UTS:CAS
CENTRE FOR AUTONOMOUS SYSTEMS

41014 Sensors and Control for Mechatronic Systems

Lecture-8: Visual Servoing

Dr. **Liang Zhao**

Centre for Autonomous Systems
University of Technology,
Sydney

UNIVERSITY OF TECHNOLOGY SYDNEY

Visual Servoing

- Discussion
  - How to control the Fetch robot to pick up a box?

Visual Servoing

- Activity-1:
    - How to control the Fetch robot to pick up a box?

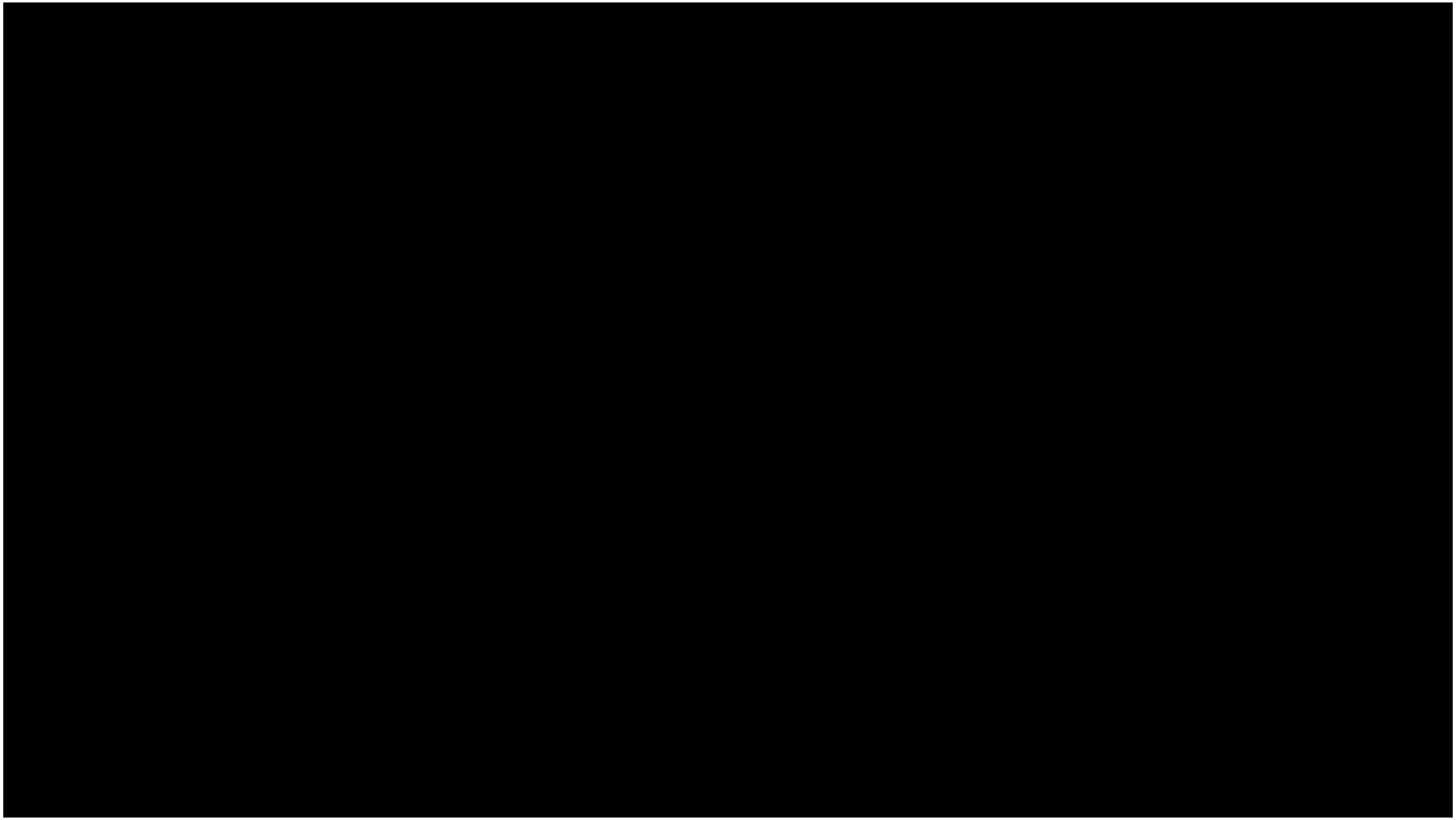
Visual Servoing

- Activity-1:
  - How to control the Fetch robot to pick up a box?

- Method
  - Other information
  - Camera
  - Detect the box
  - Hand: where to go
  - ……
  - Error?

Visual Servoing: Example

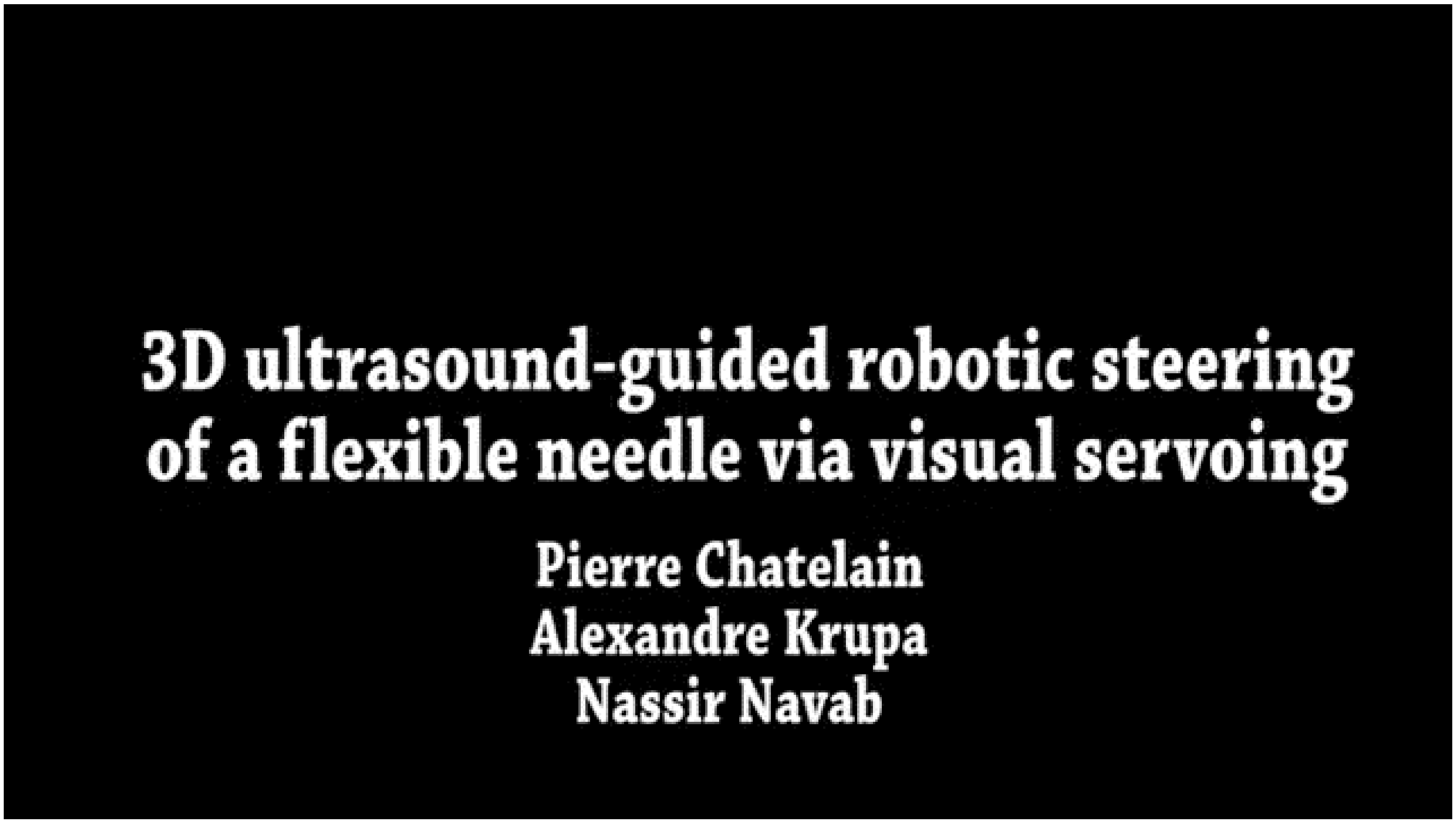
Robotic Arm: Visual Servoing (Georgia Tech)



https://www.youtube.com/watch?v=nLq9xbTuBpI

Visual Servoing: Example

Ultrasound-guided robotic steering of a needle



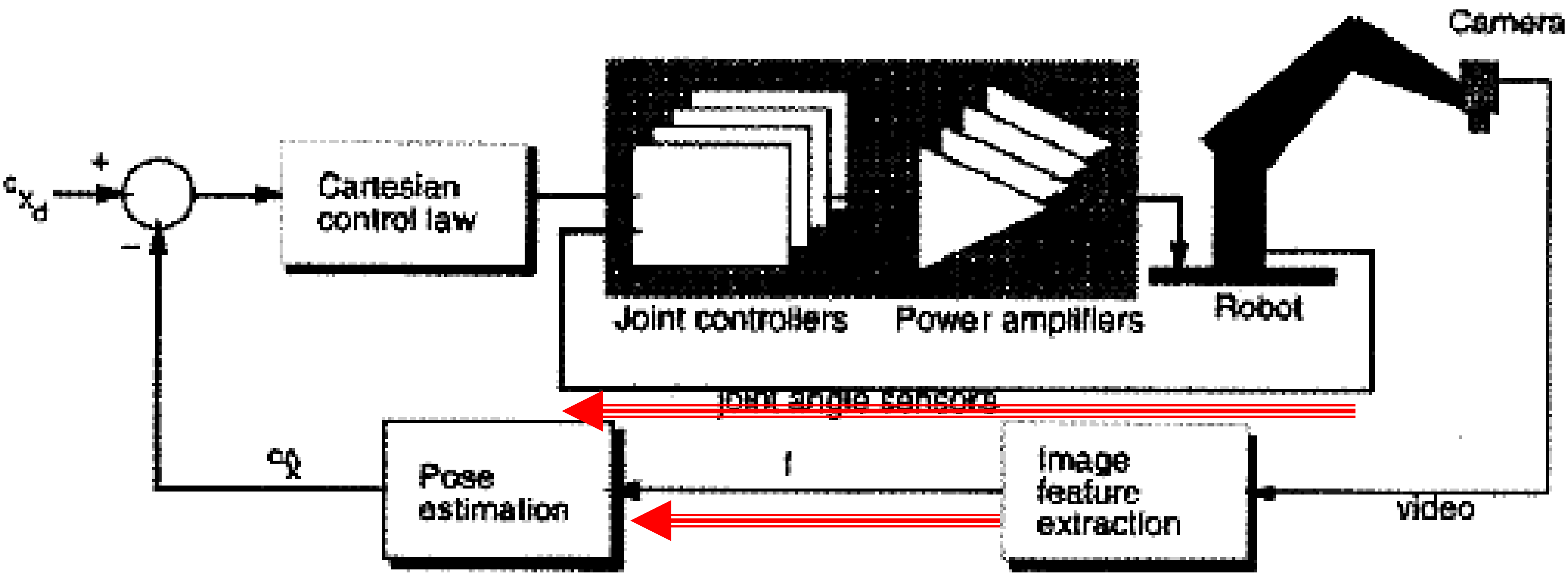
https://www.youtube.com/watch?v=8IyknL44n5s

UNIVERSITY OF TECHNOLOGY SYDNEY

- What is Visual Servoing?

Visual Servoing

- Vision System operates in a closed control loop.
- Better Accuracy than “Look and Move“ systems



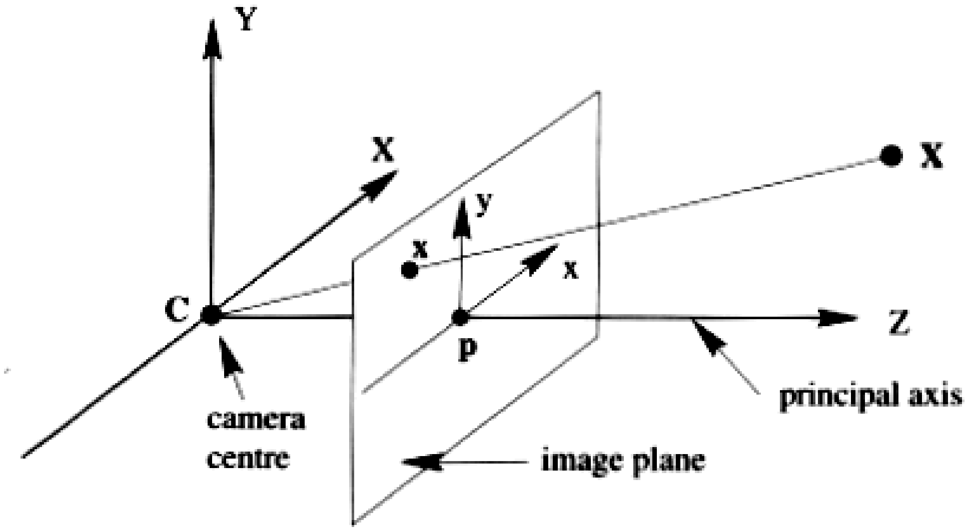
Figures from S.Hutchinson: A Tutorial on Visual Servo Control

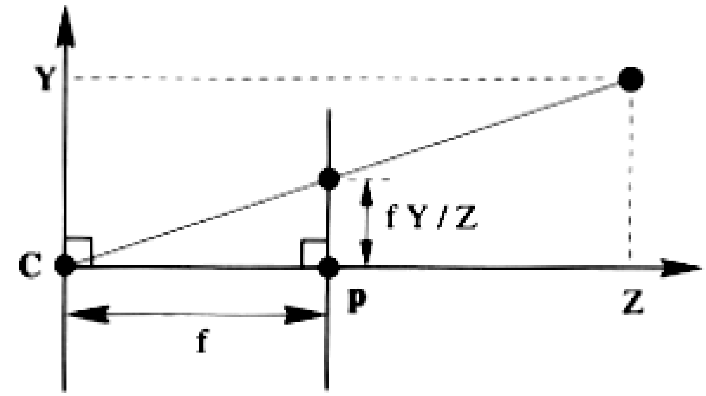
UNIVERSITY OF TECHNOLOGY SYDNEY

Vision Sensors

- Single Perspective Camera
- Multiple Perspective Cameras (e.g. Stereo Camera Pair)
- Laser Scanner
- Omnidirectional Camera
- Structured Light Sensor

- Single Perspective Camera





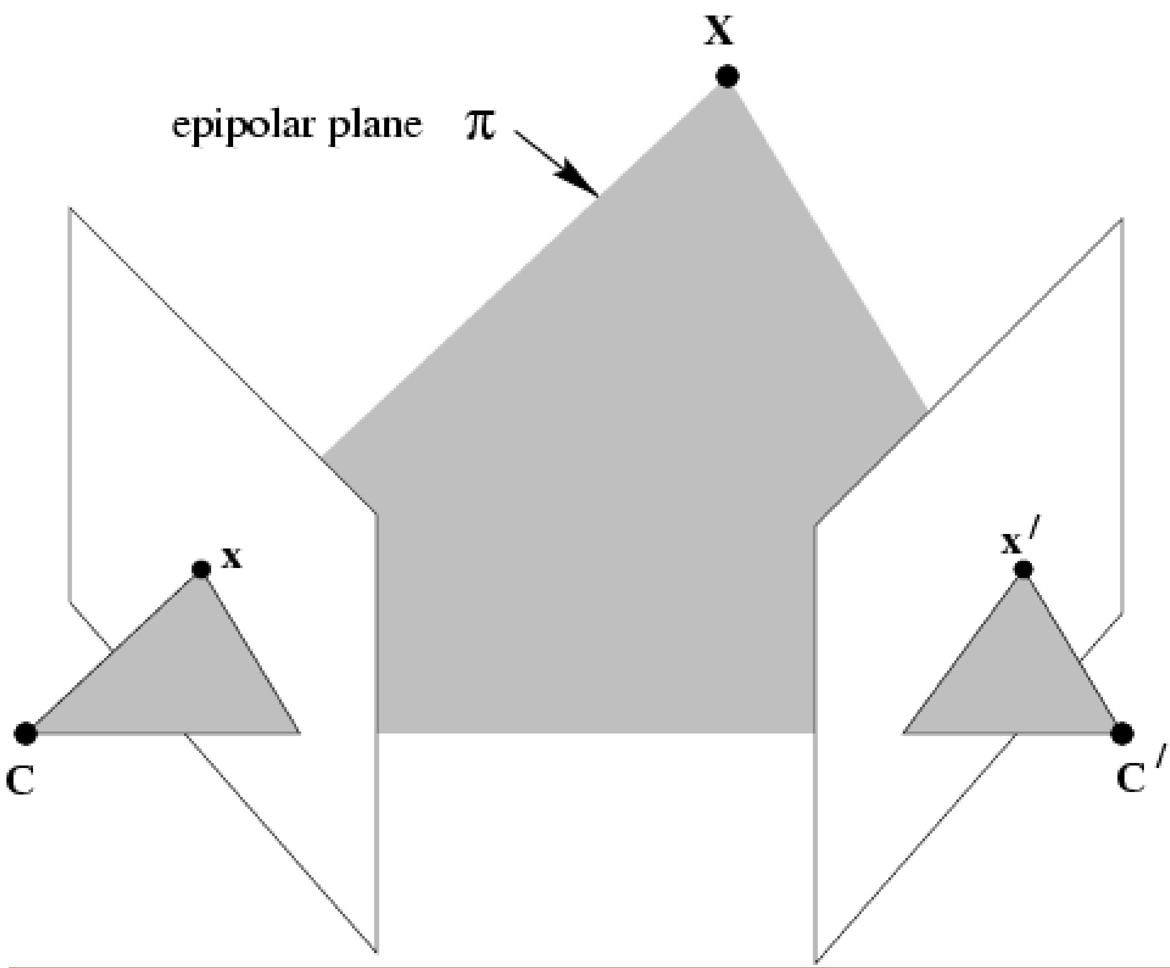
$$x = P_{3x4}X$$

Single projection

Vision Sensors

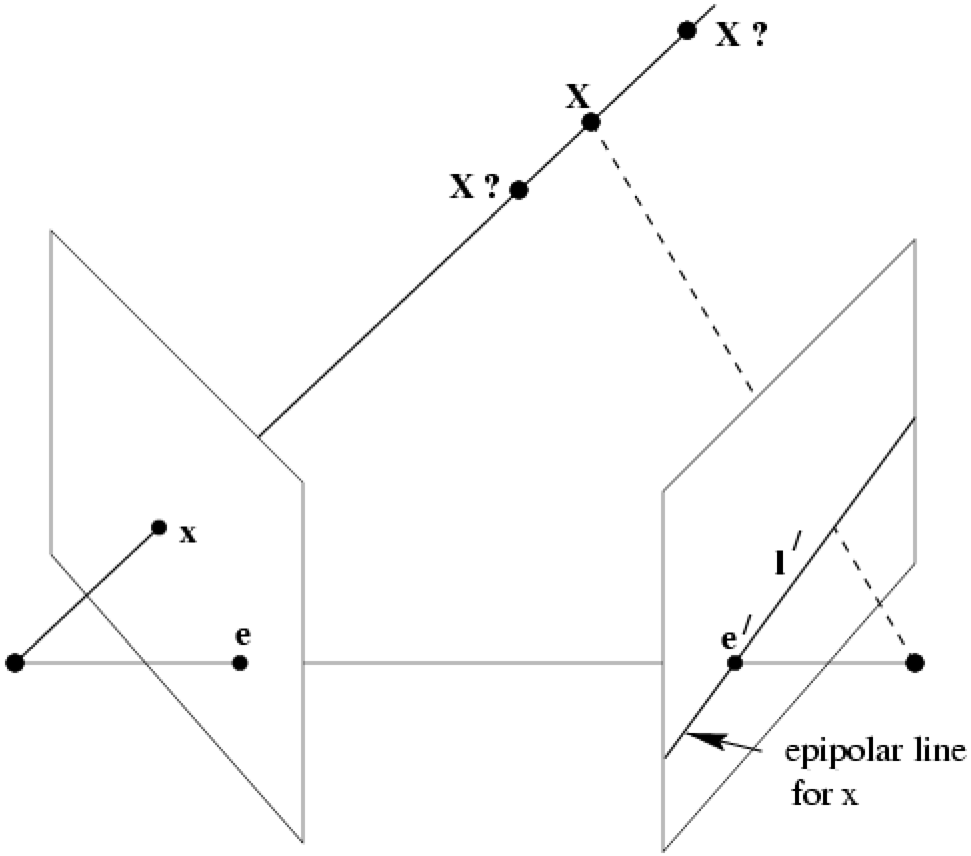
UNIVERSITY OF TECHNOLOGY SYDNEY

- **Multiple Perspective Cameras** (e.g. Stereo Camera Pair)



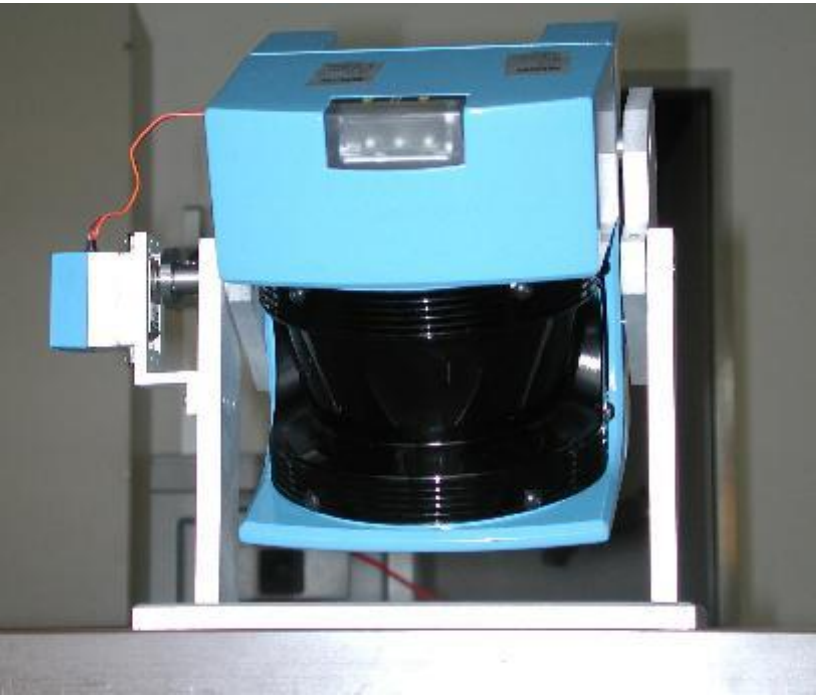
Vision Sensors

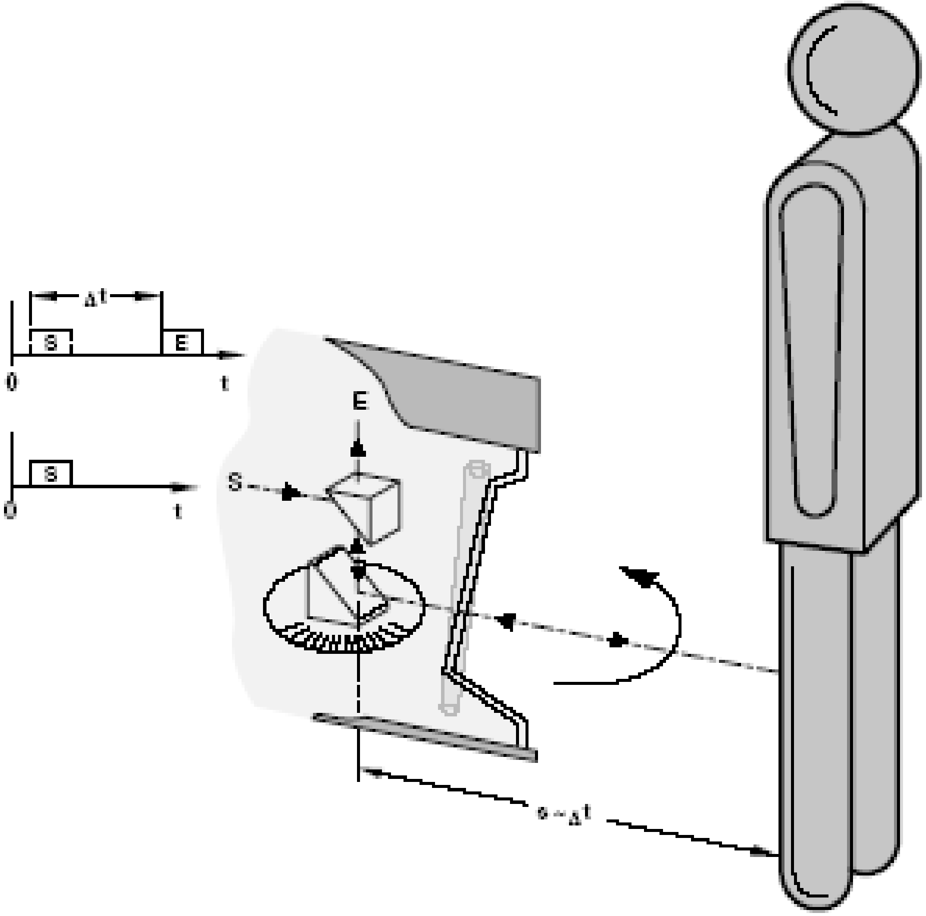
- Multiple Perspective Cameras (e.g. Stereo Camera Pair)



Vision Sensors

- Laser Scanner

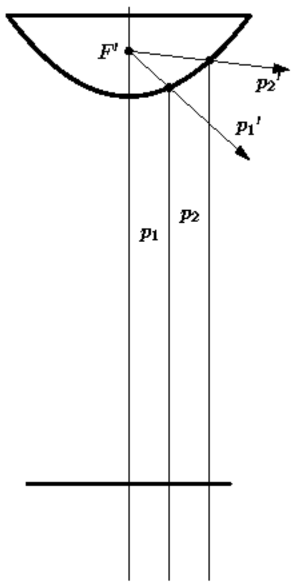


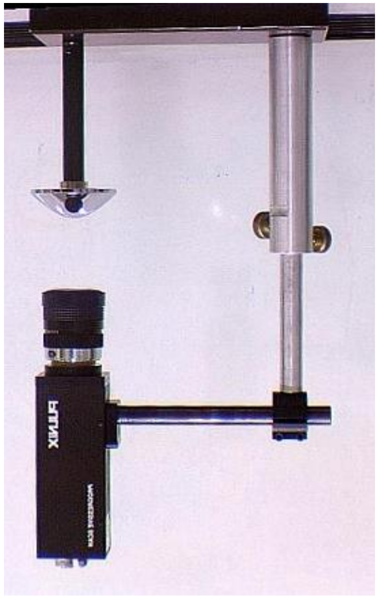


Vision Sensors

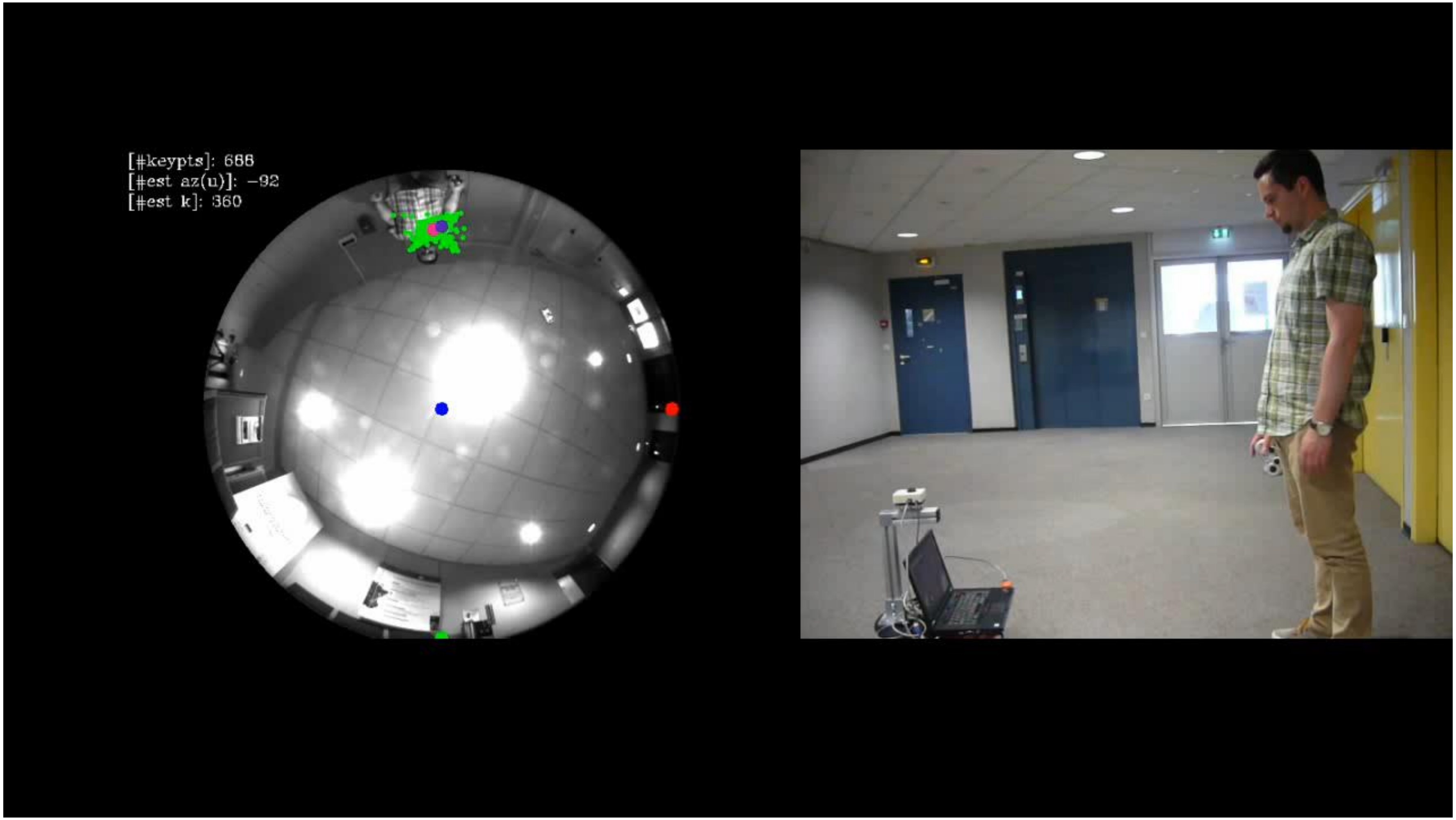
- Laser Scanner

- Omnidirectional Camera





Vision Sensors

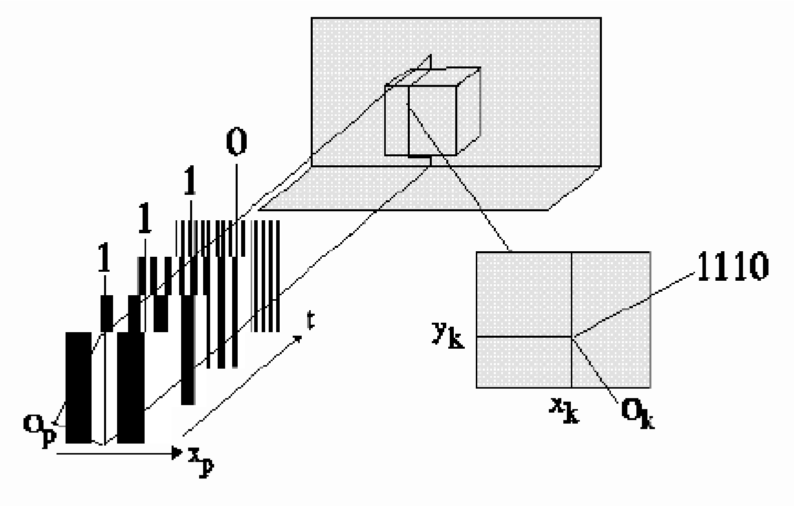


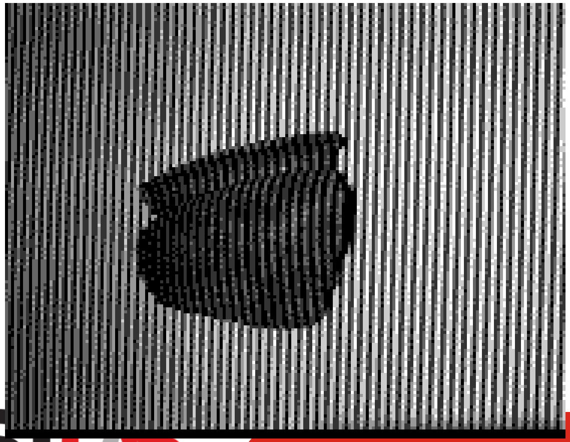
https://www.youtube.com/watch?v=ndccTfSh9JY

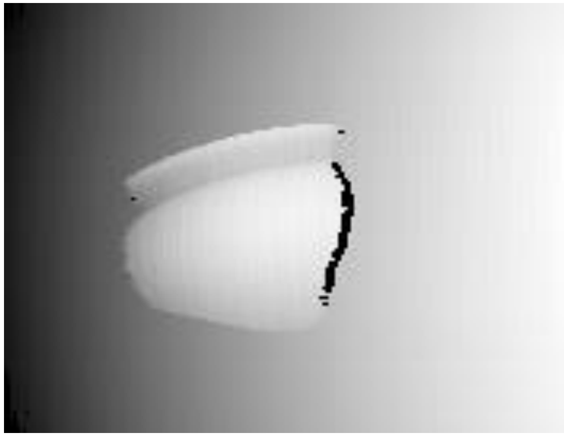
- Omnidirectional Camera

Vision Sensors

- Structured Light Sensor



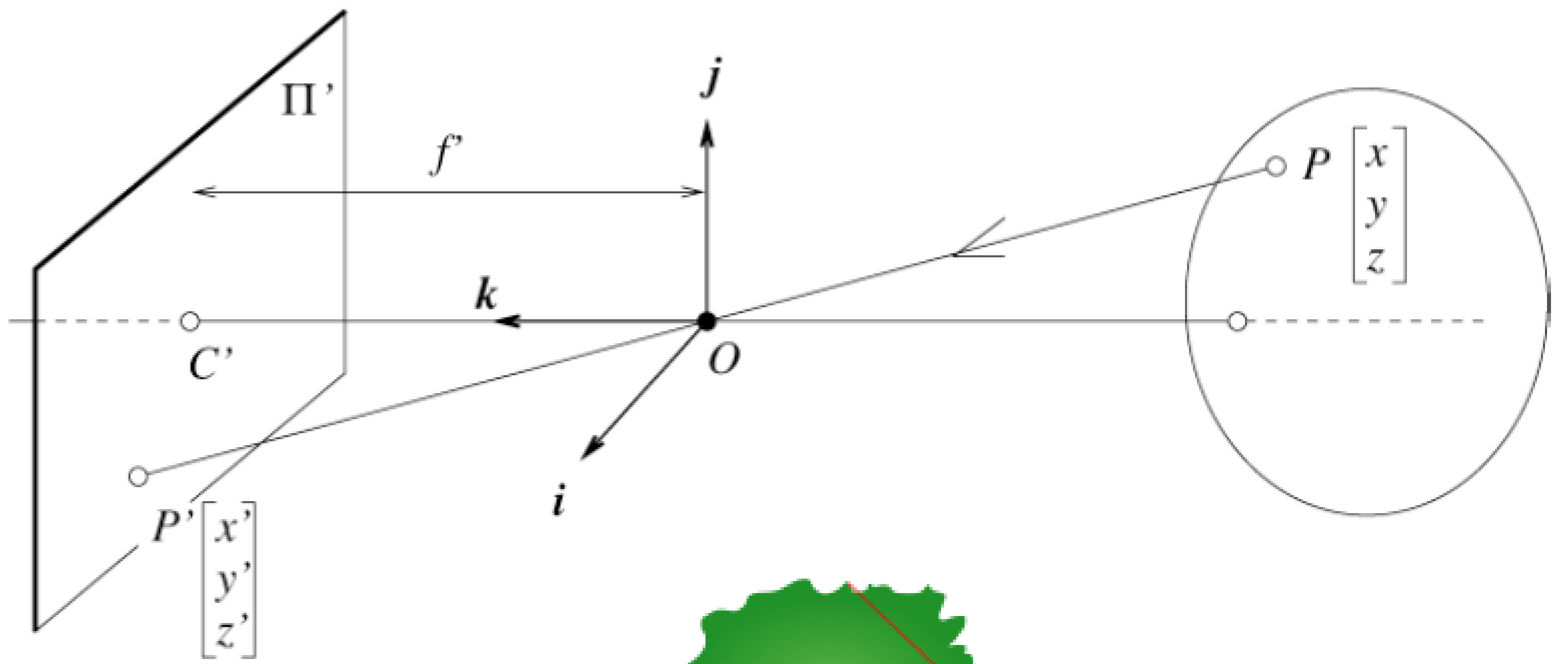


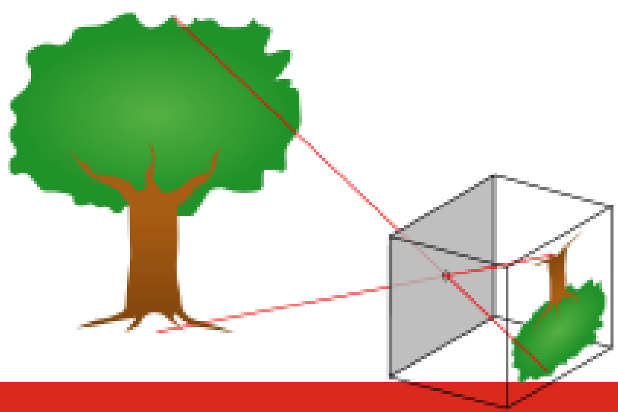


Figures from PRIP, TU Vienna

Geometry

- Single View Geometry
- **Pinhole model**





Central projection with principle point offset

$$\begin{bmatrix} fx + zp_x \\ fy + zp_y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$[p_x, p_y]'$ **is the coordinates of the principle points in image plane**

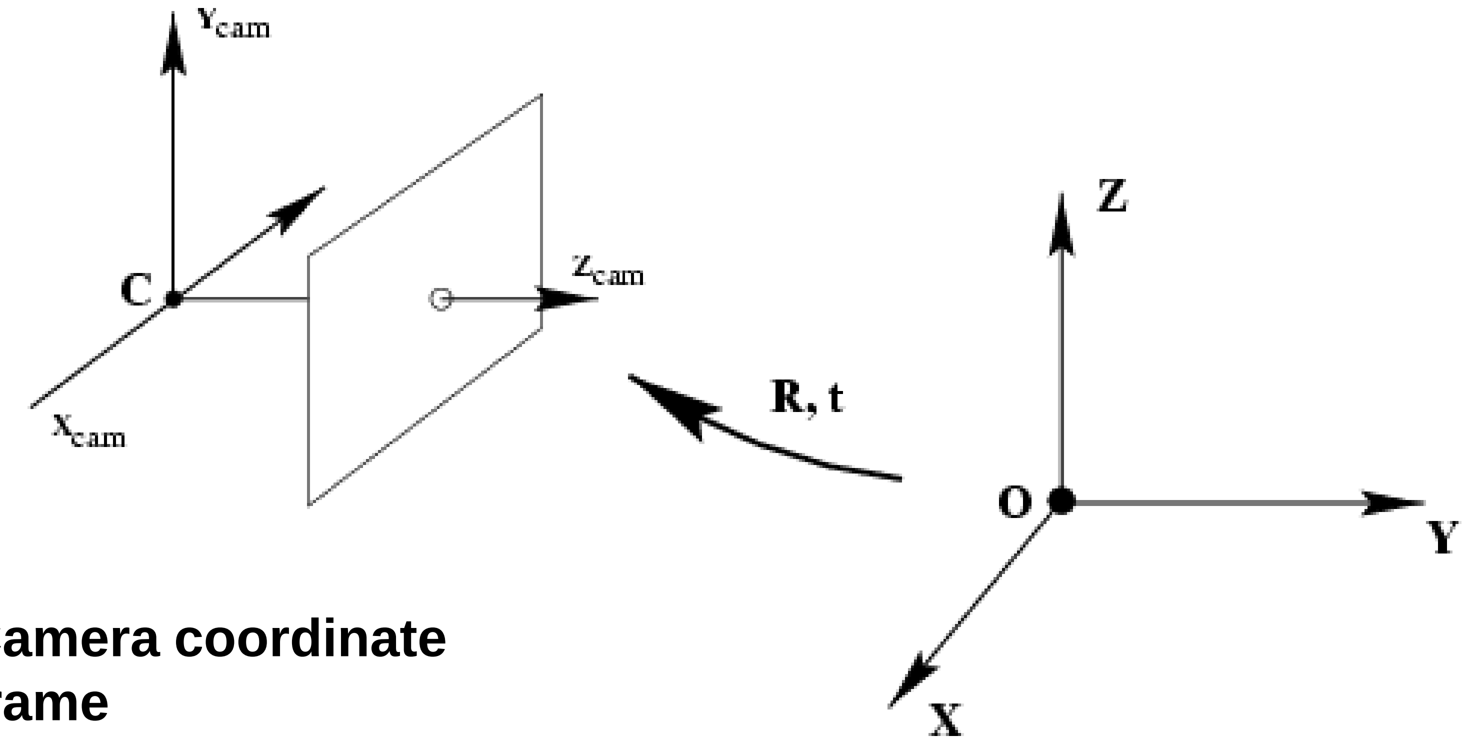
$$\mathrm{x} = \mathrm{K}[\mathrm{I} \vdots 0]\mathrm{X}_{\mathrm{cam}}$$

homogeneous

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

K is camera calibration matrix; Xcam is in camera coordinate frame

Camera rotation and translation



Camera coordinate frame

World coordinate frame

Camera is on a moving vehicle. Object is in a global reference frame.

Combine Projection and Transformation

$$\overline{X_{cam}} = R(\overline{X} - \overline{C})$$

inhomogeneous

From world coordinate frame to camera coordinate frame

Matrix?

homogeneous

Combine Projection and Transformation

$$\overline{\mathrm{X}_{\mathrm{cam}}} = \mathrm{R}(\overline{\mathrm{X}} - \overline{\mathrm{C}})$$

inhomogeneous

From world coordinate frame to camera coordinate frame

$$\mathrm{X}_{cam} = \begin{bmatrix} R & -R\overline{C} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} R & -R\overline{C} \\ 0 & 1 \end{bmatrix} \mathrm{X}$$

homogeneous

Combine Projection and Transformation

- Step 0: Points are expressed in some coordinate system that is not the cameras (e.g. a model or a robot):

  $$p = [x, y, z, 1]'$$

- Step 1: Transform the points into camera coordinates

  $$q = [x', y', z', 1]' = T\,p$$

- Step 2: Project the points

  $$u = f\,x'/z' \;;\; v = f\,y'/z'$$

Combine Projection and Transformation

$$x = K[I \vdots 0] X_{cam}$$

$$= K[I \vdots 0] \begin{bmatrix} R & -R\overline{C} \\ 0 & 1 \end{bmatrix} X$$

$$x = KR[I \vdots -\overline{C}] X$$

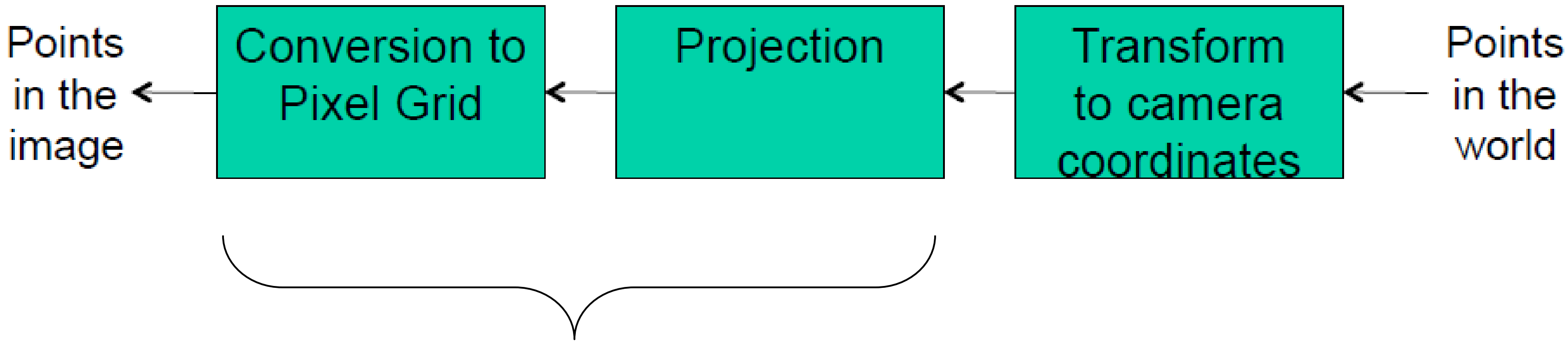
$$= PX$$

$$P = KR[I \vdots -\overline{C}]$$

P is camera projection matrix

K: camera intrinsic parameters; R,C: camera extrinsic parameters

The Projection "Chain"



$$x = (u, v) \qquad \underbrace{\hphantom{xxxxxxxxxx}}_{K} \qquad X_{cam} \qquad X$$

Camera calibration matrix

$$K = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Non-square pixels

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Square pixels

α_x and α_y represent the focal length in terms of pixel dimensions in x and y direction respectively in the image plane

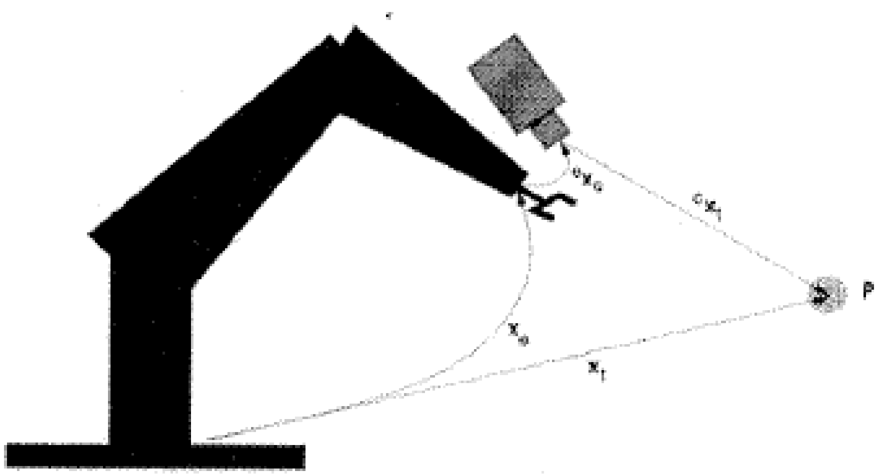
Geometry

- Activity-2:
  - Camera:
    - Image: 800*800
    - Principle point: center
    - Focal length: (400,400)
    - Pose: r = (0,0,0) => R = I, T = [10,20,2]
  - 3D Point (25,50,80)
  - Image point (u,v)?

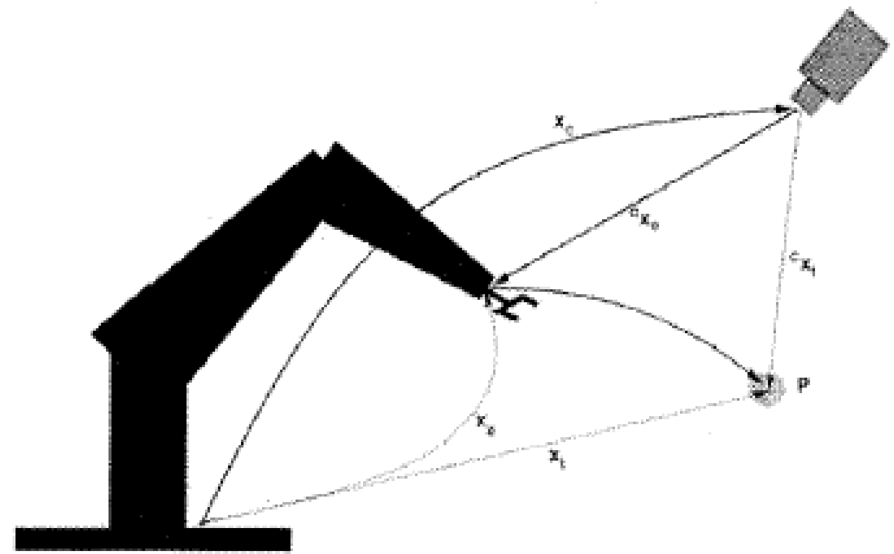
Geometry

- Activity-2:
  - Camera:
    - Image: 800*800
    - Principle point: center
    - Focal length: (400,400)
    - Pose: r = (0,0,0) => R = I, T = [10,20,2]
  - 3D Point (25,50,80)
  - Image point (u,v)?
  - **(476.9231, 553.8462)**

Camera Configurations for Visual Servoing



End-Effector Mounted



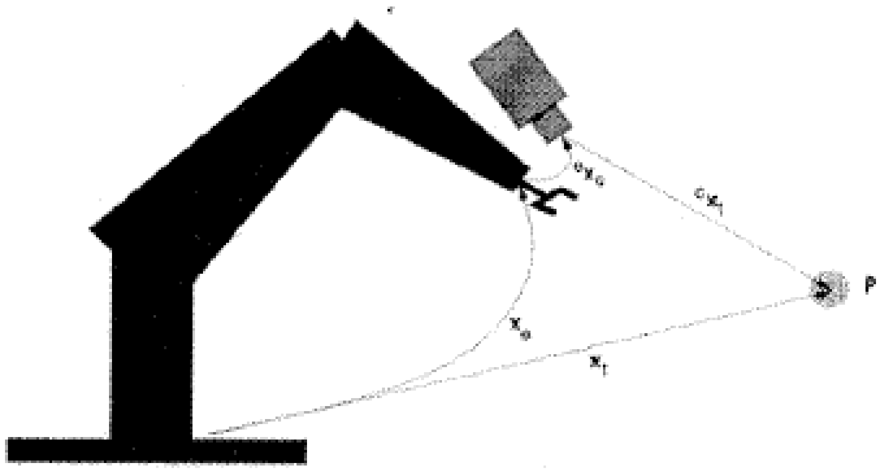
Fixed

Figures from S.Hutchinson: A Tutorial on Visual Servo Control

Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing
  - The aim of all vision-based control schemes is to minimize an error, which is typically defined by

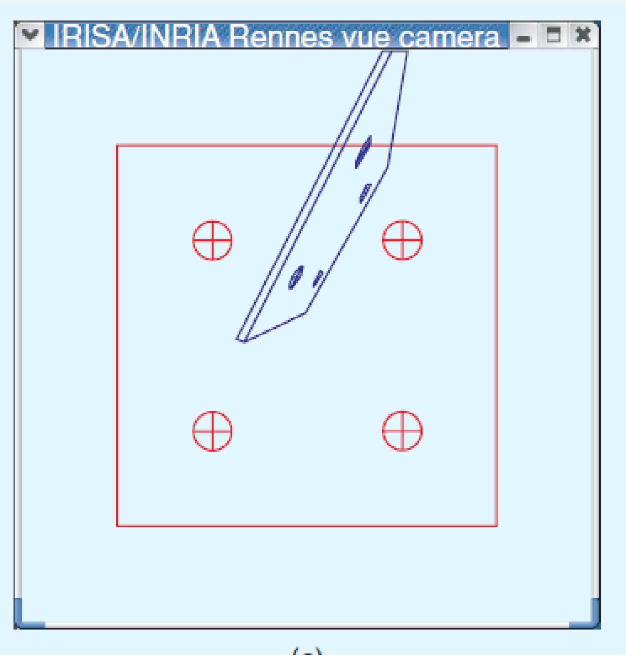
$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$



Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing
  - The aim of all vision-based control schemes is to minimize an error, which is typically defined by

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$

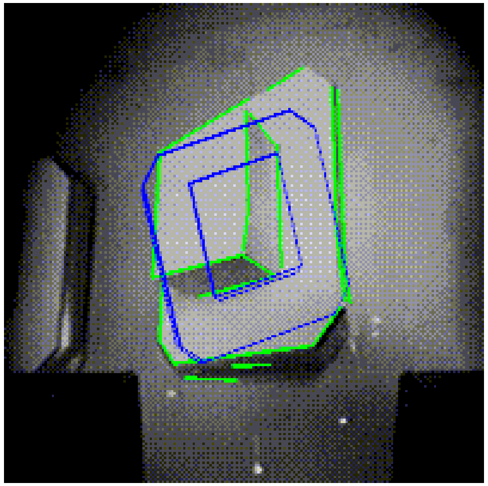


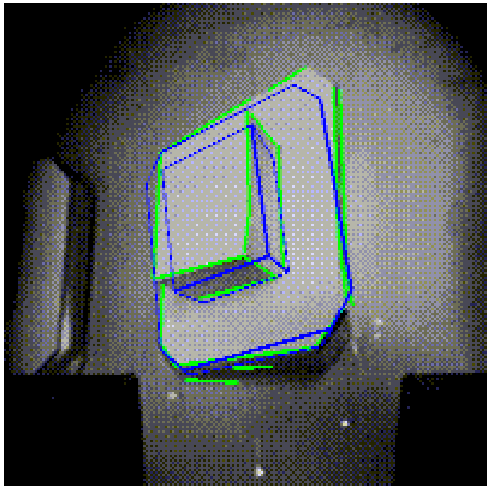
Visual Servoing

UNIVERSITY OF TECHNOLOGY SYDNEY

- End-Effector Mounted
- Basic Components of Visual Servoing
  - The aim of all vision-based control schemes is to minimize an error, which is typically defined by

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$





Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing
  - The aim of all vision-based control schemes is to minimize an error, which is typically defined by

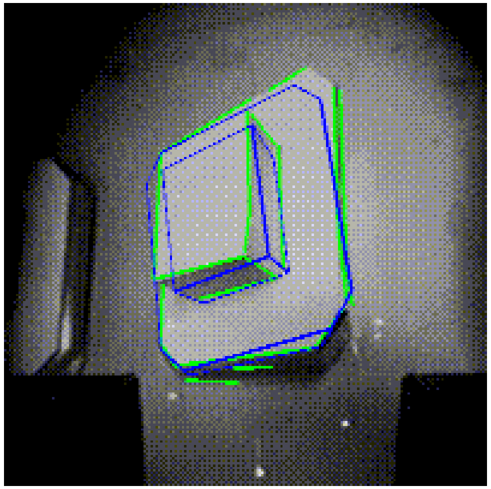
$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$

S*: desired values of the features

S( ): calculated values of the features

M(t): the measurements from Image

a: parameters of the camera



Visual Servoing

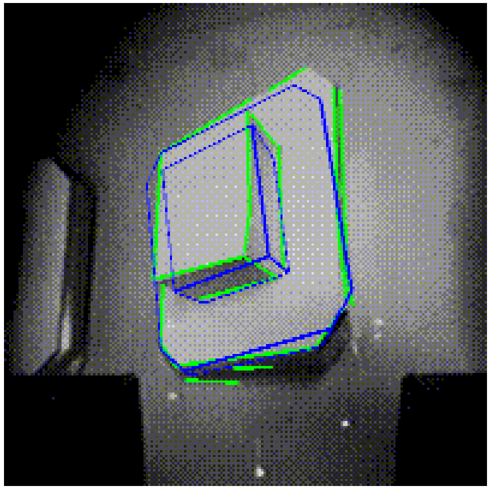
- End-Effector Mounted
- Basic Components of Visual Servoing
  - The aim of all vision-based control schemes is to minimize an error, which is typically defined by

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$

If S* is selected, velocity controller

$$\mathbf{v}_c = (v_c, \boldsymbol{\omega}_c) \qquad \dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s}} \mathbf{v}_c$$

Where Ls is the *interaction matrix* or *feature Jacobian.*



Visual Servoing

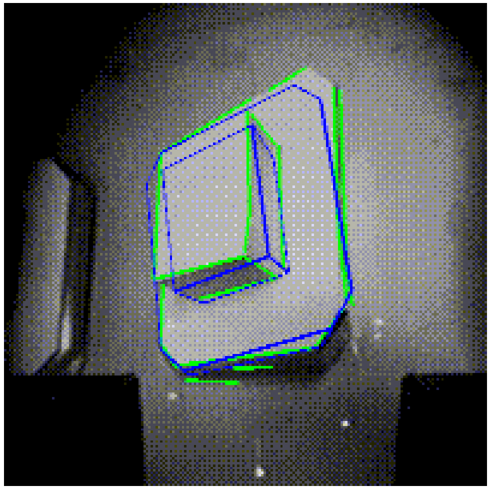
- End-Effector Mounted
- Basic Components of Visual Servoing
  - The aim of all vision-based control schemes is to minimize an error, which is typically defined by

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^* \qquad \dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v}_c$$

Relationship between e and vc

$$\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v}_c$$

With Le = Ls. Why?



Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing

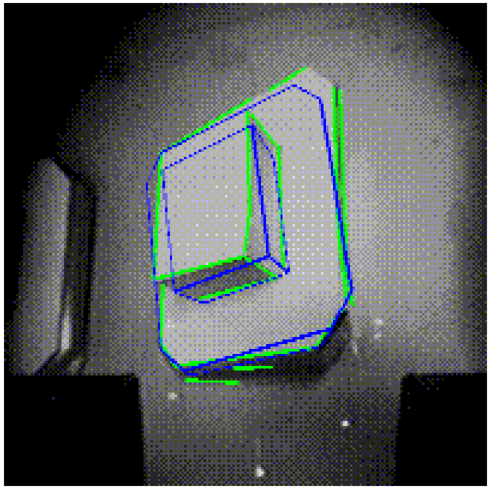
$$\dot{\mathbf{e}} = \mathbf{L_e} \mathbf{v}_c$$

How to solve

Derivative *e* w.r.t *t*

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

Linear Least Squares



Visual Servoing

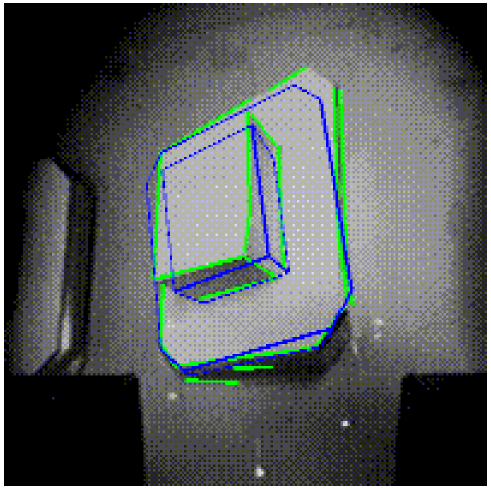
- End-Effector Mounted
- Basic Components of Visual Servoing

$$\dot{\mathbf{e}} = \mathbf{L_e}\mathbf{v}_c$$

How to solve

Linear Least Squares

$$\mathbf{v}_c = -\lambda \mathbf{L_e^+} \mathbf{e}$$



Where

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

Is the Moore-Penrose pseudo-inverse of Le.

- End-Effector Mounted
- Basic Components of Visual Servoing

Camera system

3D point (X,Y,Z)

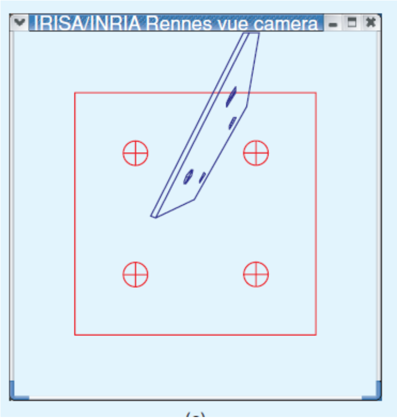
2D point (x,y)

measurement m=(u,v)

intrinsic parameters (cu,cv,f)

How to compute (x,y)?

$$\begin{cases} x = X/Z = (u - c_u)/f \\ y = Y/Z = (v - c_v)/f, \end{cases}$$



Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing

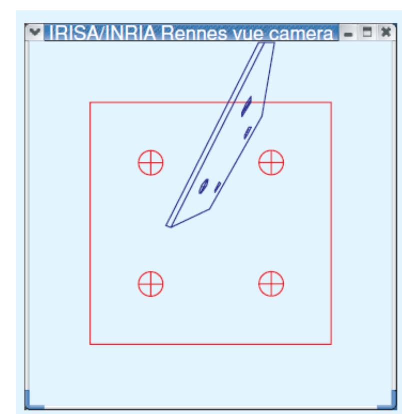
2D point (x,y)

$$\begin{cases} x & = & X/Z = (u - c_u)/f \\ y & = & Y/Z = (v - c_v)/f, \end{cases} \qquad \mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*$$

We take s = (x,y)

How to compute

$$\dot{\mathbf{s}} = \mathbf{L}_{\mathbf{s}} \mathbf{v}_c$$



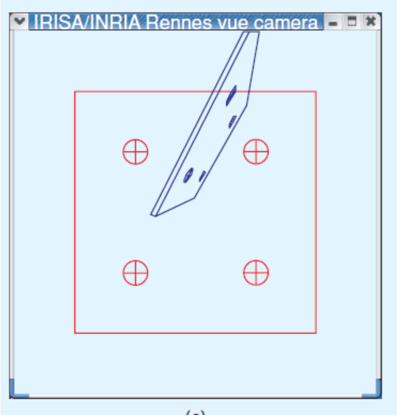
- End-Effector Mounted
- Basic Components of Visual Servoing

Derivatives

$$\begin{cases} \dot{x} &= \dot{X}/Z - X\dot{Z}/Z^2 &= (\dot{X} - x\dot{Z})/Z \\ \dot{y} &= \dot{Y}/Z - Y\dot{Z}/Z^2 &= (\dot{Y} - y\dot{Z})/Z. \end{cases}$$

We can relate the velocity of the 3-D point to the camera spatial velocity using the well-known equation

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



Visual Servoing

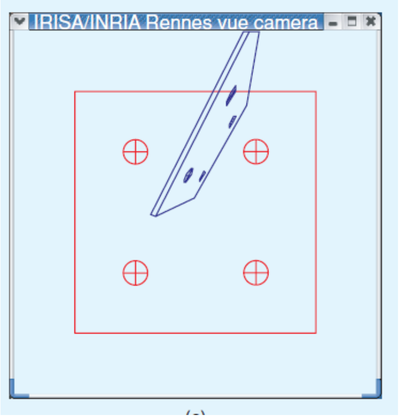
- End-Effector Mounted
- Basic Components of Visual Servoing

$$\begin{cases} \dot{x} = \dot{X}/Z - X\dot{Z}/Z^2 = (\dot{X} - x\dot{Z})/Z \\ \dot{y} = \dot{Y}/Z - Y\dot{Z}/Z^2 = (\dot{Y} - y\dot{Z})/Z. \end{cases}$$

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

We have

$$\begin{cases} \dot{x} = -v_x/Z + xv_z/Z + xy\omega_x - (1 + x^2)\omega_y + y\omega_z \\ \dot{y} = -v_y/Z + yv_z/Z + (1 + y^2)\omega_x - xy\omega_y - x\omega_z \end{cases}$$



Visual Servoing

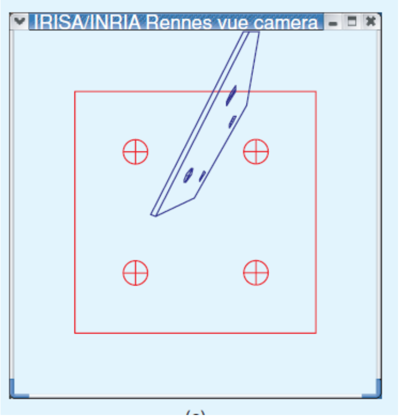
- End-Effector Mounted
- Basic Components of Visual Servoing

$$\begin{cases} \dot{x} &= \dot{X}/Z - X\dot{Z}/Z^2 &= (\dot{X} - x\dot{Z})/Z \\ \dot{y} &= \dot{Y}/Z - Y\dot{Z}/Z^2 &= (\dot{Y} - y\dot{Z})/Z. \end{cases}$$

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

We have

$$\begin{cases} \dot{x} = -v_x/Z + xv_z/Z + xy\omega_x - (1 + x^2)\omega_y + y\omega_z \\ \dot{y} = -v_y/Z + yv_z/Z + (1 + y^2)\omega_x - xy\omega_y - x\omega_z \end{cases}$$



Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing

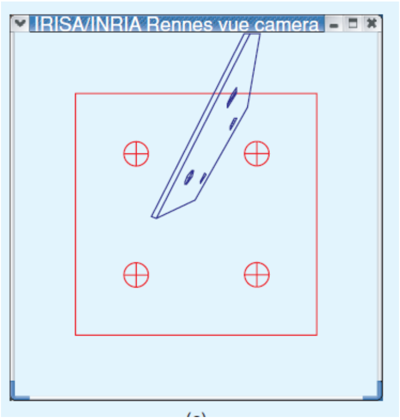
$$\begin{cases} \dot{x} = -v_x/Z + xv_z/Z + xy\omega_x - (1+x^2)\omega_y + y\omega_z \\ \dot{y} = -v_y/Z + yv_z/Z + (1+y^2)\omega_x - xy\omega_y - x\omega_z \end{cases}$$

Which can be rewritten

$$\dot{\mathbf{s}} = \mathbf{L_s v}_c$$

Where

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



Visual Servoing

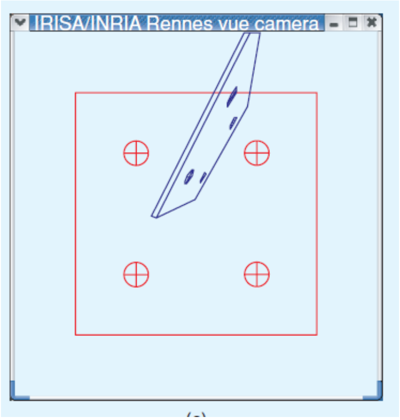
- End-Effector Mounted
- Basic Components of Visual Servoing

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

Must estimate or approximate the value of Z

At least 3 points are necessary

$$\mathbf{L_x} = \begin{bmatrix} \mathbf{L_{x_1}} \\ \mathbf{L_{x_2}} \\ \mathbf{L_{x_3}} \end{bmatrix}$$



Visual Servoing

UNIVERSITY OF TECHNOLOGY SYDNEY

❖ Activity-3:

- Camera:
  - Image: 1000*1000
  - Principle point: (400,400)
  - Focal length: (400,400)

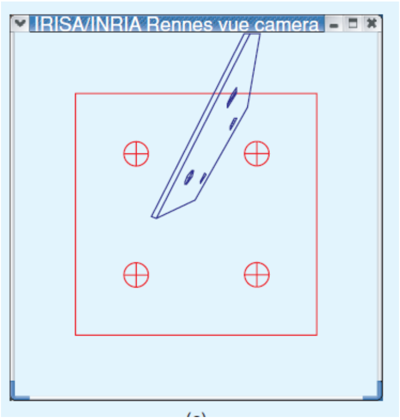
  - Pose: r = (0,0,0) => R = I, T = [10,20,2]

- Desired features
  - (0,0), (800,0), (800,0),(800,800)

- Measurements
  - (0,0), (800,0), (800,0),(800,800) +50

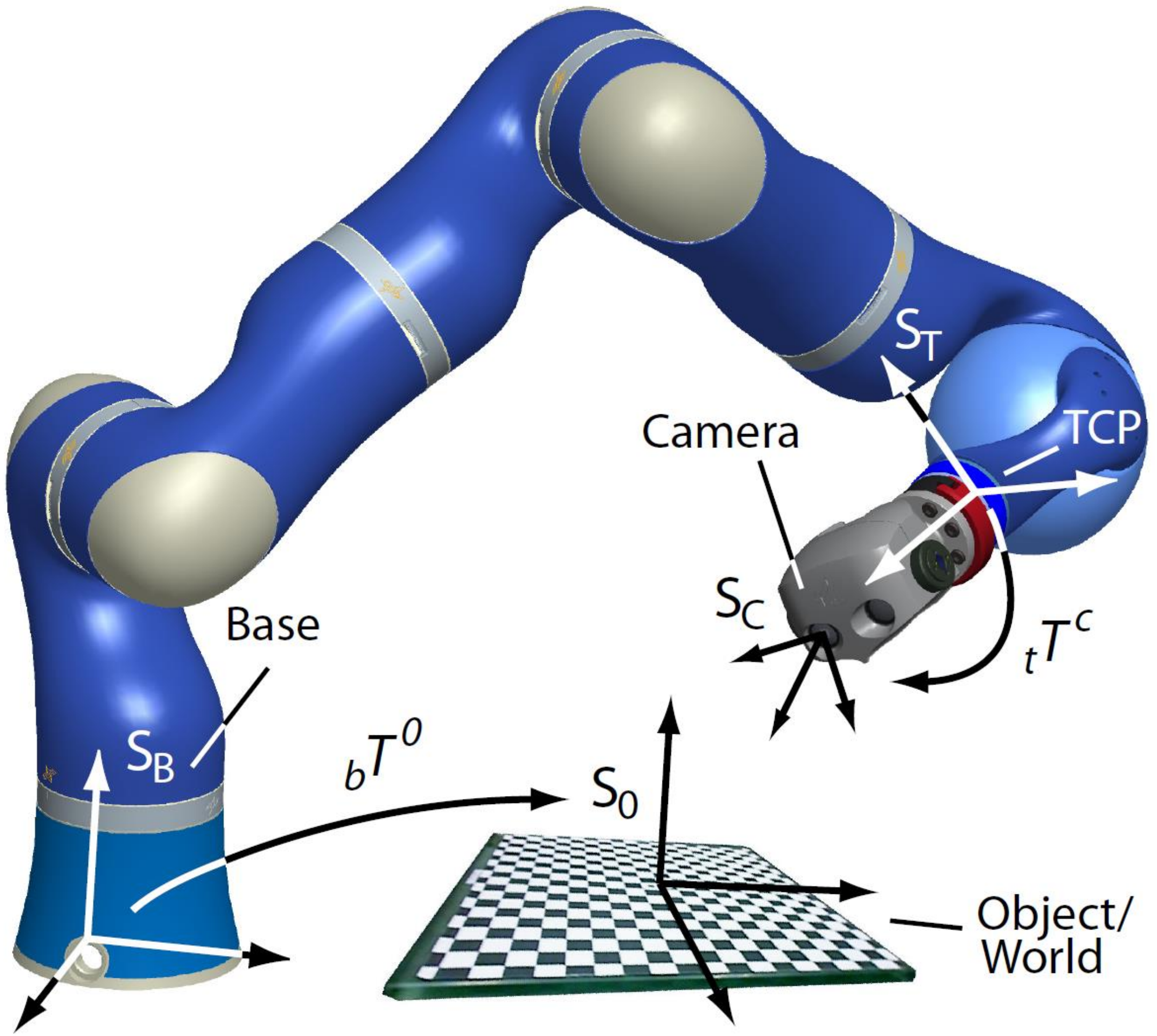
- Assume Z = 50

- Camera velocity vc?



Hand-eye Calibration

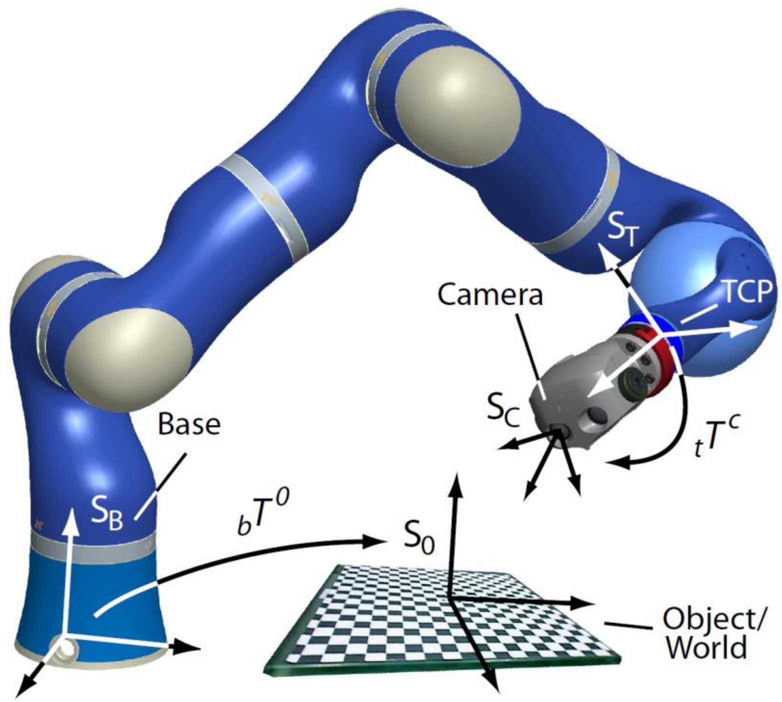
UNIVERSITY OF TECHNOLOGY SYDNEY



Hand-eye Calibration

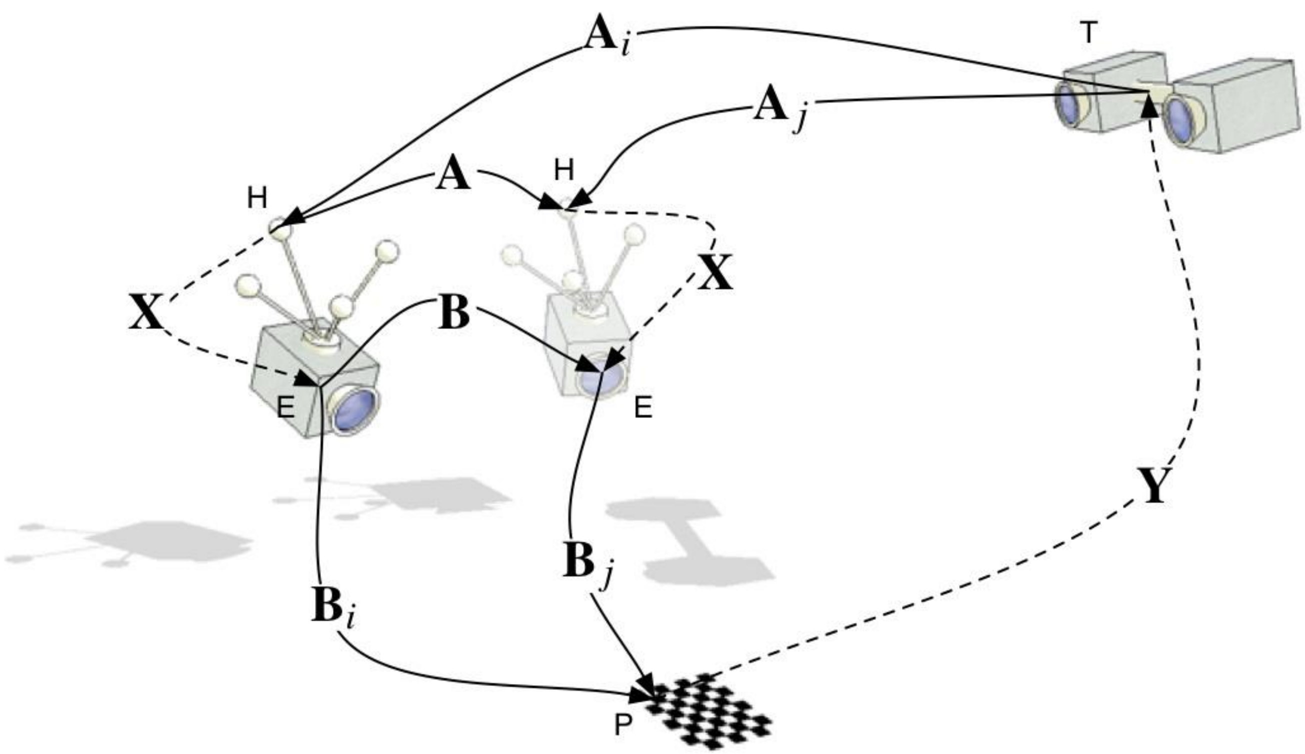
❖ Hand-eye Calibration

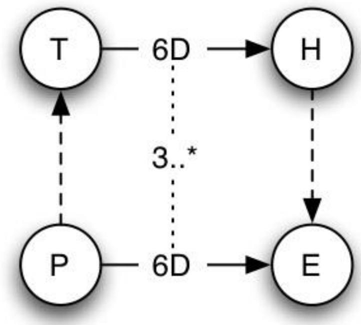
- Relative Pose
- Hand (end effector)
- Eye (Camera)



Hand-eye Calibration

- Hand-eye Calibration



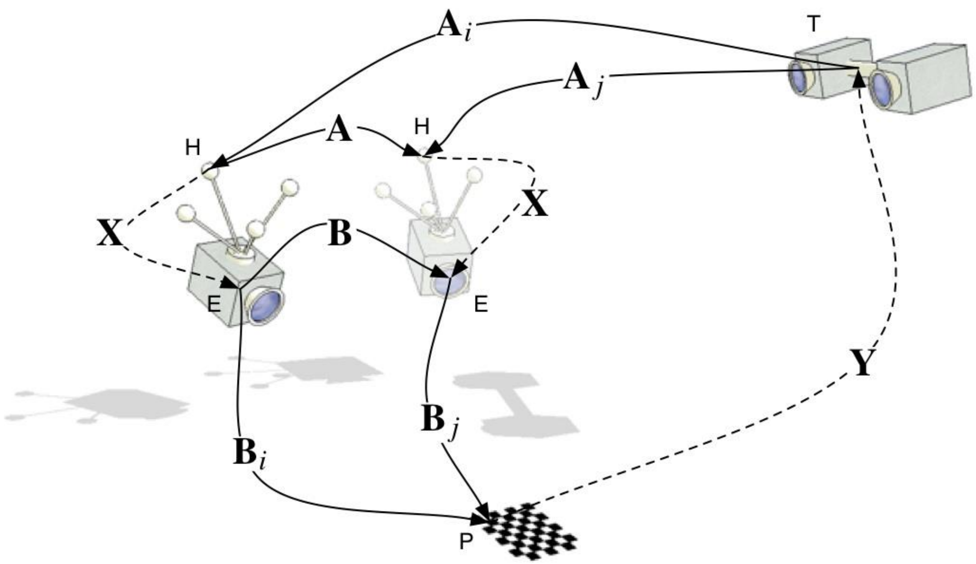


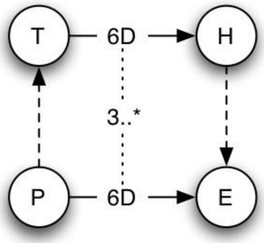
Hand-eye Calibration

UNIVERSITY OF TECHNOLOGY SYDNEY

- Hand-eye Calibration

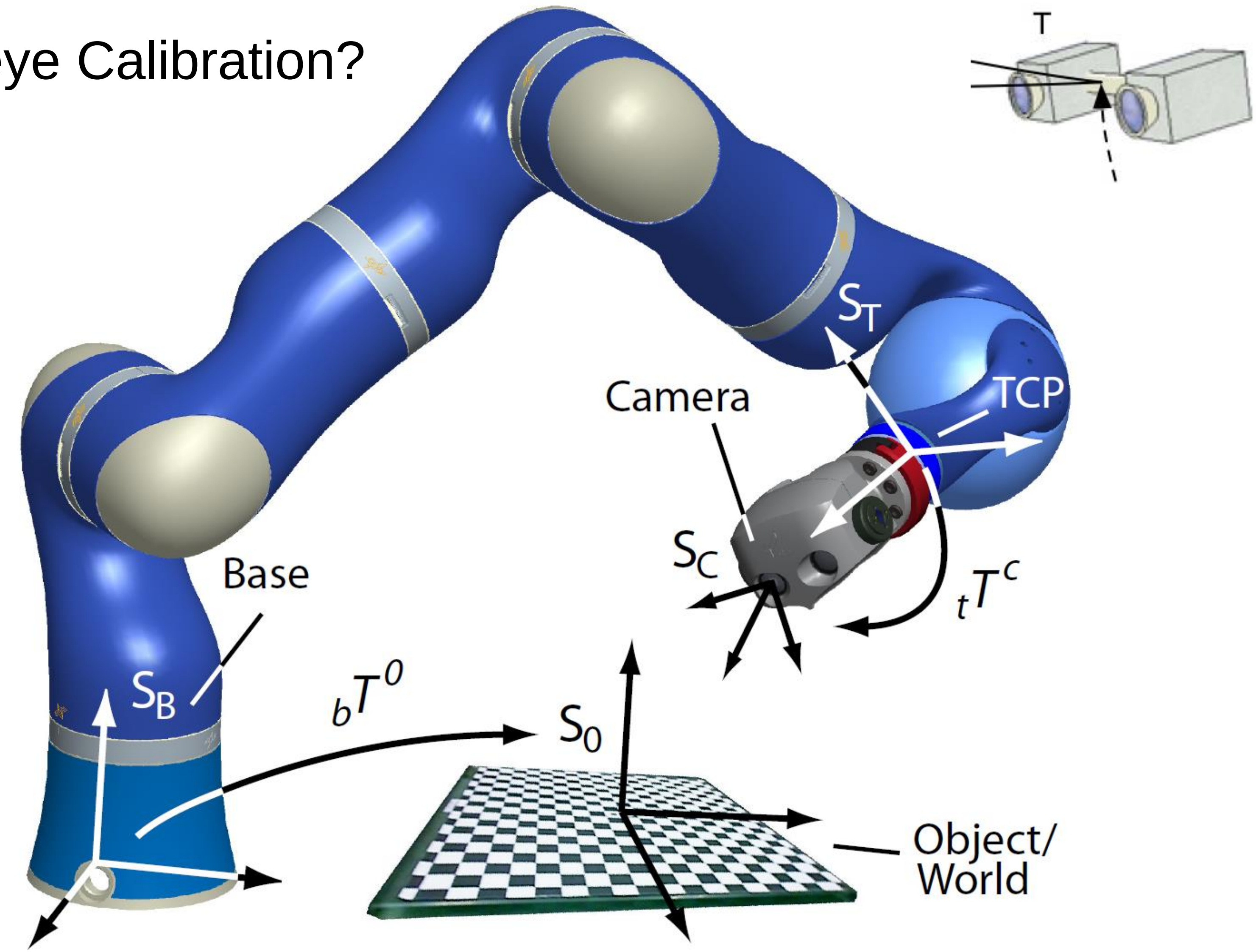
$$A*X = X*B$$





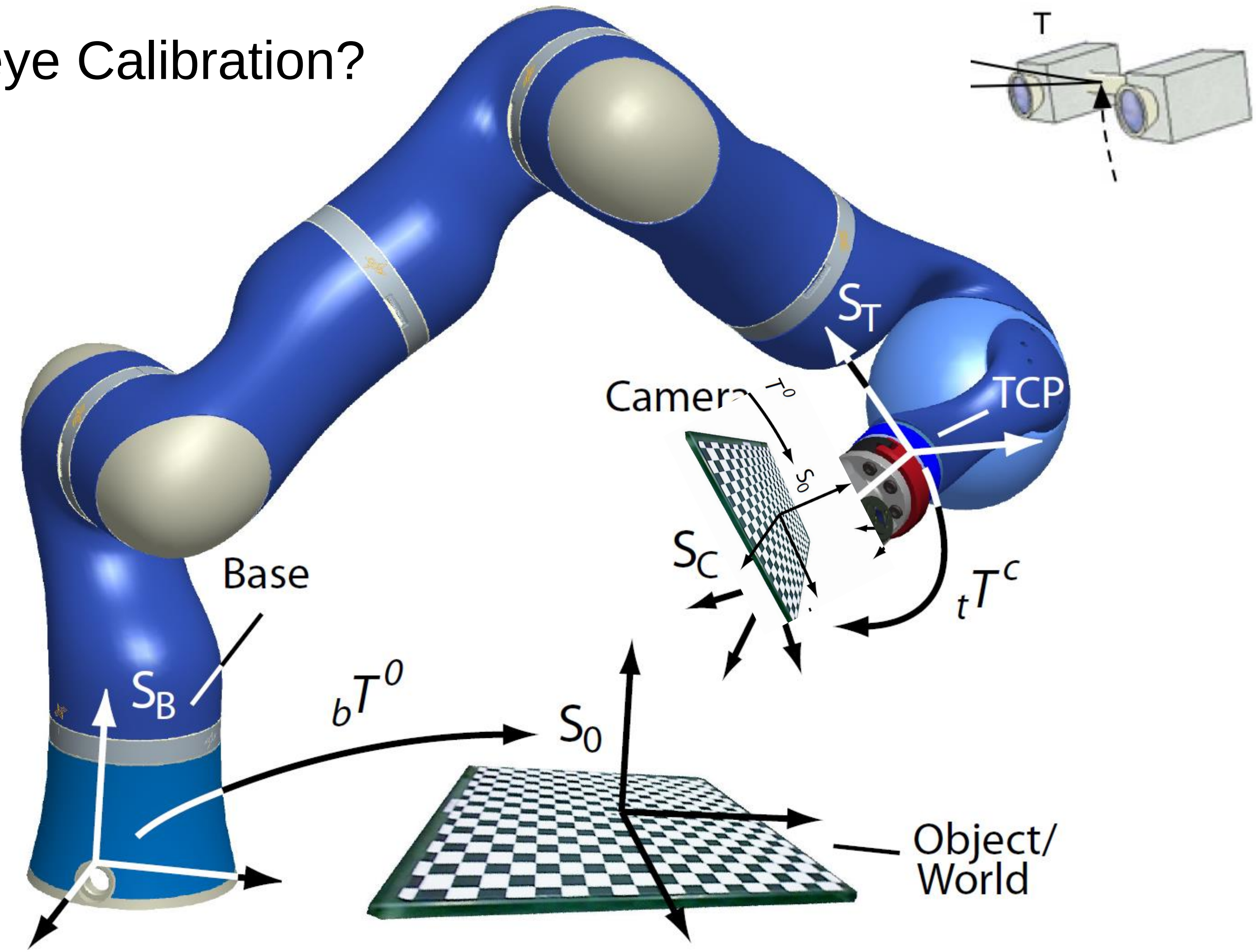
Hand-eye Calibration

- Hand-eye Calibration?



Hand-eye Calibration

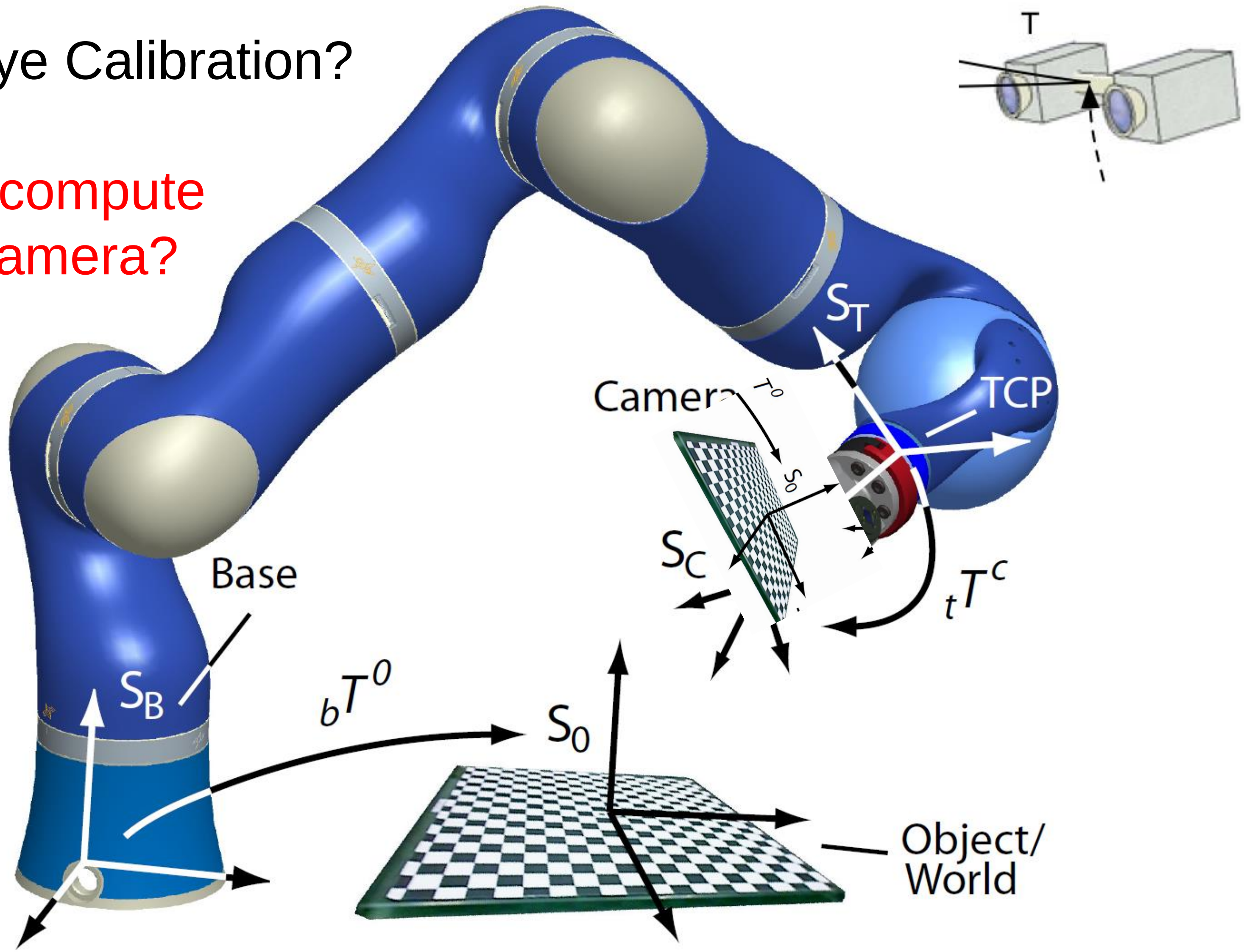
❖ Hand-eye Calibration?



Hand-eye Calibration

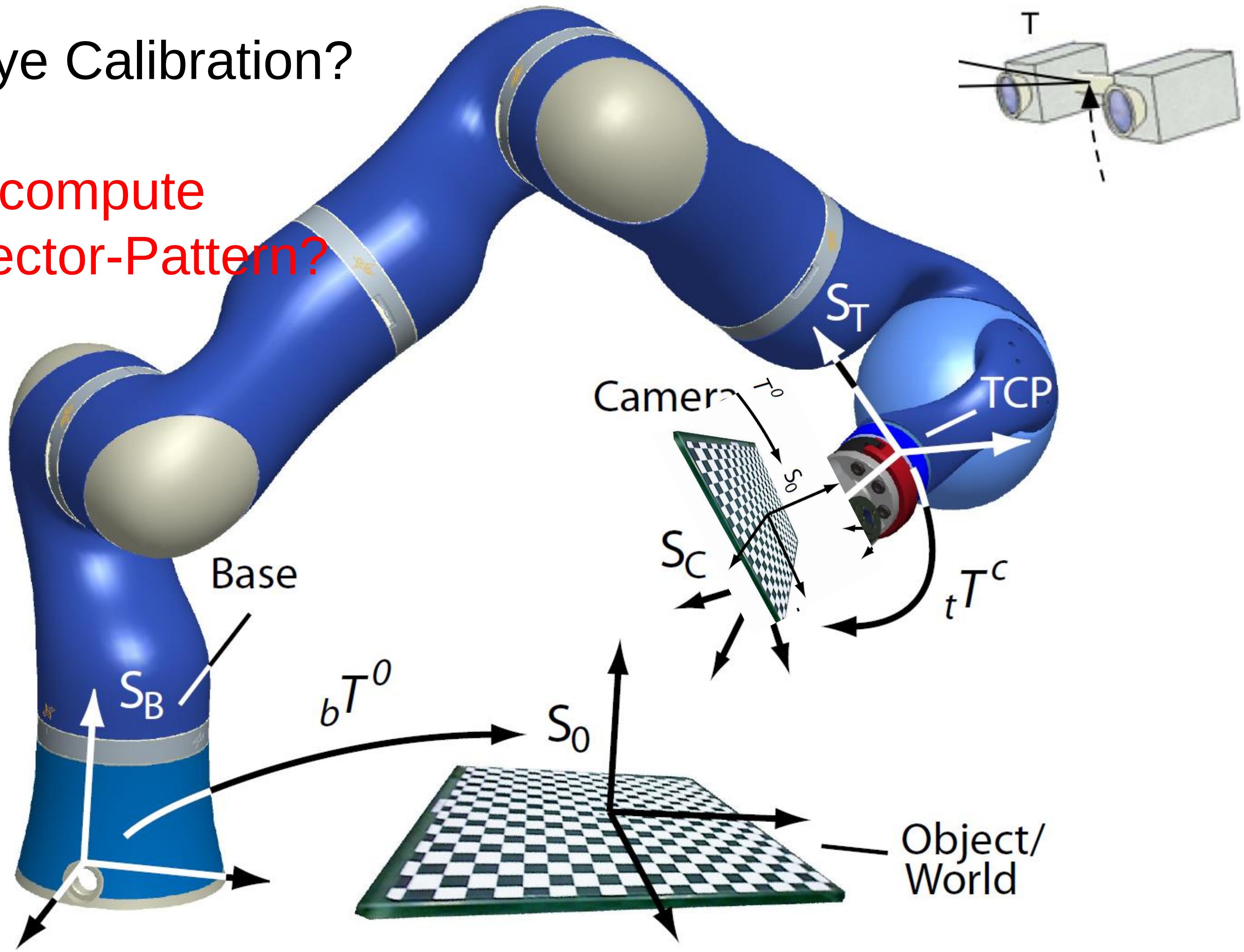
- Hand-eye Calibration?

- How to compute
- Base-Camera?



Hand-eye Calibration

- ❖ Hand-eye Calibration?
- ❖ How to compute
- ❖ End effector-Pattern?



THANK YOU

Questions?

GROUP PROJECTS