Published in IET Computer Vision
Received on 28th November 2013
Revised on 27th June 2014
Accepted on 21st July 2014
doi: 10.1049/iet-cvi.2013.0310


ISSN 1751-9632

# Kernel-based sliding mode control for visual servoing system

*Mahsa Parsapour, Hamid D. Taghirad*

*Advanced Robotics and Automated Systems (ARAS), Industrial Control Center of Excellence (ICEE), Faculty of Electrical and Computer Engineering, K.N. Toosi University of Technology, Tehran, Iran*
*E-mail: taghirad@kntu.ac.ir*

**Abstract:** In this study, a new approach to design a controller for a visual servoing (VS) system is proposed. Kernel-measurement is used to track the motion of a featureless object which is defined as sum of weighted-image value through smooth kernel functions. This approach was used in kernel-based VS (KBVS). To improve the tracking error and expand the stability region, sliding mode control is integrated with kernel measurement. Proportional–integral-type sliding surface is chosen as a suitable manifold to produce the required control effort. Moreover, the stability of this algorithm is analysed via Lyapunov theory and its performance is verified experimentally by implementing the proposed method on a five degrees of freedom industrial robot. Through experimental results, it is shown that the performance of tracking error in the proposed method is more suitable than KBVS, for a wider workspace and when the object is placed near the boundary of the camera's field of view.

## 1 Introduction

One method to simplify the interaction with robots is to use a visual servoing (VS) system. Our main goal in this paper is to track a featureless object through kernel-measurement concept. Kernel measurement was first introduced in [1, 2] and referred to as kernel-based VS (KBVS), for three translational motions and one rotational motion. Later, the KBVS method was presented for different objects in [3]. In the KBVS algorithm, the tracking of an object is accomplished through the sum of weighted image, with the weights given by a smooth kernel function. The extracted features are not taken from some parts of the image such as lines, points and so on, but are taken from full information of the image.

Using algorithms that do not need training is preferred in practice, and KBVS needs almost no training. In the kernel-based approach, tracking the object is accomplished with the kernel-measurement. Sub-processes of 'feature tracking and feedback control' in KBVS are combined together as a superior feature compared with conventional methods of VS techniques known as position-based VS (PBVS) [4–8], image-based VS (IBVS) [4, 9–11] and 2-1/two-dimensional (2D) VS [12–14].

In PBVS, the Cartesian position of the robot end effector is estimated from the positions of extracted features within the image plane. Accordingly, the end-effector position is compared with the desired position and the resulting error is used as a feedback signal to the controller. In IBVS, the desired features from the image are compared with the current features and the resulting difference as error (feedback signal) is fed directly to the controller. The image Jacobian maps the motion of features to the motion of robot end effector. The error and the Jacobian are used to determine the control inputs. Sometimes in the IBVS method, the image Jacobian matrix loses its rank and makes the control ineffective, which is one of the main drawbacks of this method. In 2-1/2D VS, the combination of feature locations and 3D end effector position forms the error feedback signal. This method remedies the problem of IBVS and keeps the control inputs in the task space. In KBVS, the control signal is obtained through the combination of tracking error and derivative of kernel-measurement with respect to time, which is called kernel-measurement Jacobian. This Jacobian matrix maps the task space of robot into the kernel-measurement space. The superiority of KBVS over conventional methods stems from the fact that there is no need to use the image Jacobian, the system dynamics and the Cartesian estimation of the target object. Furthermore, KBVS is only similar to IBVS from point of feature comparison, whereas in KBVS no direct feature is extracted from the image. The stability of the KBVS method is proved by the Lyapunov analysis [2].

Other variations of kernel functions to track the target object are presented in [15–18]. In [15], the feature for tracking is a weighted spatial average of a histogram vector with an isotropic kernel (kernel-weighted histograms) which is similar to the kernel-measurement. Tracking in the desired region is accomplished through minimising a cost function based on the Bhattacharyya coefficient. In [16], a kernel-based tracking algorithm is used in both Cartesian and log-polar coordinates that can estimate the targets' scale and rotation in the tracking process. This method is based on the kernel-based histogram of the target region that uses the Bhattacharyya coefficient. In [17], multiple kernels are used for tracking complex motions where kernels are sensitive only to 1D motion. Furthermore, to converge faster with less

computation, optimal kernels are developed in [18]. Another featureless tracking method is to use the image moments for VS of planar objects [19, 20], where the Jacobian relates different forms of the image moments to the velocity of a moving camera. The image moments are scalar values from the image and provide similar measurement in comparison to the weighted image signal in KBVS. Noting this fact, one may call KBVS a specific case of moment-based VS which is developed for six degrees of freedom (DOF), while KBVS is limited to four DOF.

Nowadays, the performance and stability improvement of VS systems gain more attention among researchers and practitioners. VS systems are uncertain and non-linear by nature. Uncertainties are in the visual parameters, the system dynamics and the external disturbances which lead to tracking error and degrade the overall stability of the VS system. To improve the VS system's performance and the region of stability, we propose combining sliding mode control (SMC) and KBVS. In our approach, firstly, the kernel-measurement is calculated from visual information. To show the calculation of this measurement, kernel functions are analysed in Section 2, describing how weighing the intensities of image pixels is performed. The controller in the SMC method is designed by defining a sliding surface that binds the system states to the surface within a finite time [21]. Hence, we suggest an appropriate sliding surface to guide in pace with kernel-measurement. This way, we deduce the control signal in the task space from the suggested sliding surface.

In contrast to our innovation, SMC is designed based on uncertain system dynamics and is robust to parameter variations and external disturbances such as IBVS in [22–24] and PBVS in [25–27]. In conventional cases, a PD sliding surface fits well to the problem formulation. To increase the tracking performance, a proportional–integral-derivative sliding surface is also implemented in [23]. However, in this paper, we suggest a proportional–integral (PI) sliding surface because of the type of kernel-measurement and the robot joint command nature. By defining the desired sliding surface, the control signal may be easily implemented. The proposed combination of SMC and kernel-measurement is implemented on an industrial robot which has an internal feedback loop. In the presence of the internal feedback loop, the robot accepts the Cartesian velocity or the incremental position commands [8]. As a result, a simplified motion kinematics model can be used for the industrial robot where the controller inputs are the joint velocity command signal.

The rest of this paper is organised as follows: in Section 2, the basic definitions for kernel-measurements in each direction are presented. Section 3 is devoted to the design of SMC based on the kernel-measurement and the motion kinematics model. This section ends by determining the type of stability using Lyapunov analysis. The performance of our proposed method (referred to as KBSMVS) is reported in Section 4 and compared with conventional KBVS for different initial conditions.

## 2 Basic definitions

The control process of KBVS integrated with SMC is schematically depicted in Fig. 1. The control process begins by capturing an image from the camera. After that, the intensities of image signal are transformed to an appropriate format for the kernel-projection process. Then, each pixel of the image signal is weighted through kernel functions, and the current kernel-measurement, $\xi$, is calculated. The difference value of $\xi$ and the goal kernel-measurement, $\xi_0$, forms the tracking error that is fed as input to the sliding mode controller. The desired control signal is obtained as the Cartesian velocity from the sliding surface. Having checked against the singularities of the robot as described in [28], the position command in the joint space will be applied via inverse kinematics. Integrating the joint velocity signal within 10 ms will provide the joint position with desirable accuracy. The target tracking will terminate when the kernel-measurement error meets a stop condition ($\boldsymbol{T}_{\xi}$). In the case of a square shape object, the stop conditions are empirically set to 0.5 for both $x$ and $y$-directions $\left(T_{\xi_x} = T_{\xi_y} = 0.5\right)$, 10 for the $z$-direction $\left(T_{\xi_z} = 10\right)$ and 30 for the $\theta$ orientation $\left(T_{\xi_\theta} = 30\right)$. Note that the empirical stop conditions are related to the types of kernel functions and shape of the target object. In brief, the objective is to obtain a control input that will drive $\boldsymbol{\xi}$ to $\boldsymbol{\xi}_0$ or equivalently the current robot pose, $\boldsymbol{q}$, to the goal pose, $\boldsymbol{q}_0$.

In what follows, $\xi$ is defined for each axis, that is, $[x, y, z, \theta]$, which is the pre-requisite of controller design. In the scheme shown in Fig. 1, the camera is mounted on the robot end-effector. Also, because of the existence of inner feedback loop, the kinematic motion model for robot is applied to all dimensions.

### 2.1 Kernel-measurement in the x – y-directions

In the planar motion, suppose that the robot end-effector is moving parallel to the image plane. The robot configuration denoted by $\boldsymbol{q}_{xy} = [x, y]^{\mathrm{T}}$ is given as

$$\dot{\boldsymbol{q}}_{xy} = \boldsymbol{u}_{xy} \tag{1}$$

$$\boldsymbol{u}_{xy} = [u_x, u_y]^{\mathrm{T}}$$

where $\boldsymbol{u}_{xy}$ is the 2D control signal.

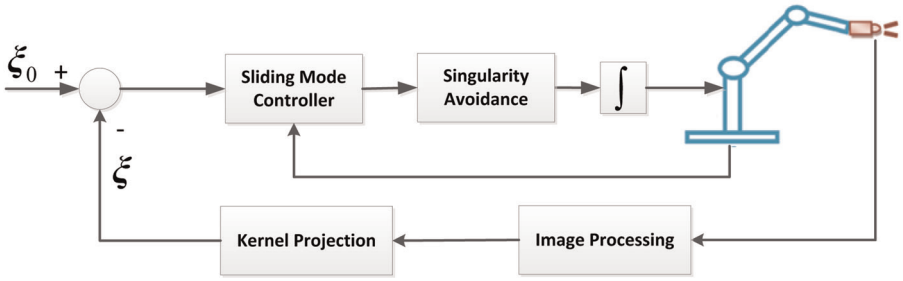


**Fig. 1** *Details of the VS control loop*




doi: 10.1049/iet-cvi.2013.0310

After image capturing, the intensities of the image, $\boldsymbol{I}(\boldsymbol{\omega}, \boldsymbol{q}_{xy})$, are weighted with a continuous and differentiable function, $\boldsymbol{K}_{xy} = [K_x, K_y]^{\mathrm{T}}$. The kernel-measurement in the $x-y$-directions is defined as follows

$$\boldsymbol{\xi}_{xy} = \int \boldsymbol{K}_{xy}(\boldsymbol{\omega})\boldsymbol{I}(\boldsymbol{\omega}, \boldsymbol{q}_{xy})\,\mathrm{d}\boldsymbol{\omega} \tag{2}$$

where the image signal $\boldsymbol{I} \in \mathbb{R}^2$ is related to the spatial index $\boldsymbol{\omega} \in \mathbb{R}^2$ and the Cartesian position, $\boldsymbol{q}_{xy}$. To define the goal kernel-measurement, $\boldsymbol{\xi}_{xy_0}$, the goal image is taken when the target is placed at the centre of the image frame. By comparison of $\boldsymbol{\xi}_{xy}$ and $\boldsymbol{\xi}_{xy_0}$, we may command the motion of the end-effector in a horizontal space. When the target tracking is completed, we would have approximately $\boldsymbol{q}_{xy}$ equal to $\boldsymbol{q}_{xy_0}$. The image signal in each position is dependent to the goal signal $\boldsymbol{I}_0$ and can be written as (3). Changing the position of the object in the image is similar to the condition where all pixels of the goal image signal, $\boldsymbol{I}_0$, are moved, but remember that the robot vision is stationary

$$\boldsymbol{I}(\boldsymbol{\omega}, \boldsymbol{q}_{xy}) = \boldsymbol{I}_0(\boldsymbol{\omega} - \boldsymbol{q}_{xy}) \tag{3}$$

Through a change of variable in the spatial index, $\omega$, as $\bar{\boldsymbol{\omega}} = \boldsymbol{\omega} - \boldsymbol{q}_{xy}$, the kernel-measurement can be written as (4). This formulation is useful in the controller synthesis

$$\begin{aligned}\boldsymbol{\xi}_{xy} &= \int \boldsymbol{K}_{xy}(\boldsymbol{\omega})\boldsymbol{I}_0(\boldsymbol{\omega} - \boldsymbol{q}_{xy})\,\mathrm{d}\boldsymbol{\omega} \\ &= \int \boldsymbol{K}_{xy}(\bar{\boldsymbol{\omega}} + \boldsymbol{q}_{xy})\boldsymbol{I}_0(\bar{\boldsymbol{\omega}})\,\mathrm{d}\bar{\boldsymbol{\omega}}\end{aligned} \tag{4}$$

In the goal position $\left(\boldsymbol{q}_{xy_0}\right)$, the kernel measurement is $\boldsymbol{\xi}_{xy_0} = \boldsymbol{\xi}_{xy}$. The kernel functions are selected as Gaussian type formulated in (5), and they are independent from each other. This makes the derivative with respect to $x$ or $y$ of $\boldsymbol{K}_{xy}$ as a diagonal matrix, and hence the controllers in the $x-y$-plane will be independent. The mean value of $\mu_x = \mu_y = -100$, and the standard deviation of $\sigma_x = \sigma_y = 70$ are selected as proposed in [3]

$$\begin{aligned}K_x(\omega) &= \frac{1}{\sqrt{2\pi}\sigma_x}\mathrm{e}^{-\left((\omega_1-\mu_x)^2\right)/2\sigma_x^2\right)} \\ K_y(\omega) &= \frac{1}{\sqrt{2\pi}\sigma_y}\mathrm{e}^{-\left((\omega_2-\mu_y)^2\right)/2\sigma_y^2\right)}\end{aligned} \tag{5}$$

Fig. 2 visualises the kernel functions' behaviour in the $x$- and $y$-directions. In this figure, the width and the height axes are related to the dimensions of the image matrix, in which $x$ denotes the width and $y$ denotes the height of the image signal. The values of $K_x$ and $K_y$, are given in two subfigures, respectively. The initial condition for integration is considered in the corner of the image plane denoted by [width, height, amplitude] = [0, 0, 0].

### 2.2 Kernel-measurement in the z-direction

For vertical motion of the end-effector, image scaling is the only criteria to guide the end-effector to translate in the $z$-direction. Thus, the $z$-axis is assumed to be parallel to the camera's optical axis. Therefore the magnitude of Fourier transform (FT) is used as an appropriate signal for weighting the image. The characteristic of FT decouples scaling from traversal motions. Rotation and scaling appear in the FT as variations in the magnitude, and translation in the image is seen as the change of phase in the FT. In the vertical direction, the change of configuration is given as follows

$$\dot{z} = u_z \tag{6}$$

where $u_z$ is the corresponding control input. Let $\boldsymbol{I}_0$ and $\boldsymbol{F}_0$ denote the goal image and the goal magnitude of FT, respectively. Other positions of the camera are the scaling format of $\boldsymbol{I}_0$ according to the following equation

$$\boldsymbol{I}(\boldsymbol{\omega}, z) = \boldsymbol{I}_0\left(\frac{\boldsymbol{\omega}}{z}\right) \tag{7}$$

The magnitudes of FT at the goal $\boldsymbol{F}_0$, and other positions $\boldsymbol{F}$ are related by (8) and $\boldsymbol{v} \in \mathbb{R}^2$ is the spatial index of FT

$$\boldsymbol{F}(\boldsymbol{v}, z) = z\boldsymbol{F}_0(z\boldsymbol{v}) \tag{8}$$

The kernel-measurement in the $z$-direction is defined as (9). By positioning the end-effector of the robot at a specific height, the goal kernel-measurement, $\xi_{z_0}$, is derived. The end-effector of the robot is controlled by comparing $\xi_z$ (the kernel-measurement in each position) and $\xi_{z_0}$

$$\xi_z = \int K_z(\boldsymbol{v})\boldsymbol{F}(\boldsymbol{v}, z)\,\mathrm{d}\boldsymbol{v} \tag{9}$$

Through a change of variable in the spatial index $\boldsymbol{v}$ as $\bar{\boldsymbol{v}} = z\boldsymbol{v}$, the depth kernel-measurement may be written as (10). This formulation is directly used in the control signal. By derivation of the kernel-measurement with respect to time,

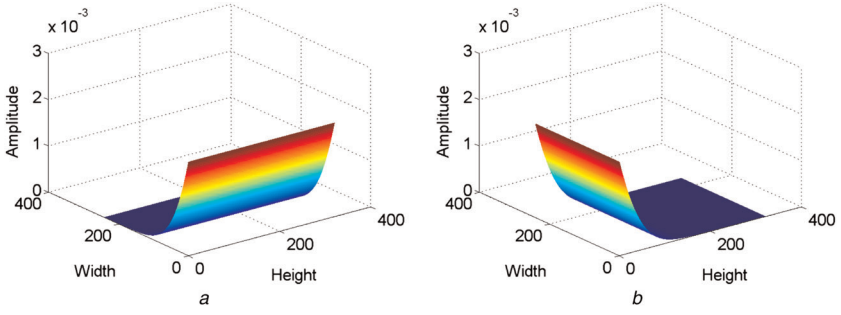


**Fig. 2** *Kernel functions in*
*a* $x$-direction
*b* $y$-direction

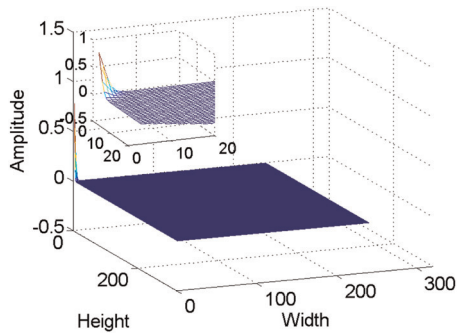


**Fig. 3** *Kernel function in the z-direction*

the FT of the image will not be changed and this fact simplifies the controller design

$$\xi_z = \int zK_z\left(\frac{\bar{\boldsymbol{v}}}{z}\right)\boldsymbol{F}_0(\bar{\boldsymbol{v}})\,\mathrm{d}\bar{\boldsymbol{v}} \tag{10}$$

Gaussian kernel function in the $z$-direction is selected as the proposed in [3], denoted by (11). The performance of this function is shown in Fig. 3

$$K_z(\boldsymbol{v}) = \mathrm{e}^{-(1/8)\|\boldsymbol{v}\|^2} \tag{11}$$

The magnitude of the FT matrix has the same dimension as the image matrix. In the depth motion, most of the information is located at the corners of FT matrix. The FT magnitude in different positions of a square object is shown in Fig. 4. The black areas have zero value and the white areas have a value of one. When the end-effector of the robot is located higher than the normal condition, the object will be seen smaller and more pixels in the corner of FT matrix will have ones compared with the normal condition. However, in the opposite side, when the robot end-effector is moved closer to the object, the FT matrix will have less white pixels than the normal condition. Hence, the way in which $K_z$ magnifies such corners is reasonable. If the target object is symmetrical, the FT of the image will be equal at each corner of the matrix; however, weighting one of these corners will be sufficient. For different shapes of the target object, we can use a uniform function and it is better to increase the standard deviation of Gaussian function to attenuate the sharpness of weighted corner.

### 2.3 Kernel-measurement in the θ orientation

To orient properly, the rotation of the camera about the $z$-axis (optical axis) is considered. Image rotation is detected through the magnitude of FT. Hence, the robot dynamics is according to

$$\dot{\theta} = u_\theta \tag{12}$$

where $u_\theta$ is the control input. Let $\boldsymbol{I}_0$ and $\boldsymbol{F}_0$ denote the image and the magnitude of FT at the goal, where $\theta = 0$ (the goal position of the end-effector). At any orientation of the camera, the image signal $\boldsymbol{I}$ is the scaled format of $\boldsymbol{I}_0$ as the following equation

$$\boldsymbol{I}(\boldsymbol{\omega},\,\theta) = \boldsymbol{I}_0(\boldsymbol{R}_\theta\boldsymbol{\omega}) \tag{13}$$

$$\boldsymbol{R}_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

The magnitudes of FT at the goal, $\boldsymbol{F}_0$, and other positions, $\boldsymbol{F}$, are related by (14) and $\boldsymbol{v} \in \mathbb{R}^2$ is the spatial index of FT

$$\boldsymbol{F}(\boldsymbol{v},\,\theta) = \boldsymbol{F}_0\left(\boldsymbol{R}_\theta^{\mathrm{T}}\boldsymbol{v}\right) \tag{14}$$

The kernel-measurement in $\theta$ orientation is defined as (15). By positioning the end-effector of the robot at $\theta = 0$, the goal kernel-measurement $\xi_{\theta_0} = \int K_\theta(\boldsymbol{v})\boldsymbol{F}_0(\boldsymbol{v},\,\theta)\,\mathrm{d}\boldsymbol{v}$ is derived. The end-effector of the robot is controlled by comparing $\xi_\theta$ (the kernel-measurement in each orientation) and $\xi_{\theta_0}$

$$\begin{aligned}\xi_\theta &= \int K_\theta(\boldsymbol{v},\,\theta)\boldsymbol{F}(\boldsymbol{v},\,\theta)\,\mathrm{d}\boldsymbol{v} \\ &= \int K_\theta(\boldsymbol{R}_\theta\bar{\boldsymbol{v}})\boldsymbol{F}_0(\bar{\boldsymbol{v}})\,\mathrm{d}\bar{\boldsymbol{v}}\end{aligned} \tag{15}$$

where $\bar{\boldsymbol{v}} = \boldsymbol{R}_\theta^{\mathrm{T}}\boldsymbol{v}$. The Gaussian kernel function in the $\theta$ dimension is selected as the following equation [3]

$$K_\theta(\boldsymbol{v}) = \mathrm{e}^{-(1/8)v_1^2} + \mathrm{e}^{-(1/8)v_2^2} \tag{16}$$

The performance of rotational kernel function is depicted in Fig. 5. The axes of Fig. 5 are like the $z$-direction. One of the properties of FT is that the Fourier of a vertical column of the image matrix is a horizontal row; a vertical line maps into a horizontal line and vice versa. The number of lines is proportional to the sides of the object. By rotating the object, these lines will rotated with respect to the rotation of object.

The magnitude of FT of the image is weighted in one corner and in two side planes (width and height axes). The magnitude of FT of a square object in various rotations is shown in Fig. 6. This object is rotated from 0° to 90°. It is obvious that a square object has the same pattern after 90° rotation. In Fig. 6, two rotating lines correspond to the sides

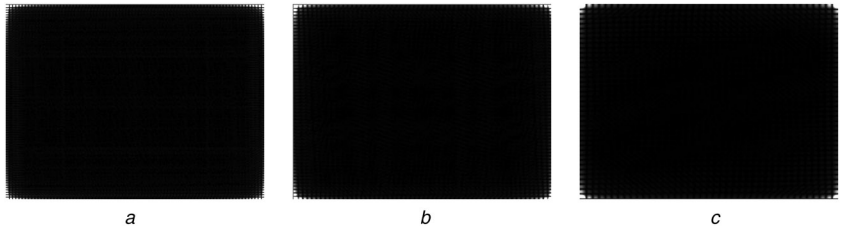


**Fig. 4** *Magnitude of Fourier in the z-direction*

*a* Greater object
*b* Normal object
*c* Smaller object




doi: 10.1049/iet-cvi.2013.0310

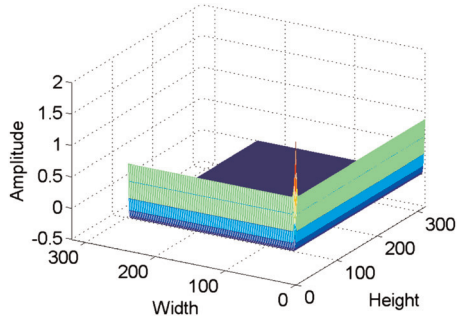


**Fig. 5** *Kernel function in the θ-orientation*

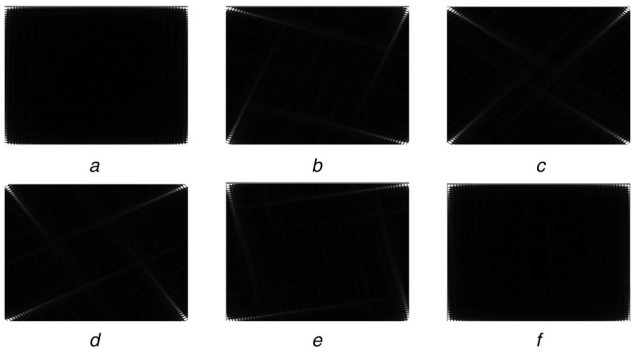


**Fig. 6** *Magnitude of Fourier in the θ-orientation*
*a* 0°
*b* 20°
*c* 40°
*d* 60°
*e* 80°
*f* 90°

of the square object and at each position they are perpendicular to each other. By rotating the target object, such lines will rotate with respect to the corners of the matrix. To determine the rotation, two sides are used for weighting, and the performance of the kernel function seems logical. Moreover, the rotation is studied at a specific height, and information in the corner of image in the vertical axis will be constant (the amount of intensities of pixels will not change).

## 3 Sliding mode control

SMC is widely used in the control of non-linear uncertain systems. SMC has some interesting and important characteristics which may not be achieved via other methods. When a system is at a sliding surface, the order of the system is reduced according to the order of the sliding surface and the resulting system is robust against variations of parameters and disturbances. In this method, it is not necessary to have a precise system model and the control algorithm is easily implementable. All these properties make SMC an ideal candidate to control a non-linear system. In such cases, a sliding surface is defined for the system, and the goal is to enforce states of the system in a finite amount of time on the sliding manifold, and to be maintained on it for future times. In general, controlling first-order systems is much simpler than higher order systems. Transforming a higher order system to a first-order one may be realised by defining a first-order sliding surface. For some class of systems, the SMC approach stabilises the system and provides a uniform performance in the presence of uncertainty. The desired control process is to conciliate between the performance of tracking and the uncertainty of parameters [29].

In most VS works, a PD type sliding surface is selected, and in most cases, the system dynamics is used to design the sliding mode controller. The main contribution of this paper is to integrate SMC with kernel-measurement and to use the simplified system dynamics to design the controller. The details of the proposed approach are given as follows.

### 3.1 Controller design

In sliding mode, the dynamic behaviour of a system is defined by a sliding surface. Here, the error of kernel-measurement is used to define the desired sliding surface and to generate the control signal. The main goal is to track an object, either stationary or moving. Definition of the sliding surface is performed prior to the design of the controller. Since our system kinematics is modelled in the velocity layer, we need to produce a control signal that also has this nature. Moreover, the type of kernel-measurement is important in defining the suitable sliding surface. Based on these two features, the sliding surface is proposed as PI type in (17) that stabilises the KBVS system. PI-type selection is one of the possible structures for our proposed system and other designs with more effective surfaces may also be considered with other considerations

$$\boldsymbol{S} = (1 + \boldsymbol{\lambda} \int) \boldsymbol{e}_{\xi} = \boldsymbol{e}_{\xi}(\boldsymbol{\omega}) + \boldsymbol{\lambda} \int \boldsymbol{e}_{\xi}(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} \tag{17}$$

$$\lambda = \begin{pmatrix} \lambda_x & 0 & 0 & 0 \\ 0 & \lambda_y & 0 & 0 \\ 0 & 0 & \lambda_z & 0 \\ 0 & 0 & 0 & \lambda_\theta \end{pmatrix}$$

where $\boldsymbol{e}_{\xi}$ is the difference between the desired kernel-measurement, $\boldsymbol{\xi}_0 = [\xi_{x_0}, \xi_{y_0}, \xi_{z_0}, \xi_{\theta_0}]^{\mathrm{T}}$ and the current kernel-measurement, $\xi = [\xi_x, \xi_y, \xi_z, \xi_\theta]^{\mathrm{T}}$. Furthermore, this surface is a four-tuple vector $\boldsymbol{S} = [S_x, S_y, S_z, S_\theta]^{\mathrm{T}}$. The controller shall be designed in such a way that all the trajectories of the system converge to the sliding manifold $\boldsymbol{S} = \boldsymbol{e}_{\xi} + \boldsymbol{\lambda} \int \boldsymbol{e}_{\xi} = 0$. The rate of convergence of the state trajectories, $\boldsymbol{\lambda}$, has to be tuned at each direction, while $\boldsymbol{\lambda}_i > 0$. When the tracking error approaches zero, the sliding surface will also converge to zero. To obtain the control signal one may find the derivative of (17) as follows

$$\dot{\boldsymbol{S}} = \frac{\mathrm{d}\boldsymbol{S}}{\mathrm{d}t} = \boldsymbol{e}_{\xi} + \boldsymbol{\lambda} \boldsymbol{e}_{\xi} = \dot{\boldsymbol{\xi}} + \boldsymbol{\lambda} \boldsymbol{e}_{\xi}(\boldsymbol{\omega}) \tag{18}$$

To compute the derivative of the sliding surface, it is necessary to calculate the derivative of kernel-measurement with respect to the position of the robot end-effector, as derived in (19), where $\boldsymbol{J}$ is the kernel-measurement Jacobian matrix, which maps the task space of the robot into the kernel-measurement space

$$\dot{\boldsymbol{\xi}} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} = \boldsymbol{J} \dot{\boldsymbol{q}} \tag{19}$$

$$\boldsymbol{J} = \begin{pmatrix} J_x & 0 & 0 & 0 \\ 0 & J_y & 0 & 0 \\ 0 & 0 & J_z & 0 \\ 0 & 0 & 0 & J_\theta \end{pmatrix}$$

The only varying signal in the kernel-measurement is the captured image. Derivation of the image is complex and it takes a lot of time. Hence, by changing the variable in (4), (10) and (15), the derivation may be converted to the kernel functions. The Jacobian at each dimension is determined by (20), while after these derivations the variables are rolled back to the original ones

$$J_x = -\int \frac{(\omega_1 - \mu_x)}{\sigma_x^2} K_x(\boldsymbol{\omega}) \boldsymbol{I}(\boldsymbol{\omega}, t)\, \mathrm{d}\boldsymbol{\omega}$$

$$J_y = -\int \frac{(\omega_2 - \mu_y)}{\sigma_x^2} K_y(\boldsymbol{\omega}) \boldsymbol{I}(\boldsymbol{\omega}, t)\, \mathrm{d}\boldsymbol{\omega}$$

$$J_z = -\int \frac{1}{4} ||\boldsymbol{v}|| K_z(\boldsymbol{v}) \boldsymbol{F}(\boldsymbol{v}, z)\, \mathrm{d}\boldsymbol{v}$$

$$J_\theta = -\int \frac{1}{4} \left[\frac{1}{v_1}, \frac{1}{v_2}\right] \left[v_1 e^{-(1/8)v_1^2}, v_2 e^{-(1/8)v_2^2}\right]^{\mathrm{T}} \boldsymbol{F}(\boldsymbol{v}, \theta)\, \mathrm{d}\boldsymbol{v} \tag{20}$$

In (19), $\dot{\boldsymbol{q}}$ is equal to the control signal, $\boldsymbol{u}$, which is a four tuple vector as $\boldsymbol{u} = [u_x, u_y, u_z, u_\theta]^{\mathrm{T}}$. By substituting (19) in (18), $\dot{\boldsymbol{S}}$ is obtained as follows

$$\dot{\boldsymbol{S}} = \dot{\boldsymbol{\xi}} + \boldsymbol{\lambda} \boldsymbol{e}_\xi = \boldsymbol{J}\boldsymbol{u} + \boldsymbol{\lambda}(\boldsymbol{\xi} - \boldsymbol{\xi}_0) \tag{21}$$

The control signal as (22) is obtained from $\dot{\boldsymbol{S}}$. The first term in the control signal states that the ideal condition is $\boldsymbol{S} = \dot{\boldsymbol{S}} = 0$, meaning that the trajectories of system shall lie on the sliding surface. A second switching term ($\boldsymbol{C}\mathrm{sgn}(\boldsymbol{S})$) is added to the $\boldsymbol{u}$ expression to force the current trajectory to stay on the sliding surface. In other words, when the object moves, the conditions drift from $\boldsymbol{S} = \dot{\boldsymbol{S}} = 0$, and the proposed algorithm is able to return the robot end-effector to the desired manifold. In (22), the parameter $\boldsymbol{C}$ at each direction is selected with respect to the change of kernel-measurement at the corresponding direction. Furthermore, to guarantee the stability of the system, the sign of $\boldsymbol{C}$ is set

$$\boldsymbol{c} = -\boldsymbol{\lambda}\boldsymbol{J}^{-1}(\boldsymbol{\xi} - \boldsymbol{\xi}_0) + \boldsymbol{C}\mathrm{sgn}(\boldsymbol{S}) \tag{22}$$

$$\boldsymbol{C} = \begin{pmatrix} C_x & 0 & 0 & 0 \\ 0 & C_y & 0 & 0 \\ 0 & 0 & C_z & 0 \\ 0 & 0 & 0 & C_\theta \end{pmatrix}$$

### 3.2 Stability analysis

In the SMC approach, the candidate Lyapunov function is considered as $\boldsymbol{V} = 1/2\boldsymbol{S}\boldsymbol{S}^{\mathrm{T}}$, which is a $4 \times 4$ positive definite matrix, and in the whole space $\boldsymbol{V} > \boldsymbol{0}$ and $\boldsymbol{V}(\boldsymbol{0}) = \boldsymbol{0}$. To deduce the type of stability, one needs to analyse the derivative of the Lyapunov function as (23). The variation of $\dot{\boldsymbol{V}}$ is related to the Jacobian matrix and the absolute value of sliding surface

$$\dot{\boldsymbol{V}} = \boldsymbol{S}\dot{\boldsymbol{S}}^{\mathrm{T}} = \boldsymbol{S}(\boldsymbol{J}\dot{\boldsymbol{q}} + \boldsymbol{\lambda}(\boldsymbol{\xi} - \boldsymbol{\xi}_0))^{\mathrm{T}}$$

$$\dot{\boldsymbol{V}} = \boldsymbol{S}(\boldsymbol{C}\boldsymbol{J}\mathrm{sgn}(\boldsymbol{S}))^{\mathrm{T}} = |\boldsymbol{S}|_{4\times 4}\boldsymbol{J}\boldsymbol{C} \tag{23}$$

To have a stable algorithm, $\dot{\boldsymbol{V}}$ must be negative definite. We know that the image signal in both the spatial and the frequency domains is positive. To determine the sign of the Jacobian matrix, we need to study the derivative of kernel function with respect to the end-effector position $\partial \boldsymbol{K}(\cdot)/\partial \boldsymbol{q}_i$ as given in the following equation

$$\frac{\partial \boldsymbol{K}(\cdot)}{\partial x} = -\frac{(\omega_1 - \mu_x)}{\sigma_x^2} K_x(\cdot)$$

$$\frac{\partial \boldsymbol{K}(\cdot)}{\partial y} = -\frac{(\omega_2 - \mu_y)}{\sigma_y^2} K_y(\cdot)$$

$$\frac{\partial \boldsymbol{K}(\cdot)}{\partial z} = -\frac{1}{4} ||\boldsymbol{v}|| K_z(\cdot) \tag{24}$$

$$\frac{\partial \boldsymbol{K}(\cdot)}{\partial \theta} = -\frac{1}{4} K_\theta(\cdot)$$

The mean value of the Gaussian function in the $x$- and $y$-directions are negative, the kernel functions are positive, and $\omega_1, \omega_2 \in \mathbb{R}^+$. As a result, $\partial \boldsymbol{K}(\cdot)/\partial x$ and $\partial \boldsymbol{K}(\cdot)/\partial y$ will remain negative. In the $z$ and $\theta$ dimensions, the respective kernel functions are positive, and $v_1, v_2 \in \mathbb{R}^+$. Hence, $\partial \boldsymbol{K}(\cdot)/\partial z$ and $\partial \boldsymbol{K}(\cdot)/\partial \theta$ will remain negative. In light of this discussion, to have a stable algorithm $\left(\dot{\boldsymbol{V}} < 0\right)$, we need to set the parameter $\boldsymbol{C}$ as strictly positive at each direction, to assure the asymptotic stability of the algorithm. Notice that this analysis provides only local asymptotic stability and it is limited to the reachable workspace of the robot.

In theory, the assurance of asymptotic stability will result in zero tracking error on the sliding surface. However, in practice, a stop condition, $\boldsymbol{T}_\xi$, is set for each direction, such that if the absolute value of the tracking error is smaller than $\boldsymbol{T}_\xi$, the robot will not be commanded further and will stop. In such case, the derivative of the Lyapunov function will be smaller than a specific limit $\left(\dot{\boldsymbol{V}} < |\boldsymbol{T}_1|\boldsymbol{J}\boldsymbol{C}\right)$. At the same time, the kernel-measurement Jacobian is negative for each dimension. Therefore the value of $|\boldsymbol{T}_1|\boldsymbol{J}\boldsymbol{C}$ is a lower bound for negative $\dot{\boldsymbol{V}}$. $\boldsymbol{T}_\xi$ constraint the stability to the uniformly-ultimate bounded (UUB) type [19]. In the UUB type, when the absolute value of the tracking error is greater than $\boldsymbol{T}_\xi$, $\dot{\boldsymbol{V}}$ is negative definite and will provide decreasing $|\boldsymbol{S}|$ into the set $\boldsymbol{T}_1$. However, in the set $\boldsymbol{T}_1$, the control signal is zero. Hence, the negative definiteness of $\dot{\boldsymbol{V}}$ cannot be guaranteed in the set $\boldsymbol{T}_1$. It is definitely very difficult to quantitatively determine the set $\boldsymbol{T}_1$ because of the integral term in the PI sliding surface, referred to in (17).

## 4 Experimental results

Fig. 7 shows the components of an experimental setup where the robot is a 5 DOF RV-2AJ industrial manipulator, augmented with one DOF linear gantry and made by Mitsubishi Company. A camera is mounted on the manipulator end-effector from Unibrain Company. The camera captures

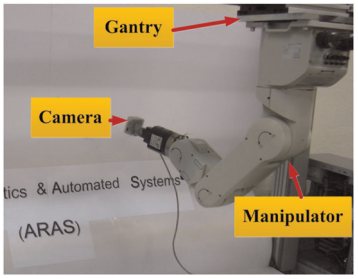


**Fig. 7** *Experimental setup configuration*




doi: 10.1049/iet-cvi.2013.0310

images with a resolution of 320 × 240 pixels. Movement is detected based on 30 frames per second. The focal length of the installed wide lens is 2.1 mm. For the normal distance between the camera and the object in the goal pose, each pixel is approximately 1.8 mm. To program the robot, a Pentium IV PC with 1.84 GHz CPU and 1 GB RAM is used. The actual algorithm is developed on Microsoft Visual Studio 2008 environment utilising OpenCV library.

### 4.1 Considerations

In the implementation of the proposed controllers, several assumptions are made as follows. Note that the spatial indices of the image as $\omega$ and $\boldsymbol{v}$ are all defined in the continuous time domain, but in the experiments they are applied as the discrete time variables. The image captured from the camera is finitely saved as 320 × 240 pixels, hence the kernel functions have to be selected finitely as we considered in [3]. Therefore, the continuous integral operator will change to the equivalent discrete summation operator, according to the following equation

$$\boldsymbol{\xi} = \boldsymbol{K}(\boldsymbol{\omega})\boldsymbol{I}(\boldsymbol{\omega}, \boldsymbol{q})\mathrm{d}\boldsymbol{\omega} \simeq \sum \boldsymbol{K}(\boldsymbol{\omega})\boldsymbol{I}(\boldsymbol{\omega}, \boldsymbol{q}) \tag{25}$$

The major consideration we observe in selecting the target object and the background of workspace is that both need to be planar in shape. This is because of the limitations of Gaussian kernel functions in (5), (11) and (16), and also the wide angle of the camera lens. A wide lens distorts the location of pixels near the edges of the frame so that straight lines farther from the centre of the lens are bent more than those closer to centre. This phenomenon is referred to as the barrel effect [30]. To have a natural representation of the object from the camera image, it is necessary to remove radial distortion. Theoretically, such elimination is possible, but in practice, elimination introduces errors which propagate to the kernel projection process output. Hence, the values have uncertainties resulting in inaccurately locating the end effector of the robot. The object background in the image will also contribute to more uncertainty in positioning. In the case of using non-planar objects, this mentioned problem is magnified. Therefore, by using a planar object and a planar background, the errors significantly decrease. We use a white object with a black background to have a better contrast of the object in the image. When the captured image is transformed to a grey-scale image, variations of light can be sensed due to the noise produced by artificial lighting of the environment. To minimise the effect of image noise, we use a threshold while other methods are also applicable to reduce the noise. The original grey-scale image pixel values are in the range between 0 and 255. Suppose that we want to discriminate the object boundary from the image by rejecting pixels below the value of $T$, as the threshold level. The problem with choosing a suitable value of $T$ is that for large values of $T$, some information in the image will be lost and for small values of $T$, some white points in the image will remain as noise. The extra white pixels will hinder the kernel measurement calculation and hence, the command signal will have errors. In [31], different ways to select $T$ is proposed, but we obtained the value of $T$ empirically. This way, we obtain a black and a white image of our target object.

The number of motion variables is equivalent to the DOF of the system. Each DOF has a corresponding kernel function, each of which contributes to the control of system motion. The only proposed kernel-based approach with $\boldsymbol{K} = [K_x, K_y, K_z, K_\theta]$ is capable of tracking the object in four dimensions. Our industrial robot can move in 5 DOFs. The fifth DOF is the tilt of the end-effector around the $y$-axis. To control the tilt of the robot's end effector, it is necessary to have a kernel function which can model the tilting moves. Thus, the tracking of the object is highly dependent on the limitations of kernel function. Note that the behaviour of $\boldsymbol{K}_i$s is defined at the end of Sections 2.1–2.3, which show how the tracking of the object is related in the appropriate dimension from the image pixel intensities.

As mentioned before, the FT of the image is invariant to any translational motion. We expect the kernel-measurement in one dimension to be independent from the kernel measurement in other dimensions. However, because of the type of kernel function, the extracted value of one dimension is sensitive to the change of other dimensional positions. Therefore, we consider priority in commanding the robot end effector. Since the FT of the image is invariant to the translational motion, first the $z$-direction motion is commanded. However, due to the fact that the vertical variations are seen in the corner of FT magnitude matrix, we cannot simultaneously compensate for both depth and roll. Hence, the second correction is in the $\theta$ orientation. Finally, tracking in the $x$-$y$ plane is performed.

In the following subsections, the performance of KBSMVS is verified in several experiments. The target object is stationary in all trials and the goal is to set the visual hand regulation. There is no specific recommendation for how to define the controller parameters. Hence, we conducted several experiments to obtain the best values of control signals as in (22). Finally, from all the experiments $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ for each dimension are adjusted empirically as $C_x = C_y = C_z = C_\theta = 1$, $\lambda_x = \lambda_y = 1$, $\lambda_z = 20$ and $\lambda_\theta = 3.5$.

### 4.2 $x - y$ Translational motion

In this case, we aim to analyse the performance of the proposed algorithm in the $x$-$y$ directions for a stationary object. The robot will stop when the kernel-measurement error reaches 1.24% of $\xi_{x_0}$ in the $x$-direction and 8.74% of $\xi_{y_0}$ in the $y$-direction. This experiment is repeated eight times with random initial conditions. In Fig. 8 (first column), the trial results are depicted and the initial conditions are highlighted with a circle. The initial conditions stand for the distance between the initial pose and the goal pose. The statistical ranges of these conditions are summarised in Table 1. One of the experiments is represented by a solid line. The 2-Norm of kernel-measurement error is also illustrated in Fig. 8*a*. Based on the results in Fig. 8*d*, the mean value and the standard deviation of the final pose errors are obtained in Table 1. To demonstrate the stability in the $x$-$y$ plane, $\dot{\boldsymbol{V}}$ is shown in Fig. 8*j*. To neatly verify the results, only the highlighted initial condition (solid line), the sliding surface and the derivative of the Lyapunov function are shown in Figs. 8*g* and *j*. The kernel measurement error causes the sliding surface to start with a non-zero value. The control signal forces $\boldsymbol{S}_{xy}$ and $\boldsymbol{e}_{\xi_{xy}}$ to decrease to zero. The performance of $\dot{\boldsymbol{V}}$ is related to $\boldsymbol{S}_{xy}$ and $\boldsymbol{J}_{xy}$ per (23). In both directions, $\dot{\boldsymbol{V}}$ is negative and UUB stability is apparent. The three images in Fig. 8*m* show the goal view, the initial picture and the final positional error of the highlighted trial; that is, the right picture shows the convergence to the desired condition as $\boldsymbol{\xi}_{xy} \rightarrow \boldsymbol{\xi}_{xy_0}$.

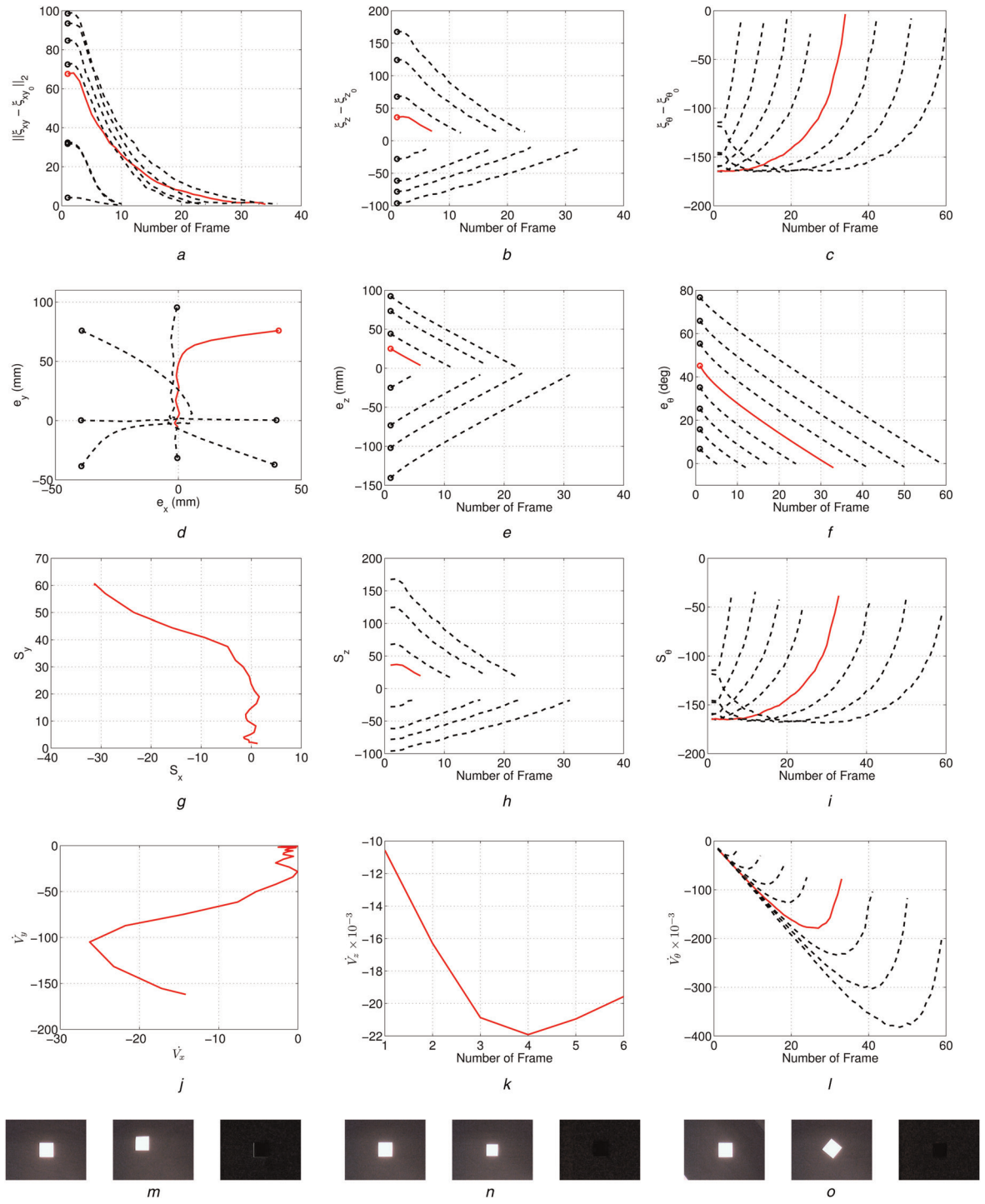


**Fig. 8** *Independent motions*

*a, d, g, j, m* (First column) 2D KBSMVS experiments
*b, e, h, k, n* (Second column) Depth KBSMVS experiments
*c, f, i, l, o* (Third column) Roll KBSMVS experiments
The trial shown by a solid line is explored in more detail in other figures
*a, b, c* (First row) Kernel measurement errors
*d, e, f* (Second row) Position errors
*g, h, i* (Third row) Sliding surface
*j, k, l* (Forth row) Derivative of Lyapunov function
*m, n, o* (Fifth row) For the same trial displayed in the solid line, the images represent: (from left to right) the image at the goal location, the initial location and the difference between the goal and the final image

### *4.3 Depth translational motion*

In this case, the robot is allowed to translate vertically along the camera's optical axis. Here, the correction in the depth motion is seen as the image magnification. We conducted eight trials with random initial positions with a mean initial distance of 72 mm and a standard deviation of 40 mm. The trials were stopped when the tracking error was 4.77% of




doi: 10.1049/iet-cvi.2013.0310

**Table 1** Initial and final position errors for 2D experiments

| Pose error | Initial condition | | Final error | |
|---|---|---|---|---|
| | Mean | Std. | Mean | Std. |
| $e_x$, mm | 29.9 | 18.2 | 2.5 | 1.9 |
| $e_y$, mm | 44.4 | 35.3 | 2.3 | 1.7 |

$\xi_{z_0}$. As per Fig. 8*e*, the robot tracks the target with the mean position error of 5.4 mm and the standard deviation of 2.8 mm. $e_{\xi_z}$, $e_z$ and $S_z$ in Figs. 8*b*, *e* and *h* have the same performance and sign because of the Gaussian kernel function. The empty space, where the sliding surfaces do not converge to zero, is due to set $\boldsymbol{T}_1$ mentioned in Section 2.2. The same analysis, such as the 2D case, can be considered for $\dot{\boldsymbol{V}}$ and the type of stability.

### 4.4 θ-orientation motion

In this case, the robot is allowed to rotate about the camera's optical axis. Analogous to the depth case, the magnitude of FT shows the correction in the roll motion. We conducted eight trials with random initial positions. In these experiments, the mean initial rotation is 45° with the standard deviation of 24.49°. The trial is stopped when the tracking error is below 8.11 with respect to $\xi_{\theta_0}$. In Fig. 8*f*, the orientation converged to the target with a mean rotational error of 1.22° and a standard deviation of 0.72°. For a square-shaped object, $\xi_\theta$ will have the same pattern after 90°. The symmetric point for $\xi_\theta$ is in 45°, and after this point a return in the respective curve can be seen in Fig. 8*c*. Hence, the initial conditions are considered between 0° and 90°. $e_{\xi_\theta}$ and $S_\theta$ have same performance and sign because of the Gaussian kernel function in Figs. 8*c* and *i*. The empty space, where the sliding surfaces do not converge to zero, is due to set $\boldsymbol{T}_1$ mentioned in Section 2.2. The same results, as in previous cases, may be inferred for $\dot{\boldsymbol{V}}$ and the type of stability.

### 4.5 4D motion

In this section, the robot is allowed to move simultaneously in all three translational DOFs and one rotational DOF. The purpose of these experiments is to show how the tracking is performed in 4D motion. According to Section 4.1, firstly, depth is corrected, secondly, roll and finally, the 2D translational motion is performed. The robot will stop when the tracking error reaches 2.48% in the *x*-direction, 17.48% in the *y*-direction, 4.77% in the *z*-direction and 10.83% in the *θ*-orientation of $\xi_0$. This experiment is also repeated eight times with random initial conditions that are highlighted with circles in Fig. 9. Fig. 9*i* shows different situations of objects for one of these eight trials.

With respect to the tracking error as shown in Fig. 9*a*, $S_z$ is moved from a definite distance and the control signal tries to decrease the depth error to the stop criteria. Furthermore, $S_\theta$ is shown in Fig. 9*b* which shows a similar pattern, and then for $\boldsymbol{S}_{xy}$ this process is repeated until the tracking error reaches the desired threshold. In states where the sliding surfaces are zero or constant, the robot is not moving in the corresponding direction. When the object is located near the edge of the

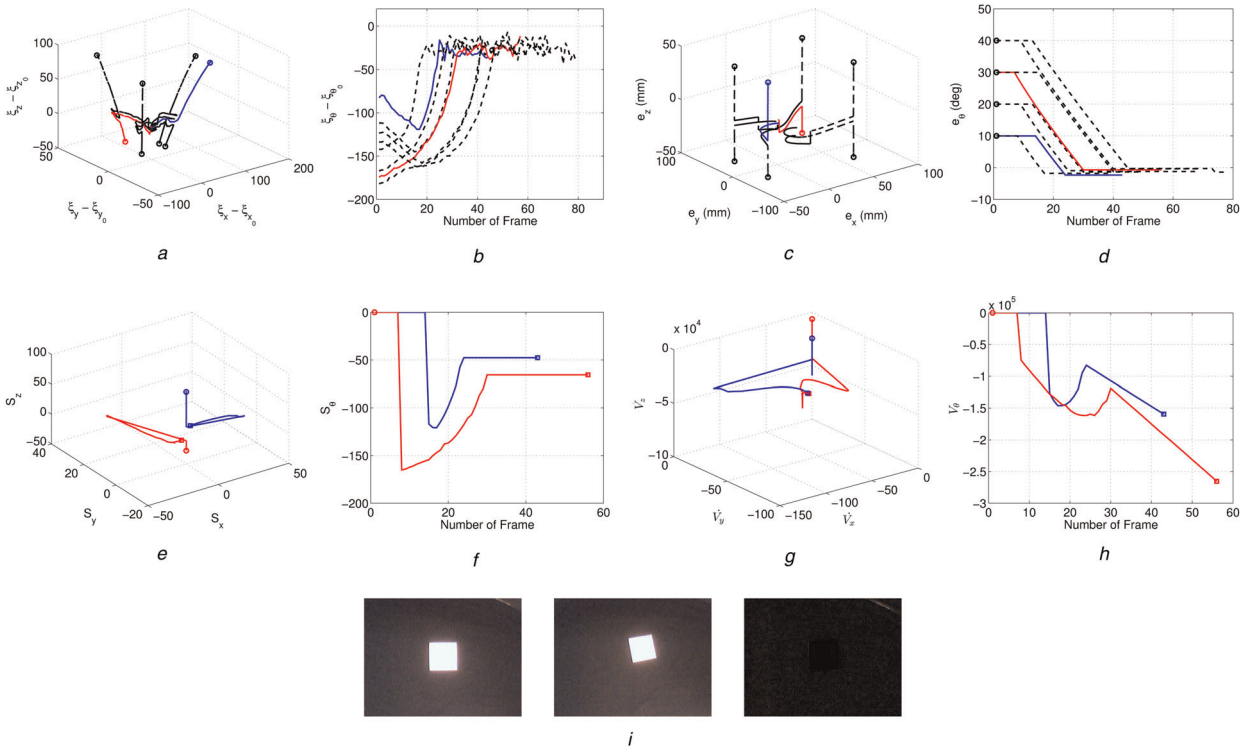


**Fig. 9** *4D experiment*

*a*, *b* Kernel measurement errors
*c*, *d* Position errors
*e*, *f* Sliding surface
*g*, *h* Derivative of Lyapunov function
*i* Three images are shown for the trial represented by the solid line (from left to right) the image at the goal location, the initial location and the difference between the goal and the final image

**Table 2** Initial conditions and position errors in 4D case

| Pose error | Initial condition | | Final error | |
|---|---|---|---|---|
| | Mean | Std. | Mean | Std. |
| $e_x$, mm | 45.0 | 12.3 | 3.1 | 3.1 |
| $e_y$, mm | 41.3 | 10.9 | 8.5 | 9.4 |
| $e_z$, mm | 44.6 | 0.3 | 13.3 | 4.0 |
| $e_\theta$, deg. | 25 | 12.91 | 1.26 | 0.62 |

camera view field, the correct information of the object cannot be received because of the wide angle of the lens (see Section 4.1). Hence, after some end effector manoeuvres in 4D space, the algorithm can again correct the depth or the roll motion.

The variations of $e_{\xi_\theta}$, in Fig. 9*b*, are due to the type of kernel-function. Owing to these variations, we cannot choose a more delicate stop condition. $e_{\xi_\theta}$ does not change its sign, therefore the control signal cannot reverse the rotation of the end effector. In case an appropriate stop condition is not chosen, the robot will not stop near $0^\circ$, and would rotate another spin of $90^\circ$ with the objective of tracking a square object. As the variations of $e_{\xi_\theta}$ are too much, there is a chance of sticking to the stop condition in return. However, if this unpleasant condition happens again, the robot will not stop and might even enter into the singularity position after spinning the whole $360^\circ$. Therefore, we have to moderate the roll stop condition. In adhering to this policy, the roll kernel-measurement error is not that critical, since the experiments verify that the robot is able to track the target with acceptable accuracy. Based on the results of Figs. 9*c* and *d*, the mean value and the standard deviation of tracking error for these experiments are summarised in Table 2.

Moreover, $\dot{V}$ and the sliding surface in the 3D and roll dimension are shown in Figs. 9*g* and *h* and Figs. 9*e* and *f*, respectively. The initial and final conditions are highlighted with circles and squares, respectively. Before any move, zero value can be seen at each direction. After that, based on the integral term, the sliding surface varies, and because of the stop condition the tracking error will converge to set $\boldsymbol{T}_1$. $\dot{V}$ is negative in all directions, and because of the stop condition, the UUB stability can be observed.

To verify the robustness of the algorithm to the shape of objects, different shapes of target object were examined, and the results of one of these objects (a cup) in all possible directions are shown in Fig. 10. We notice a problem in the Gaussian kernel-measurement of $\xi_\theta$ for different objects. $K_\theta$ in (16) is dependent to the object shape. For example, we expect that a rectangular object will have the same kernel-measurement after $180^\circ$ of rotation. However, $\xi_\theta$ will have the same performance after $90^\circ$ as a square-shaped object. This shows that $K_\theta$ in (16) cannot be used for

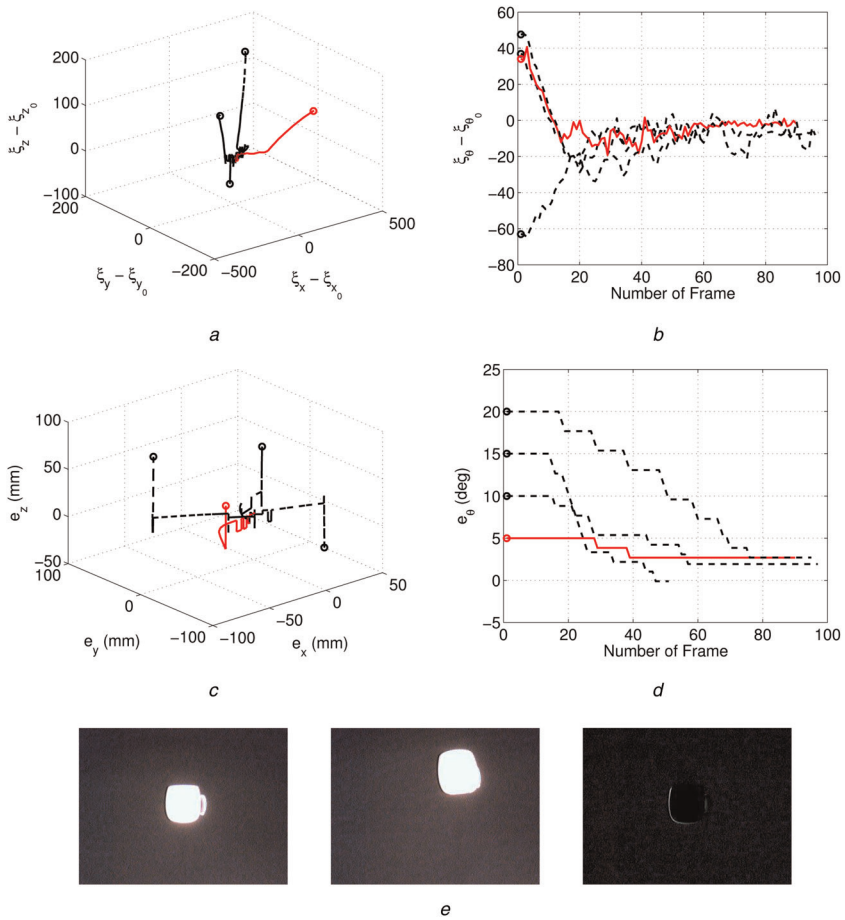


**Fig. 10** *4D motion experiments for a cup*

*a*, *b* Position errors
*c*, *d* Kernel-measurement errors
*e* For the trial represented by the solid line, three images are shown: (from left to right) the image at the goal location, the initial location and the difference between the goal and the final image




doi: 10.1049/iet-cvi.2013.0310

different objects with respect to $\theta$ and tracking of the object is performed only within limited ranges. We plan to develop more elaborate function for such tasks in future.

Fig. 10 shows four trials with different initial conditions highlighted with circles for a cup object. One of the four trials is illustrated with solid lines, and three images of the tracking object are shown in Fig. 10*f*. The 3D plot of tracking error [$x$, $y$, $z$] in Fig. 10*a* is not as smooth as the square object, because the cup is not a planar object and tracking in the depth motion is performed in several tries (see Section 4.1). Owing to the wide lens, the centre picture of Fig. 10*e* does not represent the final cup image to resemble the goal image as shown in Fig. 10*e* (left picture). The pattern of FT magnitude for the cup is not same as the square object. Therefore, the tracking range is limited because of restrictions of $K_\theta$ in (16). Fig. 10*b* shows that $e_\theta$ is larger than a simple square object, and also it can be seen in the right picture of Fig. 10*e*.

### 4.6 Comparison of KBVS and KBSMVS methods

In this section, KBVS and KBSMVS methods are compared. The initial condition of the target object is considered in 4 DOF. For better comparison, the initial and stop conditions are selected to be the same as each other. Two cases for this section are considered for positioning the target object in an initial pose. In the first case (case 1), the target object is located near the goal pose as shown in Fig. 11*e*. In the second case (case 2), the target object is located far from the goal pose as shown in Fig. 11*f*.

The position errors are graphed in Figs. 11*c* and *d*, and the numerical results are summarised in Table 3 for all directions. The initial conditions are highlighted with circles in Fig. 11. In case 1, the final pose errors in the $x$-, $y$- and $\theta$-dimensions for both methods have approximately equal values, while $e_z$ in the KBVS method is 6.1 mm more than $e_z$ in the KBSMVS method. In case 2, the final pose errors in the $y$-direction are equal; $e_x$ of KBSMVS is 1.1 mm more than $e_x$ of KBVS, while $e_z$ and $e_\theta$ of KBVS are, respectively, 5.2 and 6.1 mm more than $e_z$ and $e_\theta$ of KBSMVS. In general, the accuracy of the KBSMVS method is better than the KBVS method. In the KBVS method, it takes more iterations to track the object. This is proportional to the number of frames in Figs. 11*b* and *d*. In both methods, first the $z$-direction is corrected, but at the end, some variations in the $x$-$y$-directions and the $\theta$-orientation of kernel-measurement error are observed.

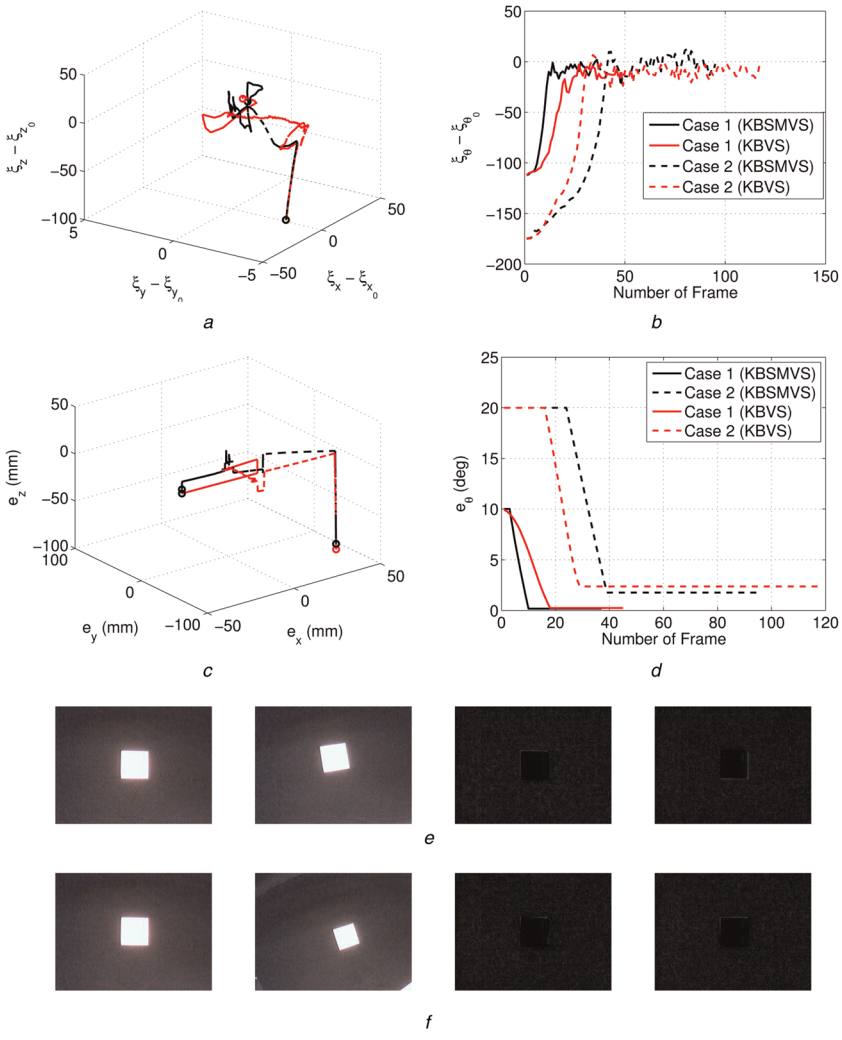


**Fig. 11** *Comparison between KBVS and KBSMVS experiments*

*a*, *b* Kernel measurement errors
*c*, *d* Position errors
*e*, *f* Four images are shown: (from left to right) the image at the goal location, the initial location, the difference between the goal and the final image of KBVS method and the difference between the goal and the final image of KBSMVS method for (e) case 1 and (f) case 2

**Table 3** Comparison between our proposed method and KBVS method

| Method | $e_{pose}$ | $e_x$, mm | $e_y$, mm | $e_z$, mm | $e_\theta$, deg. |
|---|---|---|---|---|---|
| KBSMVS | case 1 | 0.4 | 4.0 | 0.4 | 0.18 |
| | case 2 | 1.3 | 0.9 | 0.8 | 1.76 |
| KBVS | case 1 | 0.9 | 4.0 | 6.5 | 0.23 |
| | case 2 | 0.2 | 0.9 | 6.0 | 2.37 |

As another experiment, we place the target near the edge of the work space to investigate whether KBVS can work in larger work spaces or not. It is verified that both methods have good performance for small workspaces, but KBSMVS performs significantly better for larger regions. This outperformance is attributed to the design of the control signal, which is different in each method. The appearance of the Jacobian kernel-measurement matrix in each control signal is quite different. In the KBVS method, the Jacobian matrix is used directly, while in our method the inverse Jacobian matrix is used. When we placed the target object in the large region of the visual workspace, the amplitude of control signal in the KBVS method was calculated to be larger than KBSMVS. This causes the robot not to track the object successfully in the KBVS method. Therefore, the visibility of the target object might be undesirably diminished. This result in large movement of the robot interferes with the singularity avoidance routine, meaning that the produced joint pose will have notable errors.

## 5 Conclusions

In this paper, we introduced a sliding mode controller design in KBVS. Through binding kernel-measurement to the SMC, our configured system will outperform the conventional KBVS system. Comparison between KBVS and the proposed method for different initial conditions shows that for small movements, both methods track the object accurately, while in large motions, only the proposed method is able to track with our acceptable accuracy. Via SMC, it is possible to tune all DOFs of the robot simultaneously. However, the type of kernel-functions has limited tuning capabilities. For further work, it is proposed to eliminate such limitations to increase the performance of controllers' type. With respect to the analysis of 4D motion experiments for a cup, the performance of the present roll kernel function is very sensitive to the shape of the object. This shortcoming diminishes the rotational tracking performance within limited degrees. Thus, it is necessary to develop a function for such functionality.

## 6 References


1 Kallem, V., Dewan, M., Swensen, J.P., Hager, G.D., Cowan, N.J.: 'Kernel-based visual servoing'. IEEE/RSI Int. Conf. on Intelligent Robots and Systems, San Diego, CA, USA, 2007, pp. 1975–1980

2 Kallem, V.: 'Vision-based control on lie groups with application to needle steering'. PhD thesis, Johns Hopkins University, 2008

3 Kallem, V., Dewan, M., Swensen, J.P., Hager, G.D., Cowan, N.J.: 'Kernel-Based Visual Servoing: Featureless Control using Spatial Sampling Functions', 2010, Available at: http://www.libra.msra.cn/Publication/7037408/kernel-based-visual-servoing-featureless-control-using-spatial-sampling-functions

4 Chaumette, F., Hutchinson, S.: 'Visual servo control part i: basic approaches', *IEEE Robot. Autom. Mag.*, 2006, **13**, (4), pp. 82–90

5 Wilson, W.J., Williams, H.C.C., Bell, G.S.: 'Relative end effector control using Cartesian position based visual servoing', *IEEE Trans. Robot. Autom.*, 1996, **12**, (5), pp. 648–689

6 Janabi-Sharifi, F., Marey, M.: 'A Kalman-filter-based method for pose estimation in visual servoing', *IEEE Trans. Robot.*, 2010, **26**, (5), pp. 939–947

7 Taghirad, H.D., Atashzar, S.F., Shahbazi, M.: 'Robust solution to three-dimensional pose estimation using composite extended Kalman observer and Kalman filter', *IET Comput. Vis.*, 2012, **6**, (2), pp. 140–152

8 Hutchinson, S., Hager, G.D., Croke, P.I.: 'A tutorial on visual servo control', *IEEE Trans. Robot. Autom.*, 1996, **12**, (5), pp. 651–670

9 Malis, E., Mezouar, Y., Rives, P.: 'Robustness of image based visual servoing with a calibrated camera in the presence of uncertainties in the three-dimensional structure', *IEEE Trans. Robot.*, 2010, **26**, (1), pp. 112–120

10 Croke, P.I., Hutchinson, S.A.: 'A new partitioned approach to image-based visual servo control', *IEEE Trans. Robot. Autom.*, 2001, **17**, (4), pp. 507–515

11 Tahri, O., Araujo, H., Chaumette, F., Mezouar, Y.: 'Robust image-based visual servoing using invariant visual information', *Robot. Autonom. Syst.*, 2013, **61**, (12), pp. 1588–1600

12 Malis, E., Chaumette, F., Boudet, S.: '2–1/2-D visual servoing', *IEEE Trans. Robot. Autom.*, 1999, **15**, pp. 238–250

13 Chaumette, F., Malis, E.: '2–1/2D visual servoing: a possible solution to improve image-based and position-based visual servoings'. IEEE Int. Conf. on Robotics and Automation, 2000, vol. 1, pp. 630–635

14 Chaumette, F., Hutchinson, S.: 'Visual servo control for part II: advance approaches', *IEEE Trans. Robot.*, 2007, **14**, (1), pp. 109–118

15 Comaniciu, D., Ramesh, V., Meer, P.: 'Kernel-based object tracking', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2003, **25**, (5), pp. 564–575

16 Han, R., Jing, Z., Li, Y.: 'Kernel based visual tracking with variant spatial resolution model', *IET Electron. Lett.*, 2008, **44**, (8), pp. 517–518

17 Hager, G.D., Dewan, M., Stewart, C.V.: 'Multiple kernel tracking with SSD'. Proc. Computer Vision and Pattern Recognition, 2004, vol. 1, pp. 790–797

18 Dewan, M., Hager, G.D.: 'Toward optimal kernel-based tracking'. Proc. Computer Vision and Pattern Recognition, June 2006, vol. 1, pp. 618–625

19 Tahri, O., Chaumetter, F.: 'Point-based and region-based image moments for visual servoing of planar objects', *IEEE Trans. Robot.*, 2005, **21**, (6), pp. 1116–1127

20 Copot, C., Lazar, C., Burlacu, A.: 'Predictive control of nonlinear visual servoing systems using image moments', *IET Control Theory Appl.*, 2012, **6**, (10), pp. 1486–1496

21 Khalil, H.K.: 'Nonlinear systems' (Prentice-Hall, Upper Saddle River, NJ, 2005, 3rd edn.)

22 Li, F., Xie, H.L.: 'Sliding mode variable structure control for visual servoing system', *Int. J. Autom. Comput.*, 2010, **7**, (3), pp. 317–323

23 Stepanenko, Y., Cao, Y., Su, C.Y.: 'Variable structure control of robotic manipulator with PID sliding surfaces', *Int. J. Robust Nonlinear Control*, 1998, **8**, (1), pp. 79–90

24 Kim, J.K., Kim, D.W., Choi, S.J., Won, S.C.: 'Image-based visual servoing using sliding mode control'. SICE-ICASE Int. Joint Conf., Busan, October 2006, pp. 4996–5001

25 Zanne, P., Morel, G., Plestan, F.: 'Robust vision based 3D trajectory tracking using sliding mode control'. Proc. of IEEE, Int. Conf. on Robotics and Automation, April 2000, vol. 3, pp. 2088–2093

26 Zanne, P., Morel, G., Plestan, F.: 'Robust 3D vision based control and planning'. Proc. of IEEE, Int. Conf. on Robotics and Automation, April 2004, vol. 5, pp. 4423–4428

27 Wang, C., Shen, Y., Liu, Y., Wang, Y.: 'Robust visual servoing tracking of robot manipulators with uncertain dynamics and uncalibrated camera'. Seventh Int. Conf. on Control, Automations, Robotics and Vision (ICARCV02), December 2002, vol. 3, pp. 1144–1149

28 Taghirad, H.D., Shahbazi, M., Atashzar, S.F., RayatDoost, S.: 'A Robust Pose-Based Visual Servoing Techniques for Redundant Manipulators', submitted to Robotica, December 2012

29 Slotine, J.E., Li, W.: 'Applied nonlinear control' (Prentice Hall International, Englewood Cliffs, NJ, 1991)

30 Bradski, G., Kaebler, A.: 'Learning open CV, computer vision with the open CV library' (OREILLY, 2008, 1st edn.)

31 Gonzalez, R.C., Woods, R.E., Eddins, S.L.: 'Digital image processing using MATLAB' (Prentice-Hall, Upper Saddle River, NJ, 2004)





doi: 10.1049/iet-cvi.2013.0310