EE6221 Robotics and Intelligent Sensors (Part 3)

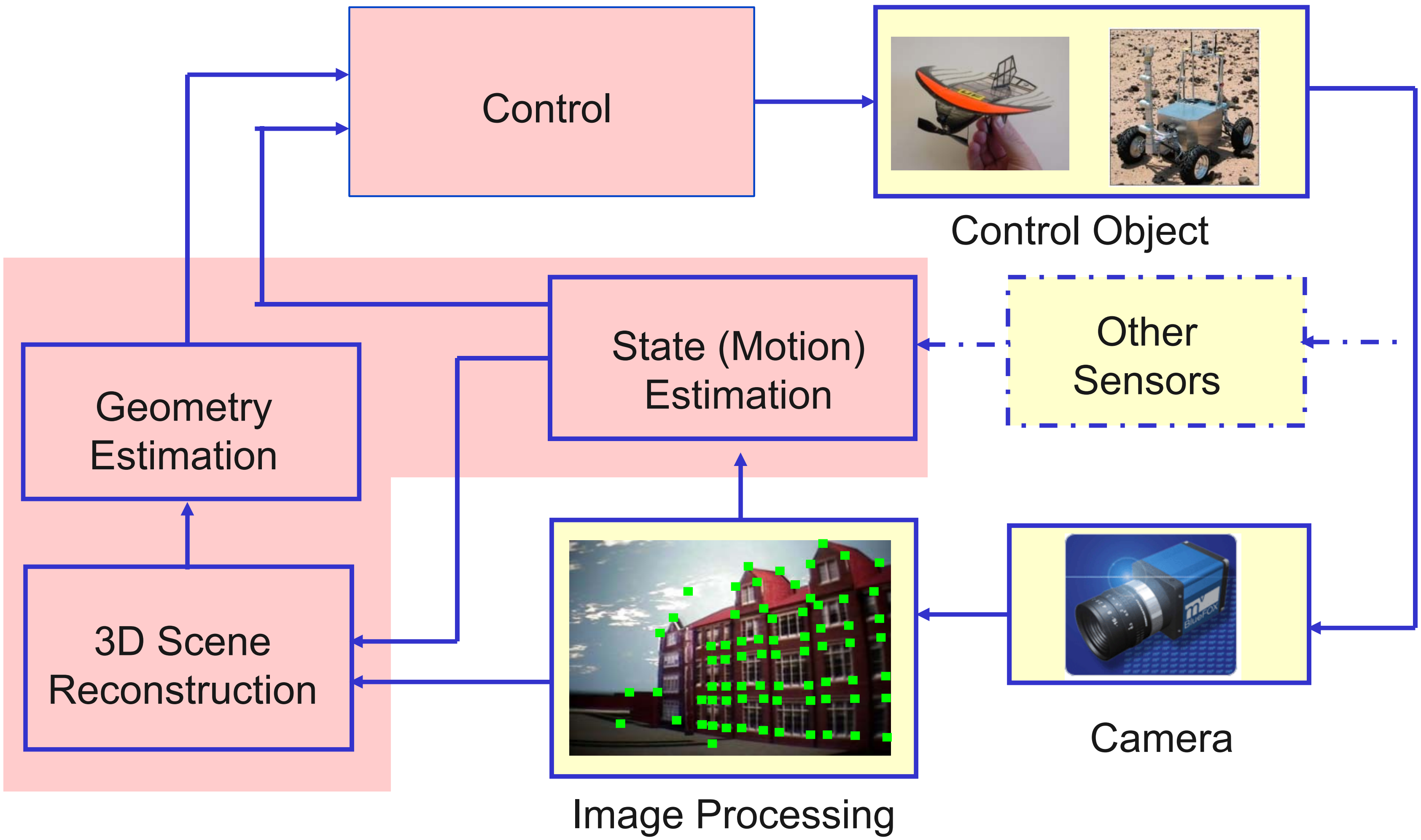
Lecture 5 & 6: Vision-Based Control & Estimation

Guoqiang Hu
School of EEE, NTU

Outline

- Visual servo control introduction
- Three visual servo control (visual servoing) methods
  - Position-based visual servo control
  - Image-based visual servo control
  - Homography-based visual servo control (also called 2.5D visual servo control)

Vision-Based Control & Estimation



Visual Servoing-Introduction

- Visual servo control (visual servoing) is the use vision/image data in the feedback control of a mechanical system
- Typically refers to control of robot manipulators or mobile robots
- There are two broad categories, Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS).
  - PBVS uses pose reconstruction methods to estimate a pose error between the current robot pose and a known goal pose.
  - IBVS regulates an error between the current image and a known goal image
  - IBVS and PBVS have strengths and weaknesses
- There are many approaches that build upon classic IBVS, PBVS or both to address unique problems, but there is no “silver-bullet” visual servoing method that is best for all situations

Visual Servoing-Introduction

- We will discuss basic PBVS and IBVS and contrast their performance.
- We will present a well known hybrid PBVS/IBVS method known as 2.5D VS that was designed to get the best of both.
- The methods presented here assume a camera mounted on a fully actuated 6DOF manipulator, the “eye-in-hand” scenario.
  - A camera mounted on a mobile robot is similar, but will likely have nonholonomic motion constraints to deal with.
  - A fixed camera viewing a manipulator or mobile robot can use PBVS or IBVS with a simple transformation from camera frame to robot frame.
- We assume the goal is a constant pose, the case of a time varying pose can be handled through standard trajectory tracking.
- We assume feature points can be extracted and tracked via feature detection and tracking methods.

Visual Servoing-Introduction

Free software resources for visual servoing

- Intel Open Source Computer Vision Library for C++
  - http://www.intel.com/technology/computing/opencv/
- Image Processing Toolbox for Matlab
  - Included in most full versions of Matlab
- Machine Vision Toolbox for Matlab
  - http://www.petercorke.com/Machine%20Vision%20Toolbox.html
- Robotics Toolbox for Matlab
  - http://www.petercorke.com/Machine%20Robotics%20Toolbox.html

Visual Servoing-Introduction

Reference reading (tutorial papers):

- S. Hutchinson, G. Hager and P. Corke, “A tutorial on visual servo control,” *IEEE Transactions on Robotics and Automation*, Oct 1996, pp. 651-670.
- F. Chaumette and S. Hutchinson, “Visual Servo Control Part I : Basic approaches,” IEEE Robotics and Automation Magazine, December 2005, pp. 82-90.
- F. Chaumette and S. Hutchinson, “Visual Servo Control Part II: Advanced Approaches,” IEEE Robotics and Automation Magazine, March 2006, pp 109-118.

Visual Servoing-History

- History of visual servoing is a bit murky
- PBVS dates at least to the late 70's when pose reconstruction methods were used to position a robot
  - Agin, "Real time control of a robot with a mobile camera," 1979
  - Birk et al., "Orienting Robot for Feeding Workpieces Stored in Bins," 1981
- IBVS dates from the late 80's
  - Weiss et al., "Dynamic sensor-based control of robots with visual feedback," 1987
  - Feddema and Mitchell, "Vision-guided servoing with feature based trajectory generation," 1989
  - Espiau et al., "A new approach to visual servoing in robotics," 1992.

Visual Servoing-History

- Hybrid visual servoing methods appeared in late 90’s
  - Malis et al., “2-1/2D visual servoing,” 1999
  - Corke and Hutchinson, “A new partitioned approach to image-based visual servo control,” 2001
- Nonlinear visual servo control methods appear in the 00’s
  - Chen et al., “Adaptive homography-based visual servo tracking for a fixed camera configuration with a camera-in-hand extension,” 2005
  - Hu et al., “Adaptive Homography-Based Visual Servo Tracking Control Via A Quaternion Formulation,” 2010.
- Uncalibrated visual servoing methods in late 00’s
  - Hu et al., “Quaternion-Based Visual Servo Control in the Presence of Camera Calibration Error,” 2010.
  - Hu et al., “Homography-Based Visual Servo Control with Imperfect Camera Calibration,” 2009.

Pinhole Camera Model - Review

- 3D feature point with coordinates in camera frame

$$\overline{m} = [x, y, z]^T$$

- Projects to image point with coordinates in the camera frame

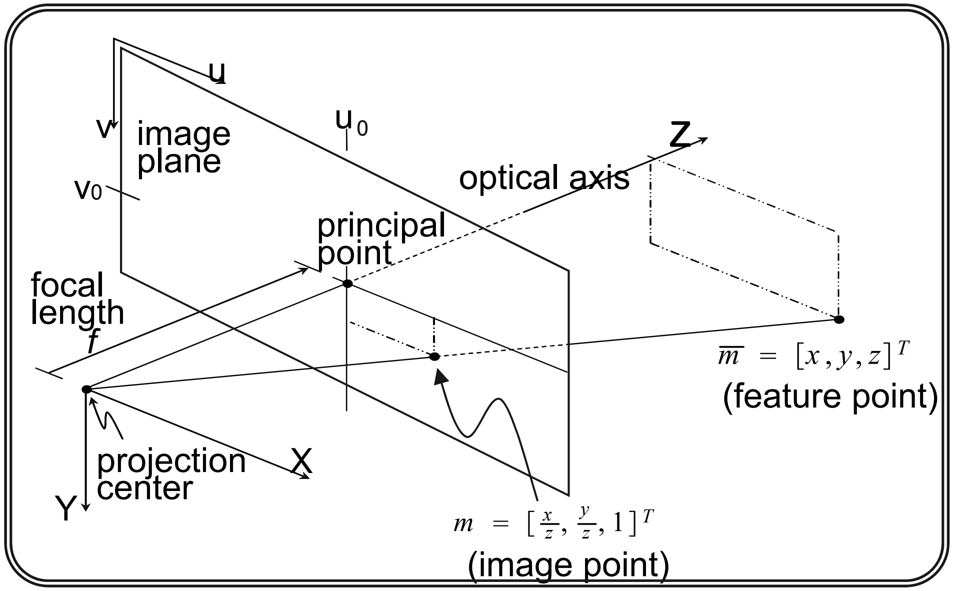
$$m = [m_x, m_y, 1]^T = \pi(\overline{m}) = \frac{\overline{m}}{z} = \left[x/z, y/z, 1 \right]^T$$

- Mapped to pixel coordinates by Calibration Matrix A

$$p = [u, v, 1]^T = Am$$

- Given p from digital image, recover m by

$$m = A^{-1} p$$



Imaging Background

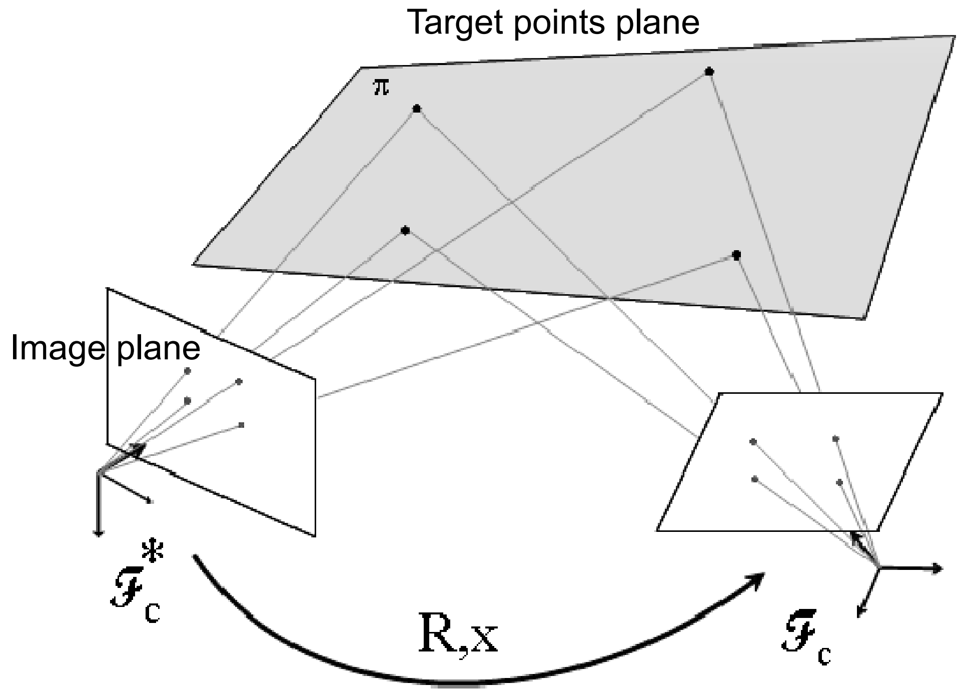
The Euclidean coordinates of the target points can be expressed in camera goal frame $\mathcal{F}_c^*$ and current camera frame $\mathcal{F}_c$

$$\bar{m}_j(t) = [x_j(t), y_j(t), z_j(t)]^T$$

$$\bar{m}_j^* = [x_j^*, y_j^*, z_j^*]^T$$

$$\underbrace{m_j(t) = [\frac{x_j(t)}{z_j(t)}, \frac{y_j(t)}{z_j(t)}, 1]^T \qquad m_j^* = [\frac{x_j^*}{z_j^*}, \frac{y_j^*}{z_j^*}, 1]^T}_{\text{Normalized Coordinates}}$$

$$\underbrace{p_j^* = Am_j^* = [u^*, v^*, 1]^T \qquad p_j = Am_j = [u, v, 1]^T \quad A \in \mathbb{R}^{3\times 3}}_{\text{Pixel Coordinates}}$$



The image points are formally related by the Rotation R and translation x between camera poses as

$$z_j m_j = z_j^* R m_j^* + x, \ \forall j \in \{1 \ldots N\}$$

Structure from Motion Review

- Noncoplanar points are related by the Essential Matrix E
- Given eight noncoplanar points we can solve for E, R and λx

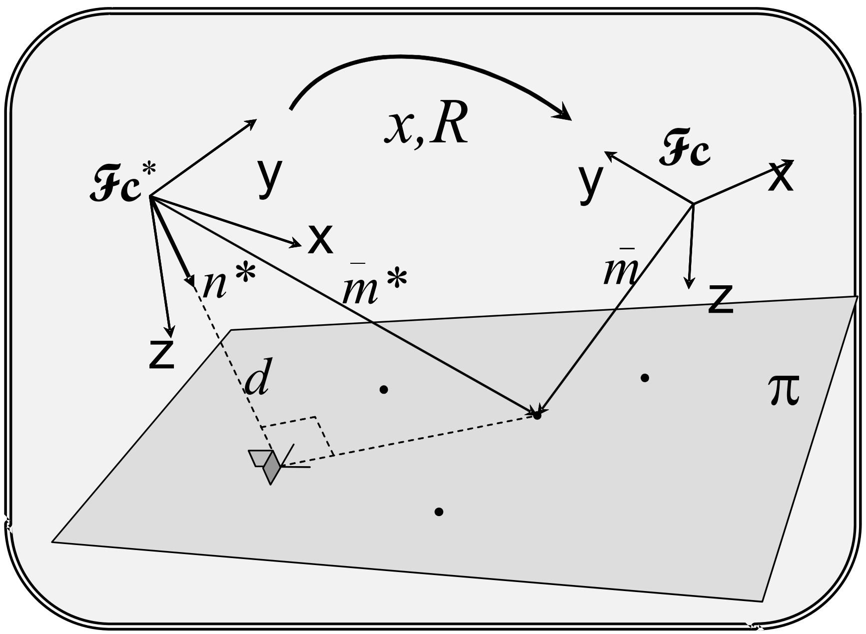
$$m_j^T [x]_\times R m_j^* = 0$$

$$m_j^T E m_j^* = 0$$

- Coplanar points are related by the Euclidean Homography Matrix H and depth ratios α_j
- Given four coplanar points we can solve for H, α_j, R and x/d^*

$$m_j = \frac{z_j^*}{z_j}\left(R + \frac{x}{d^*} n^{*T}\right) m_j^*$$

$$= \alpha_j H m_j^*$$



Camera Kinematics - Background

- Assume camera is fully actuated, 6 DOF, i.e. can move and rotate in any direction
- Rotation matrices are cumbersome in control
  - Not a vector space
  - Not a minimal representation (9 elements to represent 3 angles
- Can locally map rotation matrix to three elements
  - Euler angles
  - Roll, pitch, yaw angles
  - *Angle/axis
  - *Unit quaternions (4 elements)

Camera Kinematics - Background

- Angle/axis representation of Rotation R
  - Rotation of θ about 3D axis u

$$R \rightarrow u\theta$$

$$u\theta = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \theta \qquad u = \frac{1}{2\sin(|\theta|)} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

$$\theta = \operatorname{acos}\left(\tfrac{1}{2}(Tr(R) - 1)\right)$$

$$u\theta \rightarrow R$$

$$R = e^{[u]_\times \theta} = I_3 + [u]_\times \sin(\theta) + [u]_\times^2 (1 - \cos(\theta))$$

$$[u]_\times \theta = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \theta$$

Camera Kinematics - Background

- Camera pose can then be represented as a 6D vector

$$e_p(t) = [x(t)^T, u(t)^T\theta(t)]^T$$

$x(t) \in \mathbb{R}^3$ translation

$u(t)\theta(t) \in \mathbb{R}^3$ angle/axis rotation

- Camera velocity is also represented as a 6D velocity vector

$$\xi(t) = [v(t)^T, \omega^T(t)]^T \in \mathbb{R}^6$$

$v(t) \in \mathbb{R}^3$ linear velocity

$\omega(t) \in \mathbb{R}^3$ angular velocity

System Stability - Background

- Consider a system $x(t) \in \mathbb{R}^n$ with time derivative $\dot{x}(t)$
- Suppose there exists a function $V(x)$ such that

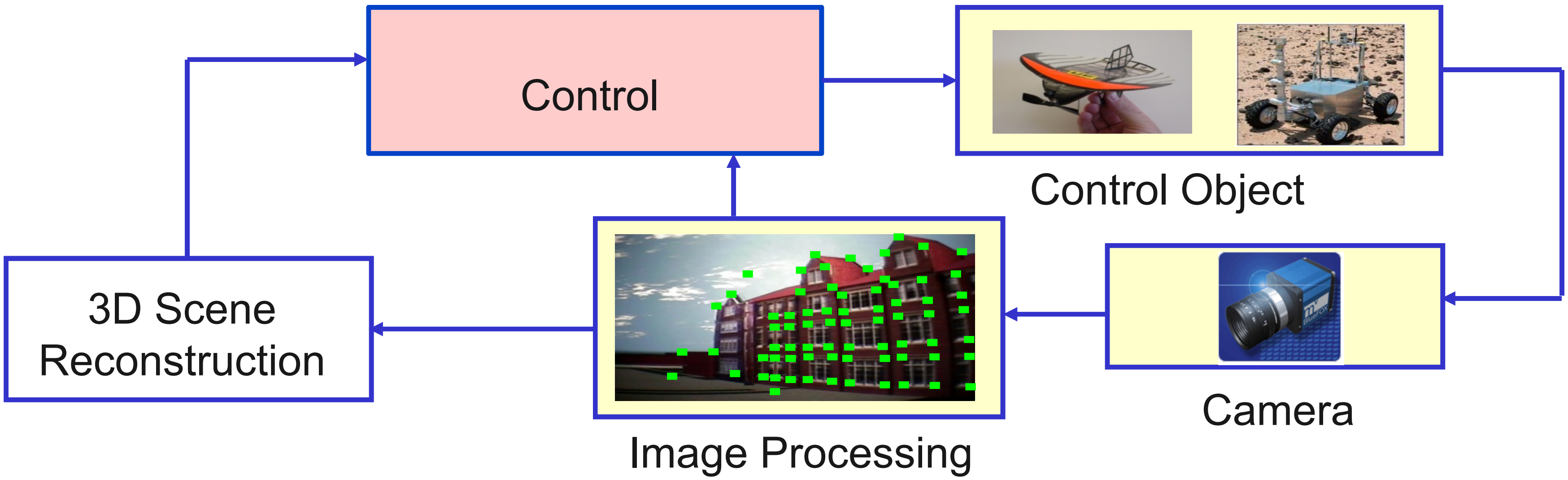
$$V(0) = 0 \; V(x) > 0 \quad \forall x$$ in some neighborhood of $x=0$

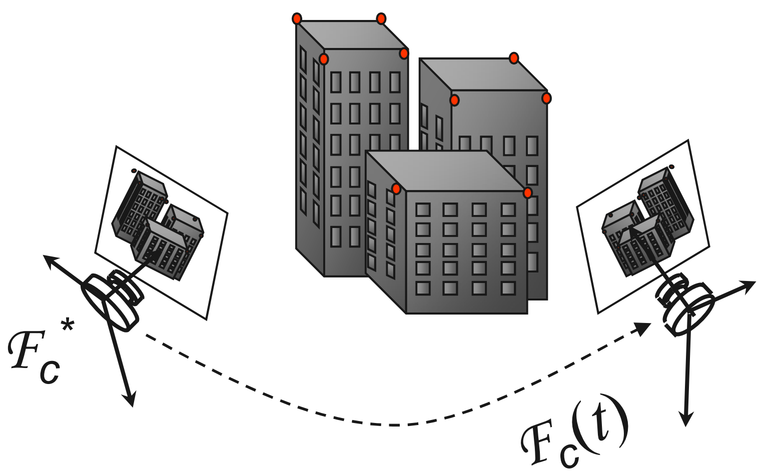
$$\dot{V}(t) = \frac{\partial V}{\partial x}\dot{x}(t) \leq 0$$

Then the system is Stable (i.e. bounded) in that neighborhood

- If $\dot{V}(t) < 0$, then the system is Asymptotically Stable (i.e. bounded and x $\rightarrow$0 as $t \rightarrow \infty$
- If the neighborhood is $\mathbb{R}^n$, then the system is Globally (Asymptotically) Stable

Visual Servo Control Problem Description





- A camera is mounted on a control object (e.g. a robot manipulator or vehicle)
- The camera is at a current pose $\mathcal{F}_c(t)$
- There exists a goal pose $\mathcal{F}_c^*$
- The task of visual servoing is to move the camera such that $\mathcal{F}_c(t) \rightarrow \mathcal{F}_c^*$, as $t \rightarrow \infty$
- In some cases $\mathcal{F}_c^*$ is known a priori, in some cases it is unknown, but we have an image taken from $\mathcal{F}_c^*$, (i.e. a goal image)

Visual Servoing Example

Method 1: PBVS
Position Based Visual Servoing

Position Based Visual Servoing

- Error is defined as a pose in Euclidean space
  - Camera acts as a "Cartesian sensor" to estimate pose error

- Define pose error signal as

$$e_p(t) = [x(t)^T, u(t)^T\theta(t)]^T \qquad \begin{array}{l} x(t) \in \mathbb{R}^3 \\ u(t)\theta(t) \in \mathbb{R}^3 \end{array}$$

- Open loop error dynamics given as a function of camera velocity

$$\dot{e}_p = L_p\xi \qquad \xi(t) = [v(t)^T, \omega^T(t)]^T \in \mathbb{R}^6$$

Position Based Visual Servoing

$$L_p = \begin{bmatrix} R_{vc} & 0_{3\times3} \\ 0_{3\times3} & R_{vc}L_\omega \end{bmatrix},$$

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

$$\sin c(\theta) = \frac{\sin(\theta)}{\theta}$$

$R_{vc}(t)$ is the rotation matrix from the frame in which ξ(t) is measured to the camera frame.

$R_{vc}(t)$ is an identity matrix if the camera frame and input velocity frame are the same.

Position Based Visual Servoing

- Stabilizing proportional feedback

$$\xi = -k_p L_p^{-1} e_p$$

k_p is pos scalar gain

$$L_p^{-1} = \begin{bmatrix} R_{vc}^T & 0_{3\times3} \\ 0_{3\times3} & L_\omega^{-1} R_{vc}^T \end{bmatrix}, L_\omega^{-1} = I + \frac{\theta}{2}\,\mathrm{sinc}^2\left(\frac{\theta}{2}\right) u_\times + (1 - \mathrm{sinc}(\theta)) u_\times^2$$

$$L_\omega^{-1} u\theta = L_\omega u\theta = u\theta$$

- Closed loop error dynamics given by

$$\begin{aligned} \dot{e}_p &= L_p \xi \\ &= L_p(-k_p L_p^{-1} ep) \\ &= -k_p e_p \end{aligned}$$

Position Based Visual Servoing

To prove stability, define Lyapunov function

$$\begin{aligned} V_p(e_p) &= \frac{1}{2} e_p^T e_p, \\ &= \frac{1}{2} \| e_p(t) \|^2 \end{aligned}$$

V_p is pos def

With time derivative

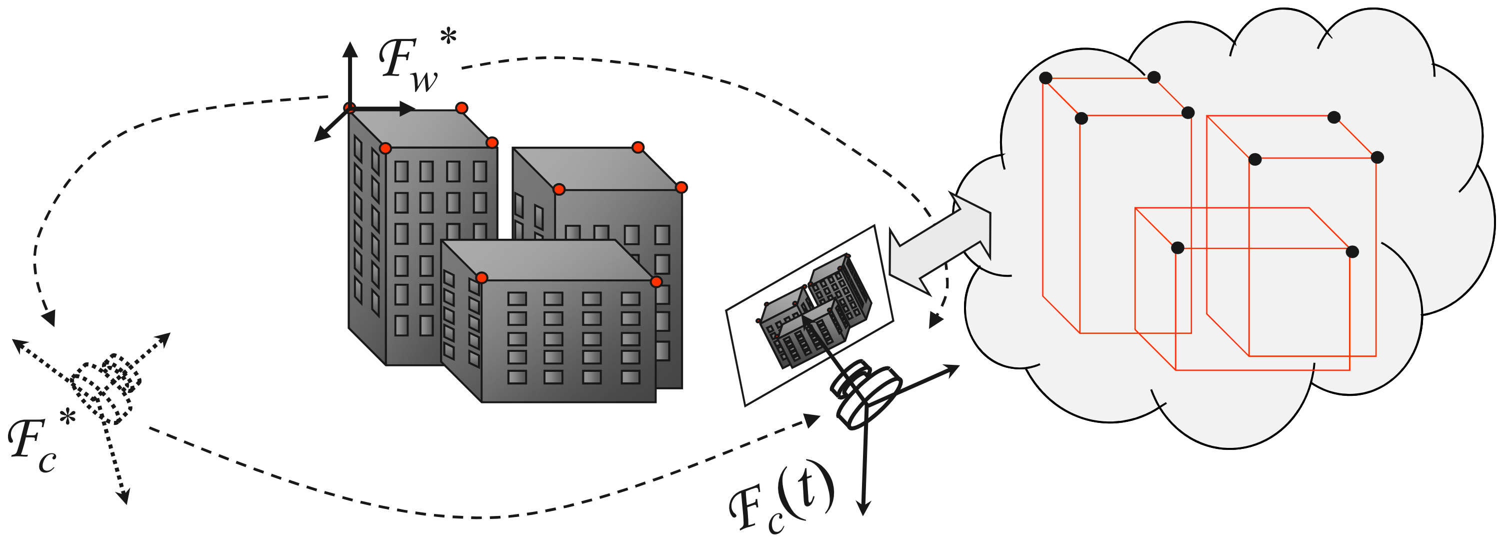
$$\begin{aligned} \dot{V}_p &= e_p^T \dot{e}_p \\ &= e_p^T(-k_p e_p) \\ &= -k_p \| e_p \|^2 \end{aligned}$$

$\dot{V}_p$ is neg def

Negative definite Lyapunov function means the controller is Globally Asymptotically Stable and $e_p \to 0$ as $t \to \infty$

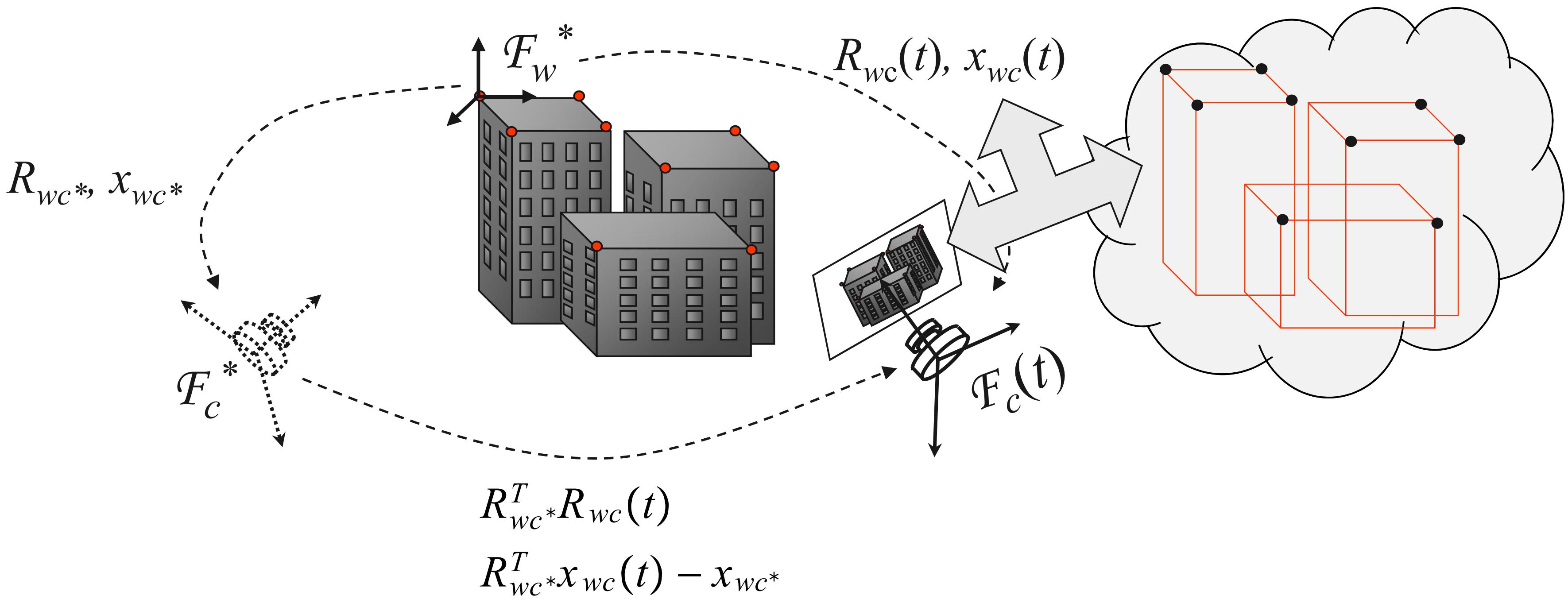
Pose Estimation for PBVS

- How to solve for the pose error?
- Model based methods require geometric knowledge of the target/scene (such as from a CAD model) and give pose of the camera relative to a frame attached to the object $\mathcal{F}_w^*$
  - D. DeMenthon and L. Davis, "Model-Based Object Pose in 25 Lines of Code." Int. Journal Computer Vision, vol 15, 1995, pp 123-141.
  - L. Quan and Z. Lan, "Linear N-Point Camera Pose Determination." IEEE Trans. Pattern Analysis and Machine Intelligence, vol 21,1999, pp 774-780.



Pose Estimation for PBVS

- Given known desired pose R_{wc*}, x_{wc*}, of camera $\mathcal{F}_c^{\,*}$ with respect to $\mathcal{F}_w^{\,*}$
- Solve for current pose $R(t)_{wc}$, $x(t)_{wc}$, of camera $\mathcal{F}_c(t)$ with respect to $\mathcal{F}_w^{\,*}$
- Solve for current pose $R(t) = R_{wc^*}^T R_{wc}(t)$ $\quad x(t) = R_{wc^*}^T x_{wc}(t) - x_{wc^*}$
  of current camera $\mathcal{F}_c(t)$ with respect to desired pose $\mathcal{F}_c^{\,*}$



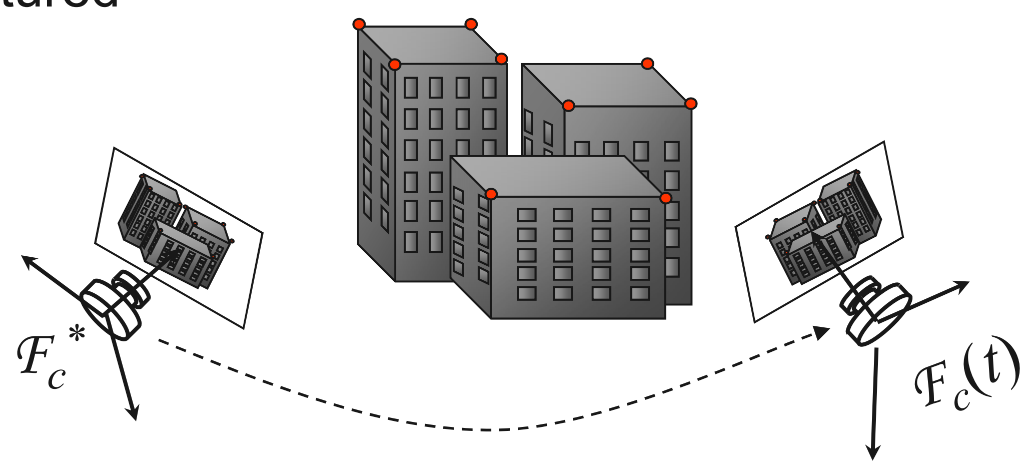
Pose Estimation for PBVS

- If a geometric model is not available, but a goal image is, Homography and Epipolar methods can work

$$m_j^T(t)[x(t)]_\times R(t)m_j^* = 0 \qquad m_j(t) = \frac{z_j^*}{z_j(t)}\left(R(t) + \left(\frac{x(t)}{d^*}\right)n^{*T}\right)m_j^*$$

$$m_j^T(t)E(t)m_j^* = 0 \qquad m_j(t) = \alpha_j(t)H(t)m_j^*$$

- Pose error defined with respect to the pose where a goal image was captured



Pose Estimation for PBVS

$$m_j^T(t)[x(t)]_\times R(t)m_j^* = 0 \qquad m_j(t) = \frac{z_j^*}{z_j(t)}\left(R(t) + \left(\frac{x(t)}{d^*}\right)n^{*T}\right)m_j^*$$

$$m_j^T(t)E(t)m_j^* = 0 \qquad m_j(t) = \alpha_j(t)H(t)m_j^*$$

$$\Updownarrow \qquad\qquad \Updownarrow$$

$$R(t),\ \lambda x(t) \qquad\qquad R(t),\ \frac{x(t)}{d^*}$$

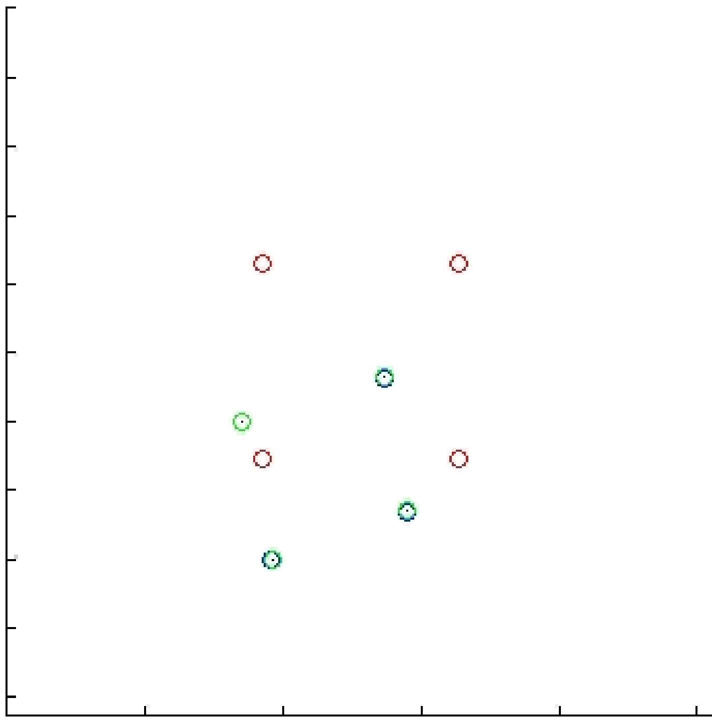
Translation only known up to scale factor, which can cause problems

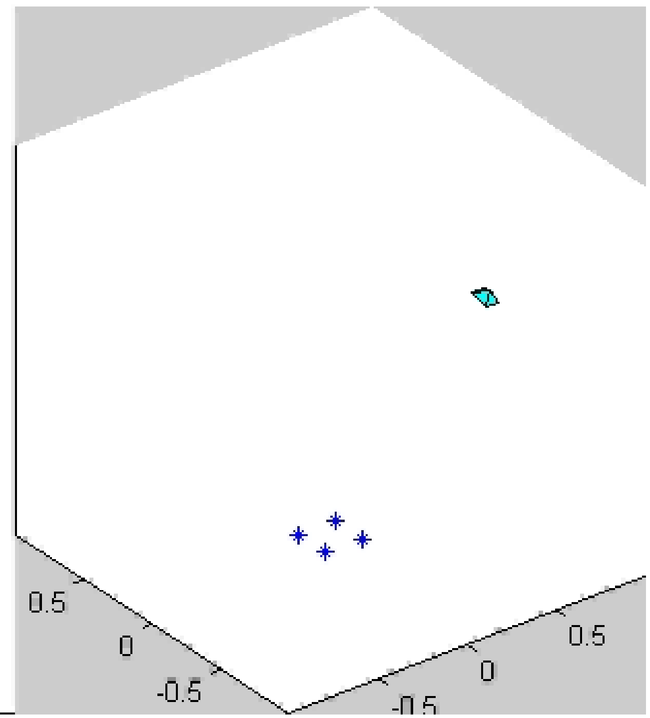
- For essential matrix decomposition, we solve for λx such that $\|\lambda x\|=1$, so velocity can't stabilize once translation error is reduced to $\|x\|<k_p$!
- Even if proper scale is known/measured, E solution breaks down as $x\rightarrow 0$!
- For Homography decomposition, we solve for x/d^*, where d^* is unknown positive constant. Unknown scale might slow down convergence, but does not affect stability.

Position Based Visual Servoing

Camera view:
Green – initial view of points
Blue – current view of points
Red – goal view of points

Birds eye view of camera
looking at feature points



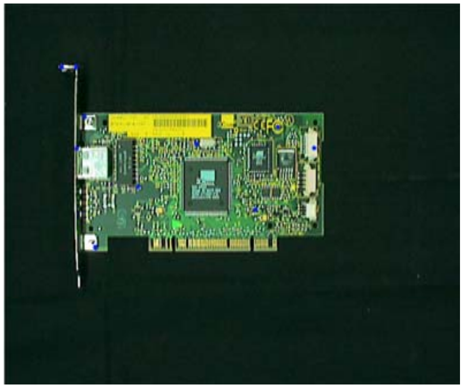


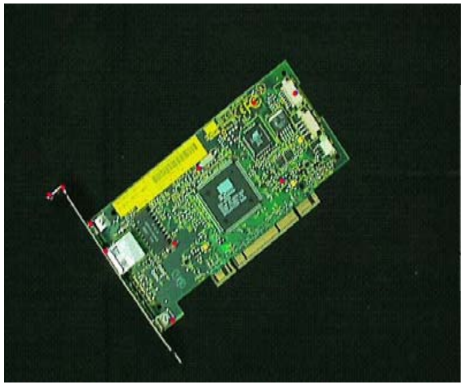
- PBVS – Estimate and control the pose of the control object
  - Pose error is exponentially stabilized
  - Feature points not controlled, may leave field of view

Method 2: IBVS
Image Based Visual Servoing

Image-Based Visual Servoing

- Error is defined in the image space
  - Measure the difference between the coordinates of several features in the current image and in a goal image
  - The camera is moved such that the features move towards their goal coordinates





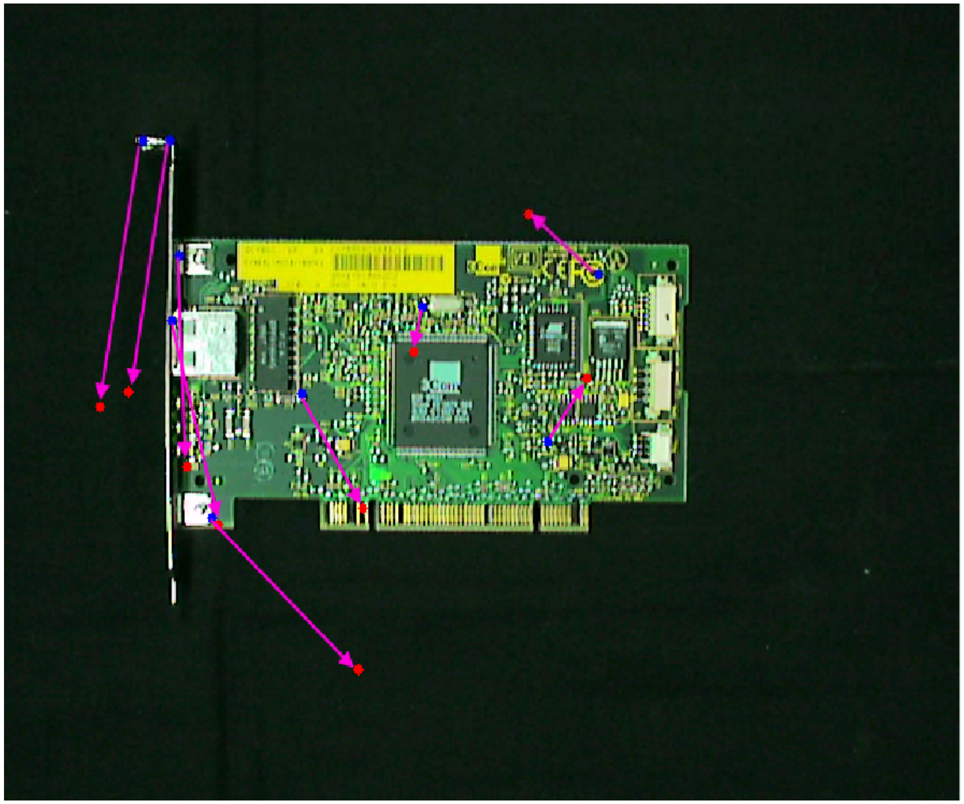


Image Based Visual Servoing

- Consider a stationary point viewed by a moving camera, the points has coordinates $m=[X(t),Y(t),Z(t)]^T$ in the camera frame $\mathcal{F}_c(t)$
- Derivative of coordinates are given as functions of camera velocity

$$\begin{aligned}\dot{X} &= -Z\omega_y + Y\omega_z - v_x \\ \dot{Y} &= -X\omega_z + Z\omega_x - v_y \\ \dot{Z} &= -Y\omega_x + X\omega_y - v_z\end{aligned} \quad \Rightarrow \quad \begin{aligned}\dot{X} &= Z\omega_y - m_y Z\omega_z + v_x \\ \dot{Y} &= m_x Z\omega_z - Z\omega_x + v_y \\ \dot{Z} &= m_y Z\omega_x - m_x Z\omega_y + v_z\end{aligned}$$

- Normalized image points given by $m=[m_x(t),m_y(t),1]^T$ with derivative

$$\begin{aligned}\dot{m}_x &= \frac{\dot{X}Z - \dot{Z}X}{Z^2} \\ \dot{m}_y &= \frac{\dot{Y}Z - \dot{Z}Y}{Z^2}\end{aligned} \quad \Rightarrow \quad \begin{aligned}\dot{m}_x &= -\frac{1}{Z}v_x + \frac{m_x}{Z}v_z + m_x m_y \omega_x - (1+m_x^2)\omega_y + m_y\omega_z \\ \dot{m}_y &= -\frac{1}{Z}v_y + \frac{m_y}{Z}v_z + (1+m_y^2)\omega_x - m_x m_y \omega_y - m_x\omega_z\end{aligned}$$

Image Based Visual Servoing

- Rewrite previous equation in matrix form

$$\dot{m}_x = -\frac{1}{Z}v_x + \frac{m_x}{Z}v_z + m_x m_y \omega_x - (1 + m_x^2)\omega_y + m_y \omega_z$$
$$\dot{m}_y = -\frac{1}{Z}v_y + \frac{m_y}{Z}v_z + (1 + m_y^2)\omega_x - m_x m_y \omega_y - m_x \omega_z$$

$$\begin{bmatrix} \dot{m}_x \\ \dot{m}_y \end{bmatrix} = \begin{bmatrix} \frac{1}{Z} & 0 & \frac{m_x}{Z} & m_x m_y & -(1 + m_x^2) & m_y \\ 0 & \frac{1}{Z} & \frac{m_y}{Z} & (1 + m_y^2) & -m_x m_y & m_x \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Needs depth estimate

$$\dot{m}_j = L_{ij}\xi$$

- L_{ij} is the Image Jacobian or Interaction Matrix for point j

Image Based Visual Servoing

- Stack point velocities and interaction matrices for all N points

$$\begin{bmatrix} \dot{m}_1 \\ \vdots \\ \dot{m}_N \end{bmatrix} = \begin{bmatrix} L_{i1} \\ \vdots \\ L_{iN} \end{bmatrix} \xi \quad \Rightarrow \quad \dot{m} = L_i \xi$$

- If N=3, L_i is 6x6 and can be inverted if full rank
  - L_i can be singular, depending on coordinates of feature points
- If N>3, L_i is $2N$x6 and we can take pseudo inverse

$$L_i^+ = (L_i^T L_i)^{-1} L_i^T$$

Image Based Visual Servoing

- Error defined in image space as the difference in current feature locations from goal feature locations

$$e_i(t) = \begin{bmatrix} m_1(t) \\ \vdots \\ m_N(t) \end{bmatrix} - \begin{bmatrix} m_1^* \\ \vdots \\ m_N^* \end{bmatrix}$$

- Open loop error dynamics given as a function of camera velocity

$$\dot{e}_i = L_i \xi$$

$$\xi(t) = \left[v(t)^T, \omega^T(t)\right]^T \in \mathbb{R}^6$$

Image Based Visual Servoing

- Stabilizing proportional feedback

$$\xi = -k_i L_i^{-1} e_i \qquad k_i \text{ is pos scalar gain}$$

- L_i^{-1} Solved numerically – no closed form inverse
- Can become singular, singularities are unforeseeable

- Closed loop error dynamics given by

$$\begin{aligned} \dot{e}_i &= -k_i L_i \xi \\ &= -k_i L_i L_i^{-1} e_i \\ &= -k_i e_i \end{aligned}$$

Image Based Visual Servoing

To prove stability, define Lyapunov function

$$V_i(t) = \frac{1}{2} e_i^T e_i$$

$$= \frac{1}{2} \| e_i(t) \|^2$$

V_i is pos def

With time derivative

$$\dot{V}_i = e_i^T \dot{e}_i$$

$$= e_i^T(-k_i e_i)$$

$$= -k_i \| e_i \|^2$$

$\dot{V}_i$ is neg def

Negative definite Lyapunov function means the controller is
Locally Asymptotically Stable and $e_i \rightarrow 0$ as $t \rightarrow \infty$

Image Based Visual Servoing

Why Locally Asymptotically Stable?

- Since Interaction Matrix L_i can become singular, its inverse is not always defined
- If L_i is nonsingular at the goal $m(t)=m^*$, then it will be nonsingular in a neighborhood of the goal and IBVS is AS in this neighborhood
- There is no known way to determine the size of the neighborhood, IBVS works well most of the time, but singularities are a problem and unpredictable

Image Based Visual Servoing

- If N>3, we take the psuedo inverse of L_i which generally exists even if the inverse for three points does not.
- Note that $L_i L_i^+ \neq I$ and is positive semidefinite
- The derivative of Lyapunov function becomes

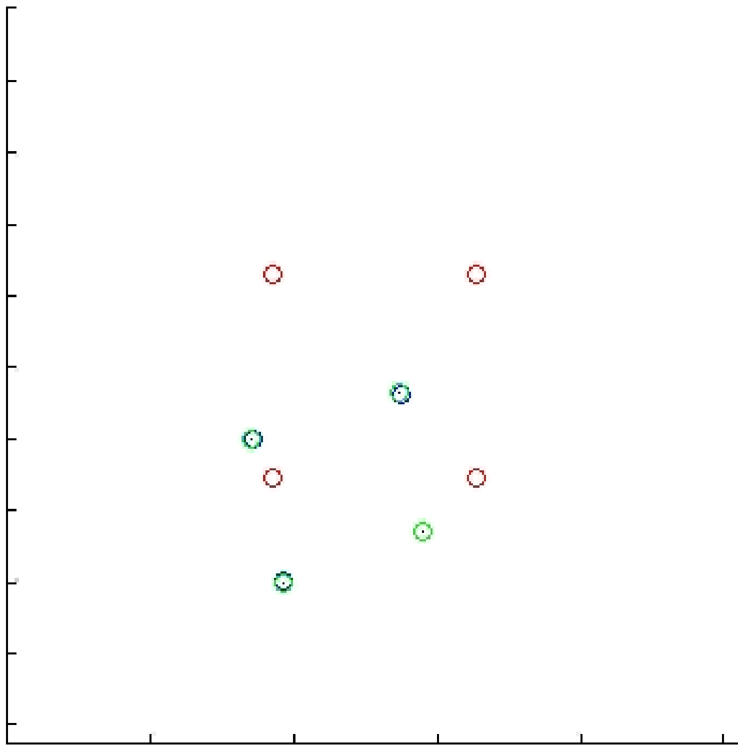
$$\dot{V}_i = -k_i e_i^T L_i L_i^+ e_i \qquad \dot{V}_i \text{ is negative semidefinite}$$

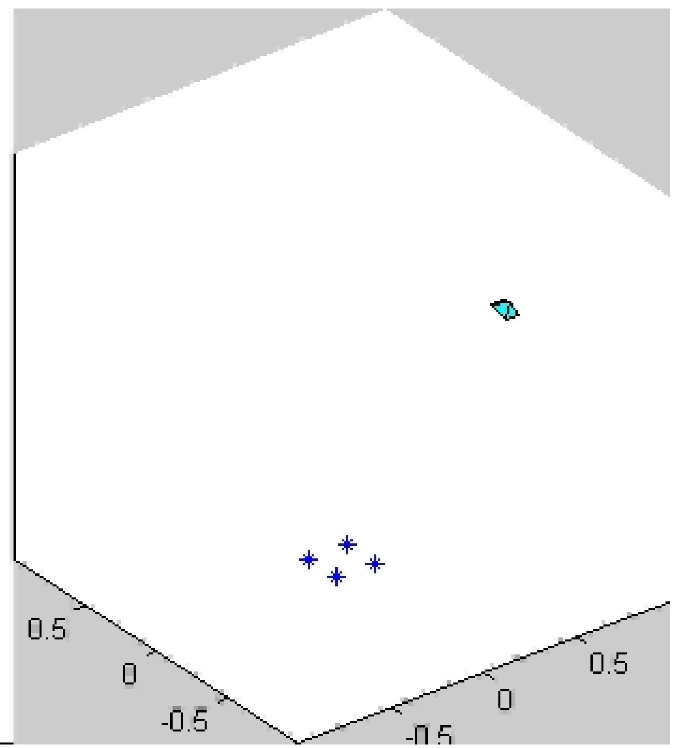
- The matrix $L_i L_i^+$ can have nullspace depending on the coordinates of the feature points
- e_i will not be in the null space of $L_i L_i^+$ in a neighborhood of the goal
- The controller is Locally Asymptotically Stable and $e_i \rightarrow 0$ as $t \rightarrow \infty$, and globally stable

Image Based Visual Servoing

Camera view:
Green – initial view of points
Blue – current view of points
Red – goal view of points

Birds eye view of camera looking at feature points



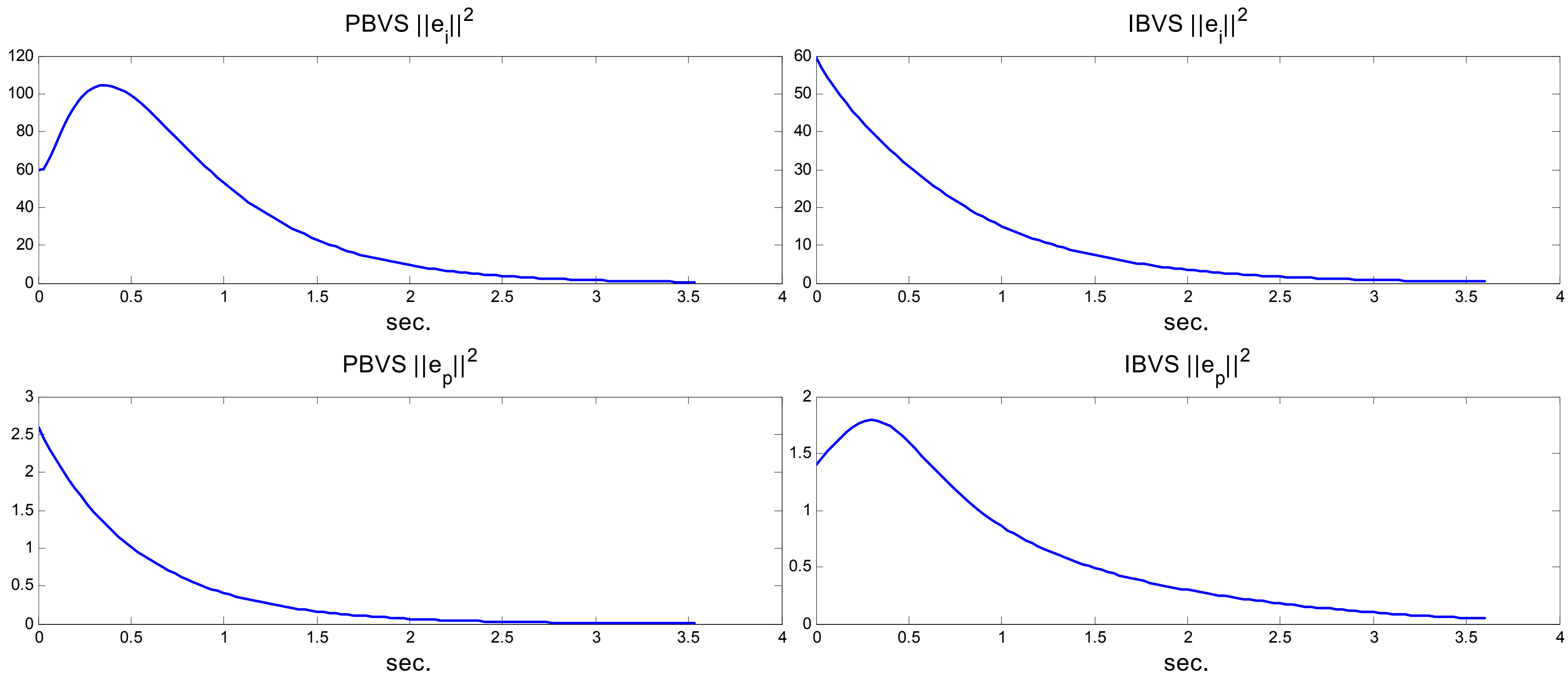


- IBVS – Control the location of image features
  - Image error is exponentially stabilized
  - Pose not controlled, large camera motions may occur. The control object (e.g., robot manipulator) can leave task space or reach joint limits

PBVS vs IBVS

- PBVS is Globally Asymptotically Stable (GAS) w.r.t. the Cartesian pose error, but no control over the image features
  - GAS is obviously desirable
  - Pose error strictly decreases, the robot follows least distance path to the goal
  - Requires an accurate target model or a goal image for Essential/Homography methods
  - Image features may leave the image
- IBVS is Locally Asymptotically Stable (LAS) w.r.t. the image feature error
  - Neighborhood of AS is unknown, seems large
  - Image error strictly decreases, the feature follow least distance path to their goal coordinates in the image plane
  - Requires a goal image
  - Requires accurate depth estimates for proper performance
  - Pose error can increase without bound

PBVS vs IBVS



PBVS $||e_i||^2$
IBVS $||e_i||^2$
PBVS $||e_p||^2$
IBVS $||e_p||^2$
120
100
80
60
40
20
0
0.5
1
1.5
2
2.5
3
3.5
4
sec.
50
30
10

PBVS vs IBVS

- Numerous attempts to address these issues
- Partitioned methods – use PBVS methods to control some camera DOF and IBVS methods to control the remaining
  - Mitigates problems of IBVS and PBVS, but cannot guarantee simultaneous asymptotic stability of both entire pose error and entire image error
- Switching methods – Switch between IBVS and PBVS when errors become too large
  - Guaranteed error bounds, but loses asymptotic stability
  - Require some knowledge of task
- Potential Functions – Cause features to follow trajectories that account for pose constraints
  - Asymptotically stable image error, bounded pose error
  - Requires much a priori knowledge