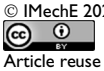
Original Article

Institution of MECHANICAL ENGINEERS
Journal of SYSTEMS AND CONTROL ENGINEERING

Proc IMechE Part I:
J Systems and Control Engineering
2022, Vol. 236(2) 338–354


Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/09596518211028379
journals.sagepub.com/home/pii
SAGE

Optimized hybrid decoupled visual servoing with supervised learning

Alireza Rastegarpanah[1,2*], Ali Aflakian[1,2*] and Rustam Stolkin[1,2]

Abstract

This study proposes an optimized hybrid visual servoing approach to overcome the imperfections of classical two-dimensional, three-dimensional and hybrid visual servoing methods. These imperfections are mostly convergence issues, non-optimized trajectories, expensive calculations and singularities. The proposed method provides more efficient optimized trajectories with shorter camera path for the robot than image-based and classical hybrid visual servoing methods. Moreover, it is less likely to lose the object from the camera field of view, and it is more robust to camera calibration than the classical position-based and hybrid visual servoing methods. The drawbacks in two-dimensional visual servoing are mostly related to the camera retreat and rotational motions. To tackle these drawbacks, rotations and translations in *Z*-axis have been separately controlled from three-dimensional estimation of the visual features. The pseudo-inverse of the proposed interaction matrix is approximated by a neuro-fuzzy neural network called local linear model tree. Using local linear model tree, the controller avoids the singularities and ill-conditioning of the proposed interaction matrix and makes it robust to image noises and camera parameters. The proposed method has been compared with classical image-based, position-based and hybrid visual servoing methods, both in simulation and in the real world using a 7-degree-of-freedom arm robot.

Keywords

Hybrid visual servoing, neuro-fuzzy neural network, optimized trajectory, local linear model tree, non-linear models

Date received: 30 November 2020; accepted: 4 June 2021

Introduction

In order to modify the behaviour of robots in dealing with unstructured environments, vision sensors are commonly used to provide contact-less information about the environment.[1] Real-time information from the camera image provides feedback to control the motion of a robot. This approach is called visual servoing (VS) which is an effective method for handling uncertainties in an unknown environment.

VS contributes to modify the system to compensate deficiencies of a mechanism and to relax the mechanical inaccuracy and stiffness of the robot.[2] This ability comes from the fact that the feature errors are regulated directly in the task space.[3] Despite this fact, how to use the image information to control the motion of a robot always been a major challenge in robotics. VS adds complexities in image space, joint space and the intersection between these two (task space) which should be considered.[4] VS control approaches are broadly classified into three categories: image-based visual servoing (IBVS), position-based visual servoing (PBVS) and hybrid visual servoing (HVS).

IBVS method computes the feedback directly from extracted features in the image space. This method is more robust to the camera calibration and kinematic errors of the robot.[5] Furthermore, the image features are less likely to be lost from the image screen.[6] However, there exist drawbacks for IBVS; first, some controller's commands are not physically executable as there is no direct control for Cartesian velocities of the robot's end-effector (EE).[7] Second, an interaction matrix (image-Jacobian) is required to map velocities from image space to velocities in the workspace of the robot. Therefore, the poor conditioning of the Jacobian matrix could cause convergence problems such as

[1]Department of Metallurgy and Materials Science, University of Birmingham, Birmingham, UK
[2]The Faraday Institution, Didcot, UK

*A.R. and A.A. are identified as the joint lead authors of this work.

Corresponding author:
Alireza Rastegarpanah, Department of Metallurgy and Materials Science, University of Birmingham, Birmingham B15 2TT, UK.
Email: a_r_adrex@yahoo.com

singularities and local minima.[8] Consequently, PBVS was proposed to address the imperfections of IBVS.[9]

In PBVS, feedback is computed using reconstructed Euclidean information to estimate the three-dimensional (3D) Cartesian pose of the target with respect to the camera pose. In PBVS method, interaction matrix problems (i.e. local minima and singularity) are avoided due to the direct calculation of the camera velocities from the task space errors; therefore, feasible trajectories for the robot could be generated.[10] However, any error in calibration of the camera could lead to an error in 3D estimation of the target, and subsequently, the entire tracking task. In addition, it is more possible to lose the features in the image screen, as the control feedback is generated from 3D estimation of the environment.[2]

HVS was proposed to benefit from the advantages of IBVS and PBVS while avoiding the drawbacks of these two.[11] In HVS methods like switching, 2 1/2D and homography-based VS, the task function uses image space information combined with the Cartesian space.[12] However, hybrid methods come with some performance deficiencies that will be discussed in detail in section 'Related works'.

Related works

Switching method is a type of HVS in which the controller switches between IBVS and PBVS with respect to (w.r.t.) their efficacy.[13] However, the controller suffers from discontinuities while switching occurs, especially when the object is close to the image borders.[14] Such discontinuities could be solved using sequencing methods;[15] nevertheless, the convergence time will increase.[2] Moreover, two failures in IBVS named camera retreat[8] and the Chaumette Conundrum could not be determined easily as image-Jacobian is not ill-conditioned in those configurations.[7] Therefore, switching between two methods (i.e. IBVS and PBVS) could not solve these kinds of complexities.[8] Even inducing rotational motions about the camera optic axis could not solve the Chaumette Conundrum, as rotational contributions cancel out one another.

Corke and Hutchinson[8] proposed a VS method that decouples the translation and rotation about Z-axis, from the image-Jacobian, in order to address the Chaumette Conundrum and the camera retreat. However, expensive computation of the pseudo-inverse is still challenging. Moreover, compensating the rotation errors about X-axis and Y-axis in the image-plane still produces unnecessary motion for the robot's joints which are not preferable.[13]

In 2 1/2D VS methods, the unnecessary motions are minimized by decomposing translations from the rotations.[16] But, these methods are computationally expensive too and require homography construction which is sensitive to image noise.[8] Another drawback of 2 1/2D VS method is its demand to have co-planar features for estimating the homography matrix. Otherwise, at least eight features are required for this estimation, while in other methods, four features are enough.[17] In addition, 2 1/2D VS decomposes homography to extract rotational parameters which come with non-unique solutions.[18]

Recently, learning-based approaches (e.g. recurrent neural network (RNN), convolution neural network (CNN), reinforcement learning (RL) and extreme learning machine (ELM)) are widely used to tackle vision tracking problems which are difficult or computationally expensive to solve by classical control methods, such as singularity avoidance, local minima or complex computation of pseudo-inverse.[19–21] Supervised learning approaches contribute to estimating the data output from previous experiences; therefore, the data set must be labelled in advance. However, unsupervised learning approaches find unknown patterns in a set of data.[22] Despite supervised learning approaches, it is not possible for regression applications to train the network without knowing the corresponding output values.[23] RNN has been used in Zhang and Li[24] to estimate the interaction matrix, but it requires a number of iterations to reduce the convergence speed. Hence, the network could not guarantee global optimization. Miljković et al.[21] proposed a controller that switches between neural RL and IBVS to solve the pseudo-inverse of the interaction matrix. However, huge chattering occurs in the computed camera velocities. Regression-based neural networks (NNs) have been considerably used to approximate the non-linear image-Jacobian.[25–28] They suffer from local minima which are complex to avoid.[29] It is worth mentioning that using NN in approximating the hybrid interaction matrices is not widely investigated in the literature.

To tackle the above-mentioned problems, we proposed an optimized VS method called hybrid decoupled visual servoing (HDVS). The contributions of our proposed method in comparison with other classical VS methods have been illustrated in Figure 1. In the proposed method, all three rotations and translation in the Z-axis have been decoupled from the image-Jacobian. These four components' errors will be regulated from the 3D reconstruction of the visual features. Consequently, the controller has independent control over translation in Z-axis and rotations. Thereafter, local linear model tree (LoLiMoT) NN has been used to approximate the pseudo-inverse of the proposed interaction matrix. Using LoLiMoT avoids singularities that could be happened in the interaction matrix, and reduces the computational complexities effectively. Accordingly, the controller becomes robust to the camera calibration errors and the image noises. LoLiMoT is a fast, effective neuro-fuzzy NN that learns a huge number of non-linear models.[30] LoLiMoT method guarantees global optimization solution which implies the generalization ability of the method.[30] Regression approaches are blind in detection of global minima, but LoLiMoT is axis orthogonal and operates by errors; thus, it will not stick in local minima.[31] The locality of this method provides online learning in one region without forgetting the other operating regions.[32]

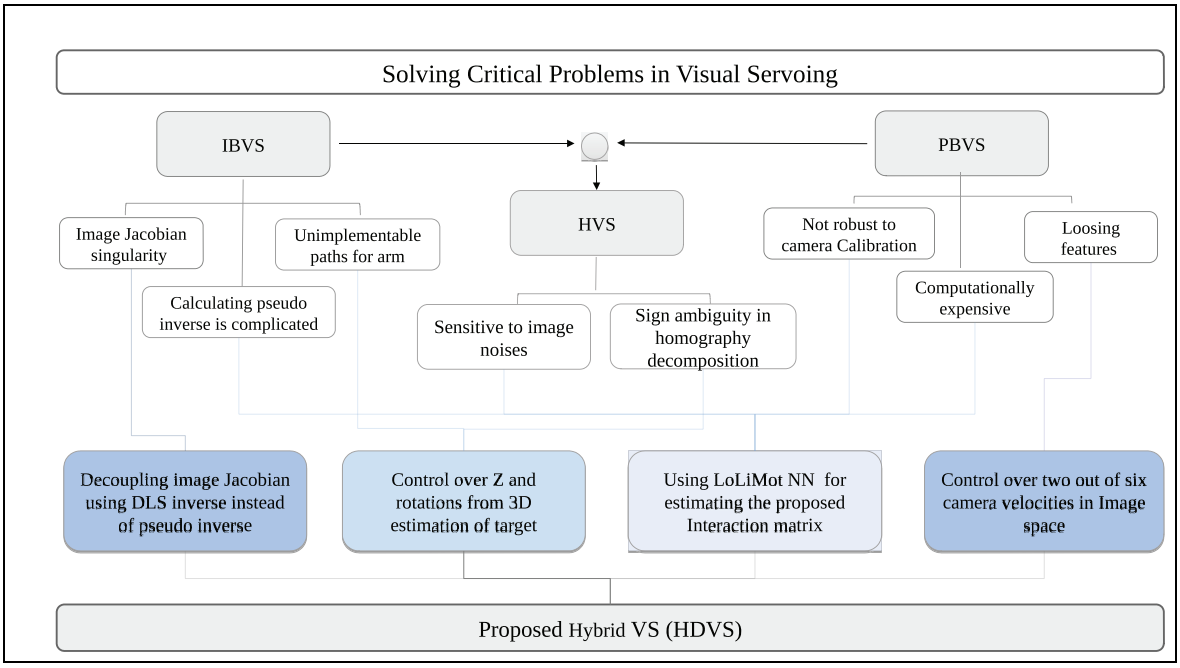


Figure 1. Problem domains in classical visual servoing and contributions of the proposed HDVS method.

Not to mention that the number of required trial-and-error steps will be reduced in LoLiMoT approach. The proposed method is producing a more optimized trajectory, both in the image space and the joint space than other hybrid methods, in terms of control effort and the convergence time. In comparison with IBVS, HDVS generates more controllable trajectories for the robot during tracking the objects. Furthermore, it is less likely to lose the object from the camera field of view, in contrast to PBVS and HVS methods. The proposed HDVS method is robust to camera calibration and image noises. In the ensuing, the method will be discussed in detail and its efficacy will be validated in both simulation and the real world.

Contributions of this study

Figure 1 depicts a framework for the proposed HDVS method, and it highlights the contributions of HDVS in comparison with other classical VS methods. The contribution of this study is summarized as follows:

- HDVS generates more optimized Cartesian trajectories (better controllability) for the robot than IBVS and HVS. All three rotations and translation in the Z-axis have been decoupled from the image-Jacobian. The errors of these four components have been regulated from 3D reconstruction of the visual features. Since the remained components (translation in X-axis and Y-axis) are controlled directly in the image-plane, then a more optimized trajectory in the image-plane would be created than PBVS. Therefore, the object is less likely to be lost from the camera field of view. This contribution has been explained explicitly in section 'Methodology'.
- HDVS benefits from damped-least square (DLS) inverse instead of pseudo-inverse to generate the joint velocities. Therefore, HDVS reduces the effect of robot's singularities and it assists to smooth the discontinuities created from decoupling process and adaptive gains. This contribution has been explained explicitly in section 'Methodology'.
- A set of LoLiMoT NNs has been trained in the presence of image noise to approximate the interaction matrix. Consequently, singularities of the interaction matrix would be avoided and computational complexities will be reduced effectively. Nevertheless, using LoLiMoT NN makes the controller robust to the camera calibration errors and image noises. This contribution has been explained explicitly in section 'Methodology'.

The remainder of this study is as follows: in section 'Methodology', a brief review of different VS controllers will be presented. Then, our proposed method will be explained. In section 'Simulation and experimental setup', the simulation and experimental setup has been presented. Thereafter, in section 'Results and discussion', a comparison of different methods has been investigated followed by results. In section 'Results and discussion', our approach has been tested on the real robot while tracking a dynamic object. Eventually, in section 'Conclusion', a brief summary alongside the conclusion and the future ideas has been stated.

Methodology

The main goal of VS tasks is to regulate the error vector created by the visual features. In what follows, a brief background of well-known VS approaches is

presented, followed by a detailed explanation of the proposed HDVS method.

IBVS

In IBVS, the controller feedback is directly attained by the features in the image space. An interaction matrix (image-Jacobian L_i) is required to link the pixel velocity to the camera velocity

This matrix for the ith feature is given by[33]

$$L_i = \begin{bmatrix} \frac{f}{Z} & 0 & -\frac{u}{Z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v \\ 0 & \frac{f}{Z} & -\frac{v}{Z} & -\frac{f^2+v^2}{f} & \frac{uv}{f} & u \end{bmatrix} \quad (1)$$

in which f is the camera's focal length and $s = (u, v)$ is the coordinates of a point in the image-plane. Let e_i be the error between the current and the desired position of each feature in the image-plane, and let $v_{cam} = (v_c, w_c)$ be the camera velocity vector. $v_c = (v_x, v_y, v_z)$ is the linear velocity vector and $w_c = (w_x, w_y, w_z)$ is the angular velocity vector of the camera. When the interaction matrix at the desired pose is not singular (i.e. $s_i(t) - s_{id}(t) = e_i(t) = 0$), the exponential decoupled decrease of the error could be obtained. As a result, the required camera velocity vector will be calculated from the following control law[33]

$$\begin{bmatrix} v_c \\ \omega_c \end{bmatrix} = -k_i L_i^{+} e_i \quad (2)$$

where k_i is a positive proportional controller gain, and L_i^{+} is the pseudo-inverse of L_i.

PBVS

In PBVS, the feedback is obtained from the pose reconstruction of the environment. The pose estimation will be calculated with the help of Euclidean methods and camera parameters, from the camera image.

The Euclidean coordinate of the features in the camera frame is $\bar{m}_i[\,x_i \quad y_i \quad z_i\,]^{\mathrm{T}}$ and the Euclidean coordinate of the features in the desired camera frame is $\bar{m}_i^*[\,x_i^* \quad y_i^* \quad z_i^*\,]^{\mathrm{T}}$. The controller in this method is defined as[17]

$$\begin{bmatrix} v_c \\ \omega_c \end{bmatrix} = -\lambda_p L_p^{-1} e_p \quad (3)$$

where λ_p is a positive controller gain and e_p is the position error between the 3D estimated position of the camera and the desired 3D position of that in the task space. Moreover, $L_p(t)$ is a 6×6 matrix obtained from the following equation[17]

$$L_p = \begin{bmatrix} R & 0 \\ 0 & L_\omega \end{bmatrix} \quad (4)$$

where $L_\omega(h(t), \theta(t))$ is a 3×3 matrix defined as

$$L_\omega = I_3 - \frac{\theta}{2}[h]\times + \left(1 - \frac{\mathrm{sinc}(\theta)}{\mathrm{sinc}^2\left(\frac{\theta}{2}\right)}\right)[h]\times^2 \quad (5)$$

where $h(t) \in R^3$ and $\theta(t) \in R$ are the rotation axis and rotation angle, decomposed from the rotation matrix. $[h]\times$ is the skew-symmetric matrix associated with the vector h.

Homography-based visual servoing

The homography-based visual servo control approaches mostly decompose the 6-degree-of-freedom (6-DOF) motion of the camera in two separate controllers in order to achieve the convergence goal: one for the translational components and the other one for the rotational components. Let m_{xi} and m_{xi}^* be the normalized Euclidean coordinate vectors and define as follows

$$m_i = \begin{bmatrix} m_{xi} & m_{yi} & 1 \end{bmatrix}^{\mathrm{T}} \triangleq \begin{bmatrix} \frac{x_i}{z_i} & \frac{y_i}{z_i} & 1 \end{bmatrix}^{\mathrm{T}} \quad (6)$$

$$m_i^* = \begin{bmatrix} m_{xi}^* & m_{yi}^* & 1 \end{bmatrix}^{\mathrm{T}} \triangleq \begin{bmatrix} \frac{x_i^*}{z_i^*} & \frac{y_i^*}{z_i^*} 1 \end{bmatrix}^{\mathrm{T}} \quad (7)$$

The error signal will be selected by the estimated position from the Euclidean information and directly from the image information[4]

$$e_v = \begin{bmatrix} m_{xi} - m_{xi}^* & m_{yi} - m_{yi}^* & \ln\left(\frac{z_i}{z_i^*}\right) \end{bmatrix}^{\mathrm{T}} \quad (8)$$

$$e_\omega = h\theta \quad (9)$$

$\dot{e}_v = L_v v_{\mathrm{c}} + L_{v\omega}\omega_{\mathrm{c}}$ and $\dot{e}_\omega = L_\omega \omega_{\mathrm{c}}$ could be defined as corresponding transitional and rotational errors. In which $L_\omega(t)$ was presented in equation (5) and $L_v(t), L_{v\omega}(t)$ are 3×3 matrices that are obtained from

$$L_v = -\frac{\alpha_i}{z_i^*}\begin{bmatrix} 1 & 0 & -m_{xi} \\ 0 & 1 & -m_{yi} \\ 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$$L_{v\omega} = \begin{bmatrix} m_{xi}m_{yi} & -1 - m_{xi}^2 & m_{yi} \\ 1 + m_{yi}^2 & -m_{xi}m_{yi} & -m_{xi} \\ -m_{yi} & m_{xi} & 0 \end{bmatrix} \quad (11)$$

where α is the product of the camera scaling factor. The translation and the rotation controllers could be defined as

$$\omega_c = -kL_\omega^{-1}e_\omega = -ke_\omega \tag{12}$$

$$v_c = -L_v^{-1}(ke_v - kL_{v\omega}e_\omega) \tag{13}$$

HDVS

In the proposed method (HDVS), the translational velocity in Z-axis and three rotational velocities (the components that cause IBVS singularity and physically unimplemented motions for the robot) have been decoupled from the image-Jacobian matrix. The error of these four parameters will be calculated from 3D estimation of the target, while the translational velocities of the X and Y components will be calculated in two-dimensional (2D). The control law in the classical IBVS method is as follows

$$\dot{s} = L_i v_{cam} \tag{14}$$

However, after decoupling the interaction matrix in HDVS, the control law will change to

$$\dot{s} = L_{xy}v_{xy} + L_r v_r \tag{15}$$

where $v_{xy} = [v_x\ v_y]^T$ and $v_r = [v_z\ w_x\ w_y\ w_z]^T$ are the decoupled velocity vectors of the camera. Besides, L_{xy} and L_r are defined as follows

$$L_{xy} = \begin{bmatrix} \frac{f}{Z} & 0 \\ 0 & \frac{f}{Z} \end{bmatrix} \tag{16}$$

$$L_r = \begin{bmatrix} -\frac{u}{Z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v \\ -\frac{v}{Z} & -\frac{f^2+v^2}{f} & \frac{uv}{f} & u \end{bmatrix} \tag{17}$$

Therefore

$$v_{xy} = L_{xy}^{+}\{\dot{s} - L_r v_r\} \tag{18}$$

Since $\dot{s} = -ke$

$$v_{xy} = L_{xy}^{+}\{-k(\|\mathrm{e}\|)e - L_r v_r\} \tag{19}$$

In order to decrease the convergence time, adaptive representation of the controller gain has been formulated as follows[34]

$$k(\|\mathrm{e}\|) = (k(0) - k(\infty))e^{\frac{-k(0)}{k(0)-k(\infty)}\|\mathrm{e}\|} + k(\infty) \tag{20}$$

where $k_0 = k(0)$ is the positive amount of gain for small amounts of $\|\mathrm{e}\|$, $k_\infty = k_{\|\mathrm{e}\|\to\infty}, k(\|\mathrm{e}\|)$ is for high amounts of $\|\mathrm{e}\|$, and $k_0^{'}$ represents the slope of k at $\|\mathrm{e}\| = 0$. At each iteration, $L_r v_r$ will be calculated and its amount will be placed in equation (19) to compute the other velocity components. The same process will be performed to compute v_r in PBVS

$$\dot{s}_p = L_{Pxy}v_{xy} + L_{Pr}v_r \tag{21}$$

where L_{Pxy} is the first two columns of L_P in equation (4), and L_{Pr} is the other four columns of L_P. Therefore

$$v_r = L_{Pr}^{+}\{-k(\|\mathrm{e}\|)e_p - L_{Pxy}v_{xy}\} \tag{22}$$

By solving equations (19) and (22) simultaneously, the camera velocity vector will be determined. The transformation matrix ξ_c^e is defined to map the velocities expressed in robot EE frame to the camera frame[35]

$$\xi_c^e = \begin{bmatrix} R_c^e & sk(t_c^e)R_c^e \\ 0 & R_c^e \end{bmatrix} \tag{23}$$

where t_c^e is the translation vector between EE and the camera frame, and R_c^e is the rotation matrix between these frames, and $sk()$ is a skew-symmetric matrix. In the eye-in-hand configuration, ξ_c^e will remain constant. However, it should be calculated in each iteration for eye-to-hand configuration. Finally, after calculating the EE velocities, using kinematics of the robot, joint velocities will be computed

$$\dot{q} = J^{\dagger\lambda}\xi_c^e v_{cam}^c \tag{24}$$

In this research, DLS inverse has been used instead of pseudo-inverse. Using DLS, the effect of robot's singularities will be reduced and it will smooth the discontinuities created from decoupling process and using the adaptive gains.[36] It is worth mentioning that using regularization techniques could also help to reduce the effect of singularity configurations, but they will increase the convergence time.[37] DLS is formulated as follows[36]

$$J^{\dagger^\lambda} = J^T(JJ^T + \lambda^2 I)^{-1} \tag{25}$$

where λ represents the damping factor which is a positive scalar.

In Figure 2, the control schema of the proposed visual servoing controller is shown. Using a vision sensor mounted on the robot's wrist, features will be detected and will be used as feedback of the controller. The red blocks represent image-based parts of the control loop, the blue blocks are position-based parts, and the grey blocks are the task space parts.

As it is depicted in the control block diagram (Figure 2), the camera velocities have been decoupled; two of them (translation in X and Y) have been considered in 2D (using pure features created from the image screen, as feedback). The remaining components have been considered in 3D (computed by partial 3D reconstruction of the environment attained by the extracted

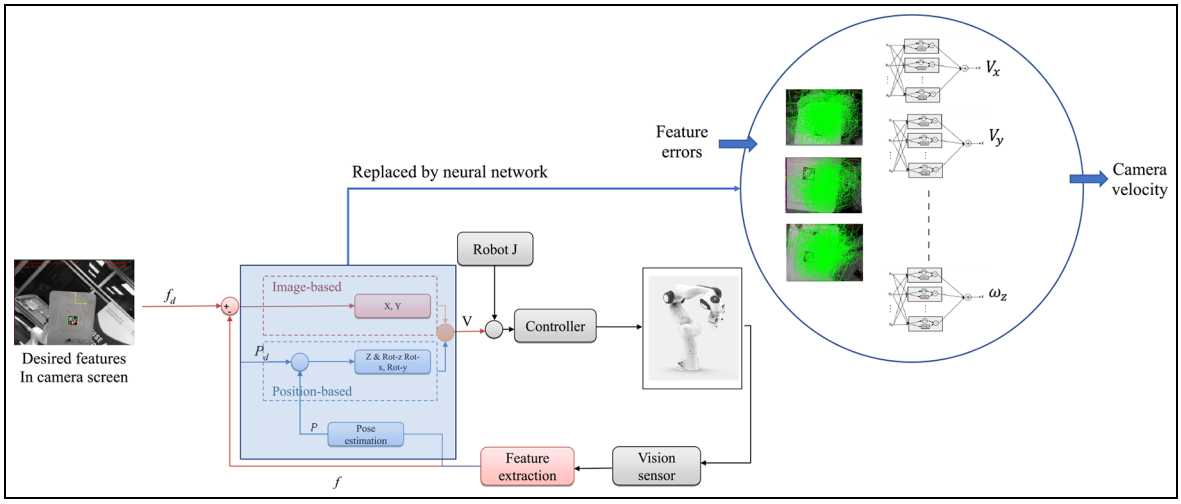


Figure 2. The control schema of the proposed visual servoing (HDVS) integrated with NN.

features). Thereafter, the computed velocities will be commanded to the robot. Using the robot's Jacobian, the controller will exchange the desired camera velocities with the desired joint velocities. The computed joint velocities will be used as input of the robot velocity controller. As it was stated earlier, DLS inverse has been used instead of pseudo-inverse. Controlling the rotations and translation separately in the Z-axis is acquired from the 3D estimation of the visual features.

Estimation of camera velocities using NN

A trained NN is employed to provide an accurate estimation of the camera velocities from the feature errors (the interaction matrix role). The approximated equations by the NN are the combination of the right-hand side of equation (19) (including pseudo-inverse of L_{xy} to calculate translational velocities in X- and Y-axis from 2D image errors) and the right-hand side of the equation (22) (including pseudo-inverse of L_{Pr} to calculate the translational velocity in Z-axis and three rotational velocities from estimated 3D errors of the image features). To this end, 76,000 sets of unique feature errors with their relevant camera velocities (calculated from the achieved equations (19) and (22)) all over the image screen (480 × 640 pixel) for the test and training data have been gathered. These results were taken by moving the camera with hand and tried to include feature's positions all around the image screen as best as possible. In Figure 2, the positions of the current features have been shown in the camera screen with green. Therefore, feature errors in the image screen were deployed as input of the NN and the calculated camera velocities were defined as the outputs. LoLiMoT NN was used for this purpose. In comparison to other neuro-fuzzy NNs, LoLiMoT is more efficient in learning non-linear systems with fewer neurons.[38]

The LoLiMoT algorithm operates by the errors and it is axis orthogonal. Therefore, the network divides the

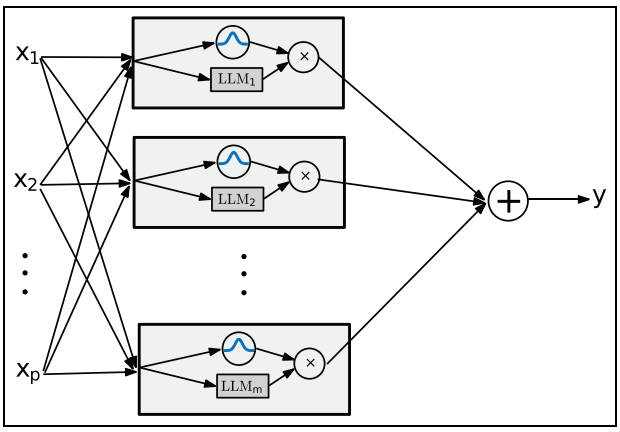


Figure 3. The structure of LoLiMoT NN with m neuron and P input.

space in a properly optimized way w.r.t. the errors. Other types of NNs, which work by the use of gradient descents, operate blind and cannot guarantee the global optimization.[39] The structure of the LoLiMoT network is depicted in Figure 3. A linear local model alongside a membership function is assigned to each neuron. For the allocated linear model, the validated area would be determined with an assigned membership function. The linear model is formulated as follows[40]

$$\hat{y}_i = \omega_{i0} + \omega_{i1}x_1 + \cdots + \omega_{ip}x_p \tag{26}$$

In equation (26), ω_{ij} is the associated parameters to ith neuron. The final output is calculated as follows

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i \phi_i(x), \quad \sum_{i=1}^{M} \phi_i(x) = 1 \tag{27}$$

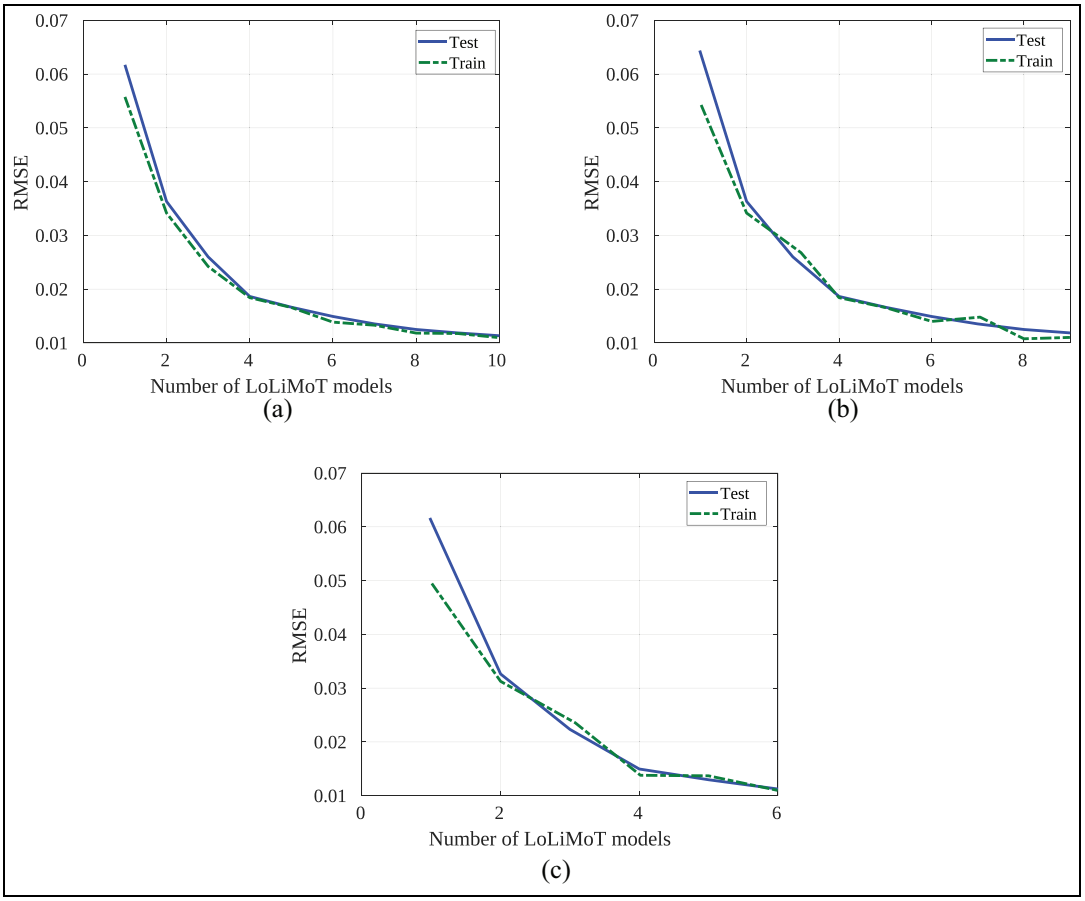


Figure 4. RMSE of test and train samples of LoLiMoT neural network for translational camera velocity outputs: (a) RMSE of V_x, (b) RMSE of V_y and (c) RMSE of V_z.

where $\phi_i(x)$ stands for the membership function which is assumed as a normalized Gaussian function in equation (27). The corresponding member function is derived by the following equation

$$\phi_i(x) = \frac{\mu_i(x)}{\sum_{j=1}^{M} \phi_j(x)}$$
$$\mu_i(x) = e^{\frac{(u_1 - c_{i1})^2}{-2\sigma_{i1}^2}} \times \cdots \times e^{\frac{(u_p - c_{ip})^2}{-2\sigma_{ip}^2}} \quad (28)$$

In equation (28), c_{ij} and σ_{ij} are the centre of the area and the standard deviation, respectively. The NN has been trained for each output separately (six camera velocities). As illustrated in Figure 2, pseudo-inverse of the decoupled image-Jacobian and pose estimation are replaced with six NNs to predict the camera velocity vector at each iteration. The feature error vector will be computed by subtraction of current and desired features. Control input has been defined by eight components of the error vector; consequently, the output would be the camera velocity vector with six components. This camera velocity vector will be transformed to the EE frame of the robot using equation (23). Eventually, joint velocities computed using the robot's Jacobian, and they were commanded to the motors of each joint. The control loop will be continued until feature errors converge to zero.

Moreover, supplementary noise had been added to the NN input data set during training to improve its robustness to the image noises. The added Gaussian noise has a standard deviation of one and a mean of zero (white noise), generated by a pseudo-random number generator. Using NN, the complexity of 3D estimation of the target position and the pseudo-inverse of the decoupled image-Jacobian have been relaxed, as described in section 'Methodology'.

For each output, one LoLiMoT network has been used. Therefore, six NNs have been trained, which comes after six camera velocities. Each network has a different number of LoLiMoT models. The number of models achieved during learning process. The learning process stops when the test and train samples acquire a pre-defined accuracy. This accuracy will be checked by the root mean square of errors (RMSEs) in both test and train samples (RMSE of 0.01 m/s for translational velocities and RMSE of 0.01 rad/s for rotational velocities). Such a behaviour will effectively decrease the number of trial and errors.

The number of allocated LoLiMoT models (membership functions alongside linear local model) and RMSE of the trained NN for the test and the train samples have been depicted in Figures 4 and 5. Each

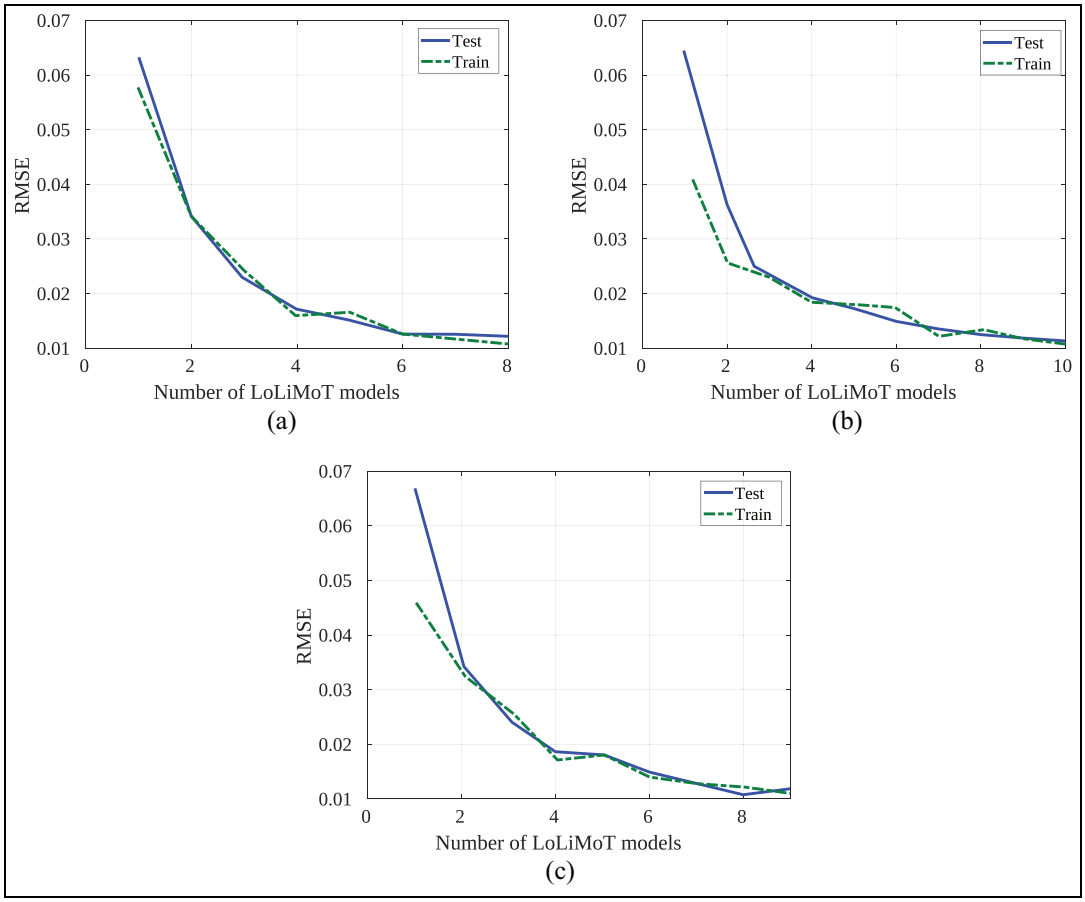


Figure 5. RMSE of test and train samples of LoLiMoT neural network for rotational camera velocity outputs: (a) RMSE of ω_x, (b) RMSE of ω_y and (c) RMSE of ω_z.

LoLiMoT model consists of a linear local model alongside a Gaussian membership function.

As Figure 4(a) suggested, having 10 models is optimal in order to have the RMSE of 0.01 in test and train samples. Having more than 10 models is redundant and could lead the network to over-fitting. The number of LoLiMoT models is 9 for the prediction of velocity controller in Y direction (Figure 4(b)). These numbers are 6 for velocity in Z (Figure 4(c)).

In Figure 5(a), it can be seen that 8 LoLiMoT models are required for rotational velocity about X-axis, 10 models for rotational velocity about Y-axis (Figure 5(b)) and 9 LoLiMoT models for rotational velocity about Z-axis (Figure 5(c)). The trained set of these six networks will be used to predict the camera velocity required to perform the visual servoing task. The updated set of feature errors will be utilized for the networks at each iteration.

Simulation and experimental setup

In order to evaluate the efficacy of the method and to compare it with other techniques, the proposed method has been modelled in simulation using ROS/Gazebo. In Figure 6, a snapshot of the simulation environment in Gazebo is shown. Two Franka manipulators have been added to the simulation platform. One with a camera mounted on its wrist, for the VS purpose, and another one carrying a tag marker on a sheet attached to the EE.

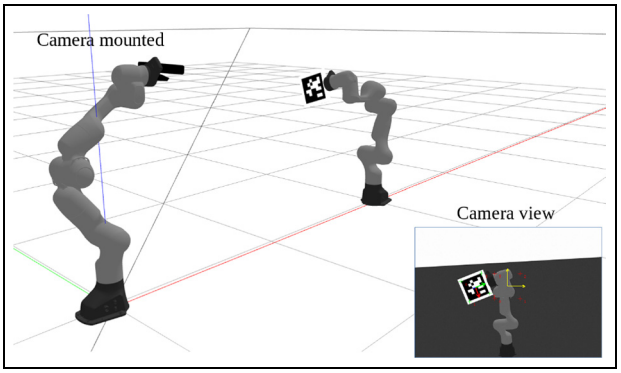


Figure 6. The simulation environment modelled in Gazebo.

The experimental setup is identical to the one used in the simulation. In this research, four corners of an AprilTag are used as points of interest. The Intel RealSense depth camera D435i has been used for eye-in-hand configuration as the vision sensor. For the VS

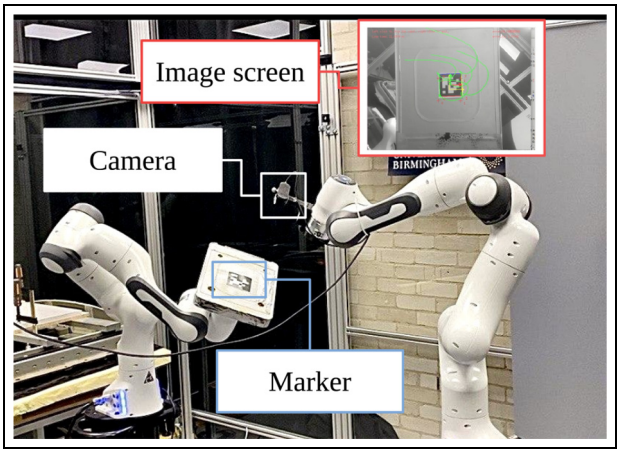


Figure 7. Experimental setup for visual servoing task.

operation, a system with the following processor had been used: AMD Ryzen 7 3700X 8-core processor 16 (thread) with 3.6 GHz base clock and 36 MB total cache. The experimental setup used in this research is depicted in Figure 7. In section 'Results and discussion', the behaviour of different control schemes in both simulation and experiments will be compared.

Results and discussion

In order to evaluate the effectiveness of the HDVS method, various scenarios have been studied and compared with HVS, IBVS and PBVS approaches. To have a logical comparison, same adaptive gains have been used for every method ($k_0 = 4$, $k_\infty = 0.4$, $k_0' = 30$). The λ gain in DLS inverse is set to 0.1 for the entire tests. The robot initial position and the AprilTag position are both same in each case study. Not to mention that the controller behaviour could be enhanced by tuning the adaptive gains which is not the interest of this article. In what follows, the proposed HDVS method and other classical methods (IBVS, PBVS and HVS) have been compared.

Case study 1: comparing the performance of PBVS with HDVS

A case study has been defined to follow a pre-defined position of the target with an uncalibrated camera. The intrinsic calibration of the camera has been degraded by 20%. The performance of HDVS and PBVS in regulating the errors is illustrated in Figure 8. In Figure 8(a), it is shown that the position-based controller has been failed to track the features. It is depicted in Figure 8(a) that the controller could not follow the desired features from iteration 350. This failure comes from the fact that there is an error in the 3D estimation of the target generated by an uncalibrated camera.

The reason that PBVS is not robust to the camera calibration is that the errors in camera parameters propagated to the errors in 3D estimation of the target. However, HDVS is robust in terms of camera parameters. Figure 8(b) suggested that HDVS tracked the desired features successfully. This is because in HDVS camera, velocities are generated directly from the trained LoLiMoT NN. Collecting data for the NN has been done with accurate camera calibration. Therefore, it is not important how inaccurate is the camera calibration in online mode (i.e. robot during VS), the controller will work well and is highly robust to camera calibration errors.

Case study 2: comparing the performance of IBVS with HDVS

IBVS failures are mostly associated with the rotations about Z-axis (Chaumette Conundrum[7]) and camera retreat.[8] Furthermore, camera ambiguities might happen between translations and rotations in the image-plane; this implies that different camera velocities could produce same motion of points in the image and some camera motions lead to no change in the image.[11] A case study has been defined in such that the desired features rotated 90° about the Z-axis. The performance of

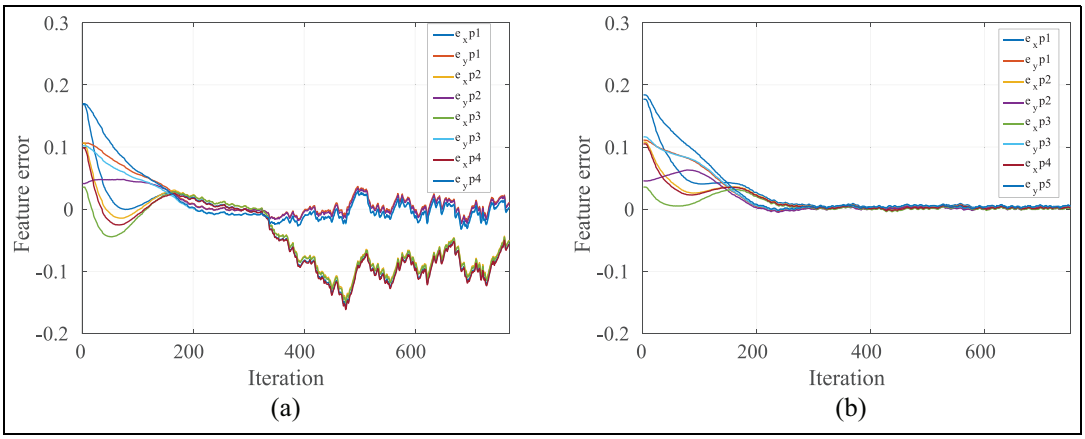


Figure 8. Simulation result of tracking desired features with an uncalibrated camera for (a) PBVS and (b) HDVS.

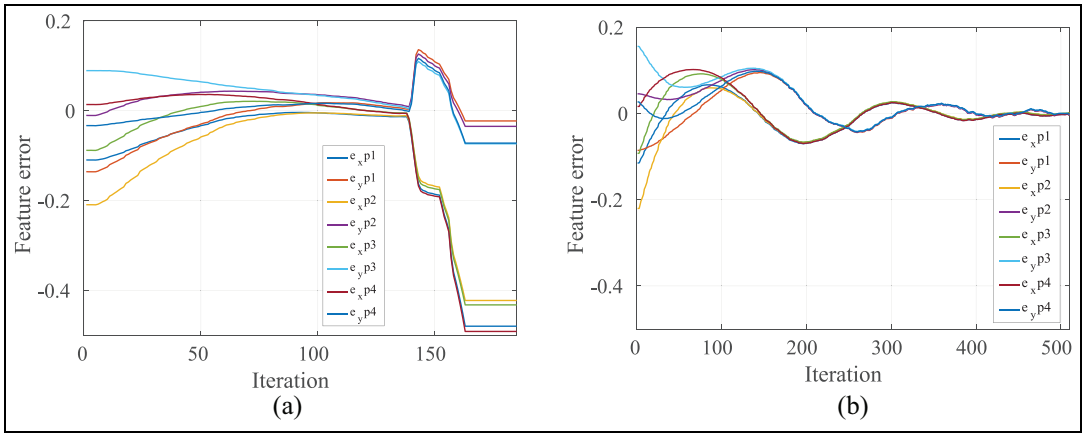


Figure 9. Simulation results for (a) IBVS and (b) HDVS control approaches.

HDVS and IBVS in converging feature errors to zero is depicted in Figure 9.

As shown in Figure 9, the controller will end up with an unpleasant behaviour in IBVS. Figure 9(a) shows that the errors could not converge to zero as the controller produces a large velocity in Z which makes the robot going to its joint limits. The main reason associates with this limitation is that reducing the rotation error about Z-axis in the image screen could be achieved by moving the camera away from the target (the so-called camera retreat phenomenon). Such a wrong decision in 2D could produce a large motion in the Z-axis in 3D. An extreme version of this behaviour is when there is a pure π rad rotation between features and their desired positions in the image. In such a case, the features will be driven towards the origin by mistake and cause image singularity.[7] In this configuration, the features cannot reach to the desired ones even if the controller runs for ever.[8] The proposed method will provide the most straightforward way of compensating errors of Z and rotations in task space. This claim comes from the fact that the error of these four parameters (translation in Z-axis and three rotations) is directly compensated from 3D estimation of the target position w.r.t. the camera position. Moreover, using LoLiMoT NN, the controller becomes robust in terms of camera calibration which will make the 3D estimation accurate.

In Figure 9(b), it is illustrated that using the proposed controller, the errors easily converge (in 509 iterations) without causing image singularity. Moreover, HDVS avoids camera ambiguity while the controller distinguishes the camera rotations from translations. Camera ambiguity could be explained from the structure of the image-Jacobian matrix in equation (1). When camera focal length is large or when the pixel coordinates are small, columns 1 and 4 become very similar. Same ambiguity could be happened for columns 2 and 5 (large focal length dominates u squared). Therefore, transnational velocities in the X-axis and Y-axis could not be easily detected from rotational velocities about Y-axis and X-axis, respectively. These components are decoupled from the image-Jacobian in HDVS, and the controller avoided these kinds of ambiguities.

Case study 3: comparing the performance of HVS with HDVS

One of the advantages of HDVS in comparison with traditional HVS methods is a significant reduction in computation costs. Computing and modelling process (equations (12) and (13)) in traditional HVS for each iteration is time-consuming and complex. However, in HDVS, neuro-fuzzy NNs are used for estimating the required velocities. Since the NNs are trained offline and used online as a predictor of required velocities, the convergence time will be reduced considerably. Moreover, HVS methods need homography construction and decomposition. Homography construction is sensitive to image noises, and there exists sign ambiguity in homography decomposition (non-unique solutions). However, HDVS is robust to image noises and globally optimized solutions will be achieved from the LoLiMoT network. In this context, a case study has been introduced in the next section to compare the performance of HDVS with HVS, IBVS and PBVS.

Case study 4: comparing the performance of four VS methods altogether

In Figure 10, the robot's EE Cartesian trajectory in task space has been depicted for all four methods for a same pre-defined position of the Tag marker. In Figure 10, the start point is the initial position of the EE and the endpoint is the position of the EE when visual errors are regulated to zero.

Figure 10 suggested that shorter camera path is obtained for HDVS (blue path) than HVS (purple path) and IBVS (red path). To clarify, the distance travelled by the camera (robot's EE) is 0.801 m in HDVS; however, this amount is 0.942 m in IBVS and 0.917 m

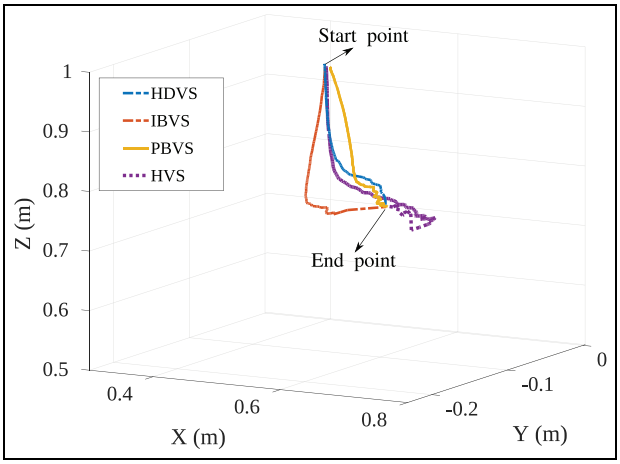


Figure 10. Comparison of EE trajectory for different methods in the real world.

in HVS. It should be mentioned that the most optimized Cartesian trajectory of the robot's EE has been obtained by the PBVS method. The camera travelled distance is 0.722 m in this approach. As shown in Figure 10, IBVS (red path) showed the most unpleasant behaviour in the Cartesian trajectory of the EE, since the controller performs blind in the task space. Therefore, the robot in IBVS is more probable to move to joint limits and collide with the obstacles, especially when large rotations are required. PBVS (yellow path) performs well in the task space of the robot; however, it is sensitive to camera calibrations. Moreover, 3D estimation of the target must be calculated and be updated online at each computationally expensive iteration. In HDVS, the Z component is also separated from the image-Jacobian which avoids unnecessary motions in 3D. In addition, HDVS computations are faster than HVS. Therefore, HDVS is more suitable for online object tracking. Figure 11 illustrates feature errors of different approaches in VS for a similar assigned position of the Tag marker.

The RMSE is 0.021 in IBVS, 0.036 in PBVS, 0.032 in HVS and 0.024 in HDVS. In Figure 11(b), the IBVS represents the most optimized path in the camera frame, followed by HDVS. The RMSE in IBVS is less than other three methods. In Figure 11(b), the maximum feature error along the entire path is 0.089 which implies that there is a very low risk to lose the object from the camera screen. PBVS has the most unpleasant RMSE in camera screen than other methods (Figure 11(a)) as it performs blind in image screen. It is more probable in PBVS to lose the object from the camera fields of view since it has the highest feature error (0.24) compared to other counterparts. From Figure 11(c) and (d), it is clear that the HDVS method is faster (converged within 350 iterations) than HVS (converged within 510 iterations). Moreover, the convergence is more pleasant (fewer overshoots with same adaptive gains) and the RMSE is smaller in HDVS.

From Figures 10 and 11, it could be concluded that the proposed hybrid method has not necessarily the best behaviour in tracking the features in image space and Cartesian space (robot space), but it has an optimized behaviour in both. It comes from the fact that two components of camera velocities computed directly from the image space (translation in X and Y), while others computed from 3D reconstruction of the environment. Moreover, in HDVS, the object is less likely to be lost from the camera field of view than HVS and PBVS. This conclusion has been derived by comparing the maximum feature error in Figure 11(d), which is less than that amount in Figure 11(a) and (c). The bigger the error, the more it is probable to lose the feature from the camera field of view.

In Tables 1 and 2, a quantitative comparison of effective parameters in VS methods has been presented. Given that the quantitative amounts are obtained by the mean of 40 experiments with 10 different paths, repeated for all different methods in the same condition.

Looking at Table 1, the mean RMSE amount of HDVS is less than PBVS and HVS. As a result, HDVS performed better in image space than PBVS and classical HVS. Table 2 shows that the mean RMSE amounts of position and orientation in PBVS were less than their counterparts in the other three methods. Therefore, PBVS had the most optimized performance in Cartesian space. As it was expected, the performance of HDVS was better than IBVS and HVS in Cartesian space.

Another conclusion from Table 1 is that the feature error range in PBVS and classical HVS is bigger than HDVS. Therefore, the target is less probable to be lost from the camera screen in HDVS than the other two methods. Not to mention that HDVS performs faster compared to classical HVS, and fewer iterations are required to perform the same task. From Table 1, it can be seen that not only the IBVS has the smallest feature error range, but also has the smallest RMSE in comparison to other approaches. Assuredly, IBVS outperformed in image-plane if no singularity or local minima happen. However, the controller performs blind in Cartesian space (the highest RMSE for position and orientation in Table 2) and undesirable large camera motions often happen. All in all, in Cartesian space, HDVS suggests more optimized performance than IBVS and HVS (w.r.t. mean RMSE amounts presented in Table 2). In addition, HDVS performs more optimized than PBVS and HVS in image space (w.r.t. mean RMSE amounts presented in Table 1).

In Figure 12, the CPU and the RAM usage while different techniques are running has been shown. The comparison between the graphs in Figure 12 clearly illustrates our claim that HDVS is less computationally heavy compared to the other three methods. The CPU usage in HDVS is around 90%; however, this amount is above 105% for other techniques.

A qualitative comparison of the aforementioned results for all different methods is presented in Table 3.

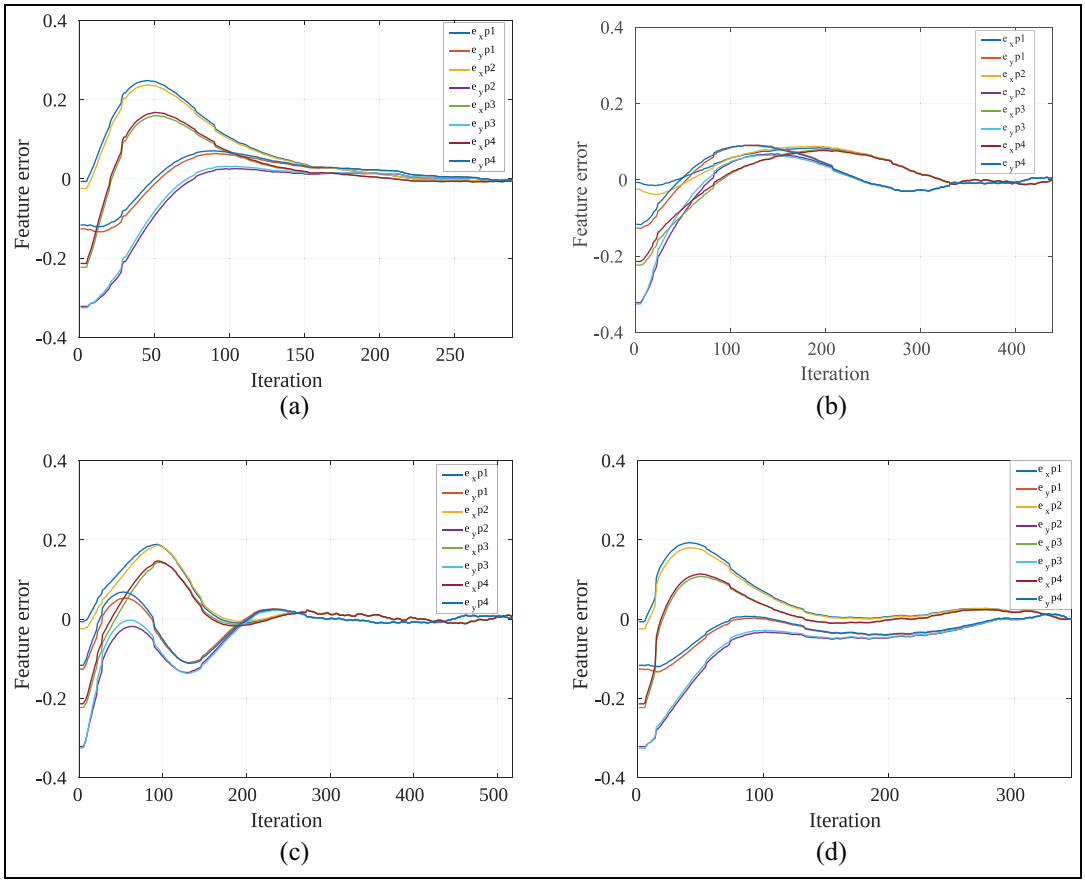


Figure 11. Comparison of feature errors for different methods in the real world: (a) PBVS, (b) IBVS, (c) HVS and (d) HDVS.

Table 1. Comparison of visual servoing methods' performance in image-plane.

Method	RMSE	Feature error range	Iteration
IBVS	0.0222	[−0.36, 0.310]	453
PBVS	0.0383	[−0.445, 0.507]	487
HVS	0.0273	[−0.448, 0.486]	624
HDVS	0.0249	[−0.419, 0.407]	505

RMSE: root mean square of error; IBVS: image-based visual servoing; PBVS: position-based visual servoing; HVS: hybrid visual servoing; HDVS: hybrid decoupled visual servoing.

Table 2. Comparison of visual servoing methods' performance in Cartesian space.

Method	RMSE of position (*m*)	RMSE of orientation (°)	Camera (EE) travelled distance (*m*)
IBVS	0.036	9.43	0.942
PBVS	0.022	6.54	0.722
HVS	0.034	8.41	0.917
HDVS	0.027	7.11	0.801

RMSE: root mean square of error; EE: end-effector; IBVS: image-based visual servoing; PBVS: position-based visual servoing; HVS: hybrid visual servoing; HDVS: hybrid decoupled visual servoing.

Table 3 illustrates that the proposed HDVS method offers an optimized trajectory both in Cartesian and image space. Furthermore, the controller is highly robust in terms of camera calibration (PBVS problem), image singularities (IBVS problem) and image noises (HVS problem). Not to mention that the proposed method has significantly relaxed the calculation, and the convergence speed is better than the classical HVS method. Figure 13 compares the success rate of four VS methods for 10 different random paths in

Table 3. A qualitative comparison of visual servoing schemes.

	IBVS	PBVS	HVS	HDVS
Optimized Cartesian trajectory	Low	High	Med	High
Optimized feature trajectory	High	Low	Med	Med
No image singularities	No	Yes	Yes	Yes
Robust to camera calibration	High	Low	Med	High
Error convergence speed	High	Low	Low	Med
No computationally complex	Med	Med	Med	High
Robust to image noises	High	Med	Low	High

IBVS: image-based visual servoing; PBVS: position-based visual servoing; HVS: hybrid visual servoing; HDVS: hybrid decoupled visual servoing; Med: medium.

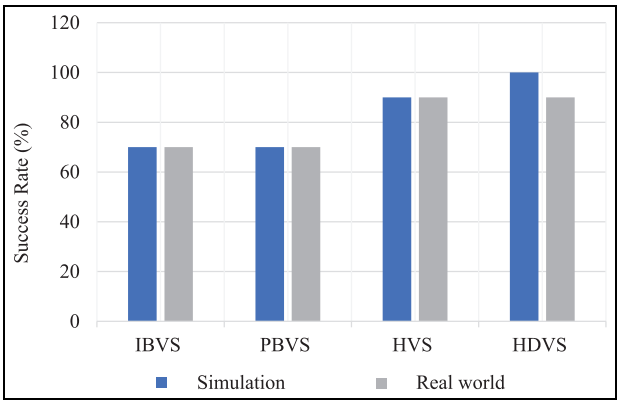


Figure 12. Comparison of CPU and RAM usage while each method is running in the real world: (a) PBVS, (b) IBVS, (c) HVS and (d) HDVS.

simulation and their counterparts in the real world. From Figure 13, it can be concluded that HDVS outperformed with its success rate in both simulation (with 10 out of 10 successful trials) and real world (with 9 out of 10 successful trials) compared to other VS methods.

Case study 5: Evaluating the performance of HDVS in tracking a moving object in the real world

In order to validate the results obtained from the simulation, a series of experiments conducted in the real world using a Franka robot arm. Both intrinsic and extrinsic calibrations of the camera have been accomplished using visual servoing platform (ViSP) libraries.[41] The adaptive gain parameters used in equation (20), where $k_0 = 4$, $k_\infty = 0.4$ and $k_0^{'} = 30$ (same as simulation). In this case study, the robot with the camera mounted on its wrist will follow the features of a moving target. The EE trajectory is defined in which it covers all possible rotations and translations. A pure rotation of 90° has been defined to show the robustness of the proposed method versus IBVS failures. Moreover, in order to point out the robustness of our proposed method against camera parameters, a supplementary error has been added to the intrinsic parameters of the camera. The controller failed to perform the task in IBVS and PBVS as a result of this change. The performance of HDVS during tracking the visual features is depicted in Figure 14.

Target is not static in this case study; therefore, there exist overshoots in the plots. The VS task was performed in 3100 iterations with RMSE of 0.074 along the entire path. In Figure 14(b), there is a peak in rotational velocity about Z-axis (iteration 100) which is created as a result of a pre-defined (90°) rotation. This large rotational velocity reveals that HDVS has a high capability in estimating the position of visual features in 3D. However, the error in the rotation could be interpreted wrongly as a translation error in Z-axis due to the camera retreat phenomenon in IBVS. Figure 15 compares the feature motions in image-plane for two HVS and HDVS methods.

In Figure 15, green points denote the desired position of their corresponding features. Figure 15 suggested that the HDVS method (Figure 15(a)) aimed to keep the features closer to their desired positions than the HVS method (Figure 15(b)). This capability comes from the fact that the controller computes the translational velocities in the X-axis and Y-axis directly from the image-Jacobian while having explicit control over the rotations and translation in Z-axis. The motion of features implies that it is more likely to lose the points in the camera field of view by HVS than HDVS. Total image coordinates RMSE amount of 0.074 for HDVS and 0.1124 for HVS validates this claim.

Figure 16(a) and (b) shows the 3D trajectory of the robot's EE generated by the HDVS and classical HVS controllers, respectively. HDVS controller offers a better performance in the task space than HVS. Total Cartesian position RMSE amount of 0.028 m for HDVS and 0.040 m for HVS, alongside total orientation RMSE of 6.388° for HDVS and 7.021° for HVS, declares this claim. Furthermore, the number of iterations to complete the task in HDVS is less than this amount in HVS. In this experiment, the number of iterations was 3078 in HDVS; however, this number was 3200 in the classical HVS method.

Conclusion

In this article, a HDVS method has been proposed. This method has been developed to overcome the

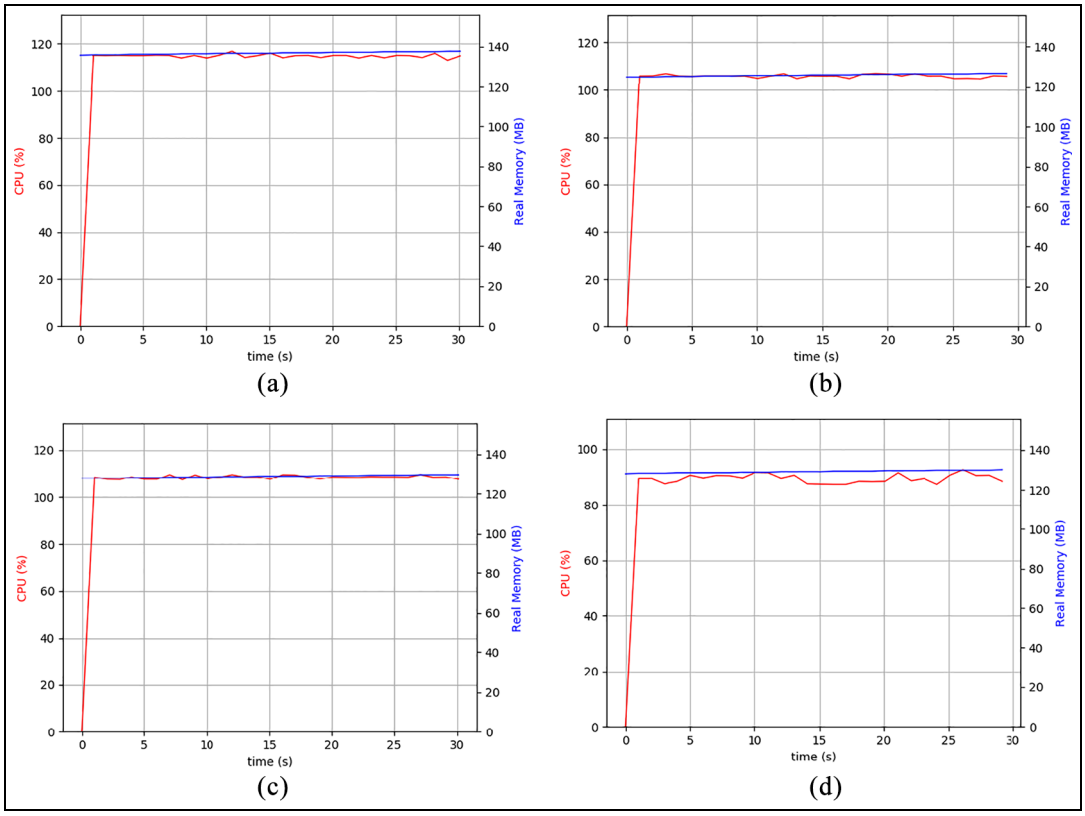


Figure 13. The success rate of four VS methods over 10 trials in the simulation and the real world.

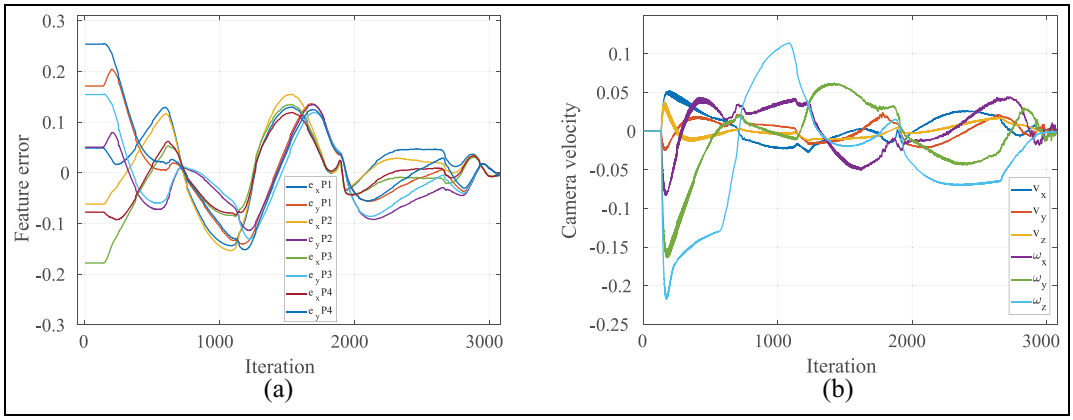


Figure 14. Analysing the performance of HDVS during tracking a dynamic object in the real world using a robot arm: (a) feature errors in HDVS and (b) camera velocity in HDVS.

drawbacks of classical IBVS and PBVS methods and to improve the classical HVS method. In HDVS, all three rotations and translation in the Z-axis have been decoupled from the image-Jacobian. These four components' errors will be regulated to zero by the 3D reconstruction of the visual features. Thereafter, a neuro-fuzzy LoLiMoT NN was used to approximate the pseudo-inverse of the proposed interaction matrix. Moreover, the convergence time was reduced with the help of adaptive gains rather than a constant gain. DLS method was applied in order to reduce the effect of robot singularities and to smooth the discontinuities.

The method not only has an optimized solution for the robot's EE, but it also considers the optimized trajectories of features in the image space. Furthermore, HDVS has improved the performance of classical HVS in both image-plane and task space. The method is robust in terms of camera parameters and image noises, and it avoids singularities of the image-Jacobian effectively. In addition, it is less likely to lose the object from the camera fields of view than PBVS and HVS methods. Results obtained from 10 different random paths in simulation and their counterparts in the real world suggested 100% and 90% success rate in executing the VS tasks in simulation and the real world, respectively.

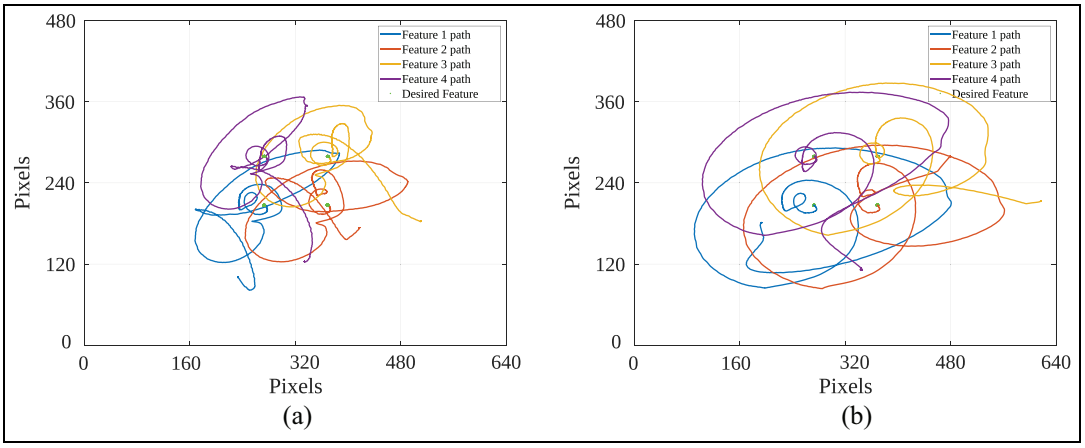


Figure 15. The trajectory of each feature to their desired ones, during the whole task of VS for (a) HDVS and (b) HVS techniques in the real world.

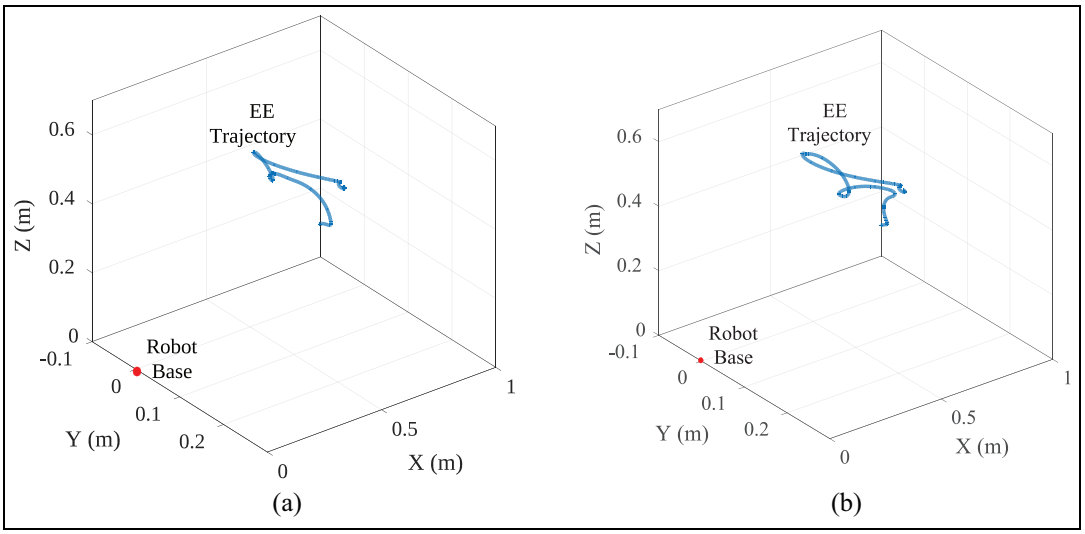


Figure 16. Three-dimensional trajectory of the EE in task space generated by (a) HDVS and (b) HVS in the real world.

Thus, future work will be devoted to extend the LoLiMoT model domain to link feature errors directly to the joint velocities. Moreover, deep NNs could be used to function as feature extractions to overcome the complexities of feature selection.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship and/or publication of this article: This research was conducted as part of the project called 'Reuse and Recycling of Lithium-Ion Batteries' (RELIB). This work was supported by the Faraday Institution (grant no. FIRG005).

ORCID iD

Alireza Rastegarpanah https://orcid.org/0000-0003-4264-6857

Data availability statement

The data that support the findings of this study (the simulation in Gazebo/ROS, C++ codes and demo clip) are openly available in Figshare (https://figshare.com/articles/journal_contribution/Optimized_Hybrid_Decoupled_Visual_Servoing_simulation_and_code/12980009) with doi (10.6084/m9.figshare.12980009.v1).[42]

References

1. Hashimoto K and Husband T. *Visual servoing: real-time control of robot manipulators based on visual sensory feedback*. Singapore: World Scientific Publishing, 1993.
2. Chaumette F, Hutchinson S and Corke P. Visual servoing. In: Siciliano B and Khatib O (eds) *Springer handbook of robotics*. Cham: Springer, 2016, pp.841–866.

3. Espiau B, Chaumette F and Rives P. A new approach to visual servoing in robotics. *IEEE T Robotic Autom* 1992; 8(3): 313–326.
4. Chaumette F and Hutchinson S. Visual servo control, part I: basic approaches. *IEEE Robot Autom Mag* 2006; 13: 82–90.
5. Deng L, Janabi-Sharifi F and Wilson WJ. Stability and robustness of visual servoing methods. In: *Proceedings of the 2002 IEEE international conference on robotics and automation* (Cat. No. 02CH37292), Washington, DC, 11–15 May 2002, vol. 2, pp.1604–1609. New York: IEEE.
6. Han XF, Laga H and Bennamoun M. Image-based 3D object reconstruction: state-of-the-art and trends in the deep learning era, 2019, https://arxiv.org/abs/1906.06543
7. Chaumette F. Potential problems of stability and convergence in image-based and position-based visual servoing. In: Kriegman DJ, Hager GD and Morse AS (eds) *The confluence of vision and control*. London: Springer, 1998, pp.66–78.
8. Corke PI and Hutchinson SA. A new partitioned approach to image-based visual servo control. *IEEE T Robotic Autom* 2001; 17(4): 507–515.
9. Wilson WJ, Hulls CCW and Bell GS. Relative end-effector control using Cartesian position based visual servoing. *IEEE T Robotic Autom* 1996; 12(5): 684–696.
10. Palmieri G, Palpacelli M, Battistelli M, et al. A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator. *J Robot* 2012; 2012: 103954.
11. Corke P. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised, extended and updated edition*, vol. 118. Berlin; Heidelberg: Springer, 2017.
12. Hafez AHA, Cervera E and Jawahar CV. Hybrid visual servoing by boosting IBVS and PBVS.In: *Proceedings of the 3rd international conference on information and communication technologies: from theory to applications*, Damascus, Syria, 7–11 April 2008, pp.1–6. New York: IEEE.
13. Gans NR and Hutchinson SA. Stable visual servoing through hybrid switched-system control. *IEEE T Robot* 2007; 23(3): 530–540.
14. Kumar DS and Jawahar CV. Robust homography-based control for camera positioning in piecewise planar environments. In: Kalra PK and Peleg S (eds) *Computer vision, graphics and image processing*. Berlin; Heidelberg: Springer, 2006, pp.906–918.
15. Chesi G, Hashimoto K, Prattichizzo D, et al. Keeping features in the field of view in eye-in-hand visual servoing: a switching approach. *IEEE T Robot* 2004; 20(5): 908–914.
16. Cervera E, Del Pobil AP, Berry F, et al. Improving image-based visual servoing with three-dimensional features. *Int J Robot Res* 2003; 22(10–11): 821–839.
17. Malis E, Chaumette F and Boudet S. 2 1/2D visual servoing. *IEEE T Robotic Autom* 1999; 15(2): 238–250.
18. Hu G, Gans N and Dixon WE. Quaternion-based visual servo control in the presence of camera calibration error. *Int J Robust Nonlin* 2010; 20(5): 489–503.
19. Shi H, Li X, Hwang KS, et al. Decoupled visual servoing with fuzzy Q-learning. *IEEE T Ind Inform* 2018; 14(1): 241–252.
20. Castelli F, Michieletto S, Ghidoni S, et al. A machine learning-based visual servoing approach for fast robot control in industrial setting. *Int J Adv Robot Syst*. Epub ahead of print 9 November 2017. DOI: 10.1177/1729881417738884.
21. Miljković Z, Mitić M, Lazarević M, et al. Neural network Reinforcement Learning for visual control of robot manipulators. *Expert Syst Appl* 2013; 40(5): 1721–1736.
22. Lampe T and Riedmiller M. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In: *Proceedings of the 2013 international joint conference on neural networks (IJCNN)*, Dallas, TX, 4–9 August 2013, pp.1–8. New York: IEEE.
23. Zhu XJ. Semi-supervised learning literature survey. *Technical report, Department of Computer Sciences, University of Wisconsin–Madison, Madison, WI*, 7 September 2005.
24. Zhang Y and Li S. A neural controller for image-based visual servoing of manipulators with physical constraints. *IEEE T Neur Net Lear* 2018; 29: 5419–5429.
25. Ramachandram D and Rajeswari M. A short review of neural network techniques in visual servoing of robotic manipulators. In: *Proceedings of the Malaysia-Japan seminar on artificial intelligence applications in industry*, Kuala Lumpur, Malaysia, 24–25 June 2003, pp.24–25. Penerbit UTM Press.
26. Farahmand AM, Shademan A, Jagersand M, et al. Model-based and model-free reinforcement learning for visual servoing. In: *Proceedings of the IEEE international conference on robotics and automation*, Kobe, Japan, 12–17 May 2009, pp.2917–2924. New York: IEEE.
27. Zhou Z, Zhang R and Zhu Z. RETRACTED: uncalibrated dynamic visual servoing via multivariate adaptive regression splines and improved incremental extreme learning machine. *ISA T* 2019; 92: 298–314.
28. Raj P, Namboodiri VP and Behera L. Learning to switch CNNs with model agnostic meta learning for fine precision visual servoing, 2020, https://arxiv.org/abs/2007.04645
29. Bi W, Wang X, Tang Z, et al. Avoiding the local minima problem in backpropagation algorithm with modified error function. *IEICE Trans Fund Electron Comm Comput Sci* 2005; 88(12): 3645–3653.
30. Nelles O, Fink A and Isermann R. Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. *IFAC Proc Volume* 2000; 33(15): 845–850.
31. Kalhor A, Araabi BN and Lucas C. Evolving Takagi–Sugeno fuzzy model based on switching to neighboring models. *Appl Soft Comput* 2013; 13(2): 939–946.
32. Nelles O. Local linear model trees for on-line identification of time-variant nonlinear dynamic systems. In: *Proceedings of the international conference on artificial neural networks*, Bochum, 16–19 July 1996, pp.115–120. Berlin; Heidelberg: Springer.
33. Hu G, Gans NR and Dixon WE. *Adaptive visual servo control*. New York: Springer, 2009.
34. Kermorgant O and Chaumette F. Dealing with constraints in sensor-based robot control. *IEEE T Robot* 2014; 30(1): 244–257.
35. Spong MW, Hutchinson S and Vidyasagar M. *Robot modeling and control*. Hoboken, NJ: John Wiley & Sons, 2006.
36. Baerlocher P and Boulic R. Task-priority formulations for the kinematic control of highly redundant articulated

structures. In: *Proceedings of the 1998 IEEE/RSJ international conference on intelligent robots and systems: innovations in theory, practice and applications* (Cat. No. 98CH36190), Victoria, BC, Canada, 17 October 1998, vol. 1, pp.323–329. New York: IEEE.
37. Fruchard M, Morin P and Samson C. A framework for the control of nonholonomic mobile manipulators. *Int J Robot Res* 2006; 25(8): 745–780.
38. Rezaie J, Moshiri B, Rafati A, et al. A modified LOLIMOT algorithm for nonlinear estimation fusion. In: *Proceedings of the 2007 IEEE international conference on information reuse and integration*, Las Vegas, NV, 13–15 August 2007, pp.520–525. New York: IEEE.
39. Kalhor A, Araabi BN and Lucas C. Reducing the number of local linear models in neuro-fuzzy modeling: a split-and-merge clustering approach. *Appl Soft Comput* 2011; 11(8): 5582–5589.
40. Aflakian A, Safaryazdi A, Masouleh MT, et al. Experimental study on the kinematic control of a cable suspended parallel robot for object tracking purpose. *Mechatronics* 2018; 50: 160–176.
41. Marchand É, Spindler F and Chaumette F. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robot Autom Mag* 2005; 12(4): 40–52.
42. Aflakian A, Rastegarpanah A and Stolkin R. Optimized hybrid decoupled visual servoing with supervised learning, 2020, https://figshare.com/articles/journal_contribution/Optimized_Hybrid_Decoupled_Visual_Servoing_simulation_and_code/12980009