Visual servoing ECM

Nunziante, Luca Gravina, Giovanbattista

Atanasious, Mohab Mahdy Helmy Becchetti, Valentina

August 2022

Abstract

The aim of this project is to perform Image Based Visual Servoing (IBVS) using the Endoscopic Camera Manipulator (ECM) of the da Vinci surgical robot, equipped with two vision sensors. The task is to keep the two end effectors of the Patient Side Manipulators (PSMs) in the center of the field of view of the ECM cameras. Experiments are performed in the CoppeliaSim environment.

1 Introduction

In the following of this work the scene on which Visual Servoing is performed is first presented, followed by a walk through of the classical steps of an IBVS scheme, namely: feature extraction, finding the relation between the motion of the camera and of the feature point on the image plane. Then, a control scheme is presented and experimental data are reported.

2 Simulation setup

The CoppelliaSim scene, which can be seen in Figure 1, consists in:

- The da Vinci surgical robot model, taken from `https://github.com/unina-icaros/dvrk-vrep`
- A red marker on each end effector of the PSMs
- A Table to set a boundary to the PSMs motion. Force feedback is generated at contact

Interaction with the scene happens through two Geomagic Touch devices, used to command the motion of the PSMs, and an Oculus Rift to render what is seen by the two vision sensors of the ECM.

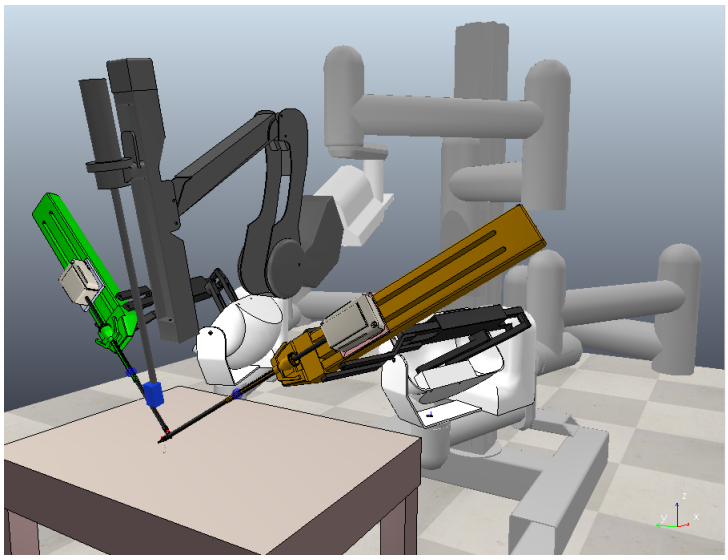


Figure 1: CoppeliaSim Scene used for the experiments.

3 Feature extraction

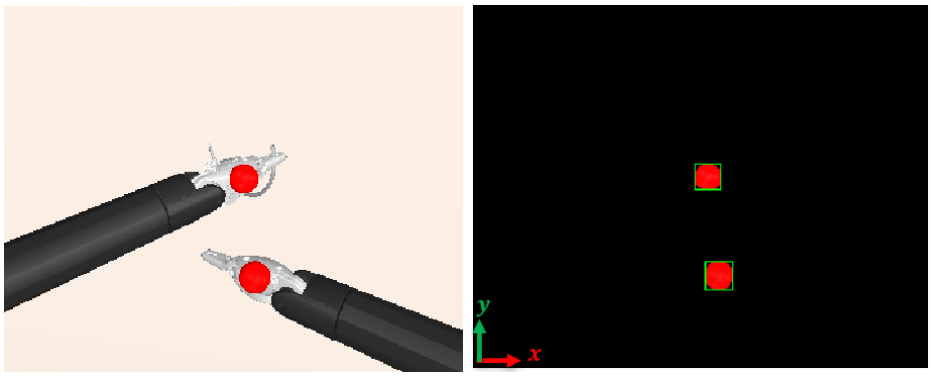
Extraction of meaningful features is done through an image processing phase, considering separately the images acquired by the two vision sensors.
Firstly, red pixels only are retrieved and blob detection is performed, obtaining the normalized blob center coordinates in $[0, 1]$. In this phase we set a minimum size for the blobs to be detected, to avoid the detection of spurious ones. Moreover, since it may happen that more than one blob is detected around the same marker, from the set of detected blobs the two furthest ones are kept and used in the following. The initial and final image are shown in Figure 2.
Since image processing is done on both vision sensors, we end up with two images like the one in Figure 2b, thus we can compute the mean point of the blobs in each image plane, respectively referred as m_L, m_R.
Then, the mean f of these two means is computed and treated as the feature point for the IBVS scheme.

4 Differential kinematics

The next step is to find the relation between the velocity of the camera and the velocity of the point feature in the image plane.

4.1 Camera parameters

First, some camera parameters are required, namely:



(a) View of the vision sensor before image pro-(b) Image after blob detection.
cessing.

Figure 2: Image processing transformation. The frame in which blob coordinates are expressed is placed in the bottom-left corner in (b).

- The perspective angle θ as can be seen in Figure 3.
- The relative position between the two vision sensors.

From θ it's possible to compute $\lambda = \frac{0.5}{tan(\theta/2)}$ that is the focal length normalized by the width of the image.

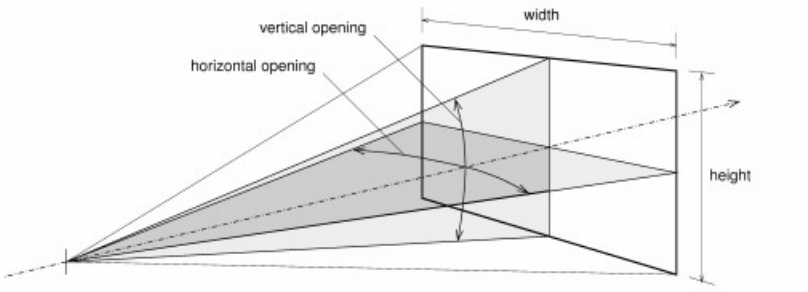


Figure 3: 3D property of a perspective view. The horizontal opening corresponds to the perspective angle used in this work.

4.2 Computing the interaction matrix

To compute the interaction matrix, a pinhole camera model is considered. Hence, the camera frame is displaced of the focal length along its z axis and has the x axis in the opposite direction w.r.t. the image plane frame. The camera frame of one vision sensor is reported in Figure 4.
Thus, we can write the relation between points in the image plane and in the

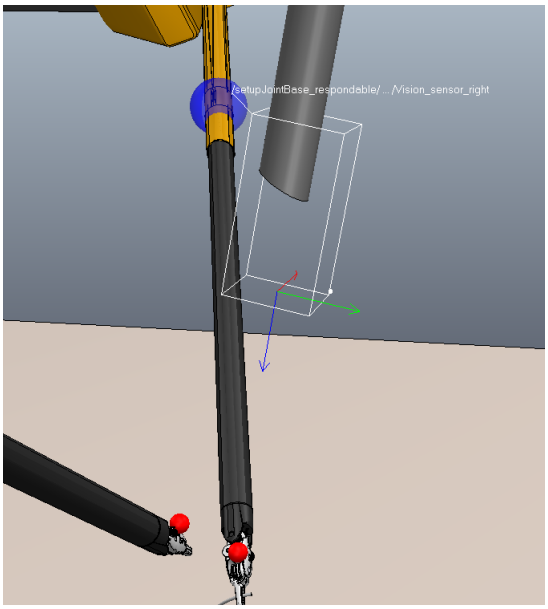


Figure 4: Camera frame for the right vision sensor of the ECM, with the last revolute joint – that provides the roll motion – set to zero. The RGB axes correspond to X,Y,Z.

camera frame as

$$\begin{aligned} u &= -\lambda \cdot \frac{X}{Z} + u_c \\ v &= \lambda \cdot \frac{Y}{Z} + v_c \end{aligned} \tag{1}$$

where u, v are the normalized feature point coordinates in the image plane, u_c, v_c are the coordinates of the optical center and X, Y, Z are the feature point coordinates in the camera frame. In this work, the optical center is the center of the image plane, i.e. $(u_c, v_c) = (0.5, 0.5)$.
The depth Z is computed with the triangulation method as follows

$$Z = \frac{B \cdot \lambda}{{}^{I_L}u - {}^{I_R}u} \tag{2}$$

where the superscripts I_L, I_R identify respectively the left and right image plane, thus ${}^{I_L}u, {}^{I_R}u$ are the abscissa of m_L, m_R respectively, and B is the stereo base as shown in Figure 5.

Computing the derivative of Eq. 1 w.r.t. time we obtain

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{Z} & 0 & -\frac{u-u_c}{Z} \\ 0 & \frac{\lambda}{Z} & \frac{v-v_c}{Z} \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \boldsymbol{J_1} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \tag{3}$$

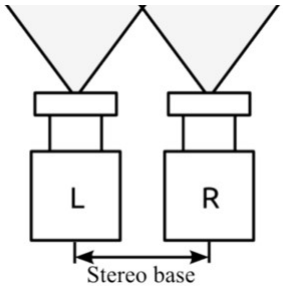


Figure 5: Stereo camera setup.

Moreover, we have the following kinematic relation between $\dot{X}, \dot{Y}, \dot{Z}$ and the camera linear and angular velocities $V, \Omega \in \mathbb{R}^3$ expressed in the camera frame

$$\begin{aligned} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} &= - V - \Omega \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ &= \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix} \\ &= \boldsymbol{J_2} \begin{bmatrix} V \\ \Omega \end{bmatrix} \end{aligned} \tag{4}$$

and substituting Eq. 4 in 3, and also substituting to X, Y in $\boldsymbol{J_2}$ their expression from 1 we can finally write

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \boldsymbol{J_1 J_2} \begin{bmatrix} V \\ \Omega \end{bmatrix} = \boldsymbol{J_p} \begin{bmatrix} V \\ \Omega \end{bmatrix} \tag{5}$$

where we find the interaction matrix $\boldsymbol{J_p} \in \mathbb{R}^{2\times 6}$ as

$$\boldsymbol{J_p} = \begin{bmatrix} \frac{\lambda}{Z} & 0 & \frac{(u-u_c)}{Z} & \frac{(u-u_c)(v-v_c)}{\lambda} & \lambda + \frac{(u-u_c)^2}{\lambda} & -(v-v_c) \\ 0 & -\frac{\lambda}{Z} & -\frac{(v-v_c)}{Z} & \lambda - \frac{(v-v_c)^2}{\lambda} & -\frac{(u-u_c)(v-v_c)}{\lambda} & u-u_c \end{bmatrix} \tag{6}$$

5 Control

The ECM is a 4-DoF actuated arm, which moves the endoscopic camera about the RCM through revolute and prismatic joints, combined in a RRPR sequence. In this kinematic structure the Remote Center of Motion (RCM) is obtained by mechanical design as can be seen in Figure 6. In this work, the Visual Servoing control is implemented commanding the three revolute joints and the prismatic one in a decoupled way.

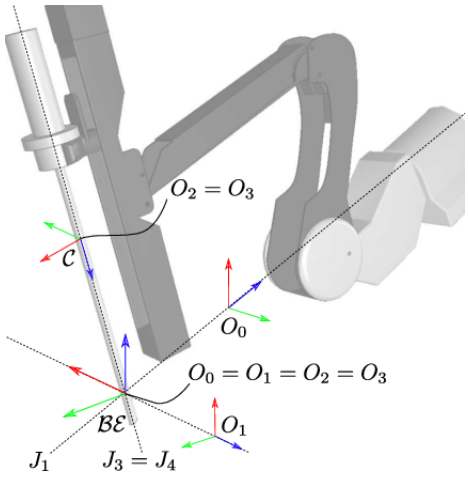


Figure 6: ECM kinematic description. The RCM is at the intersection of the joint axes. [1]

5.1 Revolute joints control

We consider the angular part of the Geometric Jacobian, obtaining the following matrix

$$\boldsymbol{J_\omega} = \begin{bmatrix} 0 & cos(q_1) & 0 & -cos(q_2)sin(q_1) \\ 0 & sin(q_1) & 0 & cos(q_2)cos(q_1) \\ 1 & 0 & 0 & sin(q_2) \end{bmatrix} \tag{7}$$

and for consistency with the interaction matrix $\boldsymbol{J_p}$, this Jacobian needs to be expressed in the camera frame. Since said frame has the same orientation as the end effector frame, the rotation is obtained pre multiplying $\boldsymbol{J_\omega}$ by the transpose of the 3×3 rotation matrix that goes from the base frame to the end effector frame, obtaining ${}^{\boldsymbol{C}}\boldsymbol{J_\omega} = ({}^{\boldsymbol{B}}\boldsymbol{R_C})^T\boldsymbol{J_\omega}$.
Moreover, by considering $\boldsymbol{J_\omega}$ and not the whole 6×4 Jacobian, only the angular velocity of the camera Ω is tracked. Thus, only the last three columns of $\boldsymbol{J_p}$ need to be considered to obtain the Image Jacobian $\boldsymbol{J} = \boldsymbol{J_{p,r}}{}^{\boldsymbol{C}}\boldsymbol{J_\omega}$ where $\boldsymbol{J_{p,r}}$ is the reduced version of the interaction matrix, hence $\boldsymbol{J_{p,r}} \in \mathbb{R}^{2\times3}$ and $\boldsymbol{J} \in \mathbb{R}^{2\times4}$. Then, we define the error in the – bidimensional – task space as $e = f_d - f$ where f_d is the desired feature point coordinates, in our case $f_d = (0.5, 0.5)$, and f is the current position of the feature point. Finally, we can compute the desired joint velocity as

$$\dot{q} = \boldsymbol{J}^\dagger k_r e \tag{8}$$

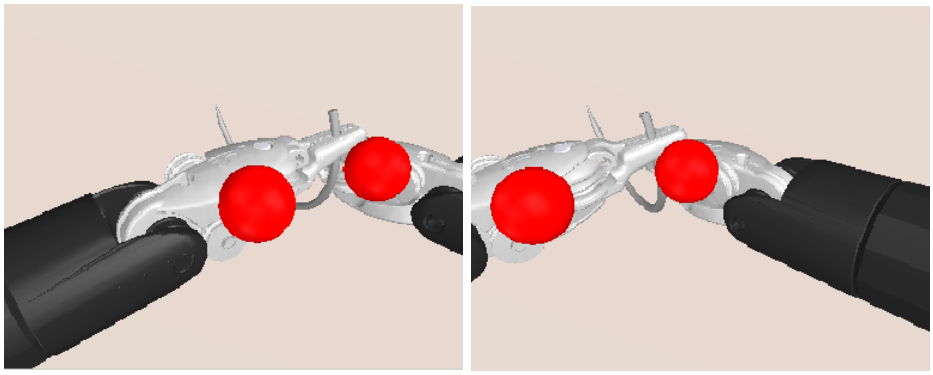
where $\boldsymbol{J}^\dagger$ is the pseudo inverse of $\boldsymbol{J}$, and k_r is a scalar proportional gain. Note that the third column of $\boldsymbol{J_\omega}$ is made of all zeros, so $\dot{q}_3$ from Eq. 8 is identically zero.

5.2 Prismatic joint control

The prismatic joint is controlled to keep both end effectors in the FOV of the vision sensors. The criteria to do so are:

- zoom in based on the distance between the detected blobs and their mean
- zoom out based on the distance between blobs and the center of the image

and both criteria are applied separately to the right and left image; then the actual prismatic joint velocity is chosen.
The difference between the two approaches is due to the mismatch between the position of the mean of the blobs in an image plane and the center of the image[1], as illustrated in Figure 7. Hence, the distance of the blobs from their mean is an indicator solely of how close to each other they are, and not of their proximity to the borders of the image.
This phenomenon is particularly clear when the PSMs end effectors are close to the ECM vision sensors, and it grows with the stereo base B.



(a) Left vision sensor view.

(b) Right vision sensor view.

Figure 7: Vision sensors view during Visual Servoing. These pictures are taken with a stereo base B larger than the one used in the following to better show the problem. Indeed, B needed to be lowered in order to make the images clear when rendered on the Oculus Rift, hence this phenomenon also weakened.

5.2.1 Zoom in control

To implement the first criterion, given the detected blobs $b_{1,L}, b_{2,L}, b_{1,R}, b_{2,R}$ and their mean m_L, m_R on the respective image plane, it's possible to compute

$$m_{d,*} = \frac{1}{k} \sum_{i=1}^{k} ||m_* - b_{i,*}|| \tag{9}$$

[1]This happens since our Visual Servoing implementation aims to keep the mean of the two means centered.

where $* \in \{L, R\}$ specify the image plane considered, and $m_{d,*}$ is the mean distance of the targets in the image from their actual mean position.
A lower threshold min_t is set and the aim is to keep the quantity in Eq. 9 above it. If that is not the case, it means that the blobs are close to each other and the prismatic joint velocity is computed as

$$\dot{q}_{3,*} \;=\; k_p \cdot (min_t - m_{d,*}), \tag{10}$$

where k_p is another proportional gain.

5.2.2 Zoom out control

When the mean distance is above the minimum threshold, the maximum distance of the blobs from the center of the image is computed, i.e.

$$d_{max,*} = max(||b_{1,*} - c||, ||b_{2,*} - c||), \tag{11}$$

where $c = [0.5\ 0.5]^T$ and again $* \in \{L, R\}$. Given a maximum threshold d_t, if $d_{max,*} > d_t$ it means that the ECM should retract, so the prismatic joint velocity is given by

$$\dot{q}_{3,*} \;=\; k_p \cdot (d_t - d_{max,*}), \tag{12}$$

Whereas, if $d_{max,*} \leq d_t$ the corresponding velocity is set to 0.

5.2.3 Velocity issuing

Whenever $m_{d,*}$ or $d_{max,*}$ violate the respective bounds, the quantities in Eq. 10 and 12 are computed and the commanded velocity is

$$\dot{q}_3 = \underset{\dot{q}_{3,L}, \dot{q}_{3,R}}{\text{argmax}} \ (|\dot{q}_{3,L}|, |\dot{q}_{3,R}|) \tag{13}$$

However, as said in Section 3, it may happen that more than one blob is detected around the same marker: if the ECM vision sensors only have one marker in their FOV, those two blobs are also the two furthest ones. In that case, $m_{d,*}$ in Eq. 10 would be very little, hence $\dot{q}_3$ would be high: to avoid this, a safety threshold is set and when any of $m_{d,L}, m_{d,R}$ is below that, the velocity of all the joints of the ECM is set to zero except for the prismatic one that is commanded to retract.
Finally, a supplementary check is performed to always keep the minimum distance between the PSMs end effectors and the cameras above a certain depth threshold. Indeed, it may happen that the minimum distance violates the threshold either because from Eq. 13 it is $\dot{q}_3 > 0$, or because the PSMs end effectors are moving towards to the cameras. In any case, $\dot{q}_3$ is overwritten to satisfy the bound.

5.3 Loss of stereo information

Besides the nominal case where the two sensors both see the two markers, it could happen that they both see one marker or that only one of them sees two and the other just one.[2]
In the former case, Visual Servoing cannot be done: the velocities of the revolute joints of the ECM are set to 0, while the prismatic is commanded to retract to possibly recover nominal behaviour.
If one of the sensors sees two markers, it is still possible to do Visual Servoing. Indeed, the depth information is lost but it is possible to use the last available estimated depth and command the three revolute joints as in the nominal case, while the prismatic is again commanded to retract. This control favours the return of the lost marker in the FOV.
Moreover, to ensure that there are two markers in the image – and not just two blobs around the same marker – also in this case the check on the safety threshold is done.

6 Experiments

Experiments were performed using two Geomagic Touch Devices and one Oculus Rift connected to a laptop with a dedicated NVIDIA GeForce 1650 GPU. The setup can be seen operating in Figure 8. With this hardware, the simulation runs in real time.

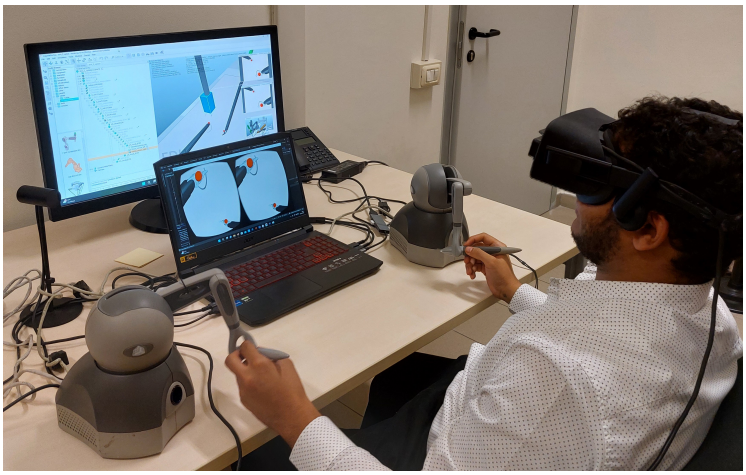
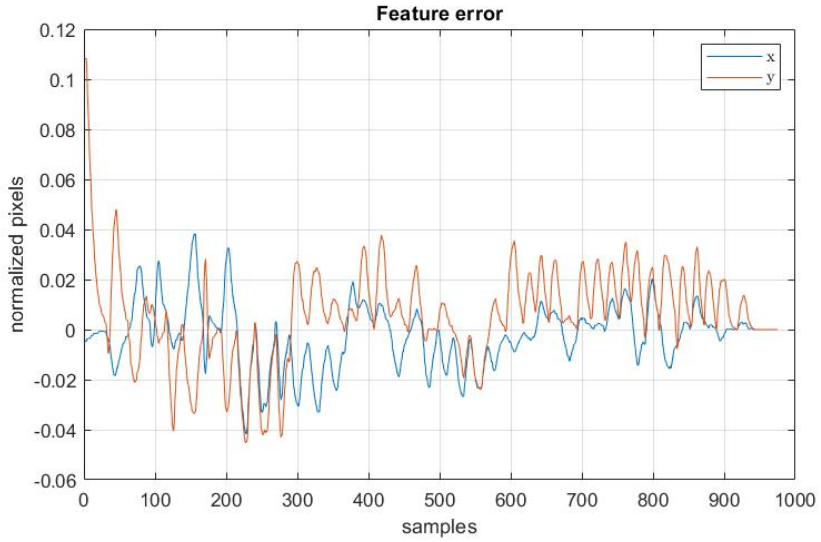


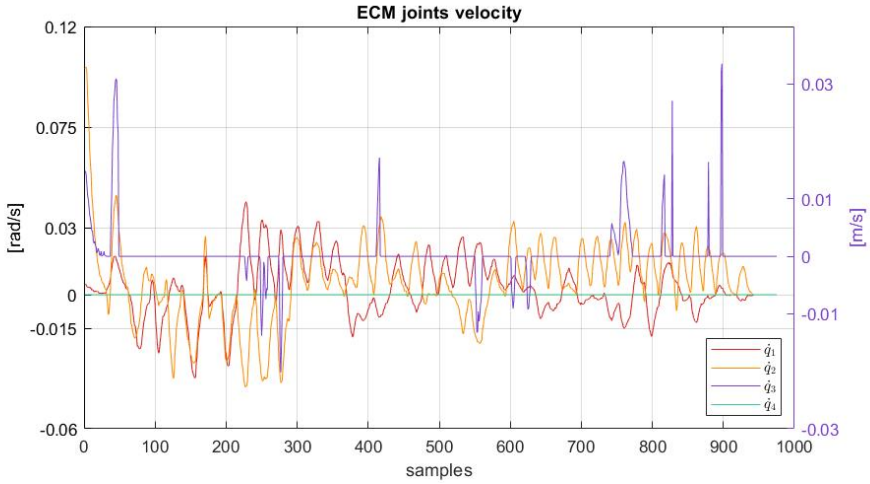
Figure 8: Operating experimental setup. The two images on the laptop screen correspond to the ones the operator sees through the Oculus. On the screen they are displayed separately while the operator will naturally see them as one.

[2] As the phenomenon in Figure 7, this also happened with a higher B.

In Figure 9a and 9b are respectively shown the bidimensional point feature error and the ECM joints velocities issued during a simulation.
In the first plot the error seems to never go to zero and this is because every time the PSMs are moved, a new transient is triggered: indeed, at the end of the simulation the error reaches the steady state value of 0, along with the ECM joint velocities, since the PSMs are still.
In the joints velocity plot it is possible to observe the different profile of $\dot{q}_3$ w.r.t. $\dot{q}_1, \dot{q}_2$: indeed, this spike behaviour is due to the control law based on thresholds, as explained in Section 5.2.
Moreover, $\dot{q}_4$ turns out to be identically zero in all the simulations: this suggests that it could be used to accomplish an additional task, e.g. keep the two blobs always horizontally aligned.



(a) Feature error in normalized pixels unit, along the x (blue) and y (orange) coordinates.



(b) Joint velocities of the ECM. The last revolute joint velocity $\dot{q}_4$ (green) is never actuated and is always identically zero.

Figure 9: Error and velocity plots during a simulation.

References

[1] Giuseppe Fontanelli, Mario Selvaggio, Marco Ferro, Fanny Ficuciello, M. Vendiuelli, and B. Siciliano. A v-rep simulator for the da vinci research kit robotic platform. pages 1056–1061, 08 2018.