41014 Sensors and Control for Mechatronic Systems

Lecture-10: Subject Review

Dr. **Liang Zhao**

Centre for Autonomous Systems
University of Technology,
Sydney

UNIVERSITY OF
TECHNOLOGY SYDNEY

2. Final Exam

Final Exam: 50%

- 2 hours
- Short Written Answers + Long Written Answers
- Restrict Open Book: 2x A4 hand writing papers

Moderation of marks

- A pass in this subject is 50% provided the following conditions are met:
- A reasonable attempt has been made at all design projects and assignments;
- Mark of at least 50% of the final exam is obtained.

3. Subject Review

❖ Introduction

- Fetch Robot Navigation and Grasping
- How many problems involved in this application?
- What sensors and control methods are used in each problem?

3.1 Introduction

UNIVERSITY OF TECHNOLOGY SYDNEY

❖ Problem 1: Navigation/Localization

- Robot needs to go from the starting point to the table
- Sensors: 2D laser and/or RGB-D camera

❖ Problem 2: Object recognition

- Robot needs to recognize the object, and estimate the pose
- Sensors: RGB-D camera

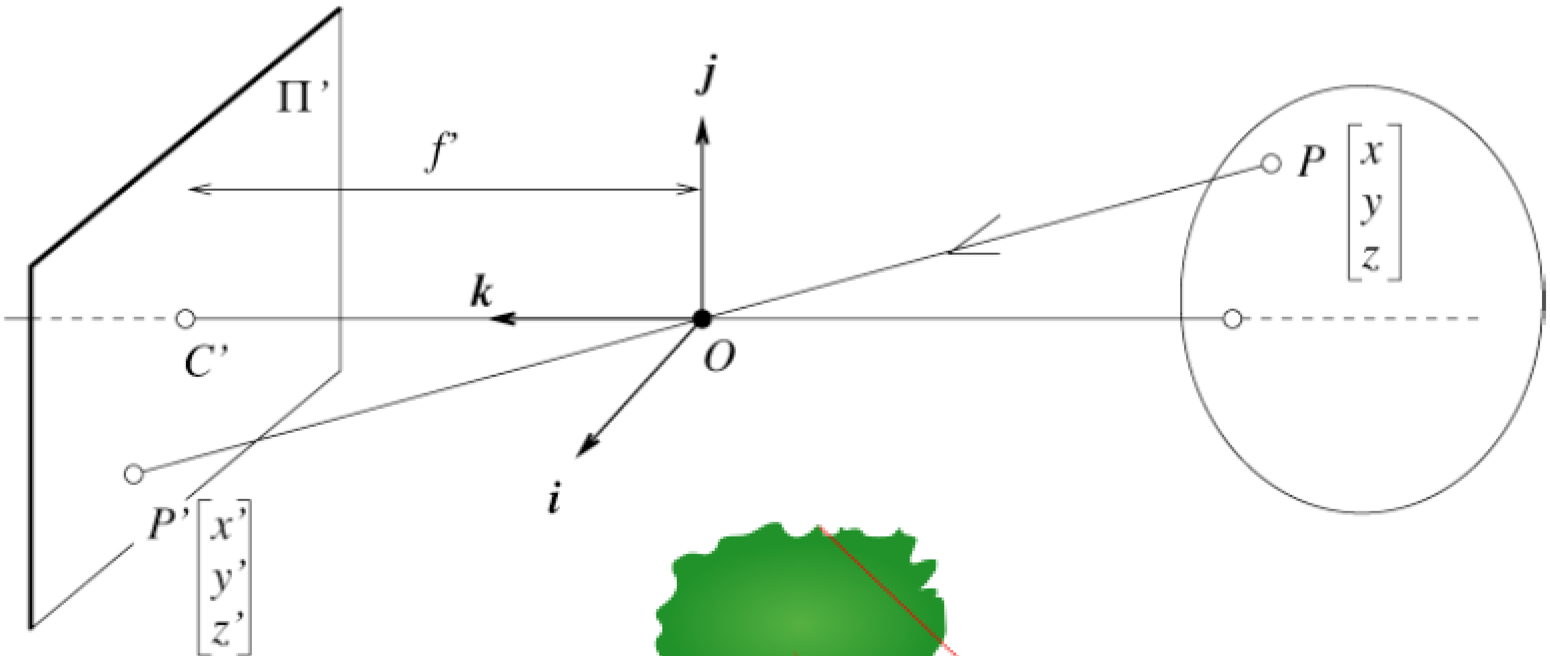
❖ Problem 3: Visual Servoing

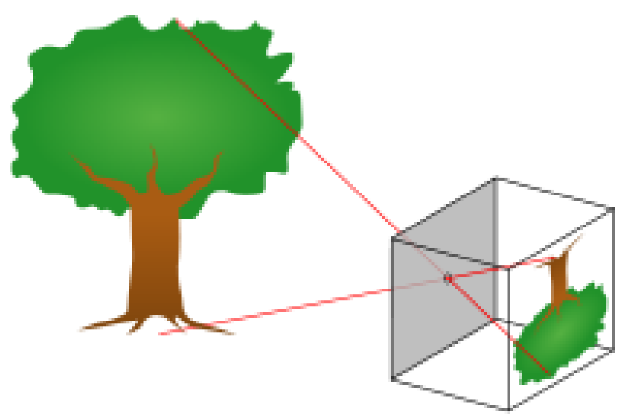
- Control the robot art to pick up the object
- Sensors: RGB-D camera, force sensor

4. Cameras:

- **Single View Geometry**

- **Pinhole model**





4.1 Cameras: Geometry

- **Central projection with principle point offset**

$$[x, y, z]' \rightarrow [f\frac{x}{z}, f\frac{y}{z}]' = \mathrm{x}$$

$$[x, y, z]' \rightarrow [f\frac{x}{z} + p_x, f\frac{y}{z} + p_y]' = \mathrm{x}$$

❖ 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**

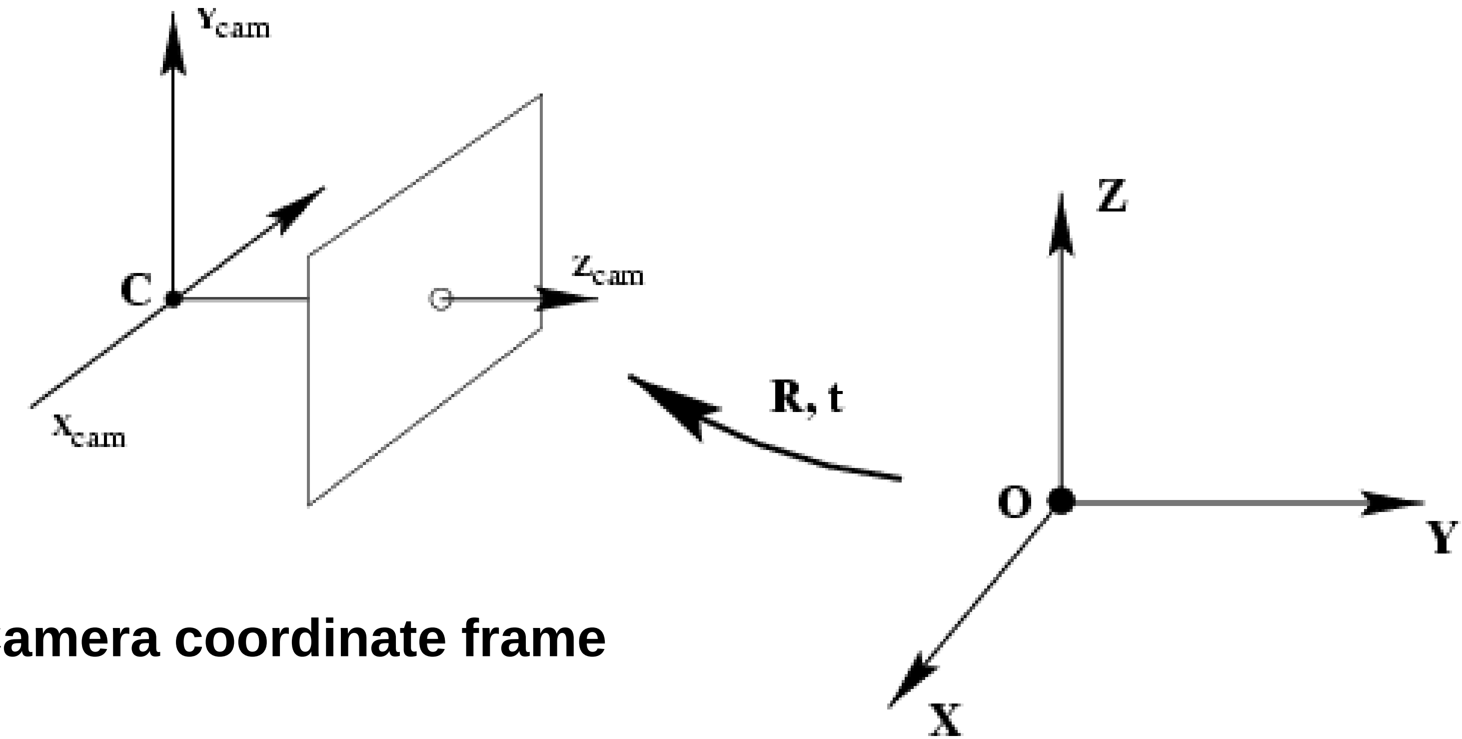
$$\mathrm{x} = \mathrm{K}[\mathrm{I} \vdots 0]\mathrm{X}_{\mathrm{cam}}$$

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

homogeneous

K is camera calibration matrix; X_cam 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.

4.1 Cameras: Geometry

UNIVERSITY OF TECHNOLOGY SYDNEY

❖ **General camera projection**

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

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

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

$$= \mathrm{PX}$$

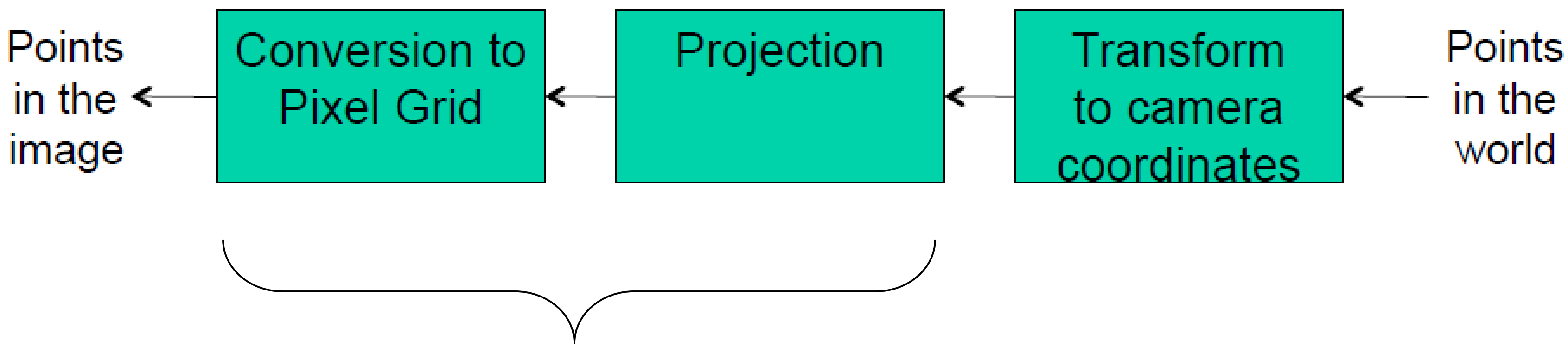
$$\mathrm{P} = \mathrm{KR}[\mathrm{I} \vdots -\overline{\mathrm{C}}]$$

P is camera projection matrix

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

UNIVERSITY OF TECHNOLOGY SYDNEY

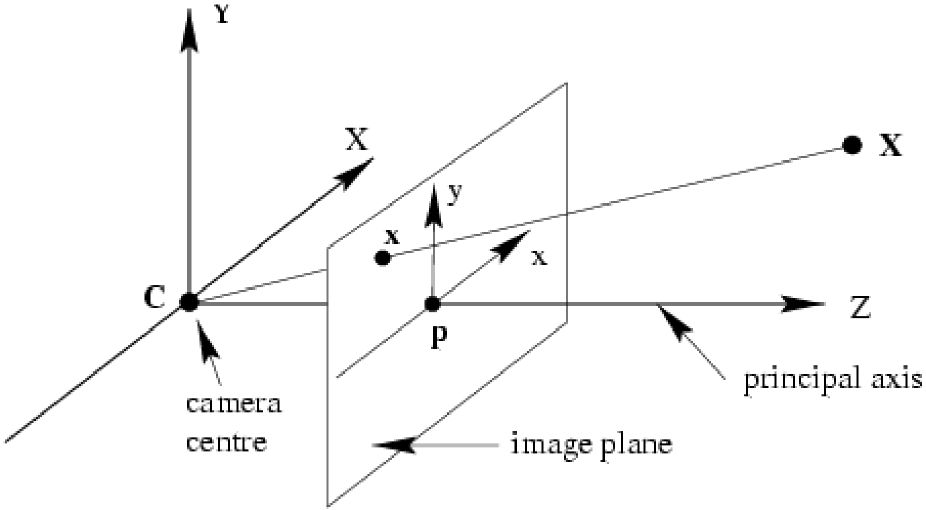
The Projection "Chain"

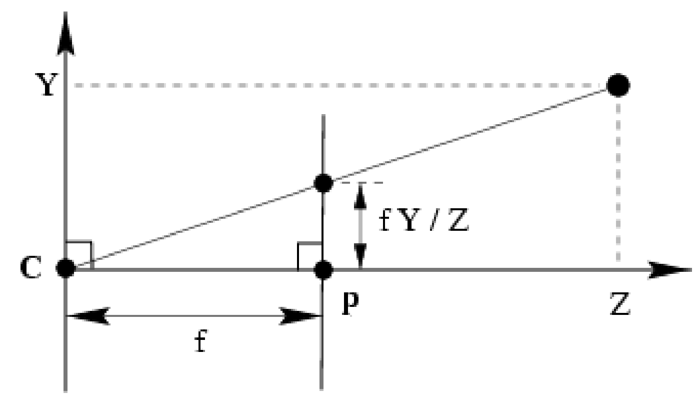


$\mathrm{x} = (\mathrm{u}, \mathrm{v})$ $\quad$ K $\quad$ $\mathrm{X}_{\mathrm{cam}}$ $\quad$ X

Activity 1

- Image resulotion: 1024*768
- Principle point: (520,389)
- Focal length: 935
- 3D Point in camera frame (15,10,80)
- **Image point (u,v)?**



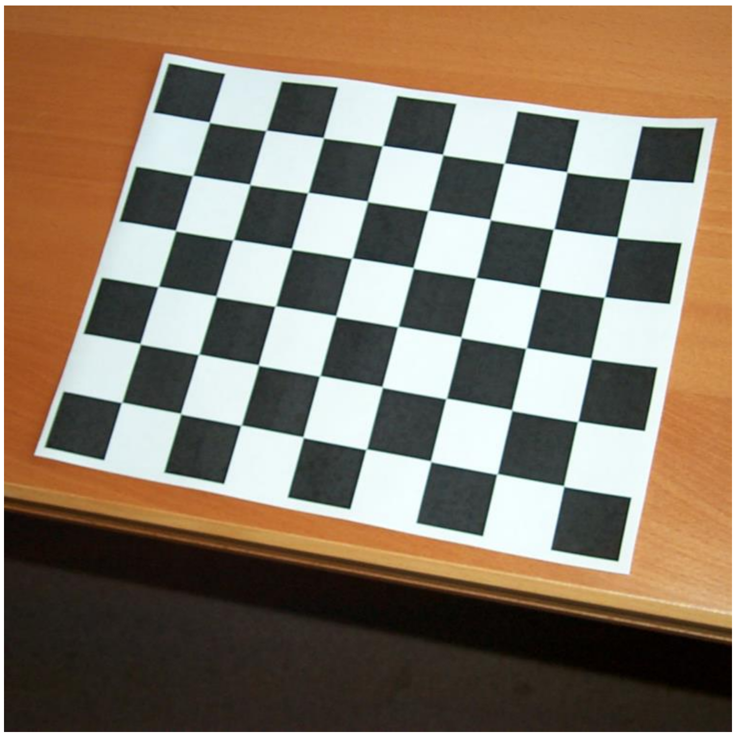


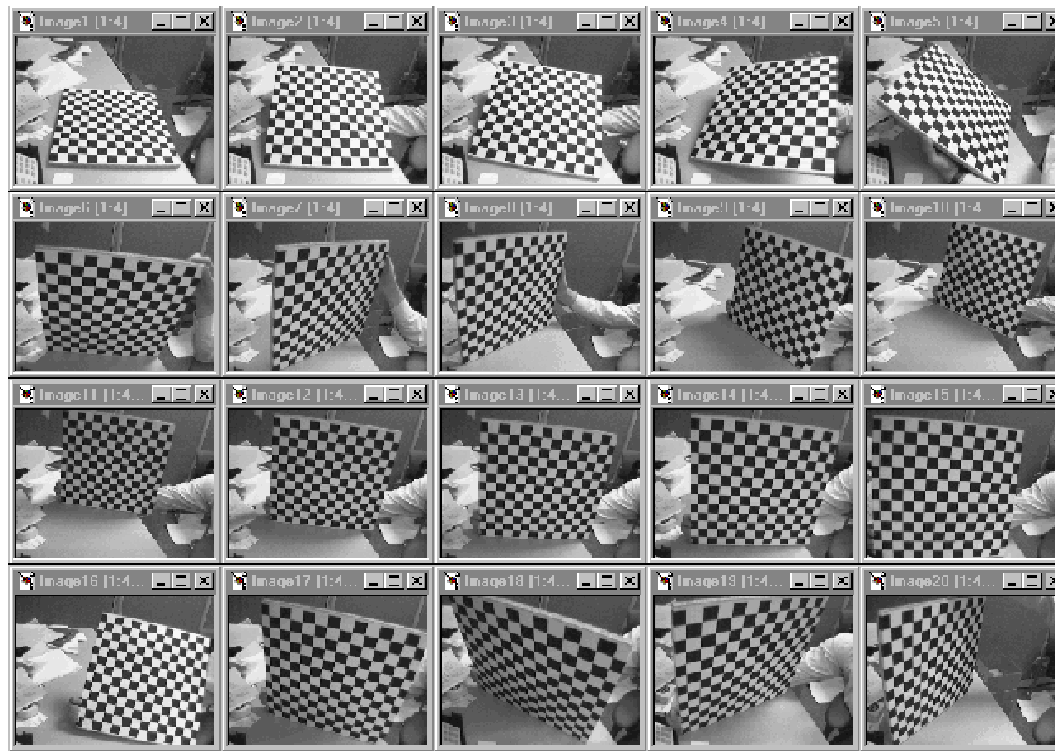
4.2 Cameras: Calibration

UNIVERSITY OF TECHNOLOGY SYDNEY

❖ Matlab implementation:

- http://www.vision.caltech.edu/bouguetj/calib_doc/index.html





4.3 Cameras: Convolution

- Modify the pixels in an image based on some function of a local neighborhood of the pixels

$$\begin{bmatrix}10 & 5 & 3\\ 4 & 5 & 1\\ 1 & 1 & 7\end{bmatrix} \xrightarrow{\text{Some function}} \begin{bmatrix} & & \\ & 7 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix}10 & 5 & 3\\ 4 & 5 & 1\\ 1 & 1 & 7\end{bmatrix} \otimes \underbrace{\begin{bmatrix}0 & 0 & 0\\ 0 & 0.5 & 0\\ 0 & 1.0 & 0.5\end{bmatrix}}_{\text{kernel}} = \begin{bmatrix} & & \\ & 7 & \\ & & \end{bmatrix}$$

4.3 Cameras: Convolution

- **Activity 2**

$$\begin{bmatrix} 0 & 25 & 50 & 100 \\ 25 & 50 & 100 & 50 \\ 50 & 100 & 50 & 25 \\ 100 & 50 & 25 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 50 & -200 \\ -200 & 50 \end{bmatrix}$$

UNIVERSITY OF TECHNOLOGY SYDNEY

Triangulation

$$\frac{X}{Z} = \frac{x_l}{f} \qquad \frac{X-B}{Z} = \frac{x_r}{f}$$

$$X = \frac{Z.x_l}{f} \qquad X = \frac{Z.x_r}{f} + B$$

$$\frac{Z.x_l}{f} = \frac{Z.x_r}{f} + B$$

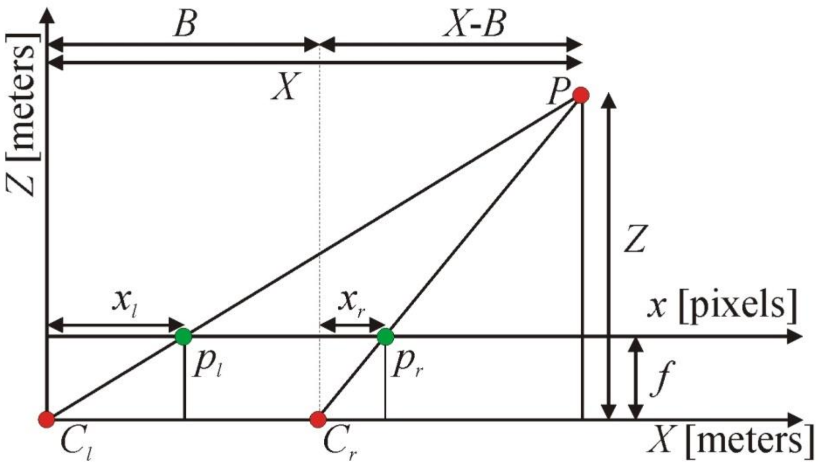
$$Z.x_l = Z.x_r + B.f$$

$$Z.(x_l - x_r) = B.f$$

$$Z = \frac{B.f}{x_l - x_r} = \frac{B.f}{d}$$

Where *d* is the disparity

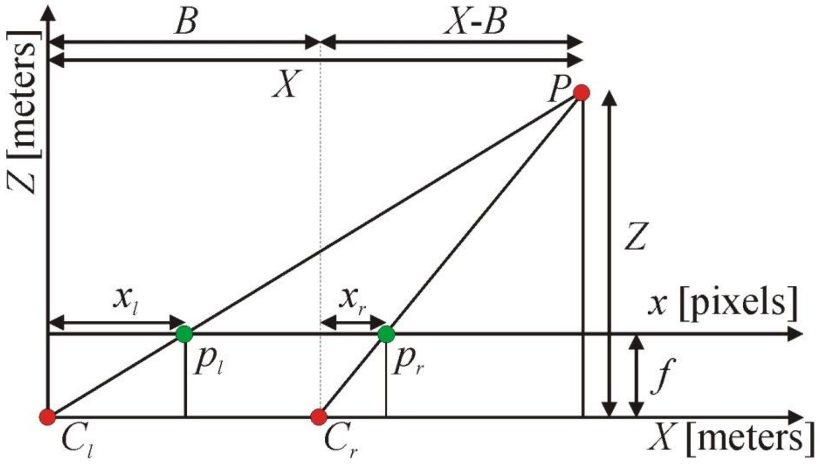
– It can be seen that the disparity is inversely proportional to the depth.



5.1 Stereo: Triangulation

Activity 1

- Image resulotion: 800*600
- Principle point: (400,300)
- Baseline: 100mm
- Focal length: 400
- Image Point:
  - Left Camera (600,300)
  - Right Camera (550,300)
- **Point 3D location in camera frame (X,Y,Z)?**

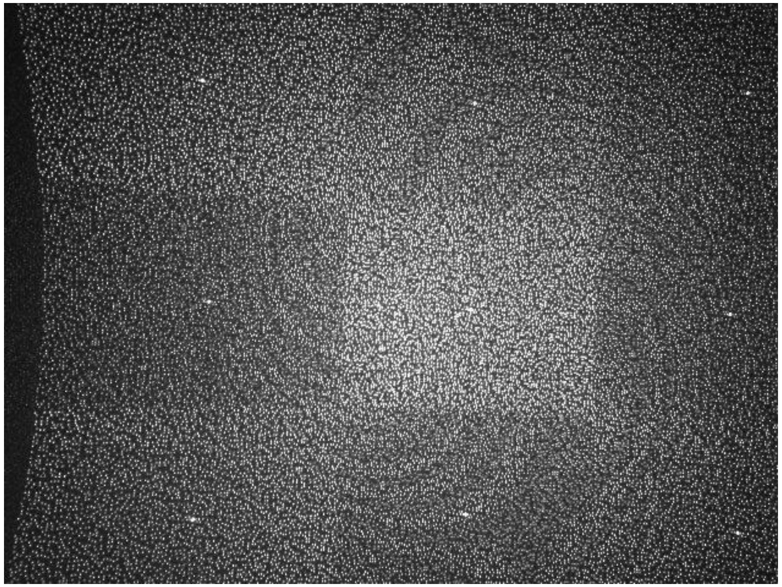


$$Z = \frac{B.f}{x_l - x_r} = \frac{B.f}{d}$$

5.2 RGB-D: Principle

RGB-D Cameras: Kinect

- **Structured Light:**
- Projects an infrared speckle pattern
- The projected pattern is then captured by an infrared camera in the sensor
- Compared part-by-part to reference patterns stored in the device.
- These patterns were captured previously at known depths.
- The sensor then estimates the per-pixel depth based on which reference patterns the projected pattern matches best.



5.2 RGB-D: Principle

UNIVERSITY OF TECHNOLOGY SYDNEY

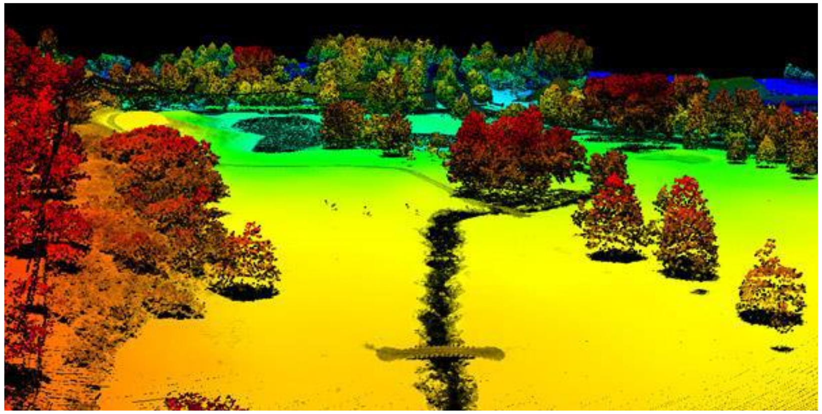
❖ RGB-D Cameras: Kinect

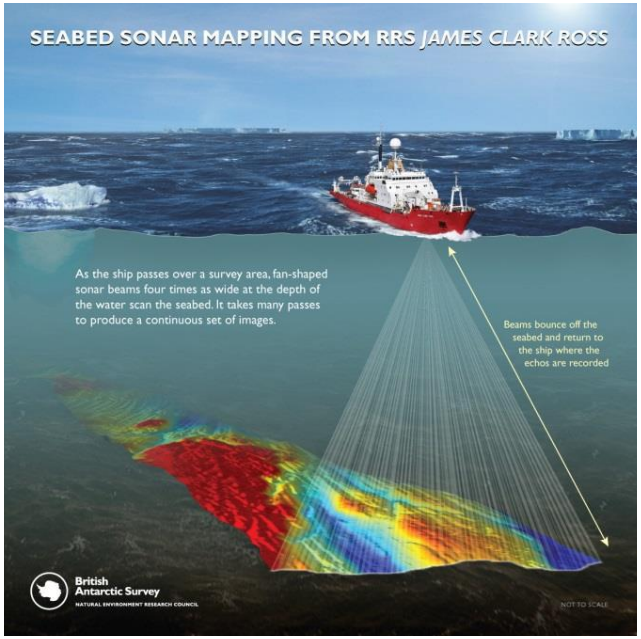
- **Point Cloud :**
- a collection of points in three dimensional space, where each point can have additional features associated with it.
- With an RGB-D sensor, the color can be one such feature.
- Approximated surface normals are also often stored with each point in a point cloud.

6. Time of Flight Sensors

❖ ToF Sensors:

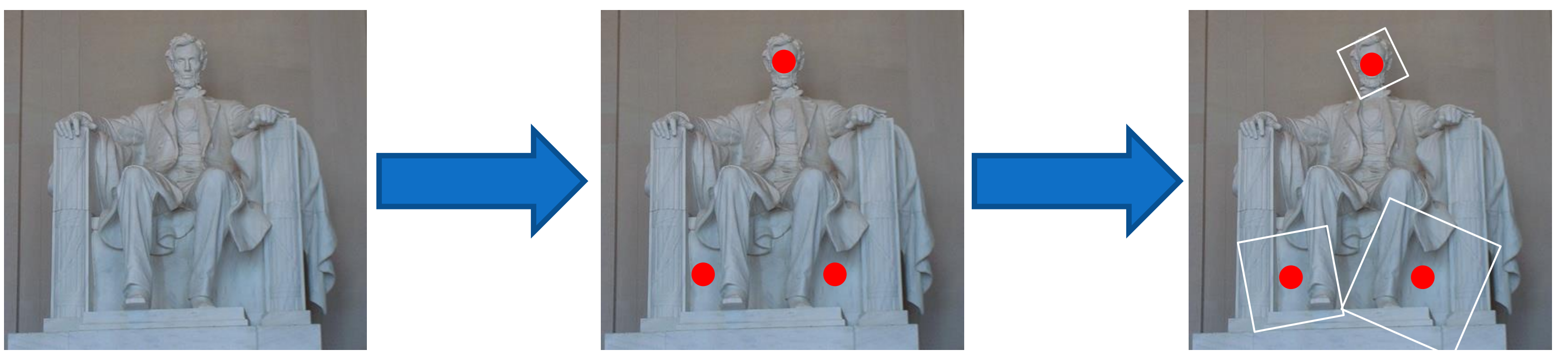
- Sonars
- Ultrasound
- Lidar
- Laser Measurement Systems
- Radar





7. Feature Extraction and Tracking

- Features:
  - edge, corner
- Extraction and matching
  - Harris, SIFT, SURF
- SIFT:
  - Features, descriptors



Test image

Detector: where are the local features?

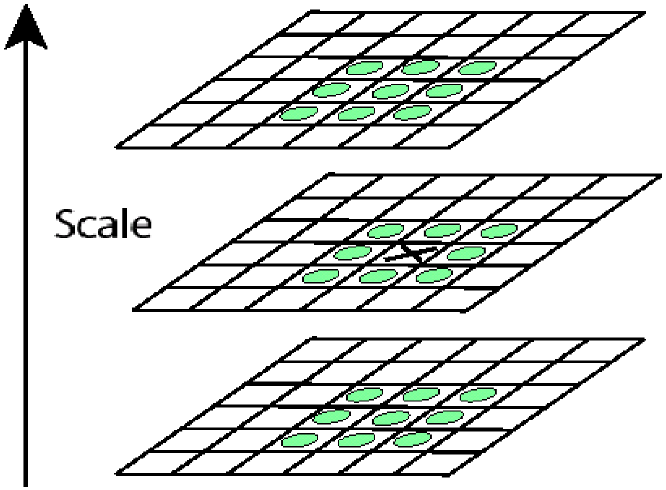
Descriptor: how to describe them?

7.1 SIFT: Feature

- Detect maxima and minima of difference-of-Gaussian in scale space

s+2 difference images.
top and bottom ignored.
s planes searched.

- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below

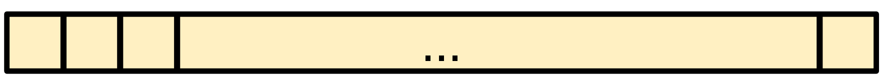


For each max or min found, output is the location and the scale.

7.1 SIFT: Descriptor

SIFT Keypoint Descriptor

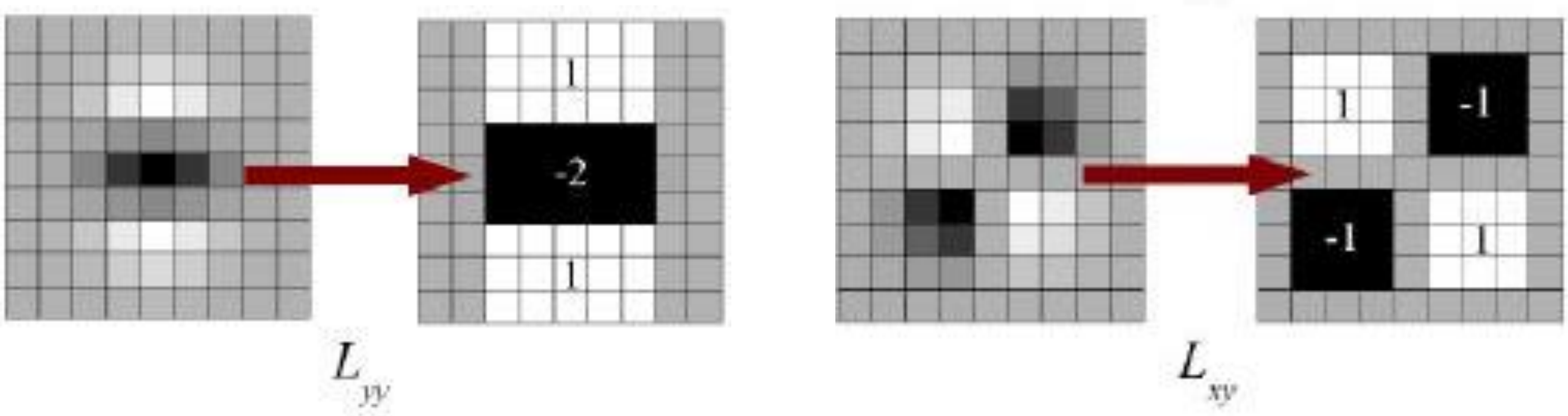
- use the normalized region about the keypoint
- compute gradient magnitude and orientation at each point in the region
- weight them by a Gaussian window overlaid on the circle
- create an orientation histogram over the 4 X 4 subregions of the window
- 4 X 4 descriptors over 16 X 16 sample array were used in practice. 4 X 4 times 8 directions gives **a vector of 128 values**.



7.2 SURF: Feature

❖ SURF: Speeded Up Robust Features

- Feature location
- SIFT: approximate Laplacian of Gaussian with Difference of Gaussian for finding scale-space.
- SURF: goes a little further and **approximates LoG with Box Filter**.

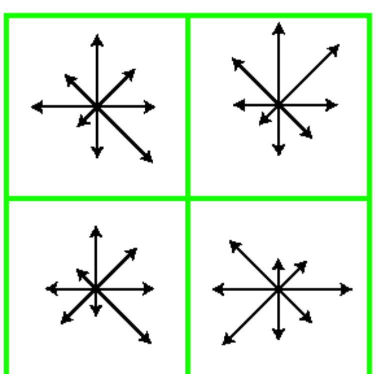


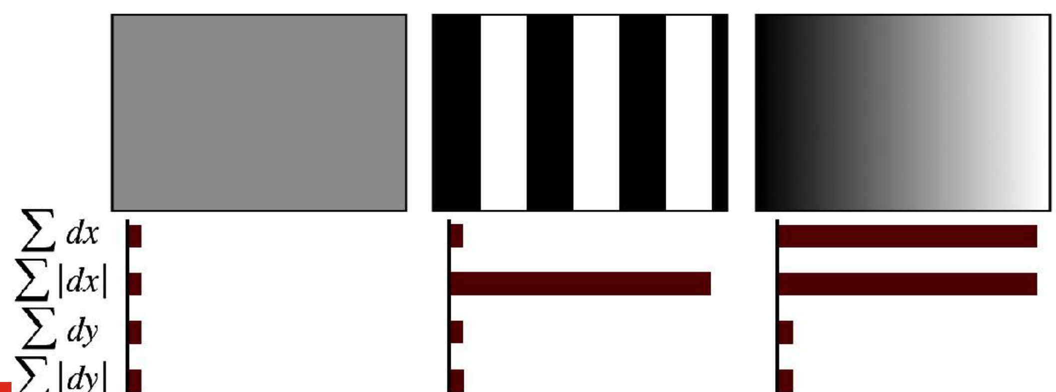
7.2 SURF: Descriptor

SURF: Speeded Up Robust Features

- Feature descriptor
- SIFT: 4 X 4 descriptors over 16 X 16 sample array. 4 X 4 times 8 directions gives **a vector of 128 values**.
- SURF: uses Wavelet responses in horizontal and vertical direction: 4x4 subregions, for each subregion **horizontal and vertical** wavelet responses are taken. **4x4x4=64**

$$v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right)$$





7.3 Harris Corner Detector

UNIVERSITY OF TECHNOLOGY SYDNEY

- Harris corner detector
- Second order moment matrix

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

- Consider the following example

- Symmetric matrix
- Sum over a small region around the interested point

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

- All the values are gradients in x (I_x) or y (I_Y) directions

- If either λ is close to 0, then this is not a corner, so look for locations where both are large

RANSAC: RANdom SAmple Consensus

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a sample of *s* data points from S and instantiate the model from this subset.

(ii) Determine the set of data points S_i which are within a distance threshold *t* of the model. The set S_i is the consensus set of samples and defines the inliers of S.

(iii) If the subset of S_i is greater than some threshold *T*, re-estimate the model using all the points in S_i and terminate

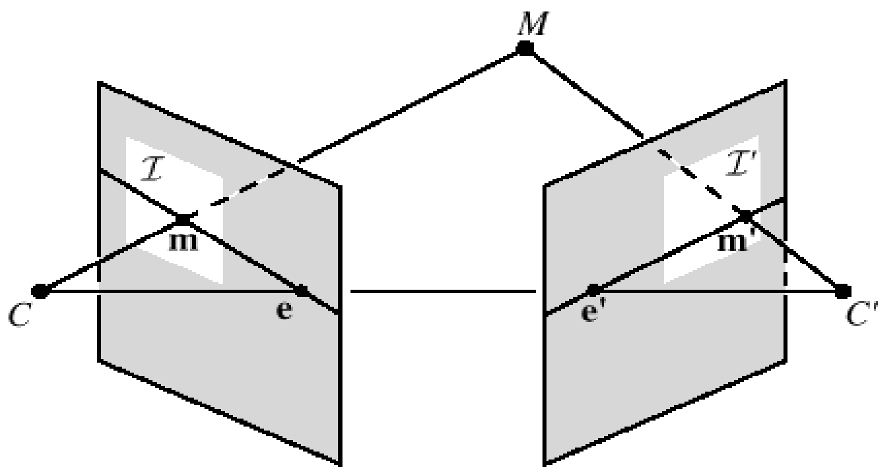
(iv) If the size of S_i is less than *T*, select a new subset and repeat the above.

(v) After *N* trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i

7.4 RANSAC Outlier Removal

UNIVERSITY OF TECHNOLOGY SYDNEY

- **RANSAC outlier removal with epipolar constraint**



$$\mathbf{x}^T \mathbf{F} \mathbf{x}' = 0$$

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = 0$$

Solution to the state space model

State space model

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \qquad x(t_0) = x_0 \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

The solution is

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau$$

where

$$e^{At} = \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}$$

<u>One dimensional example</u>

$$\dot{x}(t) = -x(t), \; x(0) = x_0.$$

UNIVERSITY OF TECHNOLOGY SYDNEY

Stability

The system

$$\dot{x}(t) = Ax(t) \qquad x(t_0) = x_0$$

is called asymptotically stable if for any initial state, the state x(t) converges to zero as t increases indefinitely.

Simple example 1

$$\dot{x} = -x, \quad x(0) = x_0$$

The solution is

$$x(t) = e^{-t} x_0$$

Simple example 2

$$\dot{x} = x, \quad x(0) = x_0$$

The solution is

$$x(t) = e^{t} x_0$$

Condition of stability: all the eigenvalues of A have negative real parts.

Eigenvalues of A are the solutions of the equation $|\lambda I - A| = 0$

Check for Controllability

Theorem: The state space model

$$\dot{x}(t) \;=\; Ax(t) + Bu(t)$$

is completely controllable if and only if the matrix

$$[B \; AB \; A^2B \; \cdots \; A^{n-1}B]$$

has full row rank.

9. Linear Discrete-Time System

- Comparing to Linear continuous-time system:
  - Discrete-time system:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ x(0) = x_0 \end{cases}$$

  - Continuous-time system:

$$\boxed{\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \qquad x(t_0) = x_0 \\ y(t) &= Cx(t) + Du(t) \end{aligned}}$$

9.1 Linear Discrete-Time System

- Linear discrete-time system:
  - In compact matrix form:

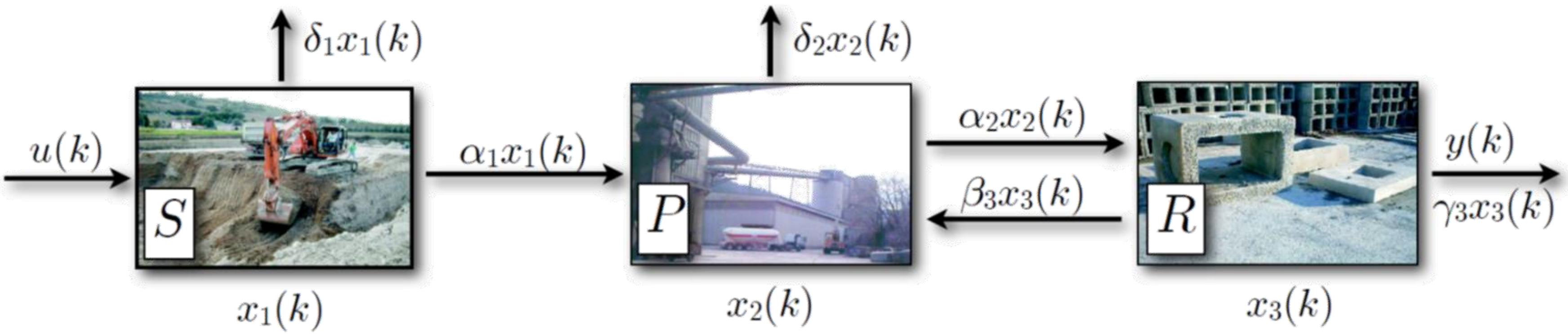
$$\begin{cases} x(k+1) &= Ax(k) + Bu(k) \\ x(0) &= x_0 \end{cases}$$

  - The solution is

$$x(k) = \underbrace{A^k x_0}_{\text{natural response}} + \underbrace{\sum_{i=0}^{k-1} A^i B u(k-1-i)}_{\text{forced response}}$$

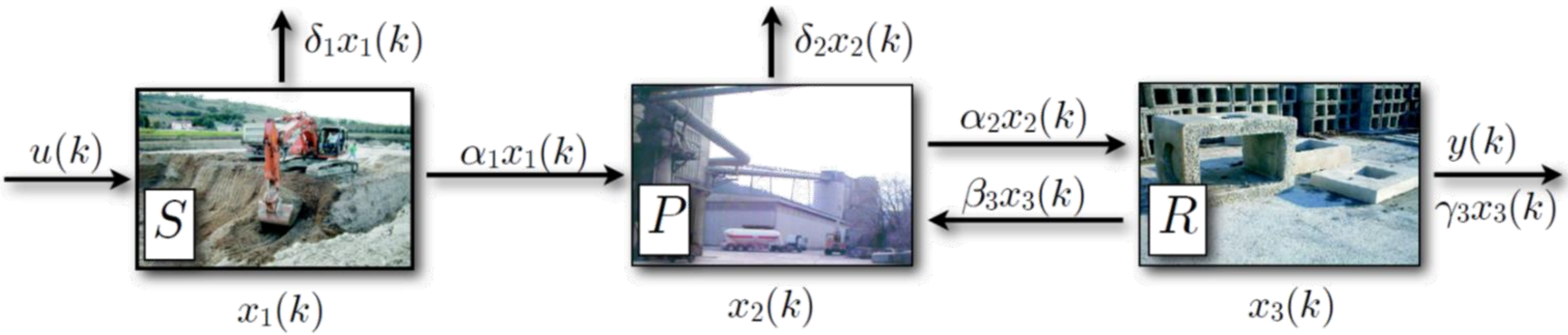
9.1 Linear Discrete-Time System

- Activity-2: Supply chain
- Problem Statement:
  - S purchases the quantity u(k) of raw material at each month k
  - A fraction δ_1 of raw material is discarded, a fraction α_1 is shipped to producer P
  - A fraction α_2 of product is sold by P to retailer R, a fraction δ_2 is discarded
  - retailer R returns a fraction β_3 of defective products every month, and sells a fraction γ_3 to customers



9.1 Linear Discrete-Time System

❖ Activity-2: Supply chain



$$\begin{cases} x_1(k+1) &= (1-\alpha_1-\delta_1)x_1(k)+u(k) \\ x_2(k+1) &= \alpha_1 x_1(k)+(1-\alpha_2-\delta_2)x_2(k)+\beta_3 x_3(k) \\ x_3(k+1) &= \alpha_2 x_2(k)+(1-\beta_3-\gamma_3)x_3(k) \\ y(k) &= \gamma_3 x_3(k) \end{cases}$$

k	month counter
$x_1(k)$	raw material in S
$x_2(k)$	products in P
$x_3(k)$	products in R
$y(k)$	products sold to customers

9.2 Discretization

UNIVERSITY OF TECHNOLOGY SYDNEY

- Approximate sampling: Euler's method

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \qquad x(t_0) = x_0 \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ x(0) = x_0 \end{cases}$$

- Approximation

$$\dot{x}(kT_s) \approx \frac{x((k+1)T_s) - x(kT_s)}{T_s}$$

- Sampling

$$\dot{x}(t) = Ax(t) + Bu(t):$$

$$x((k+1)T_s) = (I + T_s A)x(kT_s) + T_s Bu(kT_s)$$

$$\bar{A} \triangleq I + AT_s, \quad \bar{B} \triangleq T_s B, \quad \bar{C} \triangleq C, \quad \bar{D} \triangleq D$$

9.2 Discretization

- Approximate sampling: Tustin's discretization method

  - Approximation by applying the trapezoidal rule

$$x(k+1)-x(k)=\int_{kT_s}^{(k+1)T_s}\dot{x}(t)dt=\int_{kT_s}^{(k+1)T_s}(Ax(t)+Bu(t))dt$$

$$\approx\frac{T_s}{2}(Ax(k)+Bu(k)+Ax(k+1)+Bu(k))$$ (trapezoidal rule)

  - Then

$$(I-\frac{T_s}{2}A)x(k+1)=(I+\frac{T_s}{2})x(k)+T_sBu(k)$$

$$x(k+1)=\left(I-\frac{T_s}{2}A\right)^{-1}\left(I+\frac{T_s}{2}A\right)x(k)+\left(I-\frac{T_s}{2}A\right)^{-1}T_sBu(k)$$

9.3 Stability

Stability of discrete-time linear systems

- Since the natural response of $x(k+1) = Ax(k) + Bu(k)$ is $x(k) = A^k x_0$ the stability properties depend only on A. We can therefore talk about system stability of a discrete-time linear system (A,B,C,D)

Theorem:

Let $\lambda_1, \ldots, \lambda_m$, $m \leq n$ be the eigenvalues of $A \in \mathbb{R}^{n\times n}$. The system $x(k+1) = Ax(k) + Bu(k)$ is

- asymptotically stable iff $|\lambda_i| < 1, \forall i = 1, \ldots, m$
- (marginally) stable if $|\lambda_i| \leq 1, \forall i = 1, \ldots, m$, and the eigenvalues with unit modulus have equal algebraic and geometric multiplicity [a]
- unstable if $\exists\ i$ such that $|\lambda_i| > 1$

[a] Algebraic multiplicity of λ_i = number of coincident roots λ_i of $\det(\lambda I - A)$. Geometric multiplicity of λ_i = number of linearly independent eigenvectors v_i, $Av_i = \lambda_i v_i$

- The stability properties of a discrete-time linear system only depend on the modulus of the eigenvalues of matrix A

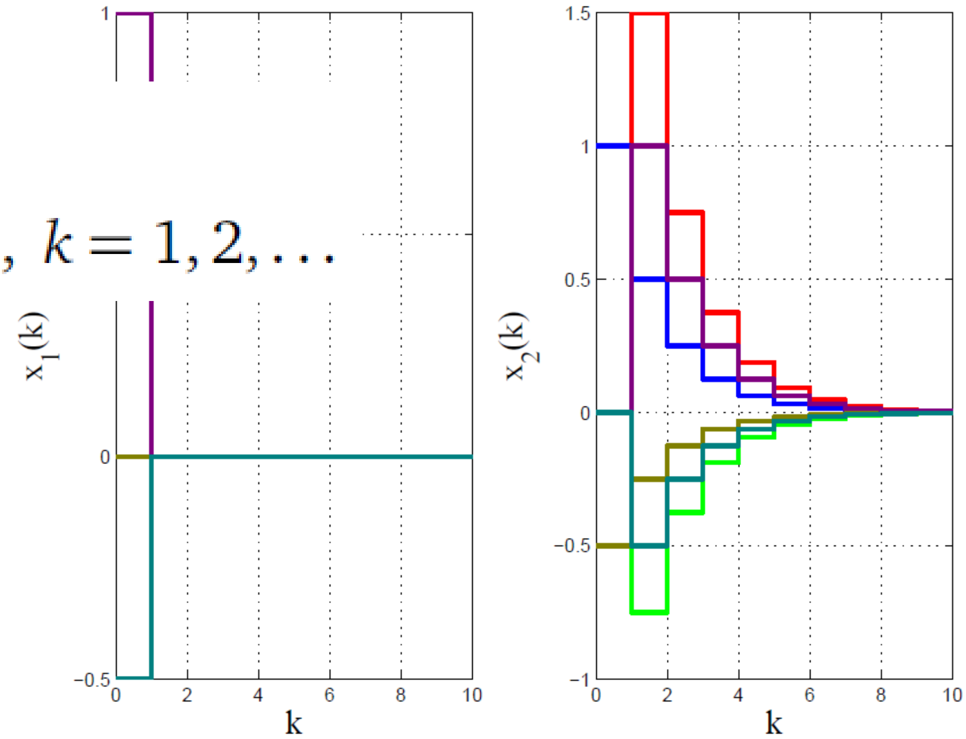
9.3 Stability

- Activity-4:

$$\begin{cases} x(k+1) = \begin{bmatrix} 0 & 0 \\ 1 & \frac{1}{2} \end{bmatrix} x(k) \\ x(0) = \begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix} \end{cases}$$

  - Solution?

$$\begin{cases} x_1(k) &= 0,\ k = 1, 2, \ldots \\ x_2(k) &= \left(\frac{1}{2}\right)^{k-1} x_{10} + \left(\frac{1}{2}\right)^{k} x_{20},\ k = 1, 2, \ldots \end{cases}$$



  - Stability?
  - Eigenvalues of A: {0,1/2}
  - asymptotically stable

9.4 Controllability

- Controllability:
  - *In order to be able to do whatever we want with the given dynamic system under control input, the system must be controllable.*
- Check for Controllability:
  - **Theorem:** The state space model

$$x(t+1) = Ax(t) + Bu(t),\ x(t) \in \mathbf{R}^n$$

  - is completely controllable if and only if the matrix

$$\mathcal{C}_t = \begin{bmatrix} B & AB & \cdots & A^{t-1}B \end{bmatrix}$$

  - has full row rank.

9.4 Controllability

- Activity-6:

$$x(t+1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t)$$

- Is the system controllable?

- Controllability Matrix

$$\mathcal{C} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

- System is not controllable.

9.5 Optimal Control

UNIVERSITY OF TECHNOLOGY SYDNEY

- Optimal Control:
  - A discrete-time linear system

$$x(k+1) = Ax(k) + Bu(k)$$

  - Design state feedback control

$$u(k) = -Kx(k)$$

  - Such that the ***Performance index***:

$$J(U) = \sum_{\tau=0}^{N-1} \left(x_\tau^T Q x_\tau + u_\tau^T R u_\tau \right) + x_N^T Q_f x_N$$

  - is minimized.
  - Where

$$U = (u_0, \ldots, u_{N-1})$$

$$Q = Q^T \geq 0, \qquad Q_f = Q_f^T \geq 0, \qquad R = R^T > 0$$

UNIVERSITY OF TECHNOLOGY SYDNEY

Linear Quadratic Regulator (LQR):

- The algebraic Riccati equation (ARE)

$$P_{\mathrm{ss}} = Q + A^T P_{\mathrm{ss}} A - A^T P_{\mathrm{ss}} B (R + B^T P_{\mathrm{ss}} B)^{-1} B^T P_{\mathrm{ss}} A$$

- P can be found by iterating the Riccati recursion, or by direct methods

- LQR optimal input is approximately a linear, constant state feedback

$$u_t = K_{\mathrm{ss}} x_t, \qquad K_{\mathrm{ss}} = -(R + B^T P_{\mathrm{ss}} B)^{-1} B^T P_{\mathrm{ss}} A$$

- It is very widely used in practice.

10. Nonlinear Discrete-Time System

UNIVERSITY OF TECHNOLOGY SYDNEY

❖ Nonlinear discrete-time state-space models:

- Nonlinear discrete-time model:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \qquad k = 0, 1, 2, \ldots \\ y(k) &= h(x(k), u(k)) \end{aligned}$$

- *u(k)*: input at time k , an m-dimensional column vector.
- *y(k)*: output at time k , a p-dimensional column vector.
- *x(k)*: state at time k , an n-dimensional column vector.
- The model is said to be *n-th* order.
- For a given initial value $x(k_0) = x_0$, always has a unique solution.

10.1 Equilibrium

- A constant trajectory, generated by a constant input function, is called *equilibrium*.
  - Equilibrium point -- a point where the system can stay forever without moving.

- Given a constant input $\bar{u}$, the equilibria are solutions of the following equations:

$$\bar{x} = f(\bar{x}, \bar{u})$$
$$\bar{y} = h(\bar{x}, \bar{u})$$

10.1 Equilibrium

- How to calculate equilibria?
  - Solve the solution of *x(k+1)-x(k)=0.*
- Example:

$$x(k+1) = \frac{1}{4}x(k)^2 + x(k) - 2u(k)$$

  - When $u(k) \equiv 0.5$, compute the equilibrium points.
- Solution:

$$x(k+1) - x(k) = \frac{1}{4}x(k)^2 - 1 = 0$$

  - The solutions are $\bar{x} = \pm 2$.

10.1 Equilibrium

- Activity 1:

- Calculate the equilibria?

$$x_1(k+1) = \frac{1}{4}x_1(k)^2 + x_1(k) - 2u_1(k)$$
$$x_2(k+1) = \frac{1}{4}x_2(k)^2 + x_1(k) - 2u_2(k)$$

  - When $u_1(k) \equiv 0.5$ and $u_2(k) \equiv 1$, compute the equilibrium points.

10.1 Equilibrium

UNIVERSITY OF TECHNOLOGY SYDNEY

- Activity 1:

- Solution:

$$x_1(k+1) - x_1(k) = \frac{1}{4}x_1(k)^2 - 1 = 0$$

$$x_2(k+1) - x_2(k) = \frac{1}{4}x_2(k)^2 + x_1 - x_2 - 2 = 0$$

  - The solutions are $\bar{x}_1 = \pm 2$.
  - When $\bar{x}_1 = 2$.

$$\frac{1}{4}x_2(k)^2 - x_2 = 0$$

  - The solutions are $\bar{x}_2 = 0 \; or \; \overline{x}_2 = 4$.

10.2 Nonlinear Discrete-Time Systems

UNIVERSITY OF TECHNOLOGY SYDNEY

- Linearization
- Stability
- State Feedback Control
- All at Equilibrium Point

11. Visual Servoing

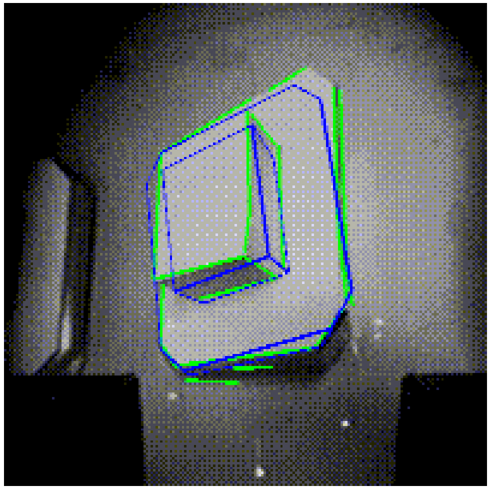
- End-Effector Mounted
- Basic Components of Visual Servoing
  - **The aim of all vision-based control schemes is to minimize an error, which is typically defined by**

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

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.*



11.1 Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing

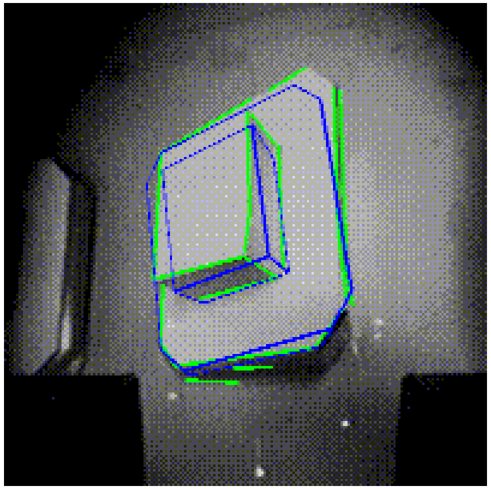
$$\dot{\mathbf{e}} = \mathbf{L_e v}_c$$

How to solve

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

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

Linear Least Squares



11.1 Visual Servoing

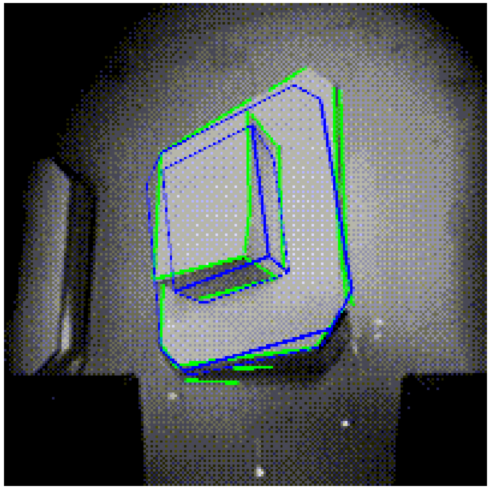
- End-Effector Mounted
- Basic Components of Visual Servoing

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

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.

11.1 Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing

Camera system

3D point (X,Y,Z)

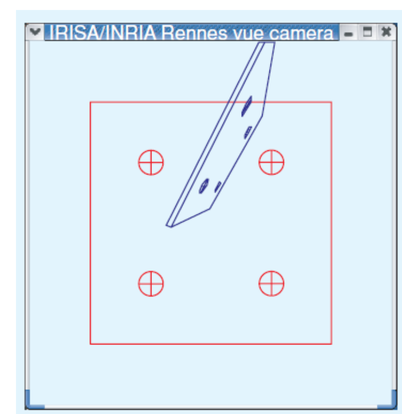
2D point (x,y)

measurement m=(u,v)

intrinsic parameters (cu,cv,f)

Hc

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



11.1 Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing

2D point (x,y)

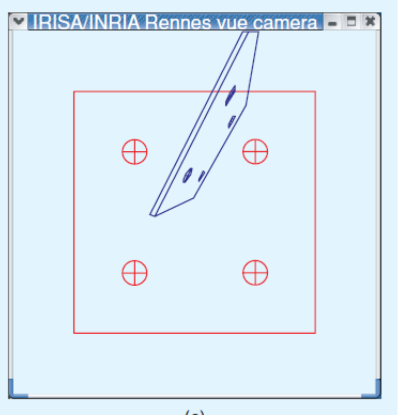
$$\begin{cases} x & = & X/Z = (u - c_u)/f \\ y & = & Y/Z = (v - c_v)/f, \end{cases}$$

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

We take s = (x,y)

How to compute

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



11.1 Visual Servoing

UNIVERSITY OF TECHNOLOGY SYDNEY

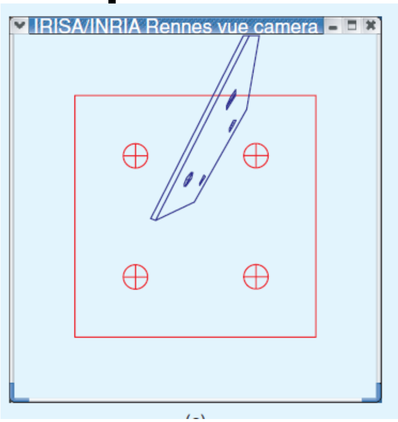
- End-Effector Mounted
- Basic Components of Visual Servoing

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

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



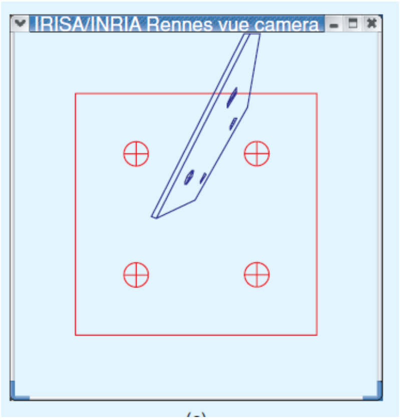
11.1 Visual Servoing

UNIVERSITY OF TECHNOLOGY SYDNEY

- 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

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

11.1 Visual Servoing

- End-Effector Mounted
- Basic Components of Visual Servoing

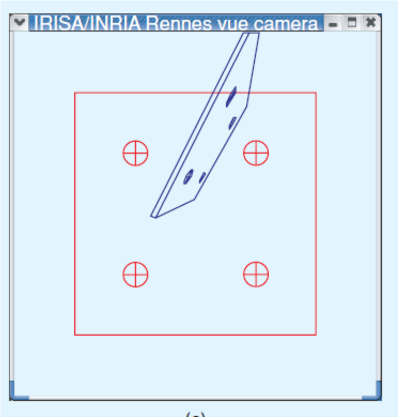
$$\begin{cases} \dot{x} = -v_x/Z + xv_z/Z + xy\omega_x - (1 + x^2)\omega_y + y\omega_z \\ \dot{y} = -v_y/Z + yv_z/Z + (1 + y^2)\omega_x - xy\omega_y - x\omega_z \end{cases}$$

Which can be rewritten

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

Where

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



11.1 Visual Servoing

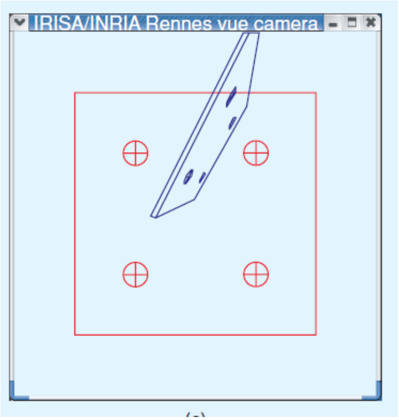
- End-Effector Mounted
- Basic Components of Visual Servoing

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

Must estimate or approximate the value of Z

At least 3 points are necessary

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



Activity-3:

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

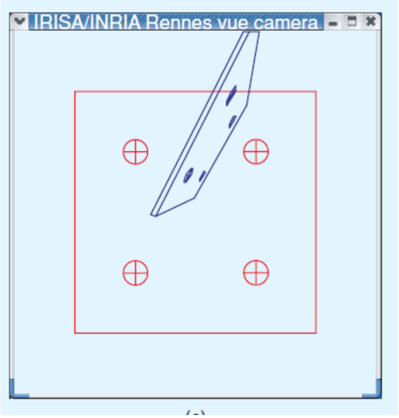
  - Pose: r = (0,0,0) => R = I, T = [10,20,2]

- 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?

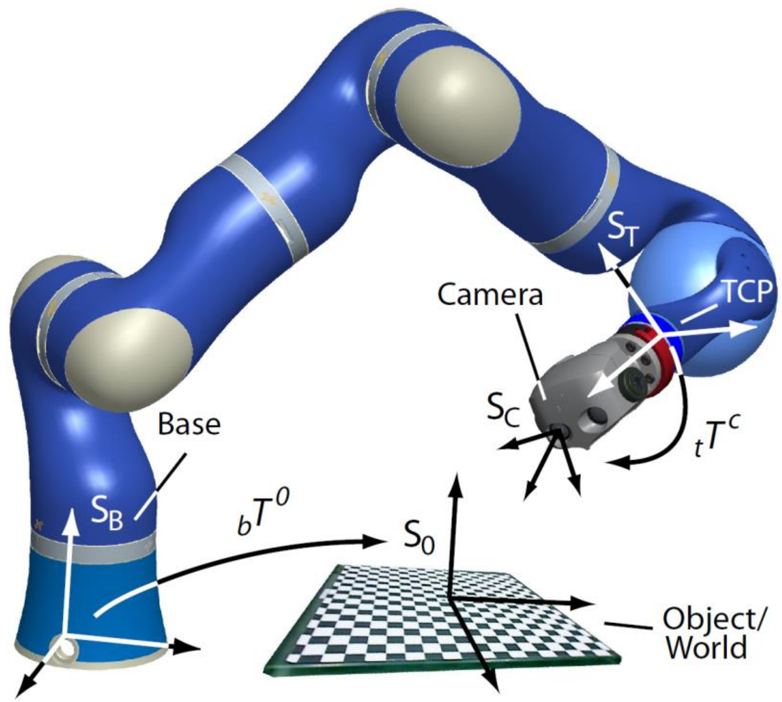


11.2 HAND-EYE CALIBRATION

UNIVERSITY OF TECHNOLOGY SYDNEY

❖ Hand-eye Calibration

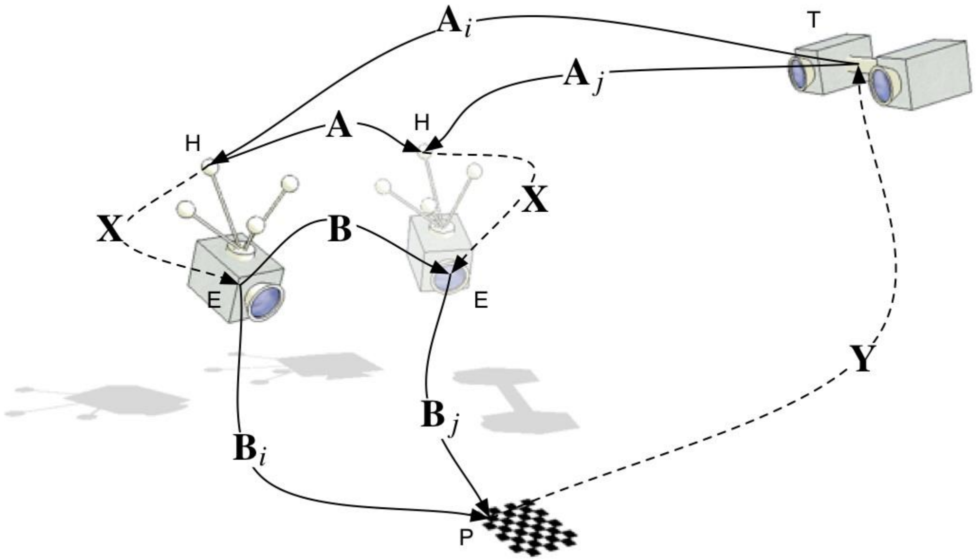
- Relative Pose
- Hand (end effector)
- Eye (Camera)

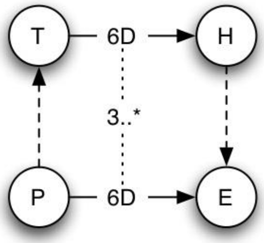


UNIVERSITY OF TECHNOLOGY SYDNEY

- Hand-eye Calibration

$$A*X = X*B$$





THANK YOU

Questions?