2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

An Efficient Algorithm for On-line Determination of Collision-free Configuration-time Points Directly from Sensor Data

Rayomand Vatcha and Jing Xiao

Abstract— **On-line, efficient perception based on sensing is essential for an autonomous robot to operate in an unknown and unpredictable environment. An efficient on-line algorithm is introduced to determine whether a robot at a future time t and a configuration $\mathbf{q}$ will be guaranteed collision-free, directly from real-world sensor data of the robot's environment at the current time τ, using stereo vision sensor. Such a problem can be formulated [1] as checking the intersection between the so-called *dynamic envelope*, which relates to the robot at a configuration-time (CT) point $(\mathbf{q}, t)$ and the current sensing time τ, and the *atomic obstacles*, which are obtained directly from low-level sensory data at τ. The algorithm achieves real-time efficiency, as confirmed by the experimental results, by classifying the atomic obstacles possibly intersecting the dynamic envelope and by grouping relevant atomic obstacles on the fly. It is suitable to be used on-line by sensing-based motion planners.**

I. Introduction

For an autonomous robot to move in an environment with unknown obstacles and changes, it must be either able to detect possible potential collisions with obstacles or the lack of them based on sensing in real time. Reactive obstacle avoidance schemes (e.g., [2]–[4]) simply steer the robot away from possible obstacles based on proximity sensing. On-line planners often assume recognizable obstacles and that obstacle motions are either known (e.g., [5]–[9]) or can be predicted (e.g., [10]–[13]). Thus, detecting if the robot at a configuration $\mathbf{q}$ and a future time t will be in collision or not is converted to the problem of checking whether the model of each obstacle, with certain geometry, at a pose at time t, intersects with the model of the robot at the configuration-time (CT) point $(\mathbf{q}, t)$. Many fast collision-checking algorithms [14]–[16] can be used to solve the problem efficiently for a limited number of obstacles.

There is good 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 tedious and also unnecessary. For example, imagine a crowded buffet restaurant, where a service robot carrying drinks has to manuever 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; moreover, 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 detect potential collision-free CT-points without recognizing unknown obstacles that may also move in unknown ways.

In [1], [21] the authors have introduced a general approach to perceive at sensing time τ whether a robot (which can be of high-DOF, such as a manipulator) will be guaranteed collision-free at CT-point $(\mathbf{q}, t)$ $(t > \tau)$ without requiring recognition of obstacles and predictions of their motions. The problem is essentially formulated as checking the intersection between the so-called *dynamic envelope*, relating the robot at $(\mathbf{q}, t)$ and the current sensing time τ, and the *atomic obstacles*, which are obtained directly from low-level sensory data at τ. One important question is how efficient such intersection checking can be, given the large number of atomic obstacles at any sensing instant. In this paper, we present an efficient algorithm for on-line checking whether a dynamic envelope intersects atomic obstacles obtained from stereo vision sensing. By identifying and grouping relevant atomic obstacles, the algorithm achieves real-time efficiency. By taking advantage of time and space coherence, our algorithm further reduces the time cost per CT-point per sensing interval for multiple CT-points.

The paper is organized as follows. Section II introduces atomic obstacles based on stereo vision sensing and also reviews the concept of dynamic envelope for perceiving collision-free CT-points. Section III presents our core algorithm. Section IV describes on how time and space coherence is exploited. Section V describes implementation and discusses experimental results. Section VI concludes the paper.

II. Atomic Obstacles and Dynamic Envelope

In [1], [21], two novel concepts were introduced: (a) atomic obstacles to model sensed information of an unknown environment at a sensing instant without assuming any actual obstacle geometry, and (b) dynamic envelope to enable discovery of guaranteed collision-free points and regions in the, otherwise, unknown CT-space. In the following, we first define atomic obstacles from stereo vision sensing and then review the concept of dynamic envelope and its use for detecting guaranteed collision-free CT-points.

A. Atomic Obstacles

Atomic obstacles are low-level sensor data to represent real obstacles at any sensing instant in an unknown and changing environment without requiring elaborate sensor information processing. Thus, atomic obstacles depend on

This research was supported by the National Science Foundation under Grant IIS-0742610.

R. Vatcha is a PhD student in Computer Science,University of North Carolina at Charlotte, USA. rvatcha@uncc.edu

J. Xiao is with the faculty of Computer Science, University of North Carolina at Charlotte, USA. xiao@uncc.edu

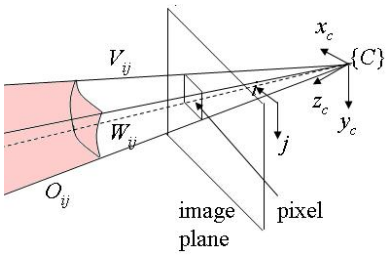


Fig. 1. The geometry of an atomic obstacle O_{ij} from a stereo vision sensor

each sensing instant and are not permanent, actual obstacles. As low-level sensor data, they can change from one sensing moment to the next, i.e., they are replaced each time new sensor data are obtained. In other words, atomic obstacles have a lifespan of only one sensing interval, either changes in the environment or changes in sensing direction can change atomic obstacles from one sensing instant to the next.

Consider the case of stereo vision sensing that provides an image of the environment at any sensing moment. Every pixel (i, j) of that image maps to a spherical surface region W_{ij} of 3-D points in the physical world. The sensor generates the 3-D point (x, y, z) in W_{ij} that is closest to the image plane, with distance d_{ij}. W_{ij} can be viewed as the projection of the square pixel on the image plane to the sphere centered at the origin of the camera frame $\{C\}$ with radius d_{ij}. Thus, W_{ij} has four corners. The four rays originating from the origin of $\{C\}$ and passing through the four corners of W_{ij} define a trapezoidal *viewing frustum* V_{ij}.

An atomic obstacle O_{ij} is formed by W_{ij} and the occluded space behind it (see Figure 1). We call W_{ij} the *front face* of the atomic obstacle O_{ij}. The environment can now be viewed as a set $S_{AO}(\tau)$ of *only* these atomic obstacles O_{ij}, $\forall(i, j)$ in the image, *for one sensing instant* τ. If S_O is the set of actual physical obstacles in the environment, then, $S_O \subset S_{AO}(\tau)$, but $S_{AO}(\tau)$ changes over sensing time τ because an occluded space at τ may no longer be occluded at a later time $\tau + \Delta\tau$.

B. Dynamic Envelope: A Review

It is reasonable to assume that even in an unknown and unpredictable environment, all obstacle speeds are bounded to be below a certain maximum possible speed v_{max}. Of course, an obstacle may have varied actual speeds in $[0, v_{max}]$. To be safe, v_{max} can be quite over-estimated, but [1] shows that the approach for discovering guaranteed collision-free CT-points is robust for over-estimated v_{max}.

With a given v_{max}, and $R(\mathbf{q})$ indicating the region in the physical space occupied by a robot at configuration $\mathbf{q}$, a dynamic envelope relating the robot at configuration $\mathbf{q}$ and a future time t to the current sensing time τ is defined below [21].

Definition 1: For a CT-point $\chi = (\mathbf{q}, t)$, a *dynamic envelope* $E(\chi, \tau_k)$, $k = 0, 1, 2, ...$, as a function of sensing time $\tau_k \leq t$, is a closed surface enclosing $R(\mathbf{q})$ in the physical space so that the distance between any point on $E(\chi, \tau_k)$ and the surface of $R(\mathbf{q})$ is,

$$d_k = v_{max}(t - \tau_k) \tag{1}$$

The following are major properties of a dynamic envelope $E(\chi, \tau_k)$, which capture non-worst case future obstacle motions, without assuming any particular kinds of obstacle motion:

1) A dynamic envelope shrinks monotonically over sensing time with speed v_{max}, i.e., $E(\chi, \tau_k) \supset E(\chi, \tau_{k+l})$, where $l > 0$, $\tau_k < \tau_{k+l} \leq t$.
2) An obstacle outside $E(\chi, \tau_k)$ will never intersect $E(\chi, \tau_{k+l})$, since an obstacle cannot move faster than v_{max}.
3) An obstacle intersecting $E(\chi, \tau_k)$ can be "squeezed" out of $E(\chi, \tau_{k+l})$, for certain τ_{k+l}, if *not* moving towards $R(\mathbf{q})$ in the maximum speed v_{max}.

Thus, at sensing time $\tau_k < t$, if the dynamic envelope $E(\chi, \tau_k)$ is detected free of atomic obstacles, it is also free of actual obstacles, and the robot will surely be collision-free at $\chi = (\mathbf{q}, t)$; else the robot may or may not be collision-free at $\chi = (\mathbf{q}, t)$ (i.e., it is uncertain).

Hence, the concept of dynamic envelope, coupled with atomic obstacles, enables the detection of collision-free CT-points without requiring to know or recognize obstacles in an unknown and changing environment. By observing a shrinking dynamic envelope over time, one can catch the earliest sensing moment when a (future) CT-point is perceived collision-free with the following Algorithm 1, called the *Collision-free Perceiver* (CFP).

Algorithm 1 Collision-Free Perceiver (CFP)

1: Input the set S_{CT} of CT-points
2: $k = 1$, $\tau_k =$ current time
3: **for each** CT-point $\chi = (\mathbf{q}, t) \in S_{CT}$ **do**
4: create dynamic envelope $E(\chi, \tau_k)$
5: **end for**
6: **while** $\tau_k < t$ and (not time-limit) and $S_{CT} \neq \emptyset$ **do**
7: **for each** CT-point $\chi = (\mathbf{q}, t) \in S_{CT}$ **do**
8: **if** no atomic obstacle intersects $E(\chi, \tau_k)$ **then**
9: **output** χ as guaranteed collision-free
10: $S_{CT} = S_{CT} - \{\chi\}$
11: **end if**
12: **end for**
13: $k = k + 1$ (Next sensing moment)
14: **end while**
15: **output** the remaining CT-points in S_{CT} may not be collision-free
16: **return**

The input CT-points to CFP for checking should be determined by an on-line motion planner. Clearly the key step in Algorithm 1 is **step 8** to check whether any atomic obstacle intersects with the dynamic envelope of a CT-point for any sensing instant. In the next section, we introduce an efficient algorithm called the *Intersection-checking between Dynamic Envelope and Atomic ObstacleS* (IDEAOS) algorithm for performing such intersection checking.

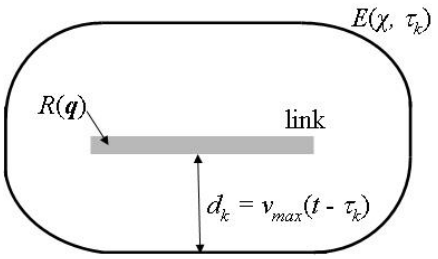


Fig. 2. Dynamic envelope of a link OBB

III. The IDEAOS Algorithm

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 [14]–[16]. Now, a dynamic envelope can be created for each link. For a rectangular rod robot, the dynamic envelope is shown in Figure 2. Since the dynamic envelope for the entire robot is the union of the dynamic envelopes of links, we just focus on how to check the intersection between the dynamic envelope of a link and the atomic obstacles. Since each atomic obstacle corresponds to a pixel of a stereo vision image, depending on the image resolution, there can be a great number of atomic obstacles. For example, even an image with a coarse resolution of 188×120 generates up to 22,560 atomic obstacles. Thus, key to the real-time efficiency of the IDEAOS algorithm is how to manage large number of atomic obstacles to minimize the number of intersection computations. Our algorithm uses the following strategies, detailed in the subsections:

- **Extraction**: Consider only 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.
- **Grouping**: Partition pixels on $P(E)$ into multi-size *super pixels*, such that each super pixel corresponds to a $m \times n$ image region of $P(E)$, with varied $m(\geq 1)$ and $n(\geq 1)$ values. The atomic obstacles corresponding to a super pixel on $P(E)$ form a *combined atomic obstacle*. With such grouping, intersection check is reduced to that between the dynamic envelope and combined atomic obstacles (which are far fewer than atomic obstacles).
- **Hierarchical Checking**: Perform intersection checks efficiently through multi-level simplified computations by subdividing the combined atomic obstacle into smaller ones when an intersection is detected between a dynamic envelope and that combined atomic obstacle. Thus, if no intersection is detected at a high-level, then there is no intersection for sure; else, re-check intersection at a lower level.

The above strategies are interweaved in the IDEAOS algorithm to take advantage of the fact that the algorithm only needs to produce a binary result, i.e., whether any atomic obstacle intersects the dynamic envelope or not. We will first explain these strategies in detail and then show how they are organized in the IDEAOS algorithm.

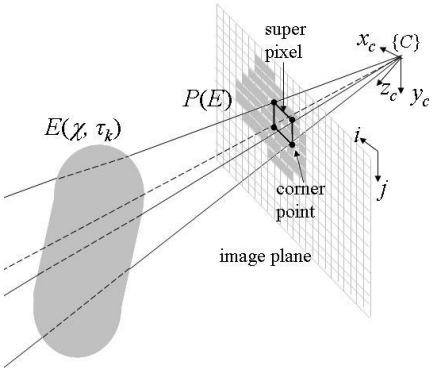


Fig. 3. Ray intersection tests to detect whether the super pixel is an internal super pixel of $P(E)$ with the $P(E)$'s boundary unknown

A. Extraction and Grouping

The idea here is to identify super pixels of varied m and n values to partition the projection $P(E)$ of the dynamic envelope on the image plane without explicitly computing the boundary of $P(E)$. A super pixel has four corner points as shown in Figure 3, and a ray that originates from the origin of the camera frame $\{C\}$ and passes a corner point is called a *corner ray*. If all corner rays of the super pixel intersect the dynamic envelope, then the super pixel is called an *internal super pixel*; else, if at least one corner ray intersects with the dynamic envelope, the super pixel is called a *boundary super pixel*.

Our strategy simultaneously discovers $P(E)$ and covers it by a partition of super pixels of multiple sizes. We say the region already discovered and partitioned by internal super pixels the *discovered region* of $P(E)$, and the remaining region of $P(E)$ the *undiscovered region* of $P(E)$.

Starting from an entirely undiscovered $P(E)$, our strategy first finds a seed $m \times n$ super pixel on the image plane by determining the upper leftmost pixel of the region and the m and n values. The ideal seed should be the maximum axis-aligned rectangular region that fits inside $P(E)$. However, since $P(E)$ is not known precisely, we make a reasonable estimate based on the OBB of the dynamic envelope and its projection on the image plane. Values of m and n should be chosen large enough because they will only be reduced later.

Next, our strategy consists of the following steps:

- **Extract**: from the seed super pixel, find all neighboring internal or boundary super pixels on the originally undiscovered regions of $P(E)$ and their neighbors, etc., in a fashion similar to connected neighborhood expansion of the flood-fill algorithm [22], [23]. Put every boundary super pixel in a first-in-first-out (FIFO) queue B. Figure 4 illustrates this process.
- **Re-seed**: Remove the first boundary super pixel from B and if it is larger than a minimum size[1] $m_0 \times n_0$,

[1]determined to make sure that the area of partition covering $P(E)$ is not much larger than $P(E)$ and yet also without using too many super pixels.

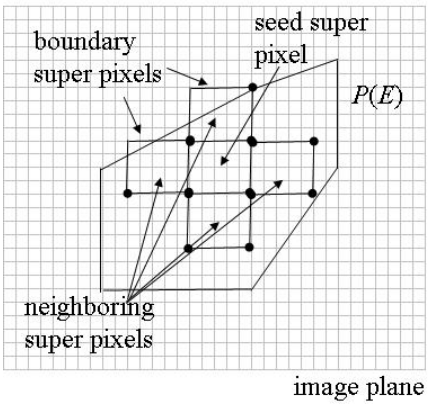


Fig. 4. Neighborhood expansion of super pixels

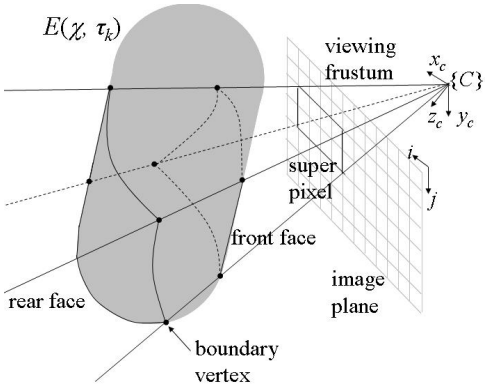


Fig. 5. Illustration of some notations

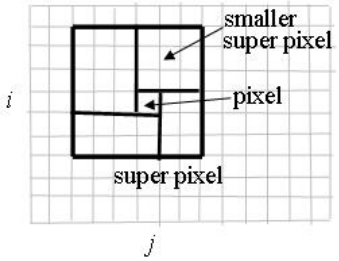


Fig. 6. Division of a super pixel into one pixel and four smaller super pixels

which is proportional to the initial $m \times n$, then

- for every corner point p inside $P(E)$, reduce m and n to halfs with p fixed to get a smaller super pixel, and go to **Extract**.

- **Return** the internal and boundary super pixels of various sizes of $P(E)$.

Note that the FIFO queue B is in the order of decreasing size of the boundary super pixels. Therefore, when the first $m_o \times n_o$ boundary super pixel is removed from the queue, the remaining queue has only $m_o \times n_o$ boundary super pixels. Thus, the partition is complete.

The above strategy generates a partition of super pixels for $P(E)$ efficiently without requiring excessive intersection checking that a standard cell decomposition method, such as a quadtree [24] requires. Our method discovers pixels or super pixels of $P(E)$ through connected expansion (from a seed by flood-fill) and thus avoids the problem of having a cell with all four corner points outside of $P(E)$ but possibly some internal points inside $P(E)$. Whereas, such a cell is possible with quadtree decomposition, and it is expensive to check if the cell is entirely outside $P(E)$, which may involve intersection checking beyond just between corner rays and the dynamic envelope.

B. Hierarchical Checking

Once a super pixel of $P(E)$ is determined, we can next check if its corresponding combined atomic obstacle intersects with the dynamic envelope or not.

We first introduce a few related terms and notations (see Figure 5) as following:

Viewing frustum of a super pixel: the rectangular pyramid region defined by the four rays originating from the origin of $\{C\}$ and passing through the four corner points of a super pixel. It extends to infinity and is the union of viewing frustums of all pixels that form the super pixel.

Minimum distance d_{cao} of a combined atomic obstacle: the distance from the origin of $\{C\}$ to the atomic obstacle closest to the origin of $\{C\}$ in the combined atomic obstacle.

Front and rear faces F_{de} and R_{de} of a dynamic envelope intersected by a viewing frustum: the intersection surface regions between the viewing frustum of a super pixel (or a pixel) and the dynamic envelope, where the region closer to the camera is the *front face* and the other region is the *rear face*. Note that the two faces become one if the viewing frustum partially intersects the dynamic envelope, i.e., not all of its corner rays intersect the dynamic envelope, and it corresponds to a boundary super pixel of $P(E)$.

Given a super pixel, if the corresponding combined atomic obstacle is behind R_{de} of the dynamic envelope as viewed from the camera, then there is no intersection between the combined atomic obstacle and the dynamic envelope; otherwise, there is an intersection.

We adopt a hierarchical refinement approach to combine simple checking and narrowing scope of consideration. Our approach takes advantage of the intersection results between corner rays of the combined atomic obstacle and the dynamic envelope (described in section III.A) to obtain eight *boundary vertices* of the front face and rear face of the dynamic envelope. The recursive algorithm *HierCheck* implements our approach that returns either intersection or no intersection, given a super pixel.

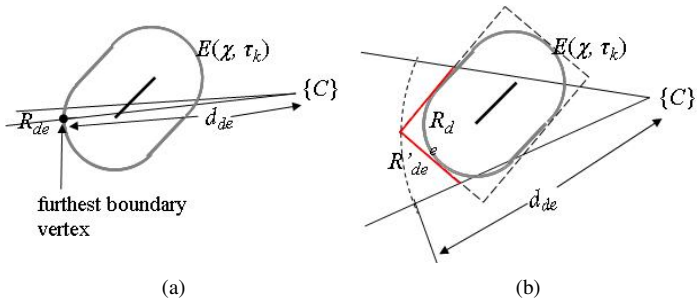
In *HierCheck* step 1, d_{cao} is obtained from comparing the distances of all atomic obstacles, which are the readings of stereo vision sensor, in the combined atomic obstacle.

In *HierCheck* step 2, d_{de} is obtained based on the OBB of the dynamic envelope if R_{de} is so large that it cannot be viewed as a flat surface. Let R'_{de} be the surface corresponding to R_{de} on the OBB of the dynamic envelope. d_{de} is the distance from the furthest feature (i.e., vertex, edge, or face) of R'_{de} to the camera origin. If, however, R_{de} is small enough

Algorithm 2 HierCheck (super pixel)

1: Find d_{cao} and the corresponding atomic obstacle O_{ij}.
2: Compute d_{de} as a small upper bound on the greatest distance from R_{de} to the origin of $\{C\}$.
3: **if** $d_{cao} \leq d_{de}$ (i.e., possible intersection), **then**
4: **if** super pixel is a pixel or d_{de} is less than the minimum distance from a boundary vertex to the origin of $\{C\}$ **then**
5: **return** intersection
6: **end if**
7: Divide the super pixel into one pixel at (i, j) and four smaller super pixels as shown in Figure 6.
8: **if** *HierCheck*(pixel) = no intersection **then**
9: **for each** smaller super pixel **do**
10: **if** *HierCheck*(smaller super pixel) = intersection **then**
11: *return* intersection
12: **end if**
13: **end for**
14: **end if**
15: **end if**
16: **return** no intersection

to be considered as a flat surface, then d_{de} is the maximum distance from a boundary vertex of R_{de} to the origin of $\{C\}$. See Figure 7 for illustrations of the two cases.



(a) (b)

Fig. 7. Illustrations of two cases of face R_{de}: (a) flat, and (b) not flat

Steps 3-16 of *HierCheck* shows that if a possible intersection is detected based on the simple inequality, it is necessary to decompose the combined atomic obstacle into smaller ones and do the checking again on them, i.e., recursively moving down to lower levels of the computation hierarchy, until either no intersection is detected or an intersection is detected between a single atomic obstacle and the dynamic envelope (or its OBB).

Hence, our approach hierarchically refines the accuracy of computation based on need. The ultimate resolution of computation depends on the resolution of the stereo vision image, and also how far R_{de} is from the camera, which reflects how far $R(\mathbf{q})$ (i.e., the region occupied by the robot at configuration $\mathbf{q}$) is from the camera and how far ahead the considered future time t is from the current sensing instant τ. The closer $R(\mathbf{q})$ is to the camera, and the closer t is to τ, the more accurate is the result of "intersection" from **HierCheck**.

However, it is important to note that the result of "no intersection" is always absolutely accurate, and thus, if our algorithm reports no intersection between a dynamic envelope $E(\chi, \tau)$ and atomic obstacles, the CT-point $\chi = (\mathbf{q}, t)$ can be said guaranteed collision-free based on sensing at τ.

TABLE I
COSTS OF COMPUTING INTERSECTIONS BETWEEN A RAY AND AN OBJECT

Object	$+, -$	$\times$	$<, >, =$	$/$	$\sqrt{\ }$
OBB	120	108	48	6	0
Sphere	8	11	0	1	1
Capsule	29	48	0	3	3

C. Integration of Strategies

The IDEAOS algorithm integrates the above strategies (i.e., extraction, grouping, and hierarchical checking) in a way to be more efficient. It does not find the partition of $P(E)$ (i.e., the projection of the dynamic envelope to the image plane) at once, but rather, after it generates one super pixel on $P(E)$, it proceeds to hierarchical checking of whether the corresponding combined atomic obstacle intersects with the dynamic envelope or not. If an intersection is detected, IDEAOS simply returns "there is intersection" and halt. Otherwise, it generates another super pixel on $P(E)$ (in the flood-fill fashion) and do the hierarchical checking again, and so on. Therefore, IDEAOS will only generate enough combined atomic obstacles to find an intersection, and if there is no intersection, the algorithm will halt only after processing all the combined atomic obstacles whose super pixels partition $P(E)$, through hierarchical checking.

IV. TIME AND SPACE COHERENCE

Recall that the IDEAOS algorithm is for realizing step 8 of the Algorithm 1 (CFP) in Section II.B. CFP repeatedly calls the IDEAOS algorithm for intersection checking for different CT-points and at different sensing intervals. It can take advantage of time and space coherence to reduce computation significantly.

For the same CT-point $\chi = (\mathbf{q}, t)$, if the camera does not move from sensing moment τ_k to τ_{k+1}, then many super pixels (or pixels) on $P(E(\chi, \tau_k))$, i.e., the projection of $E(\chi, \tau_k)$ on the image plane, are also on (the smaller) $P(E(\chi, \tau_{k+1}))$. Thus, the low-level checking results between the corresponding rays and $E(\chi, \tau_k)$ can be re-used.

Similarly, if two CT-points have sufficiently close configurations $\mathbf{q}$ and $\mathbf{q}+\Delta\mathbf{q}$, even with different future times t and $t + \Delta t$, their respective dynamic envelopes for any sensing moment τ_k could overlap significantly, and then, again, some low-level intersection checking results of one CT-point can be re-used for the other CT-point.

Table I shows the costs of computing the intersection between one ray and one object of each of the given types. If the result is re-usable, such costs are saved.

Moreover, in step 1 of algorithm **HierCheck**, the considered atomic obstacles are divided into small groups and

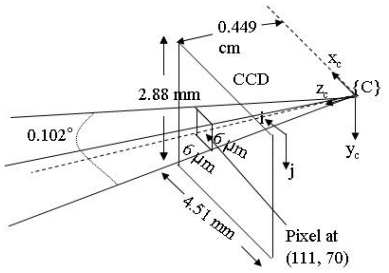


Fig. 8. Dimension of atomic obstacles as defined by the CCD of the camera for resolution 752 × 480

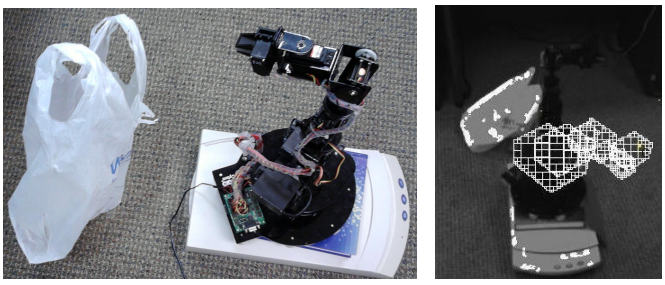
the local minimum distance in each group is computed first. Next the minimum distance is computed from the group minimums. In that way, many group minimum distances can be re-used if the corresponding atomic obstacles are shared by other dynamic envelopes (of different CT-points) at the same sensing time τ_k.

V. Implementation and Experimental Results

We have implemented the IDEAOS algorithm as the core algorithm for the CFP (Algorithm 1) on a Dell Precision T5400 computer and tested the algorithm in real-world experiments using a real 7-DOF Robai's Cyton robot arm and an indoor MobileRanger's stereo vision camera. The robot has 7 revolute joints. Each robot link is approximated by an OBB and so is its dynamic envelope.

An atomic obstacle's geometry along with the dimensions is shown in Figure 8. The sensor resolution was fixed to 752×480 and thus, at most 360,960 atomic obstacles were generated at each sensing moment. The sensing frequency was set at 8 Hz. v_{max}= 1 cm/s.

Figure 9(a) shows the environment of the robot, where two obstacles unknown to the robot are nearby, one is a scanner underneath the robot, and the other is a plastic bag. Figure 9(b) shows the same environment as viewed by the stereo vision sensor at sensing time τ_0=0s, superimposed with the projection image of the dynamic envelope of every link of the robot, which is partitioned by super pixels, for the CT-point χ_1=$(\mathbf{q}_1, t_1)$, and $\mathbf{q}_1$ is the configuration where every joint angle of the robot is at -90^o, and t_1=2s. Note that $\mathbf{q}_1$ is not the robot's current configuration in the figure.



(a) (b)

Fig. 9. (a) Testing environment, (b) Stereo vision image at τ_0= 0(s) and for the CT-point χ_1, with the partitioned projection image $P(E)$ of the robot's dynamic envelope $E(\chi_1, \tau_0)$.

TABLE II

Results of running IDEAOS for two CT-points at τ_0

CT-points		χ_1	χ_2
# super pixels	16×16	13	41
	8×8	77	78
	4×4	156	182
	2×2	291	402
	boundary 2×2	640	729
	total	1177	1432
# hierarchy		2	5
# combined atomic obstacles checked		1248	1542
# atomic obstacles involved		14,476	22,924
time cost (s)		0.265	0.328
intersection		no	yes

Table II shows the results of running the IDEAOS algorithm at sensing time τ_0 for two different CT-points χ_1 as introduced above and $\chi_2 = (\mathbf{q}_2, t_2)$, where $\mathbf{q}_2$ is the configuration with all 7 joint angles of the robot at 90^o respectively and t_2=5.57s. The results are for the entire robot, i.e., summing up the results for the dynamic envelopes of 7 individual links. Initial m and n were both set to be 16. $m_0 \times n_0$ was set to 2×2.

As shown in the table, the total number of combined atomic obstacles checked for each CT-point is more than 10 times smaller than the total number of atomic obstacles involved. That implies great saving of computation time in the similar order of magnitude by the IDEAOS algorithm rather than using a naive algorithm to directly check intersections of atomic obstacles with the dynamic envelope. Note that the total number of combined atomic obstacles is greater than the total number of super pixels generated in each case because hierarchical checking divided some combined atomic obstacles into smaller ones for checking.

For χ_1, the result of no intersection means that the CT-point is guaranteed collision-free. $P(E)$ was actually covered by the super pixels generated as shown in Figure 9(b). Because of the efficency of IDEAOS, χ_1 is discovered collision-free 1.73s *before* its time t_1=2s, which is important for on-line motion planning to utilize that CT-point. The time cost 0.265s of the detection is only about 1% of the total time between the sensing moment τ_0=0s and t_1= 2s. Recall that no actual obstacle was ever identified or recognized.

For χ_2, because IDEAOS halted as soon as an intersection was found, the super pixels generated were only a subset of super pixels for covering $P(E)$. $P(E)$ and the corresponding dynamic envelope for χ_2 was much larger than those for χ_1 because t_2=5.57s was close to three times of t_1=2s. The result of intersection means that it was not certain whether χ_2 was collision-free or not based on the sensing data at τ_0=0s. Further checking with new sensing data was necessary.

Table III shows the results of running IDEAOS by CFP for a CT-point $\chi_3 = (\mathbf{q}_3, t_3)$ multiple times with sensing data from six different sensing instants τ, where $\mathbf{q_3}$ is the configuration that every joint angle of the robot is set to 23.39^o, and t_3=4.05s. The results show the reduction of the computation cost by exploiting the sensing time coherence. IDEAOS was used to check the CT-point χ by generating

TABLE III

RESULTS OF RUNNING IDEAOS FOR A CT-POINT χ_3 WITH DATA FROM DIFFERENT SENSING INSTANTS WITHIN $[0.66, 1.52]$ (S)

τ (s)	time cost (s)	# super pixels generated
0.66	0.109	950
0.74	0.047	889
0.86	0.031	889
1.25	0.344	858
1.38	0.094	852
1.52	0.047	842

all low-level intersection checking results from scratch based on the sensing data at $\tau = 0.66$s. Later, however, for $\tau \in (0.66, 0.86]$ (s), useful previous results were repeatedly re-used so that even though the total number of super pixels decrease only slightly as the dynamic envelope shrunk over time, the drop in computation time was far more drastic. Similarly, for $\tau \in (1.25, 1.52]$ (s), the results obtained from $\tau = 1.25$s was re-used repeatedly to reduce the time cost.

Notice that the time cost for τ= 0.66s is smaller than that for τ = 1.25s, even though the dynamic envelope for the latter was much shrunk from that of the former. This was because as the dynamic envelope shrunk over time, at τ = 1.25s, there were fewer atomic obstacles in the dynamic envelope. Thus, with fewer intersections present, the algorithm had to spend more time to find one of those intersections.

Note that the CT-point $\chi_3 = (\mathbf{q}_3, t_3)$ was discovered collision-free, i.e., there was no intersection between the dynamic envelope and atomic obstacles, at τ=2s, which was 2.05s ahead of the time t_3=4.05s. The attached video clip shows the process of running IDEAOS for checking χ_3 based on the sensory data at τ=2s.

VI. CONCLUSIONS

This paper presents an efficient algorithm IDEAOS to check whether a robot at a future time t and a configuration $\mathbf{q}$, i.e., a CT-point $(\mathbf{q}, t)$, will be guaranteed collision-free or not based on real-world stereo vision sensing of the robot's environment at the current time τ. The algorithm can be used by on-line motion planning algorithms for any robot, including manipulators and mobile robots, in an environment with unknown obstacles and unknown motions without the need of identifying and tracking the obstacles. By exploiting time and space coherence, the algorithm has an even lower computation cost per CT-point on average when it is used to check multiple CT-points at the same or different sensing instants. Thus, the algorithm is well suited to detect whether a sequence of CT-points (on a trajectory) is guaranteed collision-free or not before the time period of the trajectory. Moreover, strategies in the IDEAOS algorithm can be used to handle atomic obstacles from other types of sensors, such as laser range finders, sonars, etc.

As the next step, we will explore how to best relate sensing frequency with the time cost of IDEAOS to ensure sensing effectiveness and computational efficiency. We will also systematically take into account sensing uncertainty in our algorithm to ensure the robustness of results under uncertainty.

REFERENCES

[1] R. Vatcha and J. Xiao, "Perceived CT-space for motion planning in unknown and unpredictable environments," in *WAFR*, Dec 2008.

[2] F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Trans. On Robotics*, vol. 25, pp. 902–911, Aug 2005.

[3] K. B. Ariyur, P. Lommel, and D. F. Enns, "Reactive inflight obstacle avoidance via radar feedback," in *American Control Conf.*, pp. 2978–2982, 2005.

[4] Y. Yagi, H. Nagai, K. Yamazawa, and M. Yachida, "Reactive visual navigation based on omnidirectional sensing – path following and collision avoidance," *J. Intell. Robotics Syst.*, vol. 31, no. 4, pp. 379–395, 2001.

[5] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *Intl. J. of Robotics Research*, vol. 21, pp. 999–1030, 2002.

[6] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments.," in *Robotics Science and Systems II*, The MIT Press, 2006.

[7] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite rrts for rapid replanning in dynamic environments," in *IEEE Intl. Conf. on Robotics and Automation*, pp. 1603–1609, 2007.

[8] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," in *Intl. J. of Robotics Research*, pp. 760–772, 1998.

[9] 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)*, pp. 734–739, 2002.

[10] 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), pp. 1199–1212, Oct. 2008.

[11] 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*, pp. 82–87, October 2004.

[12] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *IEEE Intl. Conf. on Robotics and Automation*, pp. 1662–1668, May 2009.

[13] 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*, pp. 2366–2371, May 2006.

[14] 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.*, pp. 189–196, 1995.

[15] P. Jimnez, F. Thomas, and C. Torras, "3D collision detection: A survey," *Computers and Graphics*, vol. 25, pp. 269–285, 2000.

[16] M. C. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *In Proc. of IMA Conf. on Mathematics of Surfaces*, pp. 37–56, 1998.

[17] A. Murarka, M. Sridharan, and B. Kuipers, "Detecting obstacles and drop-offs using stereo and motion cues for safe local motion," in *IROS*, pp. 702–708, 2008.

[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, "Perceived ct-space for discovering collision-free robot trajectories in unknown and unpredictable environments," *submitted to IEEE Transaction on Robotics*.

[22] A. Treuenfels, "An efficient flood visit algorithm," *C/C++ Users J.*, vol. 12, no. 8, pp. 39–62, 1994.

[23] P. S. Heckbert, "A seed fill algorithm," pp. 275–277, 1990.

[24] J. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.