Visual Servoing
TP: Image-based Control using Points

Erol Ozgur [*]

January 19, 2014

1 Problem

We want to move a camera from its current location to a desired location relative to a pattern (see Fig. 1) by using only image point features. We assume that we have only visual feedback. We can take images and detect image points of the given pattern at 50 Hz. Please do the following exercises:

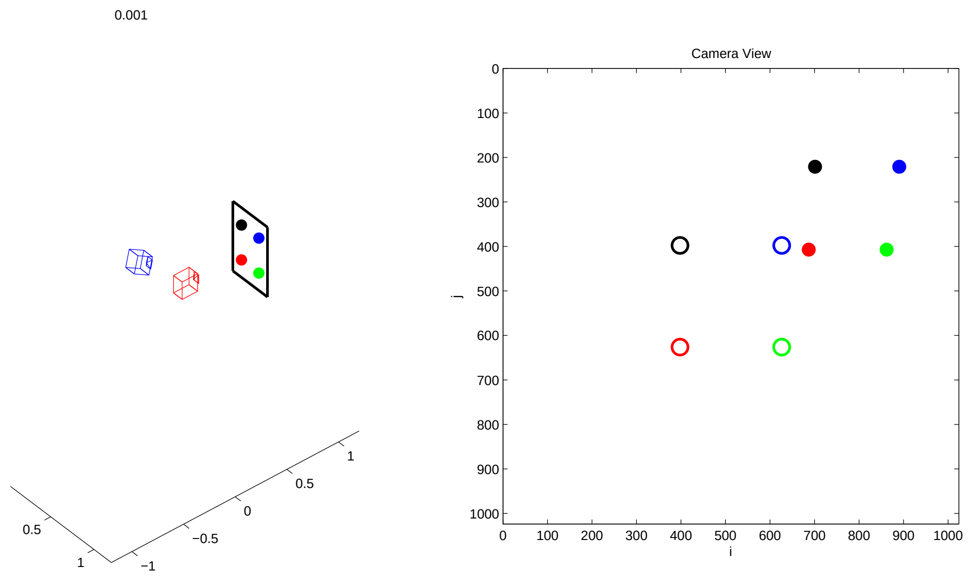

Figure 1: Image based control of the camera pose using point features. Blue camera is the current location, and the red camera is the desired location of the camera with respect a pattern with four points (left figure). Empty dots are the desired image points observed from the desired red camera location, and the full dots are the current image points observed from the current blue camera location (right figure).

1. Discuss how to realize this image based control.

[*]Author is with Pascal Institute, IFMA, Clermont-Ferrand, France, erol.ozgur@ifma.fr

2. Reformulate the problem mathematically.
3. Draw a block diagram of this image based control.
4. Before simulation in matlab, define the steps of the algorithm for this image based control.
5. Write the matlab code to simulate this image based control.
6. Plot the errors of the image based control versus time.

2 Problem Reformulation

Let the current camera be located at A with respect to the reference frame at R, and the image point features vector be noted as $\mathbf{s}_{/A}$ which is computed from location A by observing the pattern. Let the image point features vector, detected from the image taken at desired location B of the camera, be noted as $\mathbf{s}_{/B}$. Then find a control law $\xi = (\nu, \omega)$ expressed with respect to the reference frame at R that will move the Cartesian pose $\mathbf{X}$ of the camera from location A to location B while time t goes from initial time t_i to final time t_f.

3 Solution

An image point features vector $\mathbf{s} \in \Re^{2n\times 1}$ can be written with the point coordinates as follows:

$$\mathbf{s} = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \tag{1}$$

where n is the number of image points, and where x and y are the metric coordinates of the image points. We can then compute the error $\mathbf{e}$ from the measured image coordinates as follows:

$$\mathbf{e} = \mathbf{s}_{/A} - \mathbf{s}_{/B} \tag{2}$$

where $\mathbf{s}_{/A}$ and $\mathbf{s}_{/B}$ are the image feature vectors of the current and desired camera locations, respectively. We assume that the pattern is not moving, thus $\mathbf{s}_{/B}$ is constant. Taking the time derivative of the error, allows us to appear the control law ξ as below:

$$\dot{\mathbf{e}} = \dot{\mathbf{s}}_A = \mathbf{L}\, \xi_{A/A} \tag{3}$$

where $\mathbf{L} \in \Re^{2n\times 6}$ is the interaction matrix (or so-called image Jacobian) that relates the velocities of the image points (in metric units) to the velocity of the camera frame with respect to itself (i.e., camera frame). The interaction matrix $\mathbf{L}_i \in \Re^{2\times 6}$ for a image point $\mathbf{s}_i = [x_i, y_i]^T$ can be derived from the following two equations:

$$x_i = X_i/Z_i\,, \qquad y_i = Y_i/Z_i \tag{4}$$

$$\dot{\mathbf{P}}_i = -\nu - \omega \times \mathbf{P}_i \tag{5}$$

where $\mathbf{P}_i = [X_i, Y_i, Z_i]^T$ is a 3D scene point expressed in the camera frame. Please calculate the interaction matrix $\mathbf{L}$ from the above equations. We then force the error decrease exponentially towards zero by imposing $\dot{\mathbf{e}} = -\lambda\,\mathbf{e}$. This yields the control law as follows:

$$\xi_{A/A} = -\lambda\,\mathbf{L}^{-\dagger}\,\mathbf{e}\,, \qquad \lambda > 0 \tag{6}$$

We should then express the control law $\xi_{A/A}$ with respect to the fixed reference frame at R rather than itself at A. This can be done as follows:

$$\xi_{A/R} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times\,\mathbf{R} \\ \mathbf{0}_{3\times 3} & \mathbf{R} \end{bmatrix}_{A/R} \xi_{A/A} \tag{7}$$

where rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ come from the Cartesian pose $\mathbf{X}_{A/R}$ of the camera at current location A expressed with respect to the fixed reference frame at R. Notation $[\mathbf{t}]_\times$ implies the skew symmetric matrix of the given translation vector $\mathbf{t}$.

Afterwards, we can update the current Cartesian pose of the camera as follows:

$$\mathbf{X}_{A(t+\Delta t)/R} = \begin{bmatrix} \Delta t\,[\omega]_\times & \Delta t\,\nu \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}_{A(\Delta t)/R} \mathbf{X}_{A(t)/R} \tag{8}$$

4 Matlab Simulation

Open the file "ImageBasedServoPointsTP.m", which is located in the folder "tp_matlab_student", and then complete it with your servoing algorithm to simulate the image based control based on points.

Appendix - Matlab Functions

x = uthetat2dq(u, θ, t) computes the dual quaternion **x** from given rotation unit axis **u**, rotation angle θ, and translation vector **t**.

(u, θ, R, t) = dualq2uthetaRt(x) computes the rotation unit axis **u**, rotation angle θ, rotation matrix **R**, and translation vector **t** from given dual quaternion **x**.

x = muldualpq(p, q) multiplies two given dual quaternions, e.g., **p** and **q**, and gives the result dual quaternion, e.g., **x**.

x^{-1} **= conjdualqsimple(x)** computes the inverse of a given dual quaternion.

plot_pose(X, color) plots the 3D frame of a given Cartesian pose **X** in a defined color, e.g., for red 'r', for blue 'b'.

plot_camera(X, color) plots the camera at a given Cartesian pose **X** with a defined color, e.g., for red 'r', for blue 'b'.

s = skew(t) computes the skew symmetric matrix, e.g., **s**, of a given vector, e.g., **t**.

points3D = put_pattern(X) puts the pattern to a given Cartesian pose **X** and returns 4 3D pattern points, e.g., here noted as **points3D** $= [\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4]$ where $\mathbf{P} = [X, Y, Z]^T$.

plot_pattern(points3D) plots the pattern points in the 3D scene.

There are also the following standard functions of Matlab which can help you to visualize the image based control: **clf**; **hold on**; **box on**; **subplot**; **plot**; **plot3**; **drawnow**; **axis**; **view**; **axis ij**; **axis image**; **xlabel**; **ylabel**.