

ViSP 2.6.1: Visual Servoing Platform

Camera calibration

Lagadic project
<http://www.irisa.fr/lagadic>

September 6, 2011

François Chaumette
Eric Marchand
Fabien Spindler
Romain Tallonneau

Contents

1	Virtual visual servoing based camera calibration	4
1.1	Principle	4
1.2	Visual servoing and calibration	4
1.3	Multi-images calibration	5
1.4	Calibration from points	6
1.4.1	Camera model	6
1.4.2	Deriving the interaction matrix	6
1.5	Camera calibration in ViSP	8

1 Virtual visual servoing based camera calibration

1.1 Principle

The basic idea of our approach is to define the pose computation and the calibration problem as the dual problem of 2D visual servoing [1, 2]. In visual servoing, the goal is to move the camera in order to observe an object at a given position in the image. This is achieved by minimizing the error between a desired state of the image features \mathbf{p}_d and the current state \mathbf{p} . If the vector of visual features is well chosen, there is only one final position of the camera that allows to achieve this minimization. We now explain why the calibration problem is very similar.

Let us define a virtual camera with intrinsic parameters ξ located at a position such that the object frame is related to the camera frame by the homogeneous 4×4 matrix ${}^c\mathbf{M}_o$. ${}^c\mathbf{M}_o$ defines the pose whose parameters are called extrinsic parameters. The position of the object point ${}^c\mathbf{P}$ in the camera frame is defined by:

$${}^c\mathbf{P} = {}^c\mathbf{M}_o \mathbf{P}$$

and its projection in the digitized image by:

$$\mathbf{p} = pr_\xi({}^c\mathbf{P}) = pr_\xi({}^c\mathbf{M}_o \mathbf{P}) \quad (1)$$

where $pr_\xi(\cdot)$ is the projection model according to the intrinsic parameters ξ . The goal of the calibration is to minimize the error between the observed data denoted \mathbf{p}_d (usually the position of a set of features on a calibration grid) and the position of the same features computed by back-projection according to the current extrinsic and intrinsic parameters \mathbf{p} (as defined in Equation 1). In order to ensure this minimization we move the virtual camera (initially in ${}^c\mathbf{M}_o$ and modify the intrinsic camera parameters (initially ξ_i) using a visual servoing control law. When the minimization is achieved, the parameters of the virtual camera will be ${}^c\mathbf{M}_o$, that is the real pose, and ξ_f .

We will show in the next paragraphs how to perform this minimization using visual servoing an the interests of considering this approach.

1.2 Visual servoing and calibration

The goal is to minimize the error $\|\mathbf{p} - \mathbf{p}_d\|$. We therefore define the error in the image \mathbf{e} by the simple relation:

$$\mathbf{e} = \mathbf{p} - \mathbf{p}_d \quad (2)$$

The motion of the features in the image is related to the camera velocity \mathbf{v}_c and the time variation of the intrinsic parameters by:

$$\dot{\mathbf{p}} = \frac{\partial \mathbf{p}}{\partial \mathbf{r}} \frac{d\mathbf{r}}{dt} + \frac{\partial \mathbf{p}}{\partial \xi} \frac{d\xi}{dt} \quad (3)$$

that can be rewritten as:

$$\dot{\mathbf{p}} = \mathbf{H}_p \mathbf{V} \quad \text{with} \quad \mathbf{V} = \begin{bmatrix} \mathbf{v}_c \\ \dot{\xi} \end{bmatrix} \quad (4)$$

Matrix \mathbf{H}_p is classically called interaction matrix or image Jacobian in the visual servoing community. It is given:

$$\mathbf{H}_p = \begin{bmatrix} \frac{\partial \mathbf{p}}{\partial \mathbf{r}} & \frac{\partial \mathbf{p}}{\partial \xi} \end{bmatrix} \quad (5)$$

If we specify an exponential decoupled decrease of the error \mathbf{e} that is:

$$\dot{\mathbf{e}} = -\lambda \mathbf{e} \quad (6)$$

where λ is a proportional coefficient that tunes the decay rate, we finally get:

$$\mathbf{V} = -\lambda \mathbf{H}_{\mathbf{p}}^+ \mathbf{e} \quad (7)$$

where $\mathbf{H}_{\mathbf{p}}^+$ is the pseudo inverse of matrix $\mathbf{H}_{\mathbf{p}}$ ($\mathbf{H}^+ = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$ if \mathbf{H} is a full rank matrix).

Comments about the choice of \mathbf{p} . Any kind of feature can be considered within this control law as soon as we are able to compute the image Jacobian $\mathbf{H}_{\mathbf{p}}$. In [1], a general framework to compute $\frac{\partial \mathbf{p}}{\partial r}$ is proposed. On the other side $\frac{\partial \mathbf{p}}{\partial \xi}$ is seldom difficult to compute as it will be shown in the next section. This is one of the advantages of this approach with respect to other non-linear calibration approaches. Indeed we are able to perform calibration from a large variety of primitives (points, lines, circles, etc...) within the same framework. Furthermore, considering various kind of primitives within the same calibration step is also possible.

1.3 Multi-images calibration

The intrinsic parameters obtained using one image may be, in practice, very different from the parameters obtained with another image taken from another viewpoint, even if the same lens, the same camera, the same frame grabber and the same calibration grid are used. It is therefore important to consider a multi-image calibration process that integrates within a unique minimization process data from various images.

The underlying idea is to compute a unique set of intrinsic parameters that is correct for all the images (i.e., for all the camera positions). Puget and Skordas considered this idea in [3]. They computed the final intrinsic parameters as the mean value of the parameters computed for each images, the camera positions are then recomputed wrt. to these new intrinsic parameters. Our approach is different. We consider a visual servoing scheme that computes the motion of n virtual cameras and the variation of the l intrinsic parameters that have to be the same for all the images. For n images, we therefore have $6n + l$ unknown variables and $\sum_{i=1}^n m_i$ equations (where m_i is the number of features observed in the i^{th} image).

If \mathbf{p}^i is the set of features extracted from i^{th} image, the interaction matrix used in the calibration process is then given by the relation:

$$\begin{bmatrix} \dot{\mathbf{p}}^1 \\ \dot{\mathbf{p}}^2 \\ \vdots \\ \dot{\mathbf{p}}^n \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{v}_{\mathbf{c}}^1 \\ \mathbf{v}_{\mathbf{c}}^2 \\ \vdots \\ \mathbf{v}_{\mathbf{c}}^n \\ \dot{\xi} \end{bmatrix} \quad (8)$$

with

$$\mathbf{H} = \begin{bmatrix} \frac{\partial \mathbf{p}^1}{\partial \mathbf{r}} & 0 & \dots & 0 & \frac{\partial \mathbf{p}^1}{\partial \xi} \\ 0 & \frac{\partial \mathbf{p}^2}{\partial \mathbf{r}} & \dots & 0 & \frac{\partial \mathbf{p}^2}{\partial \xi} \\ \vdots & \vdots & & & \vdots \\ 0 & \dots & 0 & \frac{\partial \mathbf{p}^n}{\partial \mathbf{r}} & \frac{\partial \mathbf{p}^n}{\partial \xi} \end{bmatrix} \quad (9)$$

Minimization is handled using the same methodology:

$$\begin{bmatrix} \mathbf{v}_c^1 \\ \mathbf{v}_c^2 \\ \vdots \\ \mathbf{v}_c^n \\ \dot{\xi} \end{bmatrix} = -\lambda \mathbf{H}^+ \begin{bmatrix} \mathbf{p}^1 - \mathbf{p}_d^1 \\ \mathbf{p}^2 - \mathbf{p}_d^2 \\ \vdots \\ \mathbf{p}^n - \mathbf{p}_d^n \end{bmatrix} \quad (10)$$

1.4 Calibration from points

1.4.1 Camera model

In this section, we use the perspective camera model introduced in the *3D and vision related transformations* tutorial. Let us define by $\mathbf{M} = (X, Y, Z)^T$ the coordinates of a point in the camera frame. The coordinates of the perspective projection of this point in the image plane is given by $\mathbf{m} = (x, y)^T$ with:

$$\begin{cases} x = X / Z \\ y = Y / Z \end{cases} \quad (11)$$

If we denote (u, v) the position of the corresponding pixel in the digitized image, this position is related to the coordinates (x, y) by:

$$\begin{cases} u = u_0 + p_x x + \delta_u \\ v = v_0 + p_y y + \delta_v \end{cases} \quad (12)$$

where δ_u and δ_v are geometrical distortions introduced in the camera model. These distortions are due to imperfections in the lenses design and assembly. δ_u and δ_v can be modeled as follow :

$$\begin{cases} \delta_u(x, y) = p_x x k_{ud} (x^2 + y^2) \\ \delta_v(x, y) = p_y y k_{ud} (x^2 + y^2) \end{cases} \quad (13)$$

or :

$$\begin{cases} \delta_u(u, v) = -(u - u_0) k_{du} \left(\left(\frac{u - u_0}{p_x} \right)^2 + \left(\frac{v - v_0}{p_y} \right)^2 \right) \\ \delta_v(u, v) = -(v - v_0) k_{dv} \left(\left(\frac{u - u_0}{p_x} \right)^2 + \left(\frac{v - v_0}{p_y} \right)^2 \right) \end{cases} \quad (14)$$

The six parameters to be estimated are thus $\{p_x, p_y, u_0, v_0, k_{ud}, k_{dv}\}$. But in a first time, we will only consider the five parameters $\xi_{ud} = \{p_x, p_y, u_0, v_0, k_{ud}\}$.

1.4.2 Deriving the interaction matrix

We have to compute the interaction matrix \mathbf{H}_p that links the motion $\dot{\mathbf{p}} = (\dot{u}, \dot{v})$ of a point $\mathbf{p} = (u, v)$ in the image to $(\mathbf{v}_c, \dot{\xi}_{ud})$. For one point, this Jacobian is given by:

$$\mathbf{H}_p = \begin{bmatrix} \frac{\partial \mathbf{p}}{\partial \mathbf{r}} & \frac{\partial \mathbf{p}}{\partial \xi_{ud}} \end{bmatrix} \quad (15)$$

where $\frac{\partial \mathbf{p}}{\partial \mathbf{r}}$ is a 2×6 matrix and $\frac{\partial \mathbf{p}}{\partial \xi_{ud}}$ is a 2×5 matrix. Considering a calibration with n points, the full image Jacobian is given by the $2n \times 11$ matrix:

$$\mathbf{H} = (\mathbf{H}_{\mathbf{p}_1}, \mathbf{H}_{\mathbf{p}_2}, \dots, \mathbf{H}_{\mathbf{p}_n}) \quad (16)$$

One of the interest of this approach is that it is possible to consider the background in visual servoing. The image Jacobian $\frac{\partial \mathbf{p}}{\partial \mathbf{r}}$ that relates the motion of a point in the image to the camera motion is quite classical [1, 2] and is given by:

$$\frac{\partial \mathbf{p}}{\partial \mathbf{r}} = \begin{bmatrix} p_x & 0 \\ 0 & p_y \end{bmatrix} \begin{bmatrix} 1 + k_{ud}(3x^2 + y^2) & 2k_{ud}xy \\ 2k_{ud}xy & 1 + k_{ud}(x^2 + 3y^2) \end{bmatrix} \mathbf{L}_{\mathbf{p}} \quad (17)$$

with

$$\mathbf{L}_{\mathbf{p}} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \quad (18)$$

Furthermore, from (12), differentiating u and v for ξ_{ud} leads very easily to:

$$\frac{\partial \mathbf{p}}{\partial \xi_{ud}} = \begin{bmatrix} x(1 + k_{ud}(x^2 + y^2)) & 0 & 1 & 0 & x(x^2 + y^2) \\ 0 & y(1 + k_{ud}(x^2 + y^2)) & 0 & 1 & y(x^2 + y^2) \end{bmatrix} \quad (19)$$

We are now able to write the relation that links the motion of a point in the image to the camera motion and the variation of the intrinsic camera parameters.

Let us finally note that, if we do not want to consider distortion within the camera model, this equation can be simplified and we replace $\frac{\partial \mathbf{p}}{\partial \xi_{ud}}$ by:

$$\frac{\partial \mathbf{p}}{\partial \xi} = \begin{bmatrix} x & 0 & 1 & 0 \\ 0 & y & 0 & 1 \end{bmatrix} \quad (20)$$

with $\xi = (p_x, p_y, u_0, v_0)$

Considering the model with the five parameters $\xi_{du} = (p_x, p_y, u_0, v_0, k_{du})$, the expression of distortion is given by equation 14. According these equations, the new interaction matrix is easily obtained from :

$$\frac{\partial \mathbf{p}}{\partial \mathbf{r}} = \begin{bmatrix} p_x & 0 \\ 0 & p_y \end{bmatrix} \mathbf{L}_{\mathbf{p}} \quad (21)$$

with

$$\mathbf{L}_{\mathbf{p}} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \quad (22)$$

and :

$$\begin{aligned}
\frac{\partial u}{\partial u_0} &= 1 + k_{du} \left(3 \left(\frac{u-u_0}{p_x} \right)^2 + \left(\frac{v-v_0}{p_y} \right)^2 \right) \\
\frac{\partial u}{\partial v_0} &= 2k_{du} \left(\frac{(u-u_0)(v-v_0)}{p_y^2} \right) \\
\frac{\partial u}{\partial p_x} &= x + 2k_{du} \left(\frac{u-u_0}{p_x} \right)^3 \\
\frac{\partial u}{\partial p_y} &= 2k_{du} \frac{(u-u_0)(v-v_0)^2}{p_y^3} \\
\frac{\partial u}{\partial k_{du}} &= -(u - u_0) \left(\left(\frac{u-u_0}{p_x} \right)^2 + \left(\frac{v-v_0}{p_y} \right)^2 \right) \\
\frac{\partial v}{\partial u_0} &= 2k_{du} \left(\frac{(u-u_0)(v-v_0)}{p_x^2} \right) \\
\frac{\partial v}{\partial v_0} &= 1 + k_{du} \left(\left(\frac{u-u_0}{p_x} \right)^2 + 3 \left(\frac{v-v_0}{p_y} \right)^2 \right) \\
\frac{\partial v}{\partial p_x} &= 2k_{du} \frac{(u-u_0)^2(v-v_0)}{p_x^3} \\
\frac{\partial v}{\partial p_y} &= y + 2k_{du} \left(\frac{v-v_0}{p_y} \right)^3 \\
\frac{\partial v}{\partial k_{du}} &= -(v - v_0) \left(\left(\frac{u-u_0}{p_x} \right)^2 + \left(\frac{v-v_0}{p_y} \right)^2 \right)
\end{aligned} \tag{23}$$

1.5 Camera calibration in ViSP

Camera calibration tools are available in the `vpCalibration` class. Each instance of this class contains all the informations that are requested to calibrate one image :

- a list of 3D points and their corresponding 2D coordinates in the image,
- current camera parameters for the different models that are estimated,
- member functions to compute camera calibration,
- static member functions to compute multi-view camera calibration.

Here is an example of use of this class for a single-image calibration:

```

1  vpCalibration calibration;
2
3
4  //Add 3D points (in meters) and their 2D corresponding coordinates in the image (in pixels)
5  //in the calibration structure (X,Y,Z,u,v)
6  calibration.addPoint(0, 0, 0, vpImagePoint(55.2, 64.3) );
7  calibration.addPoint(0.1, 0, 0, vpImagePoint(245.1, 72.5) );
8  calibration.addPoint(0.1, 0.1, 0, vpImagePoint(237.2, 301.6) );
9  calibration.addPoint(0, 0.1, 0.1, vpImagePoint(34.4, 321.8) );
10
11 vpCameraParameters cam; //Camera parameters to estimate
12 vpHomogeneousMatrix cMo; //resulting pose of the object in the camera frame
13
14 //Compute the calibration with the desired method,
15 //here an initialisation with Lagrange method is done and after
16 //the virtual visual servoing method is used to finalized camera calibration
17 //for the perspective projection model with distortion.
18 calibration.computeCalibration(vpCalibration::CALIB_LAGRANGE_VIRTUAL_VS_DIST,cMo, cam);
19

```

```
20 //print camera parameters
21 std::cout << cam << std::endl;
```

For a multi-images calibration, you just have to initialize a table of `vpCalibration` objects (one for each image of the calibration grid) and use the static function :

```
1 vpCalibration::
2 computeCalibrationMulti(vpCalibrationMethodType method, //Desired method to compute
3                       //calibration
4                       unsigned int nbPose, //Number of views (size of table_cal)
5                       vpCalibration table_cal[], //Table of filled vpCalibration
6                       //structures
7                       vpCameraParameters &cam) //Camera parameters to estimate
```

References

- [1] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.
- [2] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.
- [3] P. Puget and T. Skordas. An optimal solution for mobile camera calibration. In *European Conf. on Computer Vision, ECCV'90*, volume 427 of *Lecture Notes in Computer Science*, pages 187–198, Antibes, France, April 1990.