The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Planning Collision-Free and Occlusion-Free Paths for Industrial Manipulators with Eye-to-Hand Configuration

Simon Léonard, Elizabeth A. Croft and James J. Little

Abstract— **This paper presents a motion planning algorithm for industrial manipulators with the simultaneous constraints of avoiding collisions and avoiding the occlusion of specified pixellated regions of an eye-to-hand camera. The system uses a probabilistic roadmap to satisfy the constraints imposed by the command interface of typical industrial manipulators and uses dynamic collision checking to ensure collision-free motion. In the context of a task monitored by a camera, we enhance a probabilistic roadmap with a dynamic occlusion checking algorithm that is able to determine which pixels of the camera are occluded by the robot during each motion segment. The occlusion algorithm is formulated as collision algorithm where the field of view of the camera is represented as a quadtree of frustums. The proposed algorithm is demonstrated in industrial bin picking simulations where the gripper must not occlude the targeted object throughout the task.**

I. Introduction

Recent advances in path planning are enabling robots to perform remarkably complex tasks. Likewise, advances in sensors and control are enabling robots to operate in environments that were once considered hostile to robotic systems. Most of the work done in this area, however, is not suitable for industrial robot systems. Thus, despite compelling progress in robotics, robots used in factories are still confined to repeating well structured tasks. Although planning algorithms and sensors are used to a certain extent in industrial environments, there is still a large gap between the tasks that an average industrial robot executes on an assembly line and the tasks performed in research labs.

In robotics research, vision is often used to control the motion of robots during the execution of a task [1]. An inherent feature of visual servoing is that each joint of the robot is controlled by its velocity. Typically, this approach cannot be directly applied to industrial manipulators as the manufacturers for these robots only provide position commands and these commands must be specified ahead of their execution. Robot speed is then commanded as a scaled value over a point to point path. Furthermore, the trajectories available for these systems typically do not provide the flexibility to compute complex paths. This limitation is problematic for picking up an object with an industrial manipulator using visual feedback since this requires the planner to find a sequence of collision-free linear trajectories that also preserves the visibility of the object.

S. Léonard is with the Department of Mechanical Engineering, The University of British Columbia `sleonard@mech.ubc.ca`

E.A. Croft is with the Department of Mechanical Engineering, The University of British Columbia `ecroft@mech.ubc.ca`

J.J. Little is with the Department of Computer Science, The University of British Columbia `little@cs.ubc.ca`

To address the gap between current robotics research and industrial applications, this paper proposes a novel algorithm to find collision-free and occlusion-free paths that are suitable for industrial robot manipulators.

In this work, a camera is fixed at a given position and orientation and views the environment. The manipulator can occlude areas within the camera's field of view. In the case where the robot must perform a task that requires visual feedback from the camera (e.g., tracking a visual target), our solution finds a path to perform the task while avoiding the occlusion of one or several image targets used to monitor the execution of the task. The main contribution of this research is a dynamic occlusion checking (DOC) algorithm that is capable of determining which pixels are occluded by the manipulator during a proposed motion. Our solution adapts the dynamic collision detection algorithm presented in [2] to a dynamic vision problem by modeling occlusions as collisions between an object and the frustum of a pixel. The frustums are then organized in quadtree to enable efficient pruning of collisions. Although our algorithm is conceptually similar to view frustum culling [3] the main difference is that our algorithm aims to cull a moving kinematic chain.

The backbone of our solution is a probabilistic roadmap with edges representing collision-free motions [4]. The DOC algorithm is then used on each edge to determine the pixels that "see" the manipulator during the motion. These pixels are stored in a data structure and are used to adjust the weights of the edges. Given that an image target is defined by a subset of pixels, the edges that occlude the target are not considered in the solution of the path.

This paper is arranged as follow. Section II briefly reviews relevant work done in visual servoing and path planning with visual constraints. Section III outlines the planning strategy used in this research and introduces the DOC algorithm. Section IV presents simulation results, including a performance evaluation, and Section V discusses future work.

II. Previous Work

The work presented herein is complementary to visual servoing, namely the use of visual feedback to control the motion of robots [1], [5]. The most common approach to solving the visual servoing problem is to linearize the system by using a first order time derivative. This operation results in an interaction matrix that relates the velocities of image pixels to the velocity of the camera. Visual servoing has been extended to add constraints on the path followed by a robot such as joint limits and field of view [6], [7].

Much of the research in motion planning focuses on finding paths to avoid collisions with obstacles. The motion planning literature proposes several algorithms that are anchored around sampling the configuration space [8]. Among the sampling-based methods, probabilistic roadmaps (PRM) [4] have a natural appeal for industrial applications since they enable multiple queries and the samples can be connected by collision-free motion segments that are compatible with industrial systems. With the increasing importance of sensors in robotics it is also crucial to consider the constraints imposed by sensors such as cameras when searching a solution for a path. For example, if a camera is used to sense the environment or monitor a task it is important that the motion of the robot does not interfere with the sensor. Recent research has demonstrated the importance of planning the path of a camera mounted on the end-effector to avoid occlusions of an image target by obstacles [9]. To find an occlusion-free path the visible region of a target is computed and any motion crossing the region's boundary is penalized as it occludes the image target. This research, however, does not address the problem of using a fixed camera where the sources of occlusions are the links of a manipulator. Thus, if a fixed camera is used to monitor a task, such as picking an object with a gripper, a key constraint to any solution is to avoid a visual occlusions of the target by one of the manipulator's link.

This paper presents an algorithm to address the aforementioned problem of finding collision-free and occlusion-free paths for a standalone camera. Specifically, this paper introduces the dynamic occlusion checking (DOC) algorithm to determine which pixels will become occluded if a manipulator moves between two configurations. Then the DOC algorithm is used to test each motion segment of a solution to avoid paths that result in the occlusion of an image target. The resulting solutions are conservative in that DOC does not require the *a priori* 3D coordinates of the image target. As such, the solution avoids motions for which the robot occludes an image target irrespective of its depth.

III. Probabilistic Roadmap

Probabilistic roadmaps define a sparse representation of a robot's configuration space. As for other sampling-based algorithms, N configurations $\mathbf{q}_1, \ldots, \mathbf{q}_N$ are sampled and used to construct an undirected graph.An edge $e_{i,j}$ is added to the graph if the motion between the configurations $\mathbf{q}_i$ and $\mathbf{q}_j$ is collision-free. A weight $w_{i,j}$ is associated with an edge $e_{i,j}$ to represent the inherent cost of using the motion represented by the edge. Typically, a weight will represent the length of a motion between two configurations but they are often used to represent more complex tasks.

A key element of the research presented in this paper is the role played by the collision checker. Collision checkers are used to test if an edge $e_{i,j}$ represents a collision-free motion. Many solutions have been proposed for this problem including using Minkowski sum [10] and detecting collisions at fixed time intervals [11]. The Minkowski sum approach represents an elegant solution but it is mostly used for mobile robots or for robots with a simple volumetric shapes. At the other end of the spectrum, sampling the motion at regular intervals and testing for collisions at each sample represents a brute force approach. This method is especially efficient for slow moving objects and thick obstacles otherwise collisions might not be detected. Schwartzer *et. al.* presented a very efficient dynamic collision checker for manipulators that guarantees to return collision-free motions. This algorithm has been modified to address computer vision problems such as preserving targets within the field of view [12] and avoiding occlusions [9]. The following section presents a novel adaptation of a dynamic collision checking (DCC) algorithm presented in [2] to detect occlusion caused by a robot moving in front of a fixed camera. The dynamic occlusion checking will then be used to test the edges of the PRM to ensure that the pixels associated with an image target are not occluded by the manipulator during the motion.

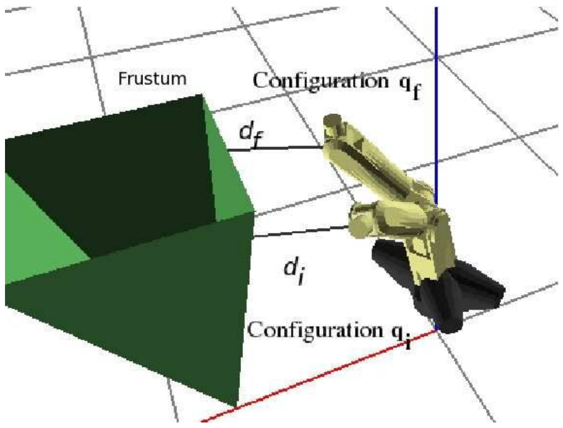

Fig. 1. Frustum of a camera and and a manipulator in its initial and final configurations.

A. Dynamic Occlusion Checking

As with view frustum culling [3], the DOC algorithm models the field of view of a pinhole camera as a frustum that acts as an obstacle in the workspace of a robot (Figure 1).

From the intrinsic and extrinsic parameters of the camera [13], the frustum of the camera is represented by 4 triangles. Using the frustum as an obstacle, the DCC algorithm is used to test if the robot moves within the field of view of the camera by testing for a collision between the links of the arm and the frustum.

The adaptive DCC algorithm is based on the idea that two bodies cannot collide if they are far away from each other and if their relative motion is sufficiently small. In the case of a manipulator with an immobile obstacle, let $\mathbf{q}_i$ and $\mathbf{q}_f$ be the initial and the final configurations of the robot and define the linear motion between $\mathbf{q}_i$ and $\mathbf{q}_f$ by $\mathbf{q}(t) = \mathbf{q}_i + (\mathbf{q}_f - \mathbf{q}_i)t$ for the parameter $0 \le t \le 1$. Furthermore, let $\ell(\mathbf{q}(t))$ be the longest distance traveled by any point on the manipulator during the motion $\mathbf{q}(t)$. Finally, let d_i be the initial distance between the robot and the obstacle and let d_f be the final distance. It can be shown that if

$$\ell(\mathbf{q}(t)) < d_i + d_f, \tag{1}$$

then the manipulator cannot collide with the obstacle [2]. If Equation 1 is not satisfied, then the inequality is applied recursively by splitting the motion $\mathbf{q}(t)$ in two intervals $\mathbf{q}(t_1)$ and $\mathbf{q}(t_2)$ with $0 \leq t_1 \leq 0.5$ and $0.5 < t_2 \leq 1$. The algorithm terminates when Equation 1 is satisfied for all the intervals or when a collision is detected.

In some applications using the entire camera field of view as an obstacle can prevent the execution of a task. For example, when monitoring a picking task with a camera, the hand must enter the field of view to pick up the object. What must be prevented, however, is the hand obstructing the object that must be picked. For such applications we can use the DCC on a subset of pixels. Since each pixel defines a frustum in space, the DCC can test each frustum for a collision. Since modern cameras provide a large number of pixels, a more efficient approach is proposed. The DOC algorithm organizes the pixels, or more precisely their frustums, in a quadtree [10]. The main advantage of using a quadtree is that it enables to use the DCC to prune out areas within the field of view. For example, in the best scenario, the DCC algorithm can prune out the entire field of view if it determines that a motion does not collide with the frustum of the field of view. The worst case scenario is when the robot blocks the entire field of view in which case the frustum of each pixel must be tested with DCC. The quadtree structure is organized as follow. The leaves of the tree consists of individual pixels along with their frustum. The leaves are grouped 2×2 and the bounding frustum is created for each group. This procedure is repeated until the root of the tree is reached. The root of the quadtree corresponds to the entire frustum of the camera.

The pseudo-code of the DOC algorithm is presented in Algorithm 1. The main modification between DOC and DCC is the call to *dorecursion*. Since the purpose of DOC is to identify all pixels that are occluded during the motion between $\mathbf{q}_i$ and $\mathbf{q}_f$, the algorithms does not return as soon as a collision is detected between a frustum and the robot. The purpose of Algorithm 1 is to traverse the quadtree by recursive calls to *dorecursion* (Algorithm 2). At each node in the tree, Algorithm 2 is used to mark the rooted subtree if any of the pixels in the subtree is occluded by the motion of the manipulator. The result of running Algorithm 1 for a given motion is a data structure that contains all the pixels that are occluded by the motion. Algorithm 1 only involves marking which pixel, or group of pixels, are occluded by a motion. This process does not depends on the shape of a specific object and, thus, is only executed once for each motion. Once all the motions are processed, the PRM planner can exclude any motion that occludes a given set of pixels. At this point the pixels occupied by an object are used at the query time to hide the occluding motions from the planer.

The call to *dorecursion* is outlined in the Algorithm 2. Line 1 of Algorithm 2 extracts the frustum obstacle from the node data structure. Lines 2 and 3 compute the distances

Algorithm 1 $DOC(\mathbf{q}_i, \mathbf{q}_f, root, robot)$

```
Require: An initial configuration $\mathbf{q}_i$
Require: A final configuration $\mathbf{q}_f$
Require: The root of a quadtree $root$
Require: A robot $robot$
1: if $dorecursion(\mathbf{q}_i, \mathbf{q}_f, root, robot)$ then
2:     $\mathbf{q}_m \leftarrow (\mathbf{q}_i + \mathbf{q}_f)/2$
3:     $DOC(\mathbf{q}_i, \mathbf{q}_m, root, robot)$
4:     $DOC(\mathbf{q}_m, \mathbf{q}_f, root, robot)$
5: end if
```

between the robot for both configurations and the frustum. The distance can be efficiently computed by using the swept sphere volumes presented in [14]. A distance less than zero indicates a collision. Line 4 evaluates the longest trajectory described by any point on the surface of the arm during the motion. This length can be bounded by an algorithm such as the one presented in [2]. If the robot does not collide with the frustum at either $\mathbf{q}_i$ or $\mathbf{q}_f$ (Line 5), then Line 6 determines if the frustum can be pruned for the motion. If the frustum can be pruned, then the algorithm returns false. Otherwise, the algorithm returns true if the node is a leaf. This indicates that a pixel of the quadtree cannot be pruned and the DOC algorithm must divide the motion segment. If the node represents a pixel that collides at either $\mathbf{q}_i$ or $\mathbf{q}_f$, then the pixel is marked as being occluded by the robot during the motion (Lines 12 to 14). If the node is not a leaf, then the result from Equation 1 is inconclusive and Lines 15 to 18 call *dorecursion* on the node's children to determine if any of them can be pruned.

Algorithm 2 $dorecursion(\mathbf{q}_i, \mathbf{q}_f, node, robot)$

```
Require: An initial configuration $\mathbf{q}_i$
Require: A final configuration $\mathbf{q}_f$
Require: The node of a quadtree $root$
Require: A robot $robot$
Ensure: The interval $[\mathbf{q}_i, \mathbf{q}_f]$ must be divided in two
1: $obstacle \leftarrow frustum(node)$
2: $d_i \leftarrow distance(robot, \mathbf{q}_i, obstacle)$
3: $d_f \leftarrow distance(robot, \mathbf{q}_f, obstacle)$
4: $l \leftarrow longesttrajectory(robot, \mathbf{q}_i, \mathbf{q}_f)$
5: if $0 < d_i$ and $0 < d_f$ then
6:     if $l < d_i + d_f$ then
7:         return false
8:     else if $isleaf(node)$ then
9:         return true
10:    end if
11: end if
12: if $d_i \leq 0$ or $d_f \leq 0$ and $isleaf(node)$ then
13:     Mark the pixel
14: end if
15: for $i = 1$ to 4 do
16:     $val[i] \leftarrow dorecursion(\mathbf{q}_i, \mathbf{q}_f, child(node, i), robot)$
17: end for
18: return $val[1]$ or $val[2]$ or $val[3]$ or $val[4]$
```

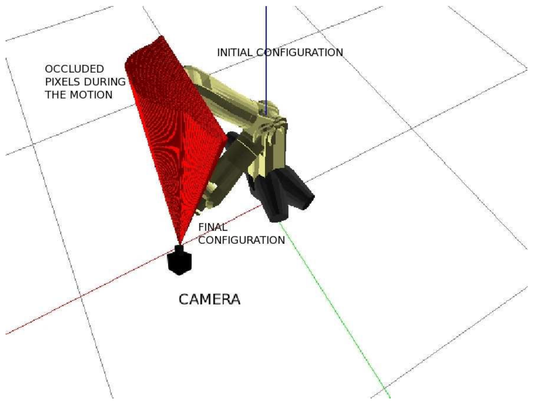

Fig. 2. The frustums of all occluded pixels for a motion between two configurations.

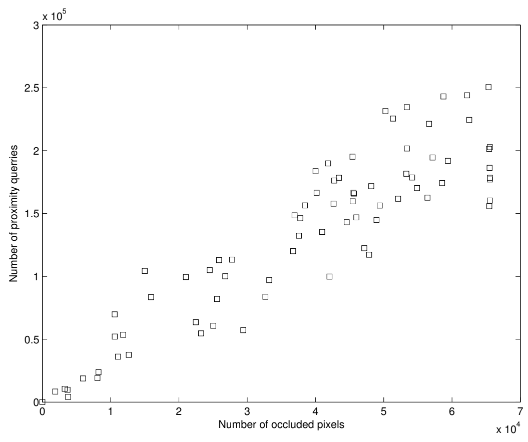

Fig. 3. Performance of the DOC algorithm for 100 random motions.

Within a PRM, Algorithm 1 is applied to each edge. Then for each edge, a list of pixels that are occluded by the motion can be stored. At query time, these lists are used by the planner to ensure that the edges that occlude the specified pixels will not be included in the motion of the robot.

IV. Experiments

To test the DOC algorithms and their use with a PRM motion planner, three experiments were conducted in simulations. The robot used in experiment is a CRS A460 and the camera intrinsic parameters were those estimated from a Point Grey Research Dragonfly with an 8mm lens. The first experiment tests the performance of the DOC algorithm for motions that enter the field of view of the camera. These experiments were used to evaluate the running time performance of the algorithm. The second experiment demonstrates the use of DOC for preventing the occlusion of a target by the manipulator during a picking task. Finally, the DOC algorithm was used interactively to create virtual obstacles from the images of the camera. The later experiment could be used to approximate real obstacles that are present in the environment without having to resort to computer assisted drawing tools to create them manually.

A. Experiment 1: DOC Performance

The DOC algorithm was tested outside the context of motion planning for the purpose of evaluating its performance in terms of running time. The worst case scenario where, a motion occludes every pixel, was investigated. For illustration purposes, the camera was placed on the floor in front of the robot and looking upwards. An example of a motion and the occluded pixels is shown in Fig. 2.

The bottleneck in Algorithm 2 comes from the repeated proximity queries between the robot and the frustums, with each query using approximately 0.5 millisecond. For the experiment the CCD array was cropped to 256×256 pixels. Since, the number of proximity queries is related to the number of pixels occluded and the computation time of proximity queries varies between implementations, Fig. 3 presents the results as a graph relating the numbers of pixels occluded versus the number proximity queries (*distance* calls). Although proximity queries between two objects can consume up to 1ms per call, the proximity queries involved with the DOC algorithm are more efficient due to the geometric simplicity of the frustums.

Fig. 3 shows that the performance of the DOC algorithm scales up linearly with the number of pixels occluded. In the worst case scenario, 65,536 pixels are occluded such that the number of proximity queries to traverse the full quadtree is 87,381. But because of the adaptive nature of the DOC algorithm this number increases when the motion is divided in smaller intervals (Algorithm 1 Lines 3 and 4).

Fig. 3 indicates that the performance of the DOC algorithm for motions that occlude a large number of pixels is not particularly good. Where the DOC algorithm is very efficient is for motions that occlude a small number of pixels. For example, for motions that occlude less than 20,000 pixels (roughly $^1/_3$ of the image), the algorithm is efficient. Since the purpose of DOC is to detect occlusions in order to avoid them, it makes little sense to keep the edges that occlude a large portion of the image. Edges related to these motions occlude a large portion of the image and they are unlikely to be used by the motion planners. Hence, within the context of PRM, the DOC algorithm can be used to discard an edge whenever its motion occludes a number of pixels that is beyond a given threshold. The advantage of discarding edges is two fold. First, it preserves "useful" edges in the graph, that is, edges that are likely to be used by the planner. Second, it keeps the running time of DOC within reasonable bounds. Such methods are often used in search algorithms to avoid finding useless solutions or solutions that are buried too deep in a search tree [15]. In the following experiments, a threshold of 20,000 occluded pixels was used to discard edges from the PRM.

B. Experiment 2: Occlusion-Free Motion Planning

The DOC algorithm was used in conjunction with a PRM to find collision-free and occlusion-free motions. Our task consisted of picking up an automotive part (conrod) that was placed on the ground in front of the robot. The camera was place above the part looking down on the part and the

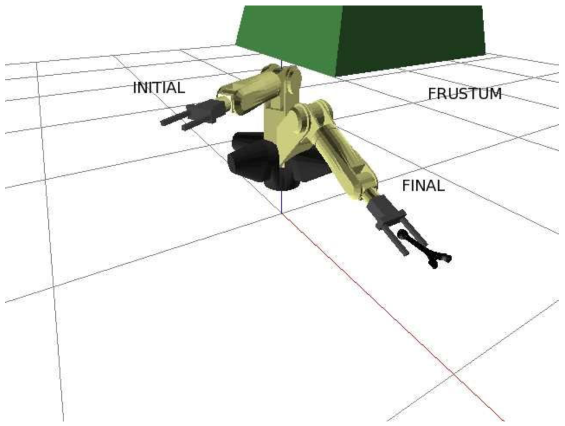

Fig. 4. Picking task with the initial configuration and the desire configuration. The camera is fixed above the workspace and looks down on the conrod.

min. angle	q_i	max. angle
$-\frac{\pi}{4}$	q_1	$\frac{\pi}{4}$
$-\frac{\pi}{2}$	q_2	0
$-\frac{\pi}{2}$	q_3	0
$-\pi$	q_4	π
$-\frac{\pi}{2}$	q_5	$\frac{\pi}{2}$
$-\pi$	q_6	π

TABLE I
JOINT INTERVALS USED TO GENERATE THE PRM OF EXPERIMENT 2.

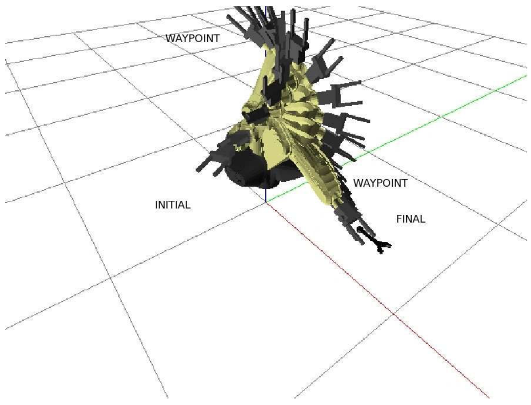

Fig. 5. Side view of the picking task with using two waypoints.

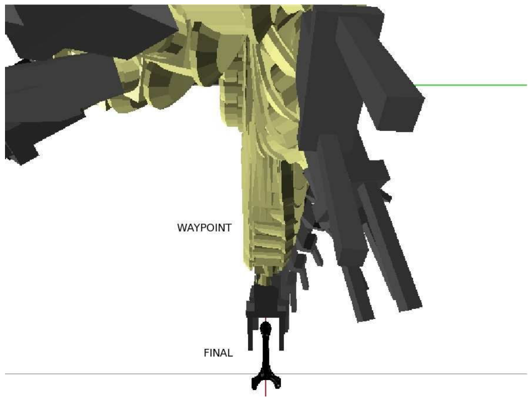

Fig. 6. Top view of the picking task with using two waypoints. This is the actual view from the camera.

workspace (Fig. 4).

The floor was defined as the sole physical obstacle and the conrod was placed 1cm above it to make room for the fingers of the gripper. All the pixels from the projection of the conrod in the camera were defined as obstacles and thus the gripper was not allowed to move in between the conrod and the camera. The PRM contained 100 vertices that were randomly samples within the intervals reported in Table I. These intervals correspond to configurations that position the gripper within the vicinity of the conrod.

The query for the planner consisted of the desired joint configuration to pick up the object. Obviously, this configuration must not occlude any of the specified target pixels or the goal will not be reachable. The desired and start configurations are illustrated in Fig. 4. The fingers of the gripper are 10cm apart and the stem of the conrod is about 2cm wide, which leaves a 4cm gap on both sides of the conrods. First, all the edges that occlude the conrod are rejected from any path. Second, DOC and DCC are used to connect the desired configuration to the nearest vertex using the L_2 norm. If the test fails, then the desired configuration cannot be reached from that vertex and an attempt to reach the goal is made from the 2nd nearest vertex and so on until all the vertices are exhausted. In the experiment reported in this paper, 6 attempts were necessary to connect the desired configuration to a vertex in the PRM. Finally, the path from the start configuration to the vertex is found from the PRM.

Fig. 5 and 6 illustrate the views of the workspace and a view from camera. Although only 2 waypoints were necessary, the path found by the planner is fairly complex as it swings the arm back and forth. Like many sampling-based methods, this is arguably a weakness for systems aimed at industrial applications. Nevertheless, the path returned by the planner did not occlude the conrod.

C. Experiment 3: Interactive Virtual Obstacle Generation

Setting up an industrial workcell for a manipulator requires configuring the robot and rehearsing the task. For a path planning task in an engineered environment the geometry of each obstacle and its position in the workspace are given to the planner. In the vast majority of the cases, this requires trained personnel with CAD models of the work cell and obstacles and positioning the obstacles at their exact coordinates.

For an non-engineered environment, however, this information is often difficult to determine. For this purpose, it is convenient for an operator to “paint” the workspace objects by using images obtained of a camera. This enables to approximate the position and geometry of an arbitrary object in the workspace by painting the obstacle using an image editing application. Alternatively, the identification of object pixels could be done by a scene analysis algorithm separate from the planning system. However this would not preclude the need of the planning algorithm to avoid occluding regions in the camera field of view.

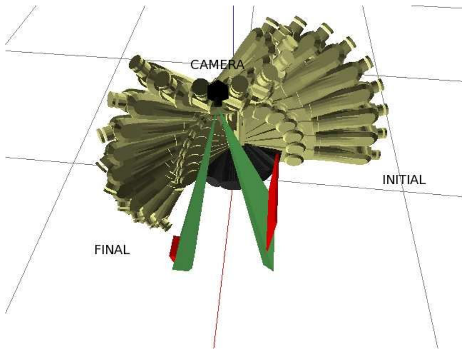

Fig. 7. Obstacles define by marking the pixels in the image provided by the camera. The covering frustums are used as obstacles.

To set up this approach, several important points must be clarified. First, the obstacles that are defined from this interaction will be interpreted as a set of frustums emerging from the camera. It is also important to mention that the shape of each frustum depends on the coordinates of the pixel and that the orientation and position of the camera determines on how the obstacles will be covered by the frustums. Thus, not every obstacle can be defined precisely with this method although it is possible to bound an obstacle for practical purposes. Second, selecting entire frustums (e.g., as those illustrated in Fig. 2) is somewhat constraining because the obstacle is defined from the camera to infinity and thus, the robot cannot move between the camera and the obstacle as it would with normal obstacles.

In this experiment, two obstacles were added to the work cell (see Fig. 7). In Fig. 7, a panel was added on the right and a cube was added on the left. The camera was placed above the workcell and pointed down. From the image of that camera, the user "painted" the pixels that belong to both obstacles. This generated a set of frustums as illustrated in Fig. 7. Then a PRM was generated with 100 nodes. Then, the edges were analysed by the DOC algorithm to detect which pixels were occluded during each motion. Finally, the planner was asked to plan the motion from a start point behind the wall to a position next to the cube. The result of this query is illustrated by the animation of Fig. 7.

It is important to note that the frustums can only cover the visible part of the obstacles and if an obstacle is not entirely visible it cannot be covered adequately. Thus, as illustrated in the Fig. 7, the frustums are not able to fully cover both obstacles since part of the obstacles are outside the field of view of the camera. Although the path illustrated in this paper did not collide with either uncovered part, this leaves the uncovered portions vulnerable to collisions.

V. Conclusion

This paper presented the DOC algorithm to detect occlusions caused by a moving manipulator and how to use the algorithm to plan collision-free and occlusion-free paths. The algorithm is based on modeling the camera as a 3D frustum that can collide with a manipulator and the collisions are interpreted as occlusions. Base on this concept, the DOC algorithm is derived from an efficient adaptive dynamic collision checking algorithm. Results show that DOC is particularly efficient when few occlusions are involved in a motion. This is used to screen the edges of probabilistic roadmaps to ensure that a minimum number of pixels are not occluded by each edge. At query time, the planner finds a path that avoids occluding specific subsets of pixels. The system was also demonstrated in an interactive mode by using the images from the camera to paint obstacles that must be avoided by the robot. This avoids the need for drawing and configuring the obstacles with computer software package for testing, training and setup purposes.

VI. Acknowledgments

This research was funded by the Natural Sciences and Engineering Research Council of Canada and Precarn. The authors would like to acknowledge our industrial collaborators at Braintech Inc.

References

[1] F. Chaumette and S. Hutchinson, "Visual servo control part i: Basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, December 2006.

[2] F. Schwarzer, M. Saha, and J.-C. Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 338–353, June 2005.

[3] U. Assarsson and T. Möller, "Optimized view frustum culling algorithms for bounding boxes," *Journal of Graphics Tools*, vol. 5, no. 1, pp. 9–22, September 2000.

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

[5] F. Chaumette and S. Hutchinson, "Visual servo control part ii: Advanced approaches," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, March 2007.

[6] A. Chan, E. A. Croft, and J. J. Little, "Trajectory specification via sparse waypoints for eye-in-hand robots requiring continuous target visibility," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Nice, September 2008, pp. 2151–2156.

[7] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 534–549, August 2002.

[8] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

[9] M. A. Baumann, D. C. Dupuis, S. Léonard, E. A. Croft, and J. J. Little, "Occlusion-free path planning with a probabilistic roadmap," in *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, September 2008, pp. 2151–2156.

[10] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry Algorithms and Applications*. Springer, 1997.

[11] G. Sanchez and J.-C. Latombe, "On delaying collision checking in prm planning: Application to multi-robot coordination," *The International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[12] S. Leonard, E. A. Croft, and J. J. Little, "Dynamic visibility checking for vision-based motion planning," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Los Angeles, California, May 2008, pp. 2283–2288.

[13] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[14] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000, pp. 3719–3726.

[15] C. Gomes, B. Selman, and H. Kautz, "Boosting combinatorial search through randomization," in *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98).*