

Dynamic Visibility Checking for Vision-Based Motion Planning

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

Abstract—An important problem in position-based visual servoing (PBVS) is to guarantee that a target will remain within the field of view for the duration of the task. In this paper, we propose a dynamic visibility checking algorithm that, given a parametrized trajectory of the camera, determines if an arbitrary 3D target will remain within the field of view. We formulate the problem of determining if an image target leaves the field of view during a trajectory as a problem of determining if the 3D coordinates of the target collides with the camera frustum during the trajectory. To solve this problem, our algorithm computes the shortest distance between the target and the frustum as well as the length of the trajectory described by the target in the camera’s coordinate frame. Furthermore, we demonstrate that our algorithm can be combined with path planning algorithms and, in particular, probabilistic roadmaps (PRM). Results suggest that our algorithm is computationally efficient even when the target moves in the vicinity of image borders. In simulations, we use our dynamic visibility checking algorithm in conjunction with a PRM to plan collision free paths while providing the guarantee that a specific target will not leave the field of view.

I. INTRODUCTION

Whether it is in the structured environments of assembly lines or in the unstructured environments of households, robots have consistently improved their performances over the last few decades. As the level of autonomy of robots increases, so is the reliance on sensors to provide feedback to robot controllers. Among sensing devices, cameras are one of the most popular in the robotics community. In particular, motion control based on visual feedback, also known as visual servoing [1], [2], has been consistently at the forefront of robotics research. The bulk of the research in visual servoing has focused on a specific architecture known as *image-based visual servoing* (IBVS). Despite the advantages of IBVS, its velocity control aspect is not suitable for the majority of industrial robots. Typically, industrial robots operate through proprietary interfaces that only allow position commands in joint space or Cartesian space. A more suitable visual servoing architecture for such robots is *position-based visual servoing* (PBVS). In PBVS, a command is defined by the Cartesian parameters of a desired position and visual feedback is used to assess the error between the current parameters and the desired ones.

In general, if visual feedback is used to control the motion of a robot, it is necessary to keep specific targets, markers

or features within the field of view. Whereas this issue is implicitly addressed by IBVS, it is not the case for PBVS. In fact, one of the most cited drawbacks of PBVS is the inability to guarantee that a target or scene will remain within the field of view [3]. This deficiency is often sufficient to cause the failure of a task, especially if vision tracking is required. For example, in [4], a Kalman filter is used to track the pose of a target with respect to the coordinate frame of the camera.

The algorithm presented in this paper addresses the aforementioned visibility problem of PBVS. Given a camera mounted on the end-effector of a manipulator with n links and given a joint space trajectory $[\mathbf{q}_1, \dots, \mathbf{q}_N]$, where $\mathbf{q}_i \in \mathbb{R}^n$, our algorithm asserts if a specified 3D point will remain within the field of view of the camera throughout the trajectory. Because our algorithm is based on a dynamic collision checking algorithm [5], it is named *dynamic visibility checking*. In order to demonstrate the attributes of dynamic visibility checking, we insert its functionality into a motion planner. Specifically, we use dynamic visibility checking in conjunction with a probabilistic roadmap (PRM). The result is a PBVS system that is able to keep a target within its field of view, hence to assess visual errors, while avoiding collisions with obstacles in the environment.

II. BACKGROUND

Keeping a target within the field of view of a camera is a fundamental requirement in visual servoing. If a target is localized from object recognition, it may be allowed to leave the field of view temporarily, but it must reenter in time to provide an observation at the next control iteration. The vast majority of visual servoing methods, however, requires that the target be present in the image at all time. This requirement stems from the high control rates and underlying visual tracking processes.

Generally, it is accepted that IBVS is a better approach to prevent targets from leaving the field of view [3]. The explanation for this is because a target is controlled directly in the image space. To some extent, however, IBVS always involves velocity control of the manipulator [6]. Typically, this functionality is not available for commercial industrial robots. Contrary to IBVS, PBVS uses position commands, which are more common with industrial manipulators. Because PBVS controls robots in a 3D Cartesian space, the image coordinates of a target are not directly controlled and, thus, can leave the field of view.

Several attempts were made to address this problem. In [7], a switching control approach is proposed. The controller switches between a PBVS mode and a backward motion. PBVS is used when the target is within a region of interest

S. Léonard is a visiting scholar in 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

in the image. When a part of the target leaves this region, a backward motion is used to move the target back into the region. Other switching methods between IBVS and PBVS include [8], [9]. Contrary to switching control, hybrid controllers combine both IBVS and PBVS elements simultaneously [10], [11]. An increasingly popular research area is online trajectory generation [12], [13], [14]. Trajectories are generated according to some guidelines such as joint limit avoidance, field of view and singularities.

A common theme to all these methods is that they all require a form of online control. Such visual servoing methods are designed for manipulators equipped with high bandwidth communication that enables advanced control algorithms and tight control loops [15].

The reality with robots in manufactures is otherwise. Communication is slow and only way-points are specified. Typically, Cartesian trajectories are limited to linear or circular interpolation and to tracking conveyor belts. With this class of manipulators in mind, our algorithm for dynamic visibility checking enables a robot to perform PBVS tasks while avoiding loosing track of a target. Furthermore, when coupled with collision avoidance, our solution increases the complexity of tasks that manipulators can perform on assembly lines.

A. Dynamic Collision Checking

The dynamic visibility checking algorithm presented in this paper is based on the dynamic collision checking algorithm proposed by Schwarzer *et al.* [5]. Given the initial and the final configurations of a manipulator, \mathbf{q}_1 and \mathbf{q}_N , the algorithm determines if the trajectory $[\mathbf{q}_1, \dots, \mathbf{q}_N]$ interferes with any obstacle in the environment. The main result establishes the relationship between the shortest distance between a link and an obstacle and the longest distance traveled by a point on the link. Specifically, the initial and final shortest distances between link i and an obstacle j are denoted by $d_{ij}(\mathbf{q}_1)$ and $d_{ij}(\mathbf{q}_N)$ respectively. Also, given a trajectory $[\mathbf{q}_1, \dots, \mathbf{q}_N]$, the longest distance traveled by a point on the link i is denoted by $\ell_i(\mathbf{q}_1, \mathbf{q}_N)$. In [5], the authors show that, given a trajectory $[\mathbf{q}_1, \dots, \mathbf{q}_N]$, link i does not collide with obstacle j if

$$\ell_i(\mathbf{q}_1, \mathbf{q}_N) < d_{ij}(\mathbf{q}_1) + d_{ij}(\mathbf{q}_N) \quad (1)$$

Equation (1) is a sufficient condition, but it is not a necessary condition. Hence, if Equation (1) is not satisfied, the algorithm proceeds by dividing the trajectory $[\mathbf{q}_1, \dots, \mathbf{q}_N]$ in two trajectories $[\mathbf{q}_1, \dots, \mathbf{q}_M]$ and $[\mathbf{q}_M, \dots, \mathbf{q}_N]$ and Equation (1) evaluates the condition for each of them. This procedure is applied recursively to all trajectories until all trajectories satisfy the condition (1) or a collision is detected.

III. DYNAMIC VISIBILITY CHECKING

Given a 3D point ${}^C\mathbf{P}$ and a trajectory $[\mathbf{q}_1, \dots, \mathbf{q}_N]$, we formulate the problem of dynamic visibility checking as a dynamic collision checking between ${}^C\mathbf{P}$ and the frustum of the camera.

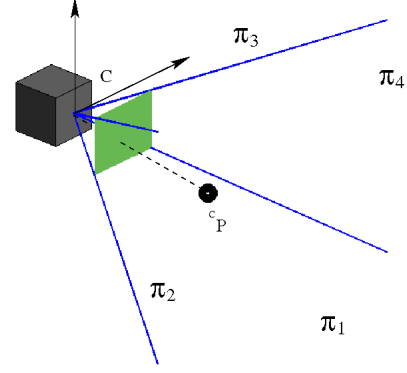


Fig. 1. Frustum of a camera and its four associated planes.

We begin by outlining the pinhole model that is used to represent the camera. Let ${}^C\mathbf{P} = [{}^CX \ {}^CY \ {}^CZ]^T$ denote the coordinates of a three dimensional point in the camera coordinate frame C (Fig. 1). The image coordinates (x_i, y_i) , in pixels, of ${}^C\mathbf{P}$ are given by [16]

$$x_i = -\frac{f}{s_x} \frac{{}^CX}{{}^CZ} + o_x \quad (2)$$

$$y_i = -\frac{f}{s_y} \frac{{}^CY}{{}^CZ} + o_y \quad (3)$$

where f is the focal length of the camera, (s_x, s_y) is the width and height of each pixel in millimeters and (o_x, o_y) are the coordinates of the image center in pixels.

The inward normals of the four planes π_1, π_2, π_3 and π_4 that define the camera frustum are given by the following cross products

$$\begin{aligned} \mathbf{n}_{\pi_1} &= [{}^CX_{max} \ {}^CY_{min} \ 1] \times [{}^CX_{max} \ {}^CY_{max} \ 1] \\ \mathbf{n}_{\pi_2} &= [{}^CX_{max} \ {}^CY_{max} \ 1] \times [{}^CX_{min} \ {}^CY_{max} \ 1] \\ \mathbf{n}_{\pi_3} &= [{}^CX_{min} \ {}^CY_{max} \ 1] \times [{}^CX_{min} \ {}^CY_{min} \ 1] \\ \mathbf{n}_{\pi_4} &= [{}^CX_{min} \ {}^CY_{min} \ 1] \times [{}^CX_{max} \ {}^CY_{min} \ 1], \end{aligned}$$

where ${}^CX_{min}$, ${}^CX_{max}$, ${}^CY_{min}$ and ${}^CY_{max}$ are given by

$$\begin{aligned} {}^CX_{min} &= -\frac{s_x}{f}(x_{i_{min}} - o_x); & {}^CX_{max} &= -\frac{s_x}{f}(x_{i_{max}} - o_x); \\ {}^CY_{min} &= -\frac{s_y}{f}(y_{i_{min}} - o_y); & {}^CY_{max} &= -\frac{s_y}{f}(y_{i_{max}} - o_y); \end{aligned}$$

and $x_{i_{min}}$, $x_{i_{max}}$, $y_{i_{min}}$ and $y_{i_{max}}$ are the smallest and largest image coordinates in pixels.

For ${}^C\mathbf{P}$ to remain within the field of view during a trajectory of the camera, ${}^C\mathbf{P}$ must not collide with π_1, π_2, π_3 or π_4 . Using this formulation, we adapt the dynamic collision checking algorithm of Section II-A to solve the dynamic visibility checking problem. In order to evaluate the condition of Equation (1) for dynamic visibility check, the following sections outline the computation of the shortest distance to collision and the length of trajectory.

A. Shortest Distance to Collision

To test the condition of Equation (1), the shortest distances between ${}^C\mathbf{P}$ and the planes $\pi_{1,\dots,4}$ are computed. First, the all the distances are evaluated by

$$d_{C\mathbf{P},\pi_k} = \frac{\mathbf{n}_{\pi_k}}{\|\mathbf{n}_{\pi_k}\|} \cdot {}^C\mathbf{P} \quad 1 \leq k \leq 4$$

and the shortest distance is found with

$$d_{C\mathbf{P}} = \min(d_{C\mathbf{P},\pi_k}) \quad 1 \leq k \leq 4.$$

We note that if ${}^C\mathbf{P}$ is outside the field of view of the camera, then $d_{C\mathbf{P}} \leq 0$.

B. Length of Trajectory

Without loss of generality, we assume that the coordinate frame of the camera coincides with the coordinate frame of the end-effector. The task is to bound the length of the trajectory described by ${}^C\mathbf{P}$ as the camera follows the trajectory $[\mathbf{q}_1, \dots, \mathbf{q}_N]$.

In order to compute $\ell_{C\mathbf{P}}(\mathbf{q}_1, \mathbf{q}_N)$, we invert the upper bound computation presented in [5]. This approximation proceeds by computing an upper bound to the contribution of each joint to the length of the trajectory described by the end-effector. First, for each joint, the configuration that maximizes the spatial velocity of the end-effector is found. Then, each configuration is used to bound the contribution of its corresponding axis by rotating its joint by the total amount specified by the trajectory. Finally, the upper bound of the trajectory corresponds to the sum of all the contributions.

For dynamic visibility checking, we compute the inverse, in the kinematic sense, of the aforementioned bound. That is, we find the configurations that maximizes the body velocity of ${}^C\mathbf{P}$. For a manipulator with revolute joints, these configurations are found by maximizing the radius between pairs of adjacent links. First, we compute the distance r_1 between ${}^C\mathbf{P}$ and the first axis. This radius is then multiplied by the amount of rotation θ_1 performed by the first joint during the trajectory. That is, the contribution of the first axis to the length of the trajectory of ${}^C\mathbf{P}$ is at most $r_1\theta_1$. Then, we estimate the bound attributed to the second axis. The configuration that maximizes the body velocity of ${}^C\mathbf{P}$ is when the length of the second link extends the radius r_1 . Again, this compound radius r_2 is multiplied by the amount of rotation θ_2 performed by the second joint. This procedure is performed for each link, starting from the base toward the end-effector. It follows that the length of the trajectory described by ${}^C\mathbf{P}$ has the upper bound

$$\ell_{C\mathbf{P}}(\mathbf{q}_1, \mathbf{q}_N) \leq \sum_{i=1}^n r_i \theta_i. \quad (4)$$

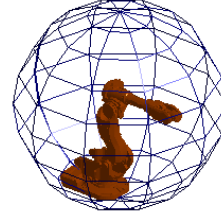
In the special case where the joint trajectories are interpolated linearly, Equation (4) becomes

$$\ell_{C\mathbf{P}}(\mathbf{q}_1, \mathbf{q}_N) \leq \sum_{i=1}^n r_i |q_{iN} - q_{i1}|.$$

The computation of the upper bound is illustrated in Fig. 2 for links 2 and 3. In these images, ${}^C\mathbf{P}$ is set to the



(a) Rotation of the base around the second axis.



(b) Rotation of the compounded radius around the third axis.

Fig. 2. Compounding radii between adjacent links.

coordinates of the base of the manipulator. Figures 2(a) and 2(b) illustrate the non-decreasing radii as they compound between adjacent links.

We note that $\ell_{C\mathbf{P}}(\mathbf{q}_1, \mathbf{q}_N)$ depends on the coordinates of ${}^C\mathbf{P}$, as its distance from the first axis is compounded to the distances between adjacent axes. Therefore, it is impossible to precompute all the r_i offline. In the case of a manipulator with revolute joints, however, the distance between adjacent axes remains constant and $\ell_{C\mathbf{P}}(\mathbf{q}_1, \mathbf{q}_N)$ can be computed efficiently. In contrast, the upper bound computed in [5] always involved the tool control point and, as such, it is possible to precompute all the r_i offline.

IV. PROBABILISTIC ROADMAP

Dynamic visibility checking determines if a 3D point remains within the field of view during a given trajectory in joint space. As mentioned in Section I, this problem is considered to be essential in several PBVS applications. In general, however, a manipulator must comply to additional constraints when performing a task. Typically, constraints such as collision avoidance require advanced path planning algorithms. Over the last decade, probabilistic roadmaps has emerged as one of the popular paradigm for path planning [17], [18]. The intuition behind PRM consists of generating several samples of a robot's configuration. These samples constitute the vertices of a graph and two vertices v_i and v_j are connected by an edge e_{iv} if the trajectory between v_i and v_j satisfies a set of conditions. In the case of collision avoidance, an edge e_{ij} is inserted in the graph if the trajectory between v_i and v_j does not collide with an obstacle. Given

TABLE I
INTRINSIC PARAMETERS OF THE CAMERA.

f	(o_x, o_y)	(s_x, s_y)
5mm	(0, 0)	(100, 100)

a start vertex v_s and a finish vertex v_f , the graph is searched for a path between v_s and v_f [19]. Each intermediate vertex in the path becomes a way point for the manipulator.

As a last contribution, we insert the dynamic visibility checking algorithm in a PRM path planner. In particular, we demonstrate that the dynamic visibility checking augments the complexity of tasks that can be performed. Our planner enables a manipulator to reach a destination v_f while avoiding collision and keeping an arbitrary target ${}^C\mathbf{P}$ within the field of view.

Our implementation is based on the k-shortest path algorithm presented in [20]. First, we build a graph in which an edge e_{ij} represents a collision free trajectory between v_i and v_j . Each edge is weighted according to the norm of the variation between its vertices, i.e. $w = \|v_i - v_j\|$. Given a start node v_s and a finish node v_f , we used Dijkstra's algorithm to find the shortest path between v_s and v_f . For each edge e_{ij} along path, dynamic visibility checking is used to determine if ${}^C\mathbf{P}$ remains within the field of view. If the target remains within the field of view for all trajectories, then no more processing is necessary. If, however, the trajectory represented by the edge e_{ij} does not keep ${}^C\mathbf{P}$ within the field of view, e_{ij} is effectively removed from the graph by giving it an infinite weight. Dijkstra's algorithm is used once again to find a shortest path from v_i to v_f and the new path is appended to the path from v_s to v_i . This procedure is repeated each time an edge on the path fails the dynamic visibility check.

V. RESULTS

All the experiments are based on simulating the kinematics of an ABB IRB6600 manipulator with a pinhole camera mounted on the end-effector. The image size are 640×480 and the intrinsic parameters of the cameras are summarized in Table V. In all the experiments, the joint space trajectories are linearly interpolated between \mathbf{q}_1 and \mathbf{q}_N . In the first set of experiments, we evaluate the performance of the dynamic visibility checking. In the second set, we demonstrate the combination of the dynamic visibility checking with a PRM to plan collision free and visible paths.

A. Dynamic Visibility Checking

We tested our dynamic visibility checking algorithm on three categories of experiments

- Experiment 1) The target remains within the field of view
- Experiment 2) The target almost leaves the field of view
- Experiment 3) The target leaves and reenter the field of view.

In the first experiment, the initial and final joint angles were set to the values presented in Table II and the

TABLE II
EXPERIMENT 1) INITIAL AND FINAL JOINT ANGLES.

	q_1	q_2	q_3	q_4	q_5	q_6
q_I	-0.3927	-1.0472	0.7854	-2.3562	-1.0472	0
q_F	0.3927	1.0472	-0.7854	-1.5708	1.0472	0.3927

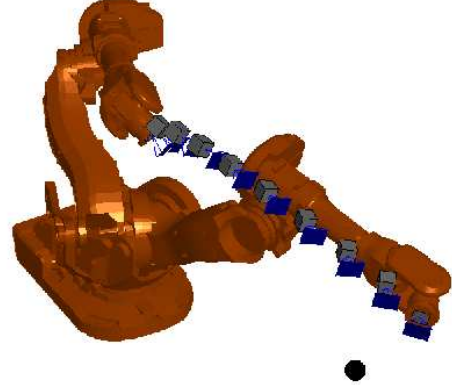


Fig. 3. Experiment 1) Trajectory of the camera

coordinates of the target in the base coordinate frame of the robots were ${}^B\mathbf{P} = [3000 \ 0 \ 1500]^T$. The dynamic visibility checking required 353 evaluations of Equation (1) and correctly determined that ${}^B\mathbf{P}$ did not leave the field of view. The trajectory of the camera and the trajectory of ${}^B\mathbf{P}$ in the image are illustrated in Fig. 3 and Fig. 4 respectively. This performance is partially attributed to the relatively large distance between the frustum and the ${}^B\mathbf{P}$ throughout the trajectory.

The second experiment involves a target moving close to a border of the image. The initial and final joints angles for this experiments are reported in Table III. The coordinates of ${}^B\mathbf{P}$ are $[3000 \ 0 \ 1200]^T$. The trajectory of the camera is shown in Fig. 5 and the resulting trajectory described by

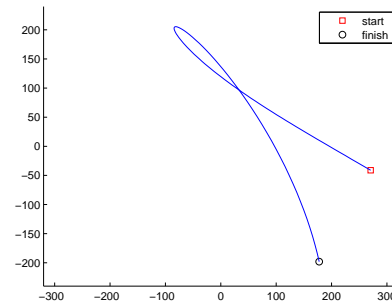


Fig. 4. Experiment 1) Trajectory of ${}^C\mathbf{P}$ in the image

TABLE III
EXPERIMENT 2,3) INITIAL AND FINAL JOINT ANGLES.

	q_1	q_2	q_3	q_4	q_5	q_6
q_I	0	0	0	0	0	0
q_F	0.3927	1.0472	-0.7854	-1.5708	1.0472	0.392

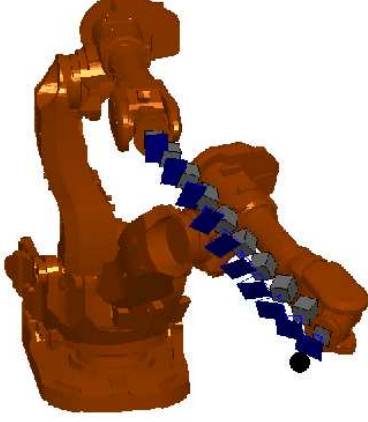


Fig. 5. Experiment 2 and 3) Trajectory of the camera.

the target in the image is shown in Fig. 6. The algorithm evaluated Equation (1) 3583 times and correctly determined that the target did not leave the field of view. This high number of evaluations is explained by the target moving close to the border of the image. Indeed, in our simulation, the smallest horizontal image coordinate recorded was -319.5 , while the left boundary was set to -320 . In 3D, ${}^B\mathbf{P}$ came within 0.5mm of colliding with the frustum of the camera. By comparison, the IRB6600 has a reach of 2.8m when fully extended. In the light of the previous experiment, the variation of the number of evaluations of Equation (1) demonstrates the adaptive nature of the algorithm as several small sub-trajectories are tested when the distance to collision is small.

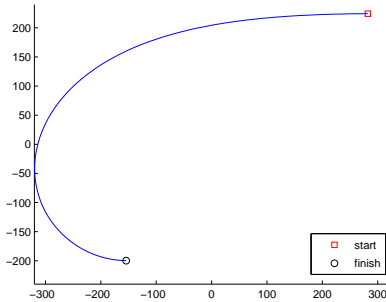


Fig. 6. Experiment 2) Trajectory of ${}^B\mathbf{P}$ in the image.

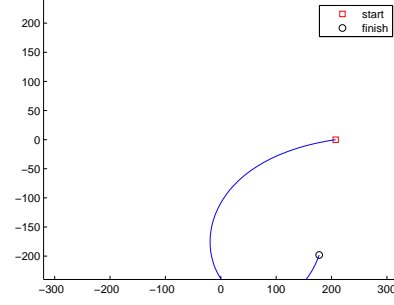


Fig. 7. Experiment 3: Trajectory of ${}^B\mathbf{P}$ in the image.

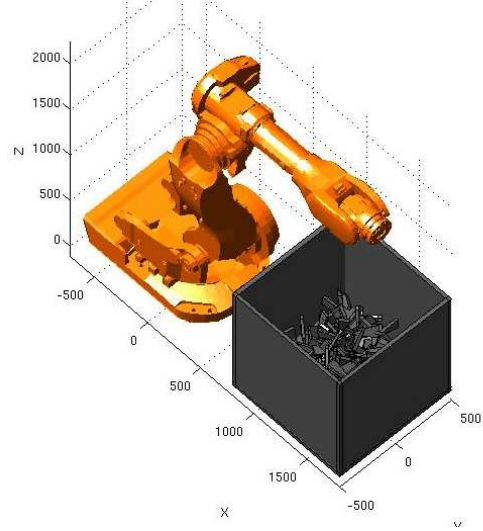


Fig. 8. IRB6600 and a bin used for bin-picking simulation.

The last experiment demonstrates a situation in which the target reenters the field of view after leaving it. The initial and final joint angles are the same as those used in Experiment 2 (Table III). The coordinates of ${}^B\mathbf{P}$, however, are changed to ${}^B\mathbf{P} = [3000 \ 0 \ 1500]$. In this experiments, our algorithm correctly determined that ${}^B\mathbf{P}$ would leave the field of view with only 86 evaluations of Equation (1).

B. Path Planning

In this experiment, we combine the dynamic collision checking with a PRM for the task of bin picking (Fig 8). The graph that represents the PRM contains 1000 vertices and each vertex is connected to at most 100 of its closest neighbors. All the samples were obtained in order to position the end-effector within a reasonable volume around the bin. An edge e_{ij} is inserted in the graph if the trajectory between v_i and v_j does not result in a collision between a link and the bin.

To simulate a vision guided pick, we generate a position command for a 10cm long tool mounted along the Z axis of the end-effector. The command in Cartesian space is converted to a command v_f in joint space and the shortest path between the start v_s and v_f is found by using Dijkstra's algorithm. Then, we used the algorithm presented in Section

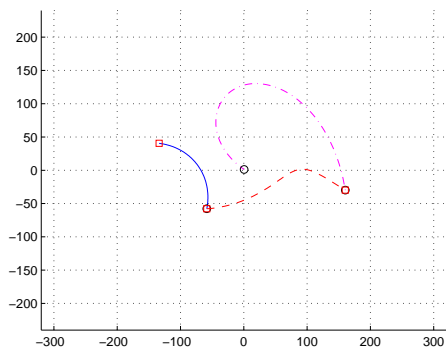


Fig. 9. Result of using three trajectories to reach the goal.

IV to ensure that the target ${}^B\mathbf{P}$, defined by the desired (X, Y, Z) coordinates of the tool, remains within the field of view of the camera. Fig. 9 illustrates the image trajectory of the target resulting from a path composed of 3 trajectories. Because the coordinate frame of the tool is 10cm along the optical axis of the camera, the final target position is projected at the center of the image. During the experiment, Dijkstra's algorithm was used 16 times because of edges that failed to ensure visibility of the target.

VI. CONCLUSION

Keeping a target within the field of view is a fundamental problem of visual servoing and it is especially acute for PBVS. This paper presents a novel approach to the problem of keeping a target within the field of view of a camera. We formulate the dynamic visibility checking as a dynamic collision checking. Given a trajectory between two joint space configurations, the dynamic visibility checking algorithm is able to determine if a 3D target will remain within the field of view. We use our algorithm in conjunction with a PRM planner to find paths that avoid collisions with obstacles and maintain a target in the field of view. Our approach is suitable for PBVS tasks with manipulators that only provide Cartesian space or joint space position commands. In the future, we hope to extend this research to include dynamic sampling in areas with high visibility.

VII. ACKNOWLEDGMENTS

This work was supported in part NSERC, Precarn Inc., and Braintech Inc., who are gratefully acknowledged for their financial support. The authors would also like to acknowledge Pantelis Elinas for his thoughts on this research.

REFERENCES

- [1] K. Ashimoto, "A review on vision-based control of robot manipulators," *Advanced Robotics*, vol. 17, no. 10, pp. 969–991, 2003.
- [2] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, October 1996.
- [3] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*, ser. Lecture Notes in Control and Information Systems, D. Kriegman, G. Hager, and A. Morse, Eds. Springer-Verlag, 1998, vol. 237, pp. 66–78.

- [4] W. J. Wilson, C. C. Williams Hulls, and G. S. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, October 1996.
- [5] 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.
- [6] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, June 1992.
- [7] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino, "Keeping features in the field of view in eye-in-hand visual servoing: A switching approach," *IEEE Transactions on Robotics*, vol. 20, no. 5, pp. 908–913, October 2004.
- [8] K. Hashimoto and T. Noritsugu, "Potential switching control in visual servo," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, April 2000, pp. 2765–2770.
- [9] N. Mansard and F. Chaumette, "A new redundancy formalism for avoidance in visual servoing," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Canada, August 2005, pp. 468–474.
- [10] P. I. Corke and S. A. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 507–515, August 2001.
- [11] E. Malis, F. Chaumette, and S. Boudet, "2-1/2-d visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, April 1999.
- [12] 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.
- [13] F. Schramm, F. Geffard, G. Morel, and A. Micaelli, "Calibration free image point path planning simultaneously ensuring visibility and controlling camera path," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 2074–2079.
- [14] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice, "Position based visual servoing : keeping the object in the field of vision," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC, May 2002, pp. 1624–1629.
- [15] P. I. Corke and M. C. Good, "Dynamic effects in visual closed-loop systems," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 671–683, October 1996.
- [16] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
- [17] 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.
- [18] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundation of probabilistic roadmap planning," *International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, July 2006.
- [19] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004, pp. 446–452.
- [20] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, pp. 712–716, 1971.