Position-Based Visual Servoing in Industrial Multirobot Cells Using a Hybrid Camera Configuration

Vincenzo Lippiello, Bruno Siciliano, *Fellow, IEEE*, and Luigi Villani, *Senior Member, IEEE*

***Abstract*—This paper deals with the problem of position-based visual servoing in a multiarm robotic cell equipped with a hybrid eye-in-hand/eye-to-hand multicamera system. The proposed approach is based on the real-time estimation of the pose of a target object by using the extended Kalman filter. The data provided by all the cameras are selected by a suitable algorithm on the basis of the prediction of the object self-occlusions, as well as of the mutual occlusions caused by the robot links and tools. Only an optimal subset of image features is considered for feature extraction, thus ensuring high estimation accuracy with a computational cost independent of the number of cameras. A salient feature of the paper is the implementation of the proposed approach to the case of a robotic cell composed of two industrial robot manipulators. Two different case studies are presented to test the effectiveness of the hybrid camera configuration and the robustness of the visual servoing algorithm with respect to the occurrence of occlusions.**

***Index Terms*—Extended Kalman filter (EKF), occlusion prediction, visual motion estimation, visual servoing.**

I. Introduction

ONE OF THE MOST important features of the new generation of industrial robots is the enhanced sensing capability, which is based on the use of exteroceptive sensors like force and vision.

In a multiarm robotic cell, visual systems are usually composed of two or more cameras that can be rigidly attached to the robot end-effectors (in the so-called eye-in-hand configuration), or fixed in the workspace (in the so-called eye-to-hand configuration) [1]. The first configuration guarantees good accuracy and the ability to explore the workspace, although with a limited sight; the second one ensures a panoramic sight of the workspace, but a lower accuracy. Hence, the use of both configurations at the same time (i.e., the so-called hybrid configuration) makes the execution of complex tasks easier and offers higher flexibility in the presence of a dynamic scenario.

Manuscript received January 22, 2006; revised August 2, 2006. This paper was recommended for publication by Associate Editor B. Nelson and Editor L. Parker upon evaluation of the reviewers' comments. This paper was presented in part at the 44th IEEE Conference on Decision and Control/European Control Conference, Seville, Spain, 2005, and in part at the 6th IEEE Symposium on Computational Intelligence in Robotics and Automation, Otaniemi, Finland, 2005.

The authors are with PRISMA Lab, Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II, 80125 Napoli, Italy (e-mail: vincenzo.lippiello@unina.it; siciliano@unina.it; lvillani@unina.it).

Color versions of Figs. 1 and 3–17 are available online at http://ieeexplore.org.

Digital Object Identifier 10.1109/TRO.2006.886832

Recently, some effort has been made to design visual servoing systems based on hybrid eye-in-hand/eye-to-hand camera configurations. In [2], an eye-to-hand camera is in charge of the robot tool positioning, while an eye-in-hand camera is in charge of the robot tool orientation. A similar approach is used in [3], where an eye-to-hand camera is employed to estimate the robot tool pose with respect to the workspace, and an eye-in-hand camera is employed as data source for object pose estimation. Further, in [4] and [5], a camera mounted on the end-effector of a robot is adopted as an eye-to-hand camera for another robot to benefit from the advantages of a mobile camera.

All the above approaches do not fully exploit the potentialities of hybrid camera configurations. In fact, the information provided by the different types of cameras is employed for different goals, and a complete integration is not really achieved.

Another important issue in multirobot cells is related to the occurrence of occlusions of the object features used as visual data. Occlusions may happen when some parts of a workpiece are hidden with respect to the cameras by other parts of the object itself (self-occlusion), or when a workpiece is hidden by a robot link or tool, or by another object (mutual occlusion). They may cause failures to any kind of algorithm based on the extraction of image features. Hence, it is important to adopt suitable strategies able to cope with this problem.

The issue of computing occlusion-free viewpoints in a known polyhedral world is considered in [6] and [7]. However, the proposed techniques do not consider real-time constraints and moving objects. A simple occlusion-resistant object-tracking algorithm is proposed in [8] and [9], where a Kalman filter is adopted to have a prediction of the target-object trajectory while occlusions occur. In [10], 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 the presence of moving workpieces.

The control scheme considered in this paper can be classified as position-based visual servoing (PBVS), because it requires the computation of position and orientation errors defined in the Cartesian space [11]. The main challenge of position-based algorithms is the real-time estimation of the pose of target objects from visual measurements.

This problem has been largely investigated in the literature. Geometrical solutions have been proposed [12], as well as numerical and iterative approaches, e.g., linear approaches based on least-square methods [13], [14] and full-scale nonlinear optimization techniques [15]–[17].

An alternative approach that is often used for pose estimation is the extended Kalman filter (EKF), thanks to its capability to combine redundant noisy measurements from images in a statistically well-grounded way (see, e.g., [18] and references therein).

The Kalman filter offers many advantages over other estimation methods, e.g., temporal filtering, recursive implementation, ability to change the measurement set during the operation. Also, the pose prediction computed by the filter allows setting up a dynamic windowing technique which may sensibly reduce the time required for image processing. These features are suitably exploited in the algorithm presented here. In fact, differently from previous approaches based on hybrid configurations (see, e.g., [2] and [4]), the pose estimation is achieved by using all the data provided by all the cameras, without any kind of *a priori* discrimination.

An important contribution of this paper is the adoption of a suitable selection algorithm that, at each sampling time, allows the selection of an optimal set of visual data to be used for pose estimation. This set is dynamically changed during the system operation on the basis of the specific task and of the current configuration of the workspace. Only the selected features are grabbed to achieve the measurements, and thus, the computational time spent for image processing is independent of the number of cameras [21].

The computational efficiency of the selection algorithm is improved, thanks to the adoption of a fast occlusion-prediction algorithm purposely designed for multiarm robotics cells. In this way, the occluded features can be eliminated from the set of features candidates to be extracted by the images. The algorithm is based on binary space partitioning (BSP) tree structures to represent the 3-D geometry of the cell. The BSP tree representation is updated in real time on the basis of the measurements of the robot joint positions, and of the estimated poses of the workpieces. Notice that in industrial applications, it is reasonable to assume that the geometry of the cell and of the target objects is known *a priori*. This information is suitably exploited here to speed up the whole estimation process.

Noticeably, the proposed formulation of the Kalman filter allows separating the dependency on the choice of the image features from the choice of the coordinates adopted to represent the pose of the target objects. Hence, it can be applied with straightforward modifications to different kinds of image features, using both eye-in-hand and eye-to-hand cameras and any kind of parameterization of the object orientation.

In this paper, a complete scheme for PBVS on an industrial multiarm robotic cell is described. The proposed formulation generalizes the results presented in previous papers [22] (concerning hybrid camera configurations) and [23] (concerning occlusion detection in multiarm robotic cells).

A salient feature of the paper is the implementation of the proposed approach to the case of a robotic cell composed of two industrial robot manipulators. Two different case studies are presented to test the effectiveness of the hybrid camera configuration and the robustness of the visual servoing algorithm with respect to the occurrence of occlusions.

The paper is organized as follows. In Section II, the whole PBVS scheme is presented. The model of the hybrid camera system and the relevant kinematic equation for the robotic cell are provided in Section III. Section IV introduces the image Jacobian used for the formulation of the EKF presented in Section V. The image-features selection and the occlusion-detection algorithms are described in Section VI. Finally, the experimental results are illustrated in Section VII.

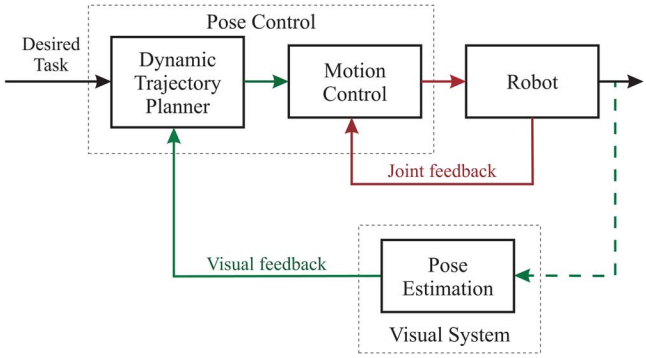


Fig. 1. Block scheme of the PBVS algorithm.

II. PBVS

A typical PBVS scheme for industrial robots is represented in Fig. 1, corresponding to the dynamic position-based look-and-move structure presented in [1]. This algorithm requires the estimation of the pose of a target object with respect to a reference frame (the base frame or the end-effector frame) by using the vision system; the estimated pose is then fed back to a pose controller. Hence, the two main operations to be performed are *pose control* and *pose estimation*.

Notice that pose estimation is a computationally demanding task, because it requires processing of the measurements of some geometric features extracted from the images of one or more cameras. Hence, the frequency of the pose-estimation algorithm is usually lower than the frequency of the pose-control loop (greater or equal to 500 Hz in industrial robots, to guarantee tracking accuracy and disturbance rejection). However, in the application considered in this paper, the bottleneck is represented by the camera frame rate (between 25 and 60 Hz for low-cost cameras used in industrial applications).

Pose control is performed through an inner–outer control loop. The inner loop implements motion control (independent joint control or any kind of joint space or task space control). In the outer loop, the block named "Dynamic Trajectory Planner" computes the trajectory for the end-effector on the basis of the current object pose and of the desired task.

The pose-estimation algorithm provides the measurement of the target object pose. The use of a multicamera system requires the adoption of intelligent and computationally efficient strategies for the management of highly redundant information (a large number of object image features from multiple points of view). This task has to be realized with real-time constraints, and thus, the extraction and interpretation of all the available visual information is not possible.

To solve this problem, an efficient technique (described by the scheme represented in Fig. 2) has been developed, which is able to improve the accuracy and robustness of the estimation

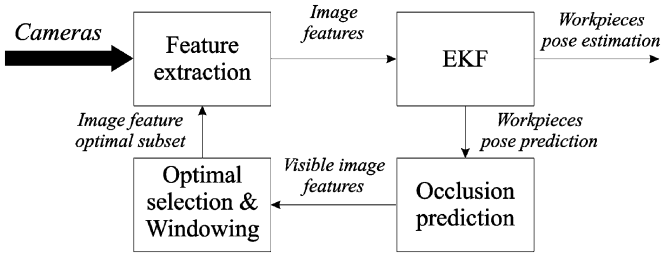


Fig. 2. Block scheme of the pose-estimation algorithm.

by exploiting a minimal set of significant visual data, suitably selected from the initial redundant set.

In detail, the EKF is used to compute an estimate of the object pose on the basis of the image features extracted from visual information. The filter provides also a prediction of the pose of the workpieces at the next sampling time, that is input to the occlusion-prediction algorithm, in charge of filtering out all the image features that are not visible at the next sampling time.

Notice that not all the visible image features provided by the occlusion-prediction algorithm are used for feature extraction. In fact, in a multicamera system, the available image features may be highly redundant, and may increase the computational cost without a significant enhancement of the estimation accuracy [20]. Therefore, a selection algorithm is adopted to dynamically select an optimal set of visible image features. This algorithm is based on the maximization of an objective function depending on a combination of suitable quality indexes, ensuring a balanced spatial distribution of the projections of the image features 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 [21]. Moreover, 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 considerably reduces the computational charge of the frame-grabbing operations.

In the following, the main issues concerning pose estimation and image-feature selection are considered.

III. Modeling

Consider a system of n_f video cameras fixed in the workspace (eye-to-hand cameras) and n_m video cameras mounted on the end-effector of one or more robots (eye-in-hand cameras), with $n = n_f + n_m$. In the following, the index c will be used to denote the quantities referred to a frame $O_c - x_c y_c z_c$ attached to camera c (eye-to-hand or eye-in-hand), with $c \in \{1, \ldots, n\}$. According to the classical pinhole camera model, the camera frame c is chosen with the z_c-axis aligned with the optical axis and the origin in the optical center of the lens. The sensor plane is parallel to the $x_c y_c$-plane at a distance $-\lambda_c$ along the z_c-axis, where λ_c is the effective focal length of the lens. The image plane is parallel to the $x_c y_c$-plane at a distance λ_c along the z_c-axis. The intersection of the optical axis with the image plane defines the principal optical point O'_c, which is the origin of the image frame $O'_c - X_c X_c$, whose axes X_c and Y_c are taken parallel to the axes x_c and y_c, respectively.

Assuming that the projective geometry of the camera is modeled by perspective projection, a point P of the object with coordinates ${}^c\boldsymbol{p} = [x \;\; y \;\; z]^T$ with respect to the camera frame is projected onto the point of the image plane with coordinates

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \frac{\lambda_c}{z} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{1}$$

Without loss of generality, the case of a single moving object is considered. The position and orientation of a frame attached to the object $O_o - x_o y_o z_o$ with respect to a base coordinate frame $O - xyz$ can be expressed in terms of the coordinate vector of the origin $\boldsymbol{o}_o = [x_o \;\; y_o \;\; z_o]^T$ and of the rotation matrix $\boldsymbol{R}_o(\boldsymbol{\varphi}_o)$, where $\boldsymbol{\varphi}_o$ is a $(p \times 1)$ vector corresponding to a suitable parameterization of the orientation. In the case where a minimal representation of the orientation is adopted, e.g., Euler angles, it is $p = 3$, while it is $p = 4$ if unit quaternions are used. Hence, the $(m \times 1)$ vector $\boldsymbol{x}_o = [\boldsymbol{o}_o^T \;\; \boldsymbol{\varphi}_o^T]^T$ defines a representation of the object pose with respect to the base frame, in terms of $m = 3 + p$ parameters.

The homogeneous coordinate vector $\tilde{\boldsymbol{p}} = [\boldsymbol{p}^T \;\; 1]^T$ of point P with respect to the base frame can be computed as

$$\tilde{\boldsymbol{p}} = \boldsymbol{H}_o(\boldsymbol{x}_o)\, {}^o\tilde{\boldsymbol{p}}$$

where ${}^o\tilde{\boldsymbol{p}}$ is the homogeneous coordinate vector of P, with respect to the object frame, and $\boldsymbol{H}_o$ is the homogeneous transformation matrix representing the pose of the object frame referred to the base frame

$$\boldsymbol{H}_o(\boldsymbol{x}_o) = \begin{bmatrix} \boldsymbol{R}_o(\boldsymbol{\varphi}_o) & \boldsymbol{o}_o \\ \mathbf{o}_3^T & 1 \end{bmatrix}$$

where $\mathbf{o}_3$ is the 3×1 null vector. Notice that if the object is rigid, the vector ${}^o\tilde{\boldsymbol{p}}$ is constant, and can be computed from a CAD model of the object.

Let $\boldsymbol{H}_c$ denote the homogeneous transformation matrix representing the pose of the camera frame c referred to the base frame. For the eye-to-hand cameras, the matrix $\boldsymbol{H}_c$ is constant, and can be computed through a suitable calibration procedure [24], while for the eye-in-hand cameras (see Fig. 3), this matrix depends on the camera current pose $\boldsymbol{x}_c$, i.e., $\boldsymbol{H}_c = \boldsymbol{H}_c(\boldsymbol{x}_c)$, and can be computed as

$$\boldsymbol{H}_c(\boldsymbol{x}_c) = \boldsymbol{H}_r\, {}^r\boldsymbol{H}_e({}^r\boldsymbol{x}_e)\, {}^e\boldsymbol{H}_c$$

where $\boldsymbol{H}_r$ is the homogeneous transformation matrix of the base frame of the robot r carrying camera c with respect to the common base frame, ${}^r\boldsymbol{H}_e$ is the homogeneous transformation matrix of the end effector e with respect to the base frame of the robot r, and ${}^e\boldsymbol{H}_c$ is the homogeneous transformation matrix of camera c with respect to frame e of the end-effector where the camera is mounted. Notice that $\boldsymbol{H}_r$ and ${}^e\boldsymbol{H}_c$ are constant and can be estimated through suitable calibration procedures [25],

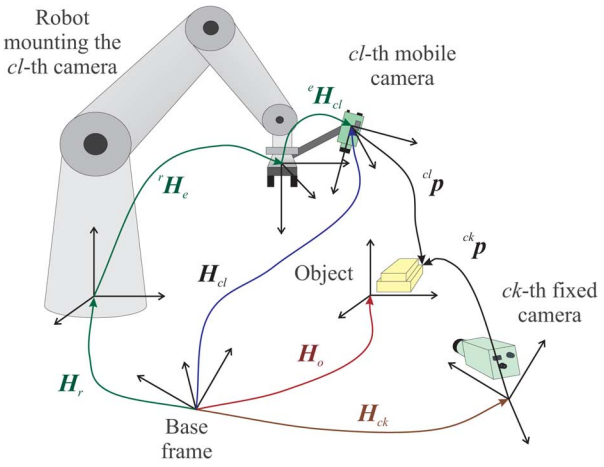


Fig. 3. Eye-in-hand/eye-to-hand cameras.

while ${}^{r}\boldsymbol{H}_{e}$ depends on the current end-effector pose ${}^{r}\boldsymbol{x}_{e}$, and may be computed using the robot kinematic model.

Therefore, the homogeneous coordinate vector of P with respect to the camera frame c can be expressed as

$$ {}^{c}\tilde{\boldsymbol{p}} = {}^{c}\boldsymbol{H}_{o}(\boldsymbol{x}_{o},\boldsymbol{x}_{c})\,{}^{o}\tilde{\boldsymbol{p}} \tag{2} $$

where ${}^{c}\boldsymbol{H}_{o}(\boldsymbol{x}_{o},\boldsymbol{x}_{c}) = {}^{c}\boldsymbol{H}^{-1}(\boldsymbol{x}_{c})\boldsymbol{H}_{o}(\boldsymbol{x}_{o})$. Notice that $\boldsymbol{x}_{c}$ is constant for eye-to-hand cameras; moreover, the matrix ${}^{c}\boldsymbol{H}_{o}$ does not depend on $\boldsymbol{x}_{c}$ and $\boldsymbol{x}_{o}$ separately, but on the relative pose of the object frame with respect to the camera frame.

The 3×1 vector ${}^{c}\boldsymbol{p}$ corresponding to ${}^{c}\tilde{\boldsymbol{p}}$ in (2) is the coordinate vector of point P with respect to the camera frame c. Hence, its time derivative ${}^{c}\dot{\boldsymbol{p}}$ has the meaning of relative velocity of point P, with respect to the camera frame c, expressed in frame c. If the camera is fixed, this velocity represents also the absolute velocity of point P with respect to the base frame, expressed in the camera frame, denoted by ${}^{c}\boldsymbol{v}_{P}$. When the camera frame is moving, the contribution to ${}^{c}\boldsymbol{v}_{P}$ due to the motion of the camera must be taken into account.

The velocity of the camera frame c with respect to the base frame can be characterized in terms of the translational velocity of the origin $\boldsymbol{v}_{O_c}$ and of angular velocity $\boldsymbol{\omega}_{c}$. These vectors, expressed in the camera frame c, define the velocity screw ${}^{c}\boldsymbol{\nu}_{c} = [\,{}^{c}\boldsymbol{v}_{O_c}^{T}\ \ {}^{c}\boldsymbol{\omega}_{c}^{T}\,]^{T}$. Then, the absolute velocity of point P can be computed as

$$ {}^{c}\boldsymbol{v}_{P} = {}^{c}\dot{\boldsymbol{p}} + \boldsymbol{\Lambda}({}^{c}\boldsymbol{p})\,{}^{c}\boldsymbol{\nu}_{c} \tag{3} $$

with $\boldsymbol{\Lambda}(\cdot) = [\,\boldsymbol{I}_{3}\ \ -\boldsymbol{S}(\cdot)\,]$, where $\boldsymbol{I}_{3}$ is the 3×3 identity matrix, and $\boldsymbol{S}(\cdot)$ denotes the 3×3 skew-symmetric matrix operator.

Equation (3) holds for any point P of the object, hence, it can be applied to the origin O_o of the object frame. This yields

$$ {}^{c}\boldsymbol{v}_{O_o} = {}^{c}\dot{\boldsymbol{o}}_{o} + \boldsymbol{\Lambda}({}^{c}\boldsymbol{o}_{o})\,{}^{c}\boldsymbol{\nu}_{c} \tag{4} $$

where ${}^{c}\boldsymbol{o}_{o}$ is the vector of the coordinates of O_o with respect to the camera frame c, ${}^{c}\dot{\boldsymbol{o}}_{o}$ is the relative velocity of O_o with respect to the camera frame c, while ${}^{c}\boldsymbol{v}_{O_o}$ is its absolute velocity; all the quantities are expressed in the camera frame c. On the other hand, the absolute angular velocity ${}^{c}\boldsymbol{\omega}_{o}$ of the object frame expressed in the camera frame c can be computed as

$$ {}^{c}\boldsymbol{\omega}_{o} = {}^{c}\boldsymbol{\omega}_{c,o} + {}^{c}\boldsymbol{\omega}_{c} \tag{5} $$

where ${}^{c}\boldsymbol{\omega}_{c,o}$ represents the relative angular velocity of the object frame with respect to the camera frame. Equations (4) and (5) can be rewritten in the compact form

$$ {}^{c}\boldsymbol{\nu}_{o} = {}^{c}\boldsymbol{\nu}_{c,o} + \boldsymbol{\Gamma}({}^{c}\boldsymbol{o}_{o})\,{}^{c}\boldsymbol{\nu}_{c} \tag{6} $$

where ${}^{c}\boldsymbol{\nu}_{o} = [\,{}^{c}\boldsymbol{v}_{O_o}^{T}\ \ {}^{c}\boldsymbol{\omega}_{o}^{T}\,]^{T}$ is the velocity screw corresponding to the absolute motion of the object frame, ${}^{c}\boldsymbol{\nu}_{c,o} = [\,{}^{c}\dot{\boldsymbol{o}}_{o}^{T}\ \ {}^{c}\boldsymbol{\omega}_{c,o}^{T}\,]^{T}$ is the velocity screw corresponding to the relative motion of the object frame with respect to the camera frame c, and the matrix $\boldsymbol{\Gamma}(\cdot)$ is defined as

$$ \boldsymbol{\Gamma}(\cdot) = \begin{bmatrix} \boldsymbol{I}_{3} & -\boldsymbol{S}(\cdot) \\ \boldsymbol{O}_{3} & \boldsymbol{I}_{3} \end{bmatrix} $$

where $\boldsymbol{O}_{3}$ denotes the 3×3 null matrix. The velocity screw ${}^{c}\boldsymbol{\nu}_{o}$ can be expressed in terms of the time derivative of the vector $\boldsymbol{x}_{o}$ through the equation

$$ {}^{c}\boldsymbol{\nu}_{o} = {}^{c}\boldsymbol{L}(\boldsymbol{x}_{o})\dot{\boldsymbol{x}}_{o} \tag{7} $$

where ${}^{c}\boldsymbol{L}(\cdot)$ is a Jacobian matrix depending on the particular choice of coordinates for the orientation, hereafter referred to as *pose representation Jacobian* [26].

IV. Image Features

An image feature is any structural feature that can be extracted from an image, corresponding to the projection of a physical feature of the object onto the camera image plane. An image feature can be characterized by a set of parameters that can be calculated from the image. Examples are, e.g., the image-plane coordinates of points, the distance between two points in the image plane and the orientation of the line connecting those two points, the area of the projected surface, and the parameters of lines in the image plane. The image-feature parameters for camera c can be grouped in a vector $\boldsymbol{f}_{c} = [\,f_{c,1}\ \ \dots\ \ f_{c,k}\,]^{T}$, where $f_{c,j}$ is a real value and k is the dimension of the image-feature parameter space. The mapping from the position and orientation of the object to the corresponding image-feature parameters can be computed using the projective geometry of the camera and can be written in the form

$$ \boldsymbol{f}_{c} = \boldsymbol{g}_{c}({}^{c}\boldsymbol{H}_{o}(\boldsymbol{x}_{o},\boldsymbol{x}_{c})) \tag{8} $$

where only the dependence from the relative pose of the object frame with respect to the camera frame has been explicitly evidenced. Usually, for visual tracking and visual servoing applications, the computation of the differential mapping is required

$$\dot{\boldsymbol{f}}_c = \boldsymbol{J}_{c,o}{}^c\boldsymbol{\nu}_{c,o} \tag{9}$$

where the matrix $\boldsymbol{J}_{c,o}$ is the Jacobian mapping of the relative velocity screw of the object frame, with respect to the camera frame, into the variation of the image-feature parameters.

Taking into account the velocity composition (6), equation (9) can be rewritten in the form

$$\dot{\boldsymbol{f}}_c = \boldsymbol{J}_{c,o}{}^c\boldsymbol{\nu}_o - \boldsymbol{J}_c{}^c\boldsymbol{\nu}_c \tag{10}$$

where $\boldsymbol{J}_{c,o}$ is also the Jacobian corresponding to the contribution of the absolute velocity screw of the object frame, hereafter referred to as *image Jacobian*, while

$$\boldsymbol{J}_c = \boldsymbol{J}_{c,o}\boldsymbol{\Gamma}({}^c\boldsymbol{o}_o) \tag{11}$$

is the Jacobian corresponding to the contribution of the absolute velocity screw of the camera frame. The Jacobian (11) is known in the literature as the interaction matrix [27].

The computation of the image Jacobian $\boldsymbol{J}_{c,o}$ depends on the type of image-feature parameters, and can be performed using a procedure similar to that adopted in [27] to compute the interaction matrix $\boldsymbol{J}_c$.

V. Pose Estimation

In this section, the problem of the estimation of the pose vector $\boldsymbol{x}_o$ of the object, with respect to the base frame, from the measurements of the image parameters $\boldsymbol{f}_c$ obtained using a system of n eye-in-hand/eye-to-hand cameras is considered. The proposed solution is based on the EKF. To this purpose, a discrete-time state-space dynamic model has to be considered, describing the object motion. The state vector of the dynamic model is chosen as $\boldsymbol{w} = [\boldsymbol{x}_o^T \ \dot{\boldsymbol{x}}_o^T]^T$. For simplicity, the object velocity is assumed to be constant over one sampling period T_s. This approximation is reasonable, in the hypothesis that T_s is sufficiently small and is usual in visual servoing applications (see, e.g., [11]). Models with constant acceleration could also be adopted (see, e.g., [28]) in the cases where this approximation is not realistic.

The dynamic modeling error can be considered as an input disturbance $\boldsymbol{\gamma}$ described by zero-mean Gaussian noise with covariance $\boldsymbol{Q}$. Hence, the discrete-time dynamic model can be written as

$$\boldsymbol{w}_k = \boldsymbol{A}\boldsymbol{w}_{k-1} + \boldsymbol{\gamma}_k \tag{12}$$

where $\boldsymbol{A}$ is the $(2m \times 2m)$ block matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_m & T_s\boldsymbol{I}_m \\ \boldsymbol{O}_m & \boldsymbol{I}_m \end{bmatrix}.$$

The output of the Kalman filter is the vector of the image-feature parameters measured on the image planes of the n cameras at time kT_s

$$\boldsymbol{\zeta}_k = [\boldsymbol{\zeta}_{1,k}^T \dots \boldsymbol{\zeta}_{n,k}^T]^T$$

where $\boldsymbol{\zeta}_{c,k} = \boldsymbol{f}_{c,k} + \boldsymbol{\mu}_{c,k}$, $c = 1, \dots, n$, $\boldsymbol{\mu}_{c,k}$ being the measurement noise of camera c. The measurement noise is assumed to be zero-mean Gaussian noise with covariance $\boldsymbol{N}_c$. The covariance matrix can be evaluated during the calibration procedure of the cameras or by means of specific experiments.

Taking into account the (8), the output model of the Kalman filter can be written in the form

$$\boldsymbol{\zeta}_k = \boldsymbol{g}(\boldsymbol{w}_k) + \boldsymbol{\mu}_k$$

where $\boldsymbol{\mu}_k = [\boldsymbol{\mu}_{1,k}^T \dots \boldsymbol{\mu}_{n,k}^T]^T$ and

$$\boldsymbol{g}(\boldsymbol{w}_k) = \begin{bmatrix} \boldsymbol{g}_1(\boldsymbol{w}_k, \boldsymbol{x}_{1,k}) \\ \vdots \\ \boldsymbol{g}_n(\boldsymbol{w}_k, \boldsymbol{x}_{n,k}) \end{bmatrix} \tag{13}$$

with $\boldsymbol{g}_c(\boldsymbol{w}_k, \boldsymbol{x}_{c,k}) = \boldsymbol{g}_c({}^c\boldsymbol{H}_o(\boldsymbol{x}_{o,k}, \boldsymbol{x}_{c,k}))$. Notice that $\boldsymbol{x}_{c,k}$ is the pose of the camera frame c at time kT_s; this quantity is known and constant for eye-to-hand cameras, while it can be computed from the robot direct kinematics for eye-in-hand cameras. For this reason, the explicit dependence of $\boldsymbol{g}$ from $\boldsymbol{x}_{c,k}$ has been omitted in (13).

Since the output model is nonlinear in the system state, the EKF must be adopted. The first step of the EKF algorithm provides an optimal estimate of the state at the next sampling time, according to the recursive equations

$$\begin{aligned} \hat{\boldsymbol{w}}_{k,k-1} &= \boldsymbol{A}\hat{\boldsymbol{w}}_{k-1,k-1} \\ \boldsymbol{P}_{k,k-1} &= \boldsymbol{A}\boldsymbol{P}_{k-1,k-1}\boldsymbol{A}^T + \boldsymbol{Q}_{k-1} \end{aligned}$$

where $\boldsymbol{P}_{k,k-1}$ is the covariance matrix of the estimate state error. The second step improves the previous estimate by using the input measurements according to the equations

$$\begin{aligned} \hat{\boldsymbol{w}}_{k,k} &= \hat{\boldsymbol{w}}_{k,k-1} + \boldsymbol{K}_k(\boldsymbol{\zeta}_k - \boldsymbol{g}(\hat{\boldsymbol{w}}_{k,k-1})) \\ \boldsymbol{P}_{k,k} &= \boldsymbol{P}_{k,k-1} - \boldsymbol{K}_k\boldsymbol{C}_k\boldsymbol{P}_{k,k-1} \end{aligned}$$

where $\boldsymbol{K}_k$ is the $(12 \times 2ns)$ Kalman matrix gain

$$\boldsymbol{K}_k = \boldsymbol{P}_{k,k-1}\boldsymbol{C}_k^T(\boldsymbol{\pi}_k + \boldsymbol{C}_k\boldsymbol{P}_{k,k-1}\boldsymbol{C}_k^T)^{-1}$$

$\boldsymbol{C}_k$ being the Jacobian matrix of the output function

$$\boldsymbol{C}_k = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\hat{\boldsymbol{w}}_{k,k-1}} = \left[\frac{\partial \boldsymbol{g}(\boldsymbol{w})}{\partial \boldsymbol{x}_o} \quad \boldsymbol{O}\right]_{\boldsymbol{w}=\hat{\boldsymbol{w}}_{k,k-1}}$$

where $\boldsymbol{O}$ is a null matrix of proper dimensions corresponding to the partial derivative of $\boldsymbol{g}$ with respect to the velocity variables; this quantity is null because function $\boldsymbol{g}$ does not depend on the velocity.

In view of (8) and (13), the computation of $\boldsymbol{C}_k$ requires the computation of the Jacobian matrix of $\boldsymbol{g}_c$ with respect to $\boldsymbol{x}_o$. In the case of eye-in-hand cameras, it is

$$\dot{\boldsymbol{g}}_c = \dot{\boldsymbol{f}}_c = \frac{\partial \boldsymbol{g}_c}{\partial \boldsymbol{x}_o}\dot{\boldsymbol{x}}_o + \frac{\partial \boldsymbol{g}_c}{\partial \boldsymbol{x}_c}\dot{\boldsymbol{x}}_c. \tag{14}$$

On the other hand, the time derivative of $\boldsymbol{f}_c$ can be computed also according to (10) as a function of ${}^c\boldsymbol{\nu}_o$ and ${}^c\boldsymbol{\nu}_c$. The velocity screw ${}^c\boldsymbol{\nu}_o$ can be computed according to (7), and ${}^c\boldsymbol{\nu}_c$ can be expressed in the form

$${}^c\boldsymbol{\nu}_c = {}^c\boldsymbol{L}(\boldsymbol{x}_c)\dot{\boldsymbol{x}}_c.$$

Hence, comparing (10) with (14), the following noticeable equality can be found:

$$\frac{\partial \boldsymbol{g}_c}{\partial \boldsymbol{x}_o} = \boldsymbol{J}_{c,o}\,{}^c\boldsymbol{L}(\boldsymbol{x}_o).$$

Obviously, the same result holds for eye-to-hand cameras (in this case, $\dot{\boldsymbol{x}}_c = \boldsymbol{0}$ and ${}^c\boldsymbol{\nu}_c = \boldsymbol{0}$).

The above equality allows separating the computation of the Jacobian dependent on the choice of the image features (i.e., the image Jacobian $\boldsymbol{J}_{c,o}$) from the computation of the Jacobian dependent on the object-pose representation (i.e., the pose representation Jacobian ${}^c\boldsymbol{L}(\boldsymbol{x}_o)$). By virtue of this property, the Kalman filter formulation becomes quite flexible, and can be applied with straightforward modifications to different kinds of image features, using both eye-in-hand and eye-to-hand cameras and any kind of parameterization of the object orientation.

As a final remark, it is worth noticing that the change of image features during system operation does not affect the state of the EKF, which remains continuous, but only the output function (13) and the Jacobian matrix $\boldsymbol{C}_k$.

VI. Image Features Selection

The selection of the features to be extracted from the images is a fundamental issue for the pose-estimation problem. Two main aspects are to be considered. The first one is that some of the features are not always visible, because they are not in the camera field of view or are (partially) occluded by the object itself, by other objects, or by the arms of the robots of the cell. The second aspect is that the accuracy of the EKF depends on the "quality" of the set of measurements, that can be dynamically changed during the task execution.

In this paper, an algorithm for a multiarm robotic system is used to recognize in real time the presence of occlusions with respect to each camera. The algorithm is based on BSP tree structures to represent the 3-D geometry of the cell [21], [29]. Moreover, a feature-selection algorithm is adopted to find an optimal subset of image features among those available after occlusion detection, to be used for pose estimation.

A. BSP Tree

A BSP tree data structure may be employed to represent the geometry of known 3-D environments, like a robotic cell with one or more robot manipulators and workpieces. To achieve a computationally fast representation, attention is limited to polyhedrical 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 polyhedrical solids.

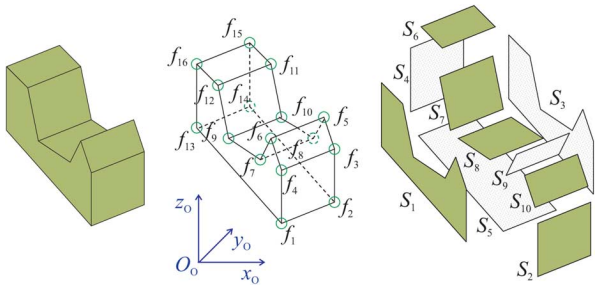


Fig. 4. Example of decomposition of a 3-D object into elementary surfaces.

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. 4, 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 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 3-D space into two subspaces containing all the surfaces (or pieces of surfaces) which are in the front and in the back, respectively, of the partition plane; all the surfaces lying on the partition plane are stored in the node. Recursively applying 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 paper, the partition planes are selected from the set of planes 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. 5, a BSP tree representing the object of Fig. 4 is reported, where the choice of the partition planes does not generate intersections.

B. Occlusion Prediction

For the purpose of the occlusion-prediction algorithm presented in this paper, the robotic cell (see Fig. 6) is seen as a collection of objects which includes the workpieces (target objects), the robot links and tools, and all of the possible obstacles that may occlude the workpieces with respect to the cameras.

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

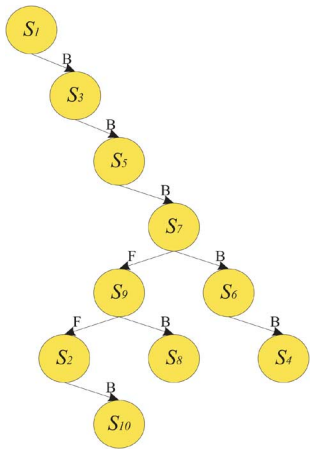


Fig. 5. Possible BSP tree representation of the object of Fig. 4 ; labels F and B denote the front and back subspaces.

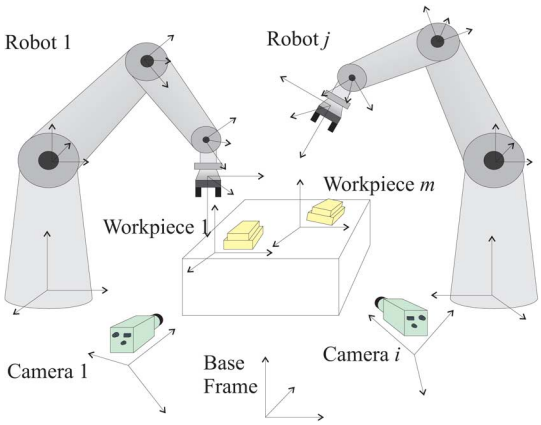


Fig. 6. Sketch of the multiarm robotic cell.

of the workpieces provided by the visual-tracking algorithm described in Section VII.

The occlusion–prediction algorithm can be decomposed into two parts: geometric modeling and occlusion detection.

The geometric modeling part, described in Fig. 7, is aimed at generating online a BSP tree, representing the 3-D geometrical model of the cell [30]. To this purpose, the CAD models 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 offline to facilitate the online BSP tree construction.

The first step of the modeling procedure is the computation of the poses of all the robot's links and tools using the joint position measurements and the direct kinematics. The second step is the computation of the pose of all the objects of the cell, estimated using the EKFs (one EKF for each object) with respect to the frame of camera c. At this point, 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.

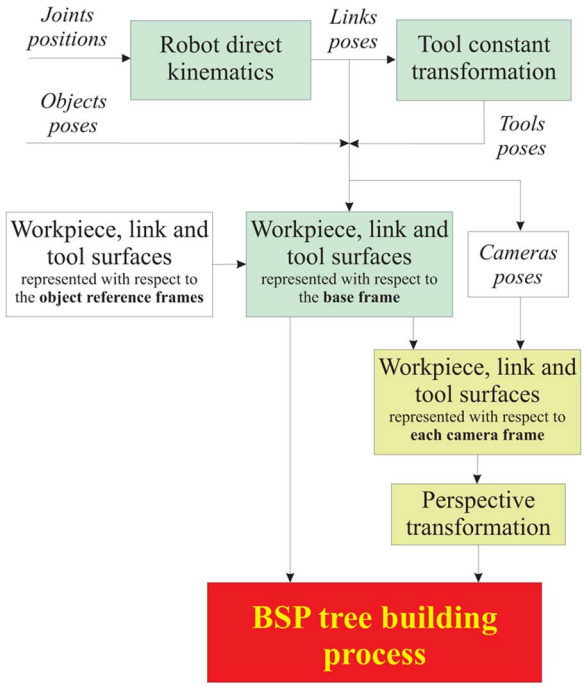


Fig. 7. Dynamic BSP tree-building process.

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 (POV) (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. These 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 POV 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 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 POV. Notice that the construction of the set proceeds so that the surfaces are considered in a sequence corresponding to their position, with respect to the POV, 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 [30]. Moreover, some modifications can be introduced, both in the modeling and the occlusion detection parts, which allow a considerable reduction of computational time [29].

Notice that the occlusion-detection algorithm was developed for the case of point features. This choice is quite general, because any object contour can be sampled and represented as a set of points.

C. Optimal Selection and Windowing

The occlusion-prediction algorithm recognizes all the feature points that are visible from a camera POV. However, this does not ensure that all the visible points are well-localizable, i.e., their positions can be effectively measured with a given accuracy. For instance, some points could be out of the image plane of the camera, or they could be too close each other to guarantee absence of ambiguity in the localization. Moreover, the number of the well-localizable feature points may be larger than the optimal number of points ensuring the best pose-estimation accuracy.

In the following, a windowing test is adopted to find the projections of the feature points that can be well-localized. Then, a selection algorithm is used to choose an optimal subset of points to be considered for feature extraction.

The measurements of the coordinates of the projections of the feature points are obtained by considering suitable rectangular windows of the image plane to be grabbed and processed. Each window must contain one feature point. The windows are centered on the positions of the feature points on the image plane, as predicted by the Kalman filter. Their semidimensions are dynamically chosen in the interval $[Wr_{\min}, Wr_{\max}]$ for the base (the side parallel to the row's direction) and in the interval $[Wc_{\min}, Wc_{\max}]$ for the height (the side parallel to the column's direction). The minimum values are set so as to achieve a prescribed accuracy and robustness in the feature extraction, while the maximum values are set on the basis of the available memory and processing time.

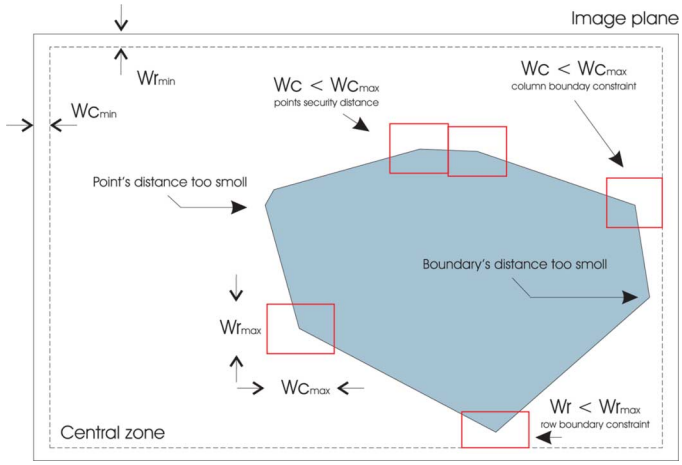


Fig. 8. Examples of significant situations during windowing test.

A windowing test can be set up to select all the projections of the feature points that can be well-localized.

First, all the points that are out of the field of view of the camera, or too close to the boundaries of the image plane, are discarded. This is achieved by eliminating all the points whose projections, as predicted by the Kalman filter, are out of a central window of the image plane. The central window is obtained by reducing the height (base) of the whole image plane of the quantity $Wr_{\min}$ ($Wc_{\min}$) from each side, as shown in Fig. 8.

Then, all the feature points that are too close to each other are discarded. This happens when the estimated distance between the projections of two or more points is lower than $S_f \cdot Wr_{\min}$ ($S_f \cdot Wc_{\min}$) along the row's (column's) direction; $S_f > 1$ is a suitable security factor.

All the remaining points are well-localizable; the effective dimensions of the corresponding windows are dynamically adapted to the maximum allowable semidimension, so as to guarantee an assigned security distance from the other points and from the boundaries of the image plane (see Fig. 8).

The number of feature points after occlusion prediction and windowing test may be larger than the number of points (five non-coplanar points) sufficient to ensure good pose estimation, according to the experimental tests in [20]. The optimality of a given set of feature points can be valued through the composition of suitably selected quality indices into an optimal objective function. The quality indices must be able to provide accuracy and robustness, and minimize the oscillations in the pose-estimation variables. To achieve this goal, it is necessary to ensure an optimal spatial distribution of the projections of the feature points on the image plane, and to avoid chattering events between different optimal subsets of feature points chosen during the object motion.

A first quality index is the measure of spatial distribution of the predicted projections on the image planes of a subset of r selected points

$$Q_s = \prod_{c=1}^{n} \left(\frac{1}{r_c} \sum_{k=1}^{r_c} \min_{\substack{j \in \{1,\dots,n\} \\ j \neq k}} \left\| \boldsymbol{p}_{j,c} - \boldsymbol{p}_{k,c} \right\| \right)$$

where $\boldsymbol{p}_{i,c}$ is the vector of the feature-point coordinates in the image plane of camera c, and $r = \sum_{c=1}^{n} r_c$, r_c being the number of points selected from camera c.

A second quality index is the measure of angular distribution of the predicted projections on the image plane of a subset of r selected points

$$Q_a = \prod_{c=1}^{n} \left(1 - \sum_{k=1}^{r_c} \left| \frac{\alpha_{k,c}}{2\pi} - \frac{1}{r_c} \right| \right)$$

where $\alpha_{k,c}$ is the angle between the vector $\boldsymbol{p}_{k+1,c} - \boldsymbol{p}_{C,c}$ and the vector $\boldsymbol{p}_{k,c} - \boldsymbol{p}_{C,c}$, $\boldsymbol{p}_{C,c}$ being the central gravity point of the whole subset of feature points of camera c, and the r_c points of the subset are considered in a counterclockwise-ordered sequence with respect to $\boldsymbol{p}_{C,c}$, with $\boldsymbol{p}_{r_c+1,c} = \boldsymbol{p}_{1,c}$. Together with the previous one, this index is aimed at maximizing the distance among the points and avoiding collinearity.

A third quality index is the measure of visibility of a given feature (considered as a collection of points of the contour), which takes into account the partially occluded features. This index is defined as

$$Q_v = \prod_{c=1}^{n} \left(\frac{1}{r_c} \sum_{k=1}^{r_c} \frac{m_{k,c}}{m_{t,k}} \right)$$

where $m_{k,c}$ is the number of visible points of the feature k from camera c, and $m_{t,k}$ is the total number of the points used to represent the feature.

The above indices are applied to each set of feature points projected on the image planes of all the cameras. In order to distribute the points among the cameras, the following index is considered:

$$Q_d = \left(\min_{c \in \{1,\dots,n\}} \frac{d_c}{q} \right) \sum_{c=1}^{n} \frac{q_c}{d_c}$$

where q_c is the number of points assigned to camera c, and d_c is the distance of the camera c from the object. This index takes into account the distance of the cameras from the object, and thus allows managing different resolution zones of different cameras.

Notice that correspondences between points seen by different cameras are not explicitly considered in the proposed technique. In fact, the only correspondences that are used are those between the points measured on the image planes of each camera and the object model. The presence of feature points distributed among different cameras allows achieving an implicit triangulation which improves depth-estimation accuracy.

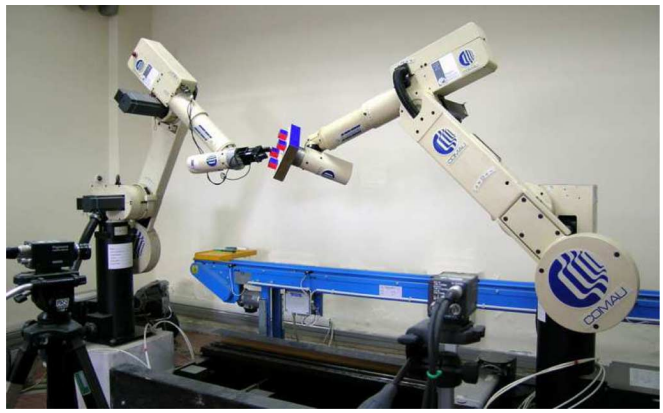


Fig. 9. Industrial robotic cell.

Since accuracy and robustness of estimation is higher when the points are not all coplanar [31], a non-coplanarity index can be considered

$$Q_c = \frac{r - r_M}{r - 3}$$

where $r_M \in [3, r]$ is the maximum number of coplanar points in the set of r points.

To avoid chattering phenomena, a quality index introducing hysteresis effects on the change of the optimal combination of points is considered

$$Q_h = \begin{cases} 1 + \epsilon, & \text{if actual} = \text{previous combination} \\ 1, & \text{otherwise} \end{cases}$$

where ϵ is a positive constant.

The proposed indices are only some of the possible choices, but guarantee satisfactory performance when used with the occlusion-prediction algorithm and the windowing test presented in this paper. Other examples of quality indices are proposed, e.g., in [31].

The objective function to be maximized is the product of the quality indices, and must be evaluated for all the possible combinations of the visible points on r positions. In order to perform a computationally efficient determination of the optimal set at each sampling time, the initial optimal combination of points is first evaluated offline; then, only the combinations that modify at most one point with respect to the current optimal combination are tested online. This allows a considerable reduction of processing time, although it is not guaranteed that the current solution remains optimal; however, it is reasonable that this solution is close to the optimal one.

VII. Experiments

The effectiveness of the proposed approach has been tested in some experiments on the industrial robotic cell composed of two industrial robots, Comau SMART-3 S (see Fig. 9). One arm (R7AX) is mounted on a sliding track, which provides an additional degree of freedom with respect to the standard six degrees of freedom 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 of a PC equipped with two Matrox GENESIS boards and two Sony 8500CE B/W cameras. The Matrox boards are used as frame grabbers and for 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 PBVS scheme represented in Fig. 1 has been adopted. The stereo visual system 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.

In the experiments presented here, the object corners are used as image features. Moreover, roll, pitch, and yaw (RPY) angles are used for orientation representation.

To test the effectiveness of the proposed algorithm, two different case studies are considered. The first case study is aimed at testing the accuracy of the hybrid camera configuration with respect to the fixed camera configuration. In the second case study, the effectiveness of the occlusion detection algorithm on a multirobot industrial cell is proven.

A. *First Case Study*

A visual-synchronization task for the dual-arm robotic cell has been realized. The R6AX robot (*leader* robot) is used to move an object in the visual space of the fixed camera; thus, the object position and orientation with respect to the base frame can be computed both from the joint position measurements, via the direct kinematic equation, and from visual measurements. The R7AX robot (*follower* robot) is visually guided to preserve a fixed mutual pose with respect to the object. Notice that the measurements of the object pose computed from joint-position measurements are used here only to evaluate the pose-estimation error of the visual system.

A hybrid visual system is adopted. The first camera (with focal length $f_e = 16$ mm) is fixed, and observes the entire workspace from a distance of about 1 m, while the second camera (with focal length $f_e = 8$ mm) is mounted on the end-effector of the R7AX robot.

An important issue is the calibration of the camera setup. In detail, the eye-to-hand camera was calibrated with respect to the base frame using the calibration algorithm proposed in [24], while the eye-in-hand camera was calibrated with respect to the end-effector frame of the follower robot using the calibration algorithm proposed in [25].

The value of the matrix $\boldsymbol{P}_{1,0}$ has been set to zero; moreover, the initial value of the state vector $\boldsymbol{w}_{1,0}$ has been set to null for the velocity components, while the pose components have been roughly estimated through direct measurements. Finally, the covariance matrixes $\boldsymbol{Q}$ and $\boldsymbol{N}$ have been chosen as $\boldsymbol{Q} = \mathrm{diag}\{0,0,0,0,0,0,5,5,5,20,20,20\} \cdot 10^{-6}, \boldsymbol{N} = \mathrm{diag}\{9,9,\ldots,9\}$, where $\boldsymbol{N}$ is a $(2s \times 2s)$ matrix, s being the number of selected features. The values of the observation noise covariances have been evaluated during the camera calibration procedure, while the values of the state-noise covariances have been set on the basis of the velocity range of the object trajectories.

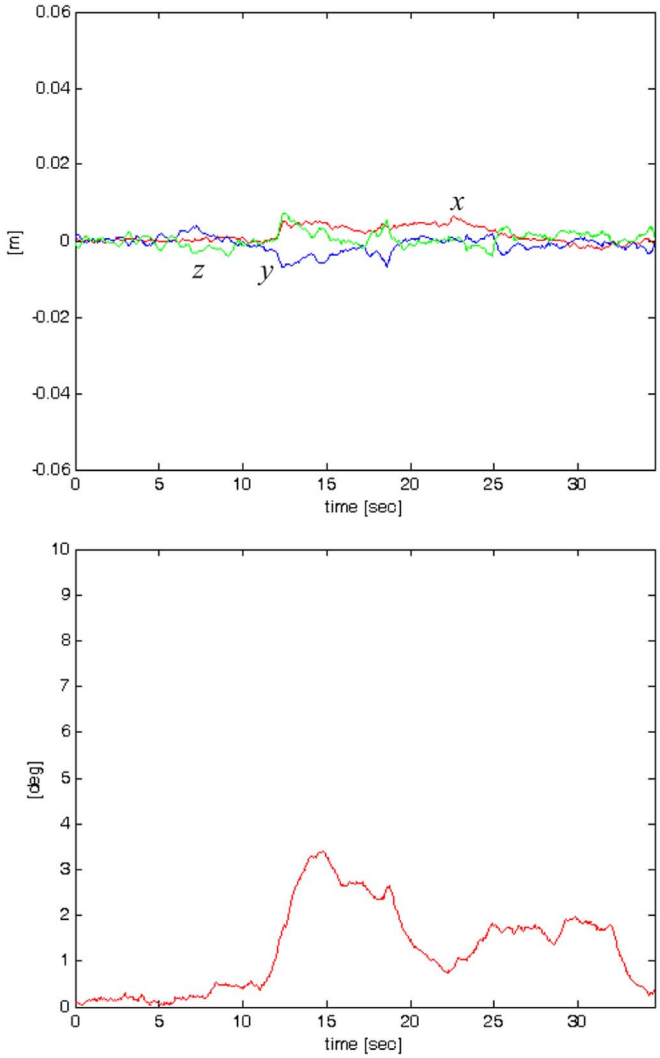


Fig. 10. Object trajectory. Top: position trajectory. Bottom: orientation trajectory.

Two different experiments are considered. In the first experiment, both cameras are employed to estimate the pose of the objects, while in the second experiment, only the fixed camera is used. This allows comparing the pose-tracking performance of the hybrid configuration with respect to that of the eye-to-hand configuration.

The trajectory of the object is represented in Fig. 10. The pose is referred to as the initial object pose. The orientation is represented using RPY angles.

The time history of the pose-estimation error for the first experiment is shown in Fig. 11. The orientation error is evaluated as the angle of an axis/angle representation corresponding to the rotation matrix of the follower end-effector frame with respect to the estimated object frame. Notice that the position error

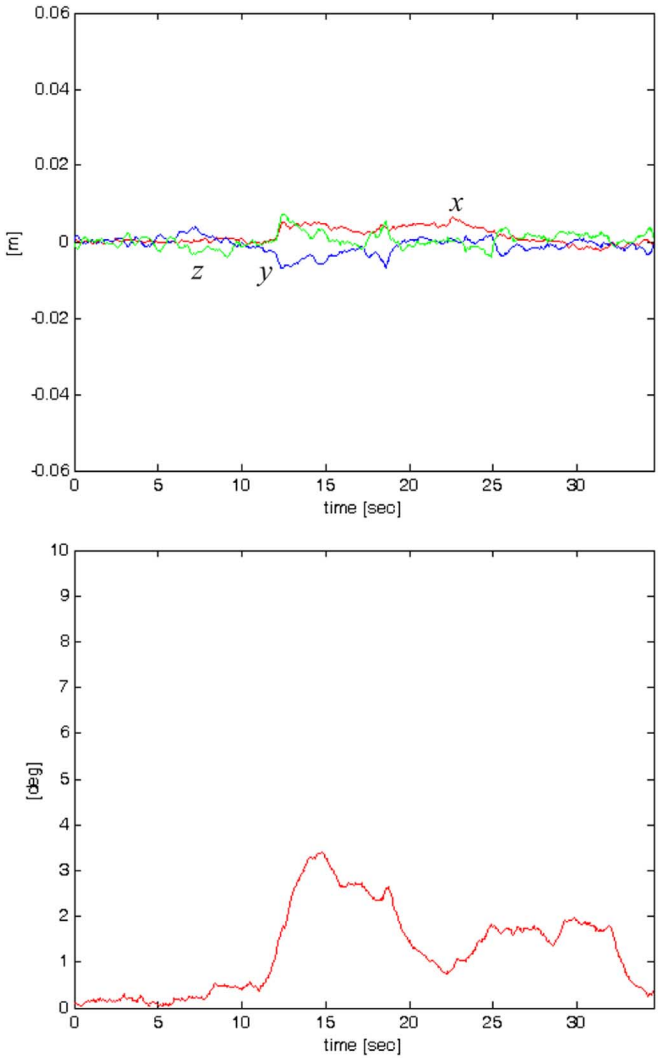


Fig. 11. Time history of the pose-estimation error for the first case study. Top: position error. Bottom: orientation error.

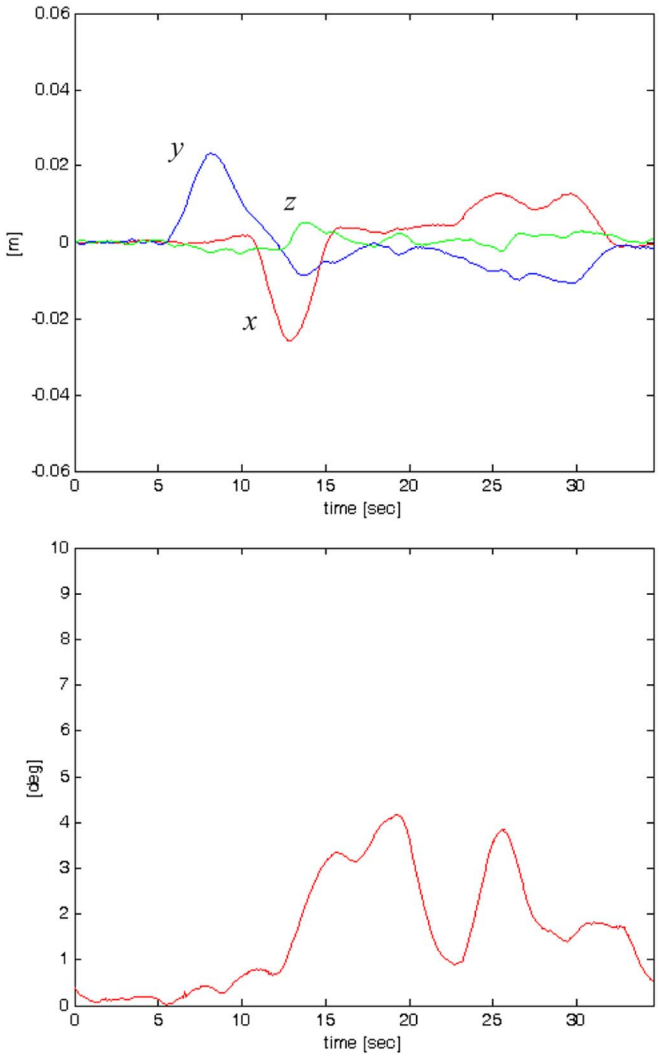


Fig. 12. Time history of the pose-estimation error for the first experiment of the first case study. Top: position error. Bottom: orientation error.

for the three components has the same magnitude and is lower than 1 cm; moreover, the orientation error is lower than 3°. Notice also that the errors are not significantly influenced by the velocity of the object; this is due to the adoption of a camera moving with the follower robot according to the object motion, in addition to the fixed camera.

In Fig. 12, the time history of the pose error of the end-effector frame of the follower robot with respect to the estimated object frame is reported. The errors are higher with respect to the estimation errors of Fig. 11, because of the time delay caused by the low frame rate of the visual system.

The time history of the pose-estimation error for the second experiment of the first case study is shown in Fig. 13. Due to the adoption of only one fixed camera, the estimation error is sensibly higher than in the first experiment (Fig. 11). Moreover, the pose-estimation error is quite sensitive to the velocity of the object, especially for the orientation. The pose error of the end-effector frame of the follower robot with respect to the estimated object frame is not affected by the camera configuration, and is not reported here for brevity.

A comparison with the eye-in-hand configuration would be also significant, but it is not reported here for brevity. The expected result is a lower accuracy in depth estimation with respect to the hybrid configuration (due to the monocular vision), but better than in the fixed-camera case, because the eye-in-hand camera remains closer to the object during tracking.

B. Second Case Study

In the second case study, three experiments of a vision-guided grasping task have been performed to test the effectiveness of the occlusion-detection algorithm. In detail, one robot is used to grasp a moving object; the other robot occludes the object during task execution.

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 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

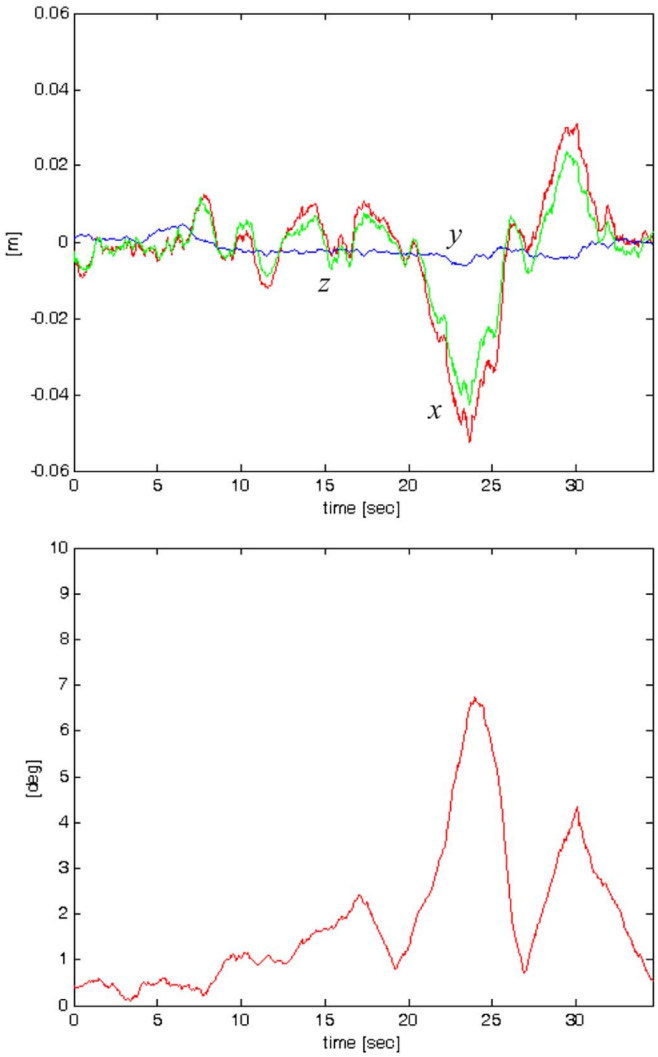


Fig. 13. Time history of the pose-estimation error for the second experiment of the first case study. Top: position error. Bottom: orientation error.

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.

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. 14. The time history of the orientation components is not reported here for brevity. The Approach phase begins after about 4 s and ends in about 12 s. During this phase, the robot recognizes the workpiece and moves over it, 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 s. 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 s, the Manipulate phase begins, and the robot returns to the HOME pose carrying the workpiece. At about 50 s, the

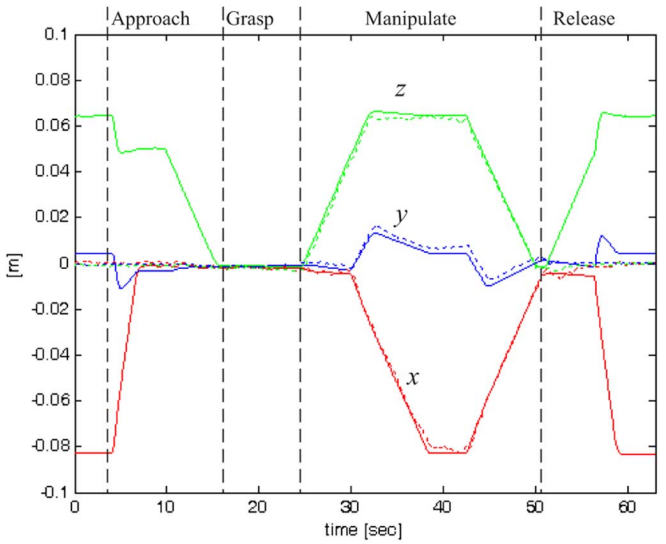


Fig. 14. Gripper (solid) and estimated object (dashed) trajectories during the first experiment of the second case study.

robot reaches the FINAL pose and releases the object; then, it returns to the HOME pose (Release phase).

In Fig. 15, the state of the visible and selected feature points is represented. In particular, the feature point number is reported on the vertical axis, and time is reported on the horizontal axis. For each feature point, a couple of horizontal tracks are considered: the bottom (top) track is marked in all the time samples where the feature point is visible (selected for feature extraction); hence, the top track can be marked only if the bottom track is marked too, but the opposite is not true. Notice that the feature 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).

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

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. 17. 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 as that in Fig. 16. In Fig. 17, 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 camera 2 between the Approach and Grasp phases. This event does not affect the accuracy of

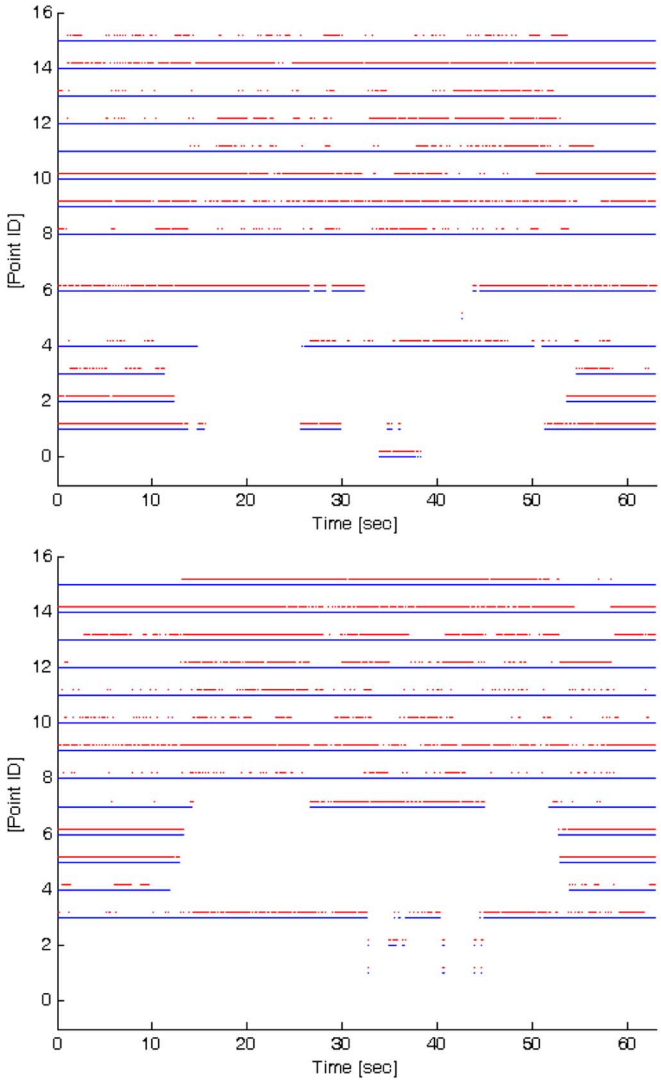


Fig. 15. Visible and selected object feature points for camera 1 (top) and camera 2 (bottom) during the first experiment of the second case study.

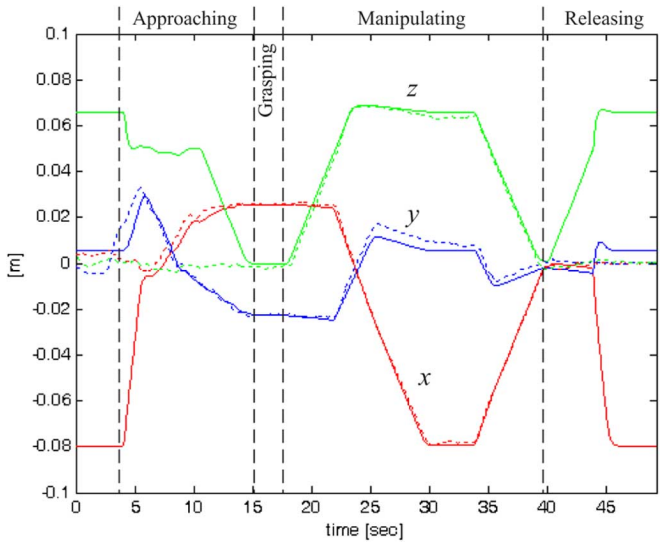


Fig. 16. Gripper (solid) and estimated object (dashed) trajectories during the second experiment of the second case study.

the pose estimation, and allows the successful execution of the Grasp phase. Moreover, during the Manipulate phase, the robot

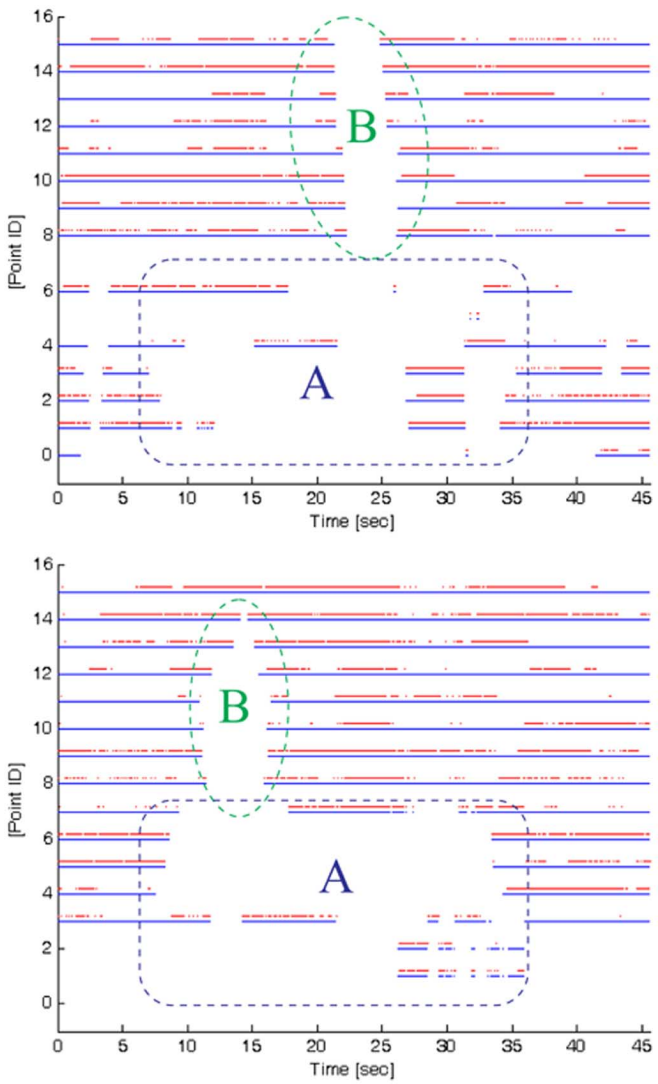


Fig. 17. Visible and selected object feature points for camera 1 (top) and camera 2 (bottom) during the third experiment of the second case study.

R6AX completely occludes the object with respect to camera 1. Again, from Fig. 16, 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).

VIII. Conclusion

A PBVS algorithm using a hybrid eye-in-hand/eye-to-hand multicamera configuration is presented in this paper. The data provided by all the cameras are fully exploited, so that the benefits of both eye-in-hand and of the eye-to-hand configurations are preserved. Moreover, the adoption of an occlusion-detection algorithm and of a selection algorithm in charge of choosing an optimal subset of image features ensures a low computational cost for image processing, independent of the number of cameras. The experimental results have shown the superior performance of the hybrid configurations with respect to the eye-to-hand configuration in terms of pose-tracking accuracy, and have confirmed the robustness of the proposed approach with respect to the occurrence of occlusions.

REFERENCES

[1] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[2] G. Flandin, F. Chaumette, and E. Marchand, "Eye-in-hand/eye-to-hand cooperation for visual servoing," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2000, pp. 2741–2746.

[3] M. Elena, M. Cristiano, F. Damiano, and M. Bonfe, "Variable structure PID controller for cooperative eye-in-hand/eye-to-hand visual servoing," in *Proc. IEEE Int. Conf. Control Appl.*, Jun. 2003, pp. 989–994.

[4] A. Muis and K. Ohnishi, "Eye-to-hand approach on eye-in-hand configuration within real-time visual servoing," in *Proc. IEEE Int. Workshop Adv. Motion Control*, Mar. 2004, pp. 647–652.

[5] T. Murao, H. Kawai, and M. Fujita, "Passivity-based control of visual feedback systems with dynamic movable camera configuration," in *Proc. IEEE Conf. Decision, Control/Eur. Control Conf.*, Dec. 2005, pp. 5360–5365.

[6] K. Tarabanis, R. Y. Tsai, and A. Kaul, "Computing occlusion-free viewpoints," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 3, pp. 279–292, Mar. 1996.

[7] P. C. Ho and W. Wang, "Occlusion culling using minimum occluder set and opacity map," in *Proc. IEEE Int. Conf. Inf. Visual.*, Jul. 1999, pp. 292–300.

[8] P. Wunsch and G. Hirzinger, "Real-time visual tracking of 3D objects with dynamic handling of occlusion," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1997, pp. 2868–2873.

[9] E. Loutas, K. Diamantaras, and I. Pitas, "Occlusion resistant object tracking," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2001, pp. 65–68.

[10] C. Laugier, A. Ijel, and J. Troccaz, "Combining vision based information and partial geometric models in automatic grasping," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1990, pp. 676–682.

[11] W. J. Wilson, C. C. W. Hulls, and G. S. Bell, "Relative end-effector control using Cartesian position based visual servoing," *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 684–696, Oct. 1996.

[12] M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives, "Determination of the attitude of 3D objects from a single perspective view," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 12, pp. 1265–1278, Dec. 1989.

[13] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, and M. Kim, "Pose estimation from corresponding point data," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 6, pp. 1426–1446, Jun., 1989.

[14] D. Dementhon and L. Davis, "Model-based object pose in 25 lines of code," *Int. J. Comput. Vis.*, vol. 15, pp. 123–141, 1995.

[15] D. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 5, pp. 441–450, May, 1991.

[16] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 932–946, Jul. 2002.

[17] A. Comport, D. Kragic, E. Marchand, and F. Chaumette, "Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 2852–2857.

[18] T. Broida and R. Chellappa, "Estimation of object motion parameters from noisy images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 1, pp. 90–99, Jan. 1986.

[19] D. Gennery, "Visual tracking of known three-dimensional objects," *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 243–270, 1992.

[20] J. Wang and W. J. Wilson, "3D relative position and orientation estimation using Kalman filter for robot control," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1992, pp. 2638–2645.

[21] V. Lippiello and L. Villani, "Managing redundant visual measurements for accurate pose tracking," *Robotica*, vol. 21, pp. 511–519, 2003.

[22] V. Lippiello, B. Siciliano, and L. Villani, "An occlusion prediction algorithm for visual servoing tasks in a multi-arm robotic cell," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, Jun. 2005, pp. 733–738.

[23] ——, "Eye-in-hand/eye-to-hand multi-camera visual servoing," in *Proc. IEEE Int. Conf. Decision, Control/Eur. Control Conf.*, Dec. 2005, pp. 5354–5359.

[24] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 10, pp. 965–980, Oct. 1992.

[25] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Trans. Robot. Autom.*, vol. 5, no. 3, pp. 345–358, Jun. 1989.

[26] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed. London, U.K.: Springer-Verlag, 2000.

[27] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robot. Autom.*, vol. 8, no. 3, pp. 313–326, Jun. 1992.

[28] D. Koller, G. Klinker, E. Rosem, D. Breen, R. Whitaker, and M. Tuceryan, "Real-time vision-based camera tracking for augmented reality applications," in *Proc. ACM Symp. Virtual Reality, Softw., Technol.*, 1997, pp. 87–94.

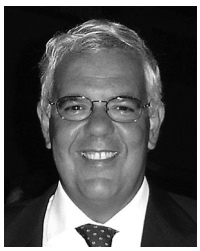
[29] V. Lippiello, "Real-time visual tracking based on BSP-tree representations of object boundary," *Robotica*, vol. 23, pp. 365–375, 2005.

[30] M. Paterson and F. Yao, "Efficient binary space partitions for hidden-surface removal and solid modeling," *Discr. Comput. Geom.*, vol. 5, pp. 485–503, 1990.

[31] F. Janabi-Sharifi and W. J. Wilson, "Automatic selection of image features for visual servoing," *IEEE Trans. Robot. Autom.*, vol. 13, no. 6, pp. 890–903, Dec. 1997.

Vincenzo Lippiello was born in Naples, Italy, on June 19, 1975. He received the Laurea degree in electronic engineering and the Research Doctorate degree in information engineering from the University of Naples, Naples, Italy, in 2000 and 2004, respectively.

Currently, he is a Postdoctoral Fellow with the Department of Computer and Systems Engineering, University of Naples. His research interests include visual servoing of robot manipulators, image processing, hybrid visual/force control, and adaptive control. He has published 20 journal and conference papers.



Bruno Siciliano (M'91–SM'94–F'00) was born in Naples, Italy, on October 27, 1959. He received the Laurea degree and the Research Doctorate degree in electronic engineering from the University of Naples, Naples, Italy, in 1982 and 1987, respectively.

Currenlty, he is a Professor of Control and Robotics, and Director of the PRISMA Lab in the Department of Computer and Systems Engineering, University of Naples. His research interests include identification and adaptive control, impedance and force control, visual tracking and servoing, redundant and cooperative manipulators, lightweight flexible arms, space robots, human-centered and service robotics. He has coauthored 7 books, 65 journal papers, 160 conference papers and book chapters, and has delivered 80 invited lectures and seminars at institutions worldwide.

Dr. Siciliano is a Fellow of ASME. He is Coeditor of the Springer Tracts in Advanced Robotics series, the Springer Handbook of Robotics, and has served on the Editorial Boards of several journals, as well as Chair or Co-chair for numerous international conferences. He is on the Board of the European Robotics Network. He has served the IEEE Robotics and Automation Society as Vice-President for Technical Activities and Vice-President for Publications, as a member of the AdCom, and as a Distinguished Lecturer. Currently, he is the RA Society President-Elect.

Luigi Villani (S'94–M'97–SM'03) was born in Avellino, Italy, on December 5, 1966. He received the Laurea degree in electronic engineering and the Research Doctorate degree in electronic engineering and computer science from the University of Naples, Naples, Italy, in 1992 and 1996, respectively.

He is currently an Associate Professor of Automatic Control in the Department of Computer and Systems Engineering, University of Naples. His research interests include force/motion control of manipulators, cooperative robot manipulation, lightweight flexible arms, adaptive and nonlinear control of mechanical systems, visual servoing, fault diagnosis, and fault tolerance for dynamical systems. He has coauthored 3 books, 35 journal papers, and 70 conference papers and book chapters.

Dr. Villani is an Associate Editor on the Conference Editorial Board of the IEEE Control Systems Society and an Associate Editor on the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.