University of Burgundy

Masters in Computer Vision and Robotics

Visual Servoing Module

Report for 3D Pose Based Visual Servoing

By:

Vamshi Kodipaka

vamshikodipaka@gmail.com

Supervisor: Dr. Omar Tahri

Dated: 22-Dec-2019

3D POSE BASED VISUAL SERVOING

1 Introduction

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

Position Based Visual Servoing is a model-based technique (with a single camera). This is because the pose of the object of interest is estimated with respect to the camera and then a command is issued to the robot controller, which in turn controls the robot. In this case the image features are extracted as well, but are additionally used to estimate 3D information (pose of the object in Cartesian space), hence it is servoing in 3D.

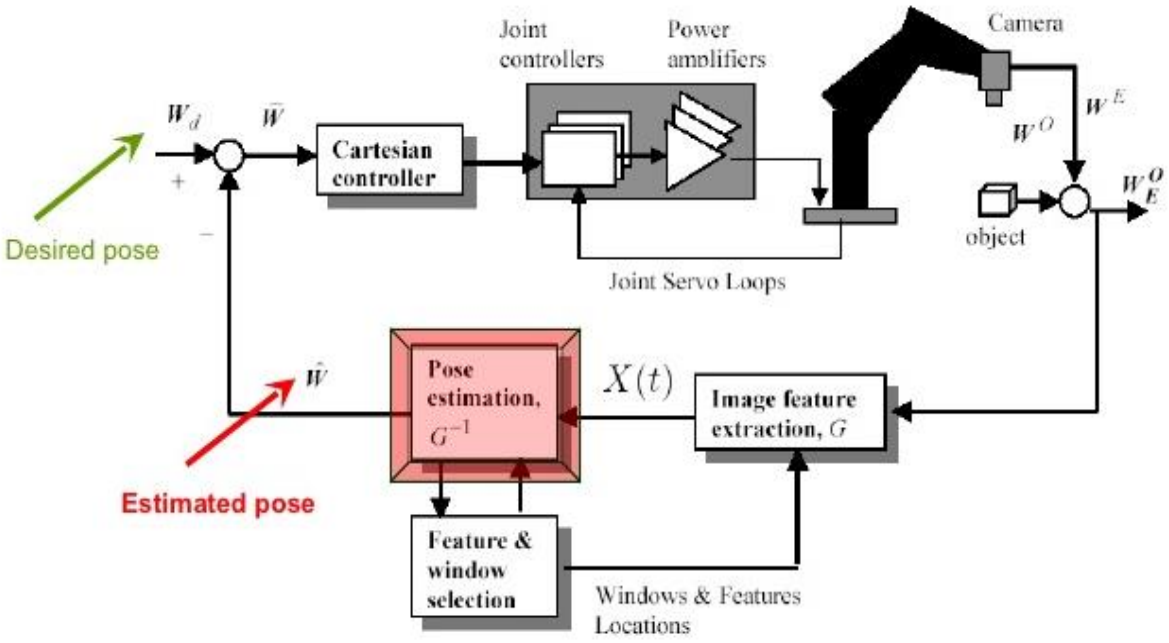
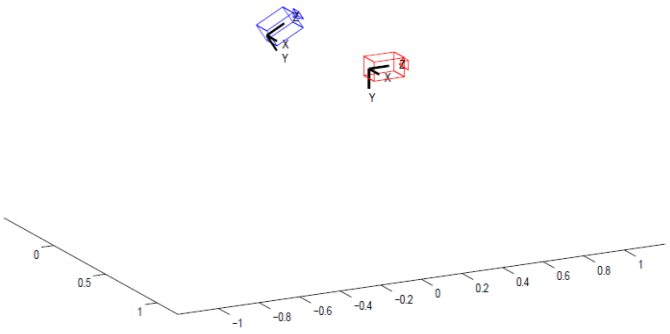


Fig-1: Block Representation of 3D Pose based Visual Servoing

Problem Statement:

We want to move a camera from its current Cartesian pose to a desired Cartesian pose (see following Fig. 2).

We assume that we can measure the Cartesian pose of the camera at every instant of time using a sensor.



Solution-1: Realizing the 3D Position Based Control

3D Camera pose control: Blue camera shows the initial pose of the camera, and the red camera shows the desired pose of the camera. 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 Virtual Reality applications.

The Pose Control problem:

The general structure of a PBVS is shown in Fig.1. 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 **s** = **s**($\boldsymbol{p_t}$), are used with the models of camera and object geometry to estimate the relative pose vector **W** of the object with respect to the end-effector. The estimated pose is compared with the desired relative pose **W***d* to calculate the relative pose error **W'** error. The coordinate frames involved in the process is given in Fig.3.

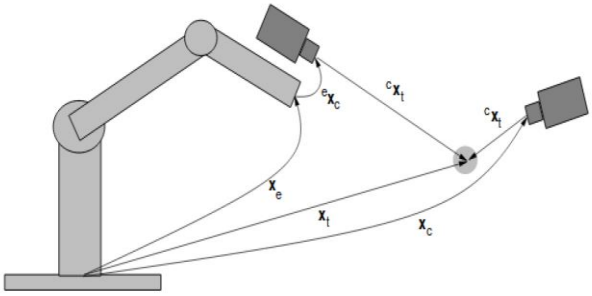


Fig.3: The coordinate frames: word, end-effector, camera and target.

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., event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, and image restoration.

Solution-2: Mathematical Formulation

General VS Calculation:

2.1 The visual features parameters extracted from the image are a function of the camera poses:

$$\mathbf{s} = \mathbf{s}(\mathbf{p}_t) \quad \ldots (1)$$ s – order of **2nX1**

2.2 By taking the derivative of the above relation we obtain

$$\dot{\mathbf{S}} = \mathbf{L}_s \mathbf{V} \quad \ldots (2)$$

where **L***s* the so called *Interaction matrix* or *feature Jacobian* of order **2nX6** and **V** the camera (Kinematic screw) denoted by **V** = ($\vec{v}$; $\vec{w}$). Thus the **V** contains 3 translations and 3 rotations.

2.3 The goal of the control law is to minimize the error given by:

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

where **S*** the desired value of the feature. Using equations (1) and (3)

2.4 Then the time variation of the error is:

$$\dot{\mathbf{e}} = \mathbf{L}_s \mathbf{V} \qquad \ldots (4)$$

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

$$\mathbf{V} = -\lambda \mathbf{L}_s^{+} \mathbf{e} \quad . \qquad (\lambda > 0) \qquad \ldots (5)$$

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

This corresponds to: $\mathbf{V} = -\lambda \widehat{\mathbf{L}_s^{+}} (\mathbf{S} - \mathbf{S}^*)$. … (6)

PBVS Calculation:

Position-based control schemes (PBVS) use the pose of the camera with respect to some reference coordinate frame to define s. Computing that pose from a set of measurements in one image necessitates the camera intrinsic parameters and the 3-D model of the object observed to be known. This classical computer vision problem is called the 3- D localization problem.

2.6 Let *s*(**X**) *belongs to* $\Re$ *is in the order of* 6*X*1 is a representation of the Cartesian pose **X** *belongs to* $\Re$ is in the order of 4X4 as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \quad s(\mathbf{X}) = \begin{bmatrix} \mathbf{t} \\ \mathbf{u}\theta \end{bmatrix} \qquad \ldots (7a)$$

where **utheta** corresponds to the rotation **R**. Here *u* is a unit rotation axis and *theta* is the rotation angle around this axis **u**.

2.7 The error can be calculated using the equation:

$$\mathbf{e} = s\left(\mathbf{X}_{B/R}^{-1} \star \mathbf{X}_{A(t)/R}\right) = \mathbf{e} = s\left(\mathbf{X}_{A(t)/B}\right) \qquad \ldots. (7b)$$

2.8 Finally, control law can be modified as:

$$\xi_{A(t)/B} = \begin{bmatrix} v \\ \omega \end{bmatrix}_{A(t)/B} = -\lambda \begin{bmatrix} \mathbf{R}^t\mathbf{t} \\ \theta\mathbf{u} \end{bmatrix}_{A(t)/B}, \quad \lambda > 0 \qquad \ldots (7c)$$

.

2.9 We should then express the control law:
A(*t*)=*B* with respect to the fixed reference frame *R* rather than desired pose frame *B*.

$$\xi_{A(t)/R} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0}_{3\times 3} & \mathbf{R} \end{bmatrix}_{B/R} \xi_{A(t)/B} \qquad \ldots.. (8)$$

→ $\mathbf{s} = ({}^{c}\mathbf{t}_o, \theta\mathbf{u}),\ \mathbf{s}^* = ({}^{c^*}\mathbf{t}_o, \mathbf{0}),$ and $\mathbf{e} = ({}^{c}\mathbf{t}_o - {}^{c^*}\mathbf{t}_o, \theta\mathbf{u})$.

2.6 Interaction matrix is given by the following equation:

$$\mathbf{L_e} = \begin{bmatrix} -\mathbf{I}_3 & [{}^c\mathbf{t}_o]_\times \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix} \quad \ldots (9)$$

In which I_3, is is a 3X3 identity matrix and $L_{\theta u}$ is

$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\operatorname{sinc}\theta}{\operatorname{sinc}^2\frac{\theta}{2}}\right)[\mathbf{u}]_\times^2, \quad \ldots. (10)$$

This PBVS scheme causes the rotational motion to follow a geodesic with an exponential decreasing speed and so that the translational parameters involved in **s** decrease with the same speed. This explains the nice exponential decrease of the camera velocity components. Furthermore, the trajectory in the image of the origin of the object frame follows a pure straight line. On the other hand, the camera trajectory does not follow a straight line.

2.6 Stability analysis: The global asymptotic stability of the system is thus obtained when the following sufficient condition is ensured

$$\mathbf{L_e}\widehat{\mathbf{L_e^+}} > 0. \quad \ldots. (11)$$

Since $L_{\theta u}$ given in eqn.10 is nonsingular when θ not equal to 2 kπ, we obtain from eqn.11 the global asymptotic stability of the system since L_e $\boldsymbol{L_e^{-1}}$=I_6, under the strong hypothesis that all the pose parameters are perfect.

Solution-3: Block Diagram of PBVS (Refer Fig.1 above)

Solution-4: Algorithm

Algorithm: Postion Based Visual Servoing

---

while *not at end of time* **do**

Build the feature vector

Find the error **e** using equation (7b)

Calculate $\widehat{\mathbf{L}_s^+}$

$$\xi_{A(t)/R} = \begin{bmatrix} \mathbf{R} & [\mathrm{t}_\times\mathbf{R}] \\ \mathbf{0}_{3\times3} & \mathbf{R} \end{bmatrix}_{B/R} \xi_{A(t)/B}$$

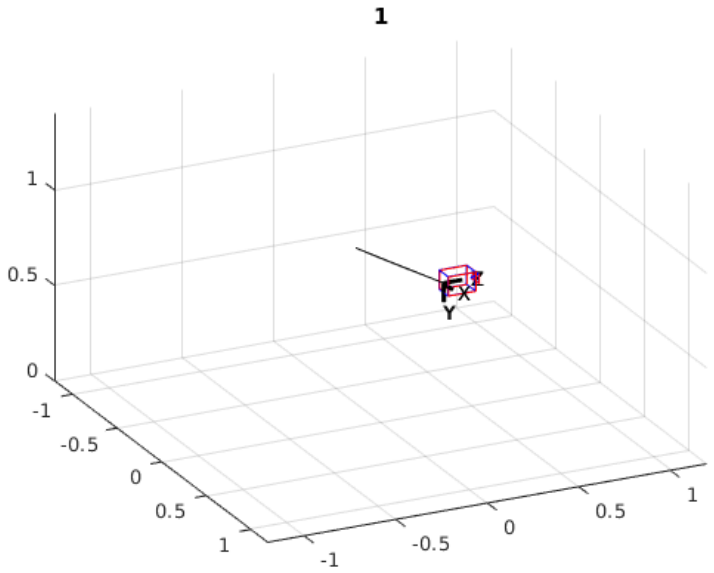
Apply the control law and get new camera location

$$\mathbf{X}_{A(T+\Delta t)/R} = \begin{bmatrix} \Delta t\,[\omega]_\times & \Delta t\, v \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}_{A(\Delta t)/R} \mathbf{X}_{A(t)/R}$$

end

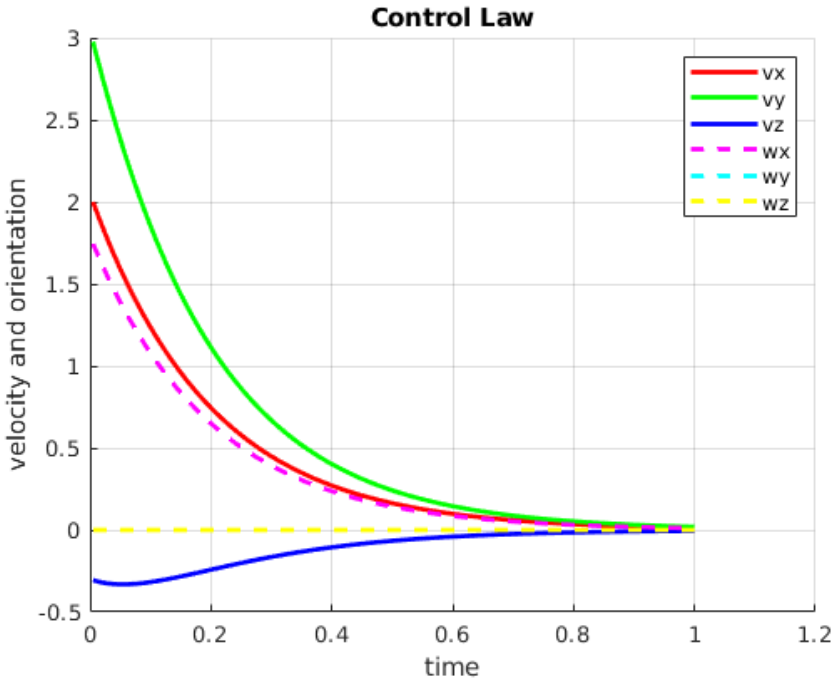
Solution-5: Matlab Simulation

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

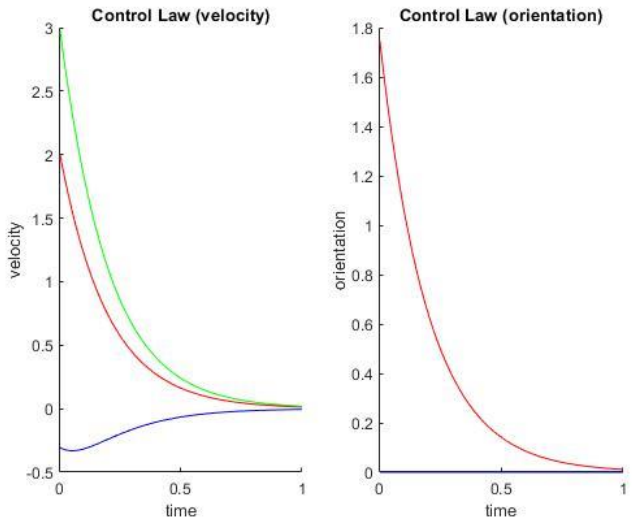


a. After Camera Motion to estimate PBVS

The parameters assumed will keep on reducing the velocity and orientation w.r.t time in 3D world.

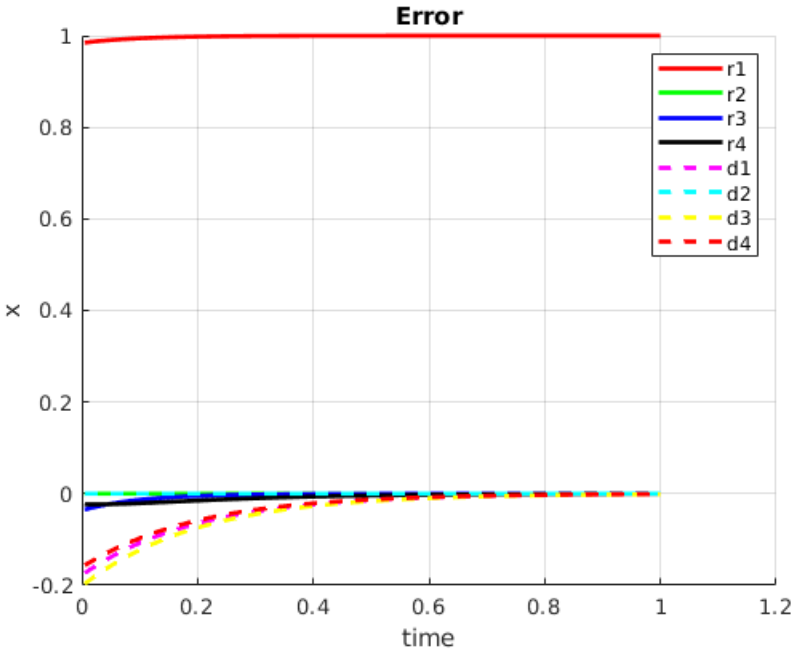


a. Control Law for PBVS



b. Velocity and Orientation of Camera in PBVS

Solution-6: Error for PBVS



a. Error plot for PBVS

Appendix - Matlab Functions

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

Conclusion:

The PBVS use no visual information. Since the control scheme imposes a behavior 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.

Results:

To see the results: RUN:: pbvsTP.m file in the zipped folder

References

[1] *Lecture slide, matlab source code and lab notes of Prof.Omar Tahri*.
[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*. IEEE 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.
[7] Git-hub: https://github.com/jtsagata/VisualServoingLab

THE END