

Blaise Pascal University  
Aubiere, France

Visual servoing labs

**Lab N°1**

**Sensor based control : visual servoing with 2D/3D points**

The aim of the lab is to use the visual servoing toolbox in order to analyse the efficiency of visual servoing when using 2D point features (image features) or 3D point features.

We will use the version 2.0 of the Visual Servoing toolbox.

First, we will concentrate on the use of 2D point features. We will use different control laws, and compare them (advantages and disadvantages). Different case studies are proposed in this way.

Second, we will switch to the use of 3D point in visual servoing. We will discover in practice some interesting properties.

Finally, we will try to compare both approaches.

Philippe Martinet  
Professor at IFMA  
Researcher at LASMEA, Blaise Pascal University  
Clermont-Ferrand, France

# 1 Theoretical modeling

## 1.1 Interaction matrix

Considering the following definitions :

- $(x, y, z)^T$  : coordinates of a 3D point  $p$
- $(X, Y)^T$  : coordinates of the corresponding perspective projected point  $P$  in perspective plane
- $(u, v)^T$  : coordinates of the corresponding projected point  $P_{im}$  in image plane
- $f_u$ , and  $f_v$  focal length in  $u$  and  $v$  direction
- $u_0$  and  $v_0$  : coordinates of the optical center in image plane.

So, we have :

$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (1)$$

$$P = \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \quad (2)$$

$$P_{im} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_u & 0 \\ 0 & f_v \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \quad (3)$$

where  $(f_u, f_v, u_0, v_0)$  represent the intrinsic parameters of the camera.

The expression of the different interaction matrices is the following :

$$L_p = \begin{bmatrix} -I_3 & \tilde{p} \end{bmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 & -z & y \\ 0 & -1 & 0 & z & 0 & -x \\ 0 & 0 & -1 & -y & x & 0 \end{pmatrix} \quad (4)$$

$$L_P = \begin{pmatrix} -\frac{1}{z} & 0 & \frac{X}{z} & X \cdot Y & -1 - X^2 & Y \\ 0 & -\frac{1}{z} & \frac{Y}{z} & 1 + Y^2 & -X \cdot Y & -X \end{pmatrix} \quad (5)$$

$$L_{P_{im}} = \begin{pmatrix} -\frac{f_u}{z} & 0 & \frac{u}{z} & \frac{u \cdot v}{f_v} & -f_u - \frac{u^2}{f_u} & \frac{f_u}{f_v} \cdot v \\ 0 & -\frac{f_v}{z} & \frac{v}{z} & f_v + \frac{v^2}{f_v} & -\frac{u \cdot v}{f_u} & -\frac{f_v}{f_u} \cdot u \end{pmatrix} \quad (6)$$

## 1.2 Sensor based control

Considering one robot (for instance, one manipulator robot) where a sensor apparatus is embedded on the end effector, then the sensor information vector  $\underline{s}$  is a function of the current position of the sensor apparatus  $\underline{r}$  and of time  $t$  (the environment can change over the time) :

$$\underline{s} = \underline{s}(\underline{r}, t) \quad (7)$$

By derivating equation 7, we obtain :

$$\dot{\underline{s}} = \frac{\delta \underline{s}}{\delta \underline{r}} \cdot \frac{d\underline{r}}{dt} + \frac{\delta \underline{s}}{\delta t} = L_{\underline{s}}^T \cdot T + \frac{\delta \underline{s}}{\delta t} \quad (8)$$

where

- $T$  represents the control command (kinematic screw) applied to the sensor apparatus,
- $L_{\underline{s}}^T$  represents the interaction matrix (related to the content of  $\underline{s}$ ),

- $\frac{\delta \underline{s}}{\delta t}$  is the contribution of the environment motion (i.e. target motion)

The sensor based control uses the notion of *TASK FUNCTION*  $\underline{e}$  (or error function) defined by :

$$\underline{e} = C \cdot (\underline{s}(r, t) - \underline{s}^*) \quad (9)$$

where

- $C$  is a combination matrix which permits to take into account more sensor information than necessary to perform the positioning task,
- $\underline{s}(r, t)$  the current sensor information vector,
- $\underline{s}^*$  the desired sensor information vector at equilibrium (when the positioning task is performed).

The main aim is to regulate to zero this task function. So, we impose an exponential decrease, that means :

$$\dot{\underline{e}} = -\lambda \cdot \underline{e} \quad (10)$$

and then we deduce the following control law :

$$U = T = -\lambda \cdot \underline{e} = -\lambda \cdot L_{\underline{s}}^{T+} \cdot (\underline{s}(r, t) - \underline{s}^*) \quad (11)$$

This control law represents in fact, a first order control law. The choice of the interaction matrix, can be done :

- at each iteration. In this case, its value is evaluated at each iteration taking into account the novel values of the sensor information. For this case, we will use the interaction matrix  $L_{\underline{s}}^T = \widehat{L}_{\underline{s}}^T$  and the control law will become  $T = -\lambda \cdot \widehat{L}_{\underline{s}}^{T+} \cdot (\underline{s}(r, t) - \underline{s}^*)$ ,

$$\underline{s}^* \approx \underline{s} + \widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t \quad (12)$$

$$\Delta \underline{s} \approx -\widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t \quad (13)$$

$$\mathbf{T} = -\lambda (\widehat{L}_{\underline{s}} \widehat{L}_{\underline{s}}^T)^{-1} \widehat{L}_{\underline{s}} \Delta \underline{s} = -\lambda \widehat{L}_{\underline{s}}^{T+} \Delta \underline{s} \quad (14)$$

- at equilibrium. In this case, its value is evaluated with the value of  $\underline{s}^*$  corresponding to the desired position. For this case, we will use the interaction matrix  $L_{\underline{s}}^T = \widehat{L}_{\underline{s}=\underline{s}^*}^T$  and the control law will become  $T = -\lambda \cdot \widehat{L}_{\underline{s}=\underline{s}^*}^{T+} \cdot (\underline{s}(r, t) - \underline{s}^*)$

$$\underline{s} \approx \underline{s}^* + \widehat{L}_{\underline{s}=\underline{s}^*}^T \mathbf{T} \Delta t \quad (15)$$

$$\Delta \underline{s} \approx -\widehat{L}_{\underline{s}=\underline{s}^*}^T \mathbf{T} \Delta t \quad (16)$$

$$\mathbf{T} = -\lambda (\widehat{L}_{\underline{s}=\underline{s}^*} \widehat{L}_{\underline{s}=\underline{s}^*}^T)^{-1} \widehat{L}_{\underline{s}=\underline{s}^*} \Delta \underline{s} = -\lambda \widehat{L}_{\underline{s}=\underline{s}^*}^{T+} \Delta \underline{s} \quad (17)$$

It is possible to define a second order control law. The second order approximation of  $\Delta \underline{s}$  is given by equation 18, where  $\mathbf{M}(\mathbf{T} \Delta t)$  represents the Hessian matrix.  $\mathbf{M}(\mathbf{T} \Delta t)$  can be approximated by equation 19 (finite differences), that gives a novel expression (equation 21) for  $\Delta \underline{s}$ . Then, a second order control law can be established using equation 22.

$$\Delta \underline{s} \approx -\widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t - \frac{1}{2} \mathbf{M}(\mathbf{T} \Delta t) \mathbf{T} \Delta t \quad (18)$$

$$\widehat{L}_{\underline{s}=\underline{s}^*}^T \approx \widehat{L}_{\underline{s}}^T + \mathbf{M}(\mathbf{T} \Delta t) \quad (19)$$

$$\mathbf{M}(\mathbf{T} \Delta t) \approx \widehat{L}_{\underline{s}=\underline{s}^*}^T - \widehat{L}_{\underline{s}}^T \quad (20)$$

$$\Delta \underline{s} \approx -\frac{1}{2} (\widehat{L}_{\underline{s}=\underline{s}^*}^T + \widehat{L}_{\underline{s}}^T) \mathbf{T} \Delta t \quad (21)$$

$$\mathbf{T} = -2\lambda (\widehat{L}_{\underline{s}=\underline{s}^*}^T + \widehat{L}_{\underline{s}}^T)^+ \Delta \underline{s} \quad (22)$$

To conclude, it exists two kind of first order control laws (equations 14 and 17) and one kind of second order control law (22) in order to implement visual servoing schemes.

## 2 Introduction to the Visual Servoing Toolbox v2.0

After starting matlab, select the working directory **VservingToolbox2.0**. In the menu **File Menu**, select the item **set Path**. Append the path to the **Visual Servoing Toolbox** using the item **Add with subfolders**. Save with the option **Save**, and then quit using **Close**. In the matlab workspace, select the **VservingToolbox2.0/vsbrowser** directory. Execute the **VSBrowser** when running **initvsb**. The graphic user interface with its different elements are presented by figure 1.

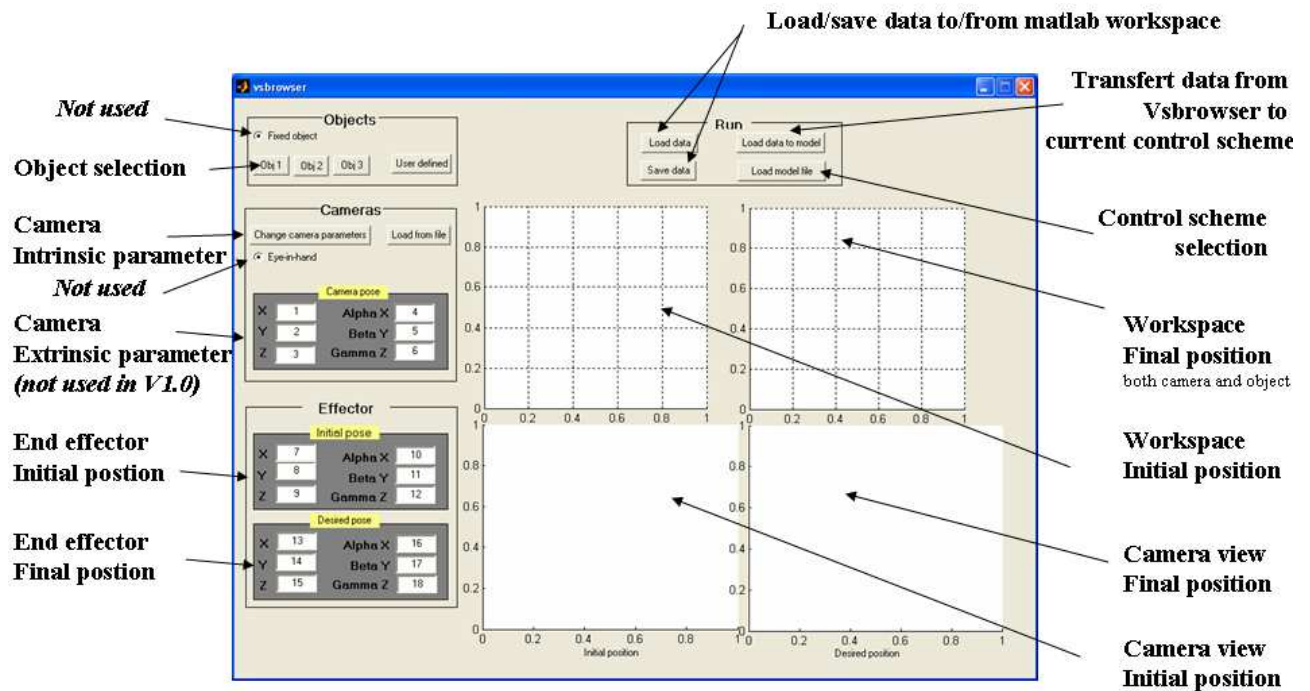


FIGURE 1 – Vsbrowser Environment

When using the VSBrowser, follow the different steps :

- Load a visual servoing scheme by using the button **Load model file** (absolutely necessary). The different available models are stored in the directory **VservingToolbox2.0/vstoolbox/Models**
- Select an objet (if not, the default object is selected).
- Configure and validate the initial and final positions of the end effector (if not, default positions are proposed).
- Load the intrinsic parameters of the camera (if not, default parameters values are proposed).
- Transfer all the values from the GUI to the matlab/simulink workspace using the button **Load data to model**.
- Select the simulink block. Some blocks are parametrized. To access to the parametrization, you have to double click on the corresponding block.
- Run the simulation.
- To visualize the results, in the workspace of matlab use **view2Dpoint**, or **view3Dpoint** ... for the 2D visual servoing or 3D visual servoing respectively.

For more precisions, have a look on the annexe.



### 3 Simulation using 2D point features

Load the model file **ibvs.mdl** using the button **load model file** of the VS Browser (in the directory **vstoolbox/Models**).

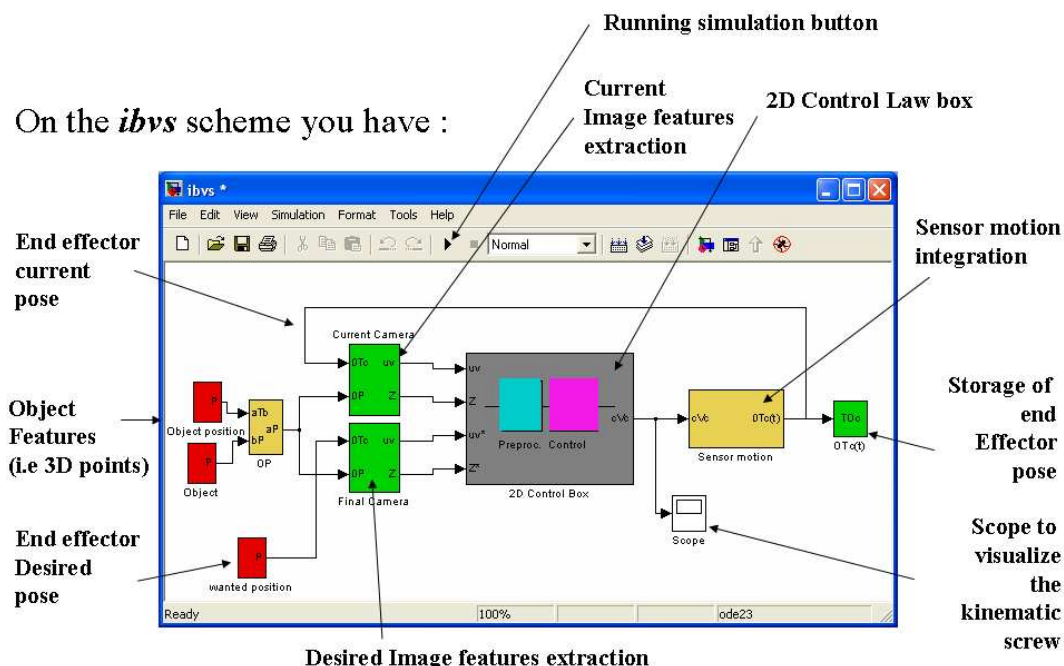


FIGURE 2 – IBVS with 2D point features

Use the different values for the end effector pose at the desired position.

cases	position	orientation	cases	position	orientation
1	0,0,-1.5	0,0,30	4	0,0,-3	0,0,0
2	0,0,-1.5	0,0,110	5	0.6,0.9,-4	0,0,0
3	0,0,-1.5	0,0,170	6	0.6,0.9,-2	-20,-20,130

Different case studies

For each case study, enter the new values for the desired end effector pose. Transfer the novel values with a click on **load data to model**. For the three control laws, run the simulation.

With the mouse, double click on the **2D Control Box**

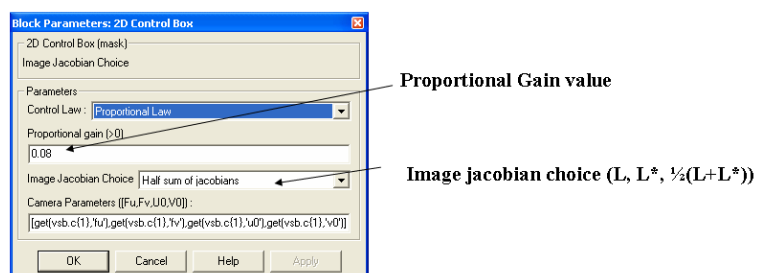


FIGURE 3 – Choice of the control law

Analyse and compare the results (using the m-file **view2Dpoint**). In the figure 10000, think to change the color of the trajectory between each simulation.

## 4 Simulation using 3D point features

Load the model file **ibvs3dp.mdl** using the button **load model file** of the VS Browser (in the directory **vstoolbox/Models**).

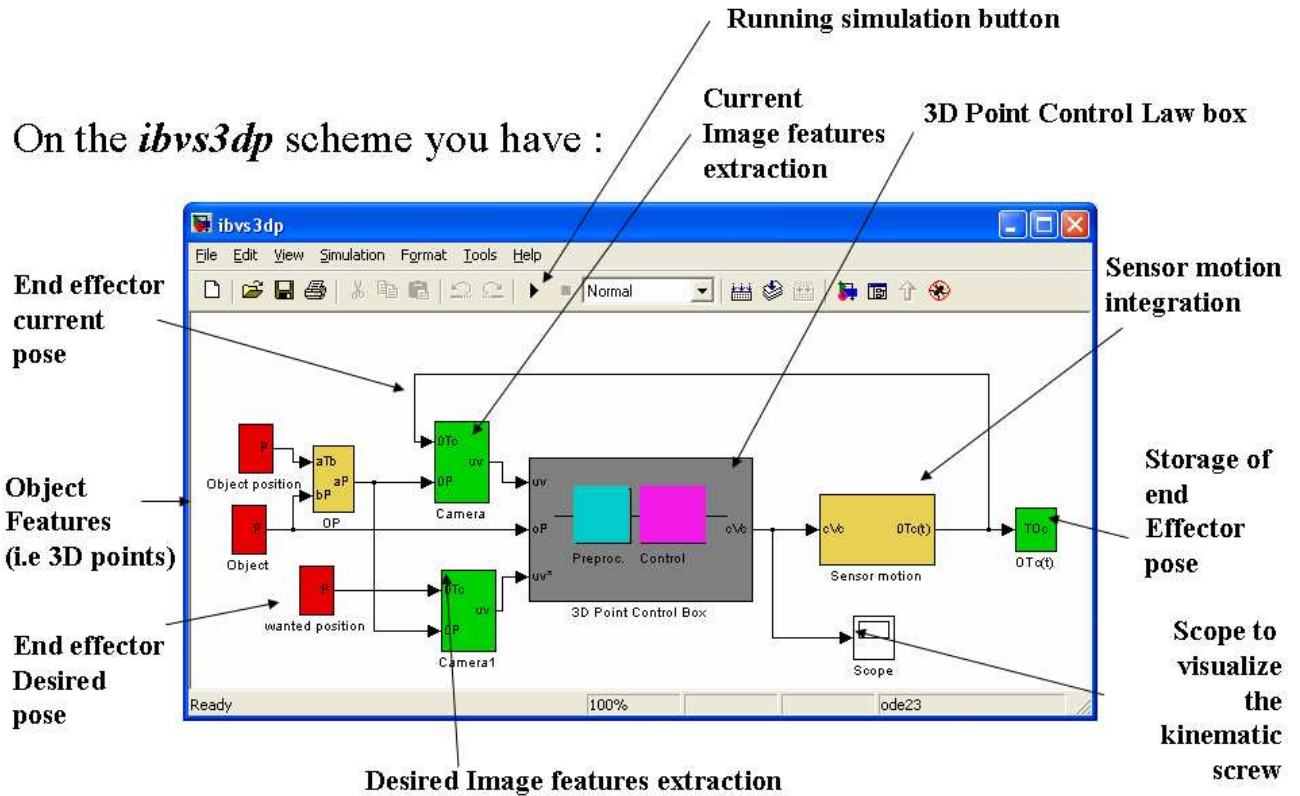


FIGURE 4 – PBVS with 3D points

Use the different values for the end effector pose at the desired position.

cases	position	orientation	cases	position	orientation
1	0,0,-1.5	0,0,30	4	0,0,-3	0,0,0
2	0,0,-1.5	0,0,110	5	0.6,0.9,-4	0,0,0
3	0,0,-1.5	0,0,170	6	0.6,0.9,-2	-20,-20,130

Different case studies

For each case study, enter the new values for the desired end effector pose. Transfer the novel values with a click on **load data to model**. For the three control laws, run the simulation.

Analyse and compare the results (using the m-file **view3Dpoint**). In the figure 10000, think to change the color of the trajectory between each simulation.

## 5 Comparison of 2D and 3D point visual servoing

Compare both approaches (2D and 3D point visual servoing). For that, use the following case studies :

cases	position	orientation	cases	position	orientation
3	0,0,-1.5	0,0,170	6	0.6,0.9,-2	-20,-20,130

Different case studies

Quit matlab and close your account at the end of the lab.

## Annexe : VSB Toolbox

After starting matlab, select the working directory **VservoingToolbox2.0**. In the menu **File Menu**, select the item **set Path**. Append the path to the **Visual Servoing Toolbox** using the item **Add with subfolders**. Save with the option **Save**, and then quit using **Close**. In the matlab workspace, select the **VservoingToolbox2.0/vsbrowser** directory. Execute the **VSBrowser** when running **initvsb**. The graphic user interface with its different elements are presented by figure 5.

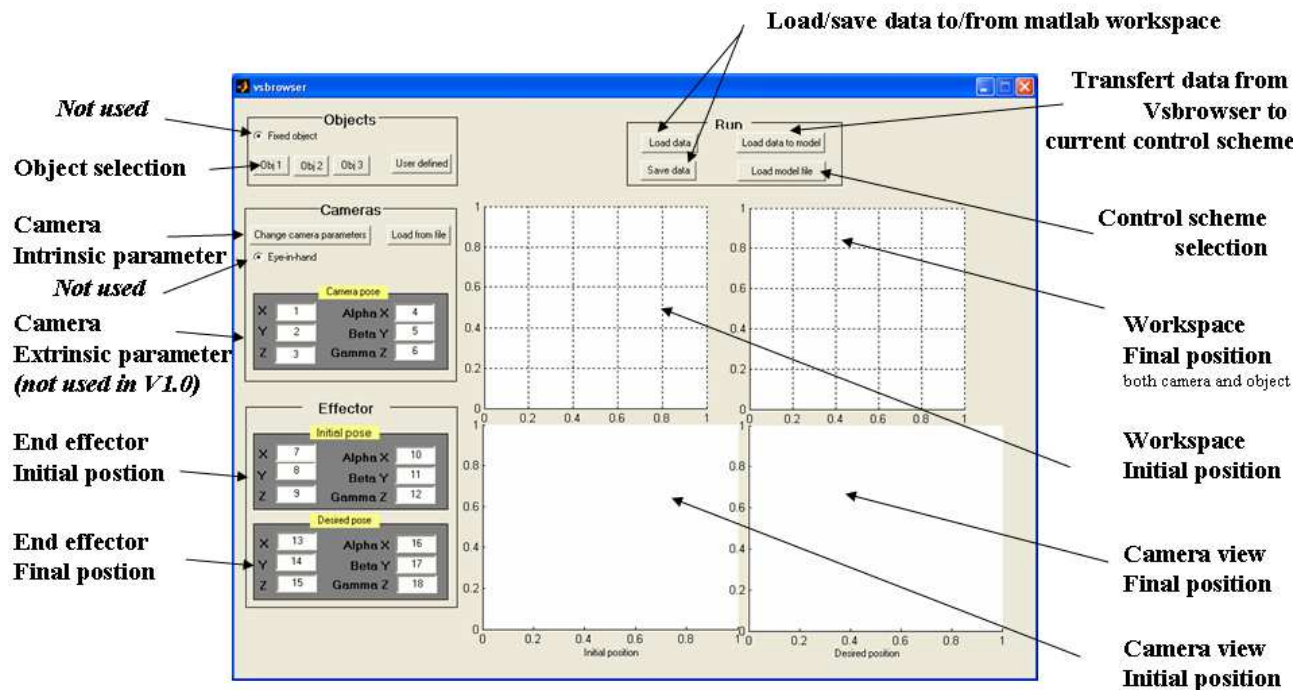


FIGURE 5 – Vsbrowser environment

When using the VSBrowser, follow the different steps :

- Load a visual servoing scheme by using button **Load model file** (absolutely necessary). The different available models are stored in the directory **VservoingToolbox2.0/vstoolbox/Models**. The figure 6 represents the result after loading **pbvs.mdl**.
- Select an objet (if not, the default object is selected). The figure 7 illustrates the result of the selection of default object and model  $n^o 2$ .
- Configure and validate the initial and final positions of the end effector (if not, default positions are proposed).
- Load the intrinsic parameters of the camera (if not, default parameters values are proposed).
- Transfer all the values from the GUI to the matlab/simulink workspace using the button **Load data to model**.
- Select the simulink block. Some blocks are parametrized. To access to the parametrization, you have to double click on the corresponding block.
- Run the simulation.
- to visualize the results, in the workspace of matlab use **view2Dpoint**, or **view3Dpoint**, or **view3Dpose** or **view2dhalf** for the 2D visual servoing, or 3D visual servoing or 2D1/2 visual servoing respectively.

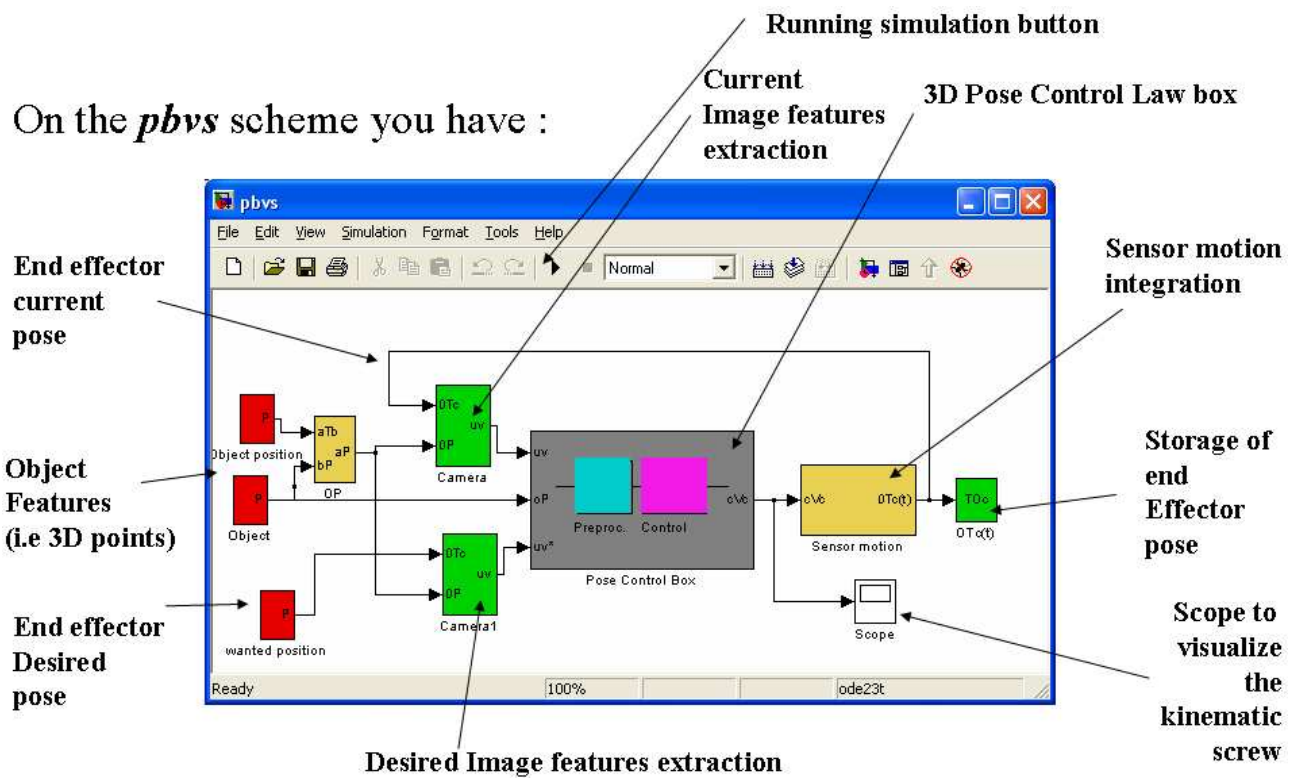
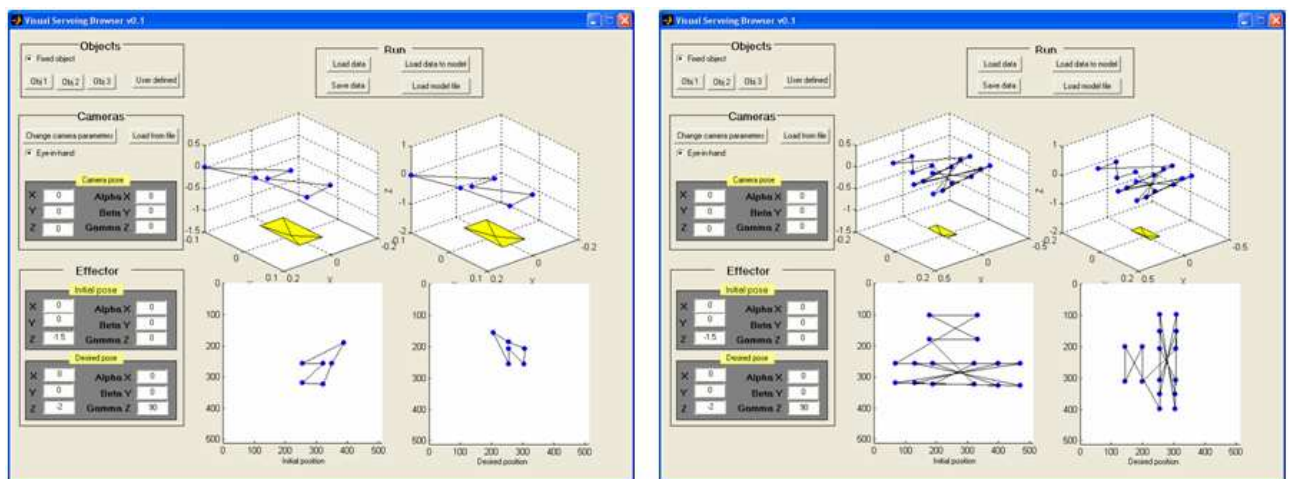


FIGURE 6 – PBVS : 3D pose



Default object model

Object model 2

FIGURE 7 – Object choice

Results visualization with *view2dpoint*

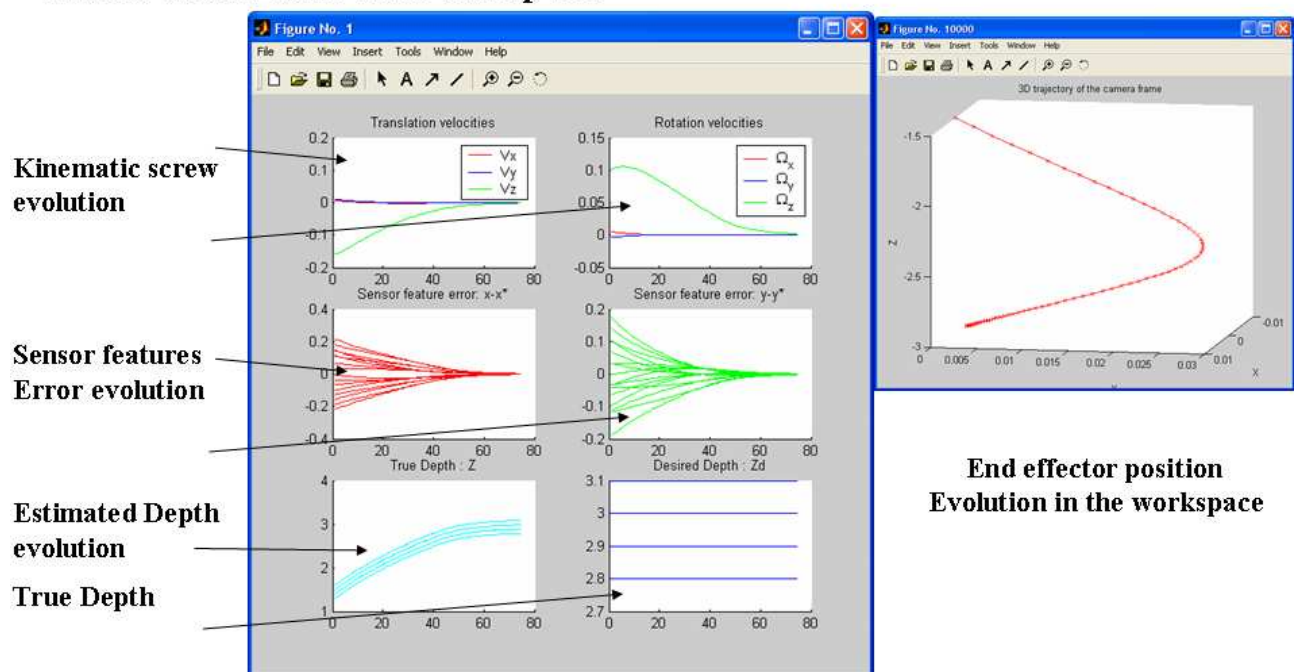


FIGURE 8 – Simulation results visualization

Blaise Pascal University  
Aubiere, France

Visual servoing labs

Lab N°2

**Sensor based control : 3D pose and 2D1/2 visual servoing**

The aim of the lab is to use the visual servoing toolbox in order to analyse the efficiency of visual servoing when using 3D pose features or 2D1/2 features.

We will use the version 2.0 of the Visual Servoing toolbox.

First, we will concentrate on the use of 3D pose features. We will use different parametrization for the orientation part ( $\mathbf{u} \cdot \theta, \mathbf{u} \cdot \sin(\theta), \mathbf{u} \cdot \sin(\frac{\theta}{2})$ ), evaluate different control laws (exact linearization, tangent linearization) , and compare them (advantages and disadvantages). Different case studies are proposed in this way.

Second, we will switch to the use of 2D1/2 visual servoing. The same kind of work will be proposed.

Philippe Martinet  
Professor at IFMA  
Researcher at LASMEA, Blaise Pascal University  
Clermont-Ferrand, France



# 1 Theoretical modeling

## 1.1 Sensor based control

Considering one robot (for instance, one manipulator robot) where a sensor apparatus is embedded on the end effector, then the sensor information vector  $\underline{s}$  is a function of the current position of the sensor apparatus  $\underline{r}$  and of time  $t$  (the environment can change over the time) :

$$\underline{s} = \underline{s}(\underline{r}, t) \quad (1)$$

By derivating equation 1, we obtain :

$$\dot{\underline{s}} = \frac{\delta \underline{s}}{\delta \underline{r}} \cdot \frac{d\underline{r}}{dt} + \frac{\delta \underline{s}}{\delta t} = L_{\underline{s}}^T \cdot T + \frac{\delta \underline{s}}{\delta t} \quad (2)$$

where

- $T$  represents the control command (kinematic screw) applied to the sensor apparatus,
- $L_{\underline{s}}^T$  represents the interaction matrix (related to the content of  $\underline{s}$ ),
- $\frac{\delta \underline{s}}{\delta t}$  is the contribution of the environment motion (i.e. target motion)

The sensor based control uses the notion of *TASK FUNCTION*  $\underline{e}$  (or error function) defined by :

$$\underline{e} = C \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*) \quad (3)$$

where

- $C$  is a combination matrix which permits to take into account more sensor information than necessary to perform the positioning task,
- $\underline{s}(\underline{r}, t)$  the current sensor information vector,
- $\underline{s}^*$  the desired sensor information vector at equilibrium (when the positioning task is performed).

The main aim is to regulate to zero this task function. So, we impose an exponential decrease, that means :

$$\dot{\underline{e}} = -\lambda \cdot \underline{e} \quad (4)$$

and then we deduce the following control law :

$$U = T = -\lambda \cdot \underline{e} = -\lambda \cdot L_{\underline{s}}^{T+} \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*) \quad (5)$$

This control law represents in fact, a first order control law. The choice of the interaction matrix, can be done :

- at each iteration. In this case, its value is evaluated at each iteration taking into account the novel values of the sensor information. For this case, we will use the interaction matrix  $L_{\underline{s}}^T = \widehat{L}_{\underline{s}}^T$  and the control law will become  $T = -\lambda \cdot \widehat{L}_{\underline{s}}^{T+} \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*)$ ,

$$\underline{s}^* \approx \underline{s} + \widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t \quad (6)$$

$$\Delta \underline{s} \approx -\widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t \quad (7)$$

$$\mathbf{T} = -\lambda (\widehat{L}_{\underline{s}} \widehat{L}_{\underline{s}}^T)^{-1} \widehat{L}_{\underline{s}} \Delta \underline{s} = -\lambda \widehat{L}_{\underline{s}}^{T+} \Delta \underline{s} \quad (8)$$

- at equilibrium. In this case, its value is evaluated with the value of  $\underline{s}^*$  corresponding to the desired position. For this case, we will use the interaction matrix  $L_{\underline{s}}^T = \widehat{L}_{\underline{s}=\underline{s}^*}^T$  and the control law will become  $T = -\lambda \cdot \widehat{L}_{\underline{s}=\underline{s}^*}^{T+} \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*)$

$$\underline{s} \approx \underline{s}^* + \widehat{L}_{\underline{s}=\underline{s}^*}^T \mathbf{T} \Delta t \quad (9)$$

$$\Delta \underline{s} \approx -\widehat{L}_{\underline{s}=\underline{s}^*}^T \mathbf{T} \Delta t \quad (10)$$

$$\mathbf{T} = -\lambda (\widehat{L}_{\underline{s}=\underline{s}^*} \widehat{L}_{\underline{s}=\underline{s}^*}^T)^{-1} \widehat{L}_{\underline{s}=\underline{s}^*} \Delta \underline{s} = -\lambda \widehat{L}_{\underline{s}=\underline{s}^*}^{T+} \Delta \underline{s} \quad (11)$$



It is possible to define a second order control law. The second order approximation of  $\Delta \mathbf{s}$  is given by equation 12, where  $\mathbf{M}(\mathbf{T}\Delta t)$  represents the Hessian matrix.  $\mathbf{M}(\mathbf{T}\Delta t)$  can be approximated by equation 13 (finite differences), that gives a novel expression (equation 15) for  $\Delta \mathbf{s}$ . Then, a second order control law can be established using equation 16.

$$\Delta \mathbf{s} \approx -\widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t - \frac{1}{2} \mathbf{M}(\mathbf{T} \Delta t) \mathbf{T} \Delta t \quad (12)$$

$$\widehat{L}_{\underline{s}=\underline{s}^*}^T \approx \widehat{L}_{\underline{s}}^T + \mathbf{M}(\mathbf{T} \Delta t) \quad (13)$$

$$\mathbf{M}(\mathbf{T} \Delta t) \approx \widehat{L}_{\underline{s}=\underline{s}^*}^T - \widehat{L}_{\underline{s}}^T \quad (14)$$

$$\Delta \mathbf{s} \approx -\frac{1}{2} (\widehat{L}_{\underline{s}=\underline{s}^*}^T + \widehat{L}_{\underline{s}}^T) \mathbf{T} \Delta t \quad (15)$$

$$\mathbf{T} = -2\lambda (\widehat{L}_{\underline{s}=\underline{s}^*}^T + \widehat{L}_{\underline{s}}^T)^+ \Delta \mathbf{s} \quad (16)$$

To conclude, it exists two kind of first order control laws (equations 8 and 11) and one kind of second order control law (16) in order to implement visual servoing schemes.

## 1.2 Interaction matrix

### 1.2.1 3D Pose

Figure 1 represents the scene. The sensor is embedded on the end effector of the robot. The aim of the control law is to position the current frame  $\mathcal{R}_C$  (rigidly linked to the end effector) in a particular configuration materialized by the frame  $\mathcal{R}_F$ . The sensor signal  $\mathbf{s} = (\mathbf{x}^T, \mathbf{y}^T)^T$  is

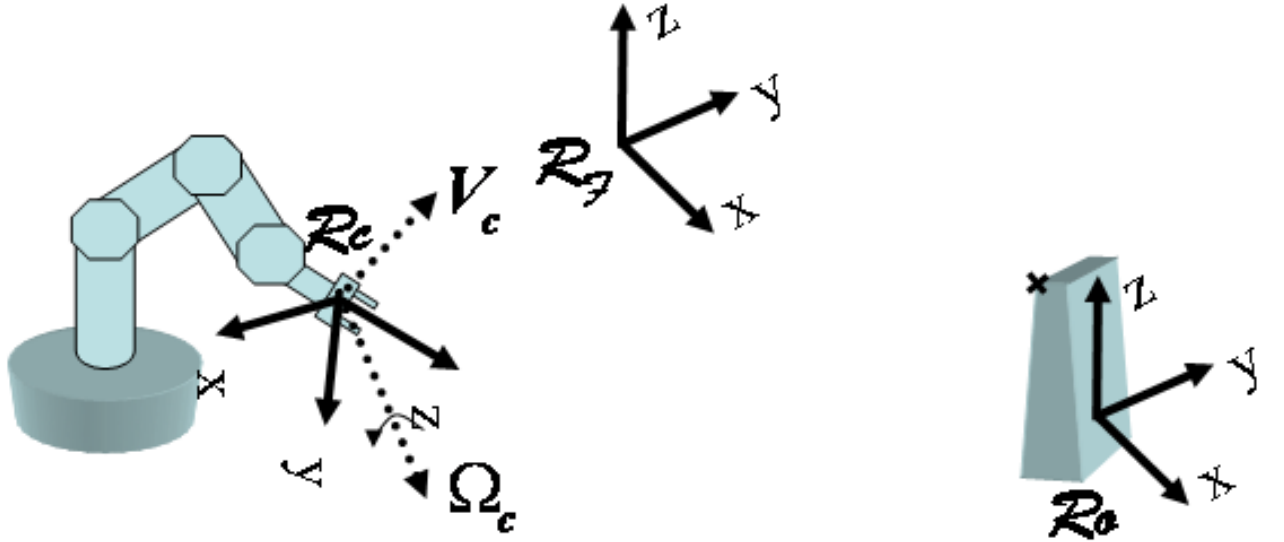


FIGURE 1 – Description of the scene

composed of :

- $\mathbf{x}$  : the position of the frame  $\mathcal{R}_C$  in the frame  $\mathcal{R}_F$
- $\mathbf{y}$  : the orientation of the frame  $\mathcal{R}_C$  in the frame  $\mathcal{R}_F$

The general form of the interaction matrix at each iteration is given by :

$$L_s = \begin{pmatrix} R & O_3 \\ O_3 & B(y) \cdot R \end{pmatrix} \quad (17)$$

Then, we can deduce that at equilibrium (as  $\mathbf{s}^* = (\mathbf{O}, \mathbf{O})^T$ ) :

$$L_{\mathbf{s}} = \begin{pmatrix} I_3 & O_3 \\ O_3 & I_3 \end{pmatrix} = I_6 \quad (18)$$

In the following, we will consider 3 cases for the parametrization of  $\mathbf{y}$  :  $\mathbf{y}_1 = \mathbf{u} \sin \theta$ ,  $\mathbf{y}_2 = \mathbf{u} \sin \frac{\theta}{2}$  and  $\mathbf{y}_3 = \mathbf{u} \theta$

### 1.2.2 Hybrid visual servoing

In this case, an hybrid sensor signal. In the  $2D \frac{1}{2}$  approach,  $\mathbf{s} = (\mathbf{p}^T, d, \mathbf{y}^T)^T$  is composed of :

- $\mathbf{p}$  : one 2D point which coordinates are  $(x_n, y_n)^T$
- $d = \log \frac{Z}{Z^*}$  : the logarithm of the relative depth of a point
- $\mathbf{y}$  : the 3D orientation (i.e.  $\mathbf{y}_3$ )

Then, the corresponding interaction matrix becomes :

$$\mathbf{L}^T = \begin{bmatrix} \mathbf{L}_v & \mathbf{L}_{(v,\omega)} \\ \mathbf{0} & \mathbf{L}_\omega \end{bmatrix}$$

with  $\mathbf{L}_v = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x_n}{Z} \\ 0 & -\frac{1}{Z} & \frac{y_n}{Z} \\ 0 & 0 & -\frac{1}{Z} \end{bmatrix}$  et  $\mathbf{L}_{(v,\omega)} = \begin{bmatrix} x_n y_n & -1 - x_n^2 & y_n \\ 1 + y_n^2 & -x_n y_n & -x_n \\ -y_n & x_n & 0 \end{bmatrix}$

and  $\mathbf{L}_\omega = \mathbf{I} - \frac{\theta}{2} [\mathbf{u}]_\times + \left(1 - \frac{\text{sinc}(\theta)}{\text{sinc}^2(\frac{\theta}{2})}\right) [\mathbf{u}]_\times^2$

## 2 Introduction to the Visual Servoing Toolbox v2.0

After starting matlab, select the working directory **VservingToolbox2.0**. In the menu **File Menu**, select the item **set Path**. Append the path to the **Visual Servoing Toolbox** using the item **Add with subfolders**. Save with the option **Save**, and then quit using **Close**. In the matlab workspace, select the **VservingToolbox2.0/vsbrowser** directory. Execute the **VSBrowser** when running **initvsb**. The graphic user interface with its different elements are presented by figure 2.

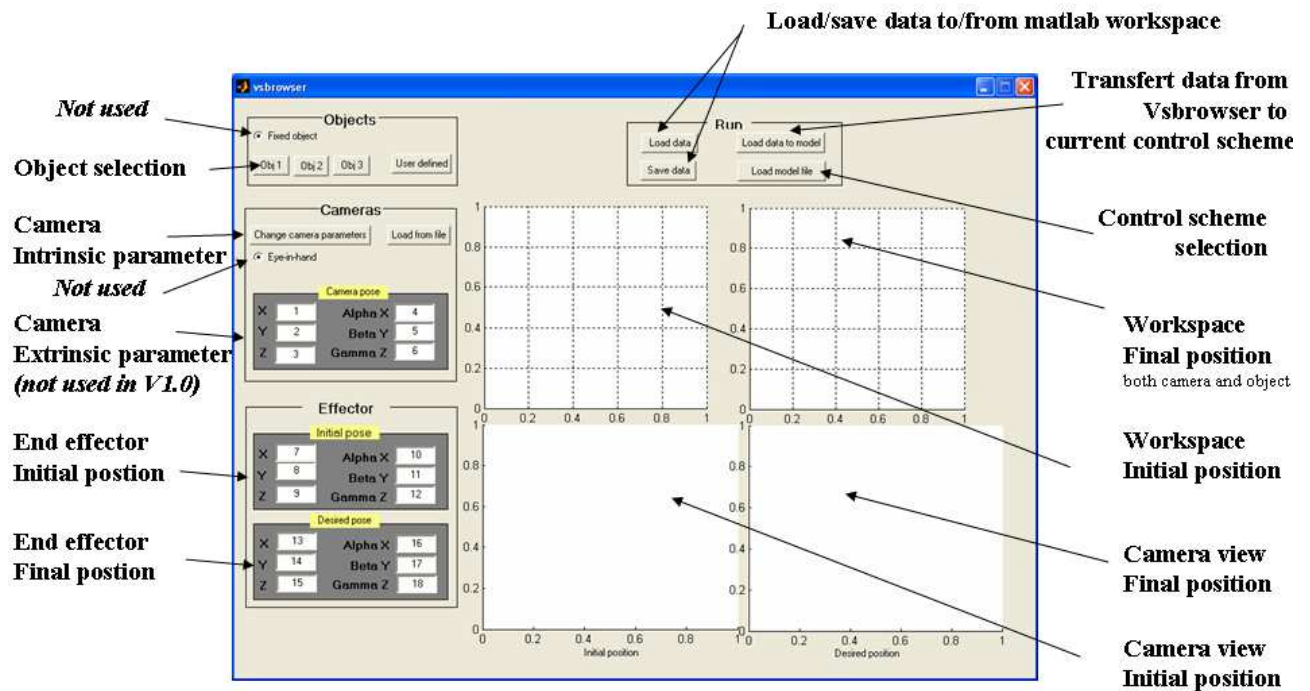


FIGURE 2 – Vsbrowser Environment

When using the VSBrowser, follow the different steps :

- Load a visual servoing scheme by using button **Load model file** (absolutely necessary). The different available models are stored in the directory **VservingToolbox2.0/vstoolbox/Models**
- Select an objet (if not, the default object is selected).
- Configure and validate the initial and final positions of the end effector (if not, default positions are proposed).
- Load the intrinsic parameters of the camera (if not, default parameters values are proposed).
- Transfer all the values from the GUI to the matlab/simulink workspace using the button **Load data to model**.
- Select the simulink block. Some blocks are parametrized. To access to the parametrization, you have to double click on the corresponding block.
- Run the simulation.
- to visualize the results, in the workspace of matlab use **view3Dpose**, or **view2Dhalf ...** for the 3D pose visual servoing or 2D1/2 visual servoing respectively.

For more precisions, have a look on the annexe.

### 3 Sensor based control : 3D pose $y = u \cdot \sin(\theta)$

On the *pbvs* scheme you have :

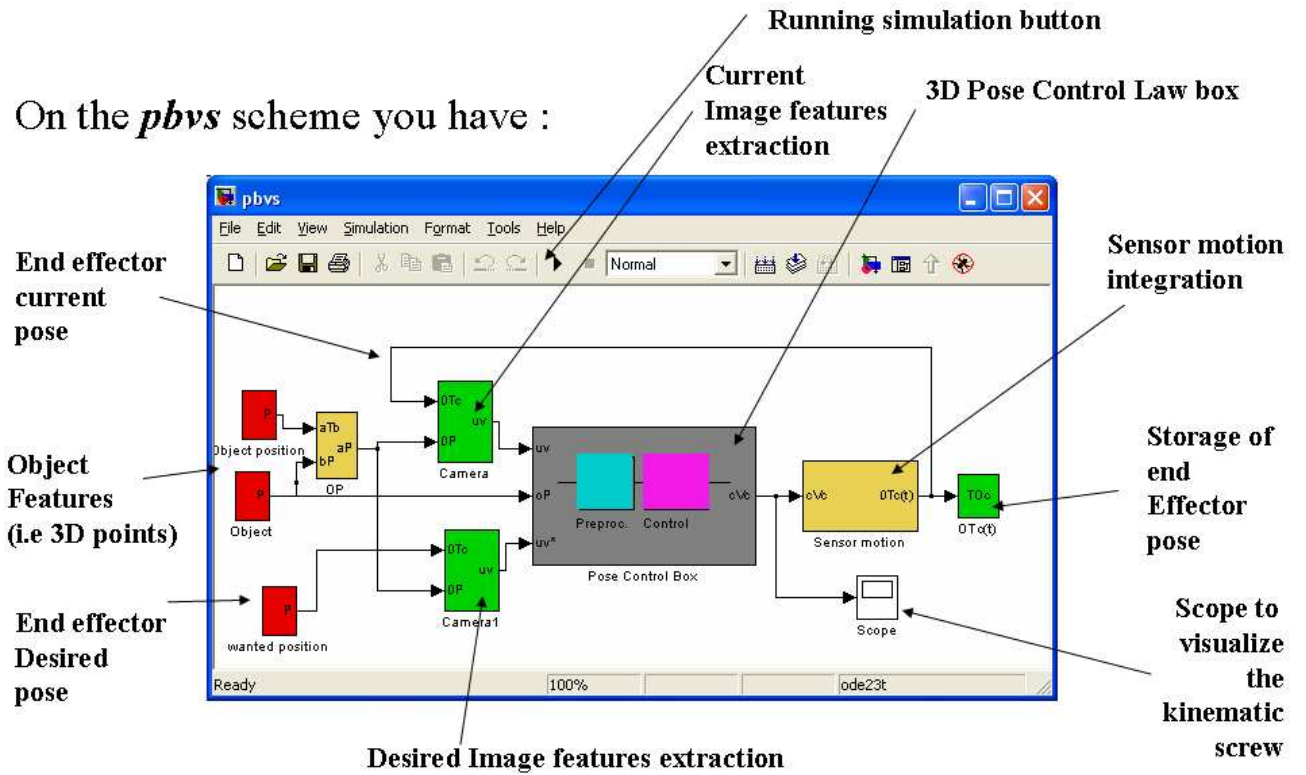


FIGURE 3 – PBVS : pose 3D

Load the model **pbvs.mdl** (figure 3) using the button **load model file** of the VS Browser (in the directory **vstoolbox/Models**).

#### Theoretical modeling

The parametrization  $y_1 = u \cdot \sin(\theta)$  give us :

$$L_{y_1} = [ O_3 \quad B(y_1) \cdot R ] \quad (19)$$

with

$$B(y_1) = \frac{1}{2} (tr(R)I - R) = \cos \theta I - \frac{\sin \theta}{2} [u]_{\times} + \frac{\cos(\theta) - 1}{2} [u]_{\times}^2 \quad (20)$$

#### Simulation

Use the different values for the end effector pose at the desired position.

cases	position	orientation	cases	position	orientation
1	0,0,-1.5	0,0,30	4	0,0,-3	0,0,0
2	0,0,-1.5	0,0,110	5	0.6,0.9,-4	0,0,0
3	0,0,-1.5	0,0,180	6	0.6,0.9,-2	-20,-20,130

Different case studies

For each case study, enter the new values for the desired end effector pose. Transfer the novel values with a click on **load data to model**. For the three control laws, run the simulation.

Analyse and compare the results (using the m-file **view3dpose**). In the figure 10000, think to change the color of the trajectory between each simulation.

## 4 Sensor based control : 3D pose $\mathbf{y} = \mathbf{u} \cdot \sin(\frac{\theta}{2})$

### Theoretical modeling

The parametrization  $\mathbf{y}_2 = \mathbf{u} \cdot \sin(\frac{\theta}{2})$  give us :

$$L_{y_2} = [ O_3 \quad B(y_2) \cdot R ] \quad (21)$$

with

$$B(y_2) = \frac{1}{2} \left( \cos \frac{\theta}{2} I - \sin \frac{\theta}{2} [u]_{\times} \right) \quad (22)$$

$$B^{-1}(y_2) = \frac{2}{\cos \frac{\theta}{2}} I + 2 \sin \frac{\theta}{2} [u]_{\times} + 2 \frac{\sin^2 \frac{\theta}{2}}{\cos \frac{\theta}{2}} [u]_{\times}^2 \quad (23)$$

### Simulation

Use the different values for the end effector pose at the desired position.

cases	position	orientation	cases	position	orientation
1	0,0,-1.5	0,0,30	4	0,0,-3	0,0,0
2	0,0,-1.5	0,0,110	5	0.6,0.9,-4	0,0,0
3	0,0,-1.5	0,0,180	6	0.6,0.9,-2	-20,-20,130

Different case studies

For each case study, enter the new values for the desired end effector pose. Transfer the novel values with a click on **load data to model**. For the three control laws, run the simulation.

Analyse and compare the results (using the m-file **view3dpose**). In the figure 10000, think to change the color of the trajectory between each simulation.

## 5 Sensor based control : 3D pose $\mathbf{y} = \mathbf{u} \cdot \theta$

### Theoretical modeling

The parametrization  $\mathbf{y}_3 = \mathbf{u} \cdot \theta$  gives us :

$$L_{y_3} = [ O_3 \quad B(y_3) \cdot R ] \quad (24)$$

with

$$B(y_3) = I - \frac{\theta}{2} [u]_{\times} + \left( 1 - \frac{\text{sinc}(\theta)}{\text{sinc}^2(\frac{\theta}{2})} \right) [u]_{\times}^2 \quad (25)$$

where

$$\text{sinc}(\theta) = \frac{\sin \theta}{\theta} \quad (26)$$

and

$$B^{-1}(\mathbf{y}_3) = I + \frac{\theta}{2} \text{sinc}^2\left(\frac{\theta}{2}\right) [u]_{\times} + (1 - \text{sinc}(\theta)) [u]_{\times}^2 \quad (27)$$

### Simulation

Use the different values for the end effector pose at the desired position.

cases	position	orientation	cases	position	orientation
1	0,0,-1.5	0,0,30	4	0,0,-3	0,0,0
2	0,0,-1.5	0,0,110	5	0.6,0.9,-4	0,0,0
3	0,0,-1.5	0,0,180	6	0.6,0.9,-2	-20,-20,130

Different case studies

For each case study, enter the new values for the desired end effector pose. Transfer the novel values with a click on **load data to model**. For the three control laws, run the simulation.

Analyse and compare the results (using the m-file **view3dpose**). In the figure 10000, think to change the color of the trajectory between each simulation.

Compare the three parametrizations for the orientation.

## 6 $2D \frac{1}{2}$ visual servoing

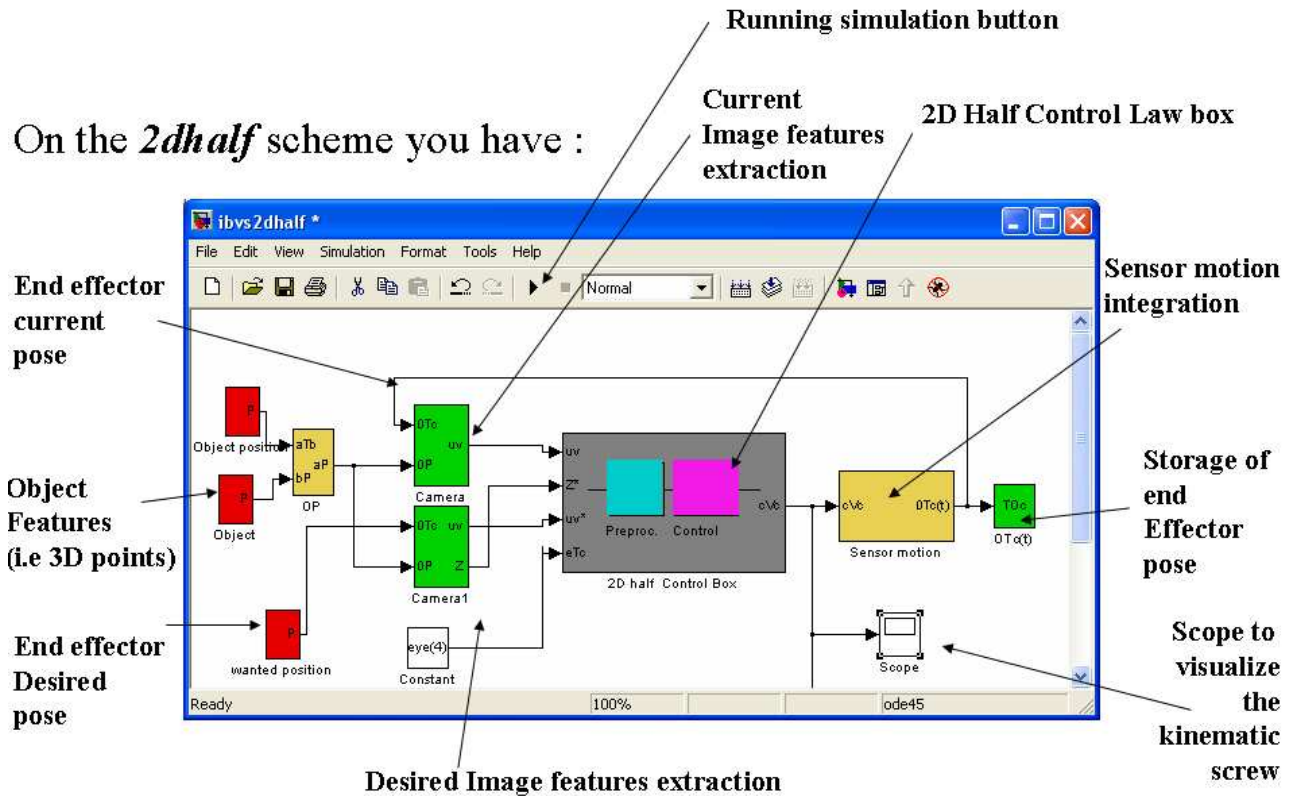


FIGURE 4 – HBVS :  $2D \frac{1}{2}$

Load the model **ibvs2dhalf.mdl** (figure 4) using the button **load model file** of the VS Browser (in the directory **vstoolbox/Models**).

### Simulation

Use the different values for the end effector pose at the desired position.

cases	position	orientation	cases	position	orientation
1	0,0,-1.5	0,0,30	4	0,0,-3	0,0,0
2	0,0,-1.5	0,0,110	5	0.6,0.9,-4	0,0,0
3	0,0,-1.5	0,0,180	6	0.6,0.9,-2	-20,-20,130

Different case studies

For each case study, enter the new values for the desired end effector pose. Transfer the novel values with a click on **load data to model**. For the three control laws, run the simulation.

Analyse and compare the results (using the m-file **view2dhalf**). In the figure 10000, think to change the color of the trajectory between each simulation.

Quit matlab and close your account at the end of the lab.

## Annexe : VSB Toolbox

After starting matlab, select the working directory **VservoingToolbox2.0**. In the menu **File Menu**, select the item **set Path**. Append the path to the **Visual Servoing Toolbox** using the item **Add with subfolders**. Save with the option **Save**, and then quit using **Close**. In the matlab workspace, select the **VservoingToolbox2.0/vsbrowser** directory. Execute the **VSBrowser** when running **initvsb**. The graphic user interface with its different elements are presented by figure 5.

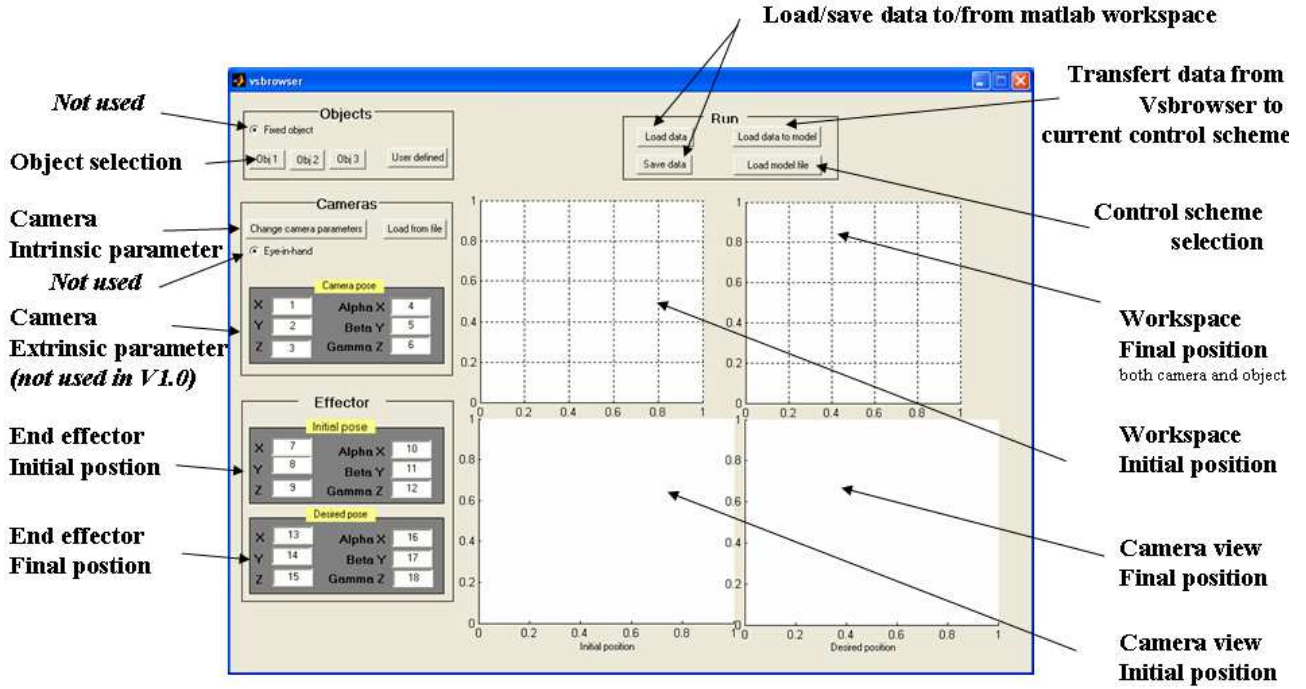


FIGURE 5 – Vsbrowser environment

When using the VSBrowser, follow the different steps :

- Load a visual servoing scheme by using button **Load model file** (absolutely necessary). The different available models are stored in the directory **VservoingToolbox2.0/vstoolbox/Models**. The figure 6 represents the result after loading **pbvs.mdl**.
- Select an objet (if not, the default object is selected). The figure 7 illustrates the result of the selection of default object and model  $n^o 2$ .
- Configure and validate the initial and final positions of the end effector (if not, default positions are proposed).
- Load the intrinsic parameters of the camera (if not, default parameters values are proposed).
- Transfer all the values from the GUI to the matlab/simulink workspace using the button **Load data to model**.
- Select the simulink block. Some blocks are parametrized. To access to the parametrization, you have to double click on the corresponding block.
- Run the simulation.
- to visualize the results, in the workspace of matlab use **view3Dpose** or **view2dhalf** for the 3D pose visual servoing or 2D1/2 visual servoing respectively.



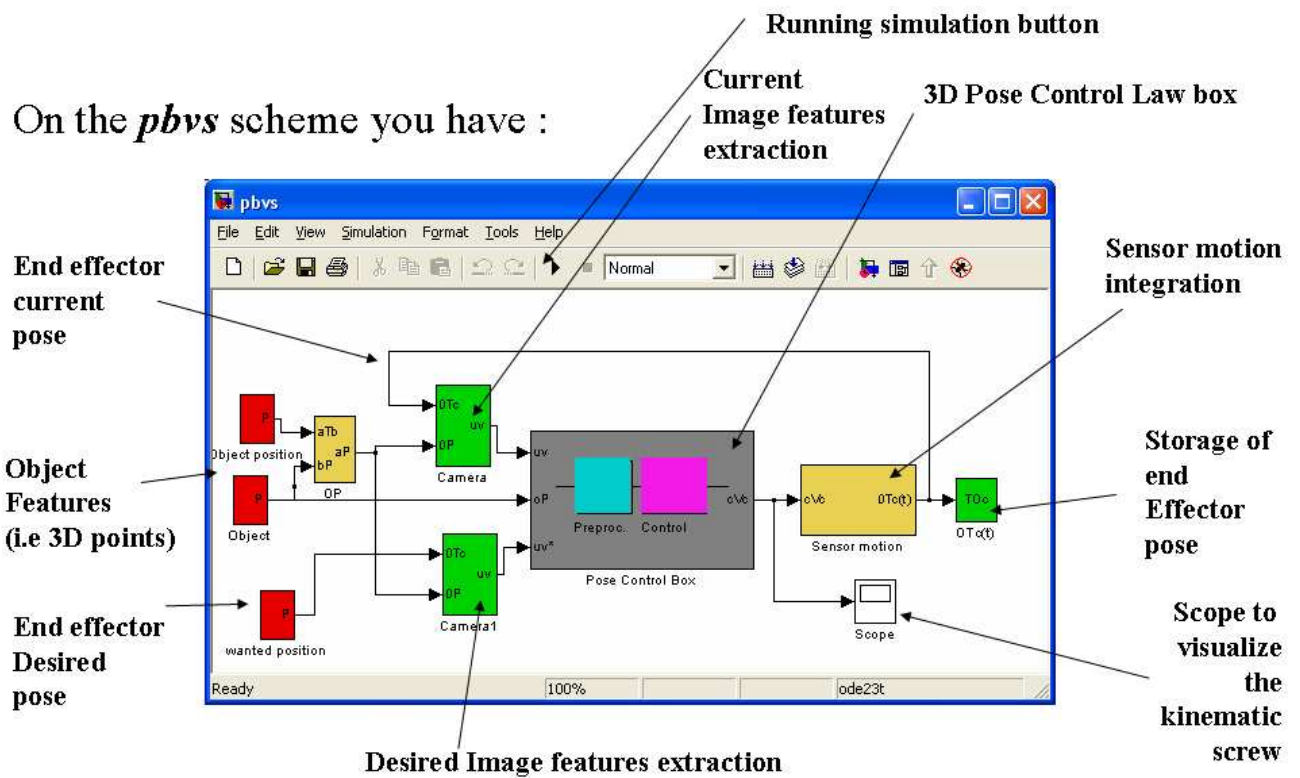


FIGURE 6 – PBVS : 3D pose

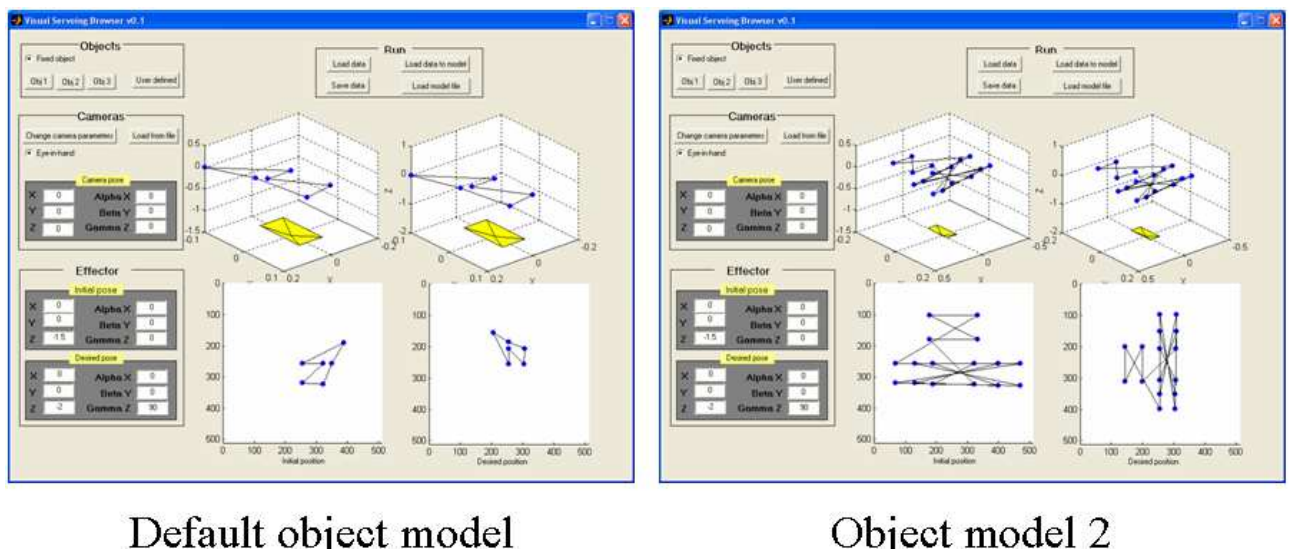


FIGURE 7 – Object choice

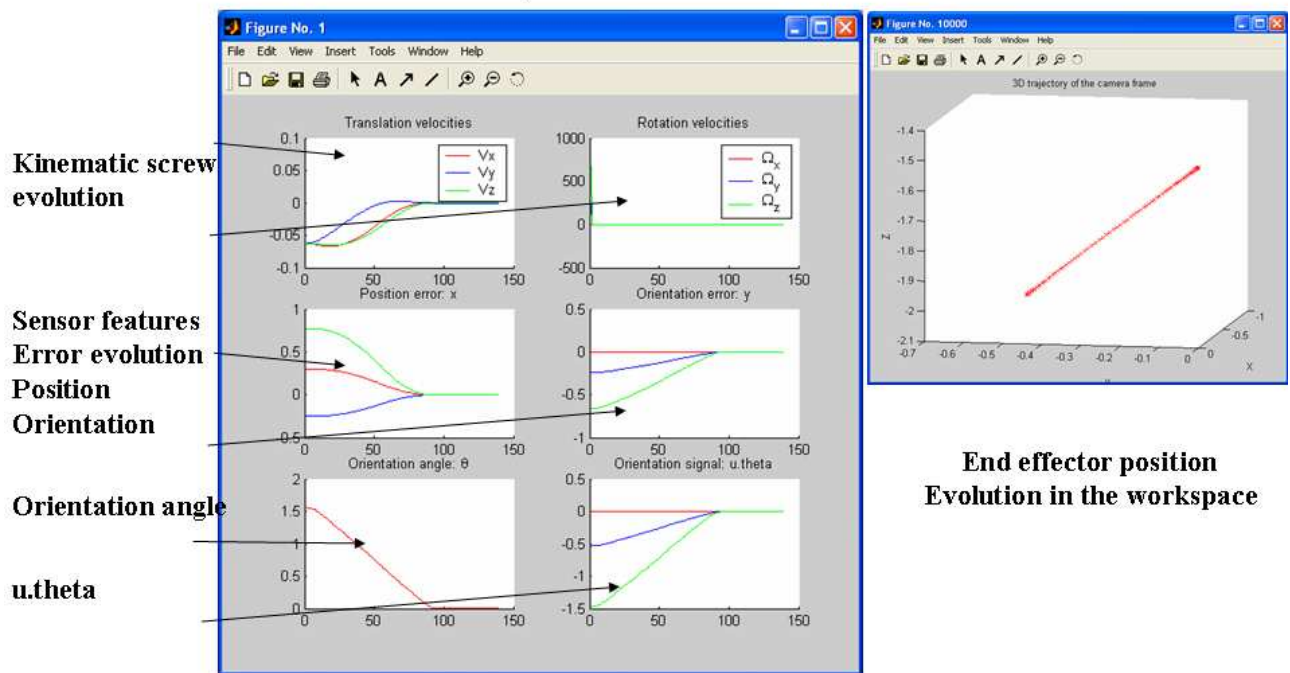
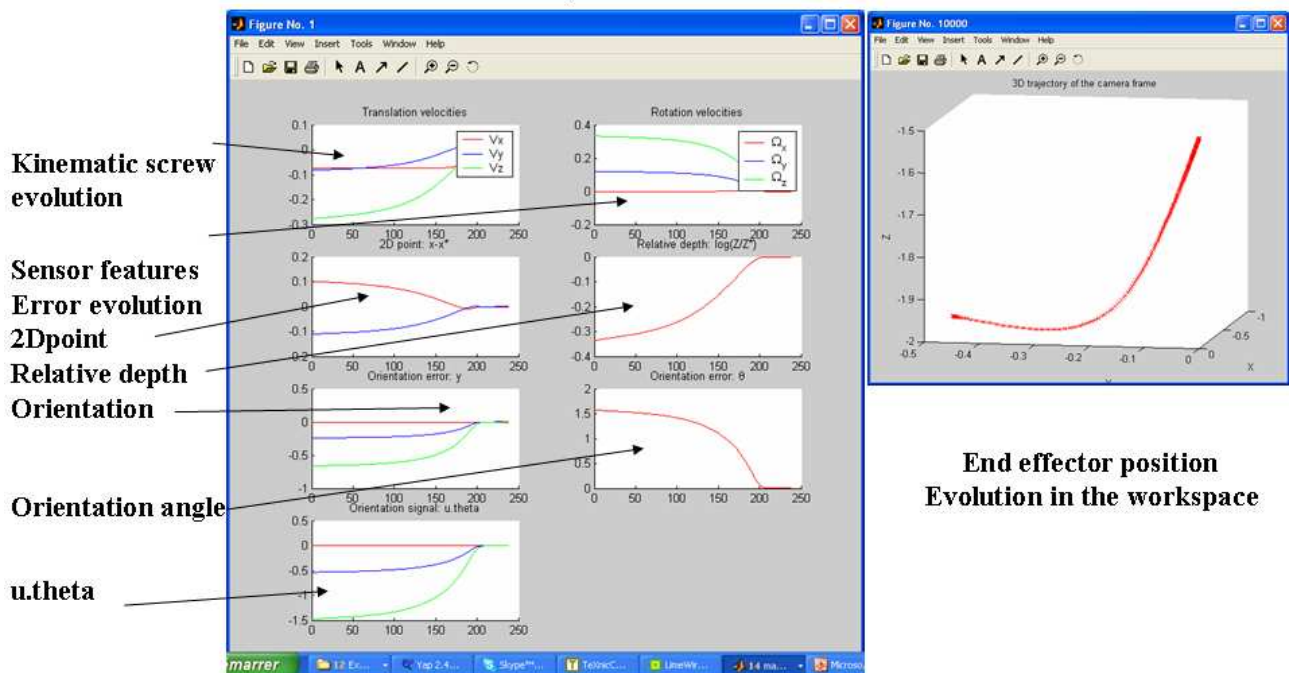
Results visualization with *view3dpose*

FIGURE 8 – Simulation results visualization : 3D pose

Results visualization with *view2dhalf*FIGURE 9 – Simulation results visualization :  $2D \frac{1}{2}$

Blaise Pascal University  
Aubiere, France

Visual servoing labs

**Lab N°3**

**Sensor based control : Application of 2D Image Based Visual servoing to the control of a PTZ camera**

The aim of the lab is to use the visual servoing techniques in the case of the control of a PTZ (Pan Tilt Zoom) camera.

First, we will concentrate on the use of sensor based control approach and on the modelling of 2D point and 2D segment features.

Second, we will evaluate different control laws (exact linearization, tangent linearization) when using *2D* points features (4 or 3 components), and *2D* segment features (4 or 3 components) and compare them (advantages and disadvantages).

Philippe Martinet  
Professor at IFMA  
Researcher at LASMEA, Blaise Pascal University  
Clermont-Ferrand, France

# 1 Theoretical modeling

## 1.1 Sensor based control

Considering one robot (for instance, one manipulator robot) where a sensor apparatus is embedded on the end effector, then the sensor information vector  $\underline{s}$  is a function of the current position of the sensor apparatus  $\underline{r}$  and of time  $t$  (the environment can change over the time) :

$$\underline{s} = \underline{s}(\underline{r}, t) \quad (1)$$

By derivating equation 1, we obtain :

$$\dot{\underline{s}} = \frac{\delta \underline{s}}{\delta \underline{r}} \cdot \frac{d\underline{r}}{dt} + \frac{\delta \underline{s}}{\delta t} = L_{\underline{s}}^T \cdot T + \frac{\delta \underline{s}}{\delta t} \quad (2)$$

where

- $T$  represents the control command (kinematic screw) applied to the sensor apparatus,
- $L_{\underline{s}}^T$  represents the interaction matrix (related to the content of  $\underline{s}$ ),
- $\frac{\delta \underline{s}}{\delta t}$  is the contribution of the environment motion (i.e. target motion)

The sensor based control uses the notion of *TASK FUNCTION*  $\underline{e}$  (or error function) defined by :

$$\underline{e} = C \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*) \quad (3)$$

where

- $C$  is a combination matrix which permits to take into account more sensor information than necessary to perform the positioning task,
- $\underline{s}(\underline{r}, t)$  the current sensor information vector,
- $\underline{s}^*$  the desired sensor information vector at equilibrium (when the positioning task is performed).

The main aim is to regulate to zero this task function. So, we impose an exponential decrease, that means :

$$\dot{\underline{e}} = -\lambda \cdot \underline{e} \quad (4)$$

and then we deduce the following control law :

$$U = T = -\lambda \cdot \underline{e} = -\lambda \cdot L_{\underline{s}}^{T+} \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*) \quad (5)$$

This control law represents in fact, a first order control law. The choice of the interaction matrix, can be done :

- at each iteration (exact linearization). In this case, its value is evaluated at each iteration taking into account the novel values of the sensor information. For this case, we will use the interaction matrix  $L_{\underline{s}}^T = \widehat{L}_{\underline{s}}^{T+}$  and the control law will become  $T = -\lambda \cdot \widehat{L}_{\underline{s}}^{T+} \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*)$ ,

$$\underline{s}^* \approx \underline{s} + \widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t \quad (6)$$

$$\Delta \underline{s} \approx -\widehat{L}_{\underline{s}}^T \mathbf{T} \Delta t \quad (7)$$

$$\mathbf{T} = -\lambda (\widehat{L}_{\underline{s}} \widehat{L}_{\underline{s}}^T)^{-1} \widehat{L}_{\underline{s}} \Delta \underline{s} = -\lambda \widehat{L}_{\underline{s}}^{T+} \Delta \underline{s} \quad (8)$$

- at equilibrium (tangent linearization). In this case, its value is evaluated with the value of  $\underline{s}^*$  corresponding to the desired position. For this case, we will use the interaction matrix  $L_{\underline{s}}^T = \widehat{L_{\underline{s}=\underline{s}^*}^T}$  and the control law will become  $T = -\lambda \cdot \widehat{L_{\underline{s}=\underline{s}^*}^{T+}} \cdot (\underline{s}(\underline{r}, t) - \underline{s}^*)$

$$\mathbf{s} \approx \mathbf{s}^* + \widehat{L_{\underline{s}=\underline{s}^*}^T} \mathbf{T} \Delta t \quad (9)$$

$$\Delta \mathbf{s} \approx -\widehat{L_{\underline{s}=\underline{s}^*}^T} \mathbf{T} \Delta t \quad (10)$$

$$\mathbf{T} = -\lambda (\widehat{L_{\underline{s}=\underline{s}^*}^T} \widehat{L_{\underline{s}=\underline{s}^*}^T})^{-1} \widehat{L_{\underline{s}=\underline{s}^*}^T} \Delta \mathbf{s} = -\lambda \widehat{L_{\underline{s}=\underline{s}^*}^{T+}} \Delta \mathbf{s} \quad (11)$$

## 1.2 Interaction matrix : 2D point feature

Considering the following definitions :

- $(x, y, z)^T$  : coordinates of a 3D point  $p$
- $(X, Y)^T$  : coordinates of the corresponding perspective projected point  $P$  in perspective plane
- $(u, v)^T$  : coordinates of the corresponding projected point  $Pim$  in image plane
- $f_u$ , and  $f_v$  focal length in  $u$  and  $v$  direction
- $u_0$  and  $v_0$  : coordinates of the optical center in image plane.

So, we have :

$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (12)$$

$$P = \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \quad (13)$$

$$Pim = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_u & 0 \\ 0 & f_v \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \quad (14)$$

where  $(f_u, f_v, u_0, v_0)$  represent the intrinsic parameters of the camera.

The expression of the different interaction matrices is the following :

$$L_p = \begin{bmatrix} -I_3 & \tilde{p} \end{bmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 & -z & y \\ 0 & -1 & 0 & z & 0 & -x \\ 0 & 0 & -1 & -y & x & 0 \end{pmatrix} \quad (15)$$

$$L_P = \begin{pmatrix} -\frac{1}{z} & 0 & \frac{X}{z} & X \cdot Y & -1 - X^2 & Y \\ 0 & -\frac{1}{z} & \frac{Y}{z} & 1 + Y^2 & -X \cdot Y & -X \end{pmatrix} \quad (16)$$

$$L_{Pim} = \begin{pmatrix} -\frac{f_u}{z} & 0 & \frac{u}{z} & \frac{u \cdot v}{f_v} & -f_u - \frac{u^2}{f_u} & \frac{f_u}{f_v} \cdot v \\ 0 & -\frac{f_v}{z} & \frac{v}{z} & f_v + \frac{v^2}{f_v} & -\frac{u \cdot v}{f_u} & -\frac{f_v}{f_u} \cdot u \end{pmatrix} \quad (17)$$

### 1.3 Interaction matrix : 2D segment feature

One segment in image plane can be defined by two kind of parametrizations :

- two 2D image points  $(u_1, v_1, u_2, v_2)^T$
- $(u_c, v_c, l, \theta)^T$  where  $(u_c, v_c)$  are the image coordinates of the segment center,  $l$  is its length, and  $\theta$  is its orientation

It exists some relations between both modeling:

$$\begin{aligned}
 u_c &= \frac{u_1 + u_2}{2} & u_1 &= u_c + \frac{l}{2} \cdot \cos \theta \\
 v_c &= \frac{v_1 + v_2}{2} & u_2 &= u_c - \frac{l}{2} \cdot \cos \theta \\
 l &= \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2} & v_1 &= v_c + \frac{l}{2} \cdot \sin \theta \\
 \theta &= \arctan(v_1 - v_2, u_1 - u_2) & v_2 &= v_c - \frac{l}{2} \cdot \sin \theta
 \end{aligned} \tag{18}$$

We can deduce the corresponding interaction matrices :

$$L_{Seg1} = \begin{pmatrix} -\frac{f_u}{z_1} & 0 & \frac{u_1}{z_1} & \frac{u_1 \cdot v_1}{f_v} & -f_u - \frac{u_1^2}{f_u} & \frac{f_u}{f_v} \cdot v_1 \\ 0 & -\frac{f_v}{z_1} & \frac{v_1}{z_1} & f_v + \frac{v_1^2}{f_v} & -\frac{u_1 \cdot v_1}{f_u} & -\frac{f_v}{f_u} \cdot u_1 \\ -\frac{f_u}{z_2} & 0 & \frac{u_2}{z_2} & \frac{u_2 \cdot v_2}{f_v} & -f_u - \frac{u_2^2}{f_u} & \frac{f_u}{f_v} \cdot v_2 \\ 0 & -\frac{f_v}{z_2} & \frac{v_2}{z_2} & f_v + \frac{v_2^2}{f_v} & -\frac{u_2 \cdot v_2}{f_u} & -\frac{f_v}{f_u} \cdot u_2 \end{pmatrix} \tag{19}$$

$$L_{Seg2} = \begin{bmatrix} L_{Seg2_1} & L_{Seg2_2} \end{bmatrix} \tag{20}$$

with

$$\begin{aligned}
 L_{Seg2_1} &= \begin{pmatrix} -\lambda_2 \cdot f_u & 0 & \lambda_2 \cdot u_c - \lambda_1 \cdot \frac{l}{4} \cdot \cos \theta \\ 0 & -\lambda_2 \cdot f_v & \lambda_2 \cdot v_c - \lambda_1 \cdot \frac{l}{4} \cdot \sin \theta \\ \lambda_1 \cdot f_u \cdot \cos \theta & \lambda_1 \cdot f_v \cdot \sin \theta & \lambda_2 \cdot l - \lambda_1 \cdot (u_c \cdot \cos \theta + v_c \cdot \sin \theta) \\ -\frac{\lambda_1}{l} \cdot f_u \cdot \sin \theta & \frac{\lambda_1}{l} \cdot f_v \cdot \cos \theta & \frac{\lambda_1}{l} \cdot (u_c \cdot \sin \theta - v_c \cdot \cos \theta) \end{pmatrix} \\
 L_{Seg2_2} &= \begin{pmatrix} \frac{u_c \cdot v_c}{f_v} + \frac{l^2}{4 \cdot f_v} \cdot \cos \theta \cdot \sin \theta & -\left(f_u + \frac{u_c^2}{f_u} + \frac{l^2 \cdot \cos^2 \theta}{4 \cdot f_u}\right) & v_c \cdot \frac{f_u}{f_v} \\ f_v + \frac{v_c^2}{f_v} + \frac{l^2 \cdot \sin^2 \theta}{4 \cdot f_v} & -\frac{u_c \cdot v_c}{f_u} - \frac{l^2}{4 \cdot f_u} \cdot \cos \theta \cdot \sin \theta & -u_c \cdot \frac{f_v}{f_u} \\ \frac{l \cdot (u_c \cdot \cos \theta \cdot \sin \theta + v_c \cdot (1 + \sin^2 \theta))}{f_v} & -\frac{l \cdot (v_c \cdot \cos \theta \cdot \sin \theta + u_c \cdot (1 + \cos^2 \theta))}{f_u} & l \cdot \cos \theta \cdot \sin \theta \left(\frac{f_u}{f_v} - \frac{f_v}{f_u}\right) \\ \frac{v_c \cdot \cos \theta \cdot \sin \theta - u_c \cdot \sin^2 \theta}{f_v} & \frac{u_c \cdot \cos \theta \cdot \sin \theta - v_c \cdot \cos^2 \theta}{f_u} & -\frac{f_u}{f_v} \cdot \sin^2 \theta - \frac{f_v}{f_u} \cdot \cos^2 \theta \end{pmatrix}
 \end{aligned}$$

where  $\lambda_1 = \frac{z_1 - z_2}{z_1 \cdot z_2}$  and  $\lambda_2 = \frac{z_1 + z_2}{2 \cdot z_1 \cdot z_2}$ .

## 2 Control of a PTZ camera with $2D$ points

We want to control a PTZ camera with  $2D$  points features. Only the control command :

- $\Omega_x$  rotational velocity around x axis
- $\Omega_y$  rotational velocity around y axis
- $\dot{f}$  Zoom velocity control (equivalent to focal length control using a linear relation)

are available on a PTZ camera.

**Theoretical part** Considering square pixels (i.e.  $f_u = f_v = f$ ), and the optical model of equation 14, deduce the theoretical expression of the interaction  $L1$  for two  $2D$  points, reduced to the control vector  $(\Omega_x, \Omega_y, \dot{f})^T$ .

**Practical part** Download and extract the directory **lab3.zip** from the web page "<http://www.skku.edu/~martinet/RobotVisionclass.html>" (cf material part). Enter the directory **lab3**.

Select the file **simuptz2pts.mdl** and open it with matlab/simulink. After starting matlab, select the working directory **lab3**. In the menu **File Menu**, select the item **set Path**. Append the path to the **lab3** using the item **Add with subfolders** (take care to remove the path to the visual servoing toolbox 2.0). Save with the option **Save**, and then quit using **Close**.

You are under simulink in a simulation environment of a system delivering  $2D$  points features  $2D$  computed from a  $3D$  object.

Make a general scheme of the simulator en precize the role and function of each block.

Under the directory **lab33**.

- Select the file **calculL2pts.m** and open it with the matlab editor.
- Modify its content with the corresponding interaction matrix.
- Select the matrix computed at equilibrium under simulink
- Run simulation. Analyze the results.
- Select the matrix computed at each iteration under simulink
- Run simulation. Analyze the results. Compare with the previous results.

In order to control 3 degrees of freedom, it is necessary to use one vector  $\underline{s}$  of dimension 3. For that, you can use 3 over 4 components on the previous visual informations.

- Under simulink, open the file **simulptz2ptsu.mdl**.
- Select the matrix computed at equilibrium under simulink
- Run simulation. Analyse the results.
- Select the matrix computed at each iteration under simulink

- Run simulation. Analyze the results. Compare with the previous results.
- Under simulink, open the file **simulptz2ptsv.mdl**.
- Select the matrix computed at equilibrium under simulink
- Run simulation. Analyze the results.
- Select the matrix computed at each iteration under simulink
- Run simulation. Analyze the results. Compare with the previous results.

Conclude on the proposed approaches.

### 3 Control of a PTZ camera with 2D segment

We want to control a PTZ camera with 2D segment feature. Only the control command :

- $\Omega_x$  rotational velocity around x axis
- $\Omega_y$  rotational velocity around y axis
- $\dot{f}$  Zoom velocity control (equivalent to focal length control using a linear relation)

are available on a PTZ camera.

**Theoretical part** Considering square pixels (i.e.  $f_u = f_v = f$ ), and the optical model of equation 14, deduce the theoretical expression of the interaction  $L2$  for 2D segment, reduced to the control vector  $(\Omega_x, \Omega_y, \dot{f})^T$ .

**Practical part** Under the directory **lab3**.

Select the file **simuptzsegn.mdl** and open it with matlab/simulink.

Then, you are under simulink in the simulation environment of one system using a 2D segment sensor computed from 3D object.

Under the directory **lab3**.

- Select the file **calculsegn.m** and open it with the matlab editor.
- Modify its content writing the corresponding interaction matrix.
- Select the matrix computed at equilibrium under simulink
- Run the simulation. Analyze the results.
- Select the matrix computed at each iteration under simulink
- Run the simulation. Analyze the results. Compare with the previous results.

In order to control 3 degrees of freedom, it is necessary to use one vector  $\underline{s}$  of dimension 3. For that, you can use 3 over 4 components on the previous visual informations.

- Under simulink, open the file **simulptzseg.mdl**.



- Select the matrix computed at equilibrium under simulink
- Run the simulation. Analyze the results.
- Select the matrix computed at each iteration under simulink
- Run the simulation. Analyze the results. Compare with the previous results.

Conclude on these two kind of modeling in regard with the task to be done.