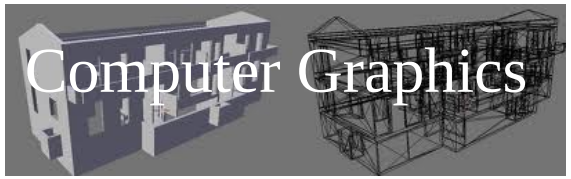
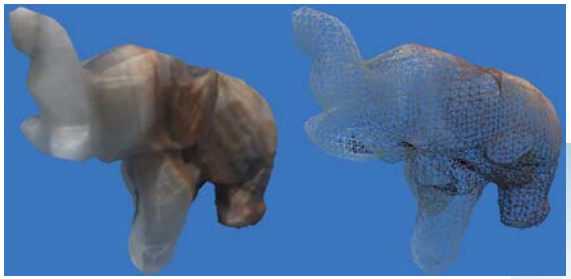
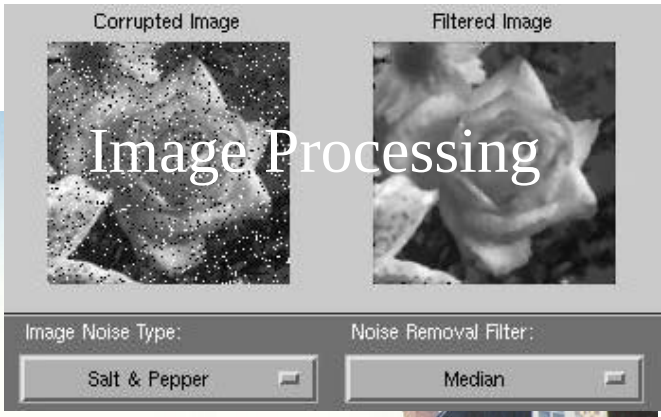
Robot vision review

Martin
Jagersand

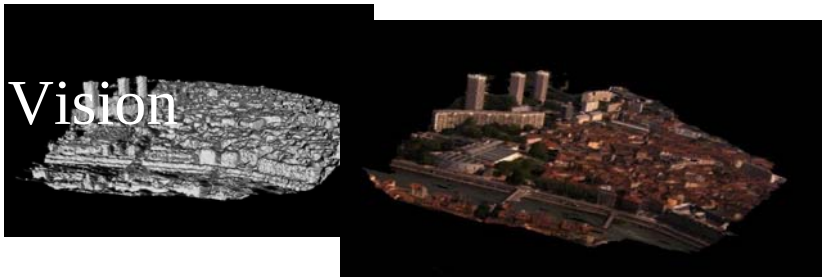
What is Computer Vision?



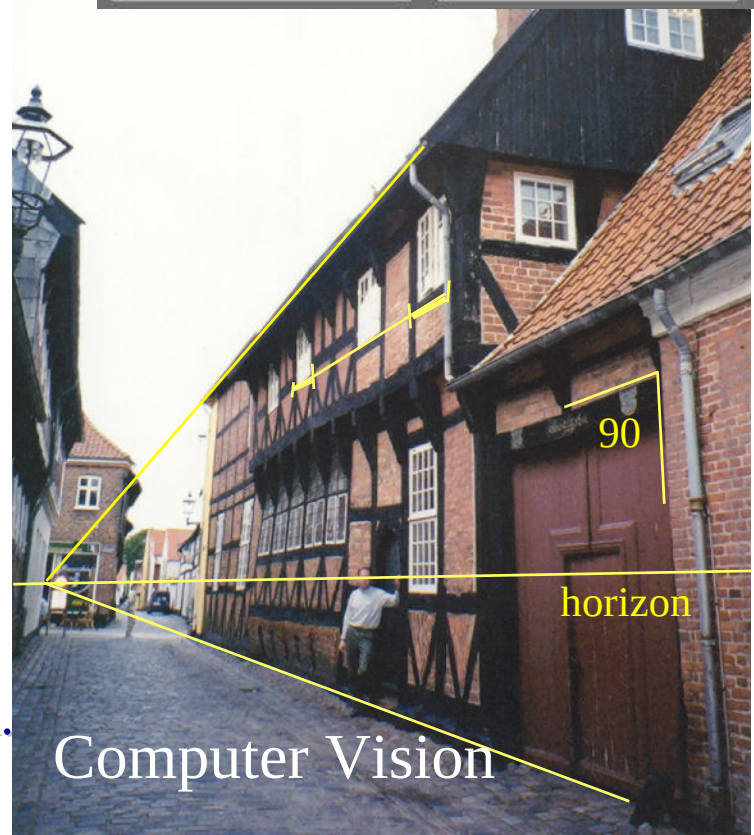




- Three Related fields
  - Image Processing: Changes 2D images into other 2D images
  - Computer Graphics: Takes 3D models, renders 2D images
  - Computer vision: Extracts scene information from 2D images and video
    - e.g. Geometry, "Where" something is in 3D,
    - Objects "What" something is"
- What information is in a 2D image?
- What information do we need for 3D analysis?

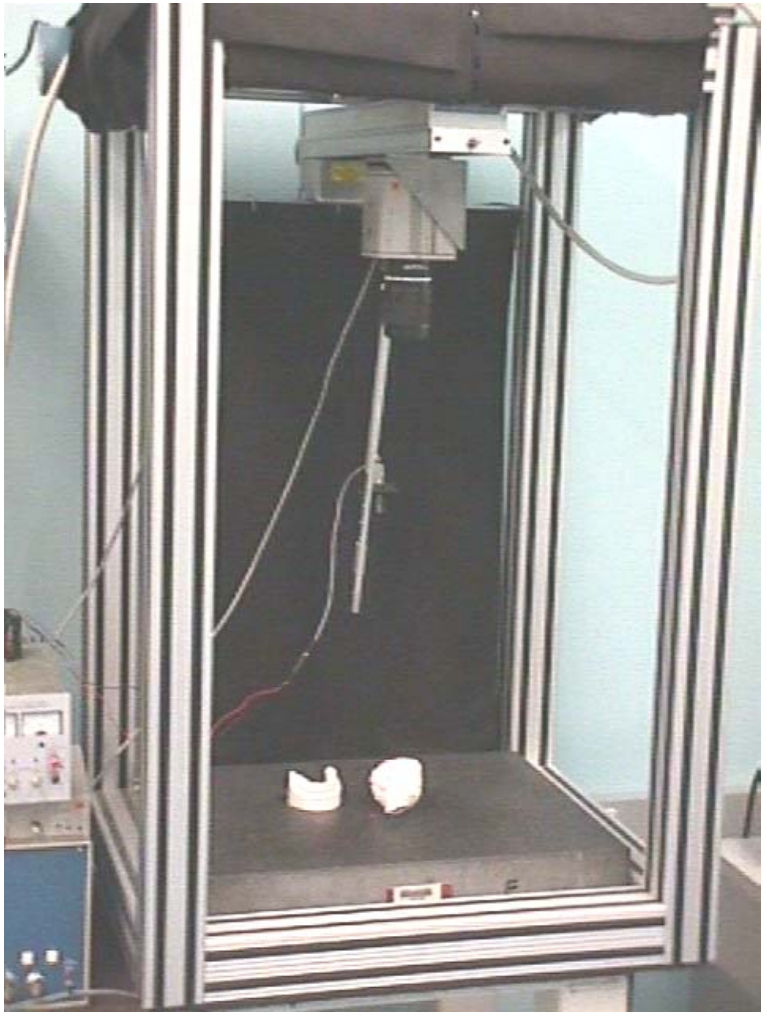


V hard. Only special cases can be solved.

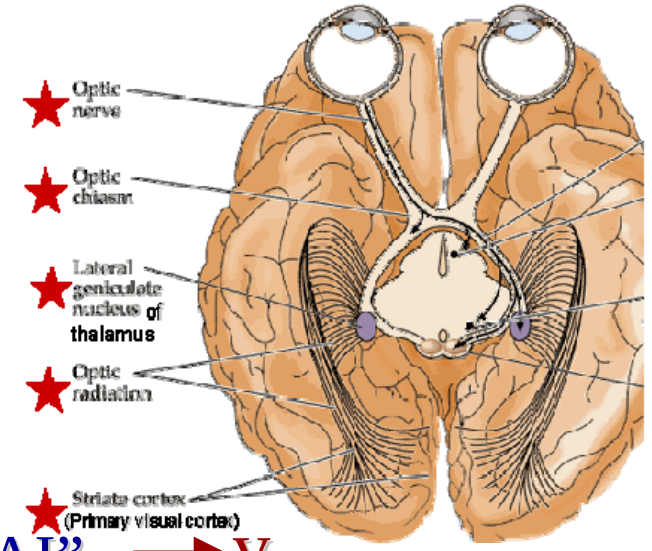


Machine Vision

- 3D Camera vision in general environments hard
- Machine vision:
  - Use engineered environment
  - Use 2D when possible
  - Special markers/LED
  - Can buy working system!
- Photogrammetry:
  - Outdoors
  - 3D surveying using cameras



Vision



- Full: Human vision
  - We don't know how it works in detail
- Limited vision: Machines, Robots, "AI"

What is the most basic useful information we can get from a camera?

1. Location of a dot (LED/Marker) [u,v] = f(I)
2. Segmentation of object pixels

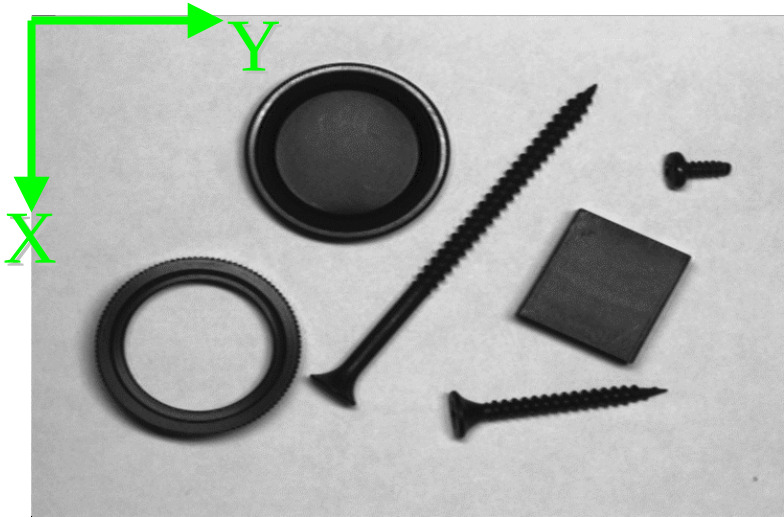
All of these are 2D image plane measurements!

What is the best camera location?

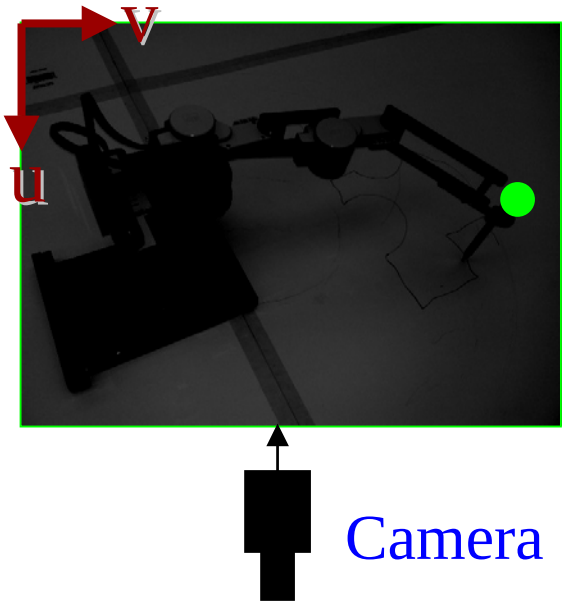
Usually overhead pointing straight down

Adjust cam position so pixel [u,v] = s[X,Y].

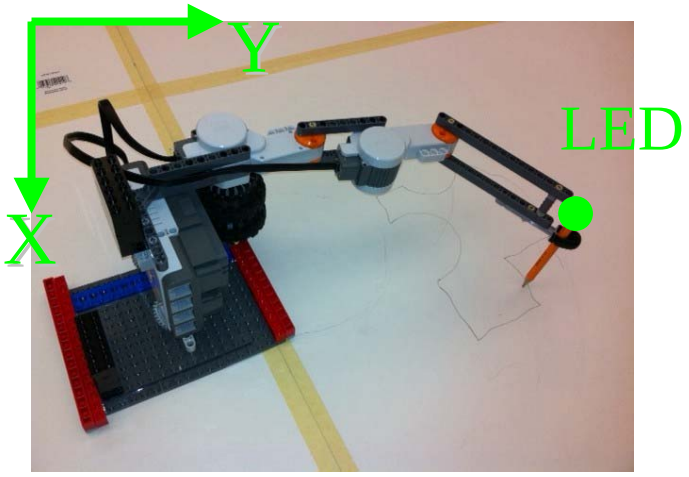
Pixel coordinates are scaled world coord



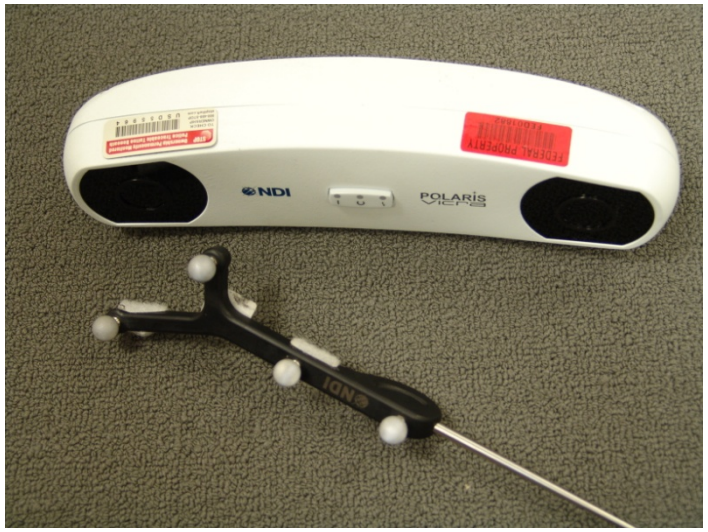
Tracking LED special markers



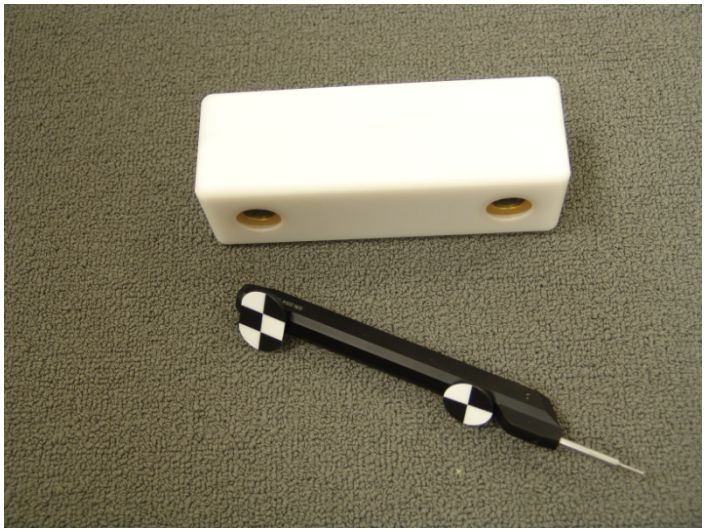
- Put camera overhead pointing straight down on worktable.
  - Adjust cam position so pixel [u,v] = s[X,Y]. Pixel coordinates are scaled world coord
  - Lower brightness so LED brighterest
- Put LED on robot end-effector
- Detection algorithm:
  - Threshold brightest pixels I(u,v)>200
  - Find centroid [u,v] of max pixels
- Variations:
  - Blinking LED can enhance detection in ambient light.
  - Different color LED's can be detected separately from R,G,B color video.



Commercial tracking systems



Polaris Vicra infra-red system
(Northern Digitial Inc.)



MicronTracker visible light system
(Claron Technology Inc.)

Commercial tracking system

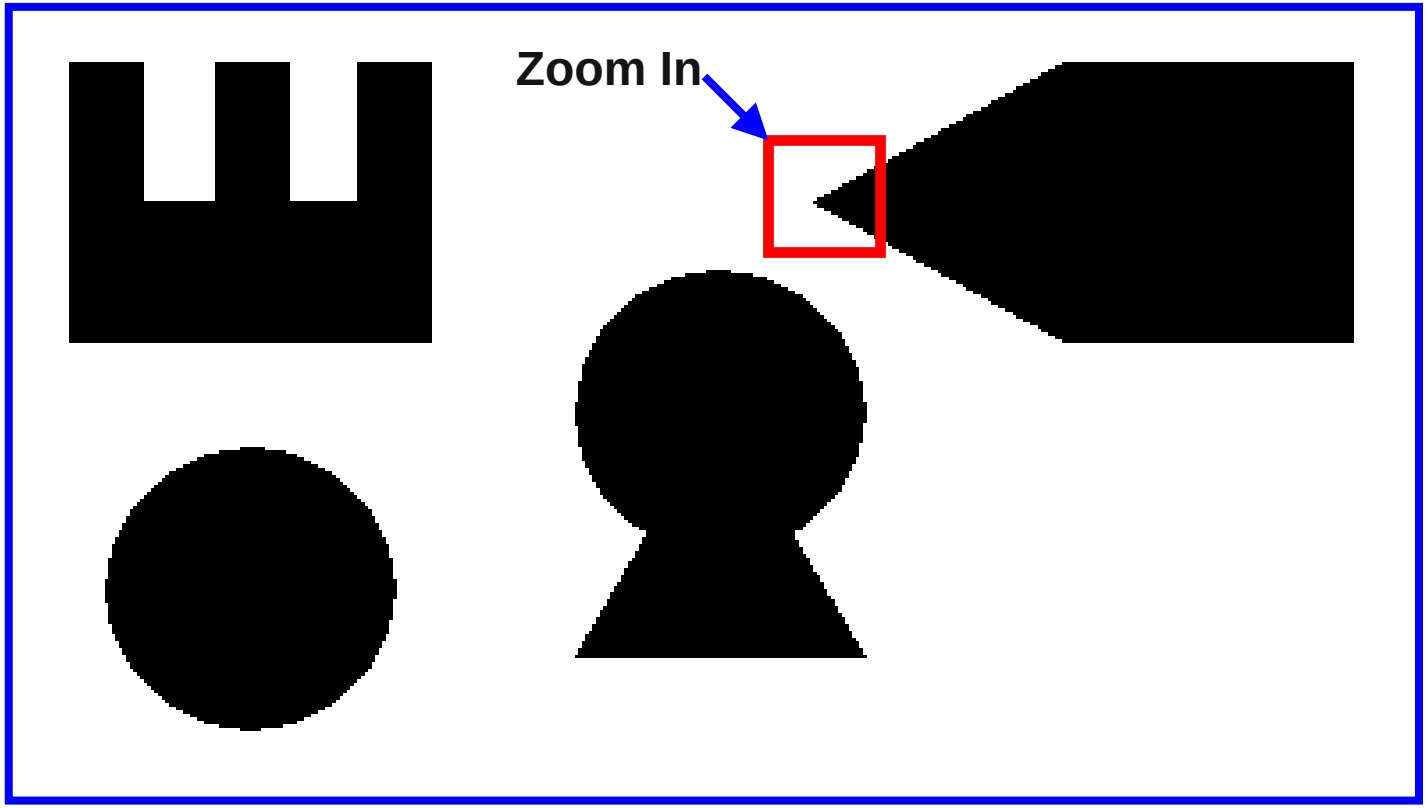
Images acquired by the Polaris Vicra infra-red stereo system:

left image

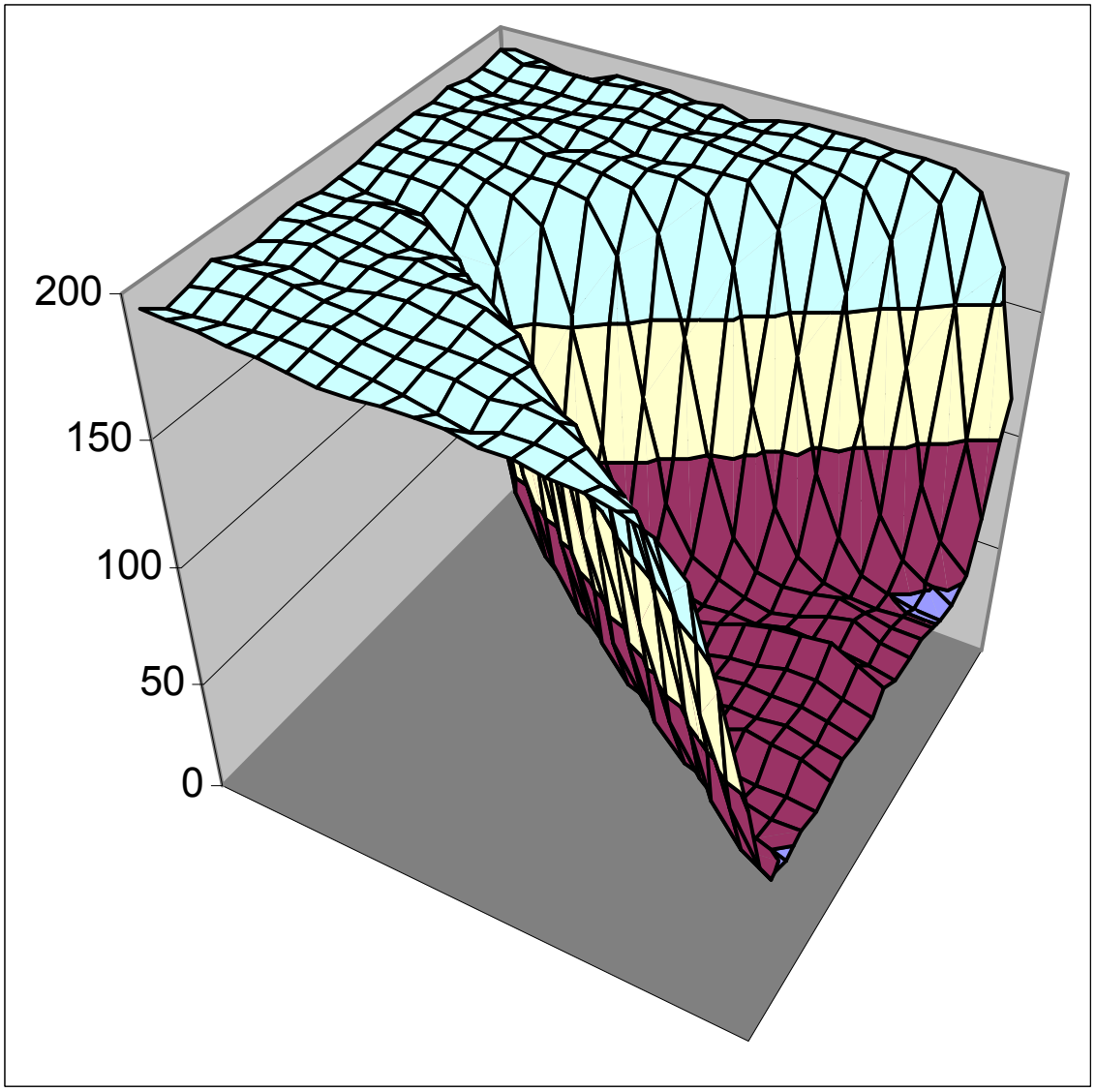
right image

IMAGE SEGMENTATION

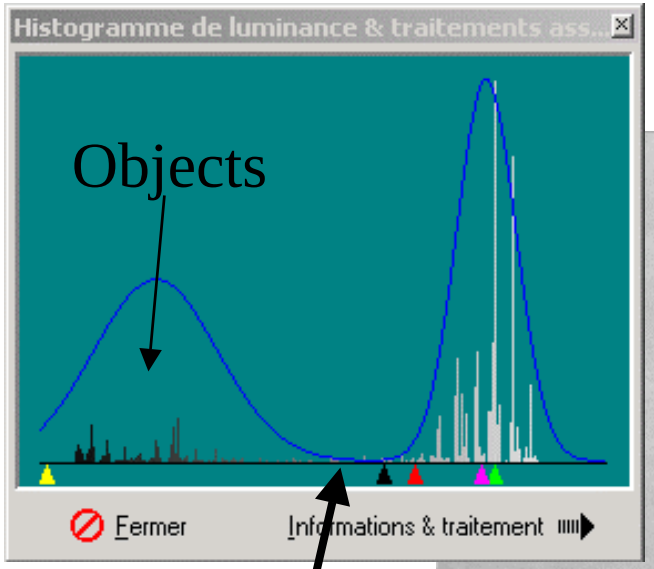
- How many *"objects"* are there in the image below?
- Assuming the answer is "4", what exactly defines an object?



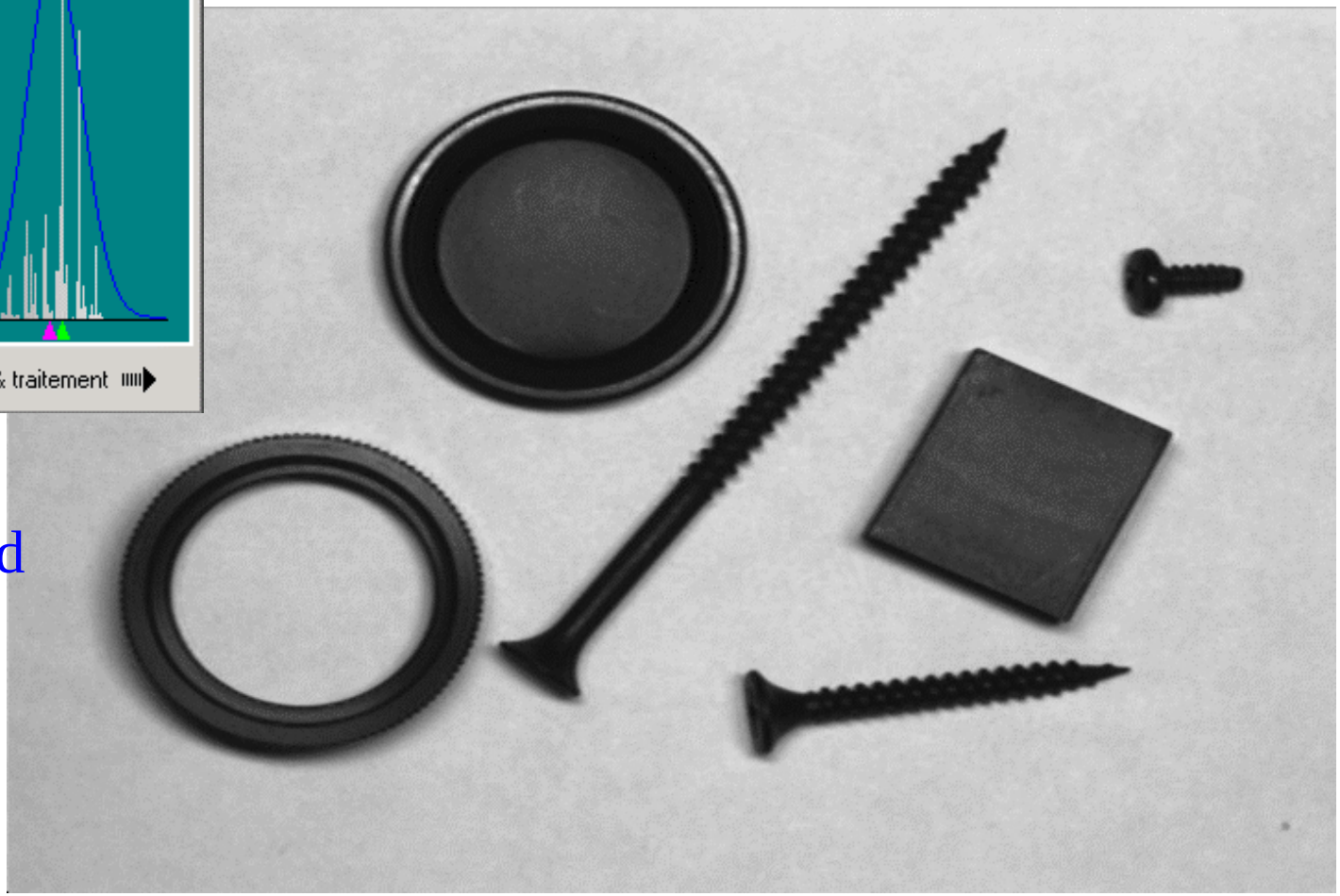
8 BIT GRAYSCALE IMAGE



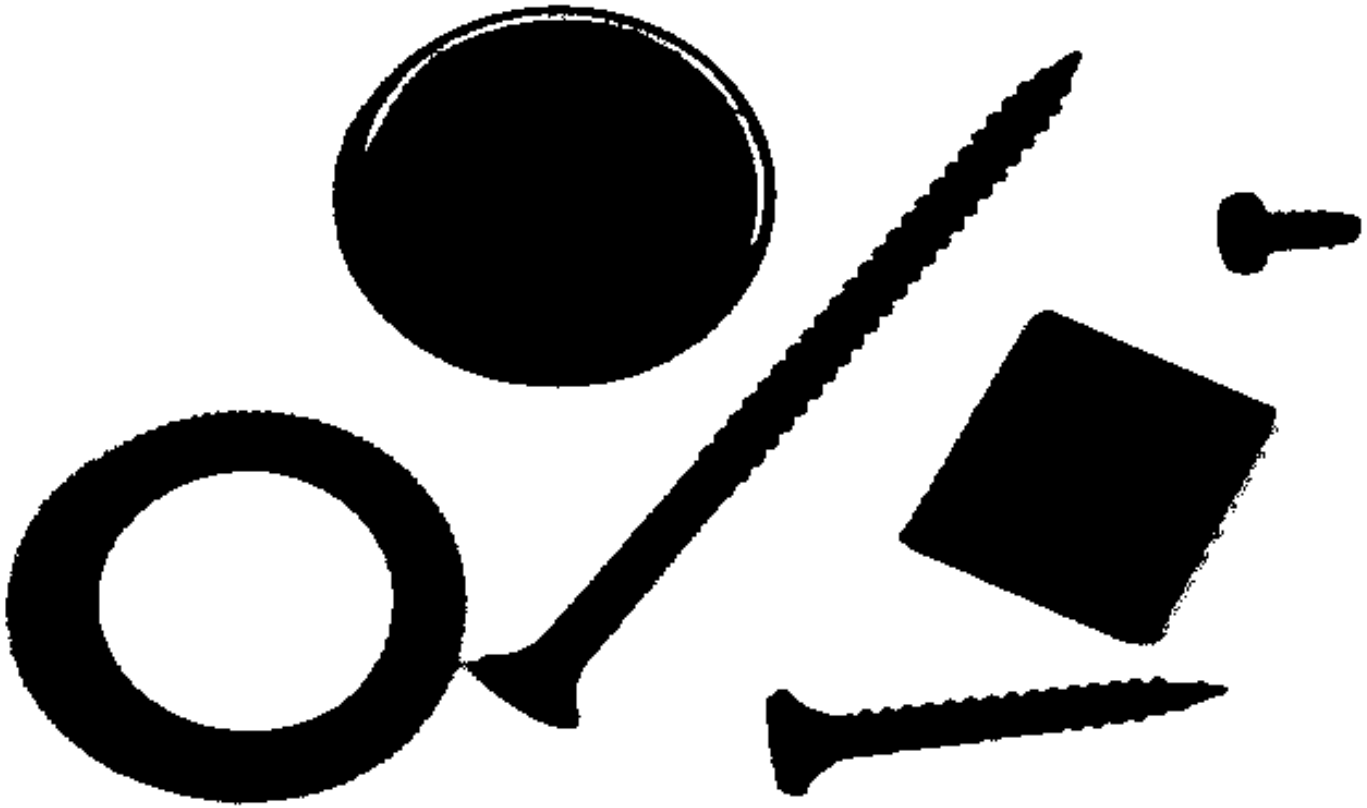
GRAY LEVEL THRESHOLDING



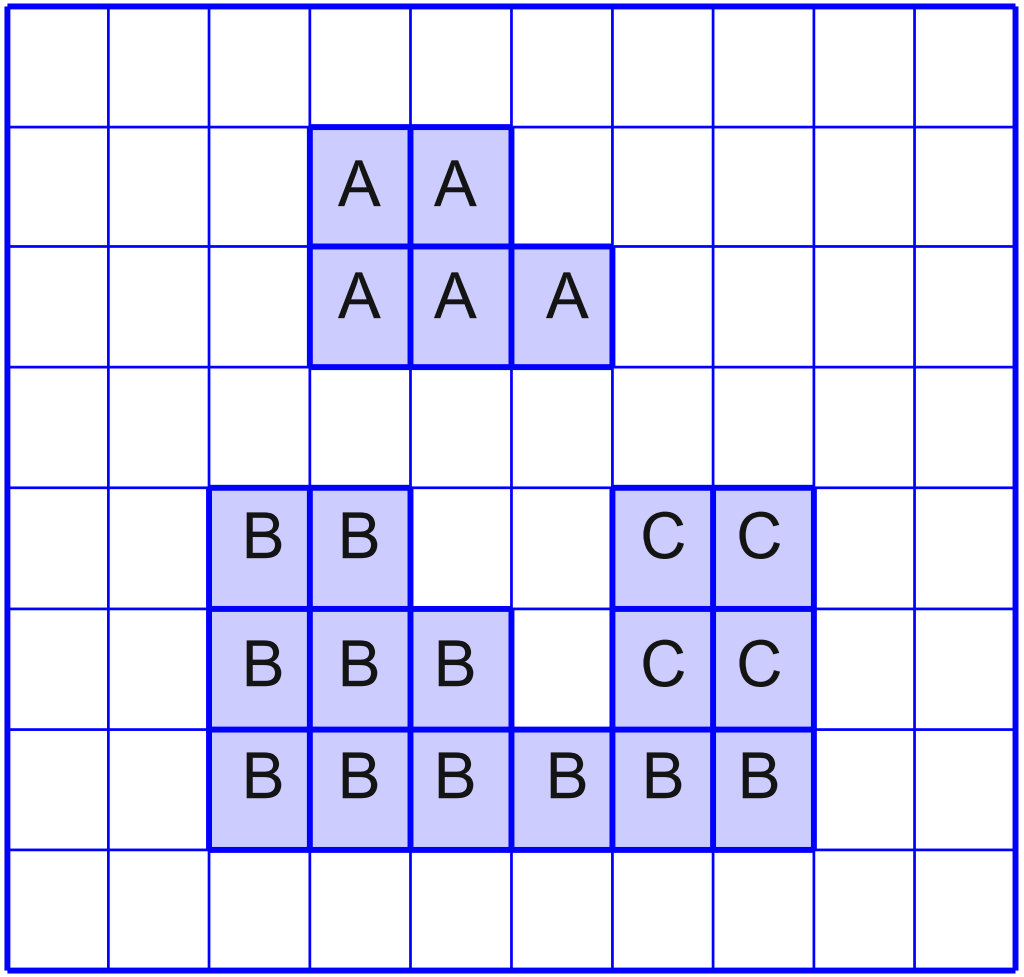
Set threshold here



BINARY IMAGE



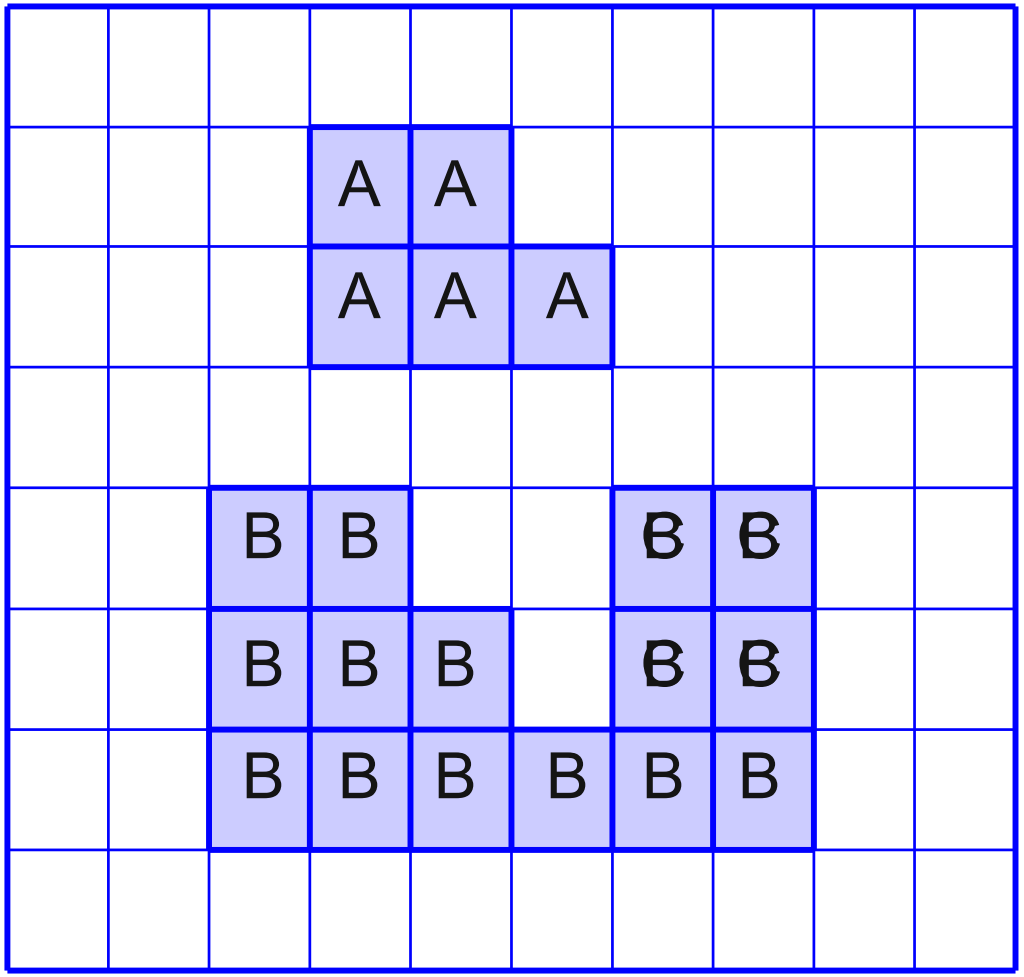
CONNECTED COMPONENT LABELING: FIRST PASS



EQUIVALENCE:

B=C

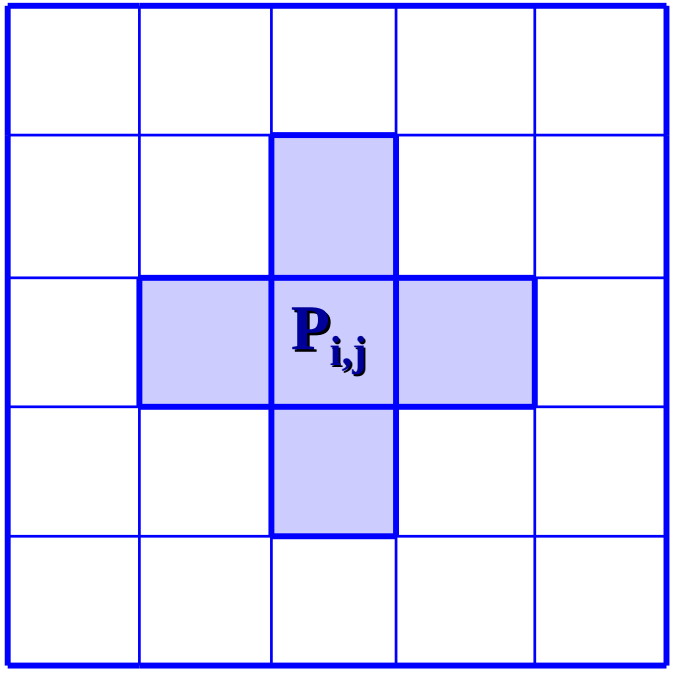
CONNECTED COMPONENT LABELING: SECOND PASS



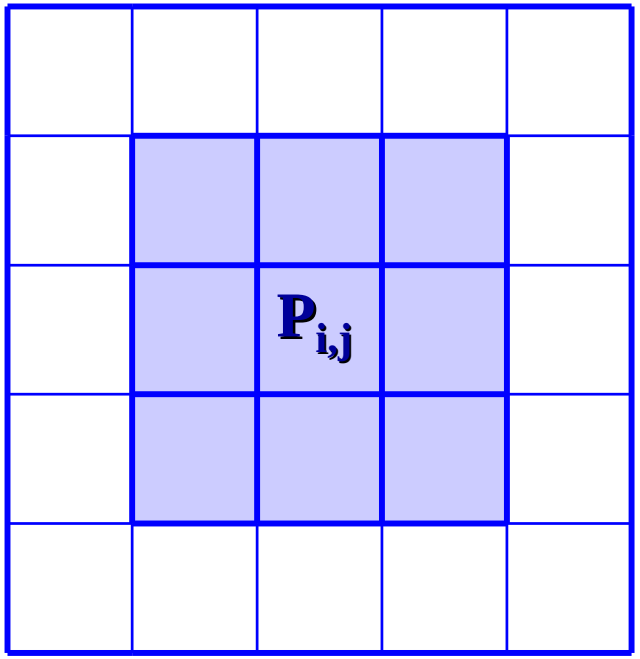
TWO OBJECTS!

IMAGE SEGMENTATION – CONNECTED COMPONENT LABELING

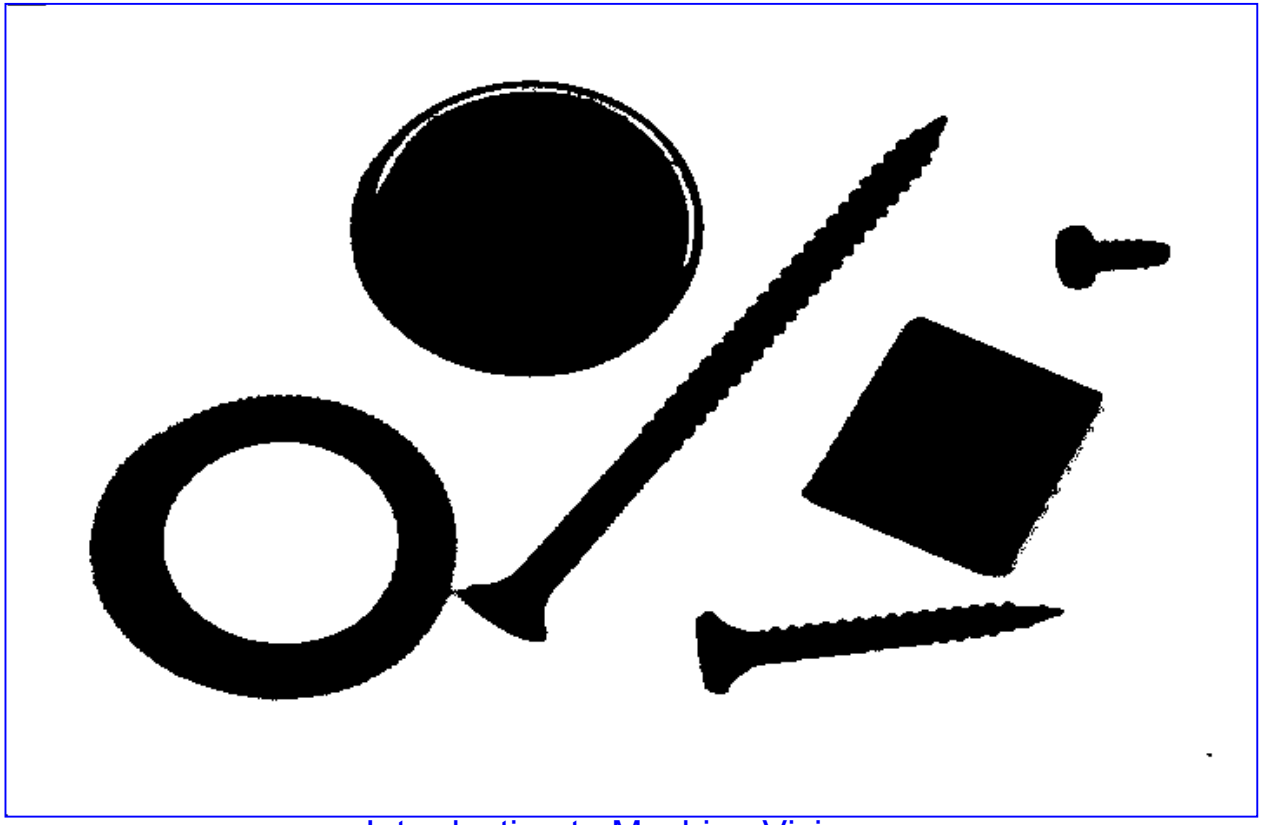
4 Neighbor Connectivity



8 Neighbor Connectivity



What are some examples of form parameters that would be useful in identifying the objects in the image below?



OBJECT RECOGNITION – BLOB ANALYSIS

- Examples of form parameters that are invariant with respect to position, orientation, and scale:
  - Number of holes in the object
  - Compactness or Complexity: $(\text{Perimeter})^2/\text{Area}$
  - Moment invariants
- All of these parameters can be evaluated during contour following.

GENERALIZED MOMENTS

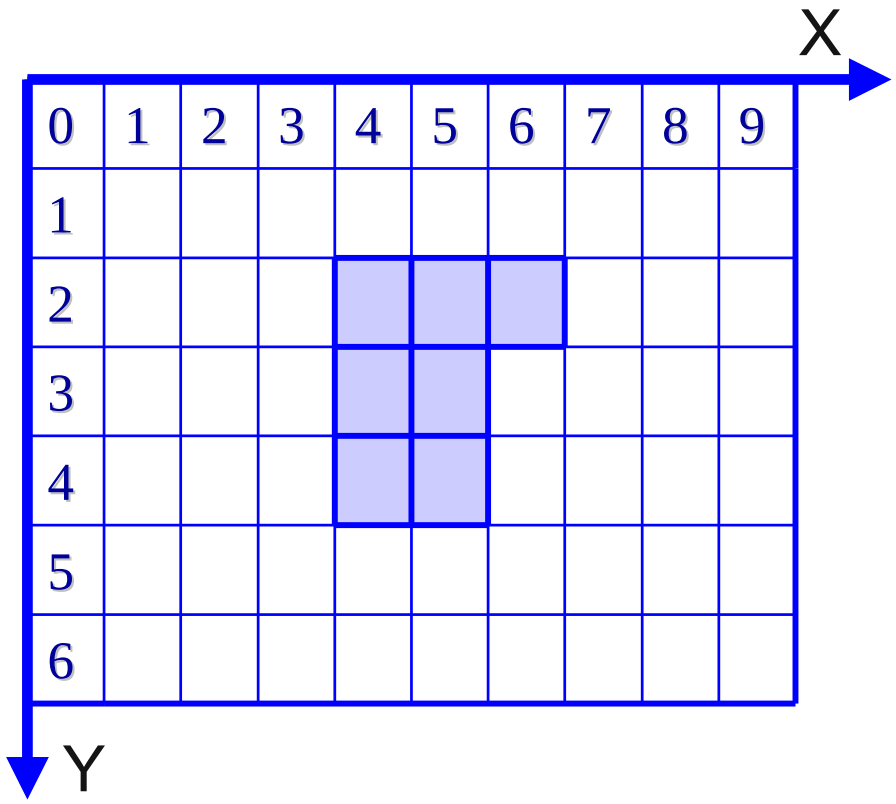
- Shape features or form parameters provide a high level description of objects or regions in an image
- For a digital image of size **n** by **m** pixels :

$$M_{ij} = \sum_{x=1}^{n} \sum_{y=1}^{m} x^i y^j f(x, y)$$

- For binary images the function f(x,y) takes a value of 1 for pixels belonging to class “object” and “0” for class “background”.

GENERALIZED MOMENTS

$$M_{ij} = \sum_{x=1}^{n} \sum_{y=1}^{m} x^i y^j f(x,y)$$



i	j	M_{ij}	
0	0	7	← Area
1	0	33	
0	1	20	
2	0	159	← Moment of Inertia
0	2	64	
1	1	93	

SOME USEFUL MOMENTS

•The center of mass of a region can be defined in terms of generalized moments as follows:

$$\overline{X} = \frac{M_{10}}{M_{00}} \qquad \overline{Y} = \frac{M_{01}}{M_{00}}$$

SOME USEFUL MOMENTS

•The moments of inertia relative to the center of mass can be determined by applying the general form of the parallel axis theorem:

$$\overline{M}_{02} = M_{02} - \frac{M_{01}^2}{M_{00}}$$

$$\overline{M}_{20} = M_{20} - \frac{M_{10}^2}{M_{00}}$$

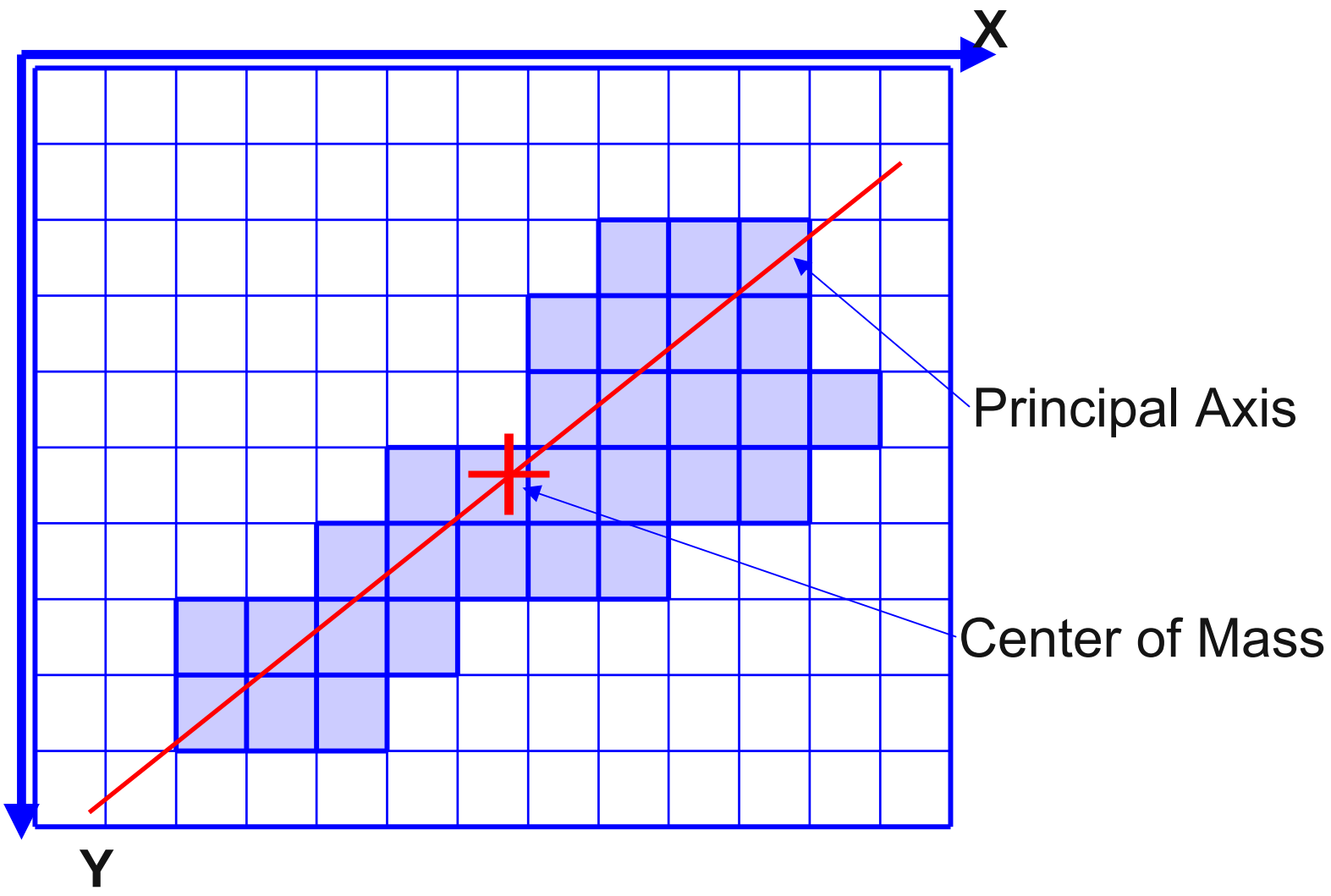
$$\overline{M}_{11} = M_{11} - \frac{M_{10} M_{01}}{M_{00}}$$

SOME USEFUL MOMENTS

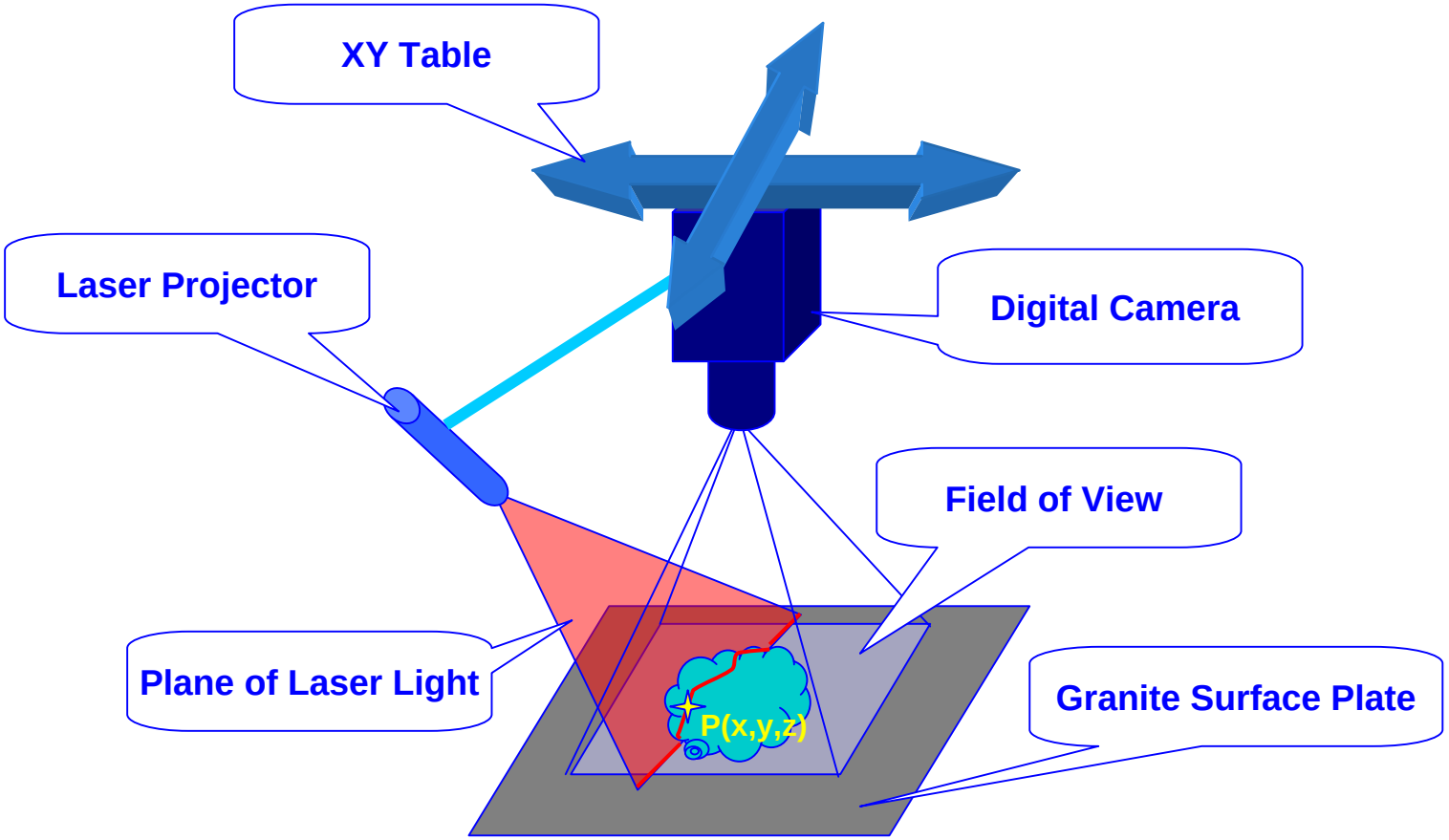
- The principal axis of an object is the axis passing through the center of mass which yields the minimum moment of inertia.
- This axis forms an angle θ with respect to the X axis.
- The principal axis is useful in robotics for determining the orientation of randomly placed objects.

$$TAN\,2\theta = \frac{2\overline{M}_{11}}{\overline{M}_{20} - \overline{M}_{02}}$$

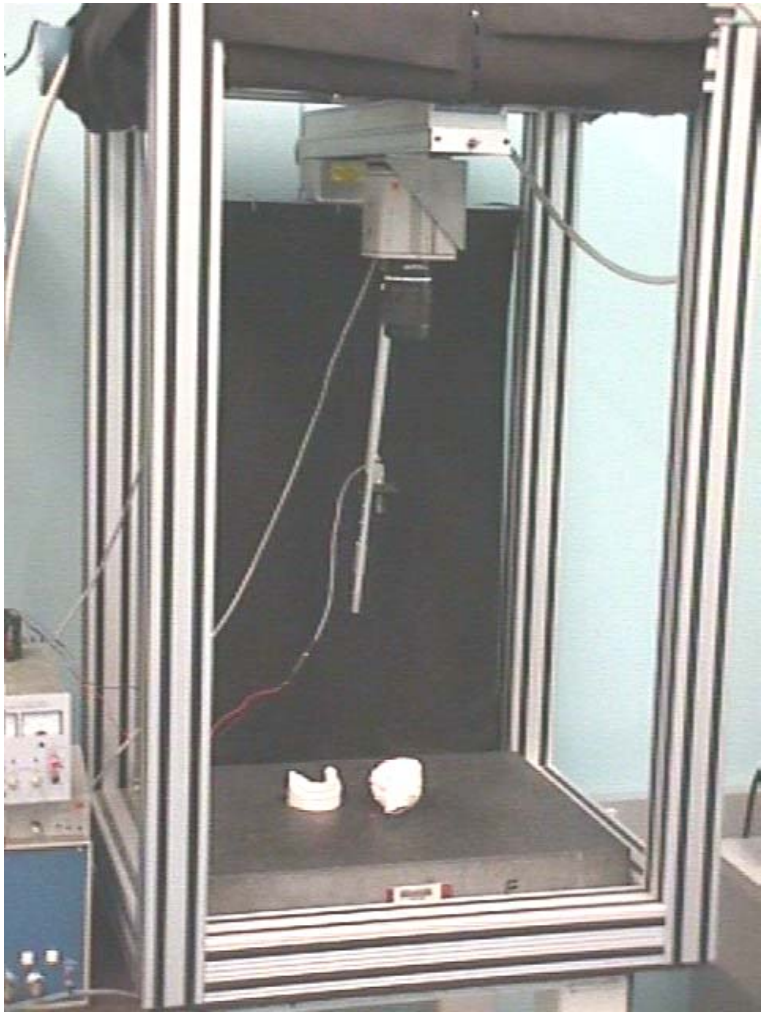
Example

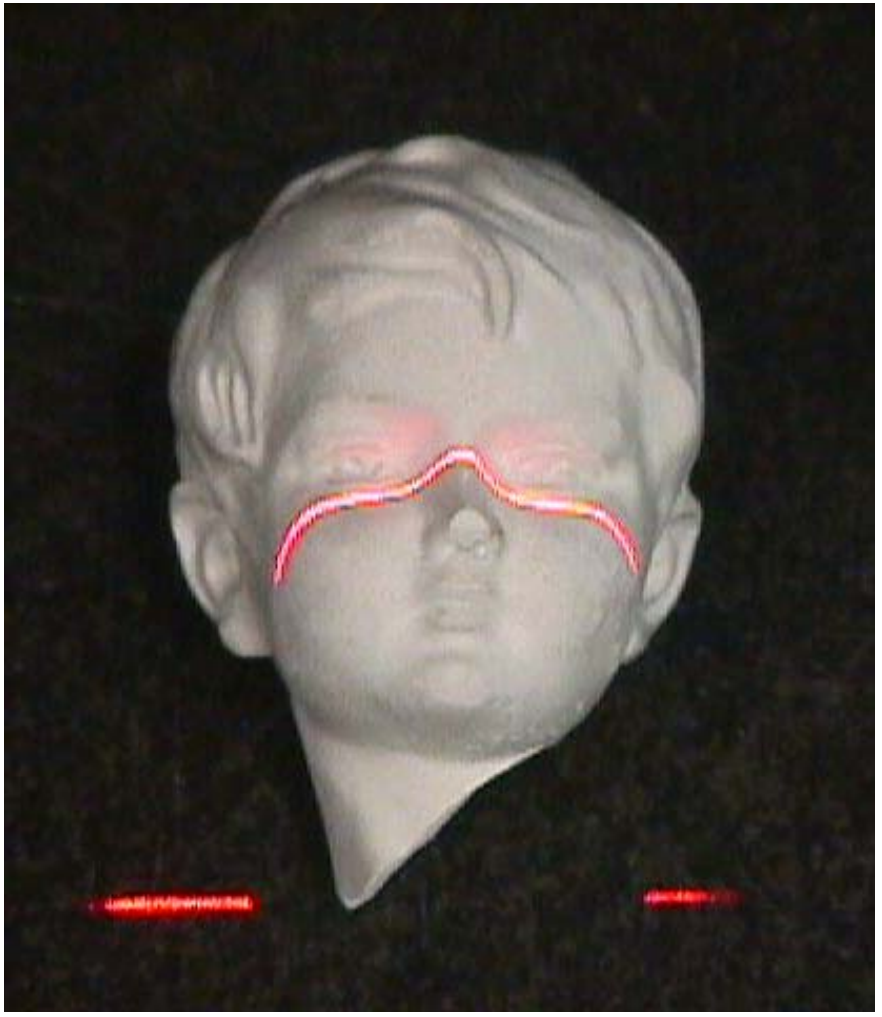


3D MACHINE VISION SYSTEM

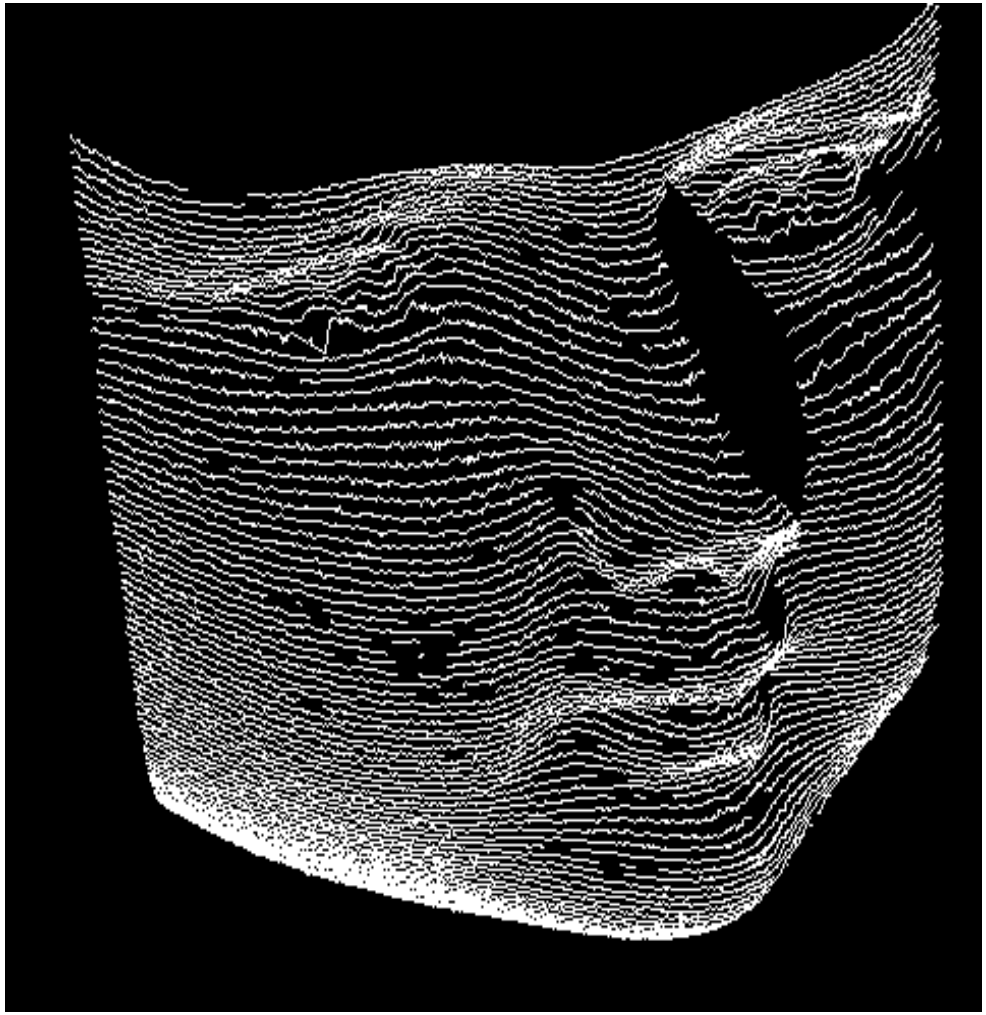


3D MACHINE VISION SYSTEM

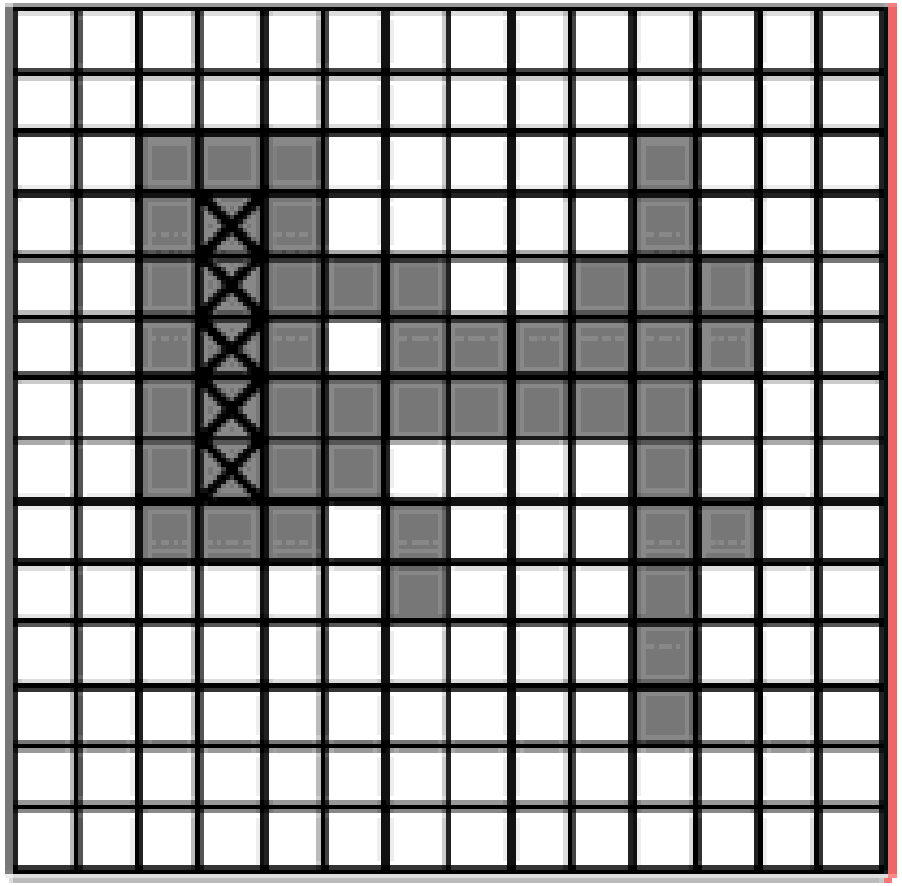




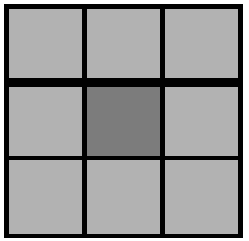
3D MACHINE VISION SYSTEM



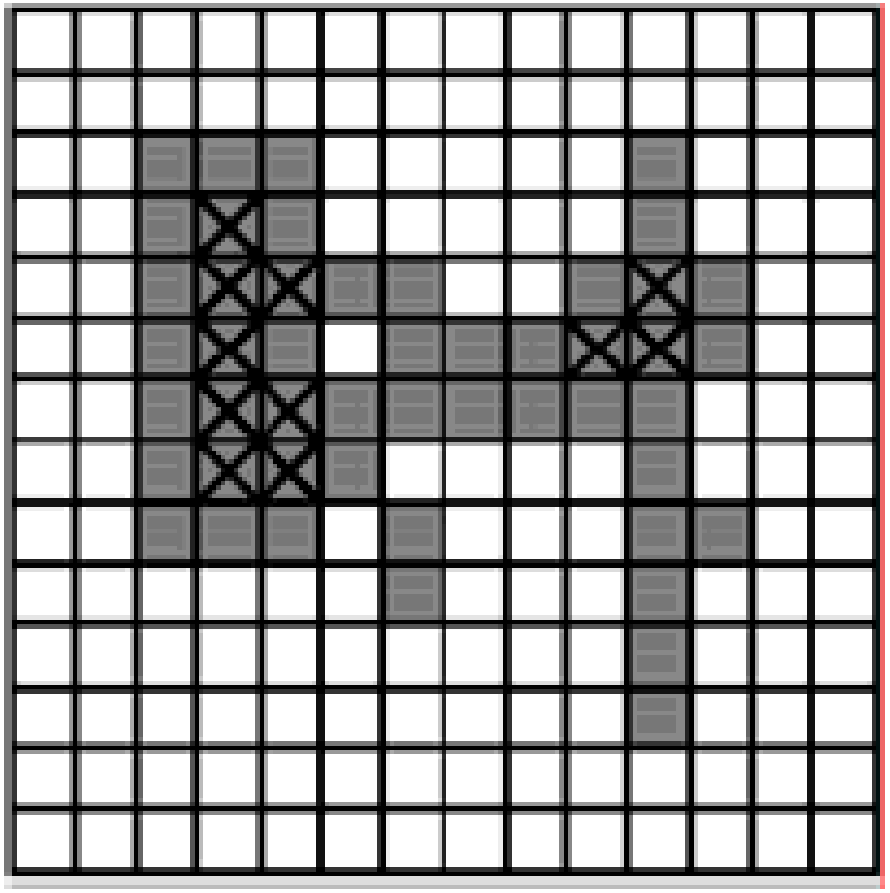
Morphological processing
Erosion



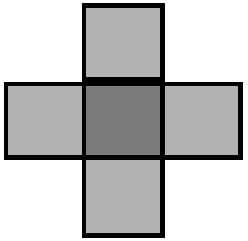
SE=



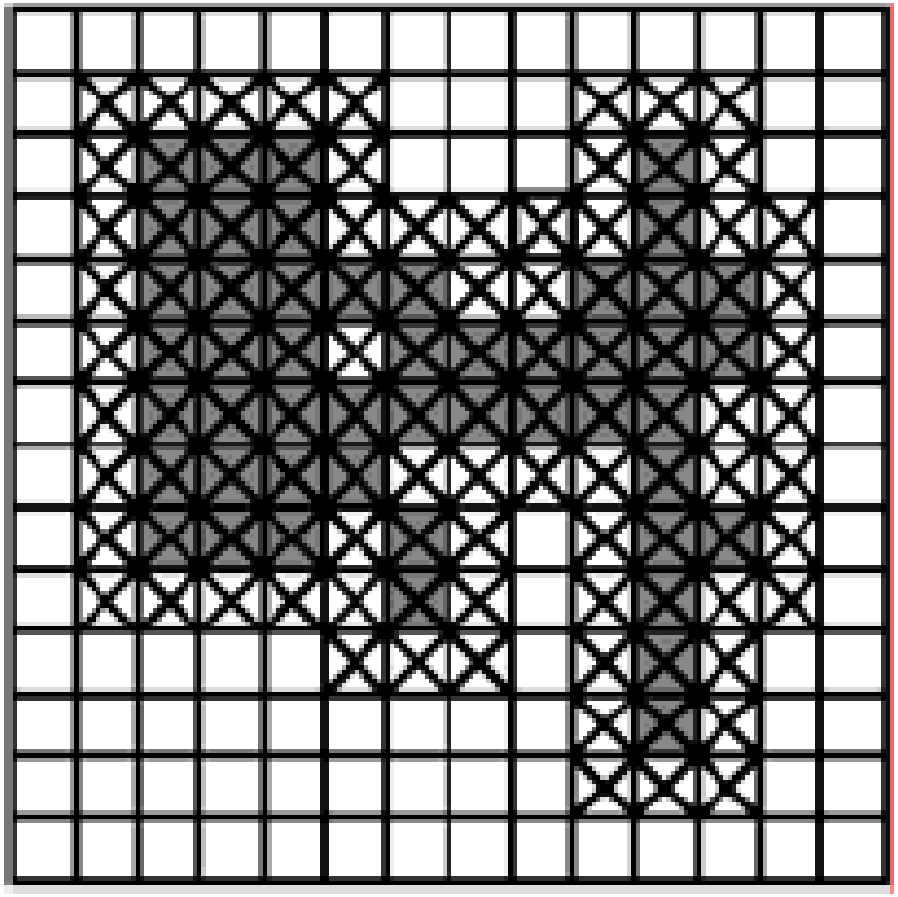
Morphological processing
Erosion



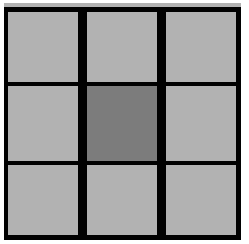
SE=



Dilation



SE=



Opening

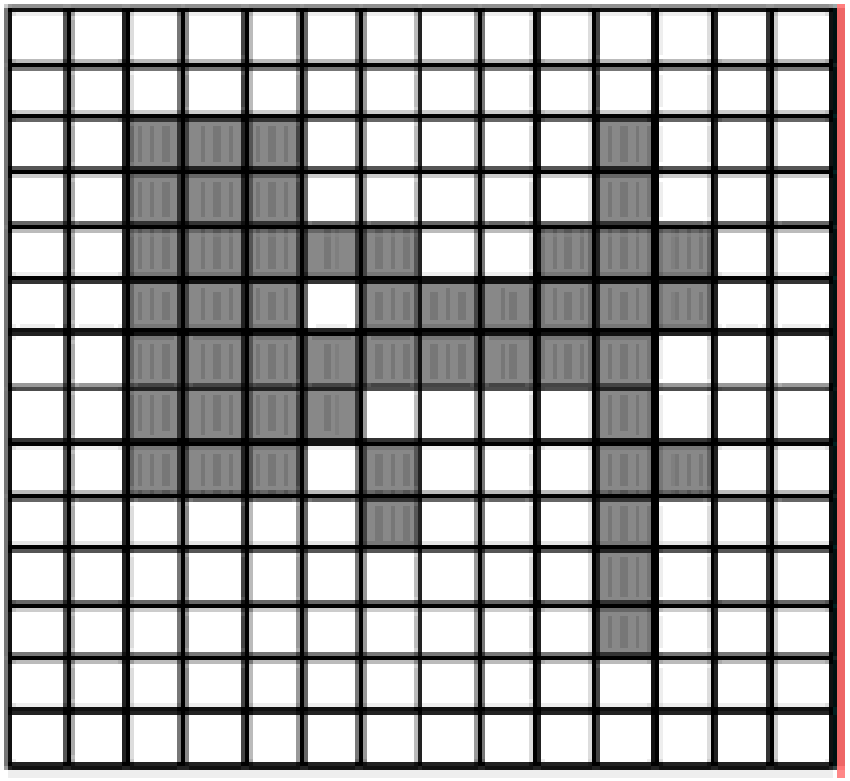
erosion followed by dilation, denoted

$$A \circ B = (A \ominus B) \oplus B$$

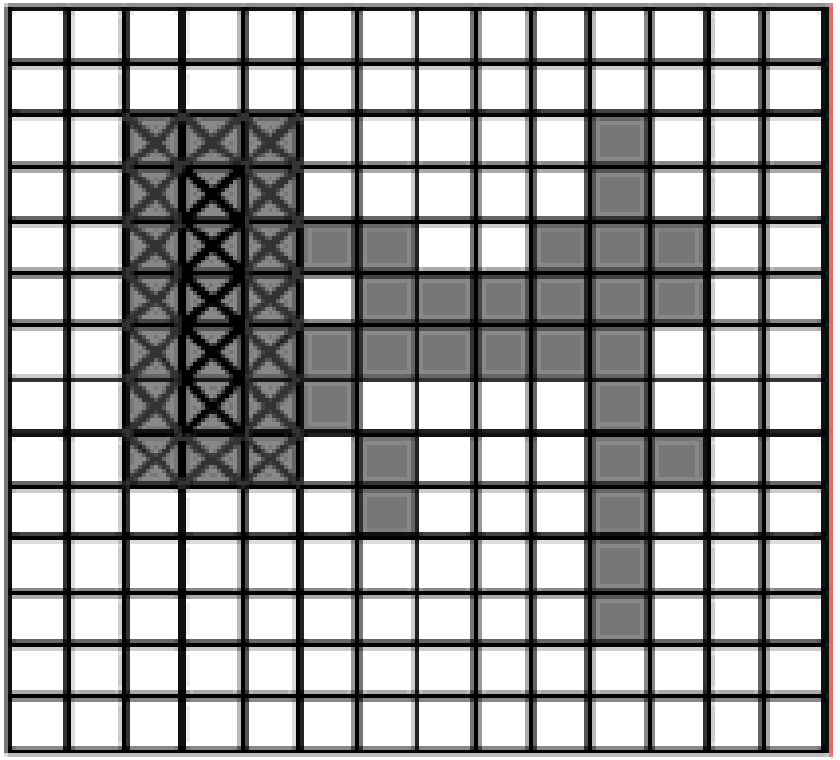
- eliminates protrusions
- breaks necks
- smoothes contour

Opening

B=



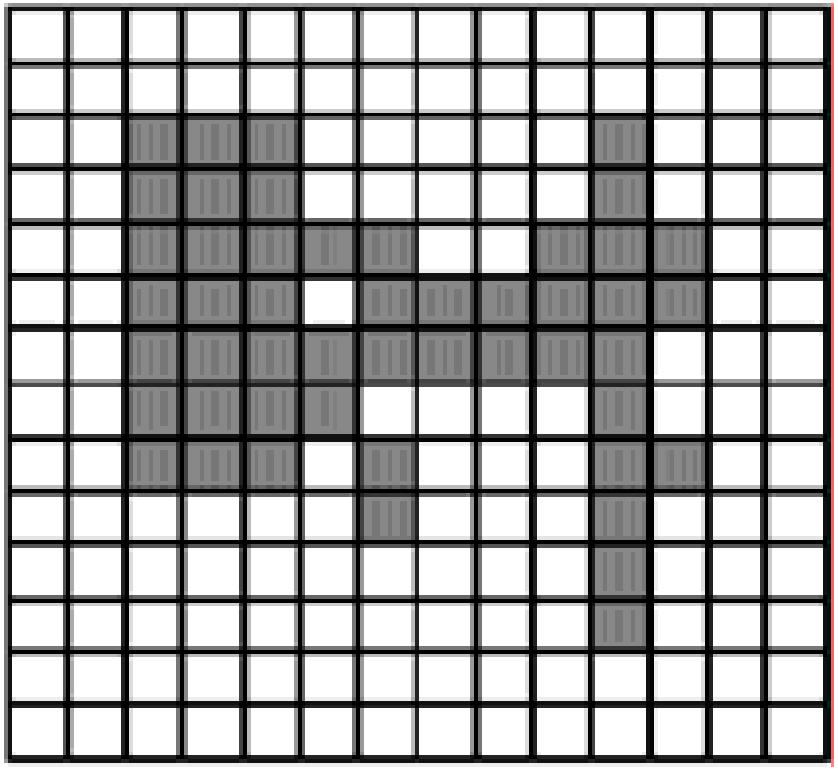
A



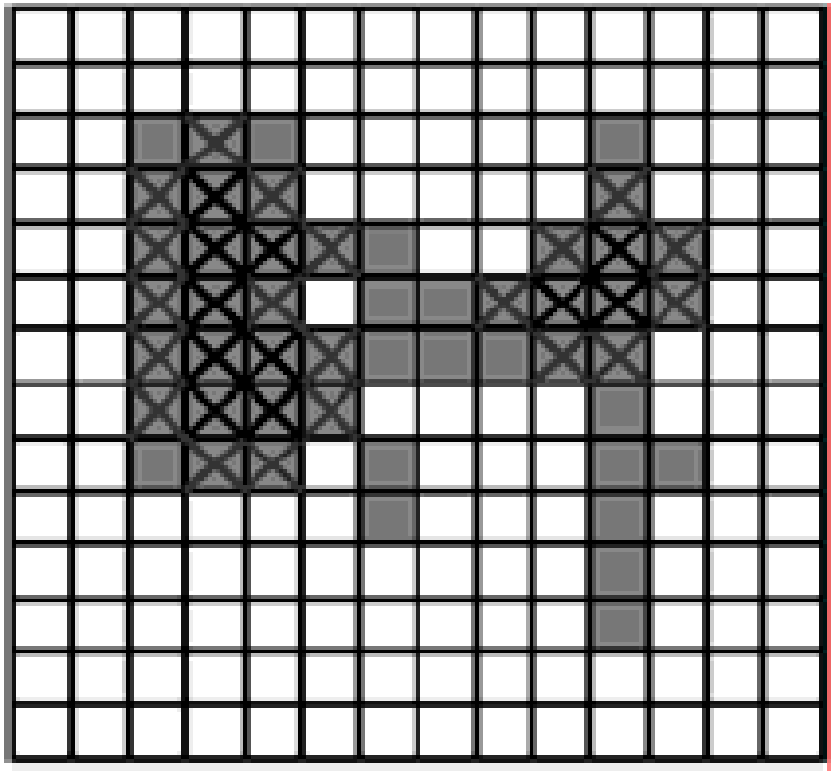
$A \ominus B$ $A \circ B$

Opening

B=

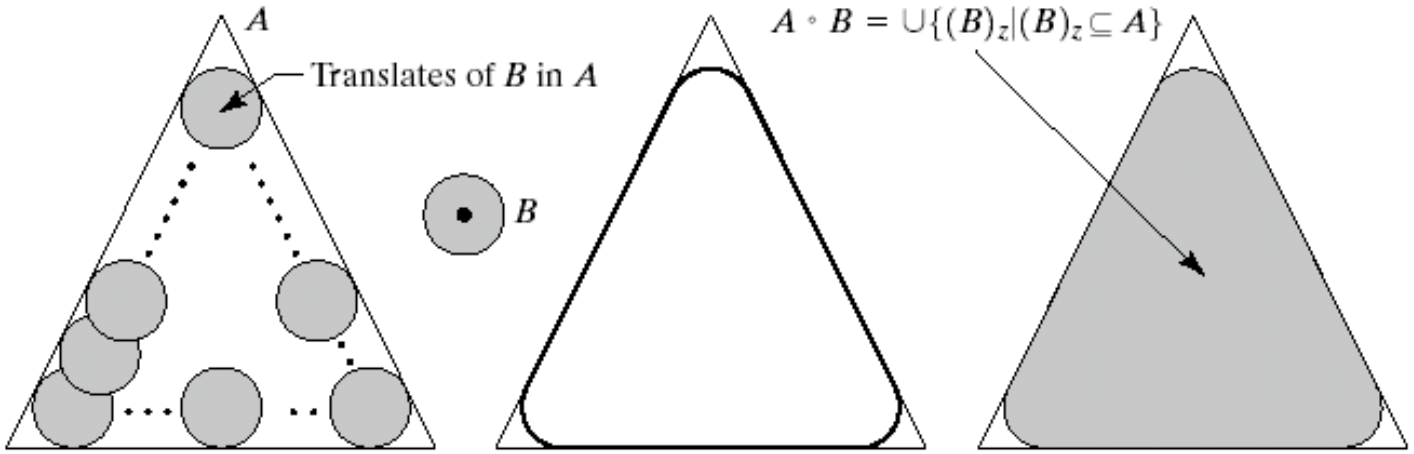


A



$A \ominus B$ $A \circ B$

Opening



a b c d

FIGURE 9.8 (a) Structuring element B "rolling" along the inner boundary of A (the dot indicates the origin of B). (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded).

$$A \circ B = (A \ominus B) \oplus B$$

$$A \circ B = \cup\{(B)_z \mid (B)_z \subseteq A\}$$

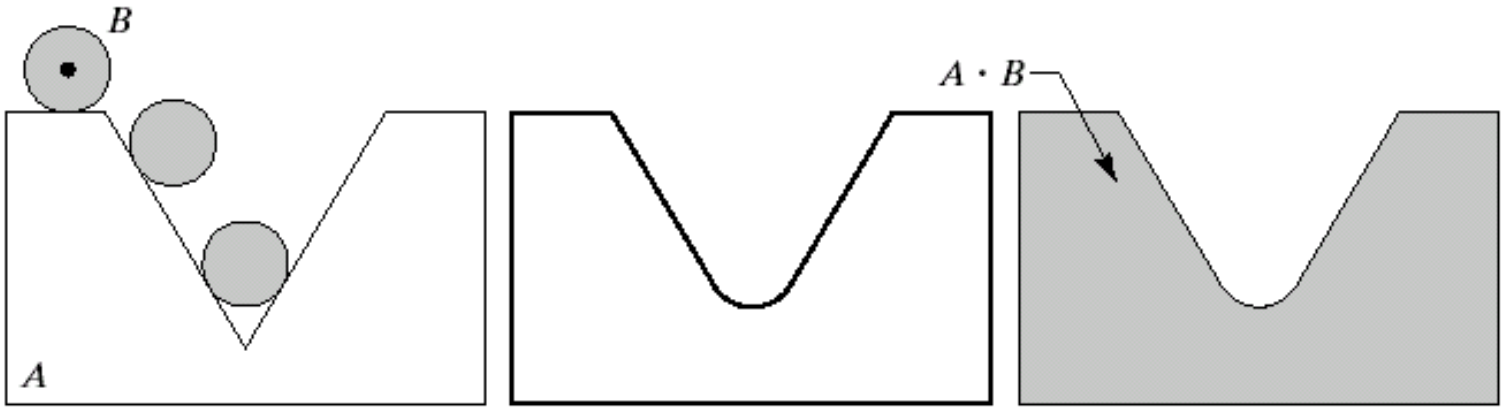
Closing

dilation followed by erosion, denoted •

$$A \bullet B = (A \oplus B) \ominus B$$

- smooth contour
- fuse narrow breaks and long thin gulfs
- eliminate small holes
- fill gaps in the contour

Closing

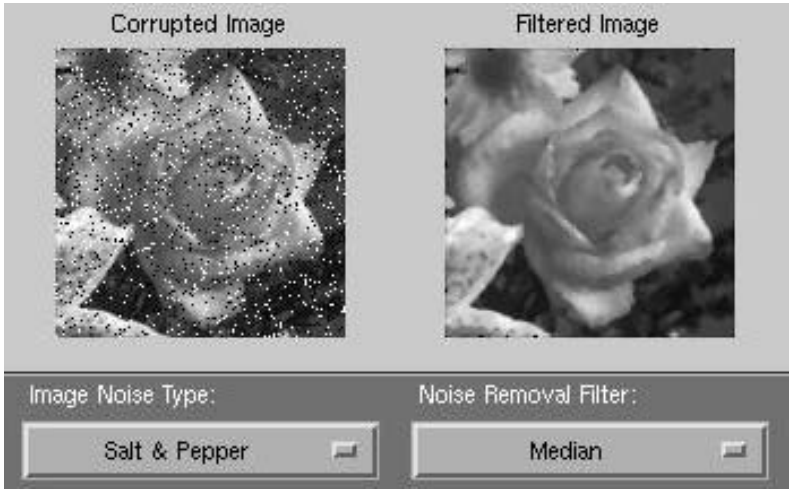


a b c

FIGURE 9.9 (a) Structuring element B "rolling" on the outer boundary of set A. (b) Heavy line is the outer boundary of the closing. (c) Complete closing (shaded).

$$A \bullet B = (A \oplus B) \ominus B$$

Image Processing

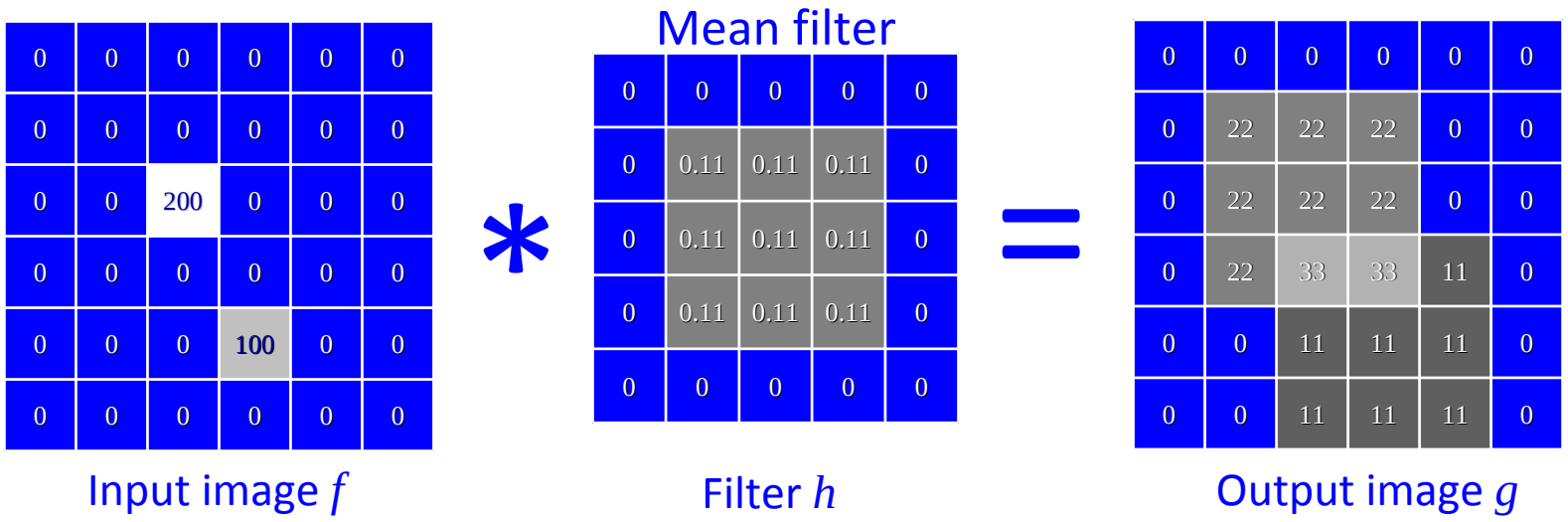


Filtering: Noise suppresion

Edge detection

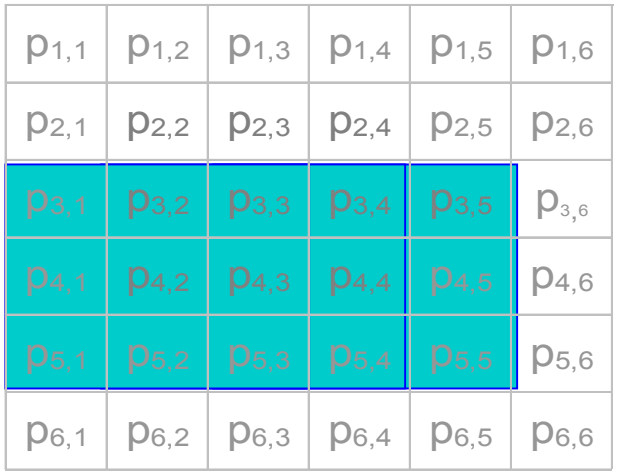
Image filtering

$$g(x,y) = \sum_{x'} \sum_{y'} f(x+x', y+y') h(x', y')$$



Animation of Convolution

To accomplish convolution of the whole image, we just ***Slide the mask***



Original Image

m1,1	m1,2	m1,3
m2,1	m2,2	m2,3
m3,1	m3,2	m3,3

Mask

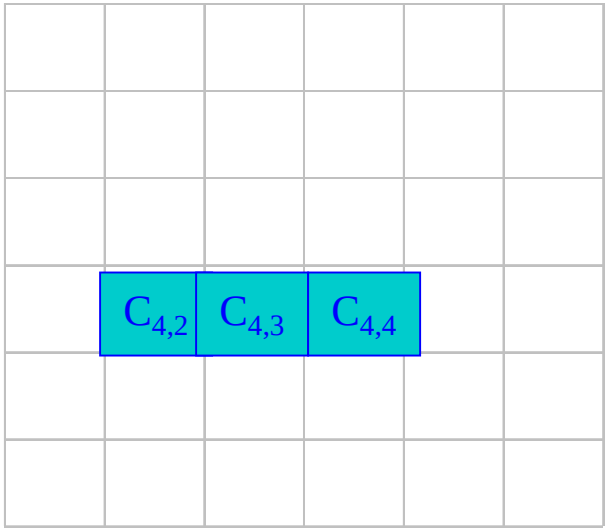
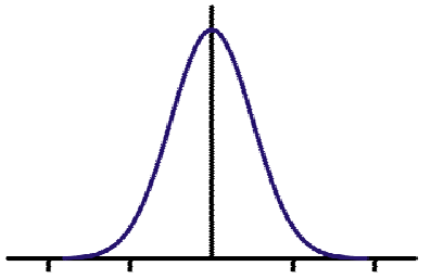


Image after convolution

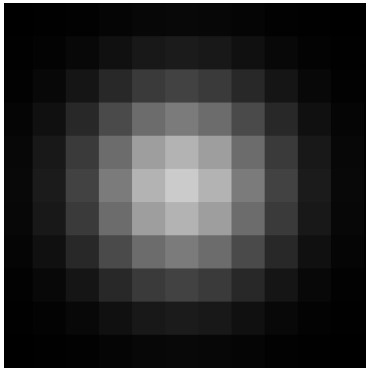
Gaussian filter



$$\mathcal{G}_\sigma(x,y) = \frac{1}{Z} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Compute empirically

Input image *f*



Filter *h*

Output image *g*

Convolution for Noise Elimination

--Noise Elimination

The noise is eliminated but the operation causes loss of sharp edge definition.

In other words, the image becomes *blurred*

Convolution with a non-linear mask
Median filtering

There are many masks used in *Noise Elimination*

Median Mask is a typical one

The principle of Median Mask is to mask some sub-image, use the median of the values of the sub-image as its value in new image

	J=1	2	3
I=1	23	65	64
2	120	187	90
3	47	209	72

Masked Original Image

Rank: 23, 47, 64, 65, **72**, 90, 120, 187, 209

median

Median Filtering

Original Image

Noisy Image

```
Denoised=
  medfilt2(NoiseImage);
```

After Median filtering

Edge detection using the Sobel operator

In practice, it is common to use:

$$g_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad g_y = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Magnitude:

$$g = \sqrt{g_x^2 + g_y^2}$$

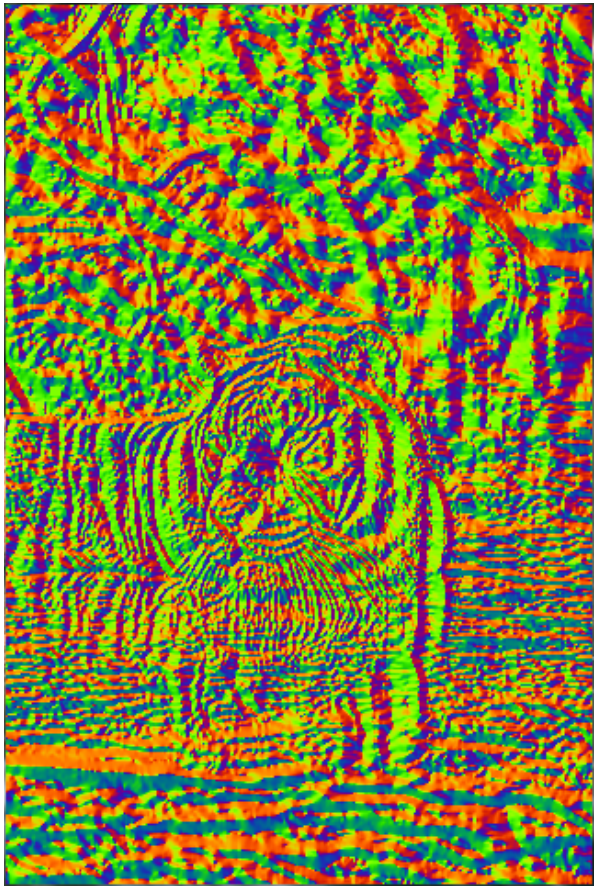
Orientation:

$$\Theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Sobel operator

Original

Magnitude

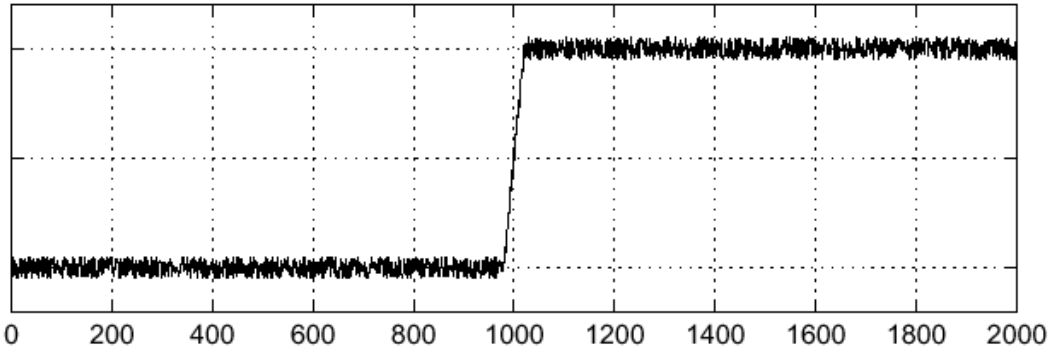


Orientation

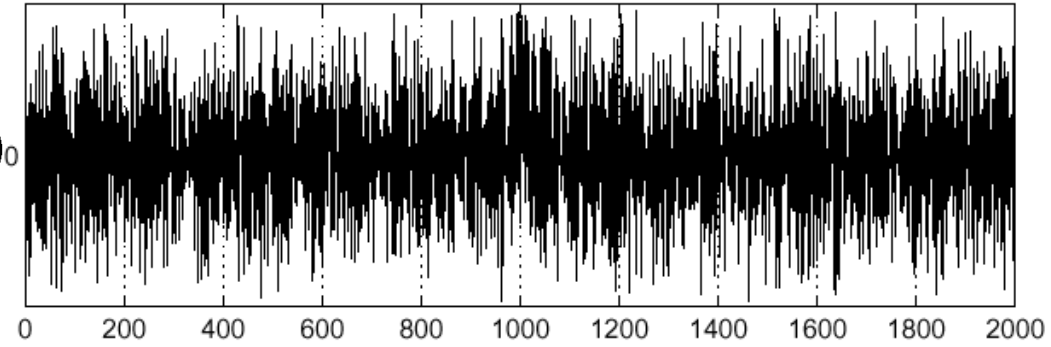
Effects of noise on edge detection / derivatives

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

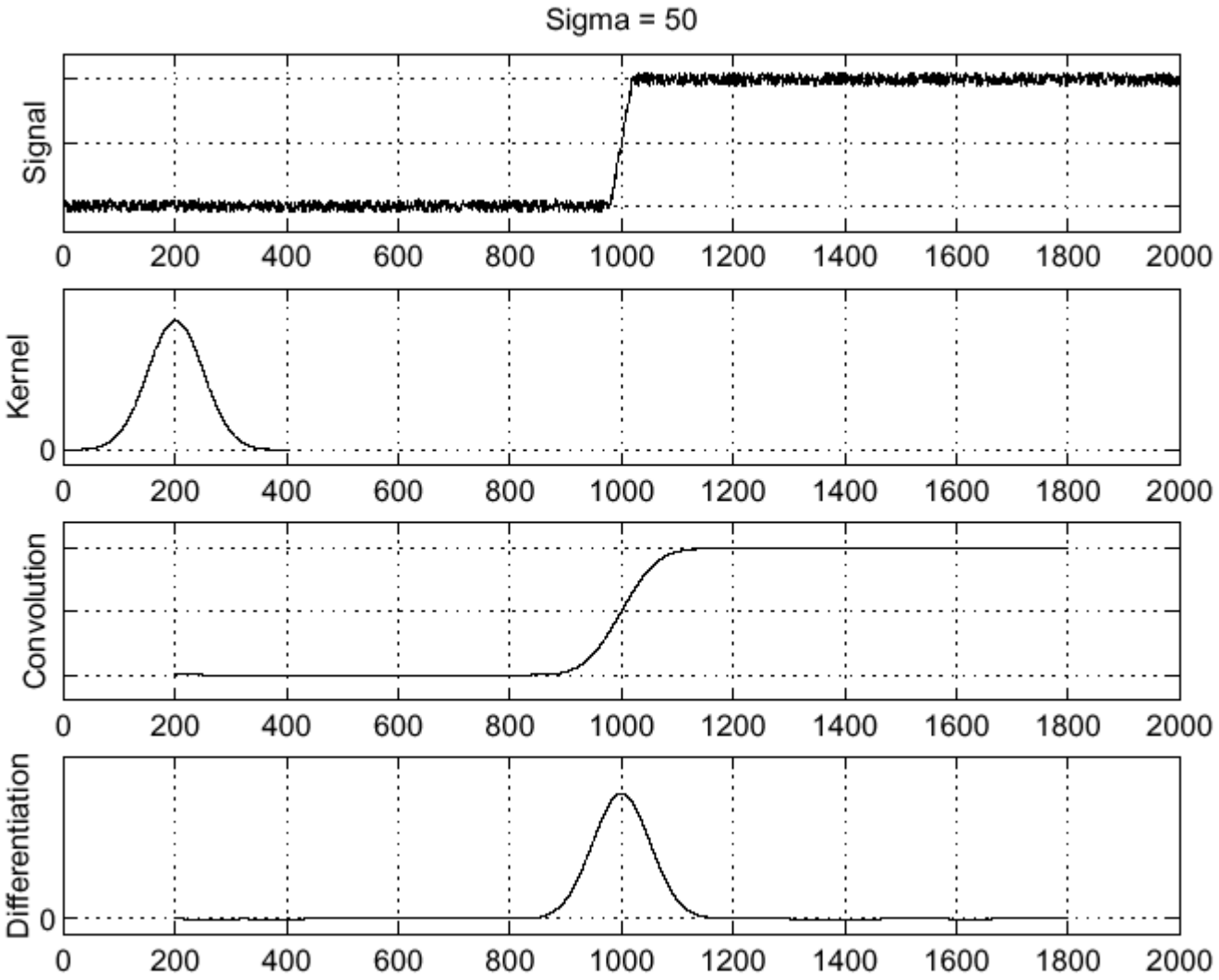
Solution: smooth first

f

h

$h \star f$

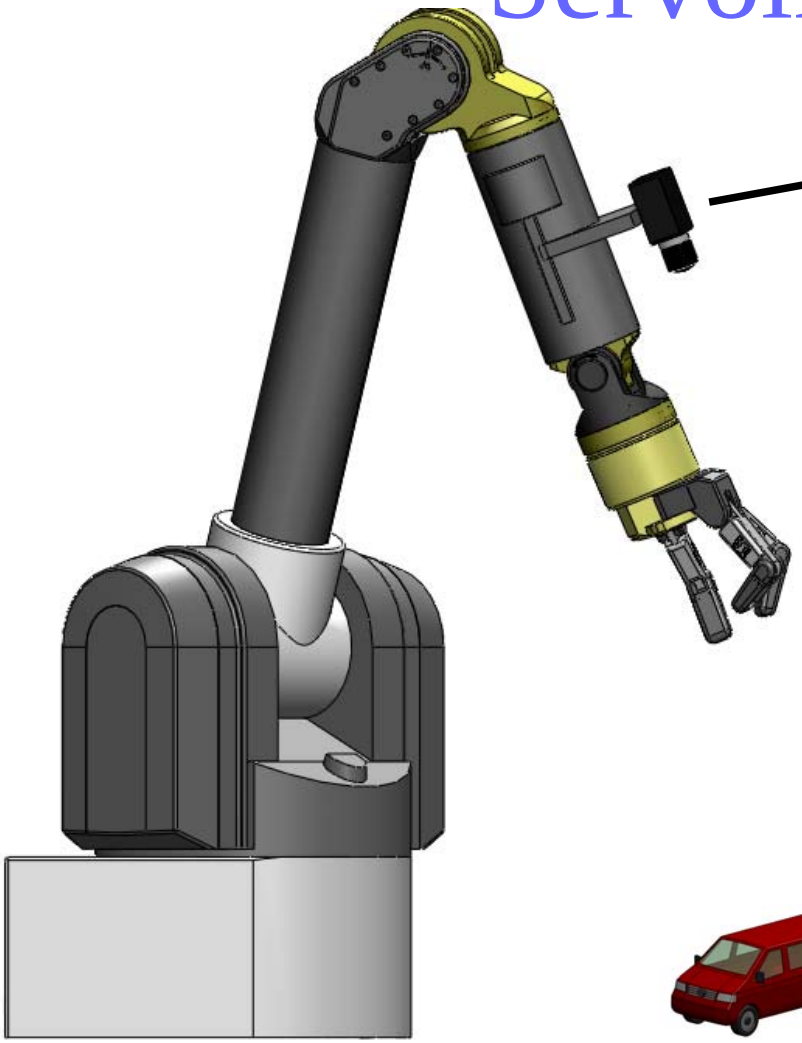
$\frac{\partial}{\partial x}(h \star f)$



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Vision-Based Control (Visual Servoing)

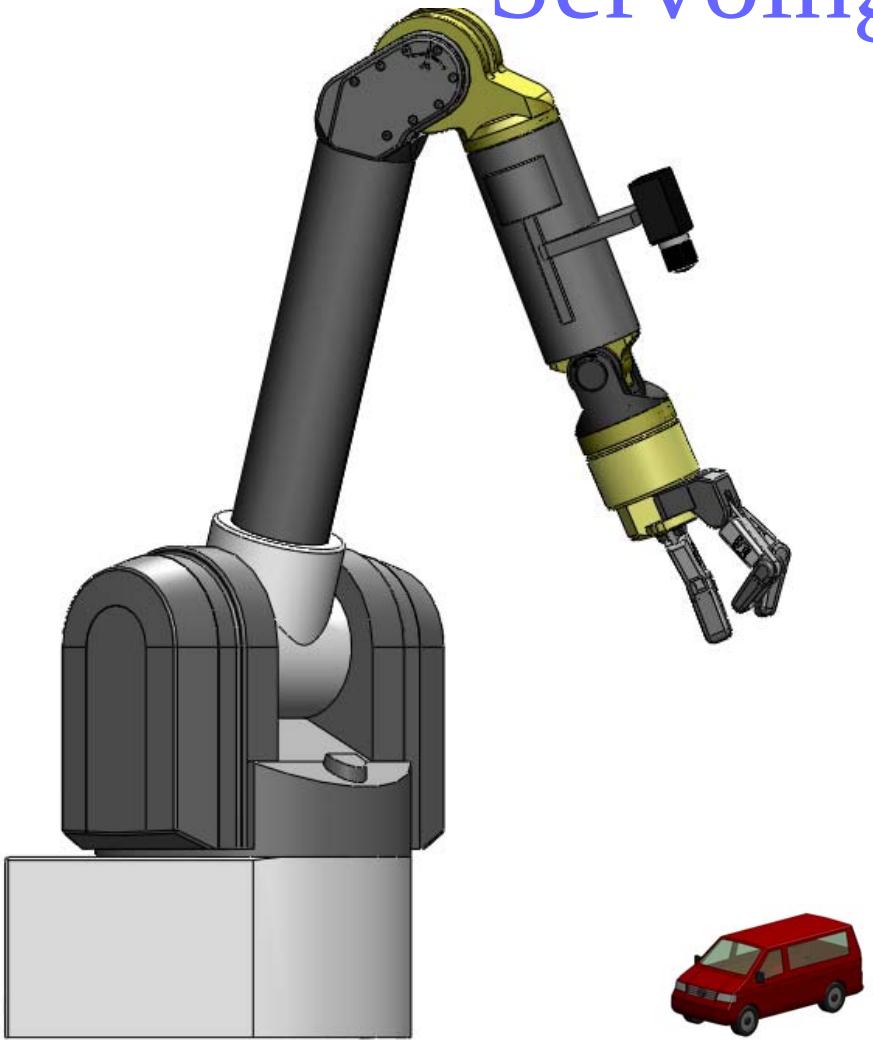


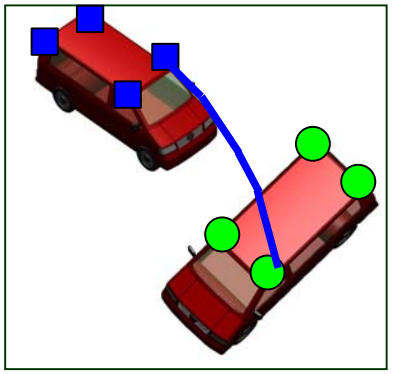
Initial Image

Desired Image

User

Vision-Based Control (Visual Servoing)

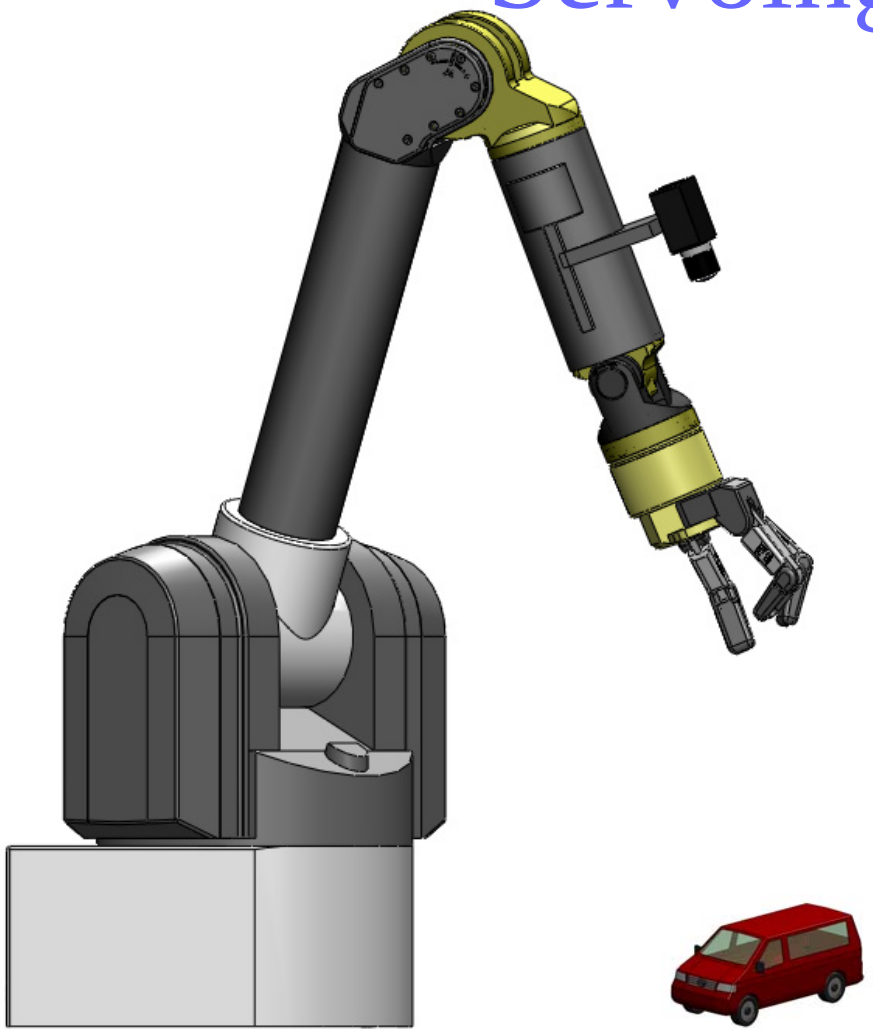


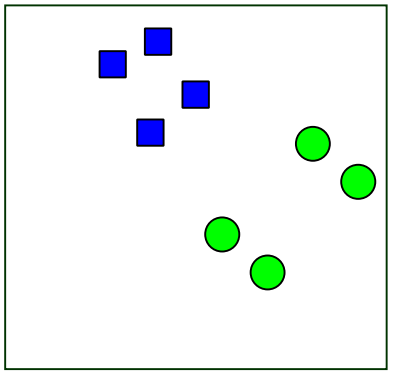


■ : Current Image Features

● : Desired Image Features

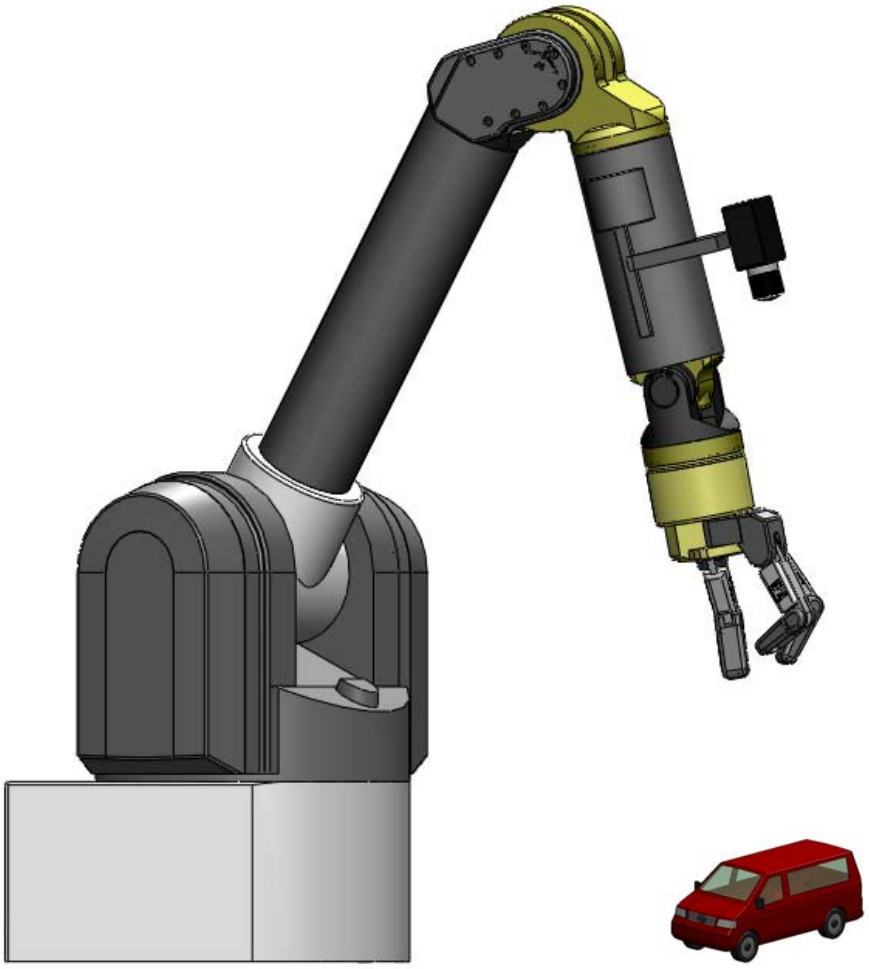
Vision-Based Control (Visual Servoing)

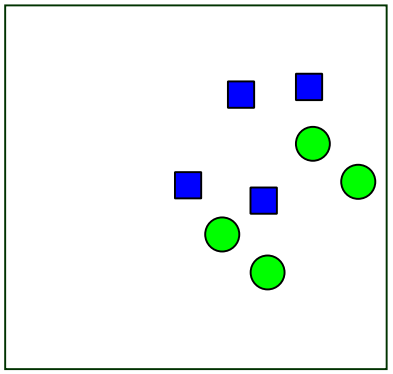




■ : Current Image Features
● : Desired Image Features

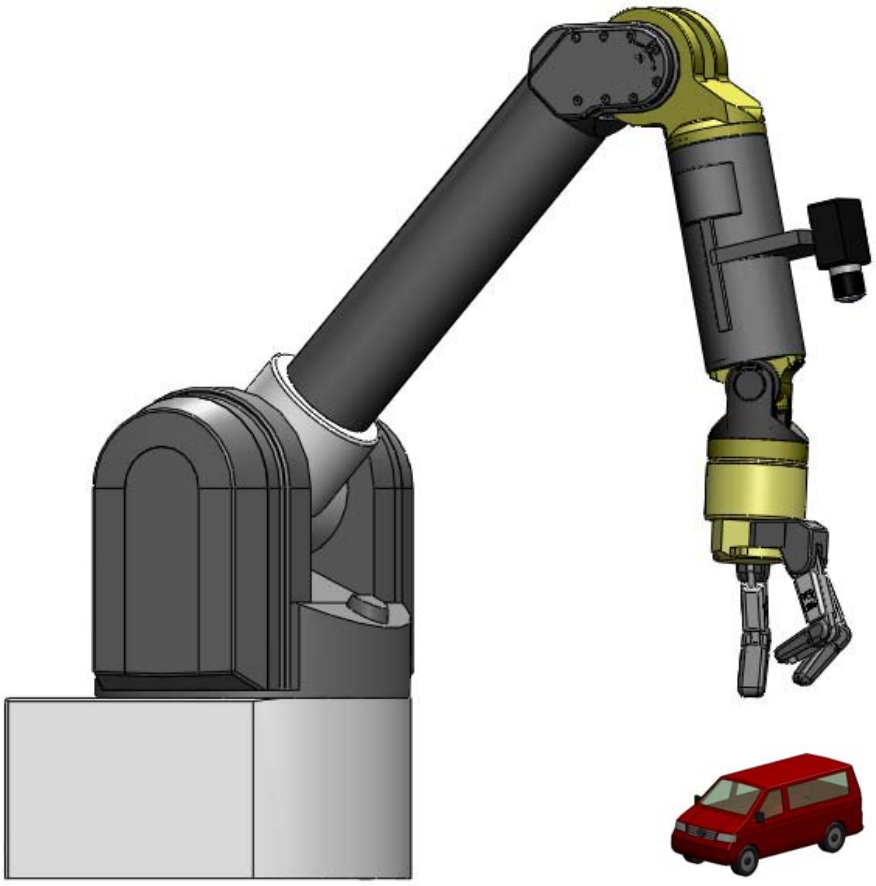
Vision-Based Control (Visual Servoing)

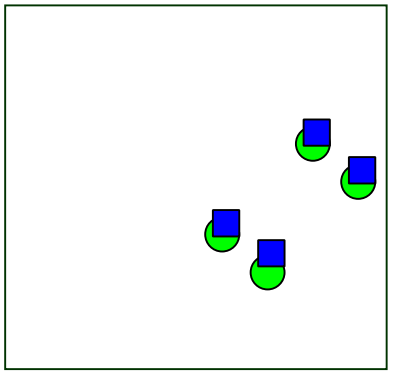




■ : Current Image Features
● : Desired Image Features

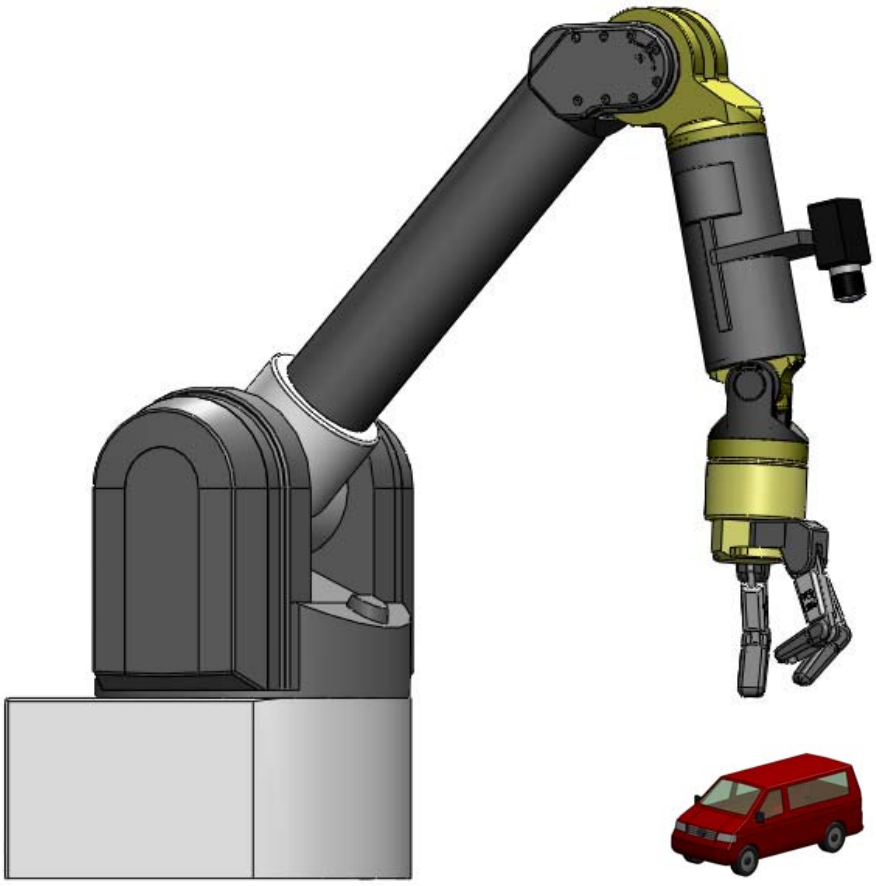
Vision-Based Control (Visual Servoing)

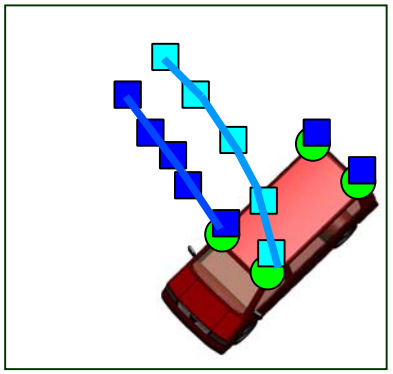




■ : Current Image Features
● : Desired Image Features

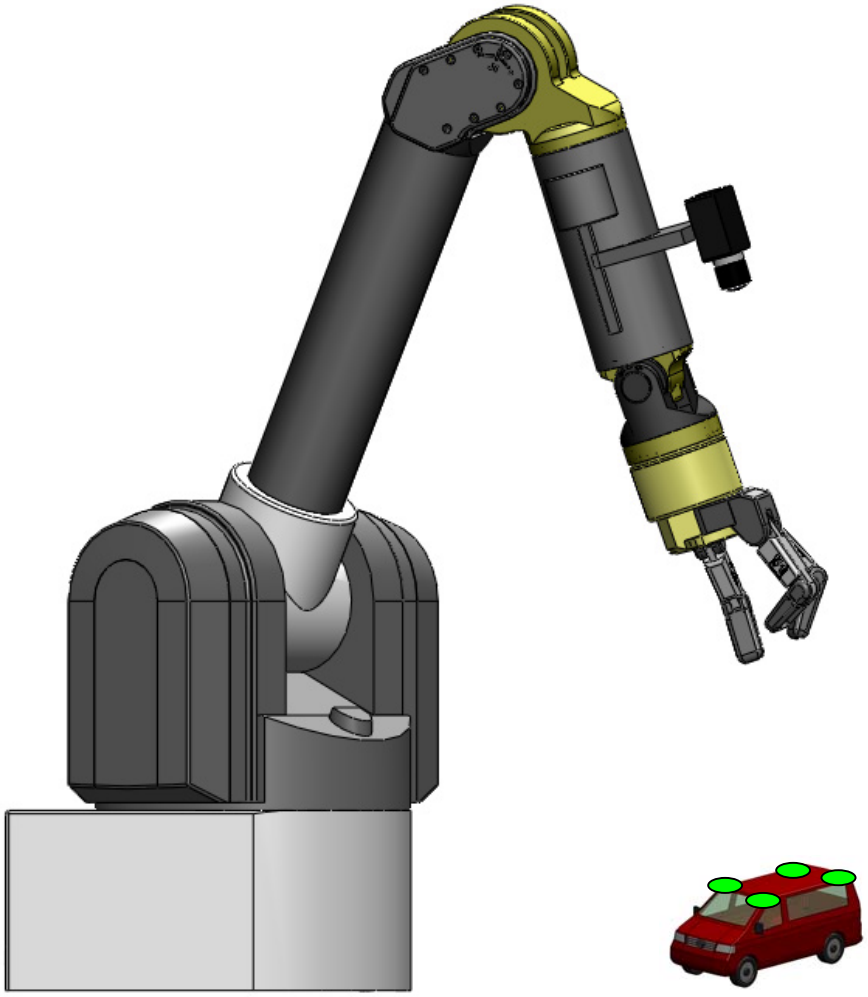
Vision-Based Control (Visual Servoing)

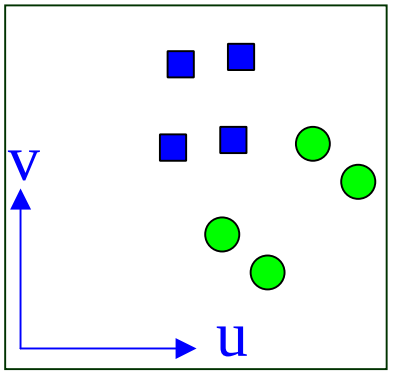




■ : Current Image Features
● : Desired Image Features

u,v Image-Space Error





■ : Current Image Features

● : Desired Image Features

$$\mathbf{E} = [\bullet - \blacksquare]$$

One point

$$\mathbf{E} = [\mathbf{y}_0 - \mathbf{y}^*]$$

Pixel coord

$$\mathbf{E} = \begin{bmatrix} y_u \\ y_v \end{bmatrix} - \begin{bmatrix} y_u \\ y_v \end{bmatrix}^*$$

Many points

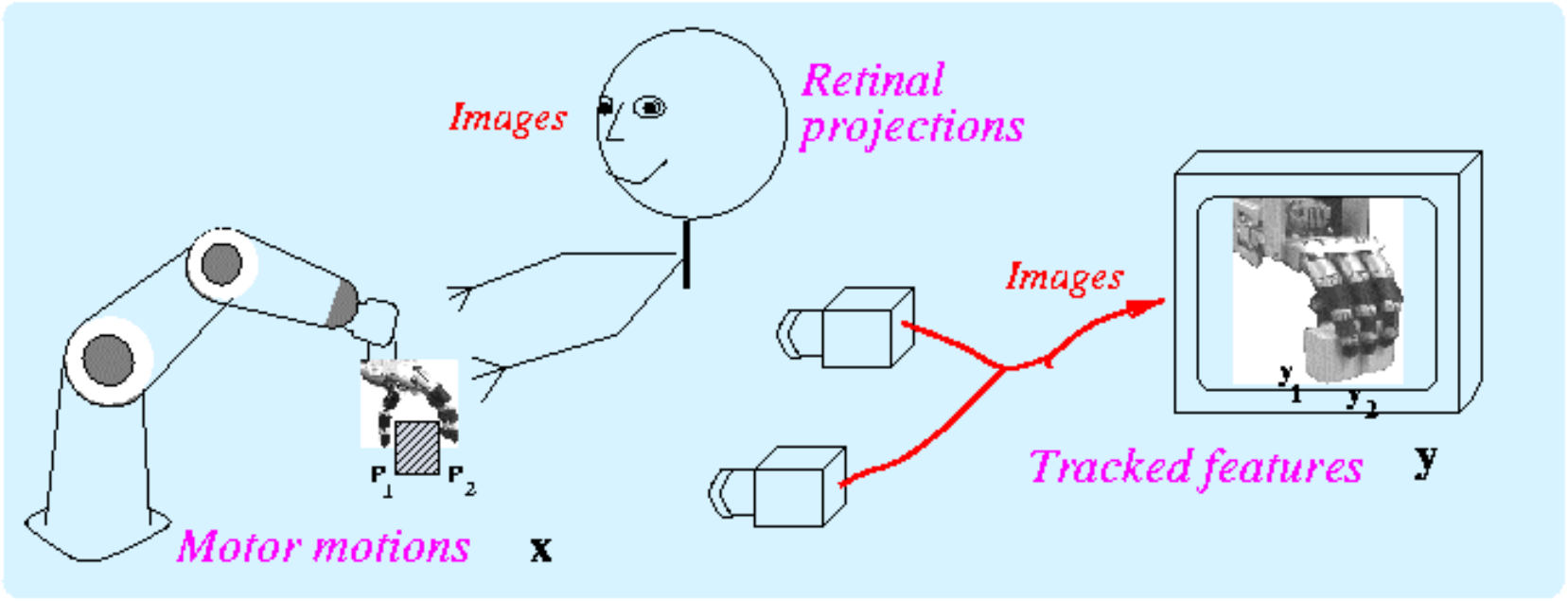
$$\mathbf{E} = \begin{bmatrix} y_1 \\ \vdots \\ y_{16} \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_{16} \end{bmatrix}^*_0$$

Other (easier) solution: Image-based motion control

Note: What follows will work for one or two (or 3..n) cameras.

Either fixed (eye-in-hand) or on the robot.

Here we will use two cam



Motor-Visual function: y=f(x)

Achieving 3d tasks via 2d image control

Vision-Based Control

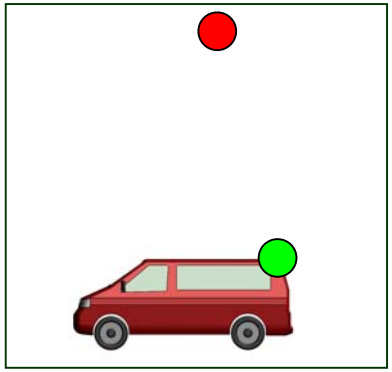
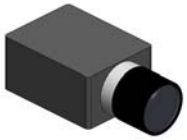
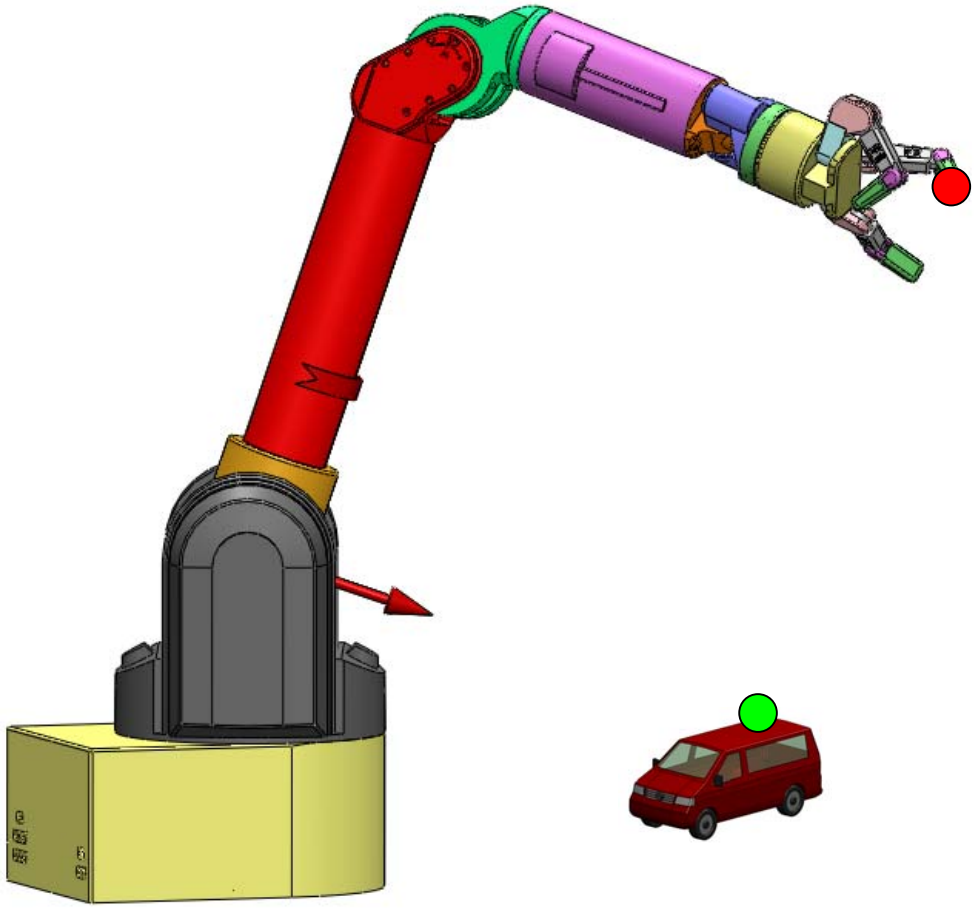


Image 1





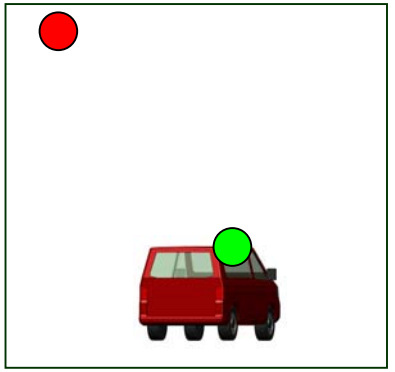
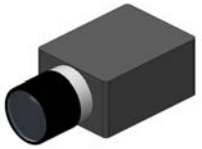


Image 2



Vision-Based Control

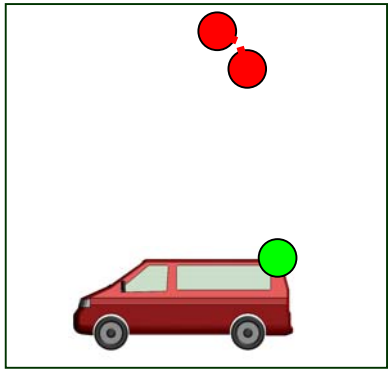
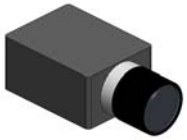
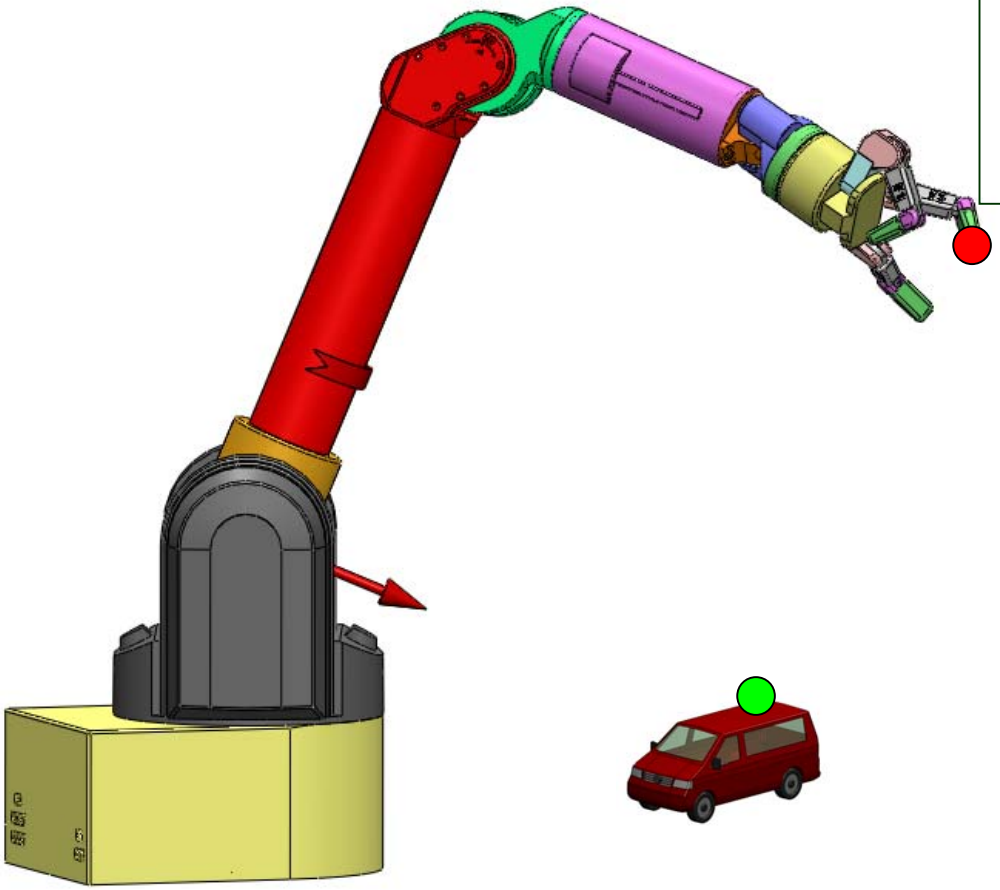


Image 1





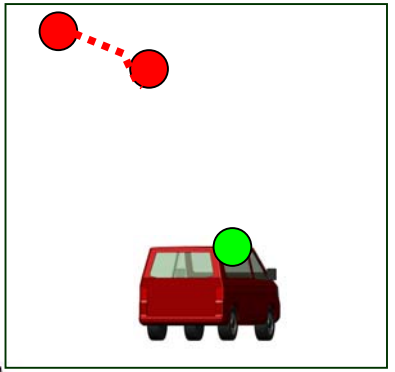
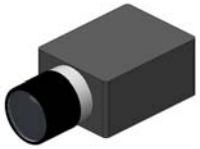


Image 2



Vision-Based Control

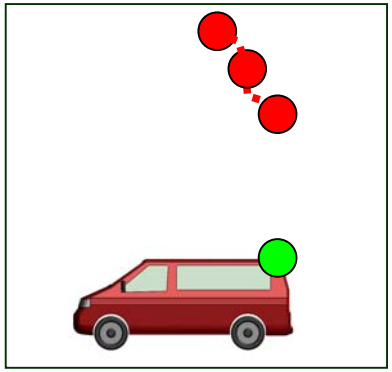
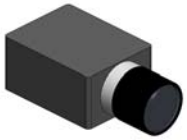
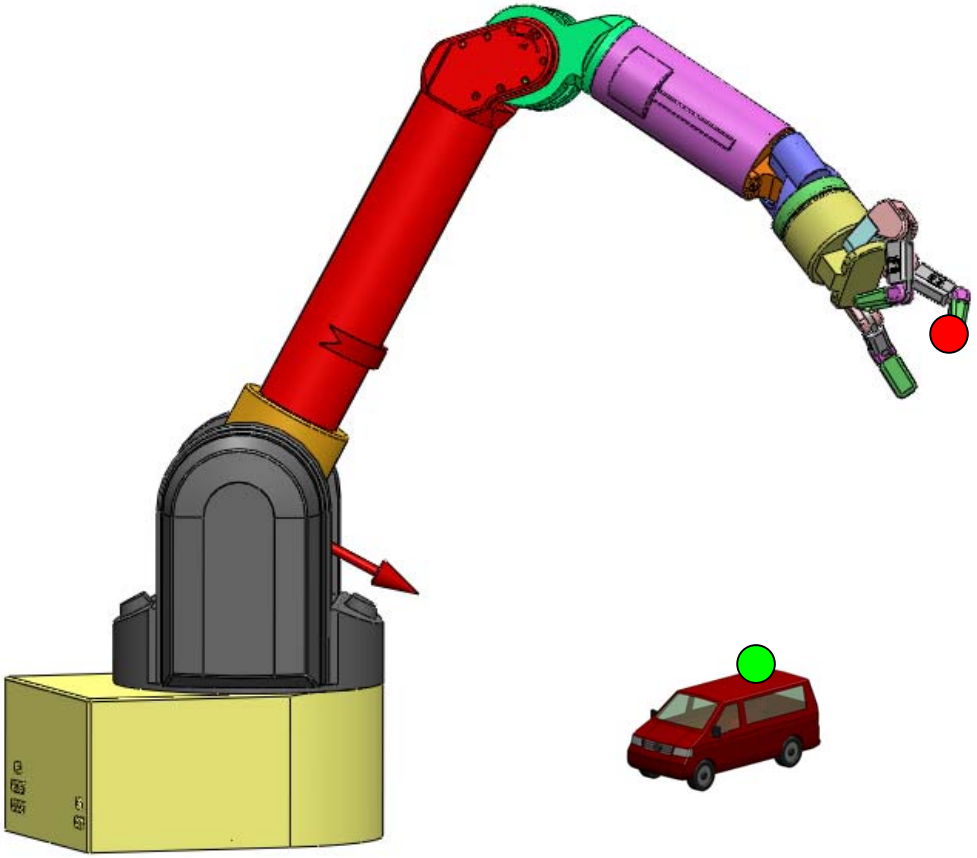


Image 1





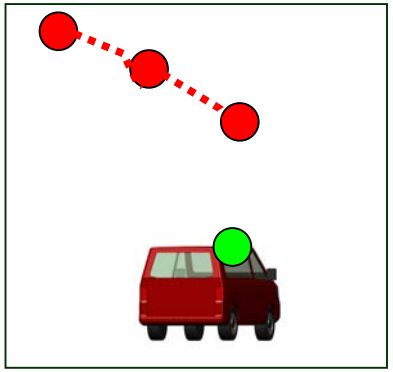
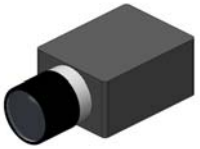


Image 2



Vision-Based Control

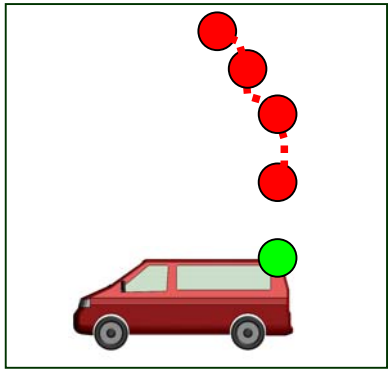
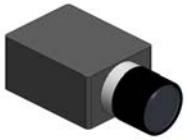
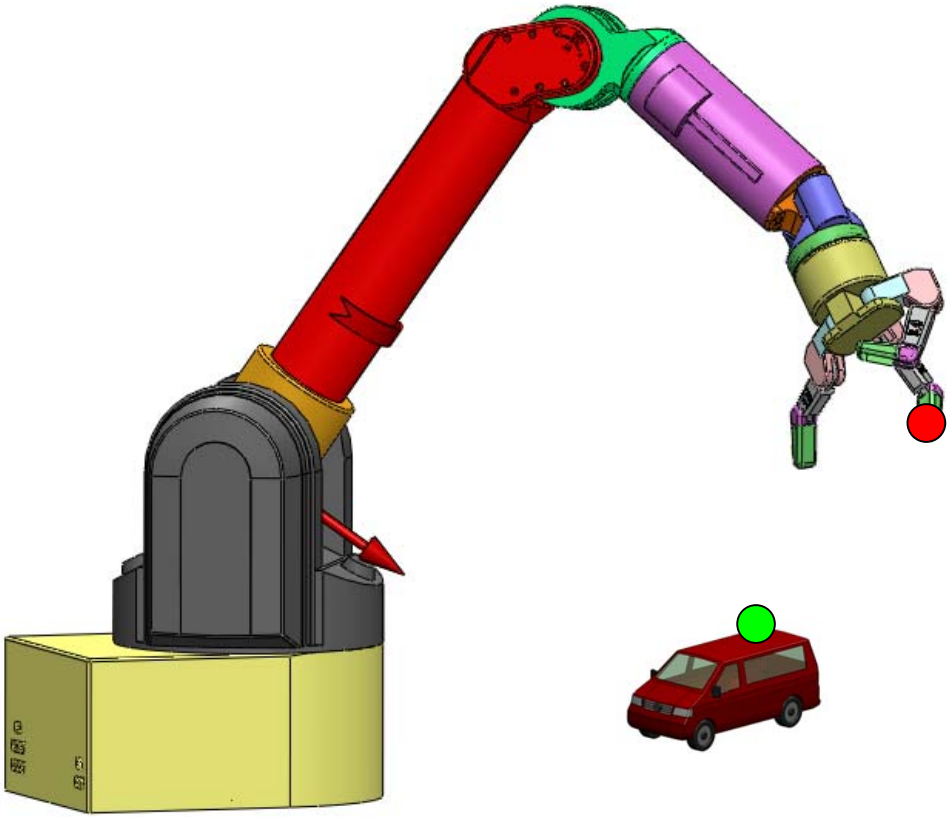


Image 1





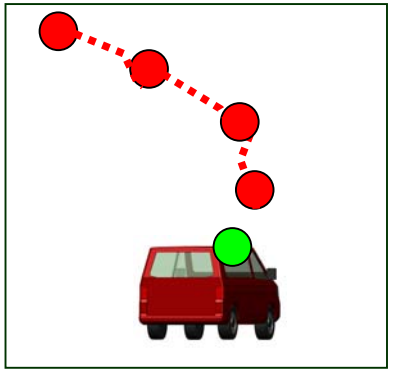
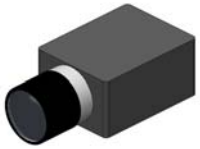


Image 2



Vision-Based Control

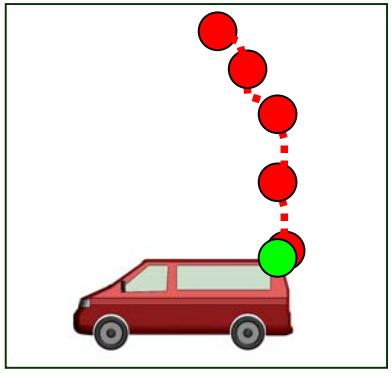
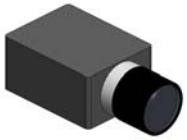
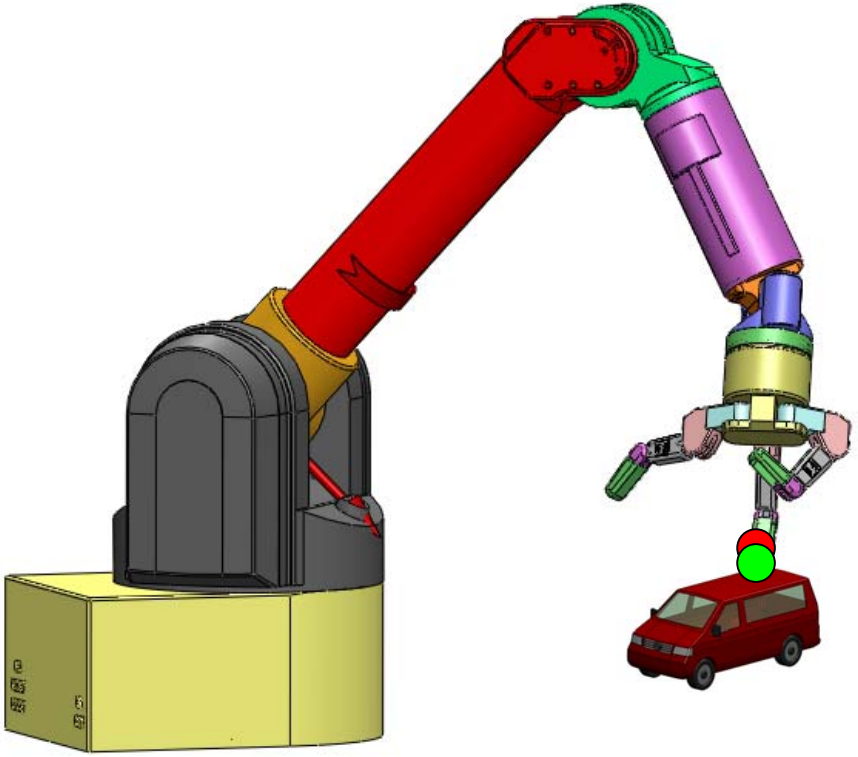


Image 1





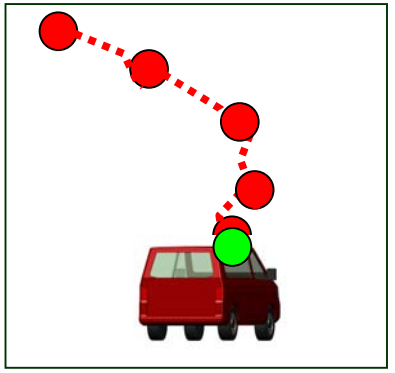


Image 2

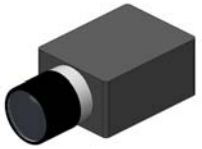
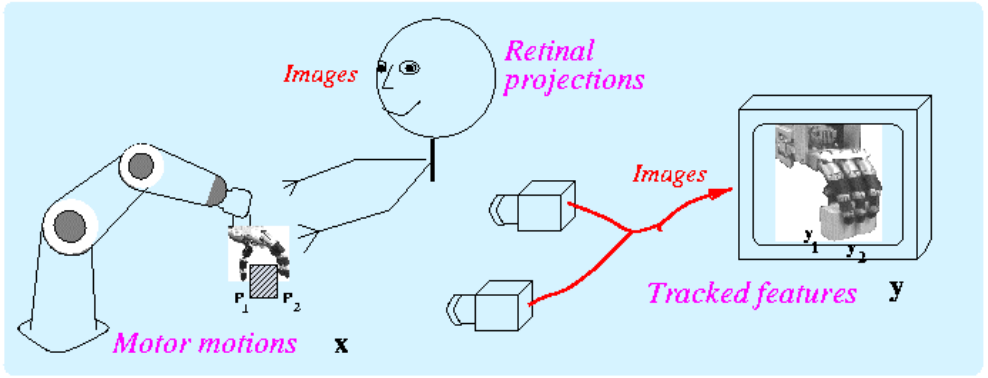


Image-based Visual Servoing

- Observed features: $\mathbf{y} = [\, y_1 \quad y_2 \quad \cdots \quad y_m \,]^T$
- Motor joint angles: $\mathbf{x} = [\, x_1 \quad x_2 \ldots \quad x_n \,]^T$
- Local linear model: $\Delta \mathbf{y} = \mathbf{J} \Delta \mathbf{x}$
- Visual servoing steps:

1 Solve: $\mathbf{y}^* - \mathbf{y}_k = \mathbf{J} \Delta \mathbf{x}$

2 Update: $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}$



Find J Method 1: Test movements along basis

- Remember: J is unknown m by n matrix

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \mathrm{f}_1}{\partial x_1} & \cdots & \frac{\partial \mathrm{f}_1}{\partial x_n} \\ \vdots & \ddots & \\ \frac{\partial \mathrm{f}_m}{\partial x_1} & & \frac{\partial \mathrm{f}_\mathrm{m}}{\partial x_n} \end{pmatrix}$$

- Assume movements

$$\begin{aligned} \Delta\mathbf{x}_1 &= [1, 0, \ldots, 0]^T \\ \Delta\mathbf{x}_2 &= [0, 1, \ldots, 0]^T \\ &\vdots \\ \Delta\mathbf{x}_n &= [0, 0, \ldots, 1]^T \end{aligned}$$

- Finite difference:

$$\mathbf{J}\mathrm{t} \begin{pmatrix} \begin{bmatrix} \vdots \\ \Delta\mathbf{y}_1 \\ \vdots \end{bmatrix} & \begin{bmatrix} \vdots \\ \Delta\mathbf{y}_2 \\ \vdots \end{bmatrix} & \cdots & \begin{bmatrix} \vdots \\ \Delta\mathbf{y}_n \\ \vdots \end{bmatrix} \end{pmatrix}$$

Find J Method 2: Secant Constraints

- Constraint along a line:
- Defines m equations

$$\Delta \mathbf{y} = \mathbf{J} \Delta \mathbf{x}$$

- Collect n arbitrary, but different measures y
- Solve for J

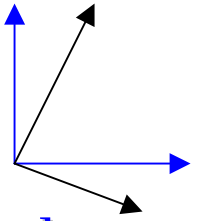
$$\begin{pmatrix} \cdots & \Delta \mathbf{y}_1^T & \cdots \\ [\cdots & \Delta \mathbf{y}_2^T & \cdots] \\ & \vdots & \\ [\cdots & \Delta \mathbf{y}_n^T & \cdots] \end{pmatrix} = \begin{pmatrix} \cdots & \Delta \mathbf{x}_1^T & \cdots \\ [\cdots & \Delta \mathbf{x}_2^T & \cdots] \\ & \vdots & \\ [\cdots & \Delta \mathbf{x}_n^T & \cdots] \end{pmatrix} \mathbf{J}^T$$

Find J Method 3: Recursive Secant Constraints

- Based on initial J and one measure pair
- Adjust J s.t. $\Delta \mathbf{y}, \Delta \mathbf{x}$
- Rank 1 update:

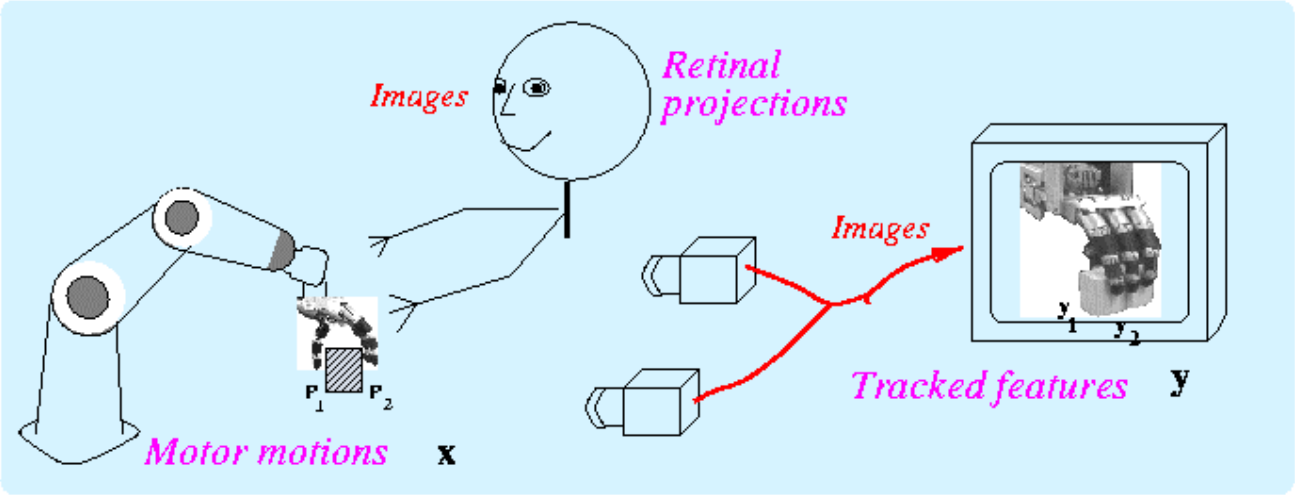
$$\Delta \mathbf{y} = \mathbf{J}_{k+1} \Delta \mathbf{x}$$

$$\hat{J}_{k+1} = \hat{J}_k + \frac{(\Delta \mathbf{y} - \hat{J}_k \Delta \mathbf{x}) \Delta \mathbf{x}^T}{\Delta \mathbf{x}^T \Delta \mathbf{x}}$$



- Consider rotated coordinates:
  - Update same as finite difference for n orthogonal moves

Achieving visual goals: Uncalibrated Visual Servoing



1. Solve for motion: $[\mathbf{y} - \mathbf{y}_k] = \mathbf{J}\Delta\mathbf{x}$
2. Move robot joints: $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$
3. Read actual visual move Δy

$$(\Delta\mathbf{y} - \hat{J}_k\Delta\mathbf{x})\Delta\mathbf{x}^T$$

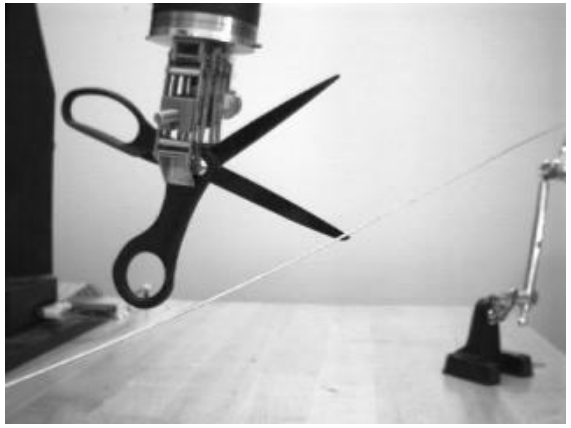
Can we always guarantee when a task is achieved/achievable?

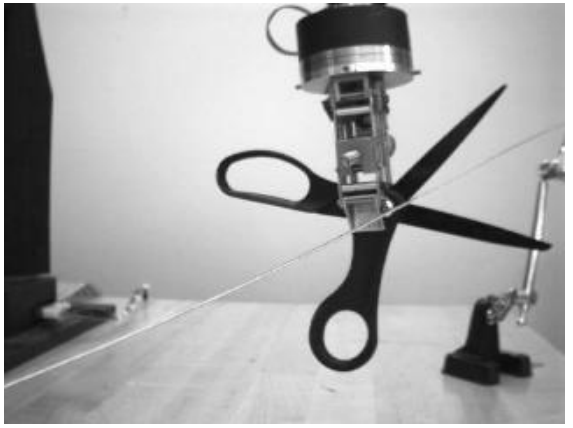
Visually guided motion control

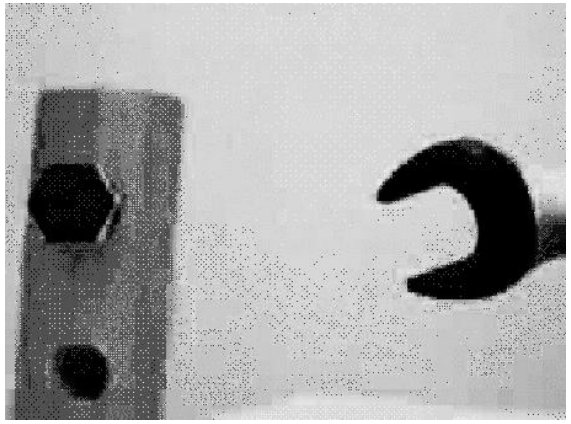
Issues:

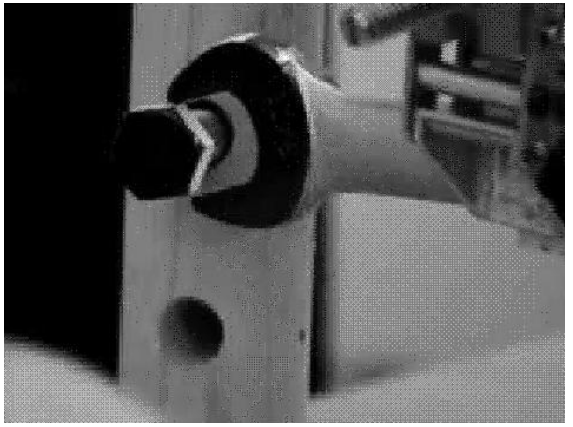
1. What tasks can be performed?
   - Camera models, geometry, visual encodings
2. How to do vision guided movement?
   - H-E transform estimation, feedback, feedforward motion control
3. How to plan, decompose and perform whole tasks?

How to specify a visual task?





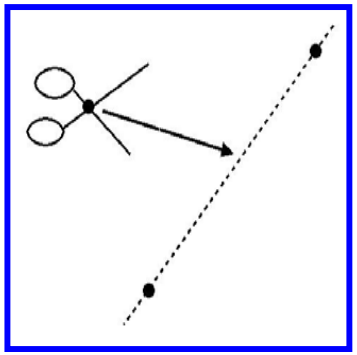


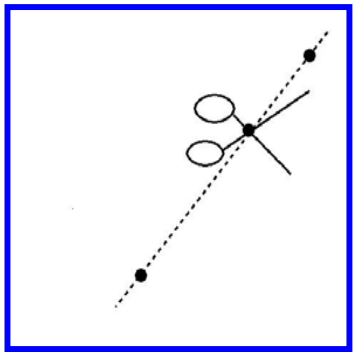


Task and Image Specifications

Task function $T(x) = 0$ Image encoding $E(y) = 0$

task space error





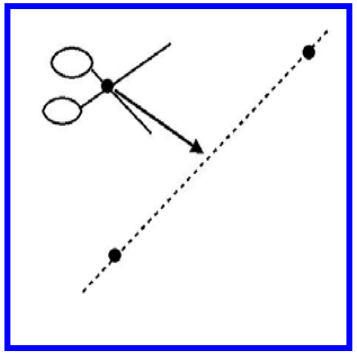


image space error

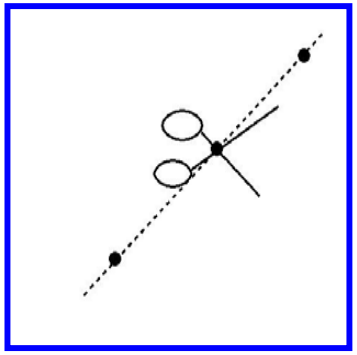
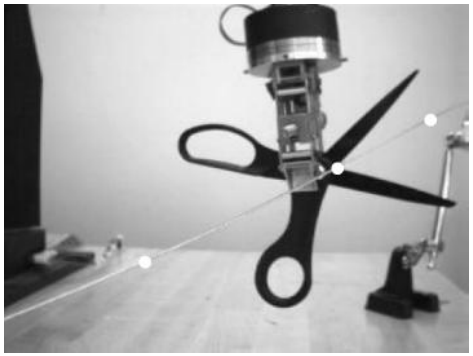


image space satisfied

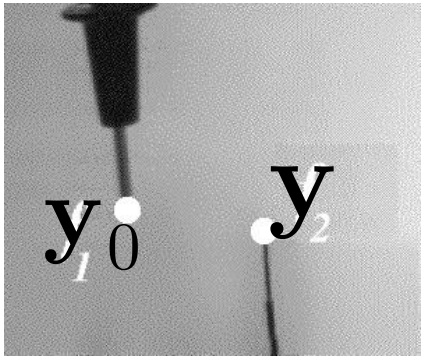


task space satisfied

Visual specifications

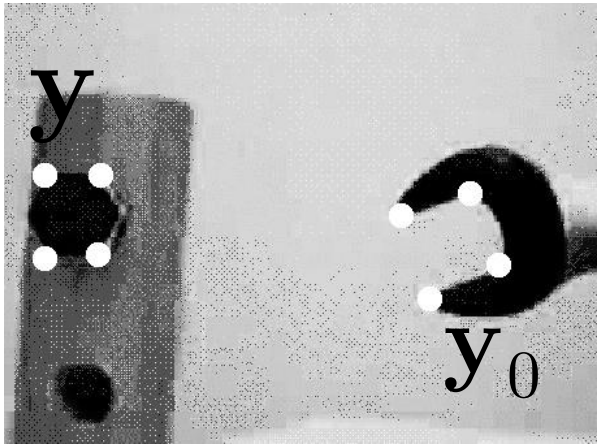
- Point to Point task "error":

$$\mathbf{E} = [\mathbf{y}_2 - \mathbf{y}_0]$$



$$\mathbf{E} = \begin{bmatrix} y_1 \\ \vdots \\ y_{16} \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_{16} \end{bmatrix}_0$$

Why 16 elements?



Visual specifications 2

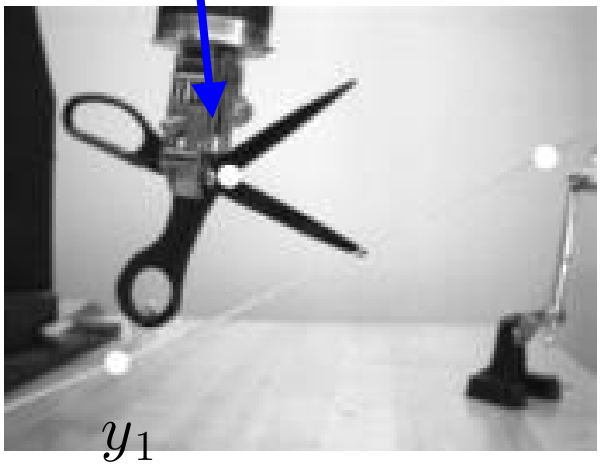
- Point to Line

Line:

$$\mathbf{E}_{pl}(\mathbf{y}, \mathbf{l}) = \begin{bmatrix} \mathbf{y}_l \cdot \mathbf{l}_l \\ \mathbf{y}_r \cdot \mathbf{l}_r \end{bmatrix}$$

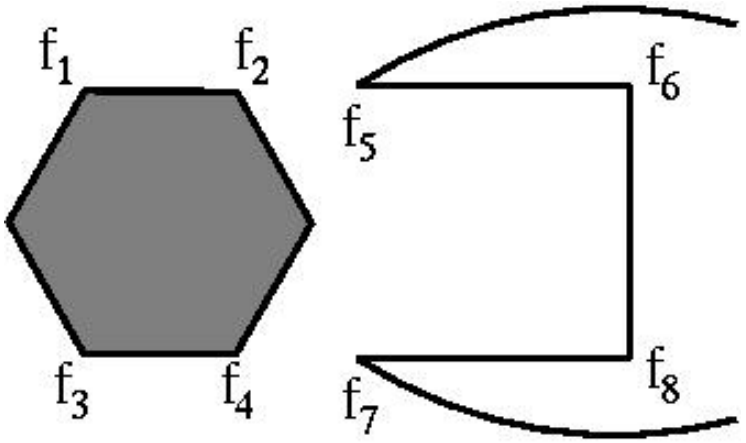
$$\mathbf{l}_l = \begin{bmatrix} y_3 \times y_1 \\ y_4 \times y_2 \end{bmatrix}$$

$$\mathbf{y}_l = \begin{bmatrix} y_5 \\ y_6 \end{bmatrix}$$



Note: y homogeneous coord.

Parallel Composition example



$$E_{\text{wrench}}(y) = \begin{pmatrix} y_2 - y_5 \\ y_4 - y_7 \\ y_6 \bullet (y_1 \times y_2) \\ y_8 \bullet (y_3 \times y_4) \end{pmatrix}$$

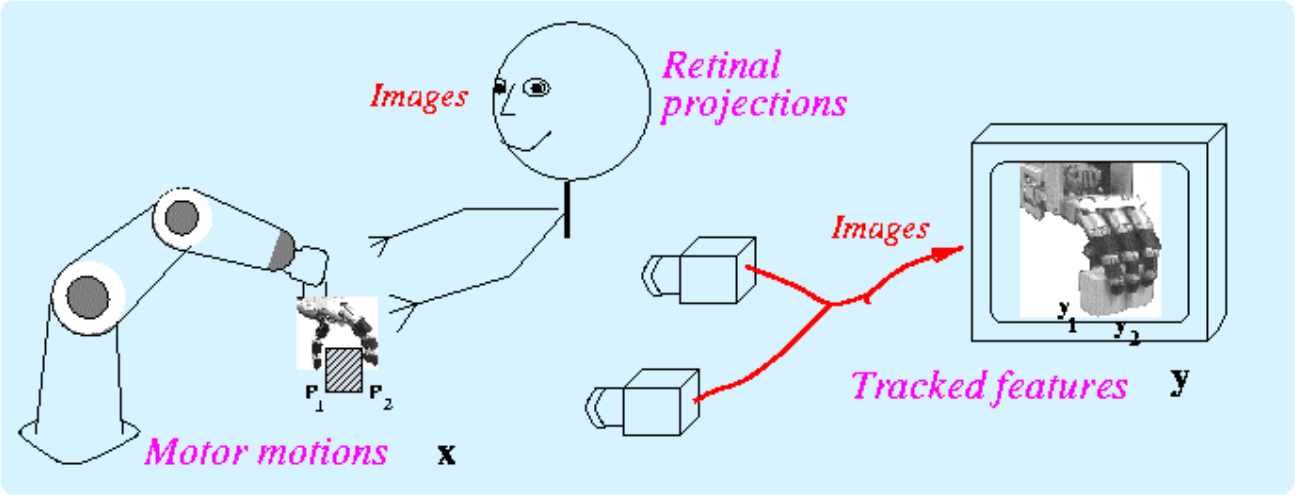
(plus e.p. checks)

Visual Specifications

Additional examples:

- Line to line
- Point to conic
  - Identify conic C from 5 pts on rim
  - Error function yCy'
- Image moments on segmented images
- Any image feature vector that encodes pose.
- Etc.

Achieving visual goals: Uncalibrated Visual Servoing



1. Solve for motion: $[\mathbf{y} - \mathbf{y}_k] = \mathbf{J}\Delta\mathbf{x}$
2. Move robot joints: $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$
3. Read actual visual move Δy

$$(\Delta\mathbf{y} - \hat{J}_k\Delta\mathbf{x})\Delta\mathbf{x}^T$$

Can we always guarantee when a task is achieved/achievable?