An Introduction to Visual Servo Control [a]

Seth Hutchinson — University of Illinois

[a] based on the articles "Visual Servo Control, Part I: Basic Approaches," and "Visual Servo Control, Part II: Advanced Approaches," in the *IEEE Robotics and Automation Magazine*, by François Chaumette and Seth Hutchinson.

Visual Servo Control — The Basic Idea

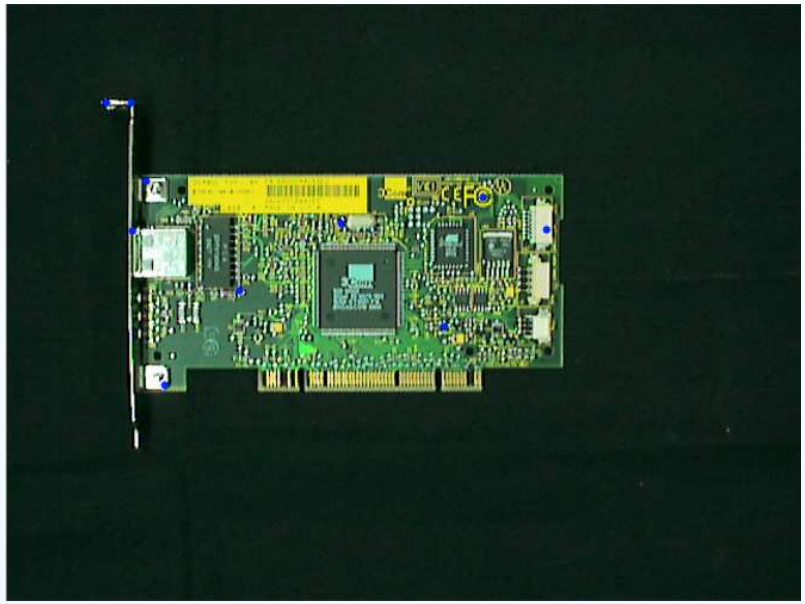
The aim of vision-based control schemes is to minimize an error $e(t)$ which is typically defined by

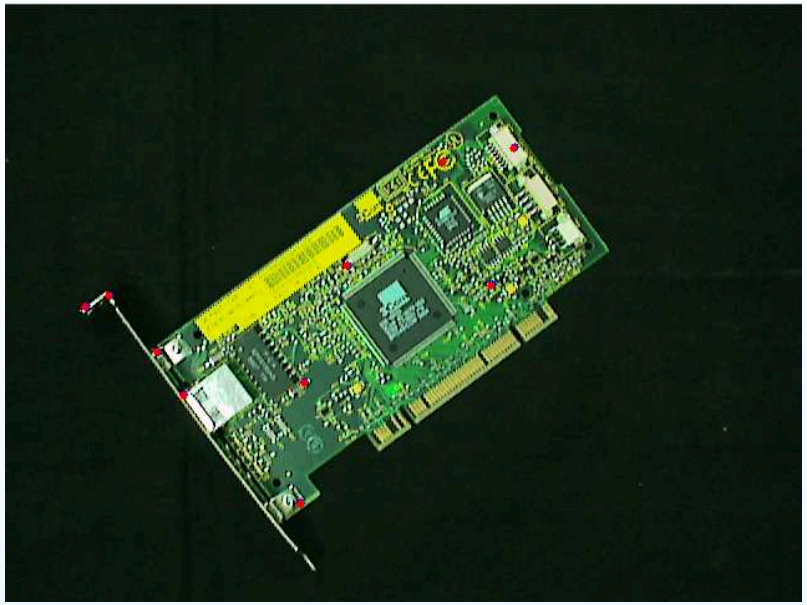
$$e(t) = s(m(t), a) - s^*$$

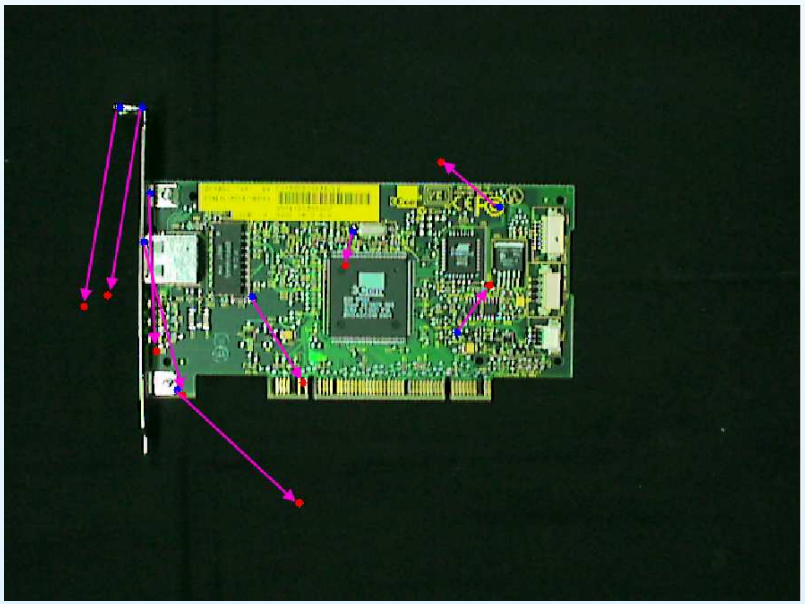
- The vector $m(t)$ is a set of image measurements (e.g., the image coordinates of interest points, or the parameters of a set of image segments).
- The image measurements are used to compute a vector of k visual features, $s(m(t), a)$.
- The vector a is a set of parameters that represent potential additional knowledge about the system (e.g., coarse camera intrinsic parameters or 3D model of objects).
- The vector s^* contains the desired values of the features.

Typically, one merely writes: $e(t) = s(t) - s^*$

An Example







$s(t) =$ coordinates of image points

Some Basic Assumptions

There are numerous considerations when designing a visual servo system. For now, we will consider only systems that satisfy the following basic assumptions:

- Eye-in-hand systems — the camera is mounted on the end effector of a robot and treated as a free-flying object with configuration space $\mathcal{Q} = SE(3)$.
- Static (i.e., motionless) targets.
- Purely kinematic systems — we do not consider the dynamics of camera motion, but assume that the camera can execute accurately the applied velocity control.
- Perspective projection — the imaging geometry can be modelled as a pinhole camera.

Some or all of these may be relaxed as we progress to more advanced topics.

Designing the Control Law — The Basic Idea

Given s, control design can be quite simple.

A typical approach is to design a velocity controller, which requires the relationship between the time variation of s and the camera velocity.

- Let the spatial velocity of the camera be denoted by $\xi = (v, \omega)$,
  - v is the instantaneous linear velocity of the origin of the camera frame and
  - ω is the instantaneous angular velocity of the camera frame.
- The relationship between $\dot{s}$ and ξ is given by

$$\dot{s} = L_s \xi$$

in which $L_s \in \mathbb{R}^{k \times 6}$ is named the *interaction matrix* [Espiau, et al. 1992], or the *image Jacobian* [Hutchinson, et al. 1996].

The key to visual servo — choosing s and the control law.

Designing the Control Law (cont)

We now derive the relationship $\dot{e} = L_e \xi$, which relates the change in error the commanded camera velocity.

Using the previous equations

$$e(t) = s(t) - s^* \quad and \quad \dot{s} = L_s \xi$$

we can easily obtain the relationship between the camera velocity and the rate of change of the error, $\dot{e} = L_e \xi$ as

$$\dot{e}(t) = \dot{s}(t) = L_s \xi$$

assuming that s^* is constant.

In this case, we have $L_e = L_s \longrightarrow$ the relationship between ξ and $\dot{s}$ is the same as between ξ and $\dot{e}$.

Now our problem is merely to find the control input $\xi = u(t)$ that gives the desired error performance.

Designing the Control Law (cont)

- In many cases, we would like to ensure an exponential decoupled decrease of the error

$$\dot{e} = -\lambda e$$

- To this end, we may choose the control law as

$$u(t) = \xi = -\lambda L_e^+ e \tag{1}$$

where $L_e^+ \in \mathbb{R}^{6\times k}$ is chosen as the Moore-Penrose pseudo-inverse of L_e,

$$L_e^+ = (L_e^\top L_e)^{-1} L_e^\top$$

when L_e is of full rank 6.

- This is a *left* pseudoinverse, since

$$L_e^+ L_e = \left[(L_e^\top L_e)^{-1} L_e^\top\right] L_e = I$$

Designing the Control Law (cont)

- The choice of $\xi = -\lambda L_e^+ e$ is the usual least squares solution, and gives

$$\dot{e} = L_e \xi = -\lambda L_e L_e^+ e$$

- This is also the solution that locally minimizes $\|\xi\|$.

- When $k = 6$, if $\det L_e \neq 0$ it it possible to invert L_e. In this case we may use the control

$$\xi = -\lambda L_e^{-1} e$$

  which gives exactly the desired behavior of $\dot{e} = -\lambda e$.

Practical Issues

In practice, it is impossible to know exactly the value of L_e or of L_e^+, since these depend on measured data.

The actual value of L is thus an approximation, and the actual control law is thus given as

$$\xi = -\lambda \widehat{L_e^+} e$$

There are several choices for $\widehat{L_e^+}$, several of which will be discussed in more detail later:

- Let $\widehat{L_e^+} = \widehat{L}_e^+$: Compute an estimate $\widehat{L}_e$, and use the pseudo inverse of the estimate.
- Directly estimate $\widehat{L_e^+}$.
- Let $\widehat{L_e^+}$ be approximated by a constant matrix L.

Context — Visual Servo in the Bigger Picture

- Learning, planning, perception and action are often tightly coupled activities.
- Visual servo control is the coupling of perception and action
  — hand-eye coordination.
- Basic visual servo controllers can serve as primitives for planning systems.
- Switching between control laws is equivalent to executing a plan.
- There are a number of analogies between human hand-eye coordination and visual servo control.

A rigorous understanding of the performance of visual servo control systems provides a foundation for sensor-based robotics.

Visual Servo Control — Some History

Visual servo control is merely the use of computer vision data to control the motion of a robot.

In some sense, Shakey [SRI, 1966-1972 or so] was an example of a visual servo system — but with a very, very slow servo rate.

The first real-time visual servo systems were reported in

- Agin, 1979
- Weiss et al., 1984, 1987
- Feddema et al., 1989

In each of these, simple image features (e.g., centroids of binary objects) were used, primarily due to limitations in computation power.

Overview

This talk will focus on the control and performance issues, leaving aside the computer vision issues (e.g., feature tracking).

The main issues — how to choose $s(t)$ and the corresponding control law.

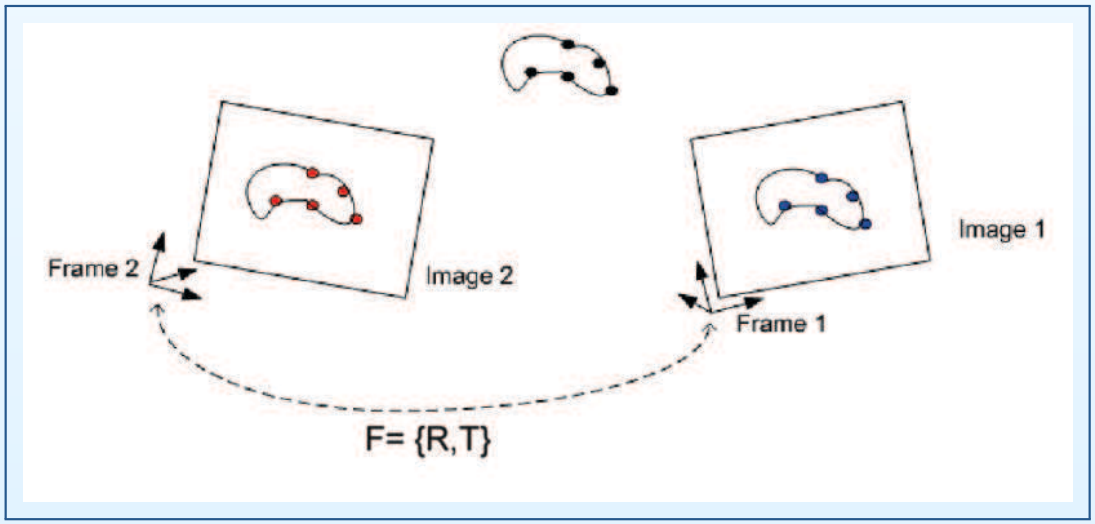
- Using 3D reconstruction to define $s(t)$
- Using image data directly to define $s(t)$
- Partitioning degrees of freedom
- Switching between controllers
- Adding constraints and dynamics

POSITION-BASED VISUAL SERVO CONTROL

POSITION-BASED VISUAL SERVO

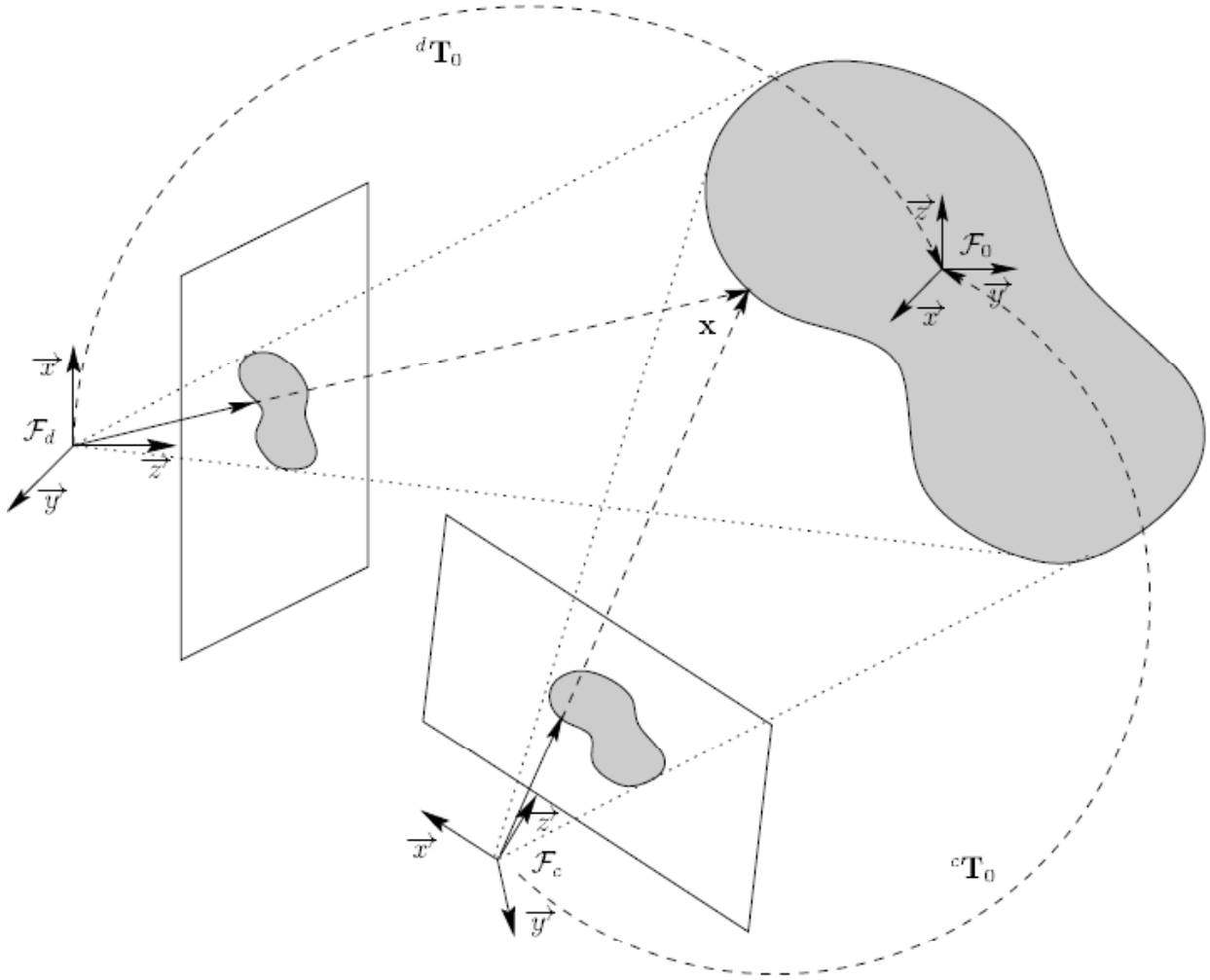
- Computer vision data are used to compute the pose of the camera d, R relative to the world frame.
- The error $e(t)$ is defined in the pose space, $d \in \mathbb{R}^3, R \in SO(3)$.
- The control signal $\xi = (v, \omega)$ is a camera body velocity.
- The camera velocity ξ is specified w.r.t. the camera frame.

If the goal pose is given by $d = 0, R = I$, the role of the computer vision system is to provide, in real time, a measurement of the pose error.



Requerimientos del PBVS

- Un control basado en posición requiere:
  - Los parámetros intrínsecos de la cámara.
  - Un modelo 3D del objeto observado.
- Aplicar un método de *3D pose estimation*



Representación de la rotación

- **Formula de Rodriguez**: dado un eje de rotación representado por el vector $\mathbf{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$, la rotación se puede representar por:

$$\mathbf{R}(\mathbf{u}, \theta) = \mathbf{I} + \sin(\theta)\,[\mathbf{u}]_\times + (1 - \cos(\theta))\,[\mathbf{u}]_\times^2$$

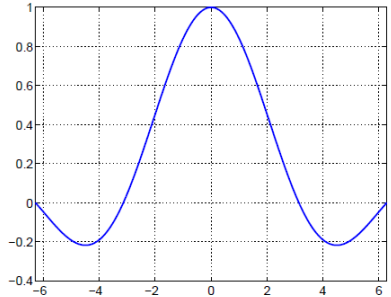
donde

$$[\mathbf{u}]_\times = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

- Así pues, dada una matriz de rotación $\mathbf{R} \in SO(3)$, la representación $\mathbf{u}\theta$ de dicha rotación se puede encontrar como:

$$\theta = \arccos\left(\frac{1}{2}(r_{11} + r_{22} + r_{33} - 1)\right), \quad \mathbf{u}\theta = \frac{1}{2}\frac{1}{\operatorname{sinc}(\theta)}\begin{bmatrix} r_{32} - r_{23} \\ r_{31} - r_{13} \\ r_{21} - r_{12} \end{bmatrix}$$

donde $\operatorname{sinc}(\theta) = \dfrac{\sin(\theta)}{\theta}$ es la función seno cardinal:

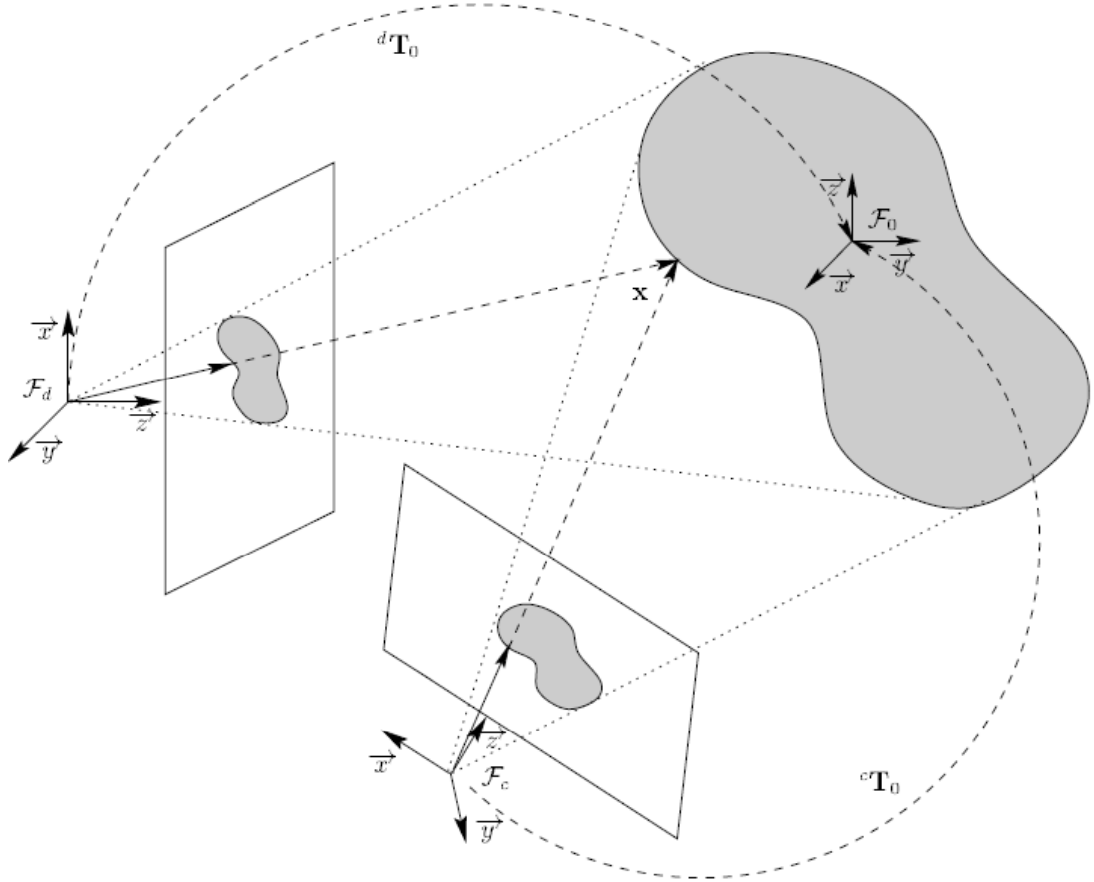


Geometría cámaras-objeto

$^{c}\mathbf{t}_{o}$ ⇒ Coordenadas del SR asociado a la posición actual de la cámara expresado en el SR objeto.

$^{c^*}\mathbf{t}_{o}$ ⇒ Coordenadas del SR asociado a la posición deseada de la cámara expresado en el SR objeto.

$\mathbf{R} = {}^{c^*}\mathbf{R}_{c}$ ⇒ Matriz de rotación que da la orientación de SR actual relativo al SR deseado.



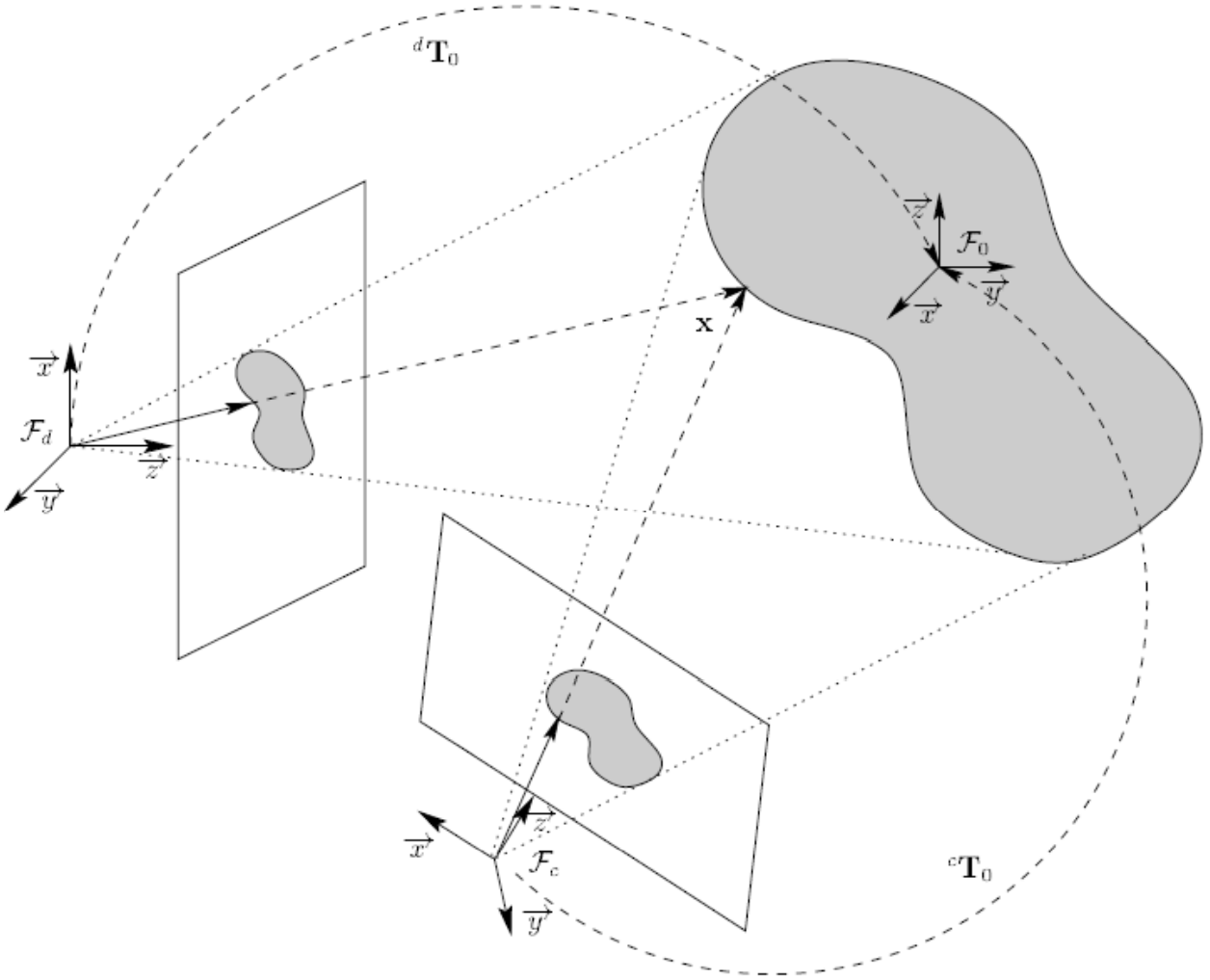
Definición de error

- **Primer caso:**
  - Error como la diferencia de posiciones relativas en el SR objeto:

$$\mathbf{s} = ({}^{c}\mathbf{t}_o, \theta\mathbf{u})$$

$$\mathbf{s}^* = ({}^{c^*}\mathbf{t}_o, \mathbf{0})$$

$$\mathbf{e} = ({}^{c}\mathbf{t}_o - {}^{c^*}\mathbf{t}_o, \theta\mathbf{u})$$



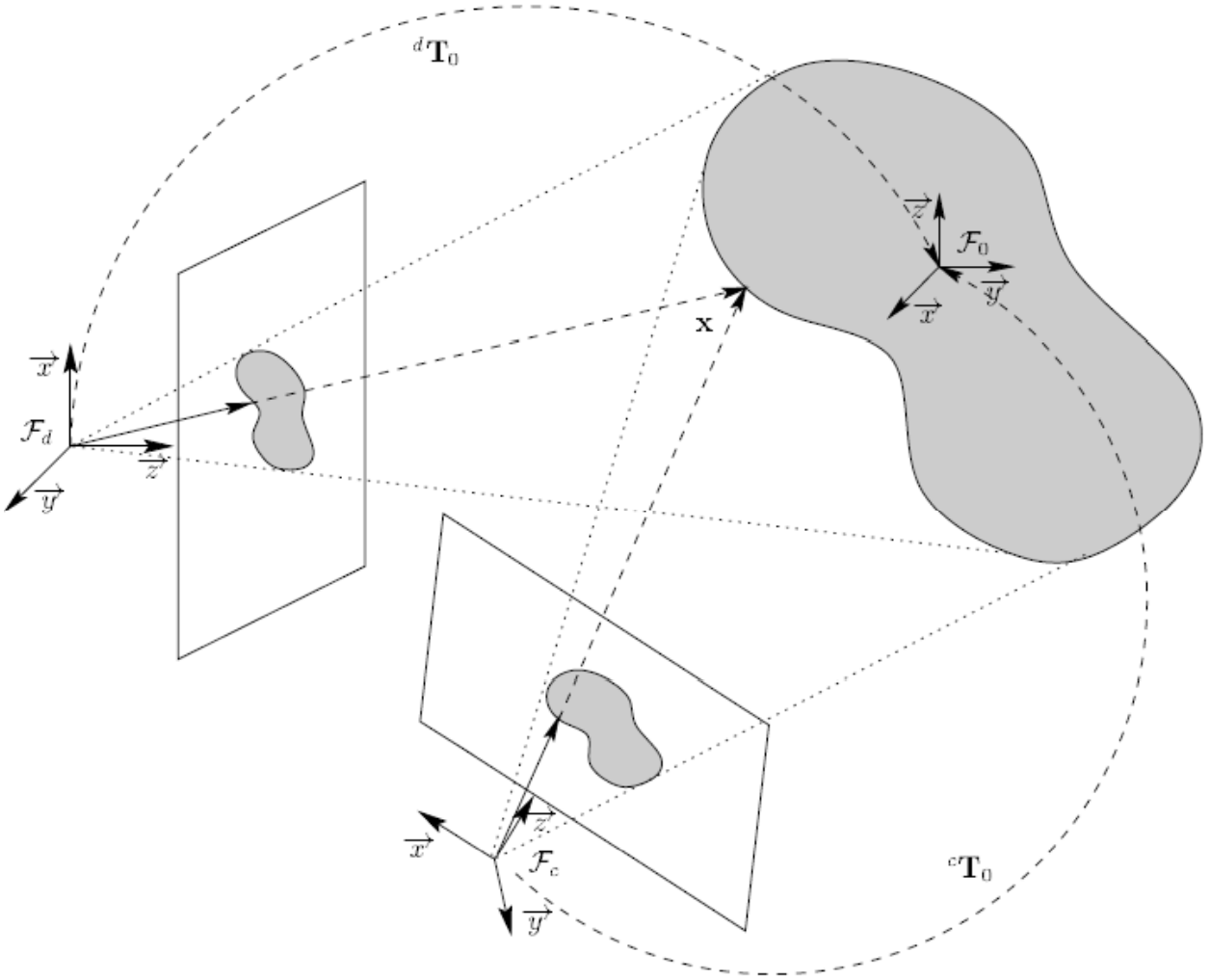
Definición de error

- **Segundo caso:**
  - Medidas visuales expresadas en el SR deseado:

$$\mathbf{s} = ({}^{c^*}\mathbf{t}_c, \theta\mathbf{u})$$

$$\mathbf{s}^* = \mathbf{0}$$

$$\mathbf{e} = \mathbf{s}$$



Caso 1: SR objeto

- Función de error: $\mathbf{e} = ({}^{c}\mathbf{t}_o - {}^{c^*}\mathbf{t}_o, \theta\mathbf{u})$
- En este caso, para derivar el error es necesario recordar:
  - Velocidad lineal de un punto 3D debida a una velocidad lineal + una velocidad angular.

$$\dot{\mathbf{X}} = -\boldsymbol{v}_c - \omega_c \times \mathbf{X} \Leftrightarrow \begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X. \end{cases}$$

- La parte de la velocidad de rotación hace uso de la representación de rotación $\mathbf{u}\theta$

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

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

Caso 1: SR objeto

- La ley de control es entonces:

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L}_{\mathbf{e}}^{-1}} \mathbf{e}$$

con

$$\widehat{\mathbf{L}_{\mathbf{e}}^{-1}} = \begin{bmatrix} -\mathbf{I}_3 & [{}^c\mathbf{t}_o]_\times \mathbf{L}_{\theta\mathbf{u}}^{-1} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}}^{-1} \end{bmatrix}$$

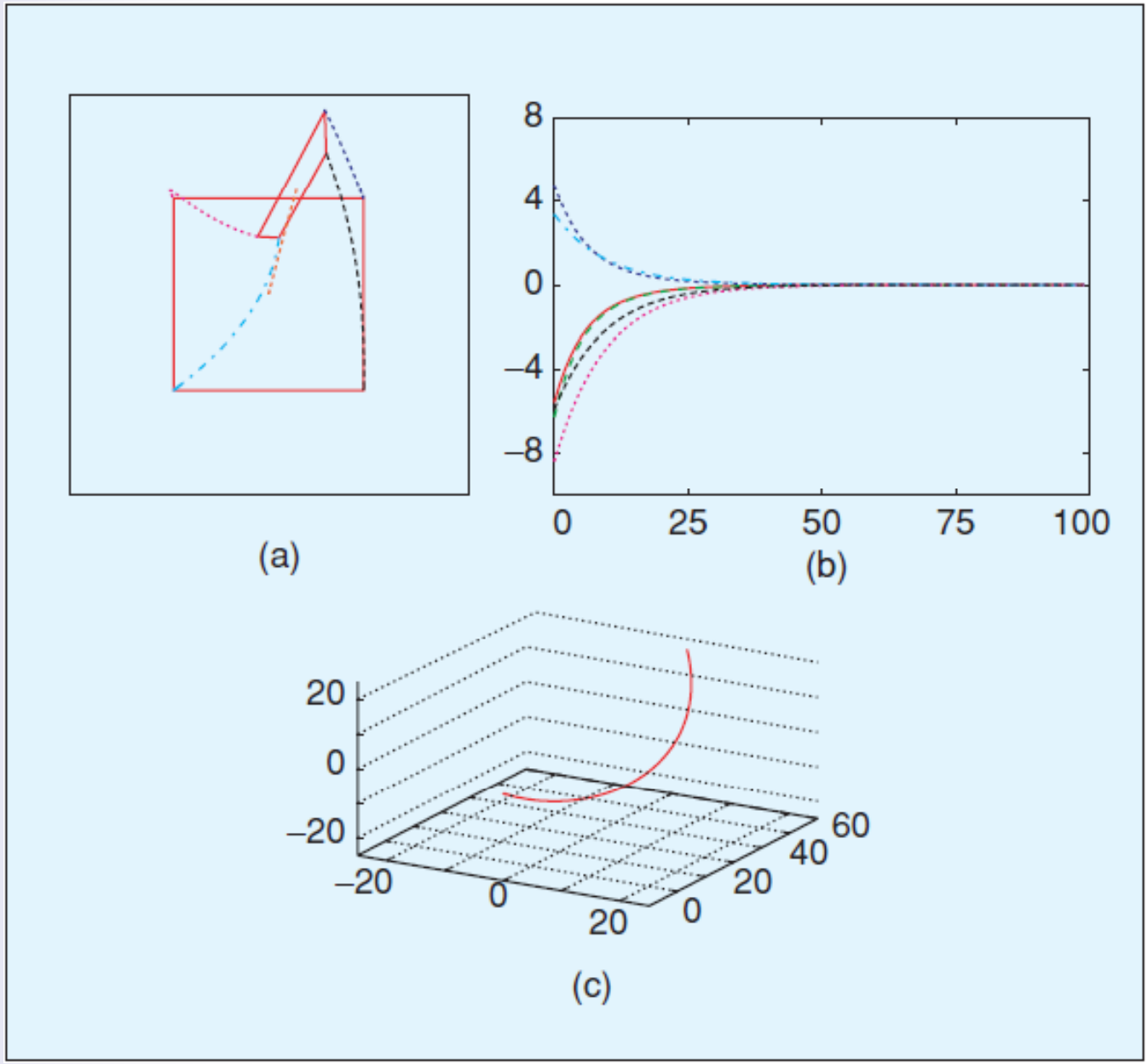
- Finalmente:

$$\begin{aligned} \boldsymbol{v}_c &= -\lambda(({}^{c^*}\mathbf{t}_o - {}^c\mathbf{t}_o) + [{}^c\mathbf{t}_o]_\times \theta\mathbf{u}) \\ \boldsymbol{\omega}_c &= -\lambda\theta\mathbf{u}, \end{aligned}$$

- Propiedad:

$$\mathbf{L}_{\theta\mathbf{u}}^{-1}\theta\mathbf{u} = \theta\mathbf{u}$$

Caso 1: SR objeto $\mathbf{s} = ({}^{C}\mathbf{t}_0, \theta\mathbf{u})$



Caso 2: SR deseado

- Función de error: $\mathbf{e} = \mathbf{s} = ({}^{c^*}\mathbf{t}_c, \theta\mathbf{u})$
- En este caso la velocidad lineal no se ve afectada por la velocidad angular.
- La velocidad lineal en cada eje se calcula como la proyección del vector de velocidad lineal de entrada aplicado.

- La parte de la velocidad de rotación es la misma que antes

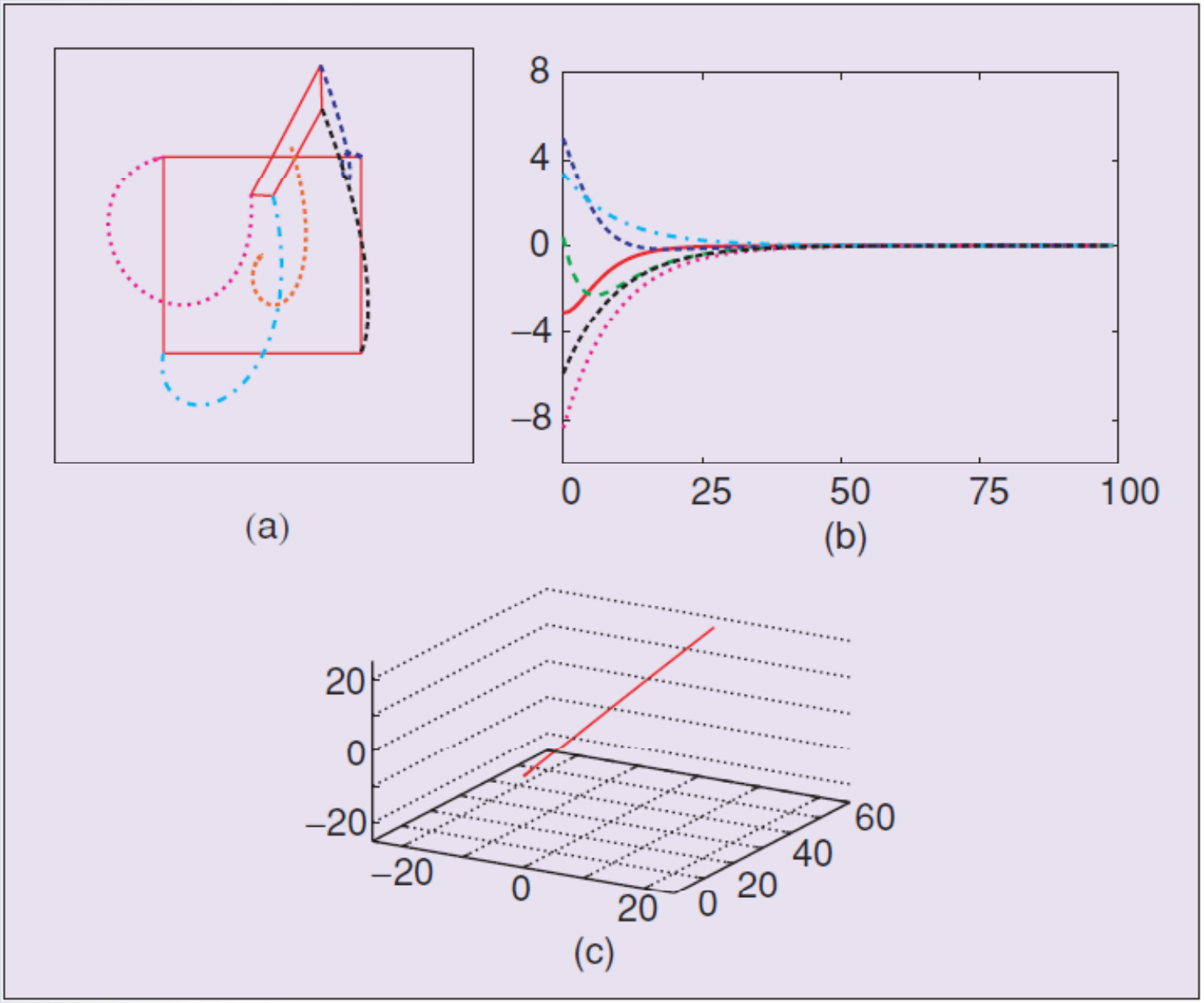
$$\mathbf{L_e} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}$$

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

- La ley de control es entonces:

$$\begin{aligned} \boldsymbol{v}_c &= -\lambda \mathbf{R}^\top {}^{c^*}\mathbf{t}_c \\ \boldsymbol{\omega}_c &= -\lambda\, \theta\mathbf{u}. \end{aligned}$$

Caso 2: SR deseado $\mathbf{s} = ({}^{C^*}\mathbf{t}_C, \theta\mathbf{u})$



PBVS (cont.)

If u, θ be the axis/angle parameterization of R, the error is given by

$$e(t) = \begin{bmatrix} d \\ u\theta \end{bmatrix}$$

and its derivative is given by

$$\dot{e} = \begin{bmatrix} R & 0 \\ 0 & L_\omega(u,\theta) \end{bmatrix} \xi = L_{\mathrm{PBVS}}(u,\theta)\xi$$

in which [Malis 98]

$$L_\omega(u,\theta) = I - \frac{\theta}{2}u_\times + \left(1 - \frac{\mathrm{sinc}\,\theta}{\mathrm{sinc}^2\frac{\theta}{2}}\right) u_\times^2$$

PBVS (cont.)

Since L_ω is non singular when $\theta \neq 2k\pi$, [Malis, Chaumette, Boudet 99], to achieve the error dynamics $\dot{e} = -\lambda e$, we can use

$$\begin{aligned} -\lambda e &= \dot{e} = L_{\text{PBVS}}(u,\theta)\xi \\ &\rightarrow \xi = -\lambda L_{\text{PBVS}}^{-1}(u,\theta)e \end{aligned}$$

The motivation: The solution of the differential equation $\dot{e} = -\lambda e$ is a decaying exponential.

That's nice — but how do we know it really works? After all, $e(t)$ is a vector, and L is *not* a constant matrix, so this isn't really a nice, scalar, first-order, linear differential equation.

Lyapunov Theory

Lyapunov theory provides a powerful tool for analyzing the stability of nonlinear systems.

- Consider a nonlinear system on $\mathbb{R}^n$

$$\dot{x} \quad = \quad f(x) \tag{2}$$

  where $f(x)$ is a vector field on $\mathbb{R}^n$, and suppose that $f(0) = 0$.
- The origin in $\mathbb{R}^n$ is said to be an **equilibrium point** for Equation (2).
- Let $\mathcal{L}(x) : \mathbb{R}^n \to \mathbb{R}$ be a continuous function with continuous first partial derivatives in a neighborhood of the origin in $\mathbb{R}^n$.
- Further, let $\mathcal{L}$ be positive definite, that is, $\mathcal{L}(0) = 0$ and $\mathcal{L} > 0$ for $x \neq 0$.
- $\mathcal{L}$ is called a **Lyapunov function candidate** for the system given by Equation (2).

Lyapunov Theory

The null solution of Equation (2) is **stable** if there exists a Lyapunov function candidate $\mathcal{L}$ such that $\dot{\mathcal{L}}$ is negative semi-definite along solution trajectories of Equation (2), that is, if

$$\dot{\mathcal{L}} = \frac{\partial \mathcal{L}}{\partial x} f(x) \leq 0$$

The null solution of Equation (2) is **asymptotically stable** if there exists a Lyapunov function candidate $\mathcal{L}$ such that $\dot{\mathcal{L}}$ is negative definite along solution trajectories of Equation (2), that is, if

$$\dot{\mathcal{L}} = \frac{\partial \mathcal{L}}{\partial x} f(x) < 0$$

Lyapunov Theory and Visual Servo

The two versions of stability provide different sorts of performance guarantees.

- Stability gaurantees that the system will remain within a neighborhood of the equilibrium point, provided the initial state is sufficiently close to the equilibrium point.
- Asymptotic stability gaurantees that the system will converge to the equilibrium point, provided the initial state is sufficiently close to the equilibrium point.

In some cases, the system error function is the simplest Lyapunov function candidate — this is the case for many visual servo control systems:

$$\mathcal{L} = \frac{1}{2}\|e(t)\|^2 \qquad \longrightarrow \qquad \dot{\mathcal{L}} = e^T(t)\dot{e}(t)$$

Lyapunov stability of PBVS

Recall our PBVS controller:

$$\begin{aligned} -\lambda e &= \dot{e} = L_{\text{PBVS}}(u,\theta)\xi \\ &\rightarrow \xi = -\lambda L_{\text{PBVS}}^{-1}(u,\theta)e \end{aligned}$$

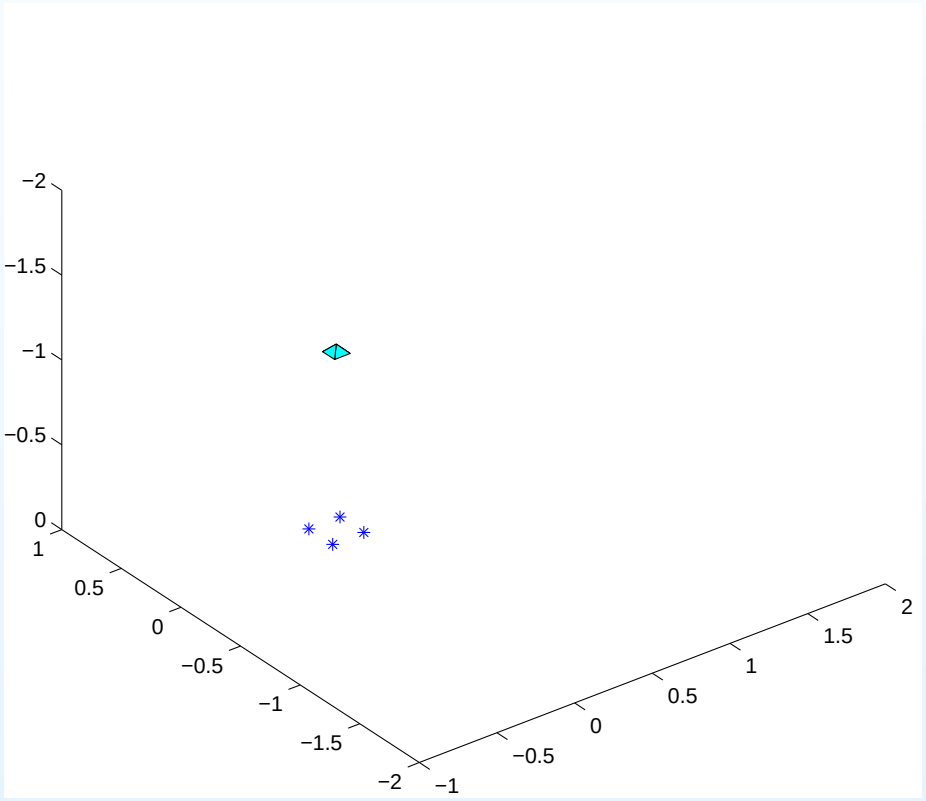
Using the Lyapunov function $\mathcal{L} = \frac{1}{2}\|e(t)\|^2$ we obtain

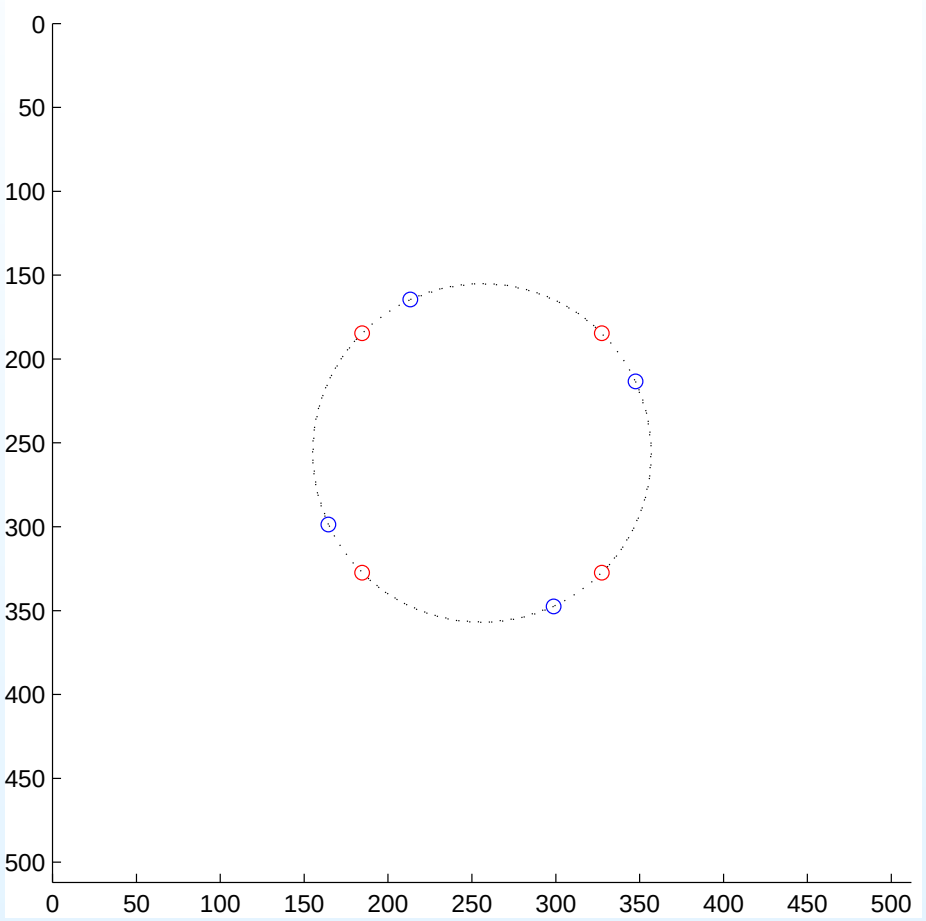
$$\begin{aligned} \dot{\mathcal{L}} &= e^T\dot{e} \\ &= e^T L_{\text{PBVS}}(u,\theta)\xi \\ &= -\lambda e^T L_{\text{PBVS}}(u,\theta)L_{\text{PBVS}}^{-1}(u,\theta)e \\ &= -\lambda\|e\|^2 \end{aligned}$$

and we have, not surprisingly, asymptotic stability.

PBVS Task Example

160^o rotation about the optical axis





Why not just use PBVS ?

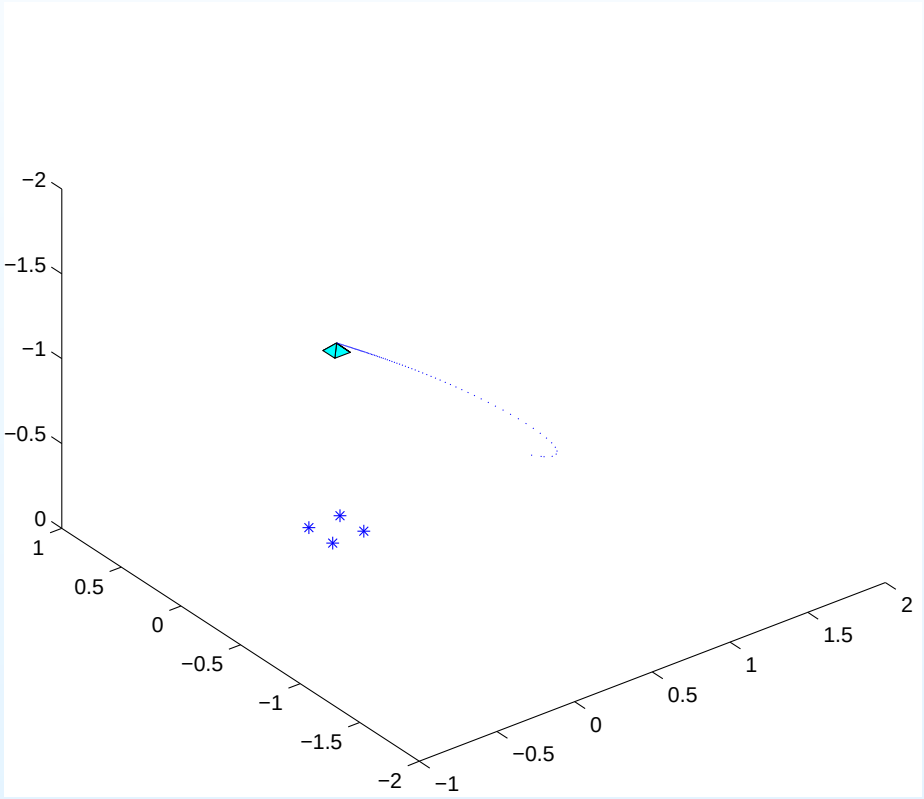
- Feedback is computed using *estimated* quantities that are a function of the system calibration parameters. Thus,

$$\dot{\mathcal{L}} = -\lambda e^T L_{\mathrm{PBVS}}(u,\theta)\hat{L}^{-1}_{\mathrm{PBVS}}(u,\theta)e$$

  and we need $L_{\mathrm{PBVS}}(u,\theta)\hat{L}^{-1}_{\mathrm{PBVS}}$ to be positive definite.
  Even small errors in computing the orientation of the cameras can lead to reconstruction errors that can impact significantly the accuracy of the system.
- Position-based control requires an accurate model of the target object—a form of calibration.
- In task space, the robot will move a minimal distance.
  In the image, features may move a non-minimal distance during execution - features may even leave the field of view.

PBVS Task Example

Large translation and rotation about all axes



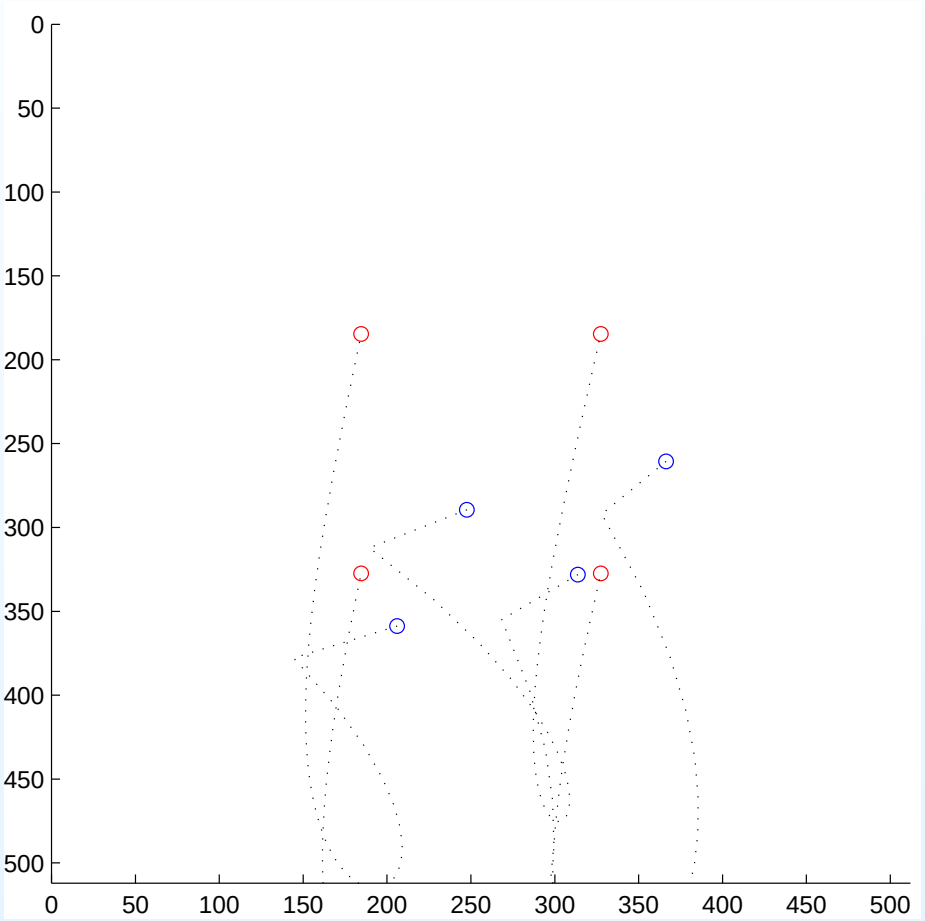


IMAGE-BASED VISUAL SERVO CONTROL

Image-Based Visual Servo Control

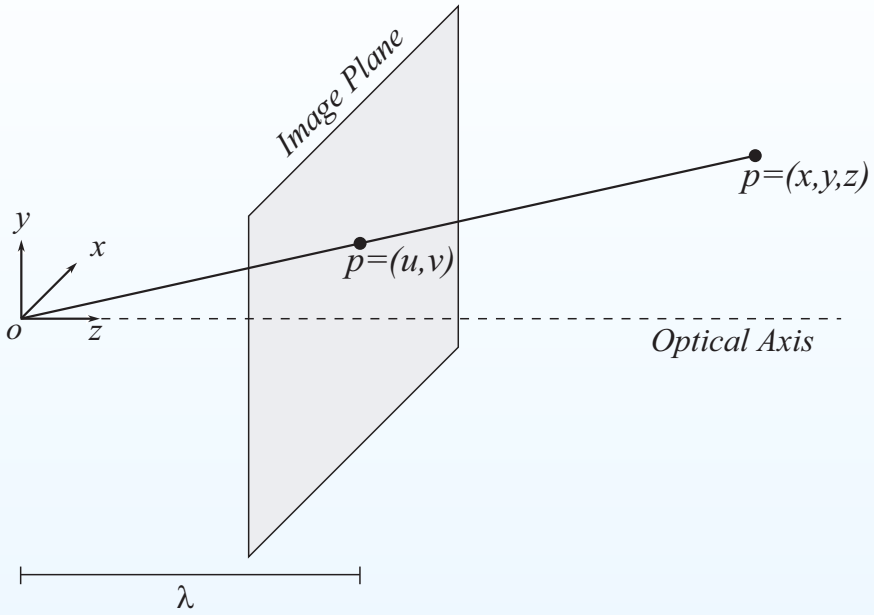
For Image-Based Visual Servo (IBVS)

- Features $s(t)$ are extracted from computer vision data.
- Camera pose is *not* explicitly computed.
- The error is defined in the image feature space, $e(t) = s(t) - s^*$.
- The control signal $\xi = (V, \Omega)$ is again a camera body velocity specified w.r.t. the camera frame, but for IBVS it is computed directly using $s(t)$.

For example, if the feature is a single image point with image plane coordinates u and v, we have $s(t) = (u(t), v(t))$.

Since $\dot{e}(t) = \dot{s}(t)$, we'll need to know the relationship between $\dot{s}$ and ξ to design a controller that achieves the error dynamics $\dot{e} = -\lambda e$.

Imaging Geometry



Consider a point P with coordinates (x, y, z) w.r.t. the camera frame. Using perspective projection, P's image plane coordinates are given by

$$u = \lambda \frac{x}{z}, \quad v = \lambda \frac{y}{z}$$

in which λ is the camera focal length.

The Interaction Matrix (for a point feature)

As an example, consider the interaction matrix for a single point with coordinates x, y, z.

To determine the interaction matrix for a point,

1. Compute the time derivatives for u and v.
2. Express these in terms of u, v, $\dot{x}, \dot{y}$, and $\dot{z}$ and z.
3. Find expressions for $\dot{x}, \dot{y}$, and $\dot{z}$ in terms of ξ and x, y, z.
4. Combine equations and grind through the algebra.