# Practical Motion Planning in Unknown and Unpredictable Environments

Rayomand Vatcha and Jing Xiao

**Abstract:**

Motion planners for robots in unknown and dynamic environments often assume known obstacle geometry and use that to predict unknown motions of obstacles through tracking, but such an assumption may not be realistic. In [1], we introduced a *collision-free perceiver* (CFP) that can detect guaranteed collision-free trajectory segments in the unknown configuration-time (CT) space of a robot without assuming known obstacle geometry or motion. However, such a guarantee by the CFP is at the expense of a finite period for perception and processing of each collision-free CT point. In this paper, we address how to incorporate the CFP, taking into account its finite processing time, into real-time motion planning to enable a robot of high degree of freedom to plan and move at the same time in an unknown and unpredictable environment while minimizing unsafe stops when the robot may collide with an obstacle. The approach was implemented and tested in experiments with a real 7-DOF robot arm and a stereo-vision sensor, indicating the potential of the approach.

## 1 Introduction

Motion planning for a robot moving in an uncertain, dynamic environment is gaining more attention in the robotics research community. One common assumption about the available information for such planning is known obstacle geometry. An additional assumption is certain knowledge of obstacle motion [2,3]. If the motion of an obstacle is unknown, a common approach is to predict the future motion by tracking the past obstacle motion (e.g. [4–9]). Thus, detecting if a robot at a configuration $\mathbf{q}$ and a future time $t$ will be in collision or not is converted to checking whether the

Department of Computer Science, University of North Carolina - Charlotte
{rvatcha, xiao}@uncc.edu

robot volume at the configuration-time (CT) point $(\mathbf{q},t)$ intersects with every obstacle volume at time $t$ . Many fast collision-checking algorithms [10–12] can be used to solve the problem efficiently for a limited number of obstacles. There are motion planners based on prediction for mobile robot motion planning (e.g., [13–15]) and for mobile manipulator motion planning [16].

However, tracking-based prediction requires either known obstacle geometry or accurate and fast segmentation of objects, which can be very challenging in cluttered and dynamic environments. In addition, prediction can only be sufficiently accurate for a short period, i.e., immediately after the time when the prediction is made. To compensate for that requires frequently repeated prediction and computation for collision-checking. Moreover, the planned robot motion is still not guaranteed collision-free due to the possibility of wrong predictions.

Assuming known obstacle geometry by a motion planner is in fact assuming very fast and accurate object recognition via sensing in an unknown and dynamic environment. There is progress in detecting and recognizing obstacles in some city road settings or off-road settings (e.g. [17, 18]), such as vegetation (e.g., [19]), people (e.g., [20]), etc. However, in very crowded environments with many unknown changes, recognizing all obstacles can be too slow, inaccurate, and also unnecessary. For example, imagine a crowded buffet restaurant, where a service robot carrying drinks has to maneuver through moving customers holding plates of food, people sitting at tables, moved chairs, etc. It can be very difficult to recognize every single obstacle on the robot's way; on the other hand, the robot does not need to recognize all the obstacles if it just wants to get the drinks to a particular table while avoiding collisions. Thus, it is desirable to study how to enable motion planning for robots without the need of recognizing unknown obstacles that may also move in unknown ways.

In [1] the authors introduced two novel concepts that do not assume obstacle geometries and do not predict obstacle motions: (a) *atomic obstacles* to represent an unknown environment directly from low-level sensor data at any sensing time $\tau$, and (b) *dynamic envelope* to detect if a robot configuration-time (CT) point ($\mathbf{q}$,t), will be guaranteed collision-free or not, by observing over sensing interval $[\tau,t)$. Only a simple assumption is made that obstacles move with any speed in [0, $v_{max}$]. The authors next introduced an efficient algorithm [21] to check for intersections between a dynamic envelope and atomic obstacles and also a strategy [22] to check if a trajectory is continuously collision-free by checking if a special set of discrete CT points are guaranteed collision-free. When a robot moves along such a collision-free trajectory, it surely will not be hit by any obstacle.

The above approach, which we call a *collision-free perceiver* (CFP), enables a planner to detect online if a trajectory is continuously collision-free or not in real world environments that are mostly unknown and dynamic. However, the detection of a collision-free CT point is often based on observation during a finite sensing interval rather than at a sensing instant. Thus, collision-checking cannot be instant even assuming unlimited computation power, let alone limited computation power. A trajectory may not be found in time by the CFP for the robot to move along, forcing the robot to stop and wait. During such a forced stop, the robot may be hit

by obstacles. If the robot has the risk of being hit during the interval of a forced stop, the stop is *unsafe*. Therefore, an important practical issue is how to minimize unsafe stops by taking into account the finite time that CFP has to detect a collision-free trajectory. We address this issue by applying CFP to a real-time adaptive motion planner (RAMP) [16] and extending RAMP to incorporate the time constraints of CFP, which we call *E-RAMP*, for an extended RAMP.

The rest of the paper is outlined as the following. In section 2, we will further review CFP and RAMP in more details. In section 3, we will describe P-RAMP. In sections 4 and 5, we present experimental validation of our approach with performance analysis. We conclude the paper in section 6.

## 2 Review of CFP and RAMP

We first review CFP based on the concepts of *atomic obstacles* and *dynamic envelope* and then describe RAMP and how RAMP can incorporate CFP.

### *2.1 Notations*

The following notation describes a robot model in the Cartesian space (i.e., the physical space):
$R(\mathbf{q})$: the region occupied by a robot $R$ in $\mathbb{R}^3$ at configuration $\mathbf{q}$.

We also use the following different temporal notations in the description of a robot's operation:

- $\tau$: time of sensing
- $t$: time of action

We denote an upper bound on all obstacle speeds in an environment as $v_{max}$.

### *2.2 Atomic Obstacles*

Atomic obstacles represent sensed obstacles in a sensing instant with low-level sensor data directly without elaborate sensor information processing. Given an image $I(\tau)$ from a line-of-sight sensor, such as a laser range finder, sonar, stereo vision, etc., each pixel $(i, j)$ maps to a point $(x, y, z)$ on a physical object, and $d_{ij}$ is the distance from $(x, y, z)$ to the origin $\{S\}$ of the sensor frame. Let $W_{ij}$ be the intersection of the viewing volume of the square pixel $(i, j)$ (as defined by the pixel's four corner rays originated from $\{S\}$) and the sphere centered at $\{S\}$ with radius $d_{ij}$. An *atomic obstacle* $O_{ij}$ is formed by $W_{ij}$ and the infinite volume of points it occludes in the physical space (see Figure 1).

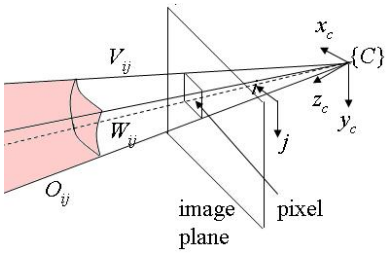



**Fig. 1** The geometry of an atomic obstacle $O_{ij}$ from a line-of-sight sensor

The union of all atomic obstacles at a sensing moment $\tau$ can be considered as forming the sensed obstacle space $O_s(\tau)$ at just $\tau$. We do not relate the atomic obstacles of one sensing moment to those of the next.

## *2.3 Dynamic Envelope*

At some sensing instant $\tau_0$, we aim to discover if a future CT-point $\chi = (\mathbf{q}, t)$, with $t > \tau_0$, is *guaranteed* collision-free. This entails observing a changing environment, during the time interval $[\tau_0, t]$, how obstacles move w.r.t. $R(\mathbf{q})$ to discover either (a) $\chi$ is collision-free at $\tau < t$, i.e., before time $t$, or (b) $\chi$ is not discovered collision-free at $\tau = t$. This is the insight of the novel concept dynamic envelope:

> A *dynamic envelope* $E(\chi, \tau)$, as a function of sensing time $\tau \leq t$, is a closed surface enclosing the region $R(\mathbf{q})$ in the physical space $\mathbb{R}^3$ (or $\mathbb{R}^2$ for 2-D planar space) such that the minimum distance between any point on $E(\chi, \tau)$ and $R(\mathbf{q})$ is
>
> $$d(t, \tau) = v_{max}(t - \tau) \tag{1}$$
>
> That is, $E(\chi, \tau) = R(\mathbf{q}) \oplus B(t, \tau)$, where $B(t, \tau)$ is a ball centered at the origin with diameter $d(t, \tau)$.

A dynamic envelope $E(\chi, \tau)$ has the following properties:

1. It shrinks monotonically over sensing time with speed $v_{max}$, i.e., $E(\chi, \tau_{i+1}) \subset E(\chi, \tau_i)$, where $i > 0$, $\tau_i < \tau_{i+1} \leq t$. $E(\chi, \tau)$ shrinks to $R(\mathbf{q})$ at $t$.
2. An actual obstacle not on or inside $E(\chi, \tau_i)$ will never be on or inside $E(\chi, \tau_{i+1})$.
3. An actual obstacle either on or inside $E(\chi, \tau_i)$ can be outside $E(\chi, \tau_j)$ for some $\tau_j \in (\tau_i, t]$, if not moving *towards* $R(\mathbf{q})$ in maximum speed $v_{max}$.

At any sensing instant $\tau_i$, since the union of all atomic obstacles contains the actual obstacles, thus, if no atomic obstacle is on or inside the dynamic envelope $E(\chi, \tau_i)$, there is no actual obstacle on or inside $E(\chi, \tau_i)$, and then, based on property 2) above, $\chi$ is guaranteed collision-free.

## 2.4 Collision-Free Perceiver (CFP)

The CFP discovers if a CT point $\chi = (\mathbf{q}, t)$ is collision-free or not by checking intersections between the dynamic envelope $E(\chi, \tau)$ and atomic obstacles for each sensing instant starting from $\tau_0$ until either the point is discovered collision-free (causing **return** from the algorithm), or maximum observing/computing time $t - \tau_0$ is met, as monitored by a system clock variable $t_{clock}$, as shown in Algorithm 1. $t_{clock}$ updates itself independently outside the CFP algorithm. The interval between two adjacent sensing instants is $\delta\tau$, i.e., the sensing frequency is $1/\delta\tau$. Note that each iteration in the **while** loop usually takes longer than $\delta\tau$. Thus, after each iteration, there is always the updated sensing data for the next iteration.

**Algorithm 1** Collision-Free Perceiver (CFP)

```
1: input CT point χ = (q,t), τ = τ0, δτ, t_clock = 0
2: get dynamic envelope E(χ,τ)
3: while τ < t and t_clock < t − τ0 do
4:    if E(χ,τ) does not intersect atomic obstacles at τ then
5:       E(χ,τ) expires
6:       return χ is guaranteed collision-free
7:    else
8:       τ = τ + δτ (for next sensor data)
9:    end if
10: end while
11: return χ may not be collision-free
```

In general, for a robot consisting of multiple links, each link can be approximated by a set of simpler well-defined geometrical objects, such as an oriented-bounding-box (OBB), a sphere, a capsule, etc. Now, a dynamic envelope can be created for each link. Since the dynamic envelope for an entire robot is the union of the dynamic envelopes of links, we only need to focus on how to check the intersection between the (simpler) dynamic envelope of a link and atomic obstacles. In order to be efficient, we use a strategy, called *extraction*, to identify atomic obstacles that are likely to intersect with a link dynamic envelope, i.e., the atomic obstacles whose indices $(i, j)$ are on the projection $P(E)$ of the dynamic envelope on the image plane.

In [21], we described an *Intersection-checking between Dynamic Envelope and Atomic ObstacleS* (IDEAOS) algorithm, which is a more efficient implementation of the **if** statement in the CFP.

## 2.5 Real-time Adaptive Motion Planner (RAMP)

The RAMP paradigm [16] is motivated by the need of real-time motion planning of high-DOF robots, such as (mobile) manipulators, in dynamic environments of unknown obstacle motions. A well known fact about motion planning for high-

DOF robots is that no complete algorithm is feasible even for known and static environment due to the formidable challenge of constructing high-dimensional C-obstacles. Thus, sampling-based planners, notably PRM [23] and RRT [24] planners and variants, are widely used.

RAMP is also sampling-based, but it is especially effective in planning high-DOF robot motion in dynamically unknown environments because of the following characteristics:

- real-time *simultaneous* planning and execution of high-DOF robot path/trajectory based on sensing;
- *anytime* and *parallel planning* with optimization, as inspired by evolutionary computation [25], through maintaining and repeatedly updating/improving a set of trajectory candidates for a robot from its current configuration to a goal configuration;
- great structural flexibility to allow for both on-line adaptation to different environmental scenarios and off-line extension to robots of very different nature.

All major components of the RAMP algorithm can be customized. The strength of RAMP lies in both its generality and its flexibility for adaptation and extension. Indeed, it has recently been extended to real-time continuum manipulator motion planning [26].

RAMP always maintains a set of diverse trajectories in the CT-space of the robot, called a *population*. The initial population of trajectories can be formed randomly. Each trajectory starts from the robot's current configuration and ends at the goal configuration and may be only partially *feasible* – defined as both collision-free and singularity free. A partially feasible trajectory is one that has a beginning feasible segment followed by an infeasible segment. The quality of a trajectory, in terms of feasibility and optimality, is evaluated through a *fitness evaluation function* that combines optimization criteria, such as shortest overall time, maximum time of the feasible segment, and so on.

As soon as a trajectory has a feasible segment starting from the robot's current configuration, RAMP allows the robot to move along it while planning for subsequent feasible trajectory segments simultaneously so that the robot can switch to the best subsequent feasible trajectory segment as it finishes the current feasible one. Three repeated cycles of processes are run simultaneously in the classical RAMP:

- Sensory data are updated in each *sensing cycle*.
- Trajectory modification and evaluation (or re-evaluation) based on sensing data is conducted in each *planning cycle*.
- The robot switches to a better trajectory from the currently executed one in each *control or adaptation cycle*.

Key to RAMP is efficient on-line detection of feasible trajectory segments of candidate trajectories. The original RAMP assumes known obstacle geometry and conducts collision checking based on predicting obstacle motions. It was also only implemented in simulation.

## 3 Technical Approach

By using CFP in RAMP for feasibility checking of a trajectory segment (through detecting collision-free CT points on it), we can eliminate the unrealistic assumption of known obstacle geometry and the drawback of feasibility checking based on prediction of obstacle motions. Since a feasible trajectory segment found by CFP is guaranteed collision-free, once the robot is moving along such a trajectory segment, say, $\Gamma_1$, it is safe, while the RAMP planner searches for subsequent feasible trajectories. Once the robot finishes $\Gamma_1$, if a subsequent feasible trajectory segment $\Gamma_2$ is found, then the robot can continue moving along $\Gamma_2$ seamlessly, and again, safely.

However, the CFP takes a finite time to detect a collision-free trajectory segment through detecting collision-free CT points. The actual time for CFP to detect a collision-free CT point $\chi = (\mathbf{q}, t)$ depends on two factors:

1. the size of the dynamic envelope $E(\chi, \tau)$, which is decided by $v_{max}(t - \tau)$ and shrinks as $\tau$ increases, and
2. the processing power of the computer and sensors.

Factor (1) is usually the dominating factor. Let $\tau_0$ be the time to start observing and checking if $(\mathbf{q}, t)$ is collision-free. If $E(\chi, \tau)$ is free of atomic obstacles by time $\tau_1 < t$, then CFP takes at least the time duration $\tau_1 - \tau_0$ to detect that $\chi$ is collision-free, even if the computation cost of detection, i.e., factor (2), is omitted.

Hence, the combined time of CFP and RAMP to decide a subsequent feasible trajectory segment can be longer than the time period of the feasible trajectory segment that the robot executes. Therefore, a subsequent feasible trajectory segment may not be found when the robot finishes executing the current segment, resulting in a *forced stop* of the robot. During such a forced stop, the robot may be hit by obstacles and thus unsafe.

Therefore, it is important that we extend the motion planner RAMP to minimize forced stops. The parallelism and flexibility of RAMP enables us to do so in the following ways:

- We add to the evaluation function of RAMP, as an additional optimization criterion, maximizing the safe time $\delta t_{safe}$ for the robot to pause at the end CT point $(\mathbf{q}_e, t_e)$ of a collision-free trajectory segment without being hit. With this added optimization criterion, RAMP is able to select a feasible trajectory segment (among all feasible trajectory segments found) that maximizes the total time $\Delta t_{safe} = \Delta t_{move} + \delta t_{safe}$, where $\Delta t_{move}$ is the time period of the segment. Note that $\delta t_{safe}$ at CT point $\chi_e = (\mathbf{q}_e, t_e)$ cannot be known precisely *before* the robot reaches $\chi_e$, but it can be (under)estimated as $d_{min}(\mathbf{q}_e, \tau)/v_{max} - (t_e - \tau)$ for $\tau < t_e$, where $d_{min}(\mathbf{q}_e, \tau)$ is the sensed minimum distance at $\tau$ between the robot and the atomic obstacles. Note also that since $\delta t_{safe}$ is meant for a collision-free CT point, $\tau$ is greater than the time $\tau_e$ when the CT point $\chi_e$ is found collision-free. We developed a method to obtain $d_{min}(\mathbf{q}_e, \tau)$ among atomic obstacles near the robot at $\chi_e$.
- We separate collision-checking using CFP from evaluation of the fitness of a trajectory, rather than embedding collision-checking into the fitness evaluation.

Collision-checking is ran constantly in the background and provides the RAMP information of the collision-free segment of a trajectory, while fitness evaluation simply uses the information to compute the value of the fitness function, which is much faster and almost instant.

We call the extended RAMP algorithm, the E-RAMP, as illustrated in Algorithm 2.

**Algorithm 2** E-RAMP

$m \leftarrow$ # of sensing cycles in a planning cycle
$n \leftarrow$ # of planning cycles in an adaptation cycle
$\Delta t_{min} \leftarrow$ small constant time
$\mathbf{q_e} \leftarrow$ starting configuration of the robot
$t_e \leftarrow$ current time $\tau$ $\{\tau$ is the clock time of the system that automatically updates$\}$
**initialize** a set $S$ of trajectories connecting the start configuration to the goal configuration of the robot
$\Delta t_{move} \leftarrow 0$
$\delta t_{safe} \leftarrow 0$

**while** the robot has not reached the goal **do**

**simultaneously** *sense*, *collision check*, *plan* and *adapt*, and *move*:

**sense**: repeat sensing cycles

**plan** at every $m$-th sensing cycle or when robot stops
**if** $t_e < \tau$ **then**
$\Delta t \leftarrow \max(\Delta t_{move} + \delta t_{safe}, \Delta t_{min})$
$t_e = \tau + \min(mn\delta\tau, \Delta t)$
Set starting time of all trajectories in $S$ to $t_e$
**end if**
**modify** $S$

**adapt**: when $\tau = t_e$ or at every $mn$-th sensing cycle
**evaluate** trajectories in $S$
$\Gamma_{best} \leftarrow$ best trajectory
$\mathbf{q_e} \leftarrow$ last configuration on the first collision-free trajectory segment of $\Gamma_{best}$
$\Delta t_{move} \leftarrow$ the time taken by the robot to move to configuration $\mathbf{q_e}$
$t_e \leftarrow \tau + \Delta t_{move} + \delta t_{safe}$
**if** $\Delta t_{move} + \delta t_{safe} = 0$ **then**
flag "unsafe" (robot is at risk of collision)
**end if**

**collision check**: call Algorithm 3

**move**: move the robot along $\Gamma_{best}$ the time period $\Delta t_{move}$
**end while**

In Algorithm 2, there are four simultaneous treads for *sense*, *collision check*, *plan* and *adapt*, and *move* respectively. The main **while** loop describes adaptation cycles. Each adaptation cycle consists of multiple planning cycles. Each sensing

**Algorithm 3** collision-checking

1: input trajectory segments of $N$ trajectories in $S$, where each trajectory segment $i$, $0 < i \leq N$, is a sequence of CT points $\chi_1^i, \chi_2^i, ...$ with time duration of $m \times n \times \delta\tau$
2: $C \leftarrow$ a sequence of CT-points in order $\chi_1^1, \chi_1^2, ..., \chi_1^N, \chi_2^1, \chi_2^2, ..., \chi_2^N, ...$
3: run CFP (Algorithm 1) for every CT point in $C$ until $S$ is updated
4: report collision-free CT points found in each trajectory; compute and report $\delta t_{safe}$ for the end collision-free CT point in each trajectory.

cycle lasts $\delta\tau$, determined by the planner, such that sufficient new information can be obtained in each sensing cycle for the CFP(Algorithm 1) to use, which is called by the collision-checking algorithm (Algorithm 3).

In each adaptation cycle, the robot simultaneously moves along the feasible segment of $\Gamma_{best}$ and plans for its next subsequent feasible segment, which starts from the end CT point $\chi_e = (\mathbf{q}_e, t_e)$ of the feasible segment of $\Gamma_{best}$. If, when the robot reaches $\chi_e$, no subsequent collision-free segment is found (i.e., not a single new CT free point is found), then the robot will stop its motion while continuing simultaneous collision-checking and planning until it finds a collision-free segment. If the time period that the robot stops is less than $\delta t_{safe}$, it means that the robot stopped safely at $\chi_e$ to continue planning for a while; otherwise the robot is forced to stop longer at $\chi_e$ at the risk of being hit by an obstacle.

The constants $m$ and $n$ are decided based on $v_{max}$ of obstacles and the size of the environment.

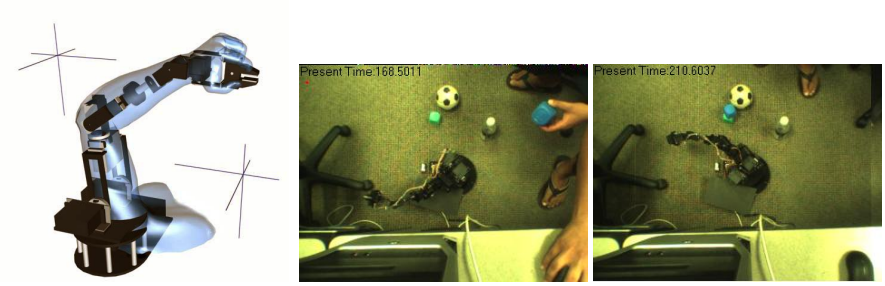
The following subroutines are used in Algorithm 2, in addition to Algorithm 3 for collision check:

- *initialize* a set of trajectories as in [16], through randomly creating intermediate knot configurations between the start and the goal configurations.
- *evaluate* the fitness function value for each trajectory, which is a cost function to both maximize the time of the feasible trajectory segment and minimize the total time of the trajectory.
- *modify* a randomly picked trajectory in $S$ to change its shape via adding/deleting or changing coordinates of knot configurations or CT points, evaluate the new trajectory, and use it to replace a non-best trajectory in $S$.

## 4 Implementation and Experiments

We applied the E-RAMP to plan motions of a real 7-DOF Robai's Cyton arm (see Figure 2(a)), using an indoor Point Grey's Digiclops stereo vision camera for overhead sensing and a DELL Precision T5400 computer with four cores and 4 GB RAM. Each robot link is approximated by an oriented bounding box. Each revolute joint has a maximum speed of 90 (deg/s) in both directions. To eliminate noise in sensing data, the image pixels were classified as (a) pixels of the robot and its accessories (i.e., robot circuit board, battery, etc.), identified by the colors, and (b)

obstacle pixels, by checking if they are in robot workspace. The software was built using the latest .NET 4.0 framework, and the program was done in C#. We con-




(a) A 7-DOF Cyton arm (b) The robot at $q_s$ in an environment (c) The robot at $q_g$

**Fig. 2** An experimental environment with the stereo-vision sensor providing a top view.

**Table 1** Planner and Task Parameters

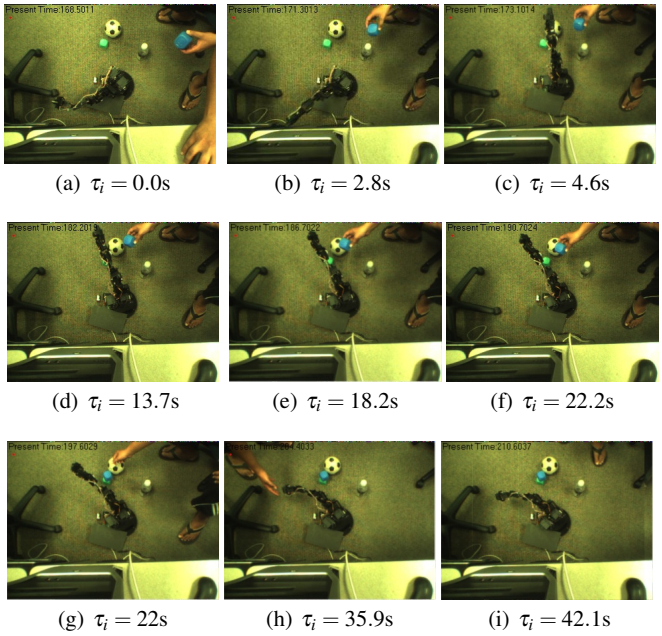
| $\|S\|$ | $v_{max}$ | $\Delta t_{min}$ | $\mathbf{q}_s$, $\mathbf{q}_g$ | Sensor Resolution | $\delta\tau$ |
|---|---|---|---|---|---|
| 5 | 1 cm/s | 0.5s | $\{-45°,-45°,-45°,0°,-45°,-45°,-45°\}$ $\{45°,45°,45°,45°,45°,45°,45°\}$ | $320 \times 240$ | 0.05s |

**Table 2** Experiments

| **Experiment #** | 1 | 2 | 3,4 | 5,6 |
|---|---|---|---|---|
| Obstacle moved | Blue block | Half-filled water bottle | Toy soccer ball | Plastic cover |

ducted six experiments with the robot in an environment consisting of a number of random obstacles such as football, blocks, table, plastic covers, half-filled water bottles, as well as a person moving those objects. The objects (except for the table) are moved by the person to create arbitrary motions unknown to the robot. Figures 2(b) and 2(c) show the starting configuration $\mathbf{q_s}$ and the goal configuration $\mathbf{q_g}$ of the robot, which values are shown in Table 4. Note that in order to reach $\mathbf{q_g}$ from $\mathbf{q_s}$, the robot end-effector cannot follow the straight-line path in the workspace because of the joint limitations. Table 4 shows the common parameter values used in those experiments. Except for $v_{max}$, which characterized the environment, the other parameter values were determined empirically based on the processing speed of the planner.

In these six experiments, different obstacles were moved by a person in more or less the same way during the robot's motion from the common $\mathbf{q_s}$ to $\mathbf{q_g}$, as shown in



(a) $\tau_i = 0.0$s (b) $\tau_i = 2.8$s (c) $\tau_i = 4.6$s

(d) $\tau_i = 13.7$s (e) $\tau_i = 18.2$s (f) $\tau_i = 22.2$s

(g) $\tau_i = 22$s (h) $\tau_i = 35.9$s (i) $\tau_i = 42.1$s

**Fig. 3** Snapshots of experiment #1

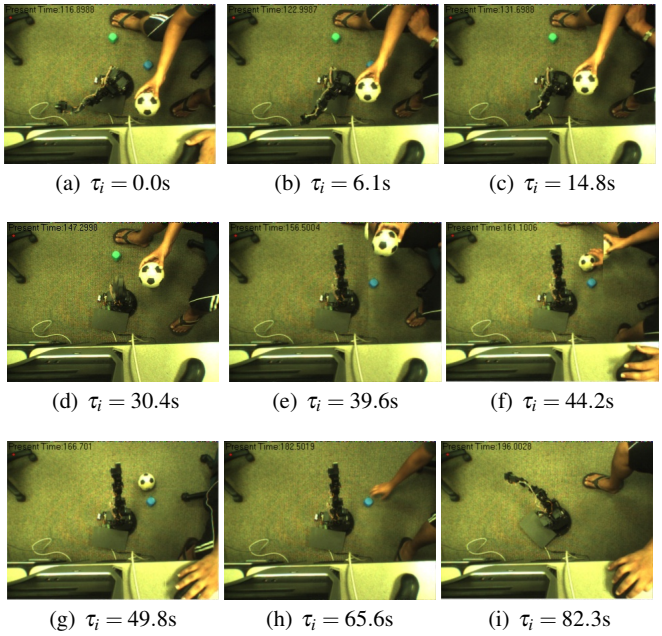
Table 2. These six experiments used three different sets of $m$ and $n$ values. Each set of $m$ and $n$ values were shared by a pair of experiments, as shown in Table 3, which we will discuss later.

We now describe one experiment for each set of $m$ and $n$ values.

In experiment #1 (see Figure 3), the person moved the blue block to approach the robot (see snapshots 3(a)–3(c)). Then the robot tried to avoid it and other obstacles while moving towards the goal as the person moved the block closer to the robot (shown in snapshots 3(d)–3(i)) and finally reached the goal (see 3(i)).

In experiment #3 (See Figure 4),initially the robot encountered the soccer moved by the person and performed a motion by bending half of the arm and then moved away from both obstacles (see snapshots 4(a)–4(c)). As the person moved the soccer towards the robot (similar to experiment #1), the robot started moving away from it (see snapshots 4(d)–4(i)) and successfully reached the goal as shown in Figure 4(i).

In experiment #5 (See Figure 5),initially the robot encountered the plastic cover moved by the person and performed a motion by bending half of the arm to stay away from the plastic cover (see snapshots 5(a)–5(c)). Later, as the plastic cover moved closer to the robot, the robot moved away from the table and near the plastic bag while staying away from the person's leg (see snapshots 5(d)–5(f)). While the plastic cover kept approaching the robot; the robot was able to find a motion to reach the goal successfully (see snapshots 5(f)–5(i)).



(a) $\tau_i = 0.0$s (b) $\tau_i = 6.1$s (c) $\tau_i = 14.8$s

(d) $\tau_i = 30.4$s (e) $\tau_i = 39.6$s (f) $\tau_i = 44.2$s

(g) $\tau_i = 49.8$s (h) $\tau_i = 65.6$s (i) $\tau_i = 82.3$s

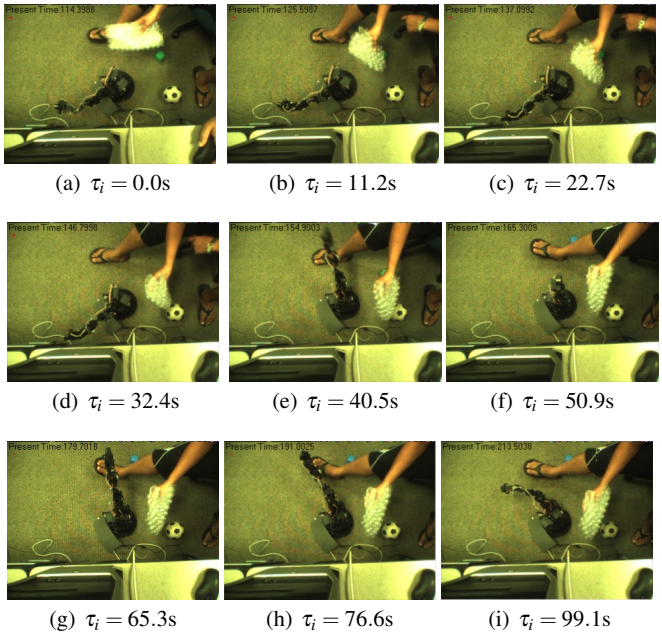
**Fig. 4** Snapshots of experiment #3

## 5 Results and Main Experimental Insights

The results for the six experiments performed are shown in Table 3. As it shows, there are unsafe stops in all cases. However, the data show that by increasing $n$, i.e., increasing the number of planning cycles per adaptation cycle, both the number of unsafe stops and the average duration of an unsafe stop decreased. Having multiple planning cycles within an adaptation cycle (i.e. $n > 1$) leads to finding on average a longer feasible path segment of $\Gamma_{best}$. Note that since we assign a constant speed for the robot, the time duration $\Delta t_{move}$ of a feasible segment is proportional to its path length.

It can be seen that the # adaptation cycles during which the robot moved was smaller than the minimum total # adaptation cycles (which is the total # planning cycles divided by $n$), and in some cases far smaller. This shows that there were more stops than the total # unsafe stops, meaning that there were safe stops and in some cases a lot of them, as in experiments 2, 3, and 5. Note that the # of $\delta t_{safe}$ computations show the number of end points of collision-free trajectory segments, that is, the number of collision-free segments. This number is smaller for cases with longer average path length of feasible segments.

As shown in Table 3, the time needed to do one iteration in CFP for finding a collision-free CT point (i.e., CT-free point) on average ranged from 3 to 15 sensing cycles. Since many CT-free points in the experiments require two iterations to be



(a) $\tau_i = 0.0$s (b) $\tau_i = 11.2$s (c) $\tau_i = 22.7$s

(d) $\tau_i = 32.4$s (e) $\tau_i = 40.5$s (f) $\tau_i = 50.9$s

(g) $\tau_i = 65.3$s (h) $\tau_i = 76.6$s (i) $\tau_i = 99.1$s

**Fig. 5** Snapshots of experiment #5

found (i.e., based on sensing data of two different sensing instants), the total time for finding a CT-free point is even longer. In spite of this high cost of CFP, the Algorithm 2 was able to lead the robot to its goal in those experiments successfully.

However, for a greater $v_{max}$, one iteration in CFP will take even more time because the dynamic envelope is larger. Thus, faster computation for the intersection check in CFP is necessary, which remains one of our on-going research topics.

## 6 Conclusions

This paper introduced an approach enabling a high-DOF robot to plan in real-time and move along a collision-free trajectory in an environment of unknown and unpredictable obstacles, while minimizing unsafe stops, in spite of a considerable cost for detecting collision-free configuration-time points. The approach was implemented and tested in a few experiments involving a real 7 degree-of-freedom manipulator and a fixed stereo-vision sensor. These experiments show the potential of this approach but also reveal issues that require further research, including (a) a faster algorithm for CFP, preferably with parallel implementation and GPU computation, (b) better handling of sensing and modeling uncertainty to increase robustness, and (c)

**Table 3** Experimental Results

| **Parameters** | | $m$ **= 20,** $n$ **= 1** | | $m$ **= 10,** $n$ **= 3** | | $m$ **= 20,** $n$ **= 2** | |
|---|---|---|---|---|---|---|---|
| **Experiment** | | 1 | 2 | 3 | 4 | 5 | 6 |
| **Total time (s)** | | 42.09 | 83.99 | 82.29 | 137.99 | 99.09 | 86.49 |
| **Total # planning cycles** | | 21 | 42 | 12 | 128 | 20 | 8 |
| **# adaptation cycles that the robot moved** | | 20 | 22 | 3 | 9 | 5 | 3 |
| **Total # unsafe stops** | | 7 | 9 | 2 | 5 | 3 | 2 |
| **Duration of** | Avg. | 0.69 | 0.48 | 0.1 | 0.33 | 0.16 | 0.1 |
| **an usafe stop** (s) | Min. | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| | Max. | 1.3 | 1.3 | 0.1 | 0.4 | 0.2 | 0.1 |
| **Path length of** | Avg. | 34.01 | 26.45 | 119.05 | 47.62 | 79.37 | 119.05 |
| **feasible seg. of** | Min. | 13.22 | 13.22 | 66.14 | 13.22 | 26.45 | 66.14 |
| $\Gamma_{best}$ (deg) | Max. | 145.51 | 79.37 | 171.97 | 92.60 | 171.97 | 171.97 |
| **Total # CT-free points found** | | 97 | 618 | 64 | 521 | 200 | 98 |
| **Time for one** | Avg. | 712.22 | 637.08 | 156.51 | 184.66 | 137.89 | 142.49 |
| **iteration in CFP** | Min. | 109.375 | 15.62 | 109.37 | 46.87 | 15.62 | 93.75 |
| **that found a CT-free pt.** (ms) | Max. | 11625 | 5109.37 | 343.7 | 859.37 | 406.25 | 343.75 |
| **Total # of** $\delta t_{safe}$ **computations** | | 83 | 196 | 51 | 452 | 100 | 31 |
| **Time for** | Avg. | 681.48 | 1850.153 | 173.99 | 178.47 | 281.40 | 192.02 |
| **computing one** | Min. | 31.25 | 140.62 | 109.37 | 15.62 | 109.37 | 125 |
| $\delta t_{safe}$ (ms) | Max. | 2656.25 | 13828.13 | 281.25 | 718.75 | 1671.87 | 312.5 |

a mechanism to enable and utilize changing sensor views to provide greater range of perception.

## References


1. R. Vatcha and J. Xiao, "Perceived CT-space for motion planning in unknown and unpredictable environments," in *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Dec 2008.
2. P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," in *Intl. J. of Robotics Research*, 1998, pp. 760–772.
3. F. Large, S. Sckhavat, Z. Shiller, and C. Laugier, "Using non-linear velocity obstacles to plan motions in a dynamic environment." in *IEEE Intl. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, 2002, pp. 734–739.
4. A. Elnagar and K. Gupta, "Motion prediction of moving objects based on autoregressive model," *IEEE Trans. on Systems, Man, and Cybernetics (Systems and Humans)*, vol. 28, no. 6, pp. 803–810, November 1998.
5. C. C. Chang and K.-T. Song, "Environment prediction for a mobile robot in a dynamic environment," *IEEE Trans. on Robotics and Automation*, vol. 13, no. 6, pp. 862–872, Dec 1997.
6. G. Gallagher, S. S. Srinivasa, J. A. Bagnell, and D. Ferguson, "Gatmo: a generalized approach to tracking movable objects," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 2043–2048.
7. A. Ess, B. Leibe, K. Schindler, and L. V. Gool, "Moving obstacle detection in highly dynamic scenes," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 56–63.
8. A. Elnagar and A. Hussein, "An adaptive motion prediction model for trajectory planner systems," in *Intl. Conf. on Robotics and Automation*, September 2003, pp. 2442–2447.

9. V. Govea, D. Alejandro, F. Large, T. Fraichard, and C. Laugier, "Moving obstacles' motion prediction for autonomous navigation," in *Int. Conf. on Control, Automation, Robotics and Vision*, December 2004.
10. J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scale environments," in *In Proc. of ACM Interactive 3D Graphics Conf.*, 1995, pp. 189–196.
11. P. Jimnez, F. Thomas, and C. Torras, "3D collision detection: A survey," *Computers and Graphics*, vol. 25, pp. 269–285, 2000.
12. M. C. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *In Proc. of IMA Conf. on Mathematics of Surfaces*, 1998, pp. 37–56.
13. A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 1662–1668.
14. V. Govea, D. Alejandro, F. Large, T. Fraichard, and C. Laugier, "High-speed autonomous navigation with motion prediction for unknown moving obstacles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, October 2004, pp. 82–87.
15. J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 2006, pp. 2366–2371.
16. J. Vannoy and J. Xiao, "Real-time Adaptive Motion Planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes," in *IEEE Trans. on Robotics*, vol. 24(5), Oct. 2008, pp. 1199–1212.
17. A. Murarka, M. Sridharan, and B. Kuipers, "Detecting obstacles and drop-offs using stereo and motion cues for safe local motion," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 702–708.
18. C. Caraffi, S. Cattani, and P. Grisleri, "Off-road path and obstacle detection using decision networks and stereo vision," *IEEE Trans. on Intelligent Transportation Systems*, vol. 8, no. 4, pp. 607–618, 2007.
19. D. Bradley, R. Unnikrishnan, and J. A. Bagnell, "Vegetation detection for driving in complex environments," in *IEEE Intl. Conf. on Robotics and Automation*, April 2007.
20. N. Bellotto and H. Hu, "Multisensor-based human detection and tracking for mobile service robots," *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, vol. 39, no. 1, pp. 167–181, 2009.
21. R. Vatcha and J. Xiao, "An efficient algorithm for on-line determination of collision-free configuration-time points directly from sensor data," in *IEEE Intl. Conf. on Robotics and Automation*, May 2010.
22. ——, "Discovering guaranteed continuously collision-free robot trajectories in an unknown and unpredictable environment," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, October 2009.
23. L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE Trans. on Robotics and Automation*, 1996, pp. 566–580.
24. S. M. LaValle, *Planning Algorithms*. Cambridge University Press, May 2006.
25. P. P. Bonissone, R. Subbu, N. Eklund, and T. R. Kiehl, "Evolutionary algorithms + domain knowledge = real-world evolutionary computation," *IEEE Trans. Evolutionary Computation*, vol. 10, no. 3, pp. 256–280, 2006.
26. J. Xiao and R. Vatcha, "Real-time adaptive motion planning for a continuum manipulator," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, October 2010.