Image-Based Visual Servoing Using an Optimized Trajectory Planning Technique

Mohammad Keshmiri, *Student Member, IEEE*, and Wen-Fang Xie, *Senior Member, IEEE*

***Abstract*—Trajectory planning is a useful technique in robotics for guiding the robot through complicated tasks. In this paper, a new semi-offline trajectory planning method is developed to perform image-based visual servoing (IBVS) tasks for a 6 DOFs robotic manipulator system. This method extends the operation range of the system compared with the traditional IBVS controllers. In this method, the camera's velocity screw is parametrized using time-based profiles. The parameters of the velocity profile are then determined by minimizing the cost function consisting of the error between the initial and desired features while respecting the system constraints. A depth-estimation algorithm is proposed to provide the trajectory planning algorithm with a good estimation of the initial depth. The algorithm for planning the orientation of the robot is decoupled from the position planning of the robot. This method eliminates the limitation caused by camera's field of view. The algorithm is validated via the experiment on a 6 DOFs Denso robot in an eye-in-hand configuration. The experimental results demonstrate that the proposed method can overcome some major IBVS drawbacks such as surpassing the system limits and causing instability of the system in fulfilling the tasks which require a 180° rotation of the camera about its center.**

***Index Terms*—Decoupled planning, optimization, robotics, trajectory planning, visual servoing.**

I. Introduction

VISION-BASED control or visual servoing has been used as a solution in robotic industry to increase the dexterity and intelligence of robotic systems, especially in unstructured environments. More and more applications emerge as the machines and computer science evolve. Visual servoing provides solution to the problem of robot positioning using visual features and references [1]–[3]. Two main different visual servoing controllers are introduced in the literature, i.e., position-based visual servoing (PBVS) and image-based visual servoing (IBVS). In PBVS, the acquired visual data are used to extract the three-dimensional (3-D) geometric structure of the environments. A PBVS controller can then be designed to control the robot to interact with the reconstructed environment. In contrast to extracting the information of the 3-D geometric structure, the image data could also be used directly to control the robot. In an IBVS controller, the current image features are compared with the image features of the desired picture taken when the robot is in its desired location and the control signal is generated to guide the robot to its desired position [4].

Manuscript received October 23, 2014; revised June 16, 2016; accepted August 17, 2016. Date of publication August 24, 2016; date of current version February 14, 2017. Recommended by Technical Editor X. Chen. This work was supported by the Discovery Grant, Natural Sciences and Engineering Research Council of Canada.

M. Keshmiri is with the Department of Computer Science, McGill University, Montreal, QC H3A 0G4, Canada (e-mail: mohammad.keshmiri@mail.mcgill.ca).

W.-F. Xie is with the Department of Mechanical and Industrial Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: wfxie@encs.concordia.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMECH.2016.2602325

Despite the great amount of development of visual servoing technology in the last two decades, it still suffers from a number of problems which prevent it from wide industrial use. High nonlinearity of a visual servoing system is the main cause for the problems. Moreover, visual servoing systems are usually implemented using an online controller, which could only observe the current state of the system and at most predict a finite horizon of the system state. Therefore, the real-time decision taken by the controller could lead to failure of the system at later stages of the task. In [5], some potential problems of implementing visual servoing are presented. The research presented in this paper is mainly focused on IBVS and its related issues. Overall, the most prominent deficiencies of IBVS, preventing it from practical employment, are listed as follows.

1) Stability of IBVS is only guaranteed in a region close to the desired pose.
2) IBVS controllers could get stuck in a local minima situation.
3) Inability to perform pure rotation about camera's center.
4) Unknown path of the robot prior to the tasks.
5) Features could leave the field of view (FOV).

Various methodologies have been presented in the literature to overcome the deficiencies. The solution provided by introducing image moment could deal with the interaction matrix singularity and local minima problem, although they were introduced to solve other problems and brought up other deficiencies to the system [6]–[8]. Model predictive visual servoing controller was introduced to deal with the constraints of the system and prevent the features from leaving the FOV [9], [10]. Augmented IBVS (AIBVS) makes the visual servoing smoother and reduces the risk of features leaving the FOV [11]. Although lots of research has been devoted to solve one or two of the above-mentioned problems, a reliable and practical solution suitable for industrial robot applications cannot be found in the literature.

As the IBVS problems persist, the researchers try to integrate the machine vision techniques with trajectory planning techniques to overcome the problems. The trajectory or path planning techniques search and provide the best or near best solution to accomplish a specific task. These paths are mostly different from the ones that a real-time controller would produce. Chesi *et al.* [12] proposed a trajectory planning method for PBVS. Homogeneous forms are used to parameterize the path and a linear matrix inequality optimization is carried out to calculate the parameters. Some other techniques were also used in PBVS path planning [13]. An adaptive trajectory regeneration method was proposed in [14] for visual servoing in an unstructured environment. Later on, the navigation guidance technique was integrated with visual servoing to achieve fast visual servoing [15], [16].

Although the reported trajectory planning techniques demonstrate good performance in executing visual servoing tasks, they were designed for PBVS. Thus, they suffer from PBVS drawbacks such as sensitivity to model and camera calibration errors. This gap motivated the researchers to develop a trajectory planning technique in an IBVS system [17], [18]. Potential field methods were used to perform IBVS online trajectory planing in robotic systems [19], [20]. Potential field techniques are useful in the presence of obstacles and when the system is subjected to constraints. However, both IBVS and PBVS trajectory planning algorithms based on potential field techniques have the following disadvantages. The robot only considers a constraint when it gets close to it, since the potential field technique is an online planning technique that investigates and plans a path based on the local information. However, when the robot is close to its constraints, it is not possible to go back and choose another path. In addition, the potential field technique suffers from local minima when the attraction force magnitude is equal to that of the repelling force on the robot in the opposite direction.

In IBVS trajectory planning, the main challenge is to find a path in image space that corresponds to a feasible path in task space. The most basic solution to this problem is using stereo vision and the epipolar geometry constraint between two camera images. Utilizing the privilege of epipolar geometry, an image trajectory is generated on both images in a way that corresponds to a feasible or even straight line trajectory in Cartesian space [21], [22]. The high load of processing and also the decreased usable FOV area in cameras are the problems of the methods. Moreover, using the probabilistic methods for path planning, as reported in some research [23], [24], is another useful approach. However, it also suffers from the excessive computational load.

In this paper, a new image-based trajectory planning algorithm is proposed to overcome the five visual servoing deficiencies reviewed above and provides a reliable visual servoing operation. In this approach, a trajectory is planned based on the information received from the image plane. However, the trajectory is generated in the Cartesian space and relates the end effector velocity to the motion of the features in image space. For this matter, the camera's velocity screw is separated into six elements. Each velocity element is parameterized using a time-based function which is referred to as the velocity profiles. The velocity profile parameters are determined through an optimization process that minimizes the features' errors between the desired features and the measured ones. In order to facilitate the optimization technique, six new features are introduced. Due to the highly coupled behavior of the features with respect to the motion of camera, the optimization process very slowly converges to a global minima. By decoupling the orientation planning from positioning planning, the optimization process can further be sped up. A convexity analysis is performed and a numerical method is used to valid and demonstrate the convexity of the optimization problem. Similar to the other IBVS systems, the depth estimation plays an important role in the proposed trajectory planning algorithm. A depth-estimation technique is introduced to obtain the initial depth that will be used in the visual servoing task. By integrating this technique, the proposed image-based trajectory planning can overcome the above-mentioned deficiencies to a great extent. In addition, this method allows the visual servoing system not to be subjected to the FOV constraints by just following the generated path, while the conventional IBVS systems are usually limited by such constraints. The proposed technique exploits the benefits of global offline planning in visual servoing. Furthermore, the high speed of the algorithm allows the fast and easy execution of the algorithm in the real industry application. Overall, the contributions presented in this paper are listed as follows.

1) Presenting a novel IBVS using a semi-offline trajectory planning.
2) Introducing a new procedure for finding the trajectory parameters through an optimization process.
3) Presenting six new visual features to facilitate the optimization procedure.
4) Proposing a depth-estimation technique to obtain the initial depth used in the visual servoing task.

Experimental tests are performed on a 6 DOF Denso robot to validate the proposed method. The results show that in the situations where the visual servoing task fails using a traditional method, it is performed successfully using the proposed method in this paper. Although the calibration error could deviate the robot from its ideal path, the developed controller can guide the robot to a position close to the desired location. The desired location is then reached using an AIBVS controller. In other words, the trajectory planning algorithm is switched to a visual servoing controller at the end of its path to improve the accuracy. In summary, the whole visual servoing procedure consists of four stages. The first stage is the depth-estimation stage. The second stage is the offline trajectory planning stage. In the third stage, the planned trajectory is executed. Finally, the trajectory planning block switches to a visual servoing controller block [11]. The controller compensates for any error from the trajectory planning procedure. Each stage is elaborated in the following sections.

This paper is organized as follows. In Section II, the visual servoing model is presented. A trajectory planning procedure is described in Section III. The constraints in visual servoing are introduced in Section IV, and the optimization and convexity analysis are given in Section V. Experimental results are presented in Section VI. Finally, the conclusion is drawn in Section VII.

II. Visual Servoing System

IBVS is performed based on the difference between the current image features and the desired ones. A picture of the object

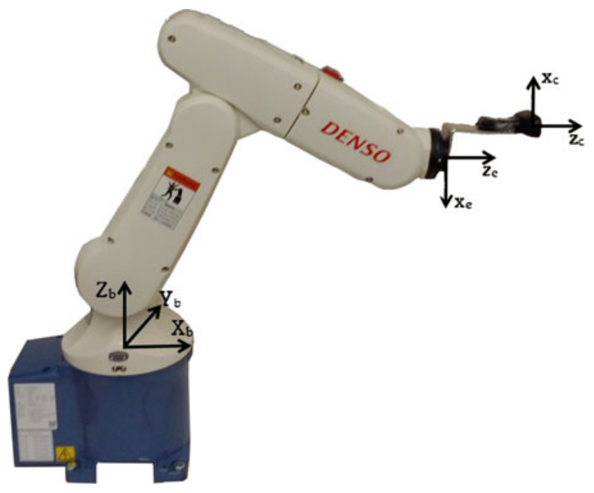


Fig. 1. Denso robot.

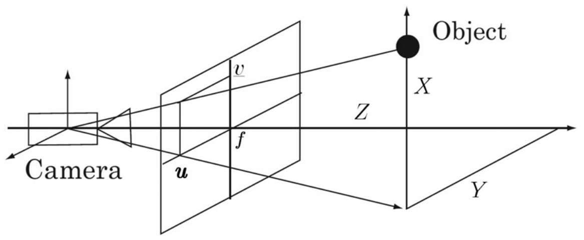


Fig. 2. Schematic of the camera projection model.

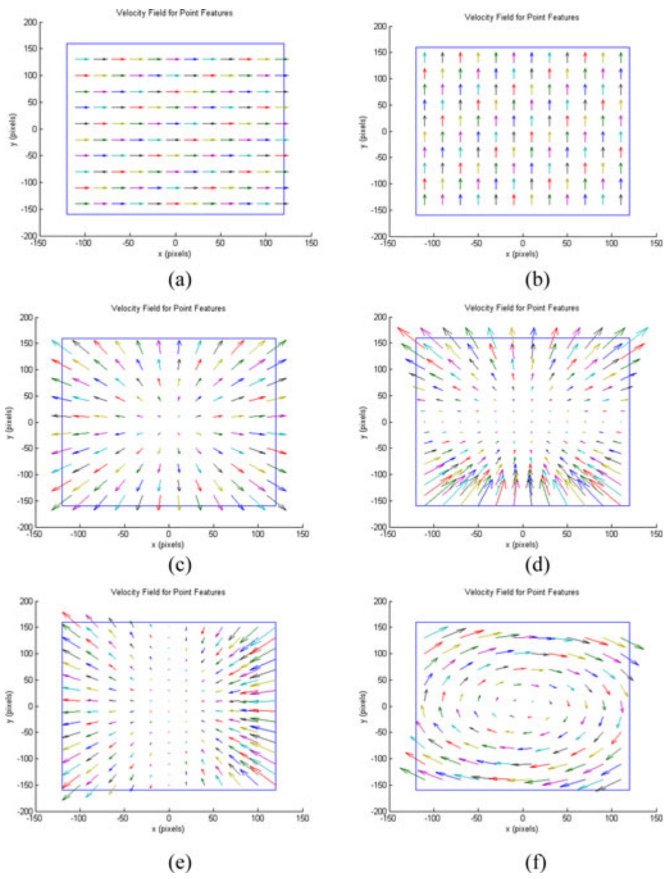


Fig. 3. Velocity field of the features subject to camera velocities. (a) Velocity field for v_x motion. (b) Velocity field of v_y motion. (c) Velocity field of v_z motion. (d) Velocity field of ω_x motion. (e) Velocity field of ω_y motion. (f) Velocity field of ω_z motion.

is taken when the robot's end effector is in the desired position with respect to the object. This picture is used to extract the desired features. The IBVS controller generates a velocity or acceleration command to eliminate the existing error. The visual servoing task is completed when the image features match the target features.

All the system modeling is based on a 6 DOF robotic system with a pinhole charge-coupled device (CCD) camera mounted on its end effector. Let $\mathfrak{F}_b$ be the robot base frame, $\mathfrak{F}_e$ be the end-effector frame, and $\mathfrak{F}_c$ be the camera frame (see Fig. 1). The object is stationary in the workspace and it is characterized by four feature points on its four corners. A merit worth mentioning of IBVS is that it does not require the object frame. Having the projection of 3-D points on the image plane of the camera (see Fig. 2), the relation between the motion of the camera and that of the features could be calculated from

$$\dot{\mathbf{p}} = \mathbf{L}_s\, {}^c\mathbf{V}_c \tag{1}$$

where

$$\mathbf{L}_s = \begin{bmatrix} -\dfrac{1}{Z} & 0 & \dfrac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\dfrac{1}{Z} & \dfrac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{2}$$

is the interaction matrix, x and y represent the point coordinates in image plane, Z is the depth of the object with respect to the camera, and ${}^c\mathbf{V}_c = [v_x\ v_y\ v_z\ \omega_x\ \omega_y\ \omega_z]^{\mathrm{T}}$ is the camera's velocity screw represented in the camera frame.

III. Path Planning

The robot could perform 6 degrees of motion to reach any desired pose (including position and orientation). The effect of each motion could be calculated using (2). Fig. 3 shows how each motion affects the feature point position. The first two elements of the velocity screw create linear motions in the same direction for all features [see Fig. 3(a) and (b)]. These two camera motions are used for displacing the features in the x- and y-directions of the image plane. A camera motion in the Z_c-direction creates an outward motion for the features, which is in the direction of the line connecting the center of the image to the image feature [see Fig. 3(c)]. A negative motion in the Z_c-direction creates an inward motion for the features. This motion could reduce the distances between the features. The fourth and fifth elements of the velocity screw create an inward motion for features on one side of the image and an outward for the features on the other side of the image [see Fig. 3(d) and (e)]. The last element of the velocity screw rotates the features about the center of image [see Fig. 3(f)].

The concept behind the trajectory planning is that any target features could be reached by using a combination of the shown feature motions. Six basic velocity profiles are generated for each of the camera's velocity screw elements. The effect of the generated velocity screw can be calculated using (2). In other words, by superposing the velocity fields caused by each element of the velocity screw, the final position of the features could be calculated. The parameters of the camera velocity are then determined by minimizing the error between the image features and the target ones.

The features velocity in image space could be written as a function of velocity screw elements, given by

$$\begin{aligned}\dot{x}_i &= \frac{-1}{Z}v_x + \frac{x_i}{Z}v_z + x_i y_i \omega_x - (1+x_i^2)\omega_y + y_i\omega_z \\ \dot{y}_i &= \frac{-1}{Z}v_y + \frac{y_i}{Z}v_z + (1+y_i^2)\omega_x - x_i y_i \omega_y - x_i\omega_z \qquad (3)\end{aligned}$$

where $\dot{x}_i$ and $\dot{y}_i$ are the velocities of the ith image feature in the x- and y-directions, respectively. Consequently, the image feature position could be calculated as

$$\begin{aligned}x_{\mathrm{it}} &= \int_{t_0}^{t} (\dot{x}_i(t))dt + x_{i0} \\ y_{\mathrm{it}} &= \int_{t_0}^{t} (\dot{y}_i(t))dt + y_{i0} \qquad (4)\end{aligned}$$

where x_{i_0} and y_{i_0} are the initial coordinates of the image features, and x_{it} and y_{it} are the locations of the image features at time t. Thus, by knowing the initial position of the features and the velocity of the camera, the position of the features can be calculated at any time.

A. Image Features

A new set of six features is required to completely stabilize the system and control all 6 DOFs

$$\mathbf{s_n} = \begin{bmatrix} x_c & y_c & p_z & \theta_x & \theta_y & \theta_z \end{bmatrix}^{\mathrm{T}} \qquad (5)$$

where x_c and y_c are the centers of the feature points and p_z is the perimeter of the lines connecting each consecutive feature point, which are given as

$$\begin{aligned}x_c &= \frac{\sum_{i=1}^{4} x_i(t)}{4} \\ y_c &= \frac{\sum_{i=1}^{4} y_i(t)}{4} \\ p_{z_i} &= \sum_{i=1}^{3}\sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2} \\ p_{z_4} &= \sqrt{(x_1-x_4)^2 + (y_1-y_4)^2} \\ p_z &= \sum_{i=1}^{3} p_{z_i} + p_{z_4}. \qquad (6)\end{aligned}$$

$\theta_x(t)$, $\theta_y(t)$, and $\theta_y(t)$ are defined based on the deformation that is made in the features by rotating the camera about ${}^cX_c$, ${}^cY_c$, and ${}^cZ_c$. These features are given by

$$\begin{aligned}\theta_x &= \frac{\theta_{11}+\theta_{12}}{2} \\ \theta_y &= \frac{\theta_{21}+\theta_{22}}{2} \\ \theta_z &= \frac{\theta_{31}+\theta_{32}}{2} \qquad (7)\end{aligned}$$

where θ_{11}, θ_{12}, θ_{21}, θ_{22}, θ_{31}, and θ_{32} are shown in Fig. 4.

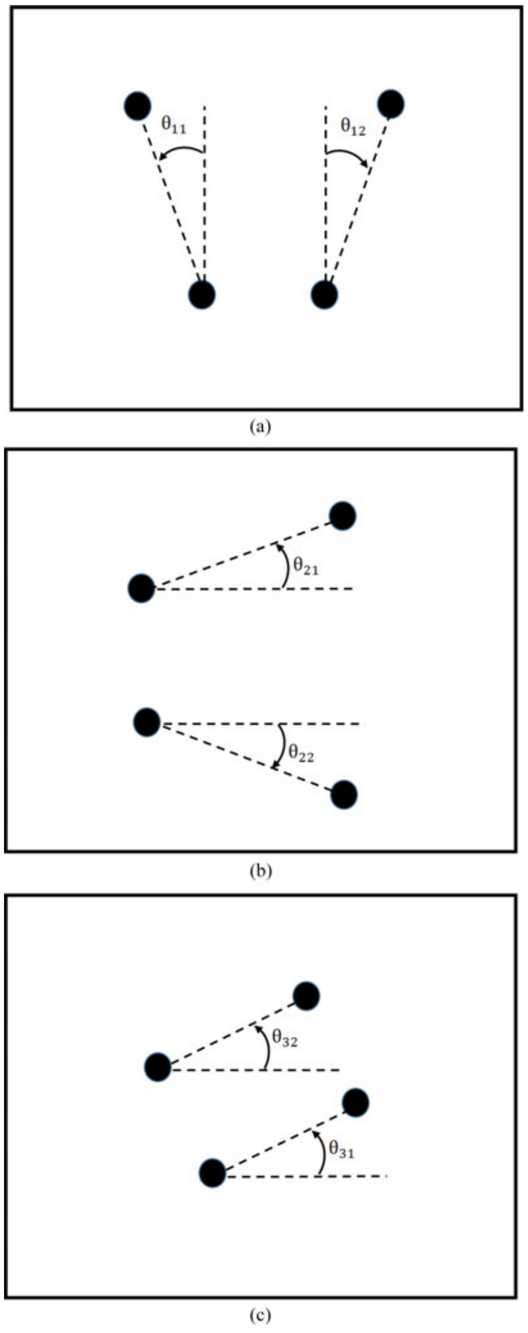


Fig. 4. Last three image features definition. (a) Fourth image feature. (b) Fifth image feature. (c) Sixth image feature.

Therefore, we can write the last three features in terms of the location of the features in image plane. Thus, we have

$$\theta_x = \frac{1}{2}\left(\operatorname{atan}\left(\frac{x_2 - x_1}{y_1 - y_2}\right) - \operatorname{atan}\left(\frac{x_3 - x_4}{y_4 - y_3}\right)\right)$$
$$\theta_y = \frac{1}{2}\left(\operatorname{atan}\left(\frac{y_3 - y_2}{x_3 - x_2}\right) - \operatorname{atan}\left(\frac{y_4 - y_1}{x_4 - x_1}\right)\right)$$
$$\theta_z = \frac{1}{2}\left(\operatorname{atan}\left(\frac{y_1 - y_4}{x_4 - x_1}\right) + \operatorname{atan}\left(\frac{y_2 - y_3}{x_3 - x_2}\right)\right). \quad (8)$$

Taking the time derivative of the features, we can extract the relationship between features velocity and camera velocity, which is shown as follows:

$$\dot{x}_c = a_{11}v_x + a_{12}v_y + a_{13}v_y + a_{14}w_x + a_{15}w_y + a_{16}w_z$$
$$\dot{y}_c = a_{21}v_x + a_{22}v_y + a_{23}v_y + a_{24}w_x + a_{25}w_y + a_{26}w_z$$
$$\dot{P}_z = a_{31}v_x + a_{32}v_y + a_{33}v_y + a_{34}w_x + a_{35}w_y + a_{36}w_z$$
$$\dot{\theta}_x = a_{44}w_x + a_{45}w_y + a_{46}w_z$$
$$\dot{\theta}_y = a_{54}w_x + a_{55}w_y + a_{56}w_z$$
$$\dot{\theta}_z = a_{64}w_x + a_{65}w_y + a_{66}w_z \quad (9)$$

where a_{ij} for $i = 1$–6 and $j = 1$–6 are given in the Appendix. The last three features only depend on the rotational velocities. This provides the flexibility to deal with the last three features in orientation planning. Furthermore, since no rotational velocity commands will be sent in the position planning part of the algorithm, w_x, w_y, and w_z could be substituted with zero values. Therefore, we have

$$\dot{\mathbf{s}}_n = \mathbf{A}^{\mathbf{c}}\mathbf{V}_{\mathbf{c}} \quad (10)$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_p & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{A}_o \end{bmatrix} \quad (11)$$

and $\mathbf{A}_p$ is a 3×3 matrix including a_{ij} from (9), where $i = 1, 2, 3$ and $j = 1, 2, 3$, and $\mathbf{A}_o$ is a 3×3 matrix including a_{ij} in (9), where $i = 4, 5, 6$ and $j = 4, 5, 6$.

B. Decoupling Orientation Planning From Position Planning

In order to avoid the highly nonlinear and coupled relationship between features and camera velocities, it is proposed to decouple the orientation planning from position planning. In addition, the newly defined six features can facilitate and speed up the optimization process. A decoupled visual servoing controller is presented in [25] without considering the constraints. In this paper, an offline decoupled trajectory planning respecting the system's constraints is investigated for the visual servoing.

The decoupling process is performed as follows. First, the last three velocity screw elements are planned in the optimization process so that they take the last three elements of the feature set $(\theta_x, \theta_y, \theta_z)$ to their desired values. Second, the first three elements of the velocity screw are planned to eliminate the error existing in the first three elements of the feature set (x_c, y_c, p_z). The last three joints of the robot are responsible for performing the orientation and the first three joints of the robot are responsible for positioning. Using the above-defined six features and decoupling, the planning process creates a fast convergent optimization process.

C. Parameterizing the Velocity Profile

A predefined velocity profile is selected and named as $\mathbf{V}_t(t)$, which is a vector with six elements. In a visual servoing task that deals with a stationary object, the robot starts from stationary situation and ends in a stationary situation. Thus, the selected profile needs to satisfy the following conditions:

$$\mathbf{V}_t(0) = 0$$
$$\mathbf{V}_t(t_f) = 0 \quad (12)$$

where t_f is the final time when we plan to have the robot stop at the target position. Some examples of these functions could be a trapezoid function, a polynomial function or half cycle of a sinusoidal wave. However, more complicated trajectories with more parameters could be used such as a higher order polynomial, especially for the cases where other objective functions such as energy or path length are used for optimization.

In this paper, a half cycle of a sinusoidal profile is used to parameterize the velocity profile. The velocity profile could be shown as follows:

$$\mathbf{V}_t(t) = \mathbf{v}_m \sin\left(\frac{\pi t}{t_f}\right), \qquad 0 \le t \le t_f \quad (13)$$

where $\mathbf{v}_m$ is the vector of maximum speed that the camera reaches within the profile and is given as

$$\mathbf{v}_m = \begin{bmatrix} v_{\mathrm{mx}} & v_{\mathrm{my}} & v_{\mathrm{mz}} & v_{m\omega_x} & v_{m\omega_y} & v_{m\omega_z} \end{bmatrix}^{\mathrm{T}} \quad (14)$$

where $v_{\mathrm{mx}}, v_{\mathrm{my}}, v_{\mathrm{mz}}, v_{m\omega_x}, v_{m\omega_y}$, and $v_{m\omega_z}$ are the maximum velocity of each element in the velocity screw, respectively. The final time t_f is selected by the user depending on the desired speed of the task. Thus, each profile has only one parameter to be designed and the overall number of design parameters of the profile is six. The velocity profile $\mathbf{V}_t(t)$ can be used as the camera's velocity screw ${}^c\mathbf{V}_c$ represented in the camera frame.

Substituting (13) in (10) and integrating it over time, we could have the location of the feature at time t_f as a function of optimization parameters in the following form:

$$\mathbf{s}_n(t) = \left(\int_{t_0}^{t}\left(\mathbf{A}\sin\left(\frac{\pi t}{t_f}\right)\right)dt\right)\mathbf{v}_{\mathbf{m}} + \mathbf{s}_{n0} \quad (15)$$

where $\mathbf{s}_{n0}$ is the feature values at time t_0.

D. Depth Estimation

To accurately calculate the location of the feature points, the distance between the object and the camera is required. The motion of the camera in the Z_c-direction is known from v_z element of the velocity screw which is given as a parameterized equation of time $(v_{\mathrm{tp}_z} = f_z(t))$. Thus, the depth Z can be calculated at

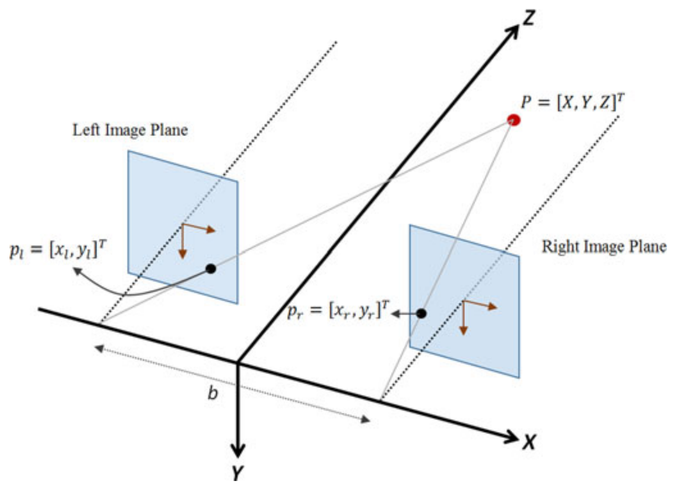


Fig. 5. Stereo camera model.

any time t from

$$Z_t = \int_{t_0}^{t} f_z(t)dt + Z_0 \tag{16}$$

where Z_0 is the initial depth of the object with respect to the camera coordinates. If the initial depth of the object is estimated accurately, the depth in the rest of the times could be calculated. Let us recall that using a stereo camera, the depth of the object could be calculated [26]–[28]. This is done by applying the epipolar geometry constraint that exist between the features in images planes of each camera. In a simple case where the two cameras are mounted parallel to each other (see Fig. 5), the depth of the object with respect to the cameras can be calculated using the disparity of the images using

$$Z_c = \frac{b}{x_l - x_r} \tag{17}$$

where Z_c is the depth of the object in the camera coordinates, x_r and x_l are the features of x coordinates in left and right cameras, respectively, and b is the distance between the cameras. We can conclude that by having two images of an object from a camera from which the second image is taken at a location with a displacement of b along X_c from the first location of the camera, the same equation could be used to calculate the object depth. Thus, by moving the camera along X_c by a small displacement b and using the initial and the final image feature positions, the depth of the object could be calculated from (17). This procedure takes about 1 s to complete, which is feasible in experiment.

IV. Constraints

One of the main issues in conventional visual servoing is that it does not limit the robot within the system constraints. In addition, by just limiting the system within the constraints, the convergence of the system to the target point cannot be guaranteed. The highly coupled nature of the visual servoing system could cause the control law to take the robot toward and beyond its boundaries, while IBVS is attempting to fix the camera's orientation. This can be easily observed in a visual servoing task using a conventional controller. Thus, limiting the system motion by using advanced control law, such as model predictive controller, is not sufficient to stabilize the system [29]. On the other hand, in a trajectory planning algorithm, the generated trajectory could be examined beforehand to guarantee that the robot reaches the target while respecting the constraints. Two main constraints are considered in this paper. The first constraint is associated with the robot's workspace. The second constraint is the robot joint limits. These constraints are discussed in details in the following sections.

It is good to note that limiting the system to keep the features inside the FOV of the camera is vital to the success of the task in a traditional IBVS. The proposed method integrates the equation of motion and predicts the features position at different time moments. Thus, it only requires the initial and the final positions of the features. Consequently, limiting the features inside the FOV is not necessary in this method.

A. Workspace Constraint

The planned trajectory is feasible only if it is inside the robot workspace at all times. Every robot has its own workspace. The typical workspace of a serial manipulator is a part of sphere with the radius equal to the length of the arms when they are aligned in the same direction. The workspace constraints $\mathbf{C}_W$ could be formulated in a polar system as follows:

$$\begin{aligned} X_c &= R_c \cos(\theta_c)\cos(\alpha_c), & 0 < R_c \le R_{c_{\max}} \\ Y_c &= R_c \cos(\theta_c)\sin(\alpha_c), & \theta_{c_{\min}} < \theta_c \le \theta_{c_{\max}}, \\ \text{and} \quad Z_c &= R_c \sin(\theta_c), & \alpha_{c_{\min}} < \alpha_c \le \alpha_{c_{\max}} \end{aligned} \tag{18}$$

where $\mathbf{P}_c = [X_c, Y_c, Z_c]^{\mathrm{T}}$ and $\mathbf{P}_{p_c} = [R_c, \theta_c, \alpha_c]^{\mathrm{T}}$ are the cameras coordinates in Cartesian and polar systems, respectively; $R_{c_{\max}}$ is the maximum possible length of the robot's arm; $\theta_{c_{\min}}$ and $\theta_{c_{\max}}$ are the minimum and maximum angles of the robot's arm about its base X-axis, respectively; and $\alpha_{c_{\min}}$ and $\alpha_{c_{\max}}$ are the minimum and maximum angles of the robot's arm about its base Z-axis, respectively.

B. Joints Space Constraint

Keeping the robot inside the workspace is not enough to accomplish a visual servoing task. In addition to workspace constraint, it is necessary to make sure the robot respects its joint limits and does not collide with itself. The joints space constraints $\mathbf{C}_J$ can be formulated as follows:

$$\mathbf{q}_{\min} \le \mathbf{q} \le \mathbf{q}_{\max} \tag{19}$$

where $\mathbf{q}$ is the robot joint vector and $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ are defined as the robot's joint limits. The end-effector position is known at all time during the servoing. A function is required to transform the robot's end-effector coordinates to robot joints' ones. This function is the inverse kinematic of the robot. The constraint $\mathbf{C}_J$ could be written as

$$\mathbf{q}_{\min} \le \mathbf{I}(\mathbf{P_c}) \le \mathbf{q}_{\max} \tag{20}$$

where $\mathbf{I}(\mathbf{P}_c)$ is the inverse kinematic function of the robot.

In order to have a completely convex optimization problem, the constraints should also be convex functions. The convexity of these functions are investigated in [12].

V. Optimization and Convexity Analysis

Let us define the objective function as the quadratic form of the selected features error, given by

$$\mathrm{OF} = (\mathbf{s}_n(t_f) - \mathbf{s}_{\mathrm{nd}})^{\mathrm{T}}\mathbf{Q}(\mathbf{s}_n(t_f) - \mathbf{s}_{\mathrm{nd}}) \tag{21}$$

where $\mathbf{s}_{\mathrm{nd}}$ is the desired image features, $\mathbf{Q}$ is a positive definite orthogonal matrix introducing the desired weight of each error in the optimization process. Therefore, considering the (15) the objective function could be written as

$$\begin{aligned}\underset{\mathbf{v_m}\in\mathbb{R}^6}{\mathrm{OF}(\mathbf{v_m})} = {} & \left(\left(\int_{t_0}^{t_f}\left(\mathbf{A}\sin\left(\frac{\pi t}{t_f}\right)\right)dt\right)\mathbf{v_m} + \mathbf{s}_{n0} - \mathbf{s}_{\mathrm{nd}}\right)^{\mathrm{T}} \\ & \times\mathbf{Q}\left(\left(\int_{t_0}^{t_f}\left(\mathbf{A}\sin\left(\frac{\pi t}{t_f}\right)\right)dt\right)\mathbf{v_m} + \mathbf{s}_{n0} - \mathbf{s}_{\mathrm{nd}}\right)\end{aligned} \tag{22}$$

where $\mathbf{v_m}$ is the optimization parameter vector. An important point that needs to be considered is that the trajectory planning procedure must be completed in a reasonable time and converge to the global minimum. Otherwise, the method would be useless for real-world applications because of the delay that is imposed to the system even if the planning is carried out offline. One important factor that leads to the global minimum is the convexity of the optimization problem. In this section, the convexity of the problem is investigated. To start, let us review the following main theorems regarding the convexity of a problem.

Theorem 1: If $f(x^*)$ is a local minimum for a convex function $f(x)$ defined on a convex feasible set S, it is also a global minimum [30].

Theorem 2: A function of n variables $f(x_1, x_2, \ldots, x_n)$ defined on a convex set S is convex if and only if the Hessian matrix of the function is positive semidefinite or positive definite at all points in the set S [30].

Let $\Delta s_n = s_n(t_f) - s_{\mathrm{nd}}$. The quadratic objective function $f(\Delta s_n) = \Delta s_n^T Q s_n$ is convex, since the Hessian matrix Q is positive definite. One applies linear function $B(t) = \int_{t_0}^{t_f}(A\ \sin(\frac{\pi t}{t_f}))dt$ and a constant $b = s_{n0} - s_{\mathrm{nd}}$ operation to the variables $\mathbf{v_m} \in \mathbb{R}^6$, i.e., $\Delta s_n = B(t)\mathbf{v_m} + b$ and the objective function $f(B(t)\mathbf{v_m} + b) = f(\Delta s_n)$ is convex. The linear operation is an affine mapping that can preserve convexity [31]. Let $\mathrm{OF}(\mathbf{v_m}) = f(B(t)\mathbf{v_m} + b)$. According to the composition with an affine mapping [31], $f(B(t)\mathbf{v_m} + b)$ is convex, so is $\mathrm{OF}(\mathbf{v_m})$. The objective function $\mathrm{OF}(\mathbf{v_m})$ (22) is convex on the convex set S, and hence the optimization is convex over set S.

The convexity of the objective function given in (22) can be validated and demonstrated through the numerical method introduced by Chinneck [32]. Accordingly, a code is generated to numerically calculate the Hessian matrix [33] of the objective function for a desired span of the optimization parameters. The design parameter's range depends on the physical limitations of the robot. In this case, the design parameters are the maximum velocities of the end effector in the corresponding DOF. Knowing the speed limits of the robotic system, this could be identified. In our test, the following ranges have been used:

$$\begin{aligned} -0.1 \le{} & v_{\mathrm{mx}} \le 0.1 \quad (\mathrm{m/s}) \\ -0.1 \le{} & v_{\mathrm{my}} \le 0.1 \quad (\mathrm{m/s}) \\ -0.1 \le{} & v_{\mathrm{mz}} \le 0.1 \quad (\mathrm{m/s}) \\ -0.1 \le{} & v_{m\omega_x} \le 0.1 \quad (\mathrm{rad/s}) \\ -0.1 \le{} & v_{m\omega_y} \le 0.1 \quad (\mathrm{rad/s}) \\ -0.3 \le{} & v_{m\omega_z} \le 0.3 \quad (\mathrm{rad/s}). \end{aligned} \tag{23}$$

To find the optimal velocity profile for the orientation, we purposely set the first three maximum velocities as zeros and set the last three maximum velocities as the decision variables, i.e., $\mathbf{v}_m = \begin{bmatrix} 0\ 0\ 0\ v_{m\omega_x}\ v_{m\omega_y}\ v_{m\omega_z} \end{bmatrix}^{\mathrm{T}}$. The optimization problem at the first part of stage 2 is posed as

$$\begin{aligned} & \underset{\mathbf{v_p}}{\mathbf{Min}}\ \mathrm{OF}(\mathbf{v_p}) \\ & \text{subject to inequalities } (23), \mathbf{C}_W, \text{ and } \mathbf{C}_J \end{aligned} \tag{24}$$

where $\mathbf{v_p} = \begin{bmatrix} v_{m\omega_x}\ v_{m\omega_y}\ v_{m\omega_z} \end{bmatrix}$.

Similarly, the velocity profile for the positioning is obtained by setting the last three maximum velocities as zeros and the first three maximum velocities as the decision variables, i.e., $\mathbf{v}_m = \begin{bmatrix} v_{\mathrm{mx}} & v_{\mathrm{my}} & v_{\mathrm{mz}} & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. The optimization problem at the second part of stage 2 is posed as

$$\begin{aligned} & \underset{\mathbf{v_o}}{\mathbf{Min}}\ \mathrm{OF}(\mathbf{v_o}) \\ & \text{subject to inequalities } (23),\ \mathbf{C}_W, \text{ and } t\mathbf{C}_J \end{aligned} \tag{25}$$

where $\mathbf{v_o} = \begin{bmatrix} v_{\mathrm{mx}}\ v_{\mathrm{my}}\ v_{\mathrm{mz}} \end{bmatrix}$.

To demonstrate the results of this investigation, without loss of generality, we choose the initial and desired locations such that the robot needs a motion in all 6 DOFs to reach the desired position. The final time t_f is selected as 10 s. The changes to the objective function of different values of the design parameters are shown in Fig. 6. To be able to show these variations in a 3-D plot format, the variations of the objective function with the changes of two parameters are shown in each figure. All available combinations are presented in the figures. The variations of the objective function with the changes in v_{mx} and v_{my} are shown in Fig. 6(a). The variations of the objective function with the changes in $v_{\mathrm{mx}} - v_{\mathrm{mz}}$ are shown in Fig. 6(b). To check the convexity of the system due to the angular motions, the changes in the objective function is introduced due to the changes in ω_{mx} and ω_{my} and the changes in ω_{mx} and ω_{mz}. These changes are shown in Fig. 6(c) and (d). Utilization of a numerical approach can validate the convexity analysis of the objective function (22).

VI. Experimental Results

In this section, the results of the experimental tests of the proposed algorithm are presented. The experimental setup consists of a VS-6556G Denso robot and a camera mounted on the end effectors (see Fig. 7). A cubic shape object is used as the target object. Four corners of the top plane of the object are used as

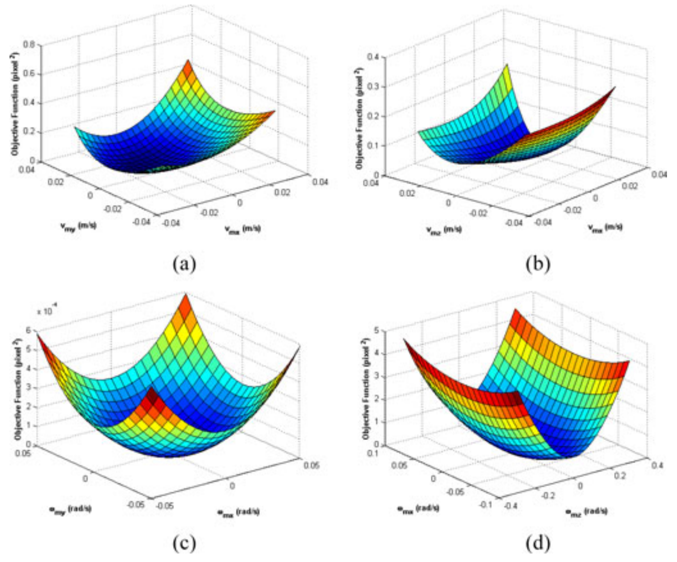


Fig. 6. Objective function due to different parameters' changes. (a) OF$_n$ versus v_{mx} and v_{my}. (b) OF$_n$ versus v_{mx} and v_{mz}. (c) OF$_n$ versus ω_{mx} and ω_{my}. (d) OF$_n$ versus ω_{mx} and ω_{mz}.

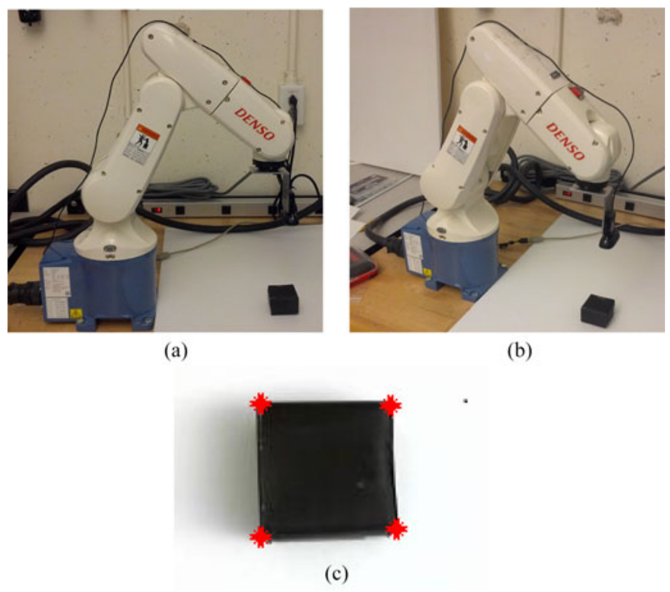


Fig. 7. Experimental setup and object. (a) Denso robot. (b) Denso robot. (c) Image of the cubic object taken by the camera on the robot.

the features. Harris algorithm is used to extract the cube corners [34]. Fig. 7(c) shows the picture of the cube taken by the camera and the extracted features.

The robot has 6 DOFs and it communicates with its controller with a frequency of 1 kHz. A CCD camera is used as the vision system, mounted on the robot's end effector. The camera intrinsic parameters are given in Table I. The camera's capturing rate is 30 fps. The object is stationary in the workspace. The visual servoing task is finished when the image features match the desired features. Each complete test consists of four stages. First, the depth-estimation algorithm moves the end-effector in X_c-direction by 5 cm to take the stereoscopic image and estimates the depth of the object. Second, using the current image

TABLE I
CAMERA PARAMETERS

Parameters	Values
Focal length	0.004 (m)
X-axis scaling factor	110 000 (pixel/m)
Y-axis scaling factor	110 000 (pixel/m)
Image plane offset of *X*-axis	120 (pixel)
Image plane offset of *Y*-axis	187 (pixel)

```
Algorithm: Trajectory Planning Procedure

1: Move the camera in X direction to estimate the depth of the features. (Stage 1)
2: Run optimization to find best camera velocity constants to remove the orientation error. This
   will be done by eliminating the error of the last three features. (Stage 2)
3: Run optimization to find best camera velocity constants to remove the position error. This will
   be done by eliminating the error of the first three features. (Stage 2)
4: Apply the velocity profile found in step 3. (Stage 3)
5: Apply the velocity profile found in step 4. (Stage 3)
6: If there is an error between the desired features and the current features, apply the velocity
   gained from the controllerto eliminate the system error. (Stage 4)
```

Fig. 8. Pseudocode for the trajectory planning.

features, desired image features and the initial depth of the object, the trajectory planning algorithm generates the appropriate angular velocity through optimization to reorient the camera to a parallel plane as the object plane. This is done by matching the three last selected features. After that, the positioning trajectory is generated by matching the first three selected features. Due to the nonlinearity of the selected objective function, an interior point algorithm [35] is used to solve the optimization problem. At the third stage, the generated velocity is applied to the robot so that it is taken to the desired position. At the fourth stage, an AIBVS [11] controller is executed to compensate for any differences between the image features and the desired image features caused by the uncertainties in the system model. As it is shown in the results, most of the tests may not require the last stage, since the trajectory planning is done by matching the features with the desired ones. The pseudocode of the trajectory planning procedure is provided in Fig. 8. Four different tests with different strategies have been performed to ensure the algorithms validity.

Test 1: In the first test, our aim is to show the performance of the system on carrying out a relatively simple visual servoing task. The initial and desired locations of the features are given in Table II.

The trajectory planning algorithm generates the velocity profiles shown in Fig. 9(e). Applying the velocities to the robot, the robot is taken to the desired position. The first sine cycle is related to the orientation planning and the second part is related to the positioning. The features trajectory in image space and the camera trajectory in 3-D space are shown in Fig. 9(c) and (d). The half sphere in this figure shows the workspace of the robot. The robot joint angles during the robot motion are shown in Fig. 9(f). Since, the system model is sufficiently accurate, the desired position is reached using the velocity profiles and the fourth stage of the algorithm is not required for this test. At the first stage of the algorithm, the robot moves the camera by 10 cm in the X_c-direction and the depth estimation is 0.4 m.

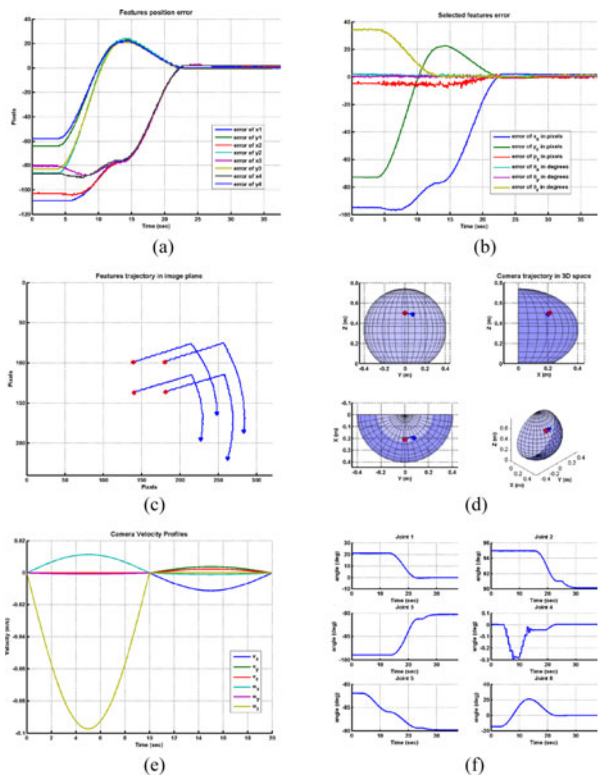


Fig. 9. Results for Test 1. (a) Features position error. (b) Selected features error. (c) Feature trajectory in image plane. (d) Camera 3-D trajectory. (e) Generated velocity profile. (f) Robot joint angles.

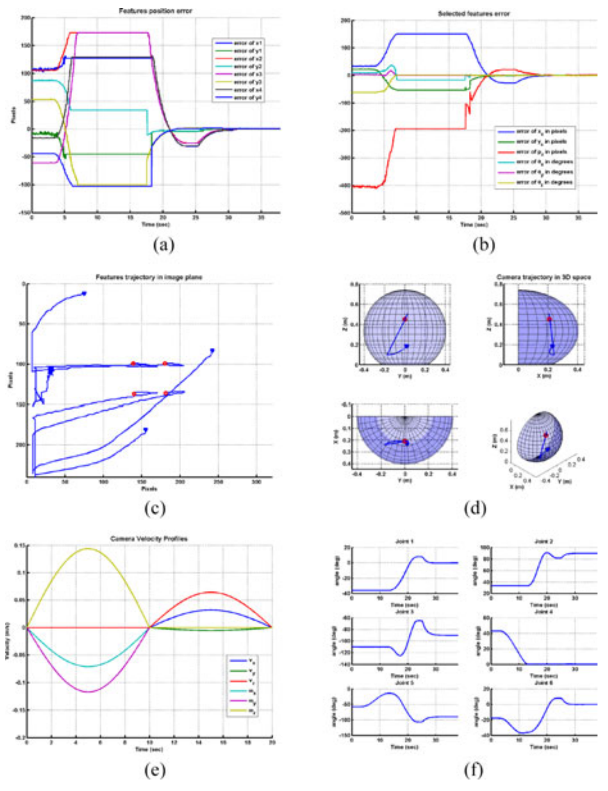


Fig. 10. Results for Test 2. (a) Features position error. (b) Selected features error. (c) Feature trajectory in image plane. (d) Camera 3-D trajectory. (e) Generated velocity profile. (f) Robot joint angles.

TABLE II
INITIAL (I) AND DESIRED (D) LOCATION OF FEATURE POINTS IN PIXEL

		Point1		Point2		Point3		Point4	
		(x	y)	(x	y)	(x	y)	(x	y)
Test 1	I	248	163	283	185	262	219	227	195
	D	138	99	179	99	179	136	138	137
Test 2	I	32	106	75	12	242	83	155	181
	D	139	100	179	98	180	135	139	136
Test 3	I	137	99	178	99	179	136	138	137
	D	190	154	129	154	128	98	190	98
Test 4	I	107	210	16	206	26	133	114	137
	D	291	212	203	229	187	154	276	136

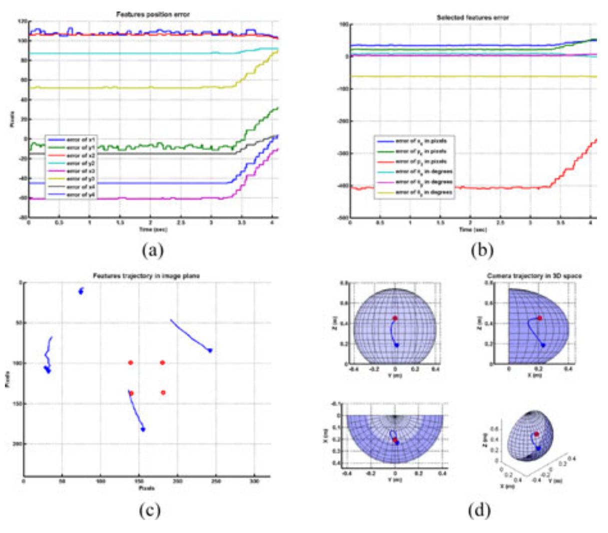


Fig. 11. Results for Test 2 for IBVS. (a) Features position error. (b) Selected features error. (c) Feature trajectory in image plane. (d) Camera 3-D trajectory.

The optimization process in this test takes less than a second to complete using a Intel Xeon E31220 3.10 GHz CPU.

Test 2: In the second test, some advantages of the proposed method over IBVS controller are shown. A relatively complicated task is chosen for this matter. The initial and final positions of the robot are given in Table II. The results of this test are given in Fig. 10.

The optimization process creates the velocity profile given in Fig. 10(e). The first part of the velocity profile is to orient the camera to be parallel to the object's feature plane. These velocity profiles only moves the last three joints. This causes the features to move out of the FOV. However, since this algorithm is carried out offline, it only needs the initial and desired locations of the features. During the visual servoing, it is assumed that the camera FOV is unlimited. The features eventually return to the real FOV of the camera as the robot completes the created path. The constant lines in the feature error and the selected features error in Figs. 10(a) and 9(b) are related to the time moment when the features are out of camera's FOV. It is shown that the task is completed keeping the robot in its workspace. The joint angles are also shown in Fig. 10(f). The same task is done using an IBVS controller. The results are given in Fig. 11.

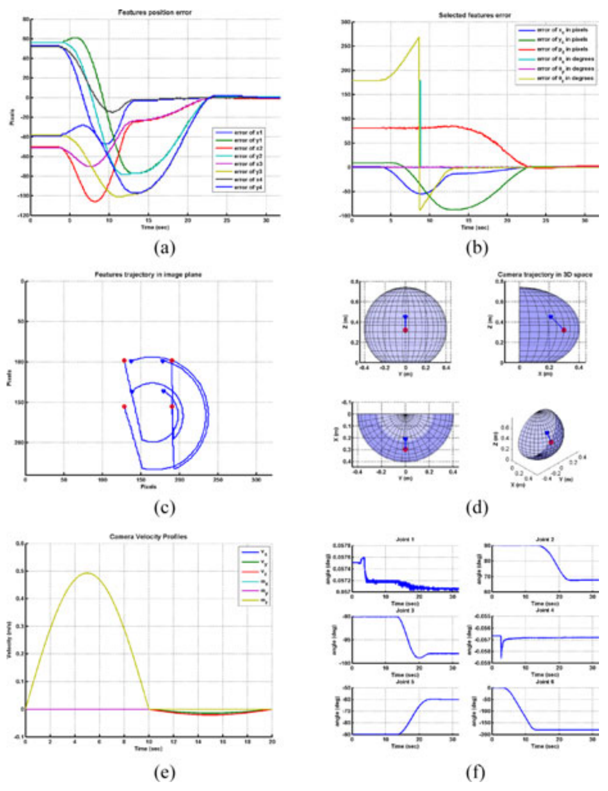


Fig. 12. Results for Test 3. (a) Features position error. (b) Selected features error. (c) Feature trajectory in image plane (d) Camera 3-D trajectory. (e) Generated velocity profile. (f) Robot joint angles.

As shown in Fig. 11(c), the rotation required for this task takes the features out of the FOV. The IBVS controller needs the features' position at each instant. As soon as the features run out of the FOV, the controller has false data from the features position, which causes the task to fail.

Test 3: In the third test, another common problem of IBVS is investigated using the proposed method. The visual servoing fails when a 180° rotation of the camera is required to reach its desired position [5]. The test includes a 180° rotation in the end-effector motion. The initial and desired locations of the feature points are given in Table II. The results of this test are shown in Fig. 12.

The same test is conducted using IBVS controllers. The results are shown in Fig. 13. The results show that, similar to the previous test, the IBVS controller tries to match the features through the shortest path available which results in a motion of camera in the Z_c-direction. This continues until the end effector reaches its physical limits and the robot stops, as shown in Fig. 13(d).

Test 4: Another challenge in conventional visual servoing is the local minima problem. In an IBVS controller, the interaction matrix is an 8×6 matrix. The inverse of this matrix, which is used to produce the control law, is an 8×6 matrix and has two vector of null space. If the features error vector is a factor of these null space vectors, the controller generates a zero-velocity vector as the control command. This causes the system to get stuck in that spot. In the trajectory planning algorithm, the inverse of the interaction matrix is not used. Consequently,

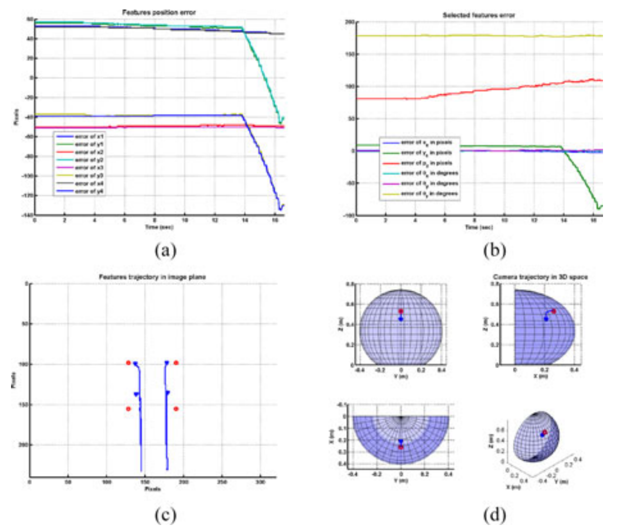


Fig. 13. Results for Test 2 for IBVS. (a) Features position error. (b) Selected features error. (c) Feature trajectory in image plane. (d) Camera 3-D trajectory.

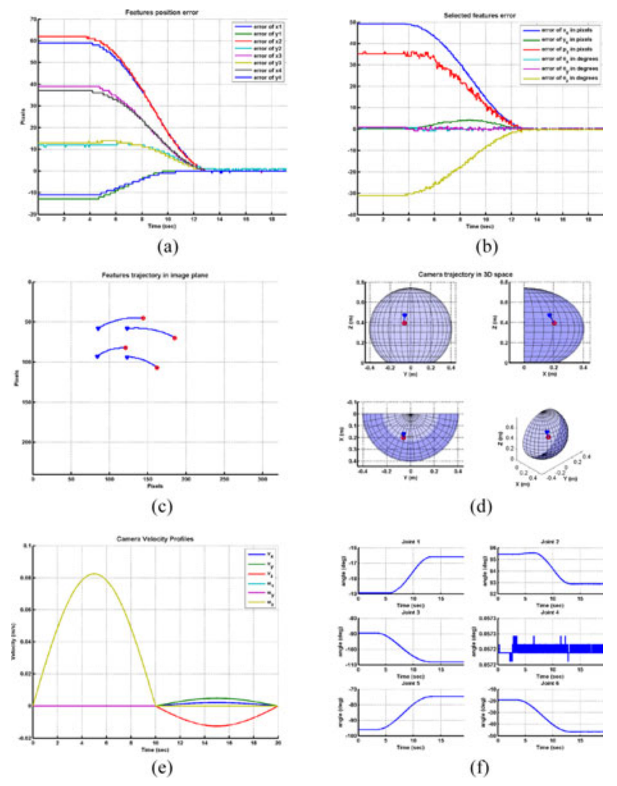


Fig. 14. Results for Test 4. (a) Features position error. (b) Selected features error. (c) Feature trajectory in image plane. (d) Camera 3-D trajectory. (e) Generated velocity profile. (f) Robot joint angles.

the local minima problem is solved. The next test demonstrates this ability in the proposed algorithm. The initial and desired locations of the feature points are given in Table II. The desired features are chosen, so that the vector of feature position error is in the null space of the interaction matrix. The results are shown in Fig. 14. We can see that the proposed algorithm produces a velocity profile to take the robot to the desired position, while the IBVS controller produces a zero velocity vector.

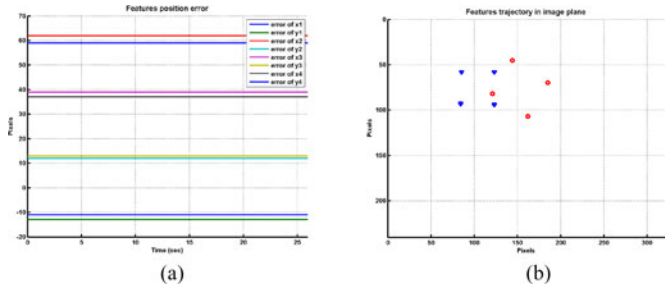


Fig. 15. Results for Test 4 using IBVS controller. (a) Features position error. (b) Selected features error.

The previous four tests demonstrate that the proposed trajectory planning algorithm provides an optimized predefined path for visual servoing. The combination of the trajectory planning with the AIBVS allows the robot to fulfill some tasks where the traditional IBVS fails, such as features leaving out of view, 180° rotation of the camera's center and having local minima problems, etc. The proposed trajectory planning algorithm can also enlarge the system operation range and is not subjected to FOV constraints, since it only needs the initial and desired features.

Conclusion

In this paper, a novel visual servoing technique is proposed. This technique is performed by planning a trajectory from the initial robot's position to a target position where the image features match the desired ones. The trajectory is based on an optimized predefined path that satisfies the system's initial and final conditions. The trajectory parameters are obtained through an optimization procedure by minimizing the error between the image features and the desired ones. In order to speed up the optimization process, six new features are introduced. Moreover, the planning procedure is decoupled to two stages of orientation planning and position planning. A depth-estimation method is proposed to provide the object depth to the trajectory planning algorithm. After performing the velocity profile generated from the trajectory planning algorithm, a visual servoing controller is used to compensate for the errors appeared in matching the features with the desired ones. The experimental tests demonstrate the proposed method exhibits its advantages over IBVS controllers in succeeding some visual servoing tasks where the IBVS controller fails.

Appendix

Eq. (9) presents the relationship between features velocity and camera velocity. In this equation a_{ij} $i = 1$–6 and $j = 1$–6 could be found as follows:

$$\begin{aligned}
a_{11} &= \frac{-1}{Z}\\
a_{12} &= 0\\
a_{13} &= \frac{x_1+x_2+x_3+x_4}{4Z}\\
a_{14} &= \frac{x_1y_1+x_2y_2+x_3y_3+x_4y_4}{4}\\
a_{15} &= -\frac{x_1^2+x_2^2+x_3^2+x_4^2+4}{4}\\
a_{16} &= \frac{y_1+y_2+y_3+y_4}{4}\\
a_{21} &= 0\\
a_{22} &= \frac{-1}{Z}\\
a_{23} &= \frac{y_1+y_2+y_3+y_4}{4Z}\\
a_{24} &= \frac{y_1^2+y_2^2+y_3^2+y_4^2+4}{4}\\
a_{25} &= \frac{x_1y_1+x_2y_2+x_3y_3+x_4y_4}{4}\\
a_{26} &= -\frac{x_1+x_2+x_3+x_4}{4}\\
a_{31} &= 0\\
a_{32} &= 0\\
a_{33} &= \frac{1}{Z}\Big(\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}+\sqrt{(x_1-x_4)^2+(y_1-y_4)^2}\\
&\quad+\sqrt{(x_2-x_3)^2+(y_2-y_3)^2}+\sqrt{(x_3-x_4)^2+(y_3-y_4)^2}\Big)\\
a_{44} &= \frac{(x_1y_2-x_2y_1)(y_1-y_2)}{2(x_1^2-2x_1x_2+x_2^2+y_1^2-2y_1y_2+y_2^2)}\\
&\quad-\frac{(x_3y_4-x_4y_3)(y_3-y_4)}{2(x_3^2-2x_3x_4+x_4^2+y_3^2-2y_3y_4+y_4^2)}\\
a_{45} &= -(-x_3^2y_4+y_2x_3^2+x_3x_4y_3+x_3x_4y_4-2y_2x_3x_4\\
&\quad-x_4^2y_3+y_2x_4^2+y_2y_3^2-2y_2y_3y_4+y_2y_4^2)/(2((x_3-x_4)^2\\
&\quad+(y_3-y_4)^2))+(y_1-y_2)(-x_2^2+x_1x_2-y_2^2+y_1y_2)\\
&\quad/\,(2((x_1-x_2)^2+(y_1-y_2)^2))\\
a_{46} &= 0\\
a_{54} &= \frac{(x_2y_3-x_3y_2)(y_2-y_3)}{2((x_2-x_3)^2+(y_2-y_3)^2)}-\frac{(x_1y_4-x_4y_1)(y_1-y_4)}{2((x_1-x_4)^2+(y_1-y_4)^2)}\\
a_{55} &= (-x_2^2y_3+y_4x_2^2+x_2x_3y_2+x_2x_3y_3-2y_4x_2x_3\\
&\quad-x_3^2y_2+y_4x_3^2+y_4y_2^2-2y_4y_2y_3+y_4y_3^2)/(2((x_2-x_3)^2\\
&\quad+(y_2-y_3)^2))-((y_1-y_4)(-x_4^2+x_1x_4-y_4^2+y_1y_4))/\\
&\quad(2((x_1-x_4)^2+(y_1-y_4)^2))\\
a_{56} &= 0\\
a_{64} &= \frac{x_1y_4^2+x_4y_1y_4+x_4y_1^2+x_1y_1y_4}{2((x_1-x_4)^2+(y_1-y_4)^2)}\\
&\quad+\frac{x_2y_3^2+x_3y_2^2-x_2y_2y_3-x_3y_2y_3}{2((x_2-x_3)^2+(y_2-y_3)^2)}\\
a_{65} &= \frac{x_1^2y_4+x_4^2y_1-x_1x_4y_4-x_1x_4y_1}{2((x_1-x_4)^2+(y_1-y_4)^2)}\\
&\quad+\frac{x_2^2y_3+x_3^2y_2-x_2x_3y_3-x_2x_3y_2}{2((x_2-x_3)^2+(y_2-y_3)^2)}\\
a_{66} &= 1. \qquad (26)
\end{aligned}$$

Acknowledgment

The authors would like to thank Quanser Company for supporting them with the experimental setup.

References

[1] F. Chaumette and S. Hutchinson, "Visual servo control. Part I. Basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

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

[3] K. Hashimoto, "A review on vision-based control of robot manipulators," *Adv. Robot.*, vol. 17, no. 10, pp. 969–991, 2003.

[4] F. Janabi-Sharifi, L. Deng, and W. J. Wilson, "Comparison of basic visual servoing methods," *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 5, pp. 967–983, Oct. 2011.

[5] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control (LNCIS Series no. 237)*. Berlin, Germany: Springer-Verlag, 1998, pp. 66–78.

[6] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 507–515, Aug. 2001.

[7] F. Chaumette, "Image moments: A general and useful set of features for visual servoing," *IEEE Trans. Robot.*, vol. 20, no. 4, pp. 713–723, Aug. 2004.

[8] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1116–1127, Dec. 2005.

[9] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 933–939, Oct. 2010.

[10] A. Hajiloo, M. Keshmiri, W.-F. Xie, and T. T. Wang, "Robust online model predictive control for a constrained image-based visual servoing," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2242–2250, Apr. 2016.

[11] M. Keshmiri, W. Xie, and A. Mohebbi, "Augmented image based visual servoing of a manipulator using acceleration command," *IEEE Trans. Ind. Electron.*, vol. 61, no. 9, Sep. 2014.

[12] G. Chesi, "Visual servoing path planning via homogeneous forms and LMI optimizations," *IEEE Trans. Robot.*, vol. 25, no. 2, pp. 281–291, Apr. 2009.

[13] G. Chesi and Y. Hung, "Global path-planning for constrained and optimal visual servoing," *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1050–1060, Oct. 2007.

[14] M. Keshmiri, M. Keshmiri, and A. Mohebbi, "Augmented online point to point trajectory planning, a new approach in catching a moving object by a manipulator," in *Proc. 2010 8th IEEE Int. Conf. Control Autom.*, 2010, pp. 1349–1354.

[15] M. Keshmiri and M. Keshmiri, "Performance comparison of various navigation guidance methods in interception of a moving object by a serial manipulator considering its kinematic and dynamic limits," in *Proc. 2010 15th Int. Conf. Methods Models Autom. Robot.*, 2010, pp. 212–217.

[16] M. Keshmiri and W. F. Xie, "Catching moving objects using a navigation guidance technique in a robotic visual servoing system," in *Proc. Amer. Control Conf.*, 2013, pp. 6302–6307.

[17] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 534–549, Aug. 2002.

[18] L. Deng, F. Janabi-Sharifi, and W. J. Wilson, "Hybrid motion control and planning strategies for visual servoing," *IEEE Trans. Ind. Electron.*, vol. 52, no. 4, pp. 1024–1040, Aug. 2005.

[19] S. Masoud and A. Masoud, "Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: A physical metaphor," *IEEE Trans. Syst., Man Cybern. A: Syst., Humans*, vol. 32, no. 6, pp. 705–723, Nov. 2002.

[20] A. Masoud and M. Bayoumi, "Intercepting a maneuvering target in a multidimensional stationary environment using a wave equation potential field strategy," in *Proc. 1994 IEEE Int. Symp. Intell. Control*, 1994, pp. 243–248.

[21] K. Hosoda, K. Sakamoto, and M. Asada, "Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3D reconstruction," in *Proc. 1995 IEEE/RSJ Int. Conf. Intell. Robot. Syst. Human Robot Interaction Cooperative Robots*, vol. 1, Aug. 1995, pp. 29–34.

[22] J. S. Park and M.-J. Chung, "Path planning with uncalibrated stereo rig for image-based visual servoing under large pose discrepancy," *IEEE Trans. Robot. Autom.*, vol. 19, no. 2, pp. 250–258, Apr. 2003.

[23] M. Kazemi, M. Mehrandezh, and K. Gupta, "Kinodynamic planning for visual servoing," in *Proc. 2011 IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 2478–2484.

[24] M. Kazemi, K. Gupta, and M. Mehrandezh, "Global path planning for robust visual servoing in complex environments," in *Proc. 2009 IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 326–332.

[25] W.-F. Xie, Z. Li, X.-W. Tu, and C. Perron, "Switching control of image-based visual servoing with laser pointer in robotic manufacturing systems," *IEEE Trans. Ind. Electron.*, vol. 56, no. 2, pp. 520–529, Feb. 2009.

[26] S. Zhang, C. Wang, and S. Chan, "A new high resolution depth map estimation system using stereo vision and kinect depth sensing," *J. Signal Process. Syst.*, vol. 79, no. 1, pp. 19–31, Apr. 2015.

[27] C. Strecha and L. Gool, "Motion – stereo integration for depth estimation," in *Proc. 7th Eur. Conf. Comput. Vis. (Lecture Notes in Computer Science)*. vol. 2351, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 170–185.

[28] B. Barrois and C. Wöhler, "3-D pose estimation of vehicles using stereo camera," in *Transportation Technologies for Sustainability*, M. Ehsani, F.-Y. Wang, and G. Brosch, Eds. New York, NY, USA: Springer-Verlag, 2013, pp. 1–24.

[29] G. Allibert, E. Courtial, and F. Chaumette, "Visual servoing via nonlinear predictive control," in *Visual Servoing Via Advanced Numerical Methods (Lecture Notes in Control and Information Sciences)*, vol. 401, G. Chesi and K. Hashimoto, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 375–393.

[30] J. Arora, *Introduction to Optimum Design*. Amsterdam, The Netherlands: Elsevier, 2004.

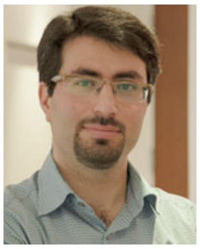
[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[32] J. W. Chinneck, "Discovering the characteristics of mathematical programs via sampling," *Optim. Methods Softw.*, vol. 17, no. 2, pp. 319–352, 2002.

[33] H. Scolnik and J. Gambini, "A new method to compute second derivatives," *J. Comput. Sci. Technol.*, vol. 2, no. 6, pp. 1–9, 2001.

[34] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Fourth Alvey Vis. Conf.*, 1988, pp. 147–151.

[35] R. Byrd, M. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, 1999.



Mohammad Keshmiri (S'12) received the B.Sc. and M.Sc. degrees in mechanical engineering from Isfahan University of Technology (IUT), Isfahan, Iran, in 2006 and 2009, respectively, and the Ph.D. degree in mechatronics from Concordia University in Montreal, QC, Canada, in 2014.

He was an active member of the Dynamic and Robotic Research Group, IUT, and has been involved in several projects of the group. He is currently a Postdoc Fellow in the Department of Computer Science, McGill University, Montreal, QC, Canada. He is also working as a Robotic Researcher in Robotmaster, Montreal, QC, Canada. His research interests include robotics and control, computer vision, nonlinear systems, visual servoing, system identification, and artificial intelligence.

Wen-Fang Xie (SM'12) received the Master's degree in automatic control engineering from Beihang University, Beijing, China, and the Ph.D. degree in electrical engineering from the Hong Kong Polytechnic University, Hong Kong, in 1991 and 1999, respectively.

She is a Professor in the Department of Mechanical and Industrial Engineering, Concordia University, Montreal, QC, Canada, where she joined as an Assistant Professor in 2003 and was promoted to Associate Professor and Professor in 2008 and 2014, respectively. Her research interests include nonlinear control and identification in mechatronics, visual servoing, model predictive control, neural network, advanced process control, and system identification.