Collision Prediction Among Rigid and Articulated Obstacles with Unknown Motion *

Yanyan Lu, Zhonghua Xi, and Jyh-Ming Lien

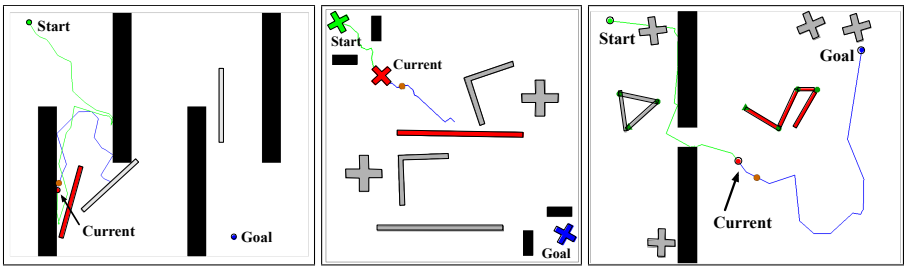
Department of Computer Science
George Mason University
Fairfax, Virginia, USA 22030
Email : {ylu4,zxi}@gmu.edu, jmlien@cs.gmu.edu

Abstract. Collision prediction is a fundamental operation for planning motion in dynamic environment. Existing methods usually exploit complex behavior models or use dynamic constraints in collision prediction. However, these methods all assume simple geometries, such as disc, which significantly limit their applicability. This paper proposes a new approach that advances collision prediction beyond disc robots and handles arbitrary polygons and articulated objects. Our new tool predicts collision by assuming that obstacles are *adversarial*. Comparing to an online motion planner that replans periodically at *fixed time interval* and planner that approximates obstacle with discs, our experimental results provide strong evidences that the new method significantly reduces the number of replans while maintaining higher success rate of finding a valid path. Our geometric-based collision prediction method provides a tool to handle highly complex shapes and provides a complimentary approach to those methods that consider behavior and dynamic constraints of objects with simple shapes.

1 Introduction

Imagine a scenario where a robot navigates itself through a disaster zone filled with static obstacles, mobile robots carrying debris with various sizes and shapes and mobile manipulators picking up and loading debris on top of the mobile robots or conveyor belts. In this scenario, the robot must plan its path without knowing how the other robots will move. Similar scenarios can be found in factory, warehouse or airport where a robot requires the same ability to navigate among other mobile robots manipulating and carrying commercial goods with various sizes and shapes. Fig. 1 illustrates three of such examples where a mobile robot, which is modeled either as a point or a polygon, navigates through environments filled with static and dynamic obstacles with various shapes. In motion planning literature, this problem is usually known as *online motion planning* or *sensor-based motion planning*.

* This work is supported in part by NSF IIS-096053, CNS-1205260, EFRI-1240459, AFOSR and USGS.



(a) Point robot (b) Polygonal robot (c) Articulated obstacles

Fig. 1. Three examples of a mobile robot moving from corner to corner through environments filled with static (black) and dynamic (grey) obstacles whose motion is unknown to the robot. Bounding these moving obstacles with circles can lead to poor collision prediction and result in many unnecessary replanning. Our method predicts the collision time for obstacles with arbitrary shapes including articulated objects. The obstacles shown in red are the ones with the earliest collision times with respect to the current configurations of the robot (also shown in red).

Online motion planning methods usually exploit the idea of temporal coherence to gain better efficiency by repairing the invalid portion of the path or (tree-based or graph-based) roadmaps [1,2,3,4] since the changes in the configuration space is usually small from frame to frame. These planning strategies are often known as *replanning methods* [5,6,7,8,9]. 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.

Motivated by this issue, several strategies [10,11,12,13,8,14,15] have been proposed to replan *adaptively* only at the critical moments when the robot and obstacles may collide. These critical moments are usually detected by *collision prediction* methods. The main challenge in predicting collision steams from the assumption that obstacle's motion is unknown. Existing methods in collision prediction exploit complex behavior prediction [14,15] or consider dynamic constraints [10,11,13,16]. However, these methods all assume either translational or disc objects, which significantly limit their applicability. This limitation seriously hinders the robot's ability to move in cluttered environments, such as those in the aforementioned scenearios and the examples in Fig. 1, where moving obstacles can have arbitrary shapes and sizes and can even be articulated objects. As we will show later, bounding these moving obstacles with discs can lead to arbitrarily poor collision estimation.

In this paper, we propose a new geometric tool that advances collision prediction beyond the translational and disc objects and can handle arbitrary polygons

and articulated objects. 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 actively determine its next replanning time by conservatively estimating the amount of time (i.e., *earliest collision time* or simply ECT) that it can stay on the planned path without colliding with the obstacles. The idea of ECT and conservative advancement are detailed in Section 3. In Sections 4 to 6, we discuss how ECT can be formulated in the three scenarios illustrated in Figs. 1(a), 1(b) and 1(c), respectively. Our prediction is determined only based on the last known positions of the obstacles and their maximum linear and angular velocities. In the experimental results (in Section 8), we demonstrate that an online planner using the proposed collision prediction method significantly reduces the number of replannings while maintaining the same or higher success rate of finding a valid path than (1) planner that replans periodically at fixed time intervals and (2) planner that bounds obstacles with circles. In essence, our main contribution is a geometric-based collision prediction method that can handle highly complex shapes. This tool provides a complimentary approach to the methods that consider complex behavior prediction or handle dynamic constraints but with only simple shapes.

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. Fraichard and Asama [17] 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. [18] 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 [19] 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. [20] 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 [16] extended VO to moving obstacles with constrained dynamics but move unpredictably. To compute the velocity obstacle of an obstacle, 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. Computation of ICS or VO requires some information about the future in the environment. When it comes to environments whose future is completely unpredictable, methods applying ICS or VO may fail to avoid approaching collisions.

The work closes to the spirit of our new method is by van den Berg and Overmars [8]. 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 [7,21,22,23,24]. For example, van den Berg et al. [7] 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 very 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. [10], Kim et al. [13] and Hubbard [11] 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 intersect are further checked for collisions using sphere trees that represent the objects. Kim et al. [13] 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 volume*s intersect, they are checked for actual collision. Chakravarthy and Ghose [12] 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 of Our Method

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*. In the rest of this paper, we assume that the robot $\mathbf{R}$ plans a path Π based on its current belief of the state of the workspace. However, instead of determining if Π is 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 $\triangle t$ before this critical moment t to update its belief and replan if necessary. We would like to emphasize again that this setting is merely a framework among many other applications of collision prediction that allows us to make our discussion more concrete.

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 the traditional motion planning methods, 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 its shape, the maximum translational velocity v_i and a maximum angular velocity ω_i around a given reference point.

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 $\Pi = \{c_1, c_2, ..., c_n\}$ 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 on $\overline{c_j c_{j+1}}$. 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 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 all its subsequent segments $\overline{c_k c_{k+1}}$ with $j < k < n$. In Section 3.1, we will provide an 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 move on Π at time 0.

Since the robot $\mathbf{R}$ moves along a known path Π, $\mathbf{R}$ knows when it reaches any 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 it maximum linear and angular velocities v_i and ω_i, there must exist an earliest time $\hat{T}$ for $\mathbf{O}_i$ to reach any $c \in \overline{c_1 c_2}$ 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 also consider the relationship between t and T modeled by the function $g(t) : t = T$. See Figs. 2(a) and 2(b).

In Fig. 2(a), a bold (red) curve represents the earliest arrival time $f(t)$ and a black straight line represents $g(t)$. These two curves subdivide the space into interesting regions.

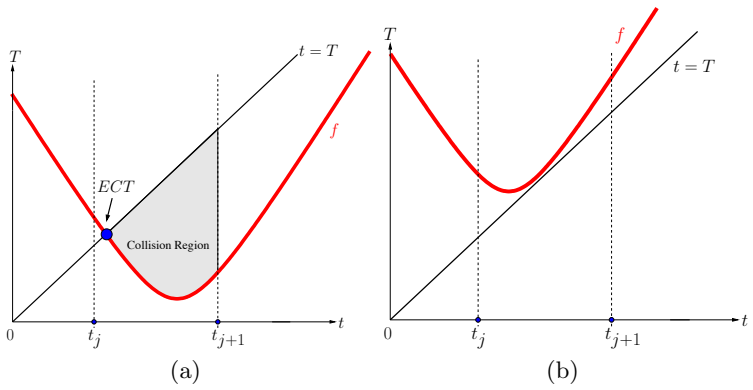


Fig. 2. The red (thicker) curves in both figures are plots of the earliest arrival time $f(t)$ for an obstacle. Black straight lines are plots of $g(t) : t = T$. (a) When there is at least one intersection (blue dot) between $f(t)$ and $g(t)$, collision region is not empty. (b) Otherwise, the collision region is empty.

- 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 endpoint 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 three examples of how $f(t)$ can be formulated when: (1) $\mathbf{R}$ is a point and $\mathbf{O}_i$ is a polygon, (2) both $\mathbf{R}$ and $\mathbf{O}_i$ are polygons and (3) $\mathbf{R}$ is a point and $\mathbf{O}_i$ is an articulated object. From these examples, we can build up $f(t)$ for complex shapes even when rotation is considered.

4 Point-Polygon Case

Let us start with 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 reference point o. Without loss of generality, let us focus on a moving segment $\overline{p_1p_2} \in \mathbf{O}_i$. As in point-point case, given a configuration $c(t) \in \overline{c_jc_{j+1}}$ which represents the location of $\mathbf{R}$ at time t, we are interested in solving $f(t)$, the earliest moment when $\overline{p_1p_2}$ hits $c(t)$.

To estimate the earliest collision time (ECT), we observe that $\mathbf{O}_i$'s rotation and translation can be considered separately. That is, $f(t)$ can be decomposed into translational and rotational components: t_T, the time that the point c needs to translate at velocity v_i, and t_R, the time that c needs to rotate at velocity ω_i. If we let the closest distance between $c(t)$ and $\overline{p_1p_2}$ be a function $\mathrm{d}(t)$ of time, we can compute ECT between $\overline{p_1p_2}$ and $\mathbf{R}$ moving from configuration c_1 to configuration c_2 using the following lemma.

Lemma 2. *The ECT between $\overline{p_1p_2}$ and $\overline{c_1c_2}$ is:*

$$\mathrm{ECT} = \arg\min_{t_R}(|t_R - t_T|) = \arg\min_{t_R}(|t_R - \mathrm{d}(t_R)/v_i|) \ , \tag{1}$$

where $\mathrm{d}(t_R)$ is the distance between $c(t_R) \in \overline{c_1c_2}$ and segment $\overline{p_1p_2}$ when $\overline{p_1p_2}$ rotates $\theta = t_R\omega$ around o.

Proof. The key to this proof is the definition of the function $\mathrm{d}(t)$. In our analysis, $\mathrm{d}(t)$ depends on two cases: (1) $\overline{p_1p_2}$ and $c \in \overline{c_1c_2}$ are sufficiently far apart, and (2) $\overline{p_1p_2}$ and c are sufficiently close. Details of the analysis can be found in the Appendix A of this paper. ■

In summary, to estimate the ECT of $\mathbf{R}$ and $\mathbf{O}_i$, we decompose $f(t)$ into translational and rotational components: t_T and t_R and solve the optimization problem in Lemma 2. Since both translation and rotation decrease the closest distance between $\mathbf{R}$ and $\mathbf{O}_i$, the time spend on translation t_T must equal the time spend on rotation t_R. Again, interested readers should refer to appendix and our technical report [25] for detail.

5 Polygon-Polygon Case

In this section, we briefly discuss the case that both the robot $\mathbf{R}$ and the obstacle $\mathbf{O}_i$ are polygons. The robot $\mathbf{R}$ rotates around its center of mass and moves alone the designated path Π. Obstacle $\mathbf{O}_i$ undergoes unknown translation and rotation around a given reference point o.

Taking the same conservative advancement approach, we will focus our discussion on the motion strategy that an edge $\overline{q_1q_2}$ of $\mathbf{O}_i$ can take to collide with an edge $\overline{p_1p_2}$ of $\mathbf{R}$ at a given time t. Our main observation of computing the ECT etween is stated in the following lemma.

Lemma 3. *Given two separating line segments $\overline{p_1p_2} \in \mathbf{R}$ and $\overline{q_1q_2} \in \mathbf{O}_i$, the earliest collision can only happen between an endpoint of $\overline{p_1p_2}$ and $\overline{q_1q_2}$ or an endpoint of $\overline{q_1q_2}$ and $\overline{p_1p_2}$. Collisions at the interior portion from both line segments can only happen after one of those two cases.*

Proof. See detailed proof in technical report [25]. ■

Essentially, Lemma 3 allows us to determine the ECT of two line segments from only two instances of *point-polygon case* discussed in Section 4. Given that $\mathbf{R}$ and $\mathbf{O}_i$ are composed of n and m line segments, respectively, their ECT can be determined via $2mn$ point-polygon case analysis.

6 Point-Articulated Obstacle Case

Let us now focus on collision prediction between a point robot and an articulated obstacle in 2D, such as a mobile manipulator. The motion of such an articulated obstacle $\mathbf{O}_i$ is unknown to the robot but constrained by the following assumptions: (1) $\mathbf{O}_i$ can translate as a rigid body and it has a maximum translational velocity v_i, and (2) every two adjacent linkages are connected by a revolute joint. In addition, every revolute joint has a maximum angular velocity ω_j. To simplify our discussion, we assume that $\mathbf{O}_i$ has no branch and is represented as a sequence of m linkages $\mathbf{O}_i = L_1L_2 \dots L_m$. Linkage L_i is closer to the base than linkage L_j *iff* $1 \leq i < j \leq m$, and we call L_i an ancestor of L_j.

We again are interested in detecting the earliest time when collisions occurred between a point robot and such an articulated obstacle $\mathbf{O}_i$. Let us first assume the robot reaches c_1 at t_1 and reaches c_2 at t_2 where $\overline{c_1c_2}$ is some path segment on its current path Π. Note that the earliest collision time that we want to predict needs to fall into the range $[t_1, t_2]$ because we consider each path segment on Π separately, and at this moment we are only interested in detecting collisions if the robot is on $\overline{c_1c_2}$. Our main observation of the ECT between $\mathbf{O}_i$ and $\overline{c_1c_2}$ is stated in the following lemma.

Lemma 4. *The computation of ECT between $\mathbf{O}_i$ and $\overline{c_1c_2}$ is decomposable w.r.t. the linkages of $\mathbf{O}_i$. Let $\mathbf{O}_i^k$ be a subset of $\mathbf{O}_i$ including the linkages between L_1 and L_k, i.e., $\mathbf{O}_i^k = L_1L_2 \dots L_{k \leq m}$, then ECT($\mathbf{O}_i, \overline{c_1c_2}$) can be written as:*

$$\mathrm{ECT}(\mathbf{O}_i, \overline{c_1c_2}) = \min_{1 \leq k \leq m} (\mathrm{ECT}(\mathbf{O}_i^k, \overline{c_1c_2})) = \min_{1 \leq k \leq m} (\mathrm{ECT}(L_k, \overline{c_1c_2})) \ . \quad (2)$$

Note that, in $\mathrm{ECT}(L_k, \overline{c_1c_2})$*, the motion of L_k is constrained by $\mathbf{O}_i^{k-1}$.*

Proof. We will provide a proof sketch here. See detailed proof in technical report [25]. Let us start from the first linkage L_1. Without considering other linkages, we can compute the earliest time t_1 when L_1 hits the robot using ideas from

Section 4. Now we move on to the next linkage L_2. Considering only linkage L_2 (whose motion is dependent on linkage L_1), the earliest collision at time t_2 between L_2 and $\overline{c_1c_2}$) without considering collision status between L_1 and $\overline{c_1c_2}$ can also be determined through a similar formulation from Section 4 (see details in Appendix B). Then there are only two possible cases: (1) L_1 hits $\overline{c_1c_2}$ earlier than L_2 and (2) L_2 hits $\overline{c_1c_2}$ earlier than L_1. Both cases can be summarized into $\min(t_1, t_2) = \min(\text{ECT}(L_1, \overline{c_1c_2}), \text{ECT}(L_2, \overline{c_1c_2}))$. This analysis process repeats for all successive links, and then we can conclude that $\text{ECT}(\mathbf{O}_i, \overline{c_1c_2}) = \min_{1 \le k \le m}(\text{ECT}(L_k, \overline{c_1c_2}))$. ■

In summary, Lemma 4 allows us to reduce the computation of the ECT between a point robot and an articulated obstacle of m linkages into m cases of point-polygon analysis.

7 Planning Motion Using Predicted Collision

So far we assume that the robot only stays on a given path. In this section, we show how to use the predicted collision in a motion planner. It is important to note that this RRT-based planner [26] discussed below is merely an example to show how earliest collision time (ECT) can be used. Other planners, such as PRM-based planners can also be combined with ECT.

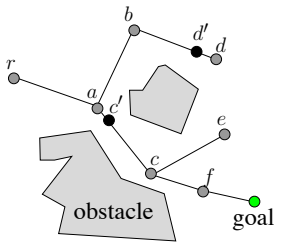


Fig. 3. An RRT augmented with earliest collision time. The tree is rooted at current configuration r of the robot. Configurations c' and d' are the predicted earliest collision locations on the paths from r to c and d, respectively.

In general, there are two desirable properties when a robot plans a path. First, a path should bring the robot near the goal. Second, the path should remain safe (valid) for as long as possible. With these two properties in mind, we propose to augment RRT with predicted collision. More specifically, the RRT is constructed as usual but each path from the root to a leaf is now associated with an ECT. The best path is then a path in the RRT that has the *latest* ECT while still reduces the geodesic distance between the robot and the goal. An example of an augment RRT is shown in Fig. 3. In this example, paths from configuration r to all leaves reduce the distance to the goal but the path π_d to configuration d has the latest ECT, thus π_d is the best path.

8 Experimental Results

We implemented the collision prediction method in C++ using Eigen linear algebra library and NLopt library. Experimental results reported in this paper

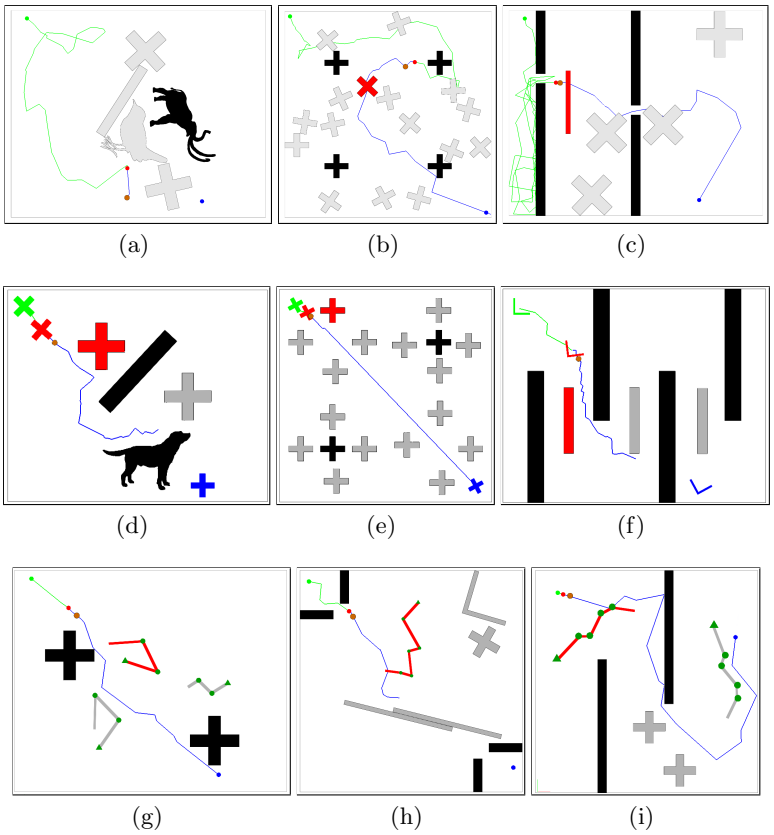


Fig. 4. (a-c) Point-polygon environments. (d-f) Polygon-polygon environments. (g-i) Point-articulated environments. In all environments, the green robot and blue robot indicate start configuration and goal configuration, respectively. The red robot indicates the current configuration and the obstacles which cause earliest collisions are colored in red. Black obstacles are static and light grey obstacles are dynamic.

are obtained from a workstation with two Intel Xeon E5-2630 2.30GHz CPUs and 32GB memory. We tested our implementation in 12 environments shown in Figs. 1 and 4. These environments contain both static and dynamic obstacles. For a dynamic obstacle, its motion is simulated using Box2D physics engine by exerting random forces. The robot knows the locations of static obstacles and the maximum translational velocity and angular velocity of dynamic obstacle. The only way that the robot knows the pose of a dynamic obstacle is through its (simulated) onboard sensors. The best way to visualize the environments is via animation. We encourage the reader to view the videos at *http://masc.cs.gmu.edu/wiki/ECT*.

8.1 Compare to a Fixed-Time Strategy

In our first experiment, we compare two planning strategies: One replans adaptively based on collision prediction using augmented RRT (see Section 7), and the other replans periodically at fixed time interval using regular RRT.

Fig. 5 shows the *success rate* and *number of replans* obtained from environments in Fig. 4. The *success rate* is the number of runs that robot reaches the goal over the total number of runs, and the *number of replans* is the number of times that the robot replans to reach the goal. The maximum translational velocity of an obstacle is set to $2m/s$ and the maximum angular velocity is set to 3 $radians/s$. The experiments are conducted for multiple situations when robot's velocity is 1, 2, 4, 8 and $16m/s$. Each data point is collected over 500 runs (i.e. 100 runs for each environment).

Success Rate and Number of Replans. From the plots in Fig. 5, we show that our approach using predicated collision helps the robot achieve nearly optimal success rate with a small number of replans. First, let us look at Fig. 5(a), Fig. 5(c), and Fig. 5(e). We see that the success rate of the proposed method is almost identical to or even better than the fixed-time strategy with very high (and almost unrealistic) replanning frequency (i.e. replan every 0.05 sec.). This is especially clear when the robot's velocity is greater than $2m/s$. However, frequent updates introduce a large number of replans. As shown in Fig. 5(b), Fig. 5(d) and Fig. 5(f), in order to provide a success rate similar to the proposed method, the fixed-time strategy needs to replan around 100 times more.

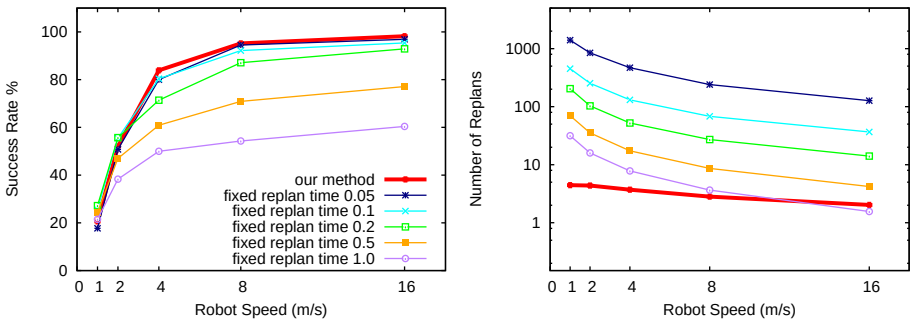
Table 1. Average running time in seconds.

Method	**Time (sec)**
Our method	2.68
Replan every 0.05 sec	25.70
Replan every 0.1 sec	8.76
Replan every 0.2 sec	4.00
Replan every 0.5 sec	1.66
Replan every 1.0 sec	0.97

Running Time. In Table 1, we provide average computation times spent on replanning over five environments for rigid obstacles. We observe that, to achieve similar success rate, our method runs 3 and 12 times faster than fixed-time strategy with time steps 0.1 and 0.05 sec, respectively.

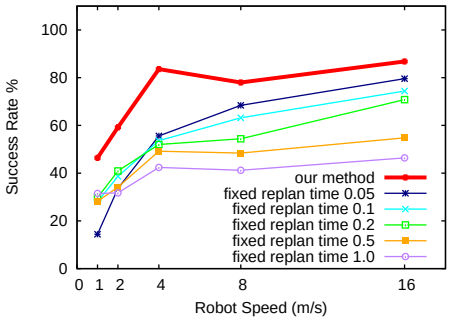
8.2 Compare to an Optimal Strategy

We further compare our method to a conservative optimal strategy [8]. In their work, every obstacle must be a disc and its swept volume over time is conserva-

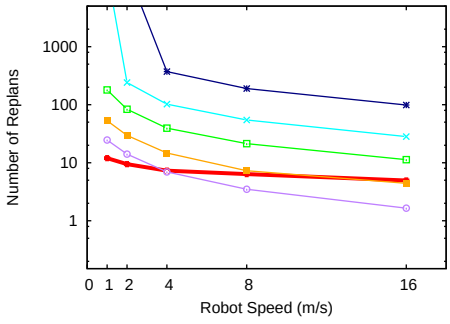


(a) Point-Polygon Cases

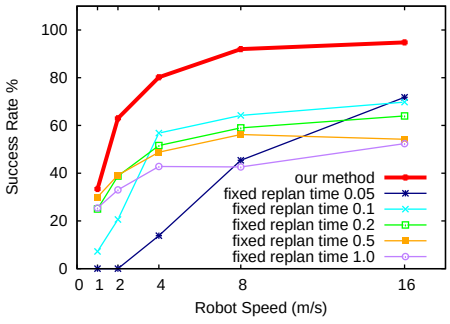
(b) Point-Polygon Cases



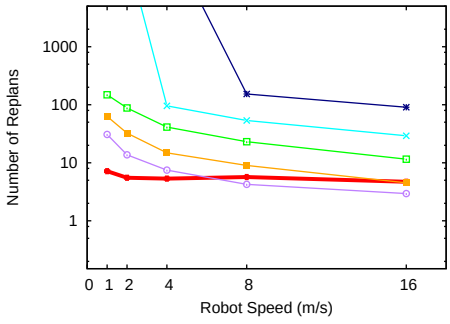
(c) Polygon-Polygon Cases



(d) Polygon-Polygon Cases



(e) Point-Articulated Cases



(f) Point-Articulated Cases

Fig. 5. Compare our method to the fixed-time strategy. The top row is obtained from environments in Fig. 1(a), and (a)-(c) in Fig. 4, the middle row is obtained from environments in Fig. 1(b) and (d)-(f) in Fig. 4, and the bottom row is obtained from environments in Fig. 1(c) and (g)-(i) in Fig. 4. Each data point in the plot is an average over 500 runs. In the fixed-time strategy, the robot replans every 0.05, 0.1, 0.2, 0.5 and 1.0 seconds. Notice that the y-axis of (b), (d) and (f) is in logarithmic scale.

tively modeled as a cone with the slope being its maximum velocity. Therefore, the path, if any, generated by their method is guaranteed to be safe.

To apply their strategy in our environments shown in Fig. 4, we replace the obstacles with their smallest bounding circles. Static obstacles are modeled as moving obstacles with zero velocity. Also note that bounding box is not allowed in their method. Our experiments found that, the robot needs to move at $22m/s$ or faster in order to find a safe path in Fig. 4(b), and at least $15m/s$ in Fig. 4(c). No path can be found at lower speed in these environments. For environments in Figs. 1(a), 4(a), and 4(c), the start or the goal is covered by one or more obstacles at the very beginning, thus no path can be found. On the contrary, the proposed method provides better flexibility while still allows the robot to achieve a nearly 90% success rate at $4m/s$ and almost 100% at $8m/s$.

9 Conclusion

In this paper, we proposed an adaptive method that predicts collisions for obstacles with unknown trajectories. We believe that this collision prediction has many potential usages and advantages. Similar to collisions 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 extracted path with unknown obstacle motion. When the robot travels on a predetermined path, collision prediction enables adaptive repairing period that allows more robust and efficient replanning. Comparing to a planning strategy that replans periodically at *fixed time interval*, our experimental results show strong evidences that the proposed method significantly reduces the number of replans while maintaining higher success rate of finding a valid path. Because of its ability to handle arbitrary shapes including articulated objects, this tool provides a complimentary approach to the methods that consider complex behavior prediction or dynamic constraints but with only simple shapes. Even though the obstacles are modeled as adversarial agents in this paper, we are currently investigate strategies to incorporate the constraints in obstacles' motion when better behavior patterns of the obstacle are known [7].

References

1. Jaillet, L., Simeon, T.: A prm-based motion planner for dynamically changing environments. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). (2004) 1606–1611
2. Kallman, M., Mataric, M.: Motion planning using dynamic roadmaps. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Volume 5., IEEE (2004) 4399–4404
3. Li, T.Y., Shie, Y.C.: An incremental learning approach to motion planning with roadmap management. In: Proc. of IEEE Int. Conf. on Robotics and Automation. (2002) 3411–3416

4. Ferguson, D., Kalra, N., Stentz, A.: Replanning with rrts. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, IEEE (2006) 1243–1248
5. Khatib, O.: Real–time obstacle avoidance for manipulators and mobile robots. Int. Journal of Robotics Research **5**(1) (1986) 90–98
6. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S.: Anytime dynamic a*: An anytime, replanning algorithm. (2005)
7. van den Berg, J., Ferguson, D., Kuffner, J.: Anytime path planning and replanning in dynamic environments. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (May 2006) 2366 – 2371
8. van den Berg, J., Overmars, M.: Planning the shortest safe path amidst unpredictably moving obstacles. In: Proc. Int. Workshop Alg. Found. Robot.(WAFR). (2006)
9. Wzorek, M., Kvarnstrom, J., Doherty, P.: Choosing path replanning strategies for unmanned aircraft systemsun. (2010)
10. Hayward, V., Aubry, S., Foisy, A., Ghallab, Y.: Efficient collision prediction among many moving objects. Internat. J. Robot. Res. **14**(2) (1995) 129–143
11. Hubbard, P.M.: Collision Detection for Interactive Graphics Applications. PhD thesis (1995)
12. Chakravarthy, A., Ghose, D.: Obstacle avoidance in a dynamic environment: A collision cone approach. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on **28**(5) (1998) 562–574
13. Kim, H.K., Guibas, L.J., Shin, S.Y.: Efficient collision detection among moving spheres with unknown trajectories. Algorithmica (2005) 195–210
14. Ziebart, B.D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A.K., Srinivasa, S.: Planning-based prediction for pedestrians. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, IEEE (2009) 3931–3936
15. Henry, P., Vollmer, C., Ferris, B., Fox, D.: Learning to navigate through crowded environments. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE (2010) 981–986
16. Wu, A., How, J.P.: Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. Autonomous Robots (2012) 227–242
17. Fraichard, T., Asama, H.: Inevitable collision states. a step towards safer robots? In: Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on. Volume 1., IEEE (2003) 388–393
18. Shiller, Z., Gal, O., Raz, A.: Adaptive time horizon for on-line avoidance in dynamic environments. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE (2011) 3539–3544
19. Martinez-Gomez, L., Fraichard, T.: 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 (2009) 100–105
20. Bautin, A., Martinez-Gomez, L., Fraichard, T.: Inevitable collision states: A probabilistic perspective. In: Proc. of IEEE Int. Conf. on Robotics and Automation, Anchorage, AK (2010) 4022 – 4027
21. Cohen, J., Lin, M.C., Manocha, D., Ponamgi, M.: I-collide: An interactive and exact collision detection system for large-scale environment. In: Symposium on Interactive 3D Graphics. (1995) 189–196
22. Gottschalk, S., Lin, M.C., Manocha, D.: OBBTree: A hierarchical structure for rapid interference detection. Computer Graphics **30** (1996) 171–180

23. Baraff, D.: Curved surfaces and coherence for non-penetrating rigid body simulation. Comput. Graph. **24**(4) (1990) 19–28
24. Hahn, J.K.: Realistic animation of rigid bodies. Comput. Graph. **22**(4) (1988) 299–308
25. Lu, Y., Xi, Z., Lien, J.M.: Conservative collision prediction among polygons with unknown motion. Technical Report GMU-CS-TR-2013-4, George Mason University (2013)
26. Lavalle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University (1998)

A Case Analysis of Point-Polygon Case

Given a moving segment $\overline{p_1p_2} \in \mathbf{O}_i$ colliding with $\mathbf{R}$. Let us analyze their relationships. As shown Fig. 6(a), the swept area of $\overline{p_1p_2}$ 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.

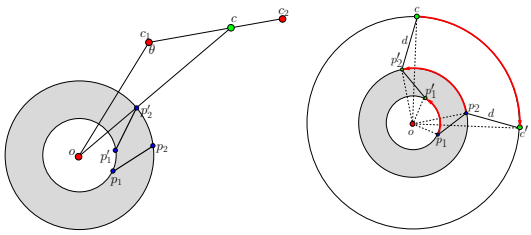


Fig. 6. (a) The swept area of $\overline{p_1p_2}$ is bounded by this grey shadowed donut-shape. The closest distance between $\overline{p_1p_2}$ and c is realized when p_2 becomes colinear with o and c. (b) If $\overline{p_1p_2}$ is fixed and c rotates around o with velocity $-\omega$, both the closest features and closest distance will be identical to the case where c is fixed and $\overline{p_1p_2}$ rotates around o with ω.

We separate our analysis into two cases: (1) $\overline{p_1p_2}$ and c are sufficiently far apart, and (2) $\overline{p_1p_2}$ and c are sufficiently close.

Let us first consider the situation that the segment $\overline{p_1p_2}$ and the point c are *sufficiently* far apart so that when $\overline{p_1p_2}$ moves at maximum (rotational and translational) speed, translation takes more time than rotation. In this case, the optimal motion is to translate $\overline{p_1p_2}$ along $\overline{oc}$ while rotating $\overline{p_1p_2}$ until p_2 is colinear with o and c. Thus, ECT of $\overline{p_1p_2}$ and c is simply

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

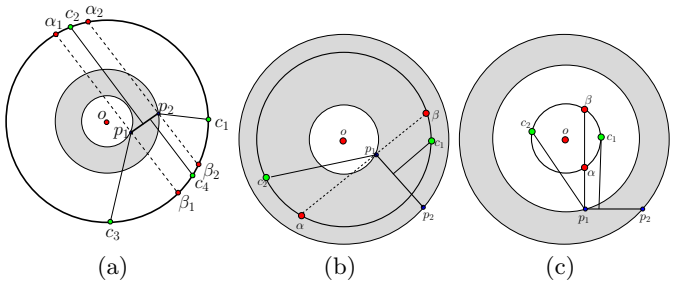


Fig. 7. Three cases when c is sufficiently close to the segment $\overline{p_1p_2}$

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

When c is sufficiently close to the segment $\overline{p_1p_2}$, $\overline{p_1p_2}$ can hit c before p_2, o and c become colinear. Depending on the relative position of c and the swept area of $\overline{p_1p_2}$, the motion strategy taken by $\overline{p_1p_2}$ will be different. As illustrated in Fig. 7, there are three cases we have to analyze. Due to space limitation, see our tech report [25] for more detailed definitions of $d(t)$ for all three cases of Fig. 7.

B Optimization Functions for the Point-Articulated Case

Considering the constraint on $\mathbf{O}_i$'s translation, the robot can hit a linkage L_k if and only if a point $p' \in \overline{c_1c_2}$ can reach L_k with the constraint of $\mathbf{O}_i$'s maximum translational velocity. Therefore, the second constraint is that the closest distance between p' and L_k which is $\mathrm{d}(p', L_k)$ needs to be no larger than $v_i \times t'$. Moreover, since the rotation of each joint is constrained to a maximum velocity, we have: $-\omega_1 \times t \leq \theta_1 \leq \omega_1 \times t$, $-\omega_2 \times t \leq \theta_2 \leq \omega_2 \times t$, to $-\omega_k \times t \leq \theta_k \leq \omega_k \times t$.

In short, we can formulate the problem of detecting the earliest collision moment between the point robot and every link $L_j \in \mathbf{O}_i$ as a non-linear optimization problem with the objective function t' subject to the following non-linear or linear constraints:

$$\begin{aligned} t &\leq t' \\ t_1 \leq t' &\leq t_2 \\ -\omega_k \times t \leq \theta_k &\leq \omega_k \times t \\ \mathrm{d}(p', L_j \in \mathbf{O}_i) &\leq v_i \times t' \end{aligned}$$

where t is the time $\mathbf{O}_i$ takes to make L_j to hit $\mathbf{R}$, t' is the time $\mathbf{R}$ takes to hit L_j, t_1 and t_2 are the moments when $\mathbf{R}$ reaches c_1 and c_2, respectively. And, ω_j is the maximum angular velocity of joint J_j, v_i is $\mathbf{O}_i$'s maximum translational velocity, and $\mathrm{d}(p', L_j)$ is the closest distance between p' and L_j. Every link which is an ancestor of L_j should satisfy its own rotational constraint $-\omega_k \times t \leq \theta_k \leq \omega_k \times t$ with $1 \leq k < j$.