reference point o . This also includes the case that $\mathbf{R}$ is translational thus can be reduced to a point via the Minkowski sum of $-\mathbf{R}$ and $\mathbf{O}_i$.

$\mathbf{R}$ and $\mathbf{O}_i$ collide when the closest distance between $\mathbf{R}$ and the boundary of $\mathbf{O}_i$ becomes zero. Let's first consider a simpler case that $\mathbf{R}$ is static and $\mathbf{O}_i$ can only translate. In this case, the trajectory that brings $\mathbf{R}$ and $\mathbf{O}_i$ together is a straight line connecting their closest features (Fig. 3(a)). During the translation, the closest features between $\mathbf{R}$ and $\mathbf{O}_i$ remain the same, and therefore identical to the point-point case discussed previously. However, when $\mathbf{O}_i$ can rotate, the closest features between $\mathbf{R}$ and $\mathbf{O}_i$ can change.

To estimate the earliest collision time (ECT), we observe that $\mathbf{O}_i$'s rotation and translation can be considered separately. That is, ECT_{ij} can be determined by analyzing the distance between $\mathbf{R}$ and the swept area of $\mathbf{O}_i$ rotating around o . Let SA_i^t be $\mathbf{O}_i$'s swept area created by rotating $\mathbf{O}_i$ with maximum angular velocity ω_i for time t as illustrated in Fig. 3(b). Because SA_i^t is the union of the swept area of every edge of $\mathbf{O}_i$, ECT_{ij} is simply the minimum among all earliest collision times of the edges in $\mathbf{O}_i$ and $\mathbf{R}$. This simple observation allows us to focus on one single edge of $\mathbf{O}_i$.

Now we consider a moving segment $\overline{p_1 p_2} \in \mathbf{O}_i$ colliding with $\mathbf{R}$. As shown Fig. 4(a), the swept area of $\overline{p_1 p_2}$ is a donut-shaped area bounded by two concentric circles centered at the reference point o traced out by p_1 and p_2 . Without loss of generality, it is assumed that p_2 forms the bigger circle.

As in point-point case, given a configuration $c(t) \in \overline{c_j c_{j+1}}$ which represents the location of $\mathbf{R}$ at time t , we are interested in solving $f(t)$ which is the earliest moment when $\overline{p_1 p_2}$ hits this $c(t)$.

FIGURE 4. (a) The swept area of $\overline{p_1 p_2}$ rotating is bounded by this grey shadowed donut-shape. The closest distance between $\overline{p_1 p_2}$ and c is realized when p_2 becomes colinear with o and c . (b) If $\overline{p_1 p_2}$ is fixed and c rotates around o with velocity $-\omega$, both the closest features and closest distance will be the same as in the case where c is fixed and $\overline{p_1 p_2}$ rotates around o with ω .

5.1 ECT of $\overline{p_1 p_2}$ and $c \in \Pi$

We separate our analysis into two cases: (1) $\overline{p_1 p_2}$ and c are sufficiently far apart, and (2) $\overline{p_1 p_2}$ and c are sufficiently close.

Let us first consider the situation that the segment $\overline{p_1 p_2}$ and the point c are *sufficiently* far apart so that when $\overline{p_1 p_2}$ moves at maximum (rotational and translational) speed, translation takes more time than rotation. In this case, the optimal motion is to translate $\overline{p_1 p_2}$ along $\overline{oc}$ while rotating $\overline{p_1 p_2}$ until p_2 is colinear with o and c . Thus, ECT of $\overline{p_1 p_2}$ and c is simply

$$(|\overline{oc}| - |\overline{op_2}|) / v_i, \text{ when } |\overline{oc}| \geq (\phi / \omega_i) v_i + |\overline{op_2}|, \quad (3)$$

where ϕ is the rotation needed to make p_2 , o and c collinear.

When c is sufficiently close to the segment $\overline{p_1 p_2}$, $\overline{p_1 p_2}$ can hit c before p_2 , o and c become collinear. Depending on the relative position of c and the swept area of $\overline{p_1 p_2}$, the motion strategy taken by $\overline{p_1 p_2}$ will be different. As illustrated in Fig. 5, there are three cases we have to analyze.

Before we detailed our analysis, we found that fixing $\overline{p_1 p_2}$ and rotating c around o significantly simplifies our discussion. That is, if $\overline{p_1 p_2}$ rotates around o with velocity ω_i , then the closest distance will not change if c rotates around o with velocity $-\omega_i$ (see Fig. 4(b)).

When c orbits around o , the closest feature between c and $\overline{p_1 p_2}$ changes among p_1 , p_2 and the points in $\overline{p_1 p_2}^\circ$, the open set of $\overline{p_1 p_2}$. If c is outside the circle traced out by p_2 (Fig. 5(a)), the closest feature can change four times from p_2 to $\overline{p_1 p_2}^\circ$ to p_1 to $\overline{p_1 p_2}^\circ$ and back to $\overline{p_1 p_2}^\circ$. If c overlaps with the swept area of $\overline{p_1 p_2}$ (Fig. 5(b)), the closest feature changes twice between p_1

FIGURE 5. Three cases when c is sufficiently close to the segment $\overline{p_1p_2}$

and $\overline{p_1p_2}^\circ$. If c is inside the circle traced out by p_1 (Fig. 5(c)), the closest feature also changes twice between p_1 and $\overline{p_1p_2}^\circ$.

Determining these closest feature changes (i.e., α and β in Fig. 5) is straightforward; they are the intersections between the circle traced out by c (around o) and the lines containing p_1 or p_2 and perpendicular to $\overline{p_1p_2}$.

If we let the closest distance between c and $\overline{p_1p_2}$ be a function $d(t)$ of time (see Appendix for more detailed definitions of $d(t)$ for all three cases of Fig. 5), and let t_T be the time that the point c needs to translate at velocity v_i , and let t_R be the time that c needs to rotate at velocity $-\omega_i$. Because t_T is a function of t_R , we let $t_T = h_T(t_R) = d(t_R)/v_i$, where $d(t_R)$ is the distance between c and segment $\overline{p_1p_2}$ when p rotates $\theta = t_R\omega$ around o . The ETC between p and $\overline{c_1c_2}$ is:

$$\text{ECT} = \arg \min_{t_R} (\max(t_R, h_T(t_R))) \quad (4)$$

$$= \arg \min_{t_R} (|t_R - h_T(t_R)|) . \quad (5)$$

Therefore, ECT is t_R such that $t_R = d(t_R)/v_i$. In other words, since both translation and rotation decrease the closest distance

between $\mathbf{R}$ and $\mathbf{O}_i$, in order to detect the earliest collision time, t_T must equal t_R .

5.2 ECT of $\overline{p_1p_2}$ and $\overline{c_1c_2} \subset \Pi$

The discussion in Section 5.1 allows us to partition an edge $\overline{c_1c_2} \subset \Pi$ into subsegment such that all configurations in each subsegment belong to one of the four classes identified in the previous section, i.e., sufficiently far, or case (a), (b) or (c) in Fig. 5. More specifically, if we relate time t to α and β in Fig. 5, we can get a plot similar to Fig. 6. In Fig. 6(a), if the robot $\mathbf{R}$ is between c_1 and c , the closest feature between $\overline{p_1p_2}$ and $\mathbf{R}$ is always p_1 . If $\mathbf{R}$ is between c and c' , the closest feature can change from p_1 to $\overline{p_1p_2}$. If $\mathbf{R}$ is between c' and c_2 , the closest feature between $\overline{p_1p_2}$ and $\overline{c'c_2}$ can change four times. If there exists a configuration c'' between c' and c_2 that is sufficiently away (not shown in Fig. 5), then the closest feature between $\overline{p_1p_2}$ and $\overline{c''c_2}$ is always p_2 . The detailed formulation of α and β over t is discussed in Appendix (B).

Recall that our goal is to determine the time of earliest collision for every configuration on $\overline{c_1c_2}$, i.e., the function $f(t)$. For the subsegments (e.g. $\overline{c_1c}$ and $\overline{c''c_2}$) that the closest feature does not change, the function $f(t)$ of the subsegments is simply $t_R = d(t_R, p)/v_i$, where the point p is p_1 or p_2 and $d(t_R, p)$ is the distance between p and the configuration of the robot at time t_R . The function $d(t_R, p)$ is detailed in Appendix. For the subsegments (e.g. $\overline{cc'}$) that the closest features change with rotation, the function $f(t)$ of the subsegments can be determined by combining $t_R = d(t_R, p)/v_i$ and $t_R = d(t_R, \overline{p_1p_2})/v_i$ that is valid only in the gray area shown in Fig. 6(a).

Therefore, the function $f(t)$ for the segment $\overline{c_1c_2}$ can be determined in the piecewise fashion by solving t_R in each of these subsegments. Fig. 7 show an example on the function $f(t)$ and the closest distance between an edge of an obstacle and the robot over time.

Similar to the analysis that we have done in Section 5, when $f(t) = t$, $\mathbf{O}_i$ collides with $\mathbf{R}$ at maximum velocity and $f(t) < t$ means $\mathbf{O}_i$ can collide with $\mathbf{R}$ at location c if $\mathbf{O}_i$ slows down. Otherwise, $\mathbf{O}_i$ cannot reach c before $\mathbf{R}$ already passes c . Therefore, we are interested in detecting the *collision region* which is above $t_R = f(t)$ and below $t = t_R$ and also bounded by $t = t_1$ and $t = t_2$.

Note that, although the function $f(t)$ can be complex, the intersections can be determined by trust-region-based root-finding methods such as Levenberg-Marquardt algorithm or the Dogleg algorithm.

FIGURE 6. (a) The relationship between t_R and α and β when $\mathbf{R}$ moves from c_1 to c_2 . (b) and (c) illustrate the configurations c and c' in (a). L_1 and L_2 are the lines perpendicular to $\overline{p_1 p_2}$ and contain p_1 and p_2 , respectively.

FIGURE 7. (a) The plot for $f(t)$ when the closest feature to c locates on $\overline{p_1 p_2}^\circ$. The collision region is bounded by the two leftmost intersections. (b) The 3D plot shows the closest distance between an edge of some obstacle and the robot changes over time.

FIGURE 8. Three testing environments. In all environments, the red dot indicates start position and the green dot indicates goal position. Darker obstacles are static and lighter obstacles are dynamic. The maximum linear and angular velocity are, respectively, (a) 5 m/s and 1 radian/s (b) 4 m/s and 2 radian/s and (c) 4 m/s and 2 radian/s.

TABLE 1. Comparison on number of updates/replannings

Test environment in Fig. 8	Env (a)	Env (b)	Env (c)
Conservative-advancement (CA)	11	10	15
classic fixed-time (FT)	124	145	122

6 Experiments and Evaluation

Experiment Set Up. We implemented the collision prediction method (in C++ and Matlab) and tested on three environments (shown in Fig. 8). These environments contain a point robot and both static and dynamic obstacles. For a dynamic obstacle, its motion is simulated using Box2D by exerting random forces. The robot knows where static obstacles are but has no idea of how a dynamic obstacle moves except for its maximum velocities. The only way for the robot to know the pose of a dynamic obstacle is through its (simulated) onboard sensors.

Comparison. In this comparative study, we use a naïve replanning strategy that simply finds a new path from scratch by building an RRT [34]. Every time after the robot updates its belief of the environment, the old RRT is abandoned and a new tree is created based on the current state of workspace. This repeats until robot reaches the goal. It is important to note that the experimental results are independent of the replanning strategy.

Table I shows the number of updates from the conservative-advancement (CA) strategy and fixed-time (FT) strategy. All data are the average over 10 runs. For the FT strategy, the fixed time interval is 0.1 for every environment from Fig. 8. The CA strategy reduces redundant updates significantly (more than 90%). Moreover, the system using FT strategy sometimes terminates early before the robot reaches goal due to infrequent updates, which causes the robot to fail to detect the incoming danger and collide with some obstacle. While the system using CA strategy find valid path for all runs, the successful rate (the number of

successful runs divided by the total number of runs) using the FT strategy is around 62.5% for Fig. 8(a), 58.8% for Fig. 8(b) and 71.4% for Fig. 8(c).

7 Conclusion

In this paper, we proposed an adaptive method that predicts collision for obstacles with unknown trajectories. We believe that collision prediction has many potential usages and advantages. Similar to collision detection in the setting of known obstacle motion, we have shown that collision prediction allows the robot to evaluate the safety of each edge on the roadmap with unknown obstacle motion. When the robot travels on a predetermined path, collision prediction enables adaptive replanning period that allows more robust and efficient replanning. While a naïve replanner is used in our experiment, the proposed method can also be used with various existing replanning methods to provide better performance. Moreover, even though the obstacles are modeled as adversarial agents, this tool can easily incorporate the constraints in obstacle's motion when better behavior patterns of the obstacle are known [10]. In some sense the work presented in this paper provides a method compliment to our previous work on critical roadmap [35] that identifies topological changes in free configuration space for obstacles whose trajectories are known to the robot. Our framework is based on the collision prediction between elements of the robot and obstacles. Therefore we are currently extending our framework to more complicated cases where both $\mathbf{R}$ and $\mathbf{O}_i$ are arbitrary polygons, polyhedra or articulated robots.

REFERENCES

- [1] Erdmann, M., and Lozano-Perez, T., 1986. "On multiple moving objects". *Algorithmica*, 2, pp. 1419–1424.
- [2] Kavraki, L. E., Svestka, P., Latombe, J. C., and Overmars, M. H., 1996. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". *IEEE Trans. on Robotics and Automation*, 12(4), August, pp. 566–580.
- [3] LaValle, S. M., and Kuffner, J. J., 1999. "Randomized kinodynamic planning". In Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 473–479.
- [4] Jaillet, L., and Simeon, T., 2004. "A prm-based motion planner for dynamically changing environments". In Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS), pp. 1606–1611.
- [5] Kallman, M., and Mataric, M., 2004. "Motion planning using dynamic roadmaps". In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, Vol. 5, IEEE, pp. 4399–4404.
- [6] Li, T.-Y., and Shie, Y.-C., 2002. "An incremental learning approach to motion planning with roadmap management". In Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 3411–3416.
- [7] Ferguson, D., Kalra, N., and Stentz, A., 2006. "Replanning with rrt's". In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, IEEE, pp. 1243–1248.
- [8] Khatib, O., 1986. "Real-time obstacle avoidance for manipulators and mobile robots". *Int. Journal of Robotics Research*, 5(1), pp. 90–98.
- [9] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S., 2005. "Anytime dynamic a*: An anytime, replanning algorithm".
- [10] van den Berg, J., Ferguson, D., and Kuffner, J., 2006. "Anytime path planning and replanning in dynamic environments". In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2366 – 2371.
- [11] van den Berg, J., and Overmars, M., 2006. "Planning the shortest safe path amidst unpredictably moving obstacles". In Proc. Int. Workshop Alg. Found. Robot.(WAFR).
- [12] Wzorek, M., Kvarnstrom, J., and Doherty, P., 2010. "Choosing path replanning strategies for unmanned aircraft systems".
- [13] Hayward, V., Aubry, S., Foisy, A., and Ghallab, Y., 1995. "Efficient collision prediction among many moving objects". *Internat. J. Robot. Res.*, 14(2), pp. 129–143.
- [14] Hubbard, P. M., 1995. "Collision detection for interactive graphics applications". PhD thesis.
- [15] Chakravarthy, A., and Ghose, D., 1998. "Obstacle avoidance in a dynamic environment: A collision cone approach". *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(5), pp. 562–574.
- [16] Kim, H. K., Guibas, L. J., and Shin, S. Y., 2005. "Efficient collision detection among moving spheres with unknown trajectories". *Algorithmica*, pp. 195–210.
- [17] Du Toit, N., and Burdick, J., 2010. "Robotic motion planning in dynamic, cluttered, uncertain environments". In Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE, pp. 966–973.
- [18] Hauser, K., 2011. "Randomized belief-space replanning in partially-observable continuous spaces". *Algorithmic Foundations of Robotics IX*, pp. 193–209.

- [19] Shiller, Z., Gal, O., and Raz, A., 2011. “Adaptive time horizon for on-line avoidance in dynamic environments”. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, pp. 3539–3544.
- [20] Luders, B. D., Aoude, G. S., Joseph, J. M., Roy, N., and How, J. P., 2011. Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns. Tech. rep., MIT Aerospace Control Laboratory: Technical Reports.
- [21] Fraichard, T., and Asama, H., 2003. “Inevitable collision states. a step towards safer robots?”. In Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, Vol. 1, IEEE, pp. 388–393.
- [22] Martinez-Gomez, L., and Fraichard, T., 2009. “Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation”. In Robotics and Automation, 2009. ICRA’09. IEEE International Conference on, IEEE, pp. 100–105.
- [23] Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J., and Fiore, G., 2009. “Real-time motion planning with applications to autonomous urban driving”. *Control Systems Technology, IEEE Transactions on*, **17**(5), pp. 1105–1118.
- [24] Bouraine, S., Fraichard, T., and Salhi, H., 2011. “Relaxing the inevitable collision state concept to address provably safe mobile robot navigation with limited field-of-views in unknown dynamic environments”. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, pp. 2985–2991.
- [25] Lalish, E., and Morgansen, K., 2008. “Decentralized reactive collision avoidance for multivehicle systems”. In Decision and Control, 2008. CDC 2008. 47th IEEE Conference on, IEEE, pp. 1218–1224.
- [26] Bautin, A., Martinez-Gomez, L., and Fraichard, T., 2010. “Inevitable collision states: A probabilistic perspective”. In Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 4022 – 4027.
- [27] Wu, A., and How, J. P., 2012. “Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles”. *Autonomous Robots*, pp. 227–242.
- [28] Cohen, J., Lin, M. C., Manocha, D., and Ponamgi, M., 1995. “I-collide: An interactive and exact collision detection system for large-scale environment”. In Symposium on Interactive 3D Graphics, pp. 189–196.
- [29] Gottschalk, S., Lin, M. C., and Manocha, D., 1996. “OBB-Tree: A hierarchical structure for rapid interference detection”. *Computer Graphics*, **30**, pp. 171–180.
- [30] Baraff, D., 1990. “Curved surfaces and coherence for non-penetrating rigid body simulation”. *Comput. Graph.*, **24**(4), pp. 19–28.
- [31] Hahn, J. K., 1988. “Realistic animation of rigid bodies”. *Comput. Graph.*, **22**(4), pp. 299–308.
- [32] Lien, J.-M., 2008. “Hybrid motion planning using Minkowski sums”. In Proc. Robotics: Sci. Sys. (RSS).
- [33] Lien, J.-M., and Lu, Y., 2009. “Planning motion in similar environments”. In Proceedings of the Robotics: Science and Systems Conference (RSS).
- [34] Lavalley, S. M., 1998. Rapidly-exploring random trees: A new tool for path planning. Tech. rep., Iowa State University.
- [35] Lu, Y., and Lien, J.-M., 2011. “Finding critical changes in dynamic configuration spaces”. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 2626–2631.

A Appendix

A.1 Distance Function $d(t)$

We first consider that case that the closest feature from $\overline{p_1 p_2}$ is p_1 . Since p_1 is fixed and $c(t)$ rotates around o with $-\omega$,

$$d(t) = |\overline{c(t)p_1}|.$$

Therefore,

$$d^2(t) = (x_t - a_1)^2 + (y_t - a_2)^2,$$

where $c(t) = (x_t, y_t)$ and $p_1 = (a_1, a_2)$. By interpolating $c(t)$ with c_j and c_{j+1} , $d(t)$ is further represented as follows.

$$d^2(t) = |\overline{op_1}|^2 + |\overline{oc}|^2 - 2 \cos \theta (a_1 x + a_2 y) + 2 \sin \theta (a_1 y - a_2 x) \quad (6)$$

$c = [x, y]$ is parameterized by $0 \leq \lambda \leq 1$ (or $t_1 \leq t \leq t_2$). Let $c_1 = [x_1, y_1]$ and $c_2 = [x_2, y_2]$, then

$$\begin{aligned} x &= x_1 + \lambda(x_2 - x_1) \\ y &= y_1 + \lambda(y_2 - y_1) \end{aligned}$$

To make it easier, $\overline{c_1 c_2}$ is rotated to be aligned with x -axis. Then

$$y = y_2 = y_1.$$

In order to detect ECT, we need to solve equation

$$\begin{aligned} d^2(t_R) &= (v_i t_R)^2 \\ &= |\overline{op_1}|^2 + |\overline{oc}|^2 - 2 \cos \theta (a_1 x + a_2 y) + 2 \sin \theta (a_1 y - a_2 x). \end{aligned} \quad (7)$$

where $\theta = \omega t_R$.

Equation 7 is a complex function with trigonometric functions and polynomials. It can be solved with trust-region methods such as Levenberg-Marquardt algorithm or the Dogleg algorithm. Since both are built-in algorithms in Matlab, therefore we just call **fsolve** in Matlab to solve the equation 7.

If $d(t)$ is defined between $\overline{p_1 p_2}^\circ$ and c , then $d(t)$ is the distance from c to the straight line containing $\overline{p_1 p_2}$. To make it simpler, $\overline{p_1 p_2}$ is rotated to be aligned with $y = x$ and $\overline{c_1 c_2}$ is updated accordingly. This line is easily computed and let it be

$$y = x + b.$$

Then $d(t)$ can be written as

$$d^2(t) = \frac{(x_t - y_t + b)^2}{2}.$$

By replacing x_t and y_t with the equation above, $d(t)$ can be represented as follows.

$$\begin{aligned} d^2(t) &= \frac{|\overline{oc}|^2 + b^2 + \sin 2\theta (x^2 - y^2)}{2} \\ &\quad - xy \cos 2\theta + b \cos \theta (x - y) + b \sin \theta (x + y) \end{aligned} \quad (8)$$

A.2 Function of α and β over time

To detect α and β , we further simplify our notation, we assume that the reference point o is the origin and $\overline{p_1 p_2}$ is aligned with the x -axis. Let $p_1 = [a_1, a_2]$ and $p_2 = [b_1, b_2]$. The line passing through p_1 and perpendicular to $\overline{p_1 p_2}$ is $L_1 : x = a_1$. The straight line passing through p_2 and perpendicular to $\overline{p_1 p_2}$ is $L_2 : x = b_1$. Let t_R be the moment when c reaches α_1 or β_1 , then t_R satisfies the following constraint:

$$\cos(\omega t_R) c_x + \sin(\omega t_R) c_y = a_1. \quad (9)$$

Similarly, the moment t_R when c reaches α_2 or β_2 satisfies

$$\cos(\omega t_R) c_x + \sin(\omega t_R) c_y = b_1. \quad (10)$$

Fig. 6(a) shows the plots of equations 9 and 10. It illustrates how t_R changes as $\mathbf{R}$ moves from c_1 to c_2 . Of course, both functions 9 and 10 are periodical because t_R is the variable of Trigonometric functions. We are only interested in the values of t_R in the first period due to ECT.