

An Occlusion Prediction Algorithm for Visual Servoing Tasks in a Multi-Arm Robotic Cell

Vincenzo Lippiello, Bruno Siciliano, Luigi Villani
PRISMA Lab, Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli Federico II
Via Claudio 21, 80125 Napoli, Italy
{lippiell, siciliano, lvillani}@unina.it

Abstract—This paper presents a new approach to the prediction of visual occlusions for a robotic system performing visual servoing tasks. The case of a multi-arm robotic cell equipped with a multi-camera eye-to-hand visual system is considered. The proposed approach is able to predict the self occlusions as well as the mutual occlusions caused by the robot links and tools. The algorithm is based on the geometric representation of the cell in terms of a BSP tree, assuming that the CAD geometry of the objects and the kinematic model of the multi-arm robotic system are known. Experimental results are presented for the case of two industrial robots performing a vision-guided grasping task.

Index Terms—Visual servoing, multi-arm robots, visual tracking, real-time operating system, Kalman filter

I. INTRODUCTION

The adoption of multi-camera visual systems to guide the execution of manipulating tasks in a robotic multi-arm cell is becoming a feasible scenario in industrial applications. A common problem of these applications is the occurrence of occlusion of the workpieces with respect to visual system, especially when eye-to-end configurations, i.e., fixed cameras, are used. Typical examples are:

- self-occlusion, in the sense that some parts of a workpiece are hidden with respect to the cameras by other parts of the object itself;
- a workpiece is hidden with respect to a camera by a robot link or by another object;
- during a manipulation task, like grasping, some parts of the workpiece may be occluded by the robot hand or gripper;

where the last two cases are examples of mutual occlusion.

The presence of occlusions may cause serious problems to any kind of algorithm based on the extraction of image features. Hence, it is important to develop suitable strategies that allow predicting the occurrence of occlusions.

The problem of computing occlusion-free viewpoints in a known polyhedral world is considered in [1], [2]. However, the proposed techniques do not consider real-time constraints and moving objects. A simple occlusion resistant object tracking algorithm is proposed in [3] where the Kalman filter is adopted to have a prediction of the target object trajectory while an occlusion occurs. In [4] an algorithm for automatic grasping is proposed, based on the selection of viewpoints that avoid occlusion for the evaluation of the grasping trajectory. However, this algorithm

does not solve the problem of occlusions in a multi-arm robotic cell in the presence of moving workpieces.

In this paper, an algorithm for a multi-arm robotic system is proposed that recognizes in real-time the presence of occlusions with respect to each camera of an eye-to-hand visual system. In this way, the occluded features can be eliminated from the set of features to be extracted by the images.

The algorithm is based on Binary Space Partitioning (BSP) tree structures to represent the 3D geometry of the cell. The BSP tree representation is updated in real time on the basis of the measurements of the joint positions and of the estimated poses of the workpieces provided by a visual tracking algorithm presented in previous works [5], [6]). The occlusion detection is realized through a suitable visit algorithm of the tree.

The paper is organized as follows. In Section II some details about BSP tree representations are provided. The occlusion prediction algorithm is presented in Section III, while in Section IV the visual tracking algorithm based on the extended Kalman filter and optimal features selection is briefly outlined. Experimental results for the case of two industrial robots performing a vision guided grasping task are presented in Section V.

II. BSP TREE GEOMETRIC REPRESENTATION

A BSP tree data structure may be employed to represent the geometry of known 3D environments, like a robotic cell with one or more robot manipulators and workpieces. To achieve a computationally fast representation, the attention is limited to polyhedral objects, characterized by planar polygonal surfaces. This choice is not too restrictive, since a large class of man-made objects of different shapes can be approximated as polyhedral solids.

The elementary data of a BSP tree representation are the object surfaces. Each surface can be seen as an anticlockwise ordered sequence of feature points (the corners of the polygon) laying on its contour. For example, for the object shown in Fig. 1, the representation of the surface S_1 is the sequence $S_1 = \{f_1, f_4, f_6, f_7, f_9, f_{12}, f_{16}, f_{13}\}$.

To represent the set of all the surfaces of an object, the following BSP tree building paradigm is adopted: each node of the tree is characterized by a partition plane that divides the 3D space into two subspaces containing all the surfaces (or pieces of surfaces) which are in the front

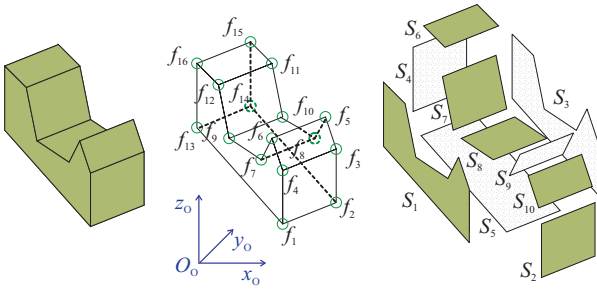


Fig. 1. An example of decomposition of a 3D object into elementary surfaces.

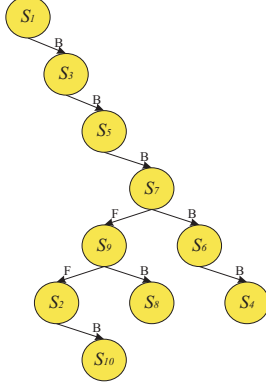


Fig. 2. A possible BSP tree representation of the object of Fig. 1; labels F and B denote the front and back subspaces.

and in the back, respectively, of the partition plane; all the surfaces laying on the partition plane are stored into the node. Applying recursively this paradigm to the two subspaces, a binary tree data structure is obtained, whose nodes contain all the surfaces of the object.

Notice that this approach does not impose connectivity constraints on the internal subspace of the represented object. Therefore, a unique BSP tree may be used to simultaneously represent many objects with respect to the same reference frame.

The conformation of the tree depends on the choice of the partition planes. For the purpose of this work, the partition planes are selected from the set of plains containing the surfaces of the objects. To reduce the complexity of the tree it is convenient to select the partition planes in a sequence that minimizes the number of intersections with the other object surfaces. As an example, in Fig. 2 is reported a BSP tree representing the object of Fig. 1, where the choice of the partition planes does not generate intersections.

III. OCCLUSION PREDICTION ALGORITHM

To the purpose of the occlusion prediction algorithm presented in this paper, the robotic cell is seen as a collection of objects which includes the workpieces (target objects), the robot links and tools, as well as all the possible obstacles that may occlude the workpieces with respect to the cameras.

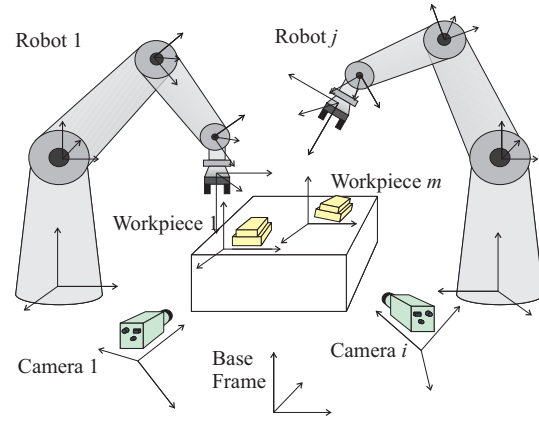


Fig. 3. Sketch of the cell with the reference frames.

At each sampling time, the algorithm provides the prediction of the positions on the camera image planes of all the visible feature points of the workpieces. The inputs of the algorithm are the robot joint measurements and the prediction of the pose of the workpieces provided by the visual tracking algorithm described in the following section.

The occlusion prediction algorithm can be decomposed in two parts: geometric modelling and occlusion detection, which are described in the following subsections.

A. Geometric modelling

This part is aimed at generating on-line a BSP tree, representing the 3D geometrical model of the cell. To this purpose, the CAD model of all the objects of the cell are assumed to be known. Moreover, each object has to be represented as a set of surfaces with respect to a reference frame fixed to the object itself. This type of representation can be generated once off-line to facilitate the on-line BSP tree construction.

Consider a system composed by w workpieces and t robots and tools, where robot i is composed by l_i links; hence the total number of objects is $n = w + t + \sum_{i=1}^t l_i$. Moreover, assume that c fixed cameras are available. The corresponding reference frames are represented in Fig. 3. In the following, the pose of a reference frame with respect to another frame will be represented in terms of homogeneous transformation matrices.

The pose of workpiece i with respect to the base frame, represented by the homogeneous transformation matrix W_i^b , is provided by the pose estimation algorithm described in the next section.

By using the Denavit-Hartenberg convention [7], a frame can be assigned to each link of the robot i . Then the homogeneous transformation matrix $L_{j,i}^i$, representing the pose of link j of the robot i with respect to robot frame i , may be evaluated as follows

$$L_{j,i}^i(q_{j,i}) = L_{1,i}^i(q_{1,i}) L_{2,i}^{1,i}(q_{2,i}) \cdots L_{j,i}^{j-1,i}(q_{j,i}),$$

where $q_{j,i} = [q_{j,1} \dots q_{j,l_i}]^T$, $j = 1, \dots, l_i$ being $q_{j,i}$ the measurement of joint variable j of robot i . The pose of

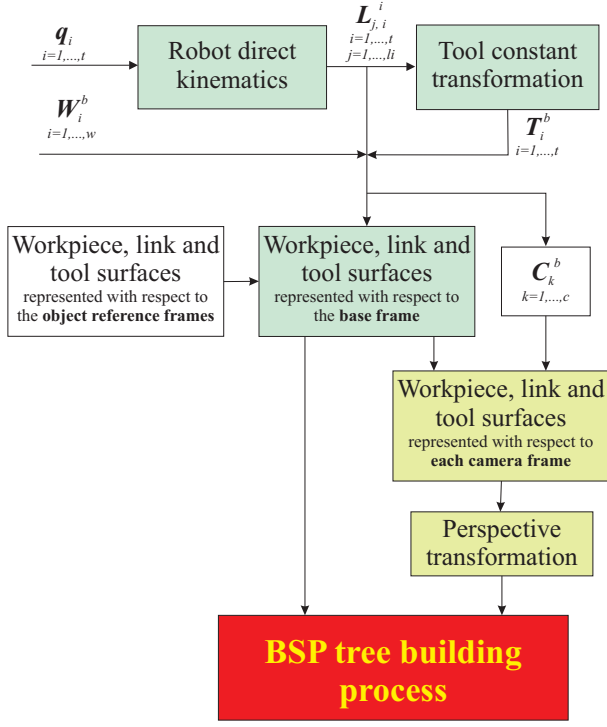


Fig. 4. Dynamic BSP tree building process.

link j of robot i can be expressed in the base frame as

$$L_{j,i}^b(q_{j,i}) = L_i^b L_{j,i}^i(q_{j,i}),$$

where L_i^b is the constant homogeneous transformation matrix representing the pose of robot frame i with respect to the base frame, that can be computed via a calibration procedure of the multi-arm robotic system.

Analogously, the homogeneous transformation matrix T_i^b representing the pose of a tool mounted on the robot i with respect to the base frame can be computed as

$$T_i^b(q_i) = L_{l_i,i}^b(q_i) T_i^i,$$

where, for notation compactness, $q_i = q_{l_i,i}$ and $T^i = T_{l_i,i}^i$ denotes the constant homogeneous transformation matrix representing the pose of the tool with respect to the last link of robot i .

The above computations are the first steps of the procedure represented in Fig. 4.

The next step is the computation of the pose of all the objects of the cell with respect to a frame fixed to camera k ($k = 1, \dots, c$), in the form

$$\begin{aligned} W_i^k &= C_k^{bT} W_i^b, \\ L_{j,i}^k(q_{j,i}) &= C_k^{bT} L_{j,i}^b(q_{j,i}), \\ T_i^k(q_i) &= C_k^{bT} T_i^b(q_i) \end{aligned}$$

where C_k^b is the homogeneous transformation matrix representing the pose of camera frame k with respect to the base frame b . This matrix, depending on the extrinsic camera parameters, is computed off-line through a suitable camera calibration procedure.

At this point, the pose of all the objects with respect to camera frame k ($k = 1, \dots, c$) can be easily evaluated. By using the camera perspective transformation, which depends on the intrinsic camera parameters, it is possible to compute the projections of all the objects of the cell (each seen as a set of surfaces) on the image plane of all the cameras. On the basis of these data, the BSP tree structure representing the current geometric configuration of the cell, as it is seen by all the cameras, can be built. Details about the construction of the BSP tree can be found in [6].

B. Occlusion detection

The detection of the occluded parts of the workpieces with respect to a given camera can be achieved by implementing a suitable recursive visit of the corresponding BSP tree representation. This algorithm allows recognizing all the feature points lying on parts of the workpieces that are not occluded with respect to the camera. It can be described by the following Pascal-like procedure:

```

procedure not_occluded (node:BSP_tree;
  view:point;visible_points:point_list);
begin
  if node NOT EMPTY then
    case classify_point(view,
      node->partition_plane)
    ON THE_PLANE:
      not_occluded(node->front_tree,
        view,visible_points);
      not_occluded(node->back_tree,
        view,visible_points);
    IN_FRONT_OF:
      not_occluded(node->back_tree,
        view,visible_points);
      process_surfaces(node->surfaces,
        view,visible_points);
      not_occluded(node->front_tree,
        view,visible_points);
    IN_BACK_OF:
      not_occluded(node->front_tree,
        view,visible_points);
      process_surfaces(node->surfaces,
        view,visible_points);
      not_occluded(node->back_tree,
        view,visible_points);
    end {case}
  end {if}
end {begin}

```

In the above procedure, the input variable `view` is the point of view (corresponding to the optical center of the image plane of the considered camera) from which the current set of visible feature points of the object is evaluated. The visible points are listed in the variable `visible_points`, which contains also the projections of these points on the image plane of the camera.

The function `classify_point()` evaluates the position of the point of view with respect to the partition plane.

The core of the occlusion prediction algorithm is the procedure `process_surfaces()`, which updates the current set of visible points by adding all the feature points of the surfaces of the current node and by eliminating all the feature points that are hidden by the surfaces of the current

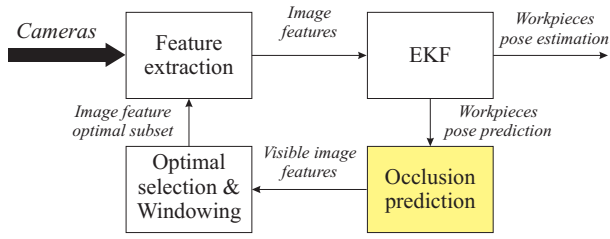


Fig. 5. Block scheme of the visual tracking algorithm.

node.

The procedure is recursive and ends when all the nodes of the tree have been visited; at the end, the current set of visible points will contain all and only the feature points visible from the point of view. Notice that construction of the set proceeds so that the surfaces are considered in a sequence corresponding to their position with respect to the point of view from the background to the foreground.

The above algorithm must be applied to all the cameras of the system.

It is important to observe that the code implementing the whole occlusion detection algorithm (visit of the tree and surfaces processing) exhibits a complexity $O(N)$, where N is the number of surfaces of the object [8]. Moreover, some modifications can be introduced both in the modelling and in the occlusion detection parts, which allow a considerable reduction of computational time [6].

IV. VISUAL TRACKING

The occlusion prediction algorithm performs a pre-selection of the feature points of the target objects to be used for the estimation of the workpiece poses. In this paper, a pose estimation algorithm based on the extended Kalman filter is adopted. A detailed description of this algorithm can be found in [5] and [6].

The extended Kalman filter provides also a prediction of the pose of the workpieces at the next sampling time, that is input to the occlusion prediction algorithm according to the closed-loop scheme represented in Fig. 5.

Notice that not all the visible feature points provided by the occlusion prediction algorithm are used for feature extraction. In fact, in a multi-camera system, the available feature points may be highly redundant and may increase the computational cost without a significant enhancement of the estimation accuracy [9]. Therefore, a suitable selection algorithm is adopted to dynamically select an optimal set of visible points. This algorithm is based on the minimization of an optimal cost function based on a combination of suitable quality indexes ensuring a balanced spatial distribution of the projections of the feature points on the image plane of each camera as well as a balanced distribution of the features among the different cameras, considering their different resolutions and focuses [6]. After the optimal selection, a windowing technique is used to compute the size and location of the windows of the image plane to be grabbed for image processing. This

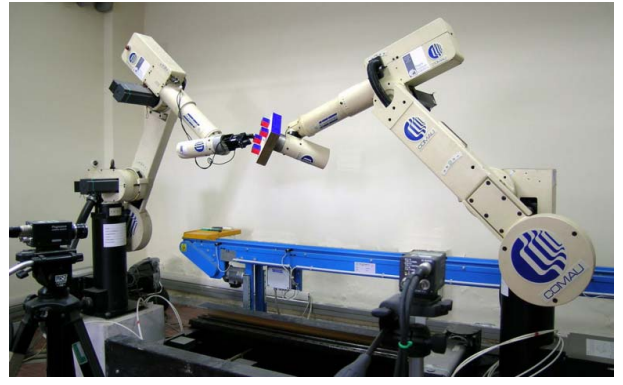


Fig. 6. Experimental setup.

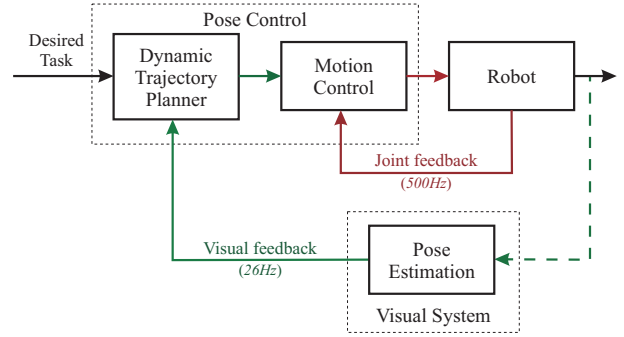


Fig. 7. Position-based control scheme.

considerably reduces the computational charge of the frame grabbing operations.

V. EXPERIMENTS

The effectiveness of the proposed approach has been tested in some experiments on the industrial robotic cell available at the PRISMA Lab.

The cell consists of two industrial robots Comau SMART-3 S (see Fig. 6). One arm (R7AX) is mounted on a sliding track which provides an additional degree of mobility with respect to the standard six degrees of mobility of the other arm (R6AX). Each robot is equipped with a pneumatic gripper with two parallel jaws. Both robots are controlled by a single PC with RTAI-Linux operating system. The experimental setup is completed with a stereo visual system composed by a PC equipped with two Matrox GENESIS boards and two Sony 8500CE B/W cameras. The Matrox boards are used as frame grabber and for a partial image processing (e.g., image windows extraction), while the PC host is in charge of executing vision-based algorithms (e.g., occlusion prediction and object motion estimation) and guarantees communication with the PC performing robot control via a standard serial connection.

The proposed algorithm has been tested on three variants of the same vision guided grasping task, involving only one robot.

The position-based visual servoing scheme represented in Fig. 7 has been adopted. The stereo visual system

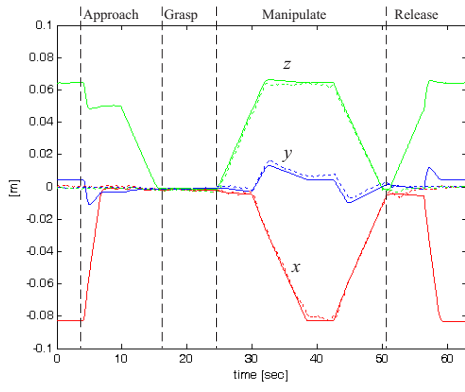


Fig. 8. Gripper (solid) and estimated object (dashed) trajectories during the first experiment.

estimates the pose of the workpiece at 26 Hz frequency, corresponding to the camera frame rate, while the pose control is performed through an inner/outer control loop. The inner loop, running at 500 Hz frequency, implements motion control. In the outer loop, the block named dynamic trajectory planner computes the trajectory for the robot tool on the basis of the current pose of the workpiece and on the desired task. The input of this block is updated at 26 Hz frequency while the output is available at 500 Hz frequency, thanks to a second-order interpolating filter.

The grasping task involves the R7AX robot and a workpiece with 16 feature points. The number of feature points used by the pose estimation algorithm has been limited to 11, selected from the set of visible points provided by the occlusion prediction algorithm. Notice that the maximum number of visible image features, for both the cameras, is 32.

The task assigned to the R7AX robot can be decomposed in the following phases:

- 1) *Approach* – When the target object is localized, starting from the HOME pose, approach the grasp pose in two steps: first go over the target object (at 5 cm height), and then descend on it;
- 2) *Grasp* – Grasp the object and check the state of the gripper;
- 3) *Manipulate* – Return to the HOME pose carrying the object;
- 4) *Release* – Go to the FINAL pose and release the object.

A. First experiment

In the first experiment, the target object and the R6AX robot are motionless. Moreover, the R6AX robot is out of the field of view of the cameras. The time history of the position trajectory of the R7AX robot and that of the estimated workpiece position are shown in Fig. 8. The time history of the orientation components is not reported here for brevity. The Approach phase begins after about 4 sec and ends in about 12 sec. During this phase the robot recognizes the workpiece and moves over it,

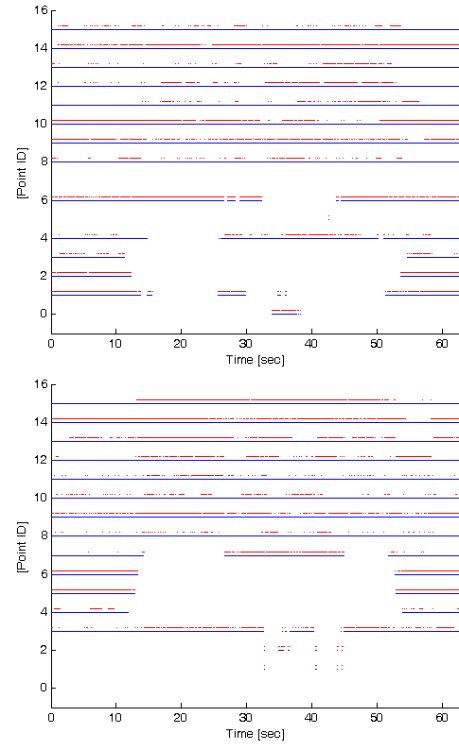


Fig. 9. Visible and selected object feature points for camera 1 (top) and camera 2 (bottom) during the first experiment.

initially keeping a distance of about 5 cm along the vertical direction (z component); then the robot begins the descent to the grasping pose. When the grasping pose has been reached, the Grasp phase begins and ends after about 8 sec. During this time the pneumatic gripper is closed and a check of the state of the jaws is performed using the magnetic sensors installed on the gripper. At about 25 sec the Manipulate phase begins and the robot return to the HOME pose carrying the workpiece. At about 50 sec the robot reaches the FINAL pose and releases the object; then, it returns to the HOME pose (Release phase).

In Fig. 9, the state of the visible and selected feature points is represented. For each point, the bottom line indicates when it is visible, the top line indicates when it is selected for feature extraction. Notice that the points that are not visible at the beginning of the task are occluded by the workpiece itself (self-occlusion) while, from the last part of the Approach phase until the first part of the Release phase, some feature points are occluded by the gripper (mutual-occlusion).

B. Second experiment

In the second experiment, the workpiece is in motion in the horizontal plane during the Approach and Grasp phases. The time history of the position trajectory of the R7AX robot and that of the estimated workpiece position are shown in Fig. 10. Differently from the previous experiment, the robot has to track the object. In fact, it can be observed that the gripper motion in the horizontal plane (x

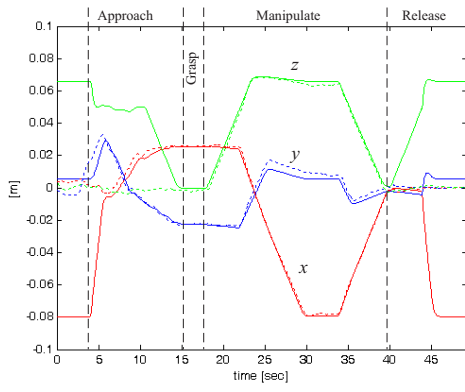


Fig. 10. Gripper (solid) and estimated object (dashed) trajectories during the second experiment.

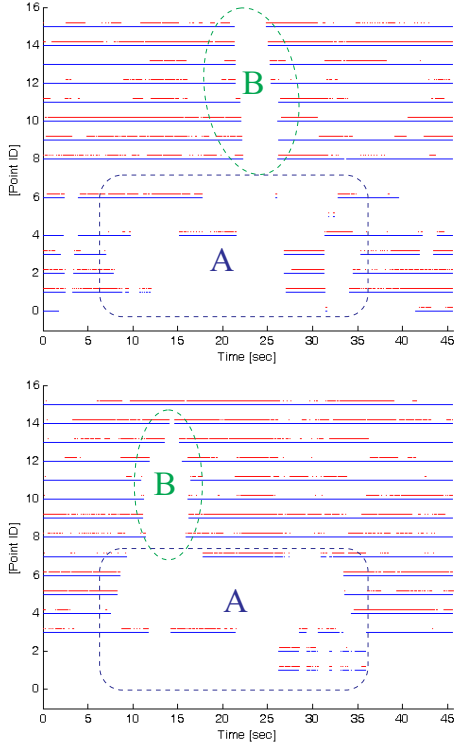


Fig. 11. Visible and selected object feature points for camera 1 (top) and camera 2 (bottom) during the third experiment.

and y components) matches the object motion during the Approach phase. Moreover, the Grasp phase is successfully executed.

C. Third experiment

In the third experiment, the workpiece is in motion as in the second experiment; moreover, the robot R6AX performs a different task and during the motion occludes the workpiece with respect to the cameras.

The state of the visible and selected feature points is reported in Fig. 11. The time histories of the position of the robot gripper and that of the estimated workpiece position

are not reported because they are practically the same to that in Fig. 10. In Fig. 11, the A-area corresponds to the occlusion caused by the gripper during the grasping, while the B-area corresponds to the occlusion caused by the robot R6AX. Notice that the motion of the robot R6AX generates a partial occlusion (only one point remains visible) on the camera 2 between the Approach and Grasp phases. This event does not affect the accuracy of the pose estimation and allows the successful execution of the Grasp phase. Moreover, during the Manipulate phase, the robot R6AX occludes completely the object with respect to camera 1. Again, from Fig. 10 it can be observed that the visual tracking algorithm maintains high accuracy, even though the estimated pose is not used after grasping (only the joint measurements are used by the dynamic trajectory planner in the Manipulate and Release phases).

VI. CONCLUSION

In this paper an algorithm for the prediction of visual occlusions in a multi-arm robotic cell performing visual servoing tasks has been presented. The algorithm is based on a computationally efficient representation of the 3D geometry of the cell, based on BSP trees. The proposed approach is able to predict the self occlusions for one or more workpieces in the cell, as well as the mutual occlusions caused by the robot links and tool. This algorithm, used in combination with a visual tracking algorithm based on the extended Kalman filter, provides a pre-selection of the objects features to be used for image window grabbing. This allows computational time savings and guarantees reliable visual measurements for pose estimation. The experimental results on a dual-arm industrial robotic cell equipped with a stereo eye-to-hand visual system have confirmed the feasibility and the effectiveness of the proposed approach.

REFERENCES

- [1] K. Tarabanis, R. Y. Tsai, and A. Kaul, "Computing occlusion-free viewpoints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 279–292, 1996.
- [2] P. C. Ho and W. Wang, "Occlusion culling using minimum occluder set and opacity map," in *Proceedings of the 1999 IEEE International Conference on Information Visualization*, July 1999, pp. 292–300.
- [3] E. Loutas, K. Diamantaras, and I. Pitas, "Occlusion resistant object tracking," in *Proceedings of the 2001 IEEE International Conference on Image Processing*, vol. 2, October 2001, pp. 65–68.
- [4] C. Laugier, A. Ijel, and J. Troccaz, "Combining vision based information and partial geometric models in automatic grasping," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, vol. 1, May 1990, pp. 676–682.
- [5] V. Lippiello, B. Siciliano, and L. Villani, "3-d objects motion estimation based on kalman filter and bsp tree models for robot stereo vision," *Archives of Control Sciences*, vol. 12, no. 2, pp. 71–88, 2002.
- [6] V. Lippiello and L. Villani, "Managing redundant visual measurements for accurate pose tracking," *Robotica*, vol. 21, pp. 511–519, 2003.
- [7] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed. London, UK: Springer-Verlag, 2000.
- [8] M. Paterson and F. Yao, "Efficient binary space partitions for hidden-surface removal and solid modeling," *Discrete and Computational Geometry*, vol. 5, pp. 485–503, 1990.
- [9] W. J. Wilson, C. C. W. 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, 1996.