Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

Conservative Collision Prediction Among Polygons with Unknown Motion

Yanyan Lu ylu4@gmu.edu
Zhonghua Xi zxi@gmu.edu
Jyh-Ming Lien jmlien@gmu.edu

Technical Report GMU-CS-TR-2013-4

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 the new method significantly reduces the number of replans 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. we would like to have a planner that can adaptively update the roadmap at "critical moments" at which the topology of the free configuration space changes.

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 when 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 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 7), 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, 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.

Apparently, computation of ICS or VO (even [26, 27]) 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, while our new method can guarantee safety by only knowing the maximum velocities of obstacles.

Yoshida et al. [28] proposed an online replanning method with parallel planning and execution and roadmap reuse. However, this strategy is only suitable for discrete environmental changes since replanning is time consuming and the robot needs to stop frequently if replanning is not finished in time. To address this issue, Yoshida and Kanehiro [29] proposed a reactive planning approach which considers both path replanning and deformation. When environmental changes are detected, it first checks if the path can be improved by local deformation. Only when the path becomes infeasible due to obstacles in its way and local improvements do not work, replanning is applied to generate a new feasible path by roadmap reuse.

Yang and Brock [30] proposed the Elastic Roadmaps for autonomous mobile manipulation. A free configuration is sampled around obstacles and moves with its associated obstacle. Therefore, the roadmap can always maintain task-consistent constraints.

Kim and Khosla [31] introduced harmonic potential functions to address the local minimum issue in Potential Fields. Feder and Slotine [32] extended this work to dynamic obstacles moving with constant translational or rotational velocities. However, the assumption that motions of both the robot and obstacles to follow harmonic functions is too strict. To relieve this limitation, Khansari-Zadeh and Billard [33] proposed an online local obstacle avoidance strategy using dynamic systems. The original motions of the robot defined by the user are specified by a continuous and differentiable dynamic system without considering any obstacles. Then given an analytical formulation describing the surface of obstacles, the original dynamic systems are locally deformed in order not to hit the 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, 34, 35, 36, 37]. 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 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. [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 intersect are further 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 volume*s 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 $\triangle t$ before this critical moment t to update its belief and replan if necessary. To make our discussion more concrete, let us 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, 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 the maximum translational velocity v_i and a maximum angular velocity ω_i around a given reference point o. o is the rotation center specified by users for an obstacle. Different obstacles can have different reference points for rotation.

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 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 all its subsequent segments $\overline{c_k c_{k+1}}$ with $j < k < n$. In Section 3.3, 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 Pre-processing

For a segment $\overline{c_1 c_2} \in \Pi$, let t_1 be the time when robot reaches c_1 and t_2 be the time when robot reaches c_2. We first perform a preprocessing step to filter out obstacles which are impossible to hit the robot on $\overline{c_1 c_2}$ between the time period $[t_1, t_2]$.

In order to achieve this, we first build an "envelope" of $\overline{c_1 c_2}$ with respect to every dynamic obstacle $\mathbf{O}_i \in \mathbf{O}$. As shown

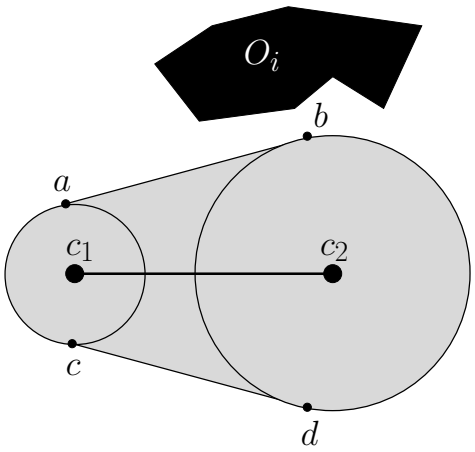

Figure 1: An "envelope" of path segment $\overline{c_1c_2}$ induced by $\mathbf{O}_i$. It is bounded by two circles which are centered at c_1 and c_2, respectively and two lines $\overline{ab}$ and $\overline{cd}$ which are tangent to these two circles.

in Fig. 1, such an envelope is bounded by two circles and two lines which are tangent to these circles. The radius of the circle centered at c_1 is $v \times t_1$ with v being $\mathbf{O}_i$'s maximum translational velocity. And the radius of the circle centered at c_2 is $v \times t_2$. A point p is on the boundary of this "envelope" if its closest distance to $\overline{c_1c_2}$ is $v \times t$ with t being the time when robot reaches p's closest point on $\overline{c_1c_2}$. Let SA be the area $\mathbf{O}_i$ sweeps over during $[t_1,t_2]$. In order to hit the robot on $\overline{c_1c_2}$, SA needs to intersect the envelope. In this way, we can filter out any obstacle whose swept region is separated from the envelope. For an obstacle, the further it is from $\overline{c_1c_2}$, the more possible it will be filtered out. Since it is nontrivial to compute swept area for an arbitrary shape with rotation, we can use a simple shape such as its oriented bounding box to approximate the original shape.

3.3 Earliest Collision Time (ECT)

Given a segment $\overline{c_jc_{j+1}} \subset \Pi$ 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 $\overline{c_jc_{j+1}}$. 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 any configuration $c \in \Pi$. Let t be the time that $\mathbf{R}$ takes to reach a configuration $c(t) \in \overline{c_jc_{j+1}}$ 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 $c \in \overline{c_1c_2}$ without violating these constraints. Since every configuration on $\overline{c_jc_{j+1}}$ 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 $g(t) : t = T$.

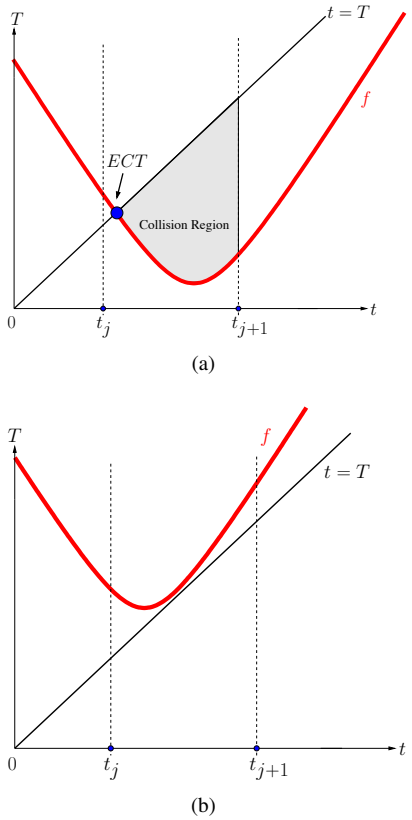

Figure 2: The red (thicker) curves in both figures are plots of the earliest arrival time $f(t)$ for an obstacle. Black straight lines are plots of $g(t) : t = T$. (a) When there is at least one intersection (blue dot) between $f(t)$ and $g(t)$, **collision region** is not empty. (b) Otherwise, the **collision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

In both figures Fig. 2(a) and Fig. 2(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 $\overline{c_jc_{j+1}}$ through one end point c_j at time t_j and leaves $\overline{c_jc_{j+1}}$ 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 $\overline{c_jc_{j+1}}$.

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

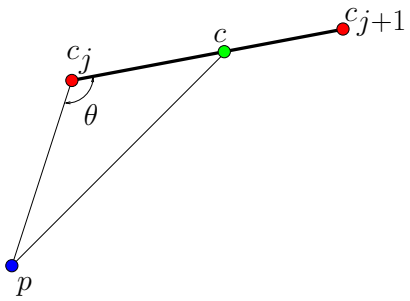

Figure 3: When both $\mathbf{O}_i$ and $\mathbf{R}$ are points, their closest distance can be computed using Law of cosines in $\triangle pc_jc$.

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

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

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

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

where $d = |\overline{pc_j}|$ and θ is the angle $\angle pc_jc_{j+1}$. This is illustrated in Fig. 3.

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 get a quadratic equation with only one variable $t = \sqrt{L^2 + d^2 - 2dL\cos\theta}\,/v_i$. By solving this equation, we can determine the **collision region** between $[t_j, t_{j+1}]$ based on the solutions. If the **collision region** is empty, there will be no collision between $\mathbf{O}_i$ and $\mathbf{R}$ on path segment $\overline{c_jc_{j+1}}$ (Fig. 2(b)).

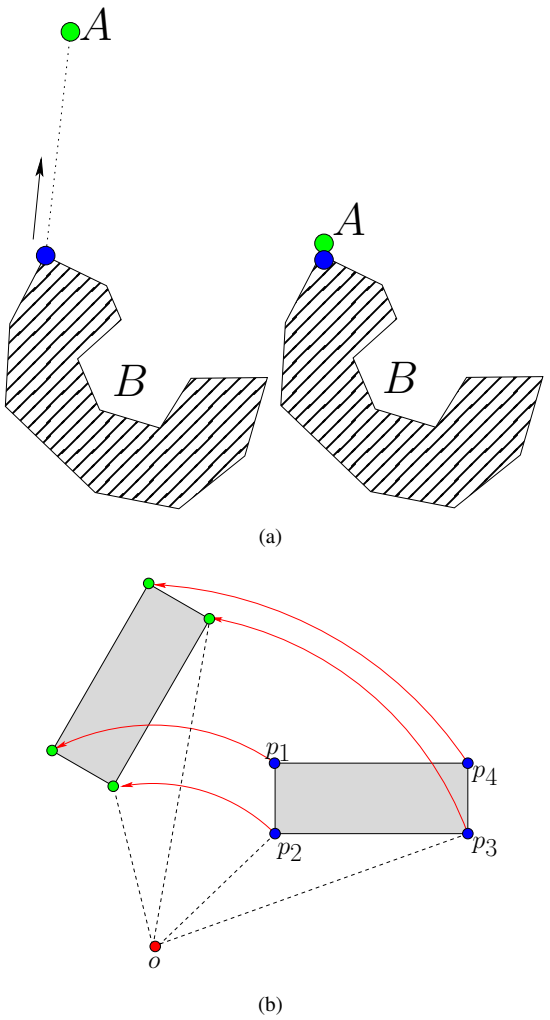

Figure 4: (a) If A is fixed and B translates towards A along the closest direction, the closest features between B and A remain the same until they collide. (b) A polygon rotates around a point o. The area it sweeps over is bounded by the trajectory of every vertex (p_1 through p_4) and edges from both original and destination shapes.

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 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 us 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. 4(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 $\mathbf{O}_i$ might 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. 4(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 $\overline{p_1p_2} \in \mathbf{O}_i$ colliding with $\mathbf{R}$. As shown Fig. 5(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.

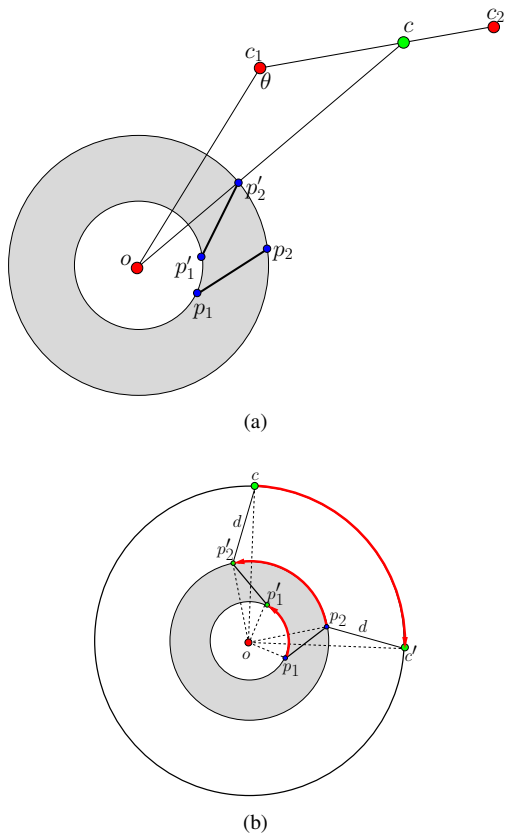

Figure 5: (a) The swept area of $\overline{p_1p_2}$ rotating is bounded by this grey shadowed donut-shape. The closest distance between $\overline{p_1p_2}$ and c is realized when p_2 becomes colinear with o and c. (b) If $\overline{p_1p_2}$ is fixed and c rotates around o with velocity $-\omega$, both the closest features and closest distance will be the same as in the case where c is fixed and $\overline{p_1p_2}$ rotates around o with ω.

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 $f(t)$ which is the earliest moment when $\overline{p_1p_2}$ hits this $c(t)$.

5.1 ECT of $\overline{p_1p_2}$ and $c \in \Pi$

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

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

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

When c is sufficiently close to the segment $\overline{p_1p_2}$, $\overline{p_1p_2}$ 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. 6, there are three cases we have to analyze.

Before we detailed our analysis, we found that fixing $\overline{p_1p_2}$ and rotating c around o significantly simplifies our discussion. That is, if $\overline{p_1p_2}$ 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. 5(b)).

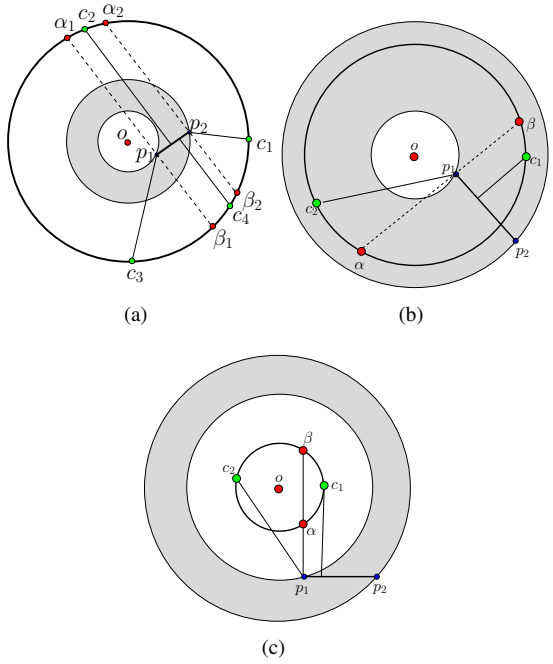

Figure 6: Three cases when c is sufficiently close to the segment $\overline{p_1p_2}$

When c orbits around o, the closest feature between c and $\overline{p_1p_2}$ changes among p_1, p_2 and the points in $\overline{p_1p_2}^\circ$, the open set of $\overline{p_1p_2}$. If c is outside the circle traced out by p_2 (Fig. 6(a)), the closest feature can change four times from p_2 to $\overline{p_1p_2}^\circ$ to p_1 to $\overline{p_1p_2}^\circ$ and back to $\overline{p_1p_2}^\circ$. If c overlaps with the swept area of $\overline{p_1p_2}$ (Fig. 6(b)), the closest feature changes twice between p_1

and $\overline{p_1p_2}^{\circ}$. If c is inside the circle traced out by p_1 (Fig. 6(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. 6) 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 $\overline{p_1p_2}$. We will talk about this in details later.

If we let the closest distance between c and $\overline{p_1p_2}$ be a function $\mathrm{d}(t)$ of time (we will talk about how to formulate $d(t)$ for all three cases of Fig. 6 later), 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 $t_T = h_T(t_R) = \mathrm{d}(t_R)/v_i$, where $\mathrm{d}(t_R)$ is the distance between c and segment $\overline{p_1p_2}$ when c rotates $\theta = -t_R\omega$ around o. The ECT between p and $\overline{c_1c_2}$ is:

$$\mathrm{ECT} = \arg\min_{t_R}\left(\max\left(t_R, h_T(t_R)\right)\right) \tag{4}$$

$$= \arg\min_{t_R}\left(|t_R - h_T(t_R)|\right)\ . \tag{5}$$

Therefore, ECT is t_R such that $t_R = \mathrm{d}(t_R)/v_i$. 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.

5.1.1 Computation Of α and β

To detect α and β, we transform the reference point o to the origin and $\overline{p_1p_2}$ to be 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 $\overline{p_1p_2}$ is $L_1 : x = a_1$. The straight line passing through p_2 and perpendicular to $\overline{p_1p_2}$ 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:

$$\cos(\omega t_R)c_x + \sin(\omega t_R)c_y = a_1\ . \tag{6}$$

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

$$\cos(\omega t_R)c_x + \sin(\omega t_R)c_y = b_1\ . \tag{7}$$

Fig. 7(a) shows the plots of equations 6 and 7. It illustrates how t_R changes as $\mathbf{R}$ moves from c_1 to c_2. Of course, both functions 6 and 7 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.

5.1.2 Distance Function $\mathrm{d}(t)$

Closest Point Is An Endpoint Of $\overline{p_1p_2}$. We first consider that case that the closest feature from $\overline{p_1p_2}$ is p_1. Since p_1 is fixed and $c(t)$ rotates around o with $-\omega$,

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

Therefore,

$$\mathrm{d}^2(t) = (x_t - a_1)^2 + (y_t - a_2)^2,$$

where $c(t) = (x_t, y_t)$ and $p_1 = (a_1, a_2)$.

After applying Laws of cosines to $\triangle op_1c(t)$,

$$\begin{aligned}\mathrm{d}(t) &= |\overline{\mathrm{c}(t)p_1}| \\ &= |\overline{op_1}|^2 + |\overline{oc(t)}|^2 - 2|\overline{op_1}||\overline{oc(t)}|\cos\angle p_1oc(t)\end{aligned} \tag{8}$$

We can get $c(t)$ by rotating c around the reference point o for θ with θ being $-\omega t$. Let $c = [x, y]$. Then

$$\begin{aligned}x_t &= \cos\theta x - \sin\theta y \\ y_t &= \sin\theta x + \cos\theta y\end{aligned}$$

By applying the above equations, $\mathrm{d}(t)$ can be further represented as follows.

$$\begin{aligned}\mathrm{d}^2(t) &= |\overline{op_1}|^2 + |\overline{oc}|^2 \\ &\quad - 2\cos\theta(a_1x + a_2y) + 2\sin\theta(a_1y - a_2x)\end{aligned} \tag{9}$$

Assume c is on $\overline{c_1c_2}$, then c can be interpolated with c_1 and c_2. In other words, $c = [x, y]$ is parameterized by $0 \le \lambda \le 1$ and $\overline{c_1c_2}$. 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, $\overline{c_1c_2}$ is rotated to be aligned with x-axis. Then

$$y = y_2 = y_1.$$

In order to detect ECT, we need to solve equation

$$\begin{aligned}\mathrm{d}^2(t_R) &= (v_it_R)^2 \\ &= |\overline{op_1}|^2 + |\overline{oc}|^2 - 2\cos\theta(a_1x + a_2y) + 2\sin\theta(a_1y - a_2x).\end{aligned} \tag{10}$$

where $\theta = -\omega t_R$.

Equation 10 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.

Closest Point Is $\overline{p_1p_2}^{\circ}$. If $\mathrm{d}(t)$ is defined between $\overline{p_1p_2}^{\circ}$ and c, then $\mathrm{d}(t)$ is the distance from c to the straight line containing $\overline{p_1p_2}$. To make it simpler, $\overline{p_1p_2}$ is rotated to be aligned with $y = x$ and $\overline{c_1c_2}$ is transformed accordingly. This line is easily computed and let it be

$$y = x + b.$$

Then $\mathrm{d}(t)$ can be written as

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

By replacing x_t and y_t with the equation above, $\mathrm{d}(t)$ can be represented as follows with θ being $-\omega t$.

$$\mathrm{d}^2(t) = \frac{|\overline{oc}|^2 + b^2 - \sin 2\theta(x^2 - y^2)}{2} - xy\cos 2\theta + b\cos\theta(x-y) - b\sin\theta(x+y) \quad (11)$$

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

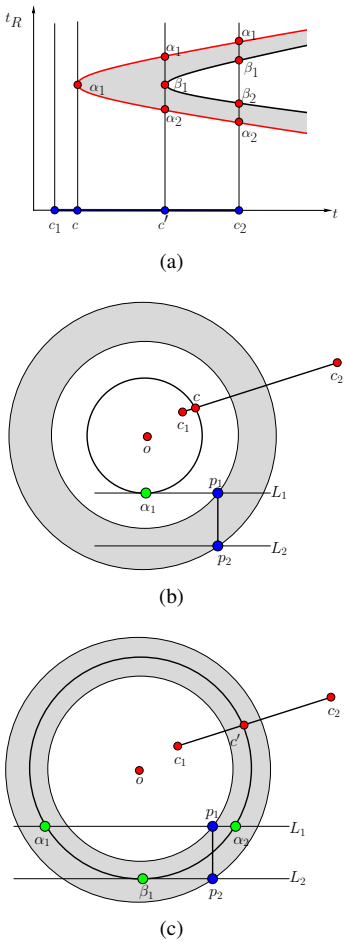

Figure 7: (a) The relationship between t_R and α and β when $\mathbf{R}$ moves from c_1 to c_2. (b) and (c) illustrate the configurations c and c' in (a). L_1 and L_2 are the lines perpendicular to $\overline{p_1p_2}$ and contain p_1 and p_2, respectively.

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. 6. More specifically, if we relate time t to α and β in Fig. 6, we can get a plot similar to Fig. 7. In Fig. 7(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. 6), then the closest feature between $\overline{p_1p_2}$ and $\overline{c''c_2}$ is always p_2.

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 $t_R = \mathrm{d}(t_R, p)/v_i$, where the point p is p_1 or p_2 and $\mathrm{d}(t_R, p)$ is the distance between p and the configuration of the robot at time t_R. The function $\mathrm{d}(t_R, p)$ 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 $t_R = \mathrm{d}(t_R, p)/v_i$ and $t_R = \mathrm{d}(t_R, \overline{p_1p_2}^\circ)/v_i$ that is valid only in the gray area shown in Fig. 7(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. 8 show an example on the function $f(t)$ and the closest distance between an edge of an obstacle and the robot over time.

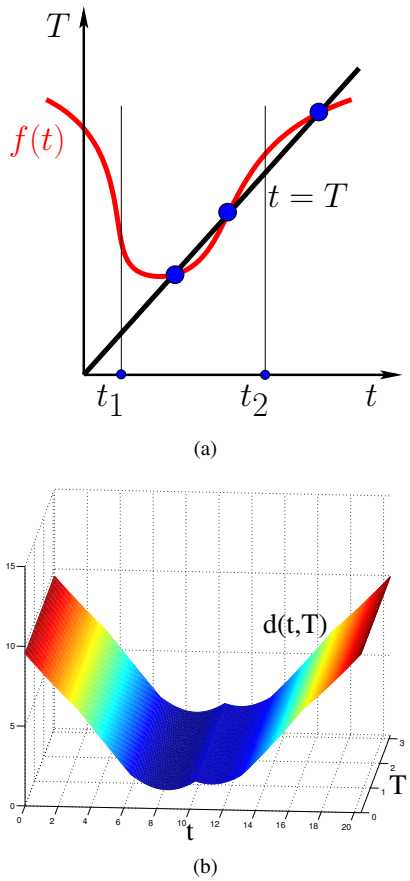

Figure 8: (a) The plot for $f(t)$ when the closest feature to c locates on $\overline{p_1p_2}^\circ$. The *collision region* is bounded by the two leftmost intersections. (b) The 3D plot shows the closest distance between an edge of some obstacle and the robot changes over time.

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.

6 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 when replanning becomes necessary. There are two desirable properties when a robot replans a path. First, we want a path to bring the robot near the goal. Second, we prefer the path to remain safe for as long as possible. With these two properties in mind, we propose to augment RRT [38] 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 earliest collision time (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. 9. 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.

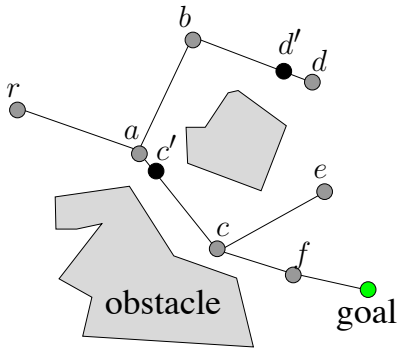

Figure 9: An RRT augmented with earliest collision time. The tree is rooted at current configuration r of the robot. Configurations c' and d' are the predicted earliest collision locations on the paths from r to c and d, respectively.

7 Experimental Results

We implemented the collision prediction method in C++ using Eigen linear algebra library. Experimental results reported in this paper are obtained from a workstation with two Intel Xeon E5-2630 2.30GHz CPUs and 32GB memory. We tested our implementation in ten environments shown in Fig. 10. Each environment is designed to demonstrate certain features. For example, Fig. 10(c) has a complicated bird shape, Fig. 10(d) has many (16) dynamic cross shapes, Fig. 10(g) has bars with large angular velocities, Fig. 10(f) or Fig. 10(i) contains narrow passages, Fig. 10(j) has long bars with large angular velocities and static bars surrounding start and goal.

These environments contain a point robot and 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 video submitted with this paper. More videos can also be found at *http://masc.cs.gmu.edu/wiki/ECT*.

7.1 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 6), and the other replans periodically at fixed time interval using regular RRT.

Fig. 11 and Fig. 12 show the *success rate* and *number of replans* obtained from environments in Fig. 10. 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 from Fig. 11(a) through Fig. 11(j) and Fig. 12(a) through Fig. 12(j) is collected over 100 runs. And each data point from Fig. 12(k) and Fig. 11(k) is collected over 1000 runs (i.e. 100 runs for each environment).

Success Rate and Number of Replans. From the plots in Fig. 11 and Fig. 12, 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. 11. We see that the success rate of the proposed method is almost identical to 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. 12, in order to provide a success rate similar to the proposed method, the fixed-time strategy needs to replan around 100 times more.

Running Time. In the table on the right, we provide average computation times spent on replanning over all five environments. We observe that, to achieve similar success rate, our method runs 3 and 12 times faster than fixed-time strategy with time step 0.1 and 0.05 sec, respectively.

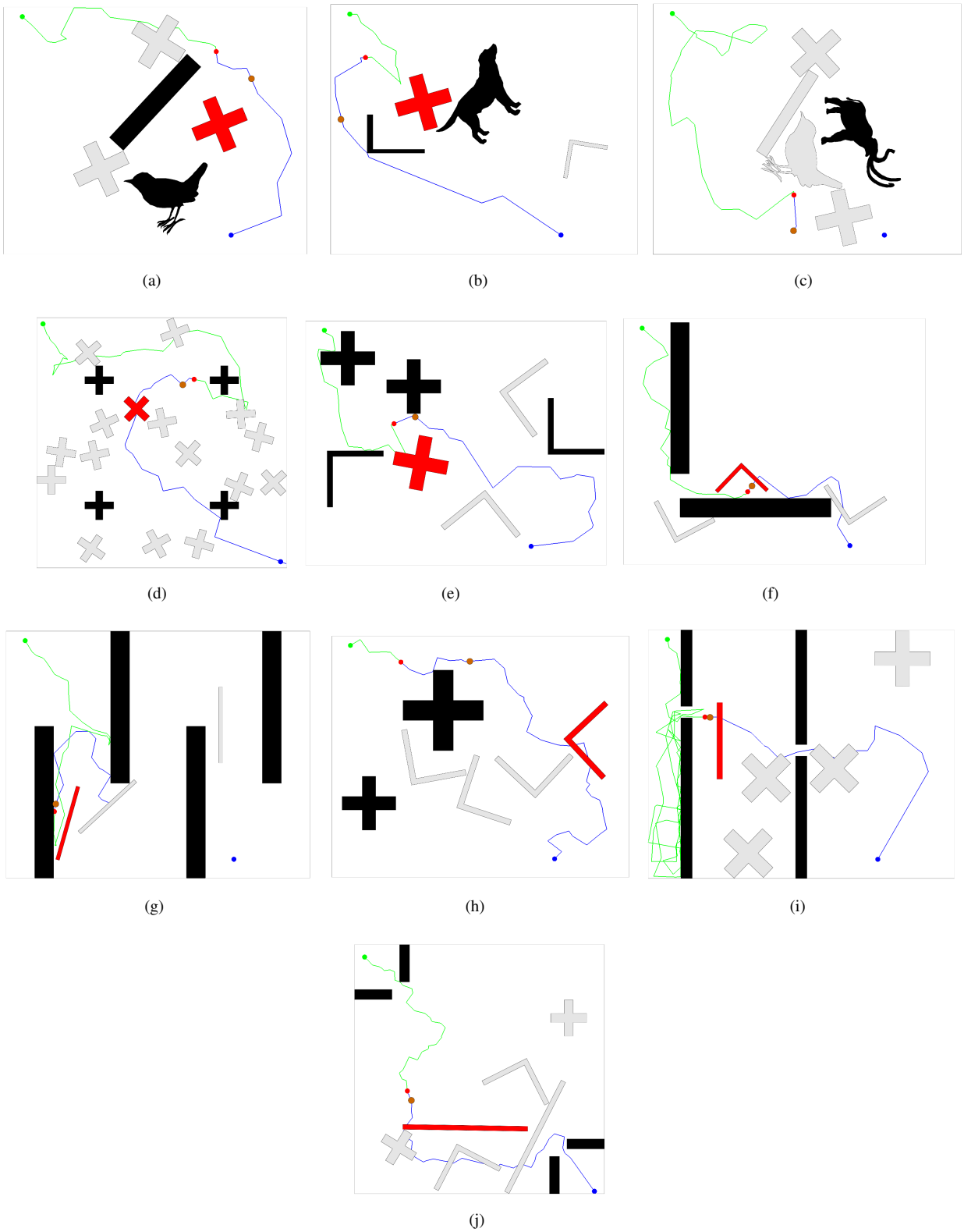

Figure 10: Ten environments used in experiments. In any of them, a green dot and a blue dot indicate start position and goal position, respectively. Black obstacles are static and light grey obstacles are dynamic. A red obstacle is the one which introduces earliest collision with the robot. A green curve shows the trajectory that the robot has traversed. So a red dot indicates the robot's current position when the image is captured. A brown dot shows the predicted location where the earliest possible collision might happen. A blue curve shows the path that robot plans to take. Notice that this path might be changed later due to possible collisions.

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

7.2 Compare to a Conservative Optimal Strategy

We further compare our method to an optimal strategy proposed by van den Berg and Overmars [11]. In their work, every obstacle must be a disc and its swept volume over time is conservatively 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. 10, 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. 10(d), and at least $15m/s$ in Fig. 10(j). No path can be found at lower speed in these environments. For environments in Fig.s 10(c), 10(g) and 10(j), 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 $8m/s$.

8 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. 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 [10].

References

[1] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, pp. 1419–1424, 1986.

[2] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, August 1996.

[3] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 473–479.

[4] L. Jaillet and T. Simeon, "A prm-based motion planner for dynamically changing environments," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2004, pp. 1606–1611.

[5] M. Kallman and M. Mataric, "Motion planning using dynamic roadmaps," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4399–4404.

[6] T.-Y. Li and Y.-C. Shie, "An incremental learning approach to motion planning with roadmap management," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 3411–3416.

[7] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 1243–1248.

[8] O. Khatib, "Real–time obstacle avoidance for manipulators and mobile robots," *Int. Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[9] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," 2005.

[10] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 2366 – 2371.

[11] J. van den Berg and M. Overmars, "Planning the shortest safe path amidst unpredictably moving obstacles," in *Proc. Int. Workshop Alg. Found. Robot.(WAFR)*, 2006.

[12] M. Wzorek, J. Kvarnstrom, and P. Doherty, "Choosing path replanning strategies for unmanned aircraft systemsun," 2010.

[13] V. Hayward, S. Aubry, A. Foisy, and Y. Ghallab, "Efficient collision prediction among many moving objects," *Internat. J. Robot. Res.*, vol. 14, no. 2, pp. 129–143, 1995.

[14] P. M. Hubbard, "Collision detection for interactive graphics applications," Ph.D. dissertation, 1995.

[15] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 5, pp. 562–574, 1998.

[16] H. K. Kim, L. J. Guibas, and S. Y. Shin, "Efficient collision detection among moving spheres with unknown trajectories," *Algorithmica*, pp. 195–210, 2005.

[17] N. Du Toit and J. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 966–973.

[18] K. Hauser, "Randomized belief-space replanning in partially-observable continuous spaces," *Algorithmic Foundations of Robotics IX*, pp. 193–209, 2011.

[19] Z. Shiller, O. Gal, and A. Raz, "Adaptive time horizon for on-line avoidance in dynamic environments," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3539–3544.

[20] B. D. Luders, G. S. Aoude, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns," MIT Aerospace Control Laboratory: Technical Reports, Tech. Rep., 2011.

[21] T. Fraichard and H. Asama, "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, 2003, pp. 388–393.

[22] L. Martinez-Gomez and T. Fraichard, "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, pp. 100–105.

[23] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[24] S. Bouraine, T. Fraichard, and H. Salhi, "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, 2011, pp. 2985–2991.

[25] E. Lalish and K. Morgansen, "Decentralized reactive collision avoidance for multivehicle systems," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 1218–1224.

[26] A. Bautin, L. Martinez-Gomez, and T. Fraichard, "Inevitable collision states: A probabilistic perspective," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, 2010, pp. 4022 – 4027.

[27] A. Wu and J. P. How, "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles," *Autonomous Robots*, pp. 227–242, 2012.

[28] E. Yoshida, K. Yokoi, and P. Gergondet, "Online replanning for reactive robot motion: Practical aspects," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, p. 5927.

[29] E. Yoshida and F. Kanehiro, "Reactive robot motion using path replanning and deformation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '2011)*, May 2011.

[30] Y. Yang and O. Brock, "Elastic roadmaps: Globally taskconsistent motion for autonomous mobile manipulation in dynamic environments," in *Proc. Robotics: Sci. Sys. (RSS)*, 2007.

[31] J. oh Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Trans. on Robotics and Automation*, vol. 8, no. 3, p. 338349, June 1992.

[32] H. J. S. Feder and J.-J. E. Slotine, "Real-time path planning using harmonic potentials in dynamic environments," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '97)*, May 1997.

[33] S. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, 2012.

[34] J. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scale environment," in *Symposium on Interactive 3D Graphics*, 1995, pp. 189–196.

[35] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," *Computer Graphics*, vol. 30, pp. 171–180, 1996.

[36] D. Baraff, "Curved surfaces and coherence for non-penetrating rigid body simulation," *Comput. Graph.*, vol. 24, no. 4, pp. 19–28, 1990.

[37] J. K. Hahn, "Realistic animation of rigid bodies," *Comput. Graph.*, vol. 22, no. 4, pp. 299–308, 1988.

[38] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

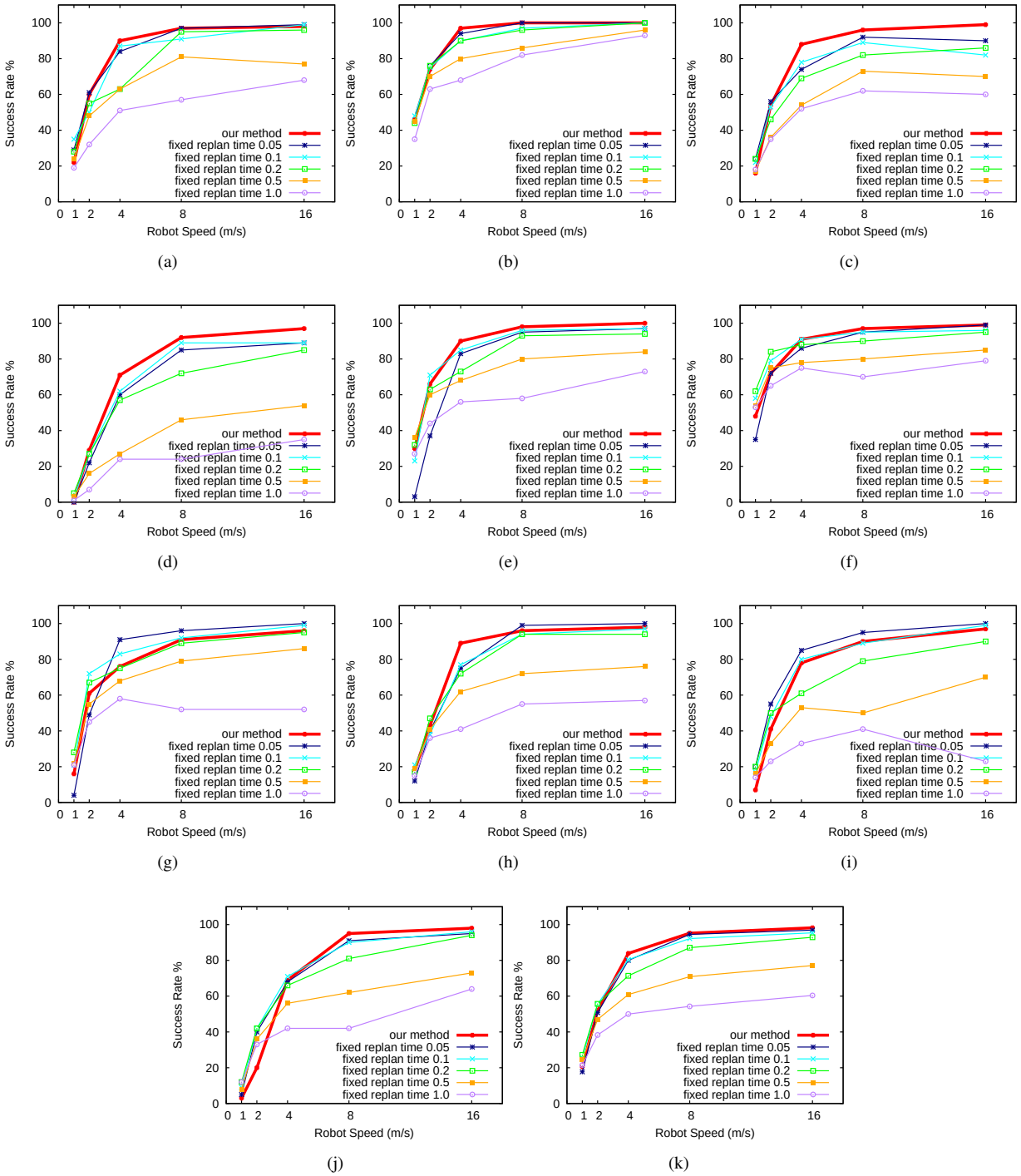

Figure 11: Compare our method to the fixed-time strategy on success rates for environments in Fig. 10. In the fixed-time strategy, the robot replans every 0.05, 0.1, 0.2, 0.5 and 1.0 seconds. For every specific robot velocity, (k) plots the average success rate over all these 10 environments.

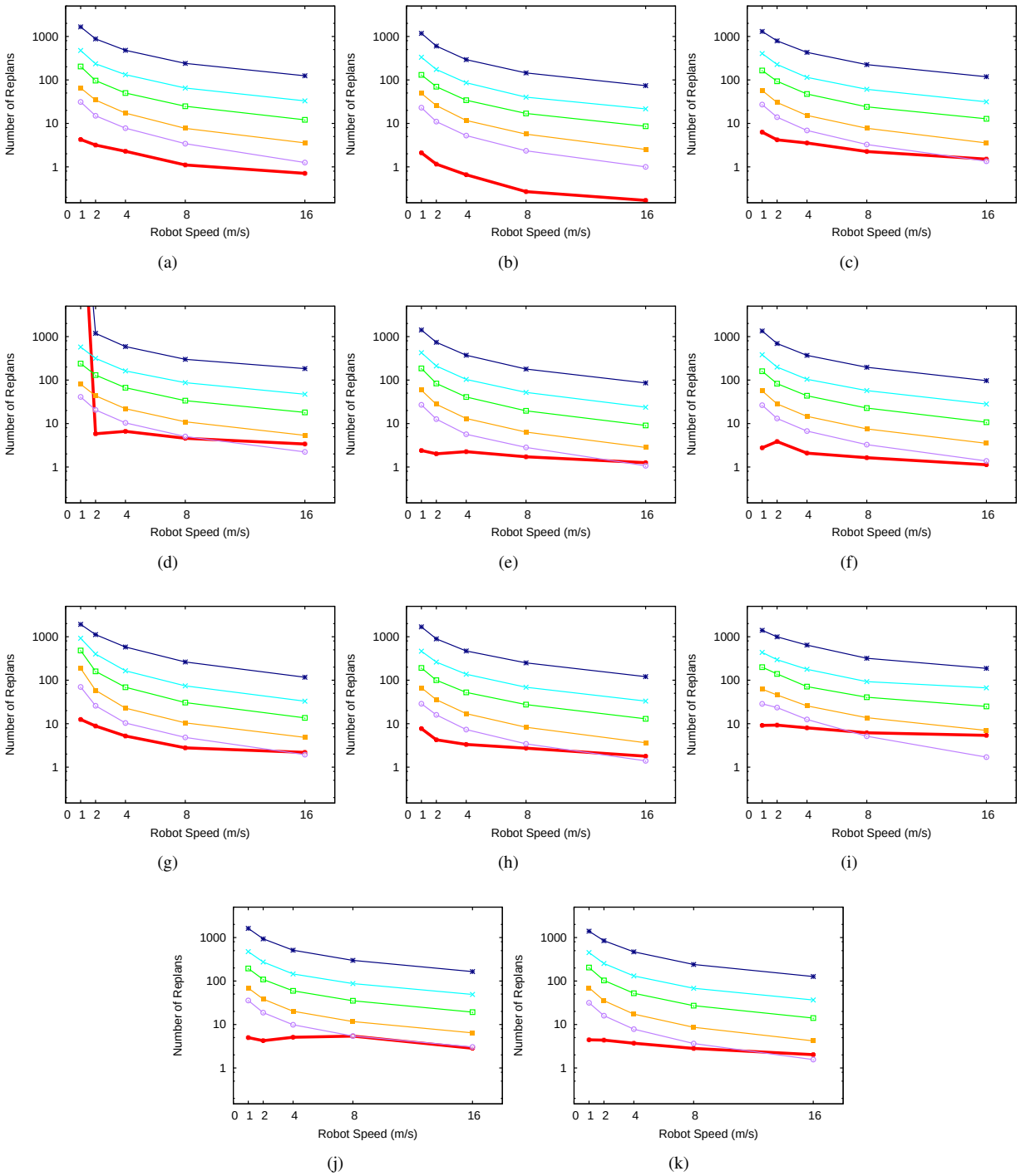

Figure 12: Compare our method to the fixed-time strategy on number of replans for environments in Fig. 10. In the fixed-time strategy, the robot replans every 0.05, 0.1, 0.2, 0.5 and 1.0 seconds. For every specific robot velocity, (k) plots the average number of replans over all these 10 environments. Notice that the *y*-axis of (b) is in logarithmic scale.