University of Burgundy

Masters in Computer Vision

Visual Servoing

IBVS & PBVS

Free Camera 3D Pose Control

by

Tsagkatakis Ioannis

Under the supervision of

Dr. Erol Ozgur

January 14, 2019

Contents

1 Introduction **2**

2 The Pose Controll problem **2**

2.1 The Control law 3

2.1.1 Stability analysis 4

3 Image Based Visual Servoing **5**

3.1 Algorithm 5

4 Position Based Visual Servoing **5**

4.1 Algorithm 6

5 Matlab Experiments **6**

5.1 Image Based Visual Servoing 6

5.1.1 A simple example with 4 points 6

5.1.2 Loosing Features: 3 points 7

5.1.3 Loosing Features: only 3 points 7

5.1.4 Loosing Features: 2 points 7

5.1.5 The Coplanar Problem 8

5.1.6 The Local Minimuma Problem 9

5.1.7 The Diverge Problem 10

5.2 Position Based Visual Servoing 15

5.2.1 Matlab simulation 15

6 Conclusions **16**

6.1 MatlabCode 16

7 Bibliography **16**

1 Introduction

Visual servo control refers to the use of computer vision data to control the motion of a robot. Visual servo control relies on techniques from image processing, computer vision, and control theory.

2 The Pose Controll problem

As the location of the camera is always known we have a Position Based Visual Servoing (PBVS) problem. The control system can be either a real system like a robot or a virtual system like in Viruar Reality applications. The general structure of a PBVS is shown in Figure 2. A PBVS system operates in Cartesian space and allows the direct and natural specification of the desired relative trajectories in the Cartesian space. The parameters extracted from the image $\mathbf{s} = \mathbf{s}(\mathbf{p}_t)$, are used with the models of camera and object geometry to estimate the relative pose vector $\widehat{\mathbf{W}}$ of the object with respect to the end-effector. The estimated pose is compared with the desired relative pose $\mathbf{W}_d$ to calculate the relative pose error $\widetilde{\mathbf{W}}$. The coordinate frames involved in the process is given in Figure 1 on page 3. A Cartesian control law reduces the relative pose error, and the Cartesian control command transformed to the joint-level commands for the joint-servo loops by appropriate kinematic transformations. By separating the pose-estimation problem from the control-design problem, the control designer can take advantage of well-established robot Cartesian control algorithms.

2.1 The Control law

The visual features parameters extracted from the image are a function of the camera poses as given by

$$\mathbf{s} = \mathbf{s}(\mathbf{p}_t) \tag{1}$$

By taking the derivative of the above relation we obtain

$$\dot{\mathbf{S}} = \mathbf{L}_s \mathbf{V}\,, \tag{2}$$

where $\mathbf{L}_s$ the so called *Interaction matrix* or *feature Jacobian* and $\mathbf{V}$ the camera (Kinematic screw) denoted by $\mathbf{V} = (\vec{v}, \vec{\omega})$. Thus thw $\mathbf{V}$ contains 3

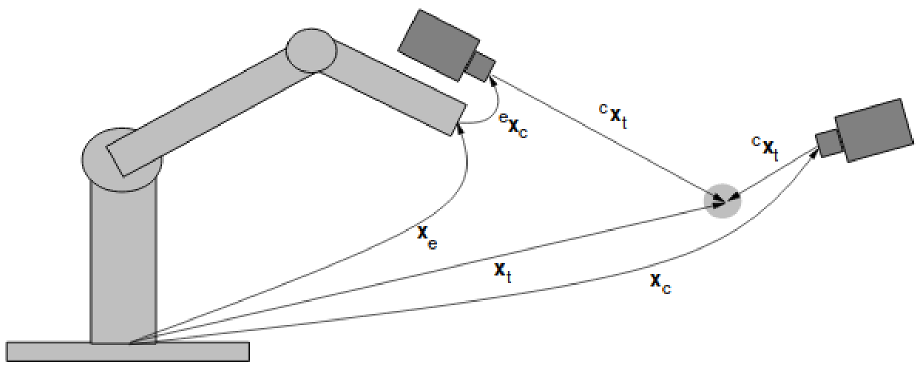


Figure 1: The coordinate frames : word, end-effector, camera and target.

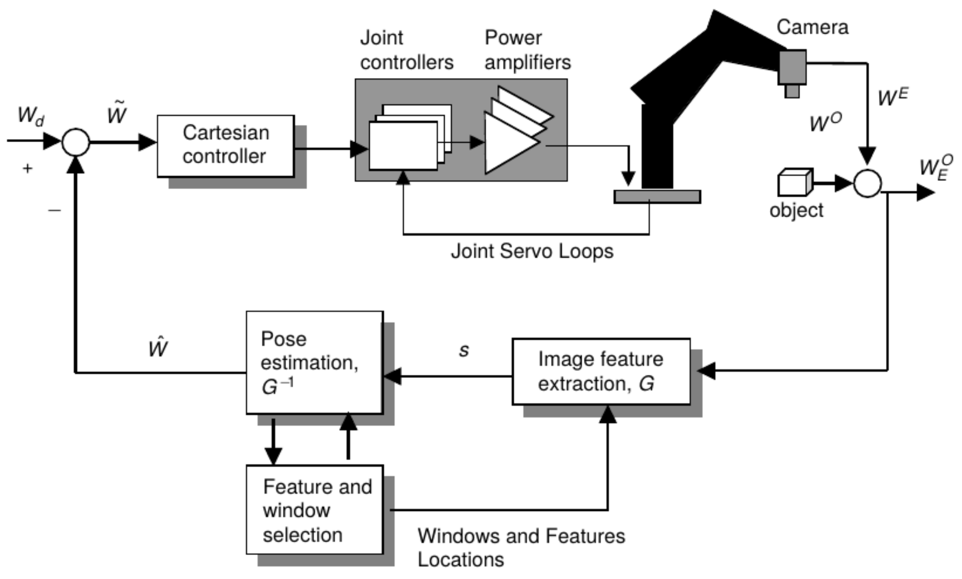


Figure 2: Structure of Position based Visual Servoing (PBVS).

translatiosn and 3 rotations. The goal of the control law is to minimize the error given by

$$\mathbf{e}(t) = \mathbf{S}(\mathbf{p}_t) - \mathbf{S}^*, \tag{3}$$

where $\mathbf{S}^*$ the desired value of the feature. Using equations (1) and (3) then the time variation of the error is

$$\dot{\mathbf{e}} = \mathbf{L}_s \mathbf{V} \tag{4}$$

A good control law is to have an exponential decoupled decrease of the error $\dot{\mathbf{e}} = -\lambda\,\mathbf{e}$. and we obtain:

$$\mathbf{V} = -\lambda \mathbf{L}_s^+ \mathbf{e} \tag{5}$$

where $\mathbf{L}_s^+ \in \Re^{6\times k}$ is chosen as the Moore-Penrose pseudoinverse of $\mathbf{L} = \left(\mathbf{L}^T\mathbf{L}\right)^{-1}\mathbf{L}^T$, when $\mathbf{L}$ is of full rank 6. In real visual servo systems, it is impossible to know perfectly in practice either $\mathbf{L}$ or $\mathbf{L}^+$, so an approximation or an estimation $\widehat{\mathbf{L}_s^+}$ must be realized. This corresponds to

$$\mathbf{V} = -\lambda \widehat{\mathbf{L}_s^+}\left(\mathbf{S} - \mathbf{S}^*\right) \tag{6}$$

2.1.1 Stability analysis

$$\begin{cases} \widehat{\mathbf{L}_s}\widehat{\mathbf{L}_s^+} = \mathbf{I} & \text{Ideal behavior} \\ \widehat{\mathbf{L}_s}\widehat{\mathbf{L}_s^+} > 0 & \text{The error } \mathbf{e} \text{ decreases} \\ \widehat{\mathbf{L}_s}\widehat{\mathbf{L}_s^+} < 0 & \text{The error } \mathbf{e} \text{ grows} \end{cases} \tag{7}$$

3 Image Based Visual Servoing

The interaction matrix is given by the equation

$$\begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+x^2 & -xy & -x \end{bmatrix} \tag{8}$$

which is a 2×6 matrix with the first 3 columns corresponds to a translation and the last 3 to a rotation.

3.1 Algorithm

Algorithm 1 Image Based Visual Servoing

```
while not at end of time do
    Build the feature vector $s_A$
    Calculate the interaction matrix using (8)
    Find the error $\mathbf{e}$
    Calculate $\widehat{\mathbf{L}_s^+}$
    Calculate the control law
        $$\xi_{A/R} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \\ \mathbf{O}_{3\times3} & \mathbf{R} \end{bmatrix}_{A/R} \xi_{A/A}\, ,\ \xi_{A/A} = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{9}$$
    Apply the control law and get new camera location
end
```

4 Position Based Visual Servoing

Let $s(\mathbf{X}) \in \Re^{6\times1}$ is a representation of the Cartesian pose $\mathbf{X} \in \Re^{4\times4}$ as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}\ ,\ \ s(\mathbf{X}) = \begin{bmatrix} \mathbf{t} \\ \mathbf{u}\,\theta \end{bmatrix} \tag{10}$$

where $\mathbf{u}\,\theta$ corresponds to the rotation $\mathbf{R}$. Here u is a unit rotation axis and θ is the rotation angle around this axis $\mathbf{u}$. The error can be calculated using the equation

$$\mathbf{e} = s\left(\mathbf{X}_{B/R}^{-1} \star \mathbf{X}_{A(t)/R}\right) = \mathbf{e} = s\left(\mathbf{X}_{A(t)/B}\right) \tag{11}$$

The control law is

$$\xi_{A(t)/B} = \begin{bmatrix} v \\ \omega \end{bmatrix}_{A(t)/B} = -\lambda \begin{bmatrix} \mathbf{R}^t\mathbf{t} \\ \theta\mathbf{u} \end{bmatrix}_{A(t)/B}\ ,\ \ \lambda > 0 \tag{12}$$

4.1 Algorithm

Algorithm 2 Position Based Visual Servoing

```
while not at end of time do
    Build the feature vector
    Find the error $\mathbf{e}$ using equation (11)
    Calculate $\widehat{\mathbf{L}_s^+}$
        $$\xi_{A(t)/R} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}_\times \mathbf{R}] \\ \mathbf{0}_{3\times 3} & \mathbf{R} \end{bmatrix}_{B/R} \xi_{A(t)/B} \quad (13)$$
    Apply the control law and get new camera location
        $$\mathbf{X}_{A(T+\Delta t)/R} = \begin{bmatrix} \Delta t\,[\omega]_\times & \Delta t\, v \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}_{A(\Delta t)/R} \mathbf{X}_{A(t)/R} \quad (14)$$
end
```

5 Matlab Experiments

5.1 Image Based Visual Servoing

We want to move a camera from its current location to a desired location relative to a pattern (see Figure 3 on page 7) 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:

5.1.1 A simple example with 4 points

In the first example we have 2 cameras, and the pattern is clearly visible from them. From the simulation results on Figure 4 on page 8, we observe that the camera follows a very good and straight path. The velocity is getting smaller and smaller as we aproaching the target. We notice that the interactiom matrix have some very high eigenvalues and some very small ones. So the DoF of the eigenvalues with big values is the only ones that drives the control of the system.

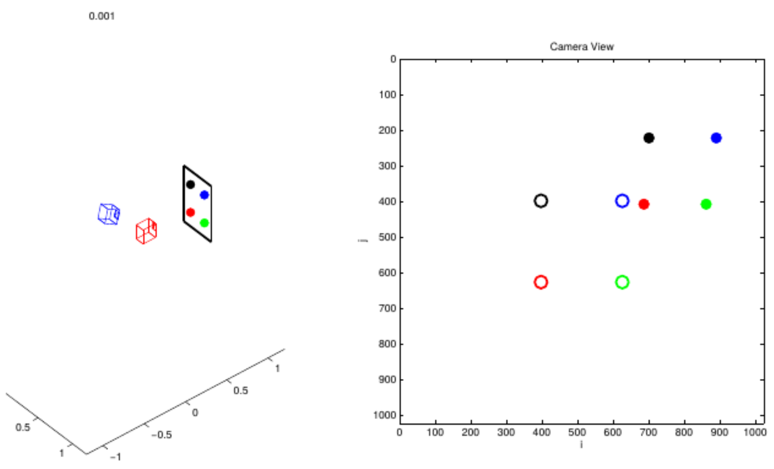


Figure 3: Image based control of the camera pose using point features. Blue camera is the current location, and the red camera is the desired location of the camera with respect a pattern with four points (left figure). Empty dots are the desired image points observed from the desired red camera location, and the full dots are the current image points observed from the current blue camera location (right figure). .

5.1.2 Loosing Features: 3 points

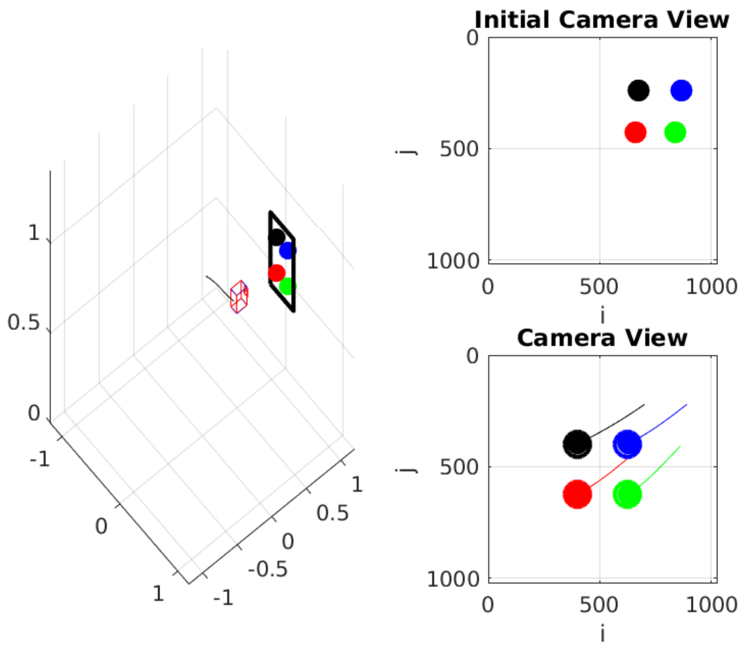
If we have only 3 points then we get 6 features witch is the minimum required. That not gurantes that we get a good solution in every case. Here the final camera location is almost correct. The movement have some instabilities, but that is expected due to the poorness interaction matrix. The simulation results is given at Figure 5 on page 9. The matlab code is on file `Demo2.m`.

5.1.3 Loosing Features: only 3 points

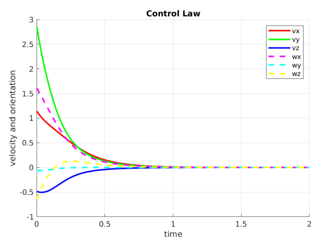
In this experiment we track only 3 points, even if all points are presented on the camera. The camera motion is far away from ideal, it even goes behind the scene, increasing the error, but it finally manage to get a good solution. The simulation results is given at Figure 6 on page 10. The matlab code is on file `Demo2_3var.m`.

5.1.4 Loosing Features: 2 points

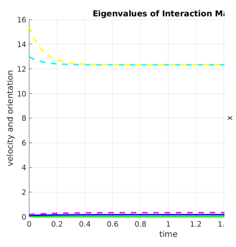
By using only 2 point we get better results. That was not expected, but that is because of the initial view have 2 lines that is directed connected with



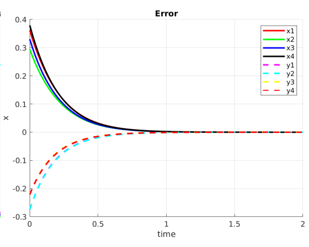
(a) The camera motion



(b) Control Law



(c) Eignevalues



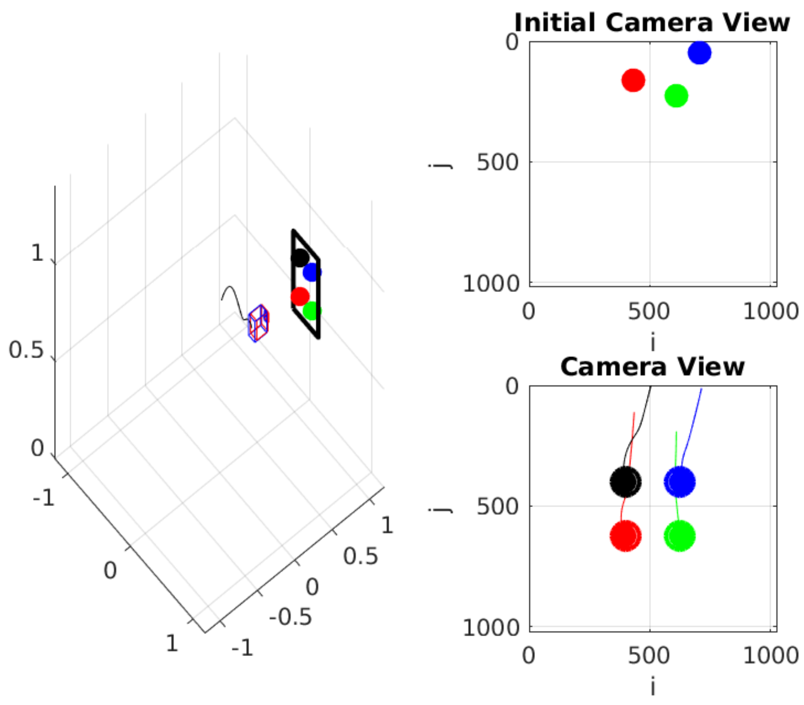
(d) Errors

Figure 4: Simple case : Observable pattern, the control law.

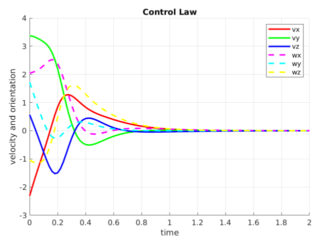
the target lines it was easy to get a very smooth solution. The simulation results is given at Figure 7 on page 11. The matlab code is on file `Demo3.m`.

5.1.5 The Coplanar Problem

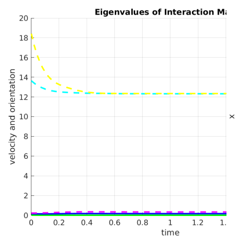
Here the human solution is just a rotation. But the solution to minimize the distances is to move the camera backwards as the error is decreasing. All the eignvalues are small in this case. The simulation results is given at Figure 8 on page 12. The matlab code is on file `Demo4.m`.



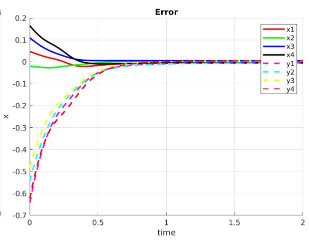
(a) The camera motion



(b) Control Law



(c) Eignevalues

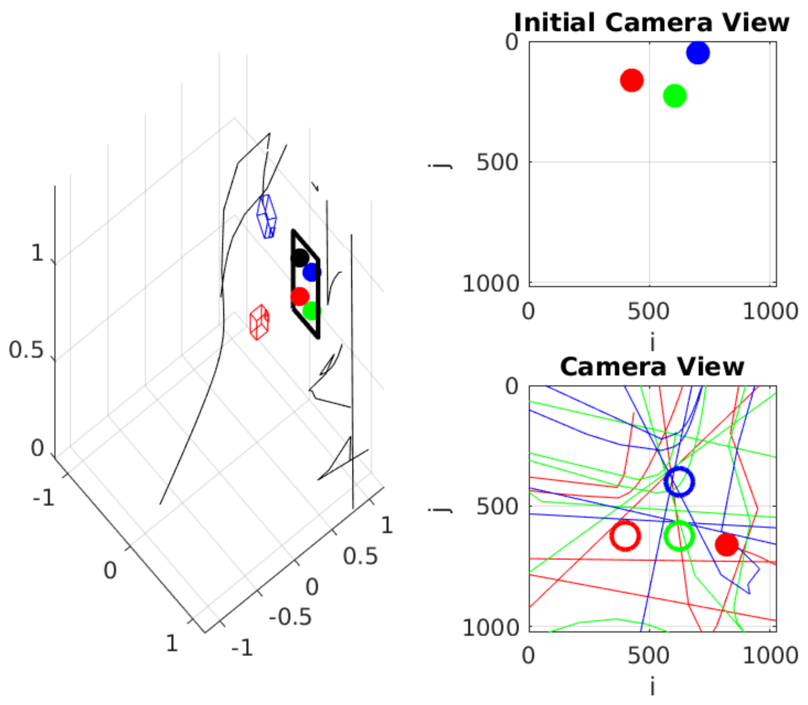


(d) Errors

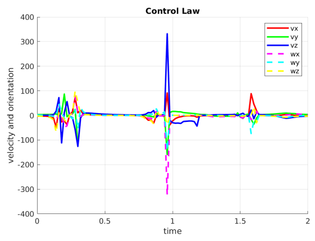
Figure 5: Using 3 features : Observable pattern, the control law.

5.1.6 The Local Minimuma Problem

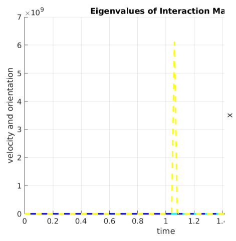
The matrix $\widehat{\mathbf{L}\mathbf{L}_s^+} \in \Re^{k \times k}$ is at most of rank 6. With 4 points we have $k = 8$ so it have a non-trivial null space. Thus configurations that corresponds to local minima exists. In the setup of the experiment, the simulation runs into a local minima and the control loop was unable to find a smooth path or a valid solution. The simulation results is given at Figure 9 on page 13. The motion is shown in Figure 10 on page 13. The matlab code is on file `Demo5.m`.



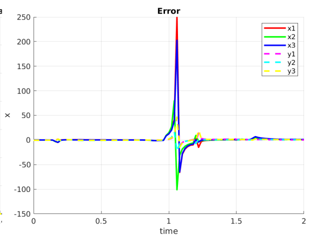
(a) The camera motion



(b) Control Law



(c) Eignevalues

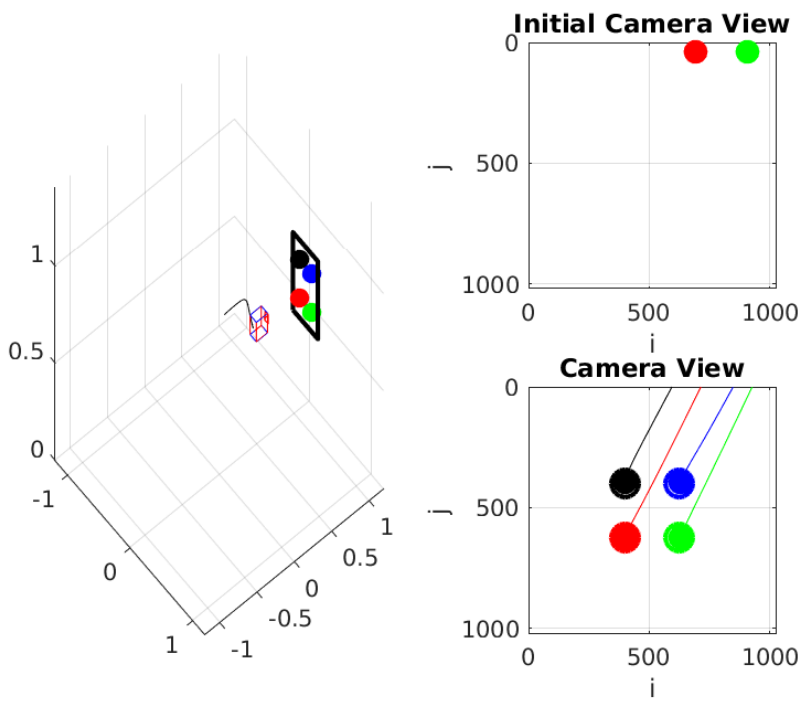


(d) Errors

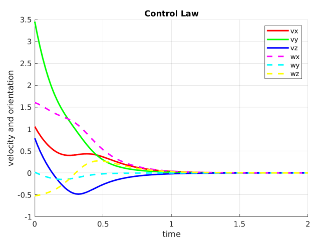
Figure 6: Using only 3 features : Observable pattern, the control law.

5.1.7 The Diverge Problem

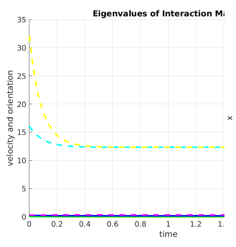
Here a case where the control algorithm is diverging is demonstrated. The simulation results is given at Figure 12 on page 14. The motion is shown in Figure 11 on page 14. The matlab code is on file `Demo6.m`.



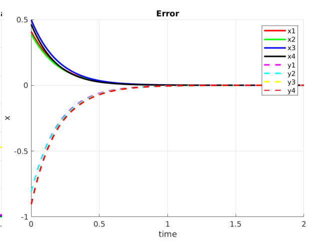
(a) The camera motion



(b) Control Law

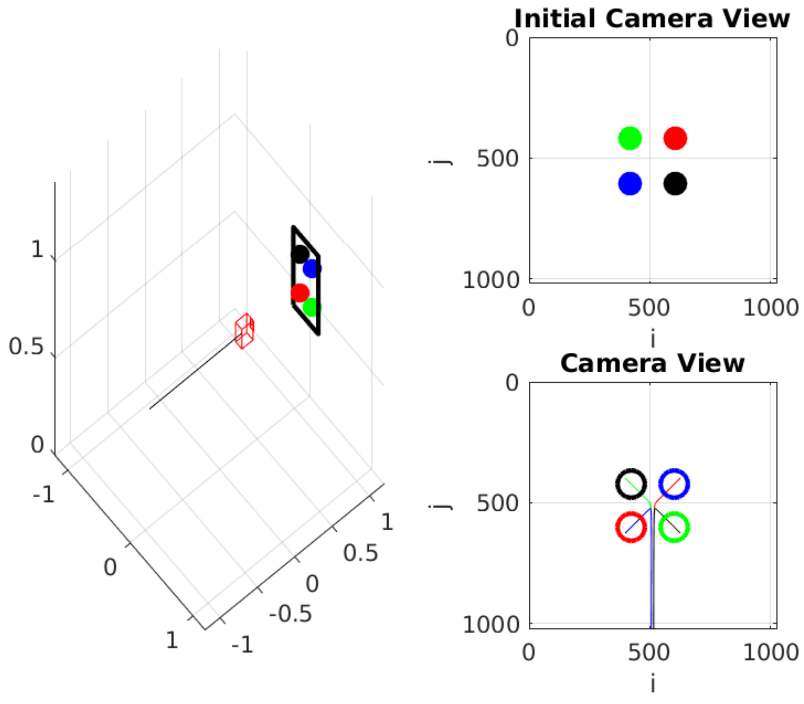


(c) Eignevalues

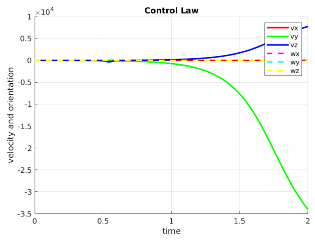


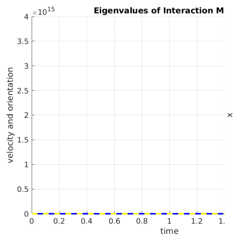
(d) Errors

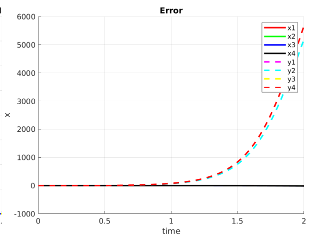
Figure 7: Using 2 features : Observable pattern, the control law.



(a) The camera motion

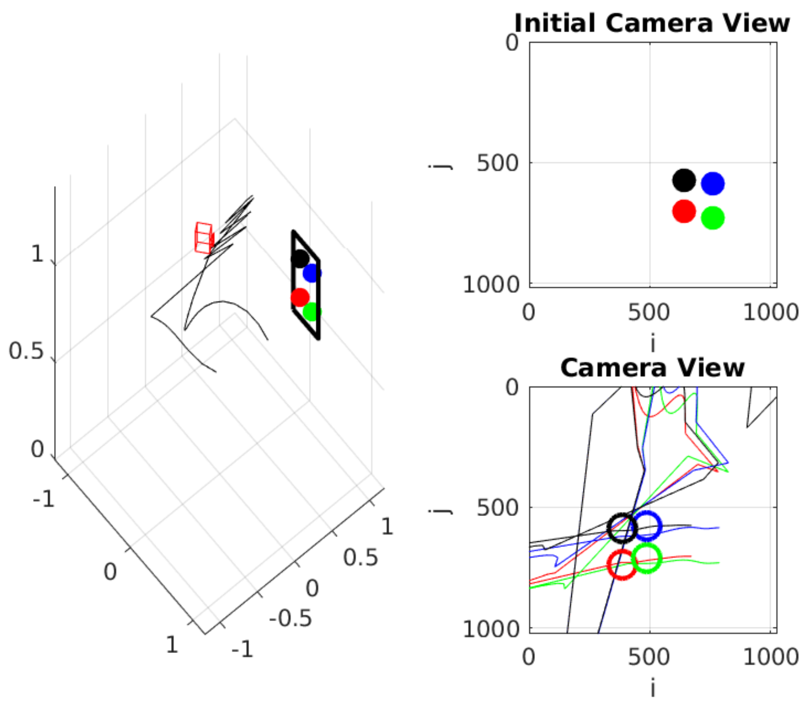




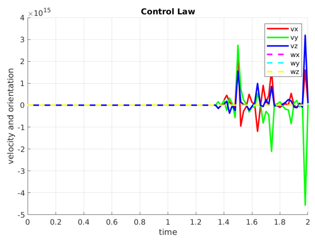


(b) Control Law (c) Eignevalues (d) Errors

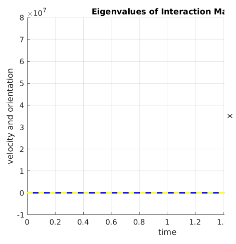
Figure 8: The coplanar problem.



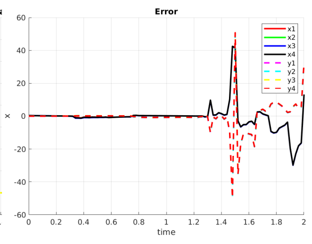
(a) The camera motion



(b) Control Law

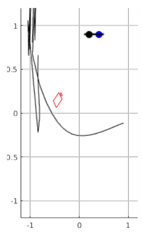


(c) Eignevalues

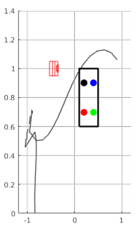


(d) Errors

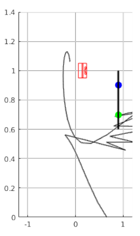
Figure 9: The local minima problem.



(a) XY view

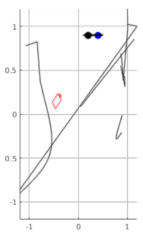


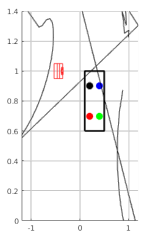
(b) XZ view



(c) YZ View

Figure 10: The local minima problem: Motion Views


(a) XY view


(b) XZ view

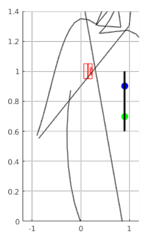
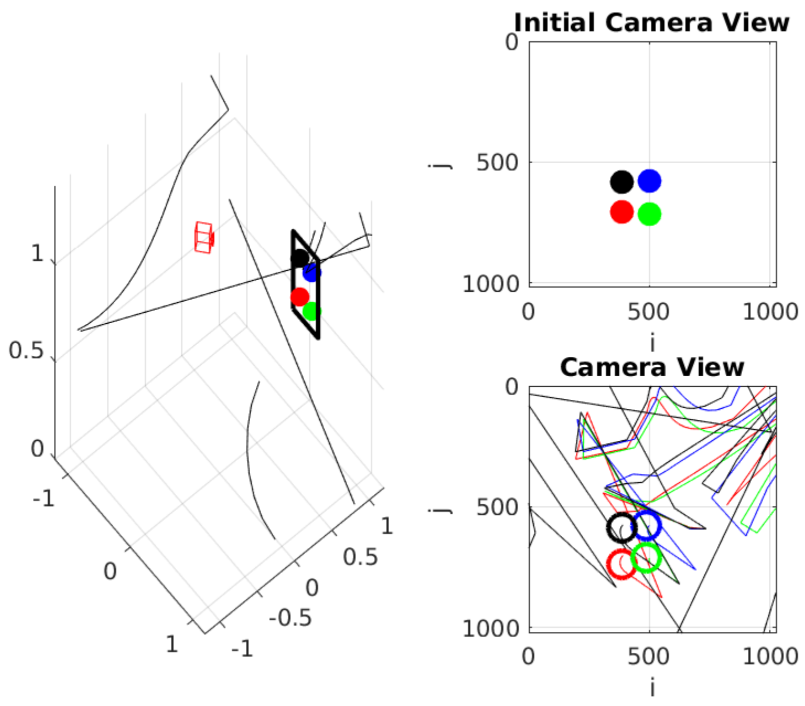
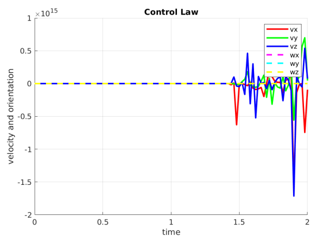

(c) YZ View

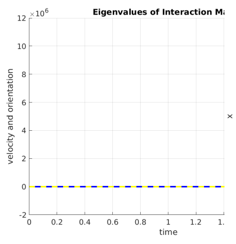
Figure 11: The diverge problem: Motion Views



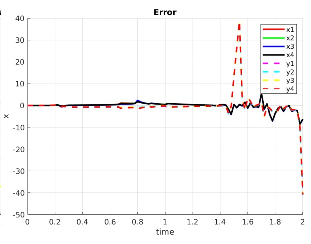
(a) The camera motion



(b) Control Law



(c) Eignevalues



(d) Errors

Figure 12: The diverge problem.

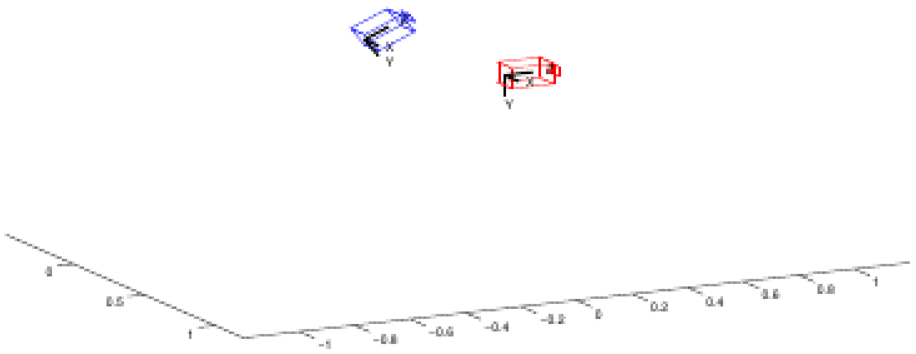
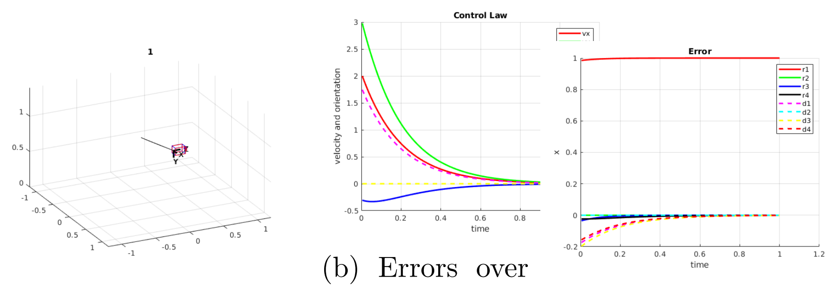


Figure 13: Camera pose control. Blue camera shows the initial pose of the camera, and the red camera shows the desired pose of the camera. .



(a) Camera movement (b) Errors over time (c) Camera control law

Figure 14: Position Based Visual Servoing

5.2 Position Based Visual Servoing

We want to move a camera from its current Cartesian pose to a desired Cartesian pose (see Figure 13 on page 15). We assume that we can measure the Cartesian pose of the camera at every instant of time using a sensor.

5.2.1 Matlab simulation

Here the camera pose at each time step is known. The movement of the camera is very smooth and it reach the desired location without problems. The simulation results is given at Figure 14 on page 15. The matlab code is on file `Demo1PBVS.m`.

6 Conclusions

The IBVS is a useful for controlling a robot using visual information. But as it it does not take into account the position of the camera in its 3D world and have many stability issues. When only one image is used, even small errors in the image measurements can lead to errors in the pose that can impact significantly the accuracy of the system. The PBVS use no visual information. Since the control scheme imposes a behaviour of **s** which is here expressed in the Cartesian space, it allows the camera to follow theoretically an optimal trajectory in that space but generally not in the image space. Even if the two basic approaches presented give in practice satisfactory results in most cases, their respective shortcomings have led to many works and improvements. Clearly a combination of both methods will give us better results.

6.1 MatlabCode

The code of this project is on Github repository `https://github.com/jtsagata/VisualServoingLab`.

7 Bibliography

References

[1] *Lecture and lab notes.*

[2] Farrokh Janabi-Sharifi, Visual Servoing : Theory and applications, Chapter 15 (book unknown)

[3] F. Chaumette, S. Hutchinso, *Visual Servo Control, Part I: Basic Approaches.* EEE Robotics and Automation Magazine, 13(4):82-90, December 2006.

[4] F. Chaumette, S. Hutchinson, *Visual Servo Control, Part II: Advanced Approaches.* Addison Wesley, Massachusetts, IEEE Robotics and Automation Magazine, 14(1):109-118, March 2007

[5] S. A. Hutchinson, G. D. Hager, and P. I. Corke, *A tutorial on visual servo control.* IEEE Trans. Robot. Automat., 12(5):651—670, Oct. 1996.

[6] P. I. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB.* Springer, 2013.