



# **Digital Image Processing**

**Lecture #3**

**Ming-Sui (Amy) Lee**

# [ Announcement ]

- **Class Information**

- **Homework #1**

- **Package posted on the course website**

- **Homework assignment #1**
      - **Sample codes**
      - **Submission guideline**
      - **Bonus #1**

- **Deadline: 11:59 a.m. on Mar. 15, 2016**

- **Upload to CEIBA**

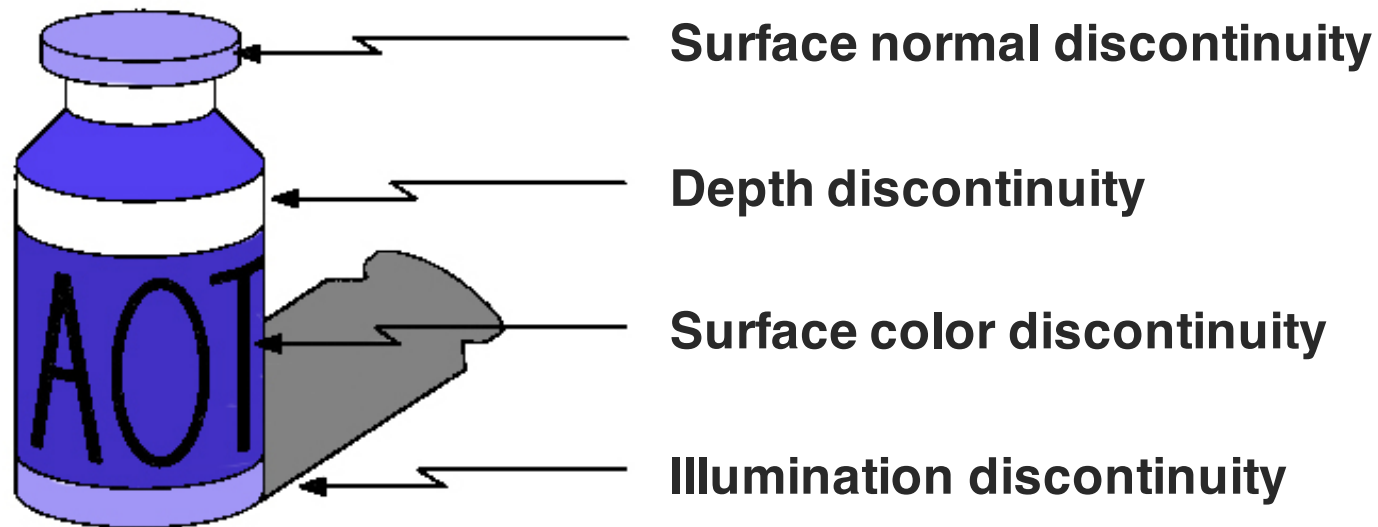
- **Electronic version**
      - **Written report**
      - **Bonus**



# Edge Crispening

# [ Edges ]

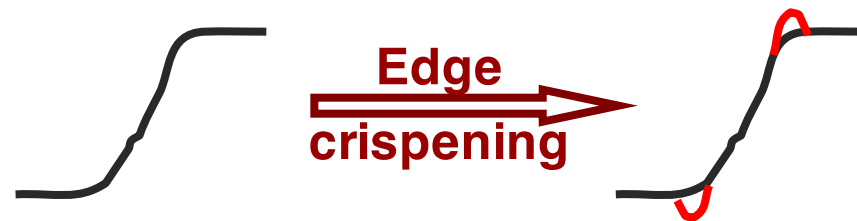
- Edges are caused by a variety of factors



# Edge Crispening

## ■ Motivation

- A photograph with accentuated edges look more appealing



$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Edge → high frequency
- High pass filtering

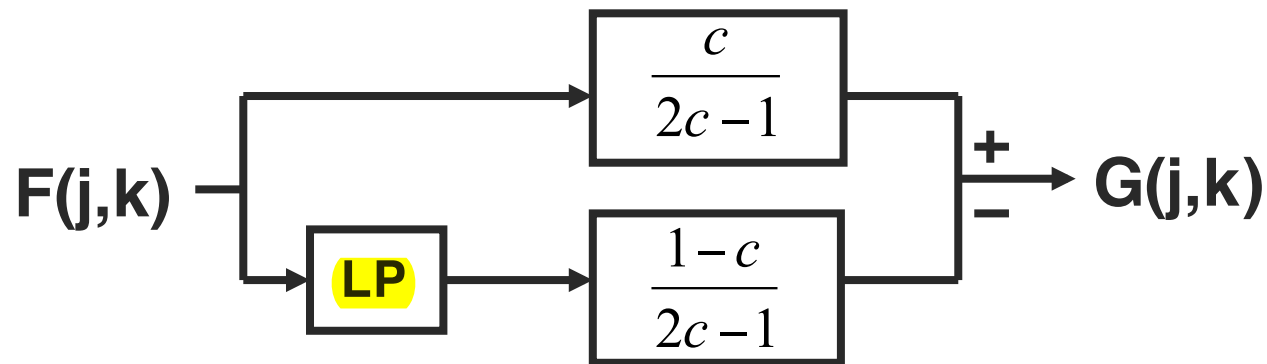
$$H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

→ amplify the noise at the same time

# Edge Crispening

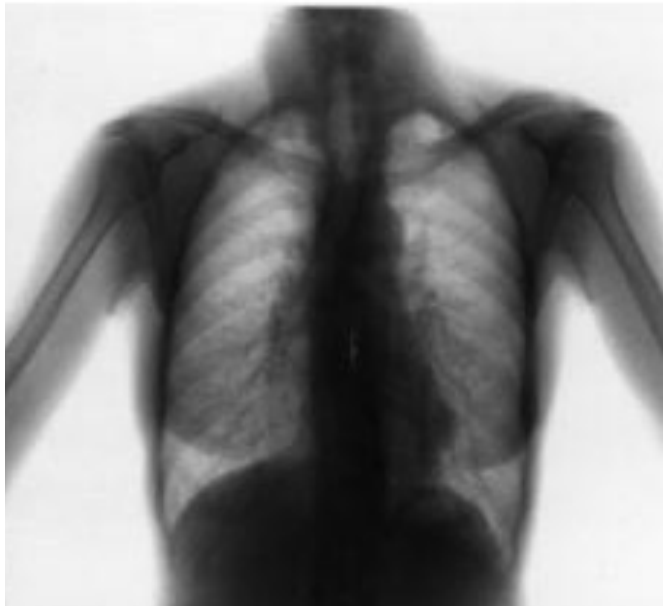
## ■ Unsharp Masking

- Appropriate combination of all-pass and low-pass filters

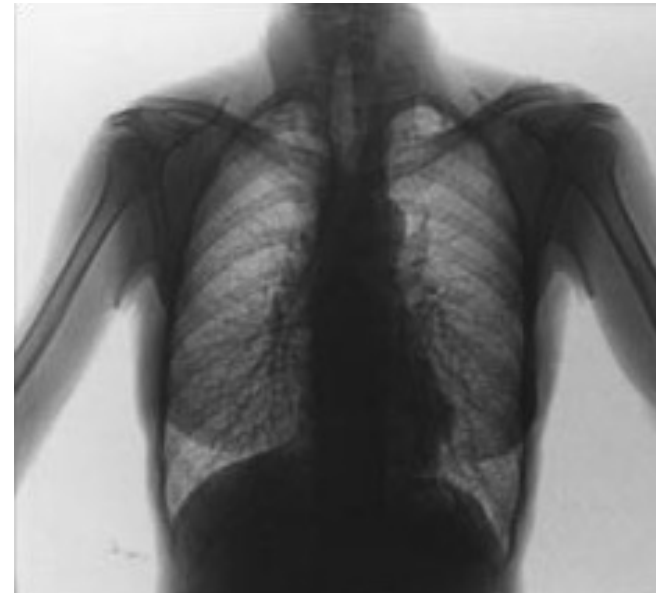


$$G(j,k) = \frac{c}{2c-1} F(j,k) - \frac{1-c}{2c-1} F_L(j,k), \quad \text{where } \frac{3}{5} \leq c \leq \frac{5}{6}$$

# [ Edge Crispening ]



Original image



After sharpening  
 $L=7, c=0.6$



(a) Normal resolution



(b) Low resolution



(c) Unsharp masking



# Edge Detection



# [ Edge Detection ]

## ■ Motivation

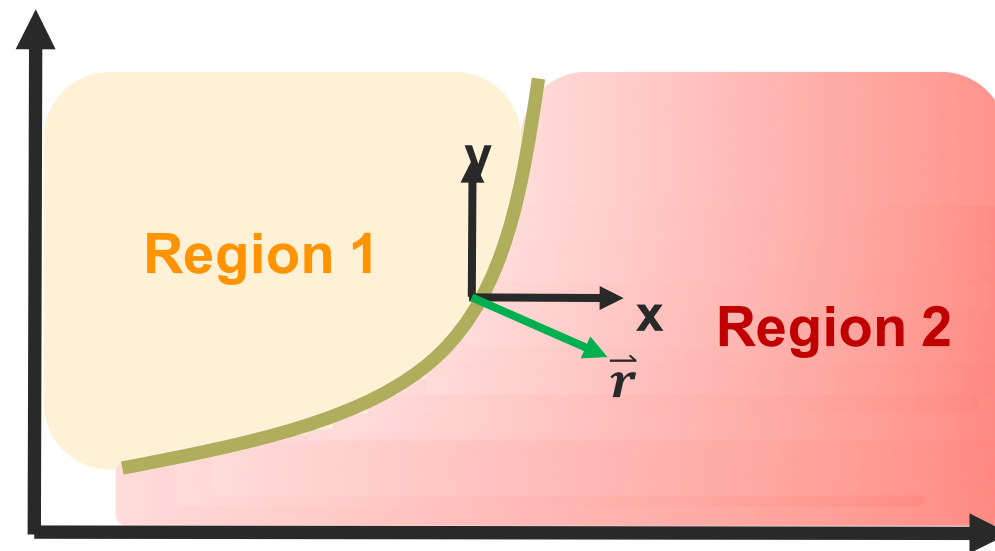
- Human eyes are more sensitive to edges
- Characterize object boundaries
- Fundamental step in image analysis
  - Segmentation, registration, identification, etc.

## ■ Edge description

- Model-based methods
  - Rarely used
- Non-parametric approaches
  - 1<sup>st</sup> and 2<sup>nd</sup> order derivatives

# Edge Detection (1<sup>st</sup> order)

## ■ Orthogonal gradient generation



$$\frac{dF}{dr} = \frac{\partial F}{\partial x} \frac{\partial x}{\partial r} + \frac{\partial F}{\partial y} \frac{\partial y}{\partial r} = \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial r} \end{pmatrix} = \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix}$$

# Edge Detection (1<sup>st</sup> order)

## ■ Orthogonal gradient generation

- When  $\begin{pmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \end{pmatrix}$  (gradient direction) is **parallel** to  $\begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix}$

$\left\| \frac{dF}{dr} \right\|$  has **maximum** value

$$\Rightarrow \left\| \frac{dF}{dr} \right\| = \left\| \begin{pmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \end{pmatrix} \right\| = \sqrt{\left( \frac{\partial F}{\partial x} \right)^2 + \left( \frac{\partial F}{\partial y} \right)^2} \quad \theta = \tan^{-1} \left( \frac{\frac{\partial F}{\partial y}}{\frac{\partial F}{\partial x}} \right)$$

Magnitude

Orientation

# [ Edge Detection (1<sup>st</sup> order) ]

## ■ Discrete case

### ○ Approximation I – 3 points

#### ■ Row gradient

$$\frac{\partial F}{\partial x}(j, k) \cong F(j, k) - F(j, k - 1) = G_R(j, k)$$

#### ■ Column gradient

$$\frac{\partial F}{\partial y}(j, k) \cong F(j, k) - F(j + 1, k) = G_C(j, k)$$

$A_0$	$A_1$	$A_2$
$A_7$	$F(j, k)$	$A_3$
$A_6$	$A_5$	$A_4$

$$\Rightarrow G(j, k) = \sqrt{G_R^2(j, k) + G_C^2(j, k)} \quad \theta(j, k) = \tan^{-1} \left( \frac{G_C(j, k)}{G_R(j, k)} \right)$$

# [ Edge Detection (1<sup>st</sup> order) ]

## ■ Example

$$G_R(j,k) = F(j,k) - F(j,k-1)$$

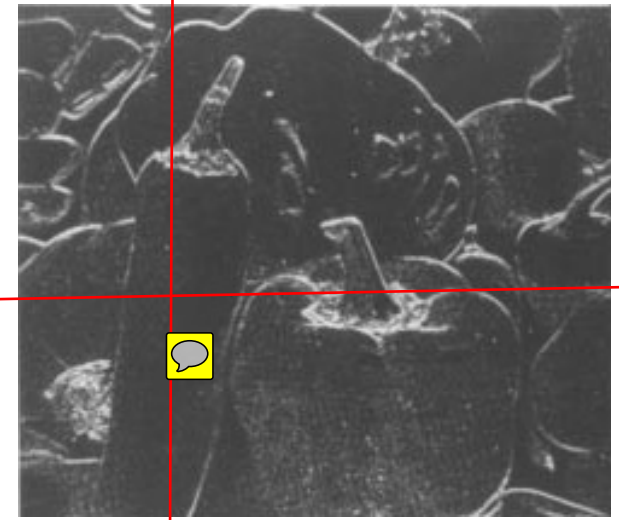
$$G_C(j,k) = F(j,k) - F(j+1,k)$$



Original image



Horizontal magnitude



Vertical magnitude

# [ Edge Detection (1<sup>st</sup> order) ]

## ■ Discrete case

### ○ Approximation II – 4 points

### ○ Roberts cross differentiation (0~90→45~135)

$A_0$	$A_1$	$A_2$
$A_7$	$F(j, k)$	$A_3$
$A_6$	$A_5$	$A_4$

#### ■ Row gradient

$$G_1(j, k) = F(j, k) - F(j + 1, k + 1)$$

#### ■ Column gradient

$$G_2(j, k) = F(j, k + 1) - F(j + 1, k)$$

$$\Rightarrow G(j, k) = \sqrt{G_1^2(j, k) + G_2^2(j, k)} \quad \theta(j, k) = \tan^{-1} \left( \frac{G_2(j, k)}{G_1(j, k)} \right) + \frac{\pi}{4}$$

# Edge Detection (1<sup>st</sup> order)

## ■ Discrete case

### ○ Approximation III –9points

$A_0$	$A_1$	$A_2$
$A_7$	$F(j, k)$	$A_3$
$A_6$	$A_5$	$A_4$

#### ■ Row gradient

$$G_R(j, k) = \frac{1}{K+2} [(A_2 + KA_3 + A_4) - (A_0 + KA_7 + A_6)]$$

#### ■ Column gradient

$$G_C(j, k) = \frac{1}{K+2} [(A_0 + KA_1 + A_2) - (A_6 + KA_5 + A_4)]$$

$$\Rightarrow G(j, k) = \sqrt{G_R^2(j, k) + G_C^2(j, k)} \quad \theta(j, k) = \tan^{-1} \left( \frac{G_C(j, k)}{G_R(j, k)} \right)$$

■ k=1: Prewitt Mask ; k=2: Sobel Mask

# Edge Detection (1<sup>st</sup> order)



- Compute row and column gradients

- ◆ Analyze the statistics of the magnitude (histogram)

- Pick a threshold  $T$
- If  $G(j,k) \geq T$ 
  - set it as an edge point
  - otherwise ( If  $G(j,k) < T$  )
    - non-edge point
- **Q: How to select  $T$ ?**
- ◆ Examine the cumulative distribution function



# [ Canny Edge Detector ]

## ■ Why Canny?

### ○ Good Detection

- The optimal detector must minimize the probability of false positives as well as false negatives

### ○ Good Localization

- The edges detected must be as close as possible to the true edges

### ○ Single Response Constraint

- The detector must return one point only for each edge point

# Canny Edge Detector

## ■ Five Steps:

- **Noise reduction**
- Compute gradient magnitude and orientation
- Non-maximal suppression
- Hysteretic thresholding
- Connected component labeling method



Smooth the image  
with a Gaussian filter



Example:

5x5 Gaussian filter with  $\sigma = 1.4$

$$F_{NR} = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * F$$



# Canny Edge Detector

## ■ Five Steps:

- Noise reduction
- **Compute gradient magnitude and orientation**
- Non-maximal suppression
- Hysteretic thresholding
- Connected component labeling method

$$G(j,k) = \sqrt{G_R^2(j,k) + G_C^2(j,k)}$$

$$\theta(j,k) = \tan^{-1} \left( \frac{G_C(j,k)}{G_R(j,k)} \right)$$



Magnitude

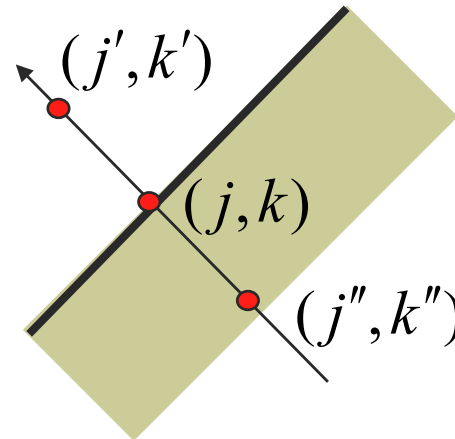
# Canny Edge Detector

## ■ Five Steps:

- Noise reduction
- Compute gradient magnitude and orientation
- **Non-maximal suppression**
- Hysteretic thresholding
- Connected component labeling method

Search the nearest neighbors  $(j', k')$  and  $(j'', k'')$  along the edge normal

$$G_N(j, k) = \begin{cases} G(j, k) & \text{if } G(j, k) > G(j', k') \\ & \text{and } G(j, k) > G(j'', k'') \\ 0 & \end{cases}$$



# Canny Edge Detector

## ■ Five Steps:

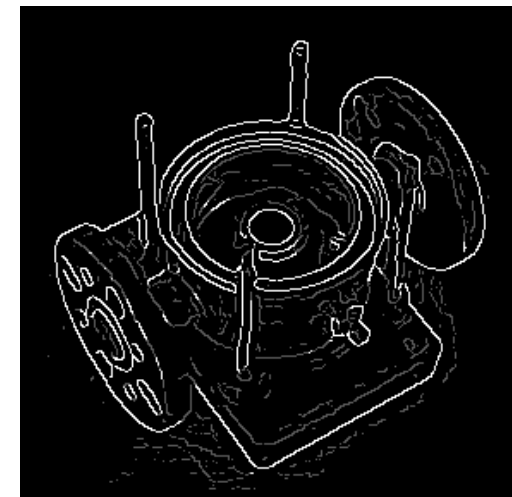
- Noise reduction
- Compute gradient magnitude and orientation
- Non-maximal suppression
- **Hysteretic thresholding**
- Connected component labeling method

Label each pixels according to two threshold:  $T_H, T_L$

$G_N(x, y) \geq T_H$       Edge Pixel

$T_H > G_N(x, y) \geq T_L$       Candidate Pixel

$G_N(x, y) < T_L$       Non-edge Pixel

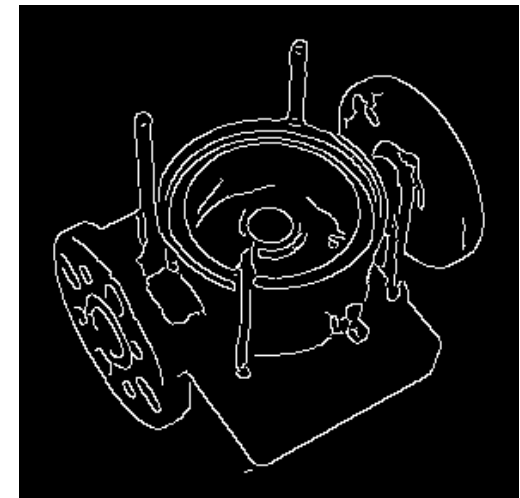
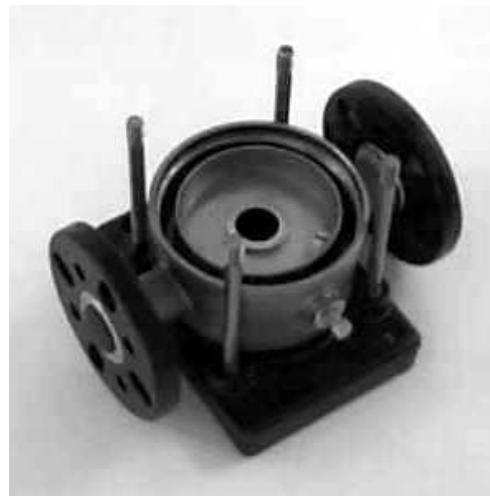


# Canny Edge Detector

## ■ Five Steps:

- Noise reduction
- Compute gradient magnitude and orientation
- Non-maximal suppression
- Hysteretic thresholding
- **Connected component labeling method**

If a candidate pixel is connected to an edge pixel directly or via another candidate pixel then it is declared as an edge pixel



Edge Map

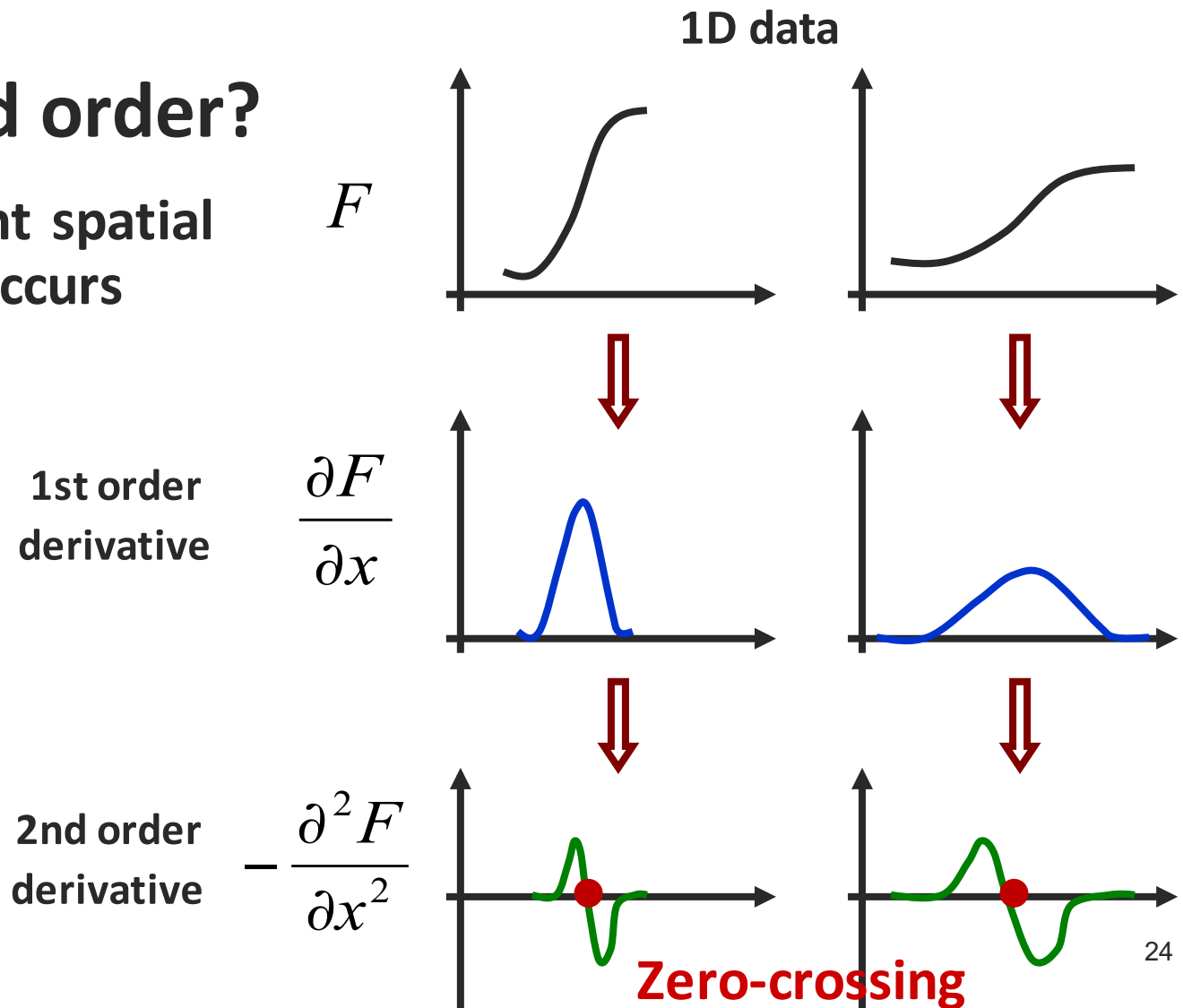


# **Edge Detection – Part II**

# [ Edge Detection (2<sup>nd</sup> order) ]

## ■ Why 2nd order?

- Significant spatial change occurs





# Edge Detection (2<sup>nd</sup> order)

## ■ Laplacian Generation

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \Rightarrow \nabla^2 F(x, y) = \frac{\partial^2 F(x, y)}{\partial x^2} + \frac{\partial^2 F(x, y)}{\partial y^2}$$

## ■ Discrete Approximation

$$-\frac{\partial^2}{\partial x^2} \cong \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \Rightarrow 2f(x) - (f(x+h) + f(x-h))$$

### ○ By Taylor series expansion

$$\begin{aligned} & 2f(x) - (f(x+h) + f(x-h)) \\ = & 2f(x) - \left[ f(x) + hf'(x) + \frac{h^2}{2} f''(x) + f(x) + (-h)f'(x) + \frac{h^2}{2} f''(x) + \dots \right] \\ \cong & \boxed{-h^2 f''(x)} \end{aligned}$$

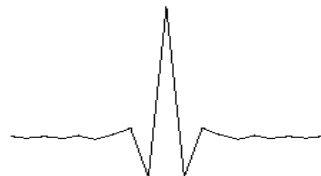
# [ Edge Detection (2<sup>nd</sup> order) ]

## ■ Discrete Approximation

$$-\frac{\partial^2}{\partial x^2} \cong \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \quad -\frac{\partial^2}{\partial y^2} \cong \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}^T$$

## ○ combine together

$$-\nabla^2 \cong \frac{1}{4} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}; \quad \nabla^2 \cong \frac{1}{4} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



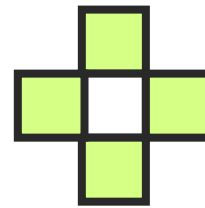
gain-normalized

# Edge Detection (2<sup>nd</sup> order)

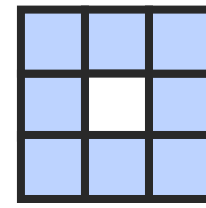
## ■ Laplacian impulse response

### ○ four-neighbor

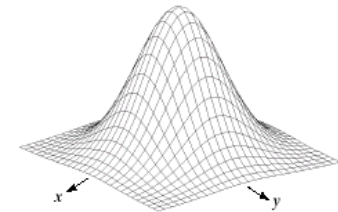
$$H = \frac{1}{4} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



4-neighbor



8-neighbor



### ○ eight-neighbor

$$H = \frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

non-separable

$$H = \frac{1}{8} \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$$

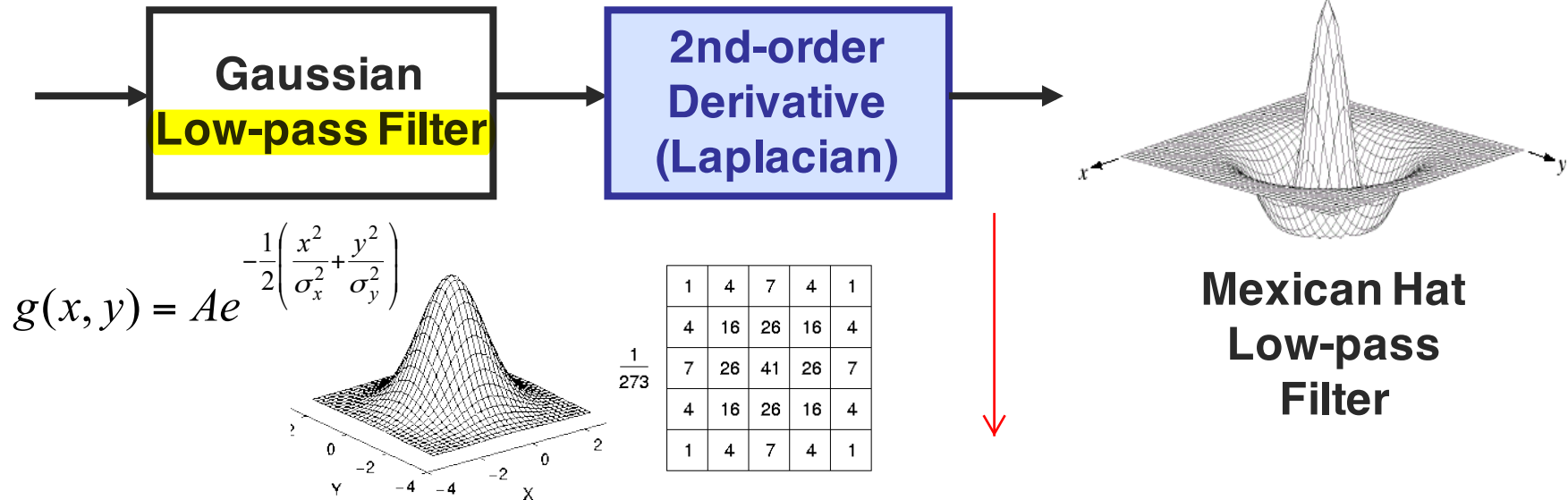
separable

$$H_1 = \frac{1}{8} \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

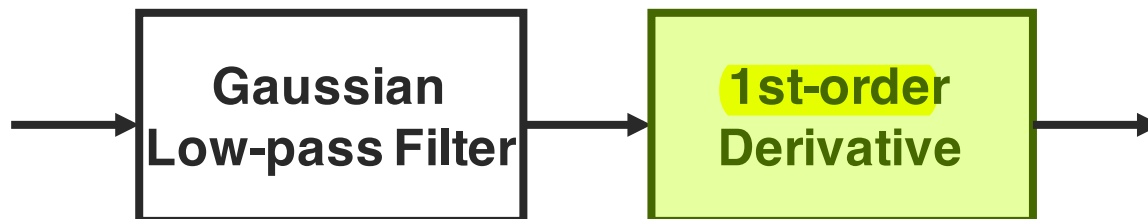
$$H_2 = \frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

# Edge Detection (2<sup>nd</sup> order)

## ■ Laplacian of Gaussian (LOG) – p.474



## ■ Difference of Gaussians (DOG)

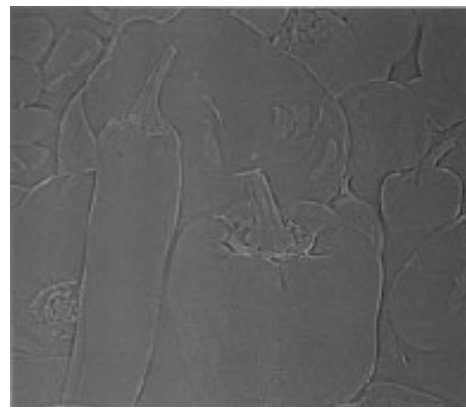


# [ Edge Detection (2<sup>nd</sup> order) ]

## ■ Examples



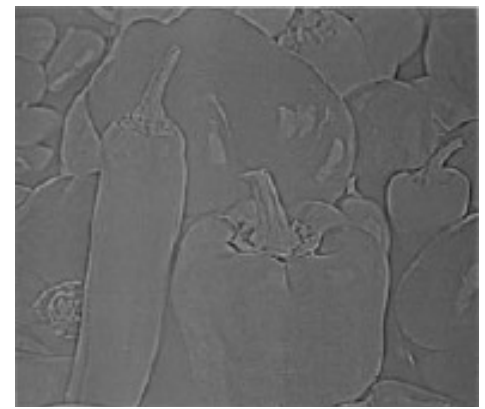
Four-neighbor



Eight-neighbor



Separable  
eight-neighbor



Laplacian of  
Gaussian (LOG)

**Are we done yet?**

# [ Edge Detection (2<sup>nd</sup> order) ]

## ■ 2nd Order Edge Detection



### ○ How to detect zero-crossing?

- many ways

# [ Edge Detection (2<sup>nd</sup> order) ]

## ■ Zero-crossing



### 3 steps:

- Generate the histogram of  $G$
- Set up a threshold to separate zero and non-zero,  $G'$
- For  $G'(j,k)=0$ , decide whether  $(j,k)$  is a zero-crossing point

# [ Edge Detection (2<sup>nd</sup> order) ]

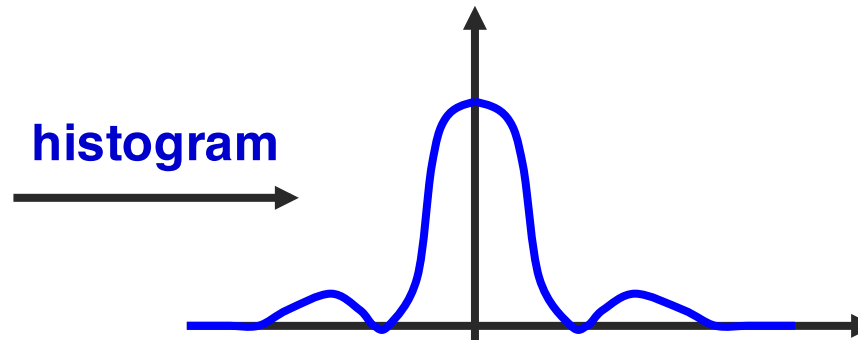
## ■ Zero-crossing

### ○ 3 steps:

- **Generate the histogram of  $G$**
- Set up a threshold to separate zero and non-zero to get  $G'$
- For  $G'(j,k)=0$ , decide whether  $(j,k)$  is a zero-crossing point



histogram



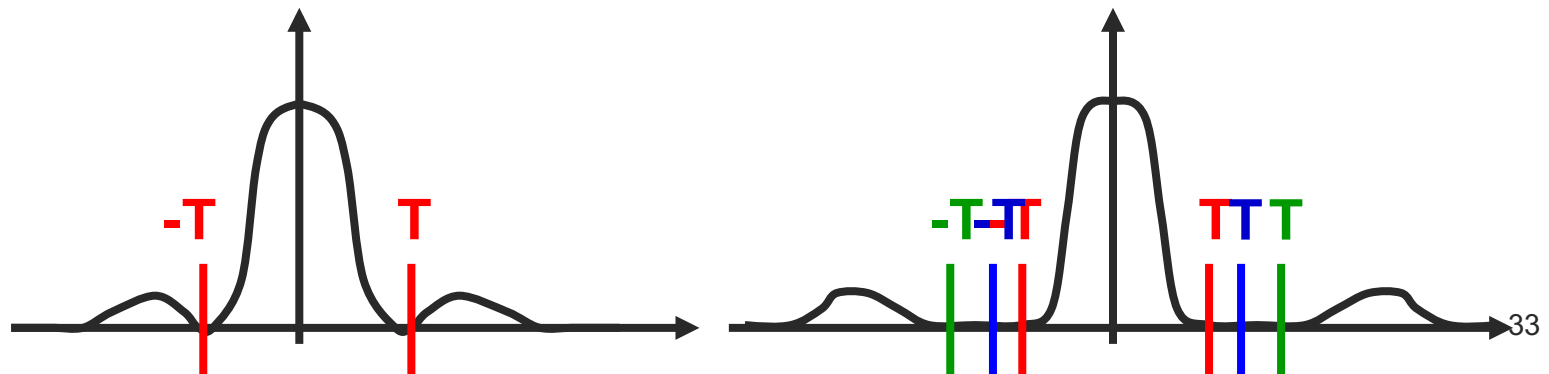


# [ Edge Detection (2<sup>nd</sup> order) ]

## ■ Zero-crossing

### ○ 3 steps:

- Generate the histogram of  $G$
- **Set up a threshold to separate zero and non-zero to get  $G'$**   $|G(j,k)| \leq T \Rightarrow G'(j,k) = 0$
- For  $G'(j,k)=0$ , decide whether  $(j,k)$  is a zero-crossing point



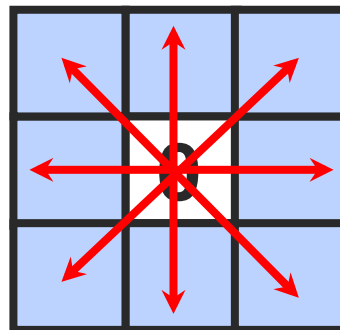
# [ Edge Detection (2<sup>nd</sup> order) ]

## ■ Zero-crossing

### ○ 3 steps:

- Generate the histogram of  $G$
- Set up a threshold to separate zero and non-zero to get  $G'$
- For  $G'(j,k)=0$ , decide whether  $(j,k)$  is a zero-crossing point → edge map

$$G'(j,k) = 0$$



$$\{-1,0,1\}$$

# [ Edge Detection (2<sup>nd</sup> order) ]

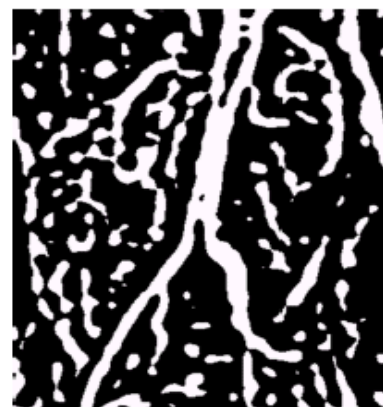
## ■ Example



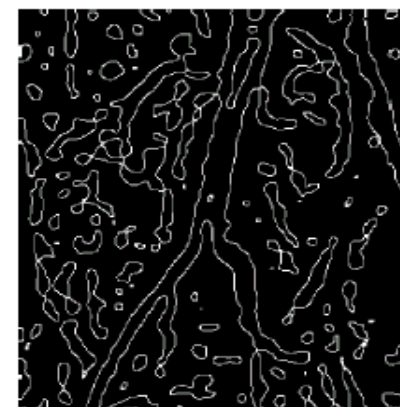
Original



Laplacian



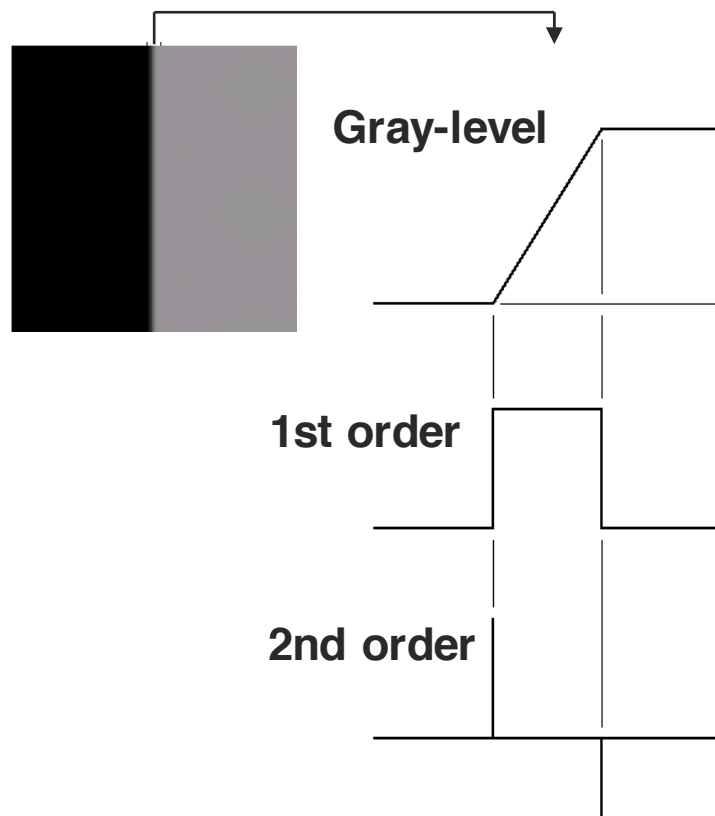
Thresholded



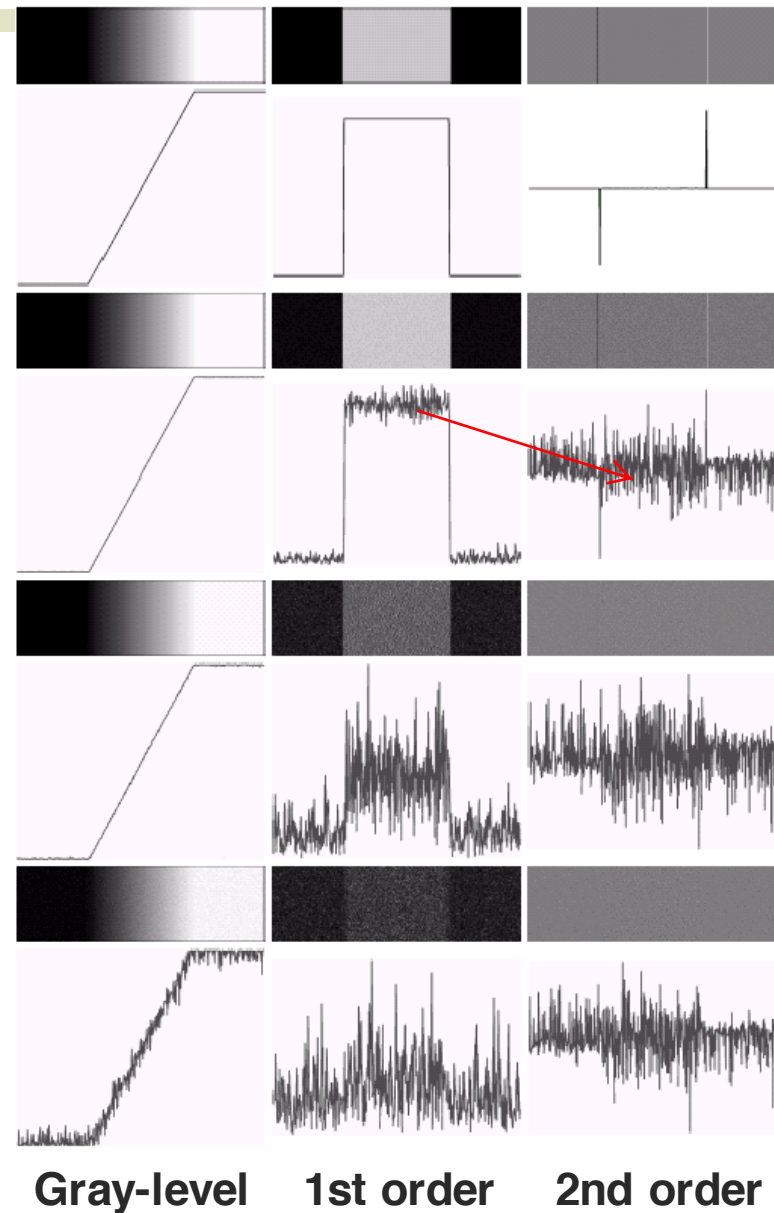
Edge map

# Edge Detection

## Noisy image

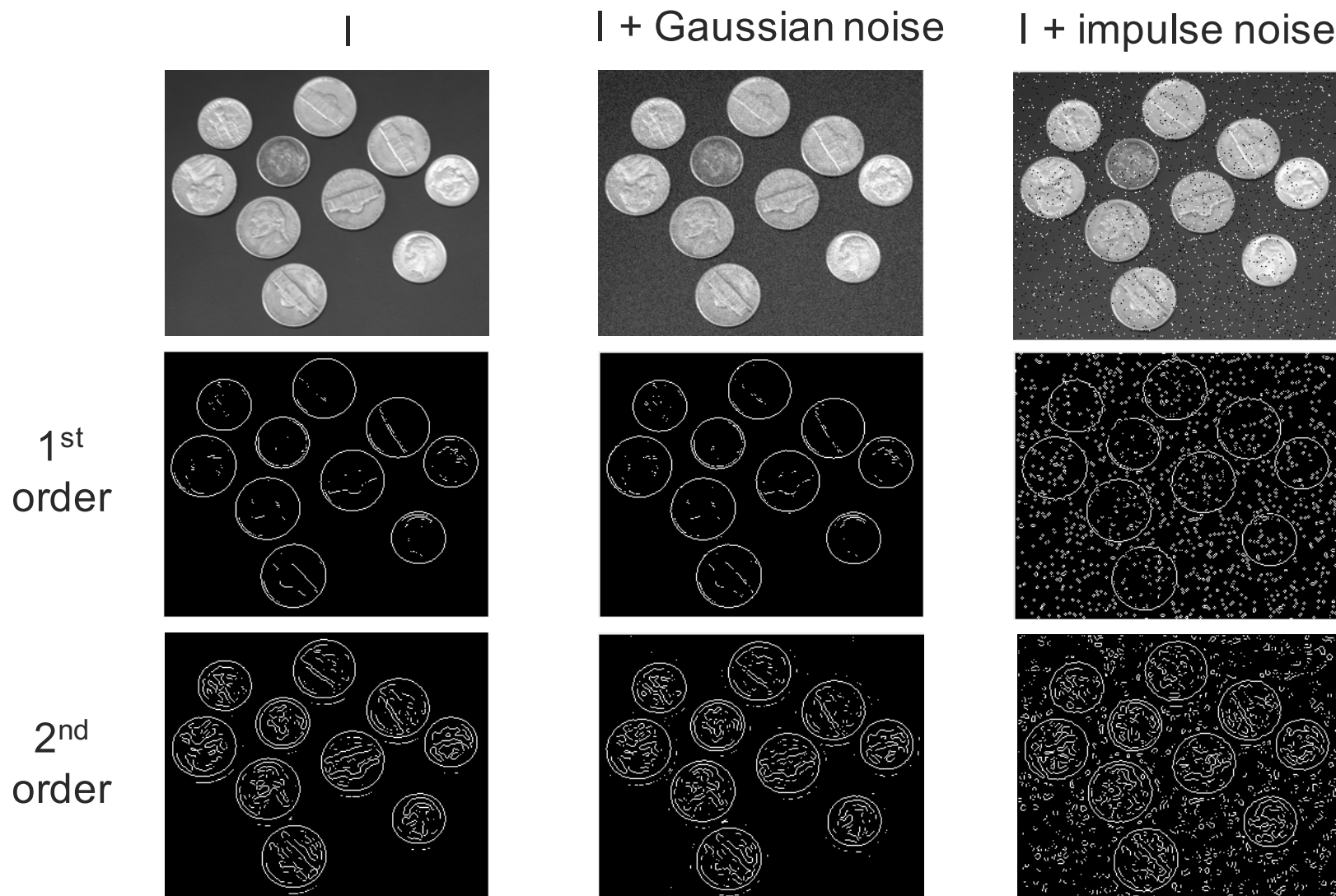


“Pre-processing”



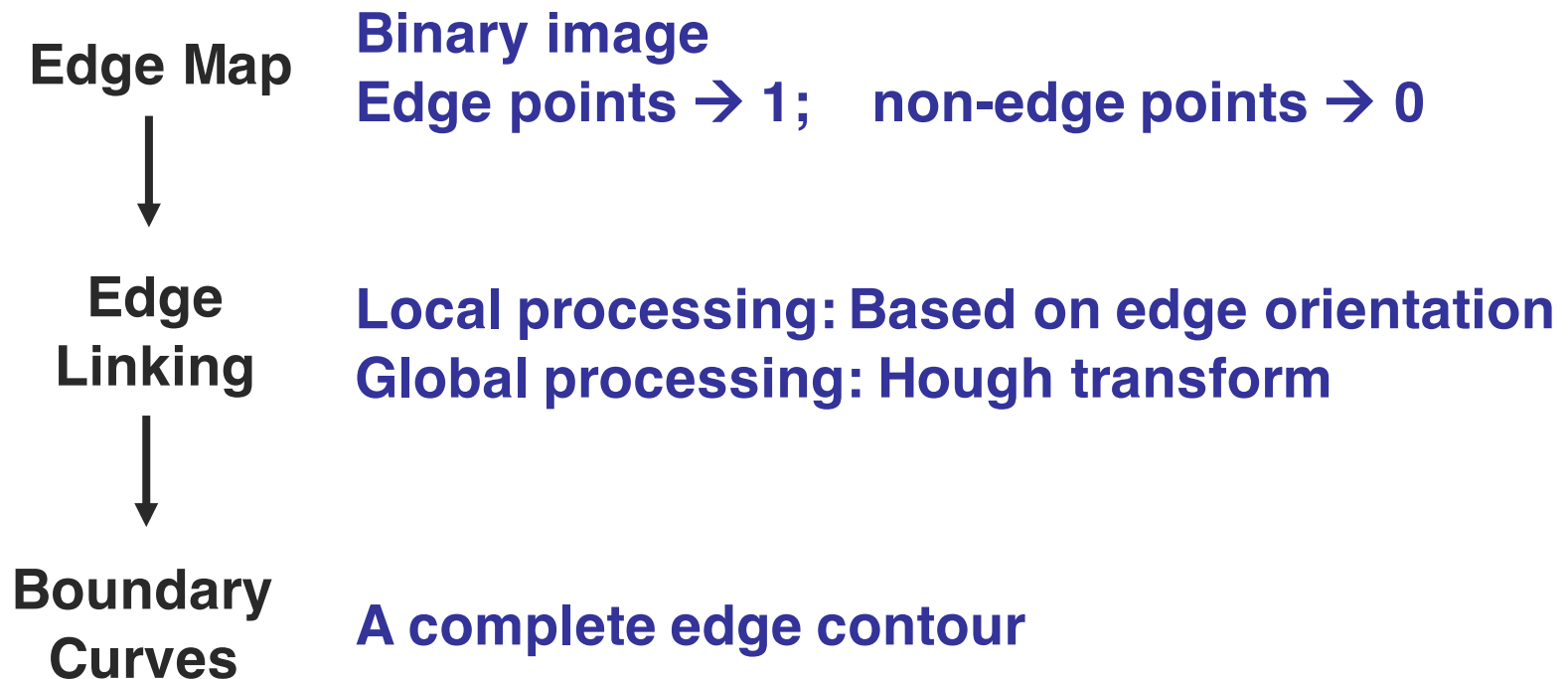
very sensitive to noise for 2nd order filter

# Edge Detection



# Edge Detection

## ■ Post-Processing





**Review**

# [Review]

## ■ Noise Cleaning

- Uniform noise → low-pass filtering
- Impulse noise → non-linear filtering
- Mixed noise → ?

## ■ Edge Crispening

- Unsharp masking

## ■ Edge Detection

- 1<sup>st</sup>-order edge detection -- threshold
- 2<sup>nd</sup>-order edge detection -- zero-crossing





# **Geometrical Modification**

# Geometrical Modification

## ■ Goal

- Translate, scale, rotate, reflect or nonlinear warp an image

## ■ Applications

- Zoom-in/zoom-out
- Image registration
- Image mosaicking
- Special effects
  - Use 2D image to simulate the 3D environment
  - <http://www.erich3d.com/>
- Etc.

# Geometrical Modification

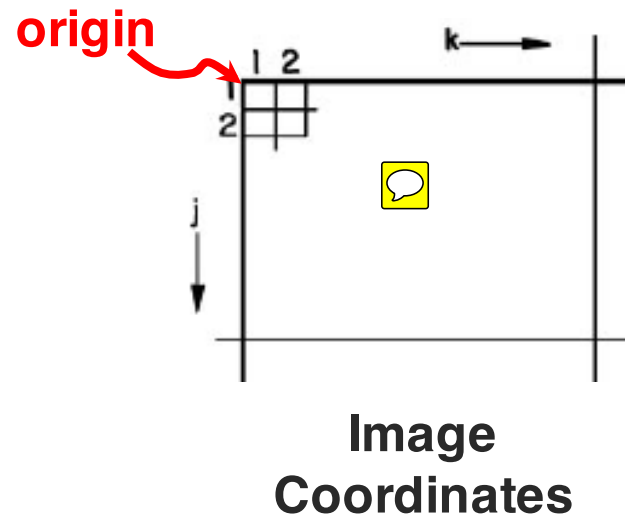
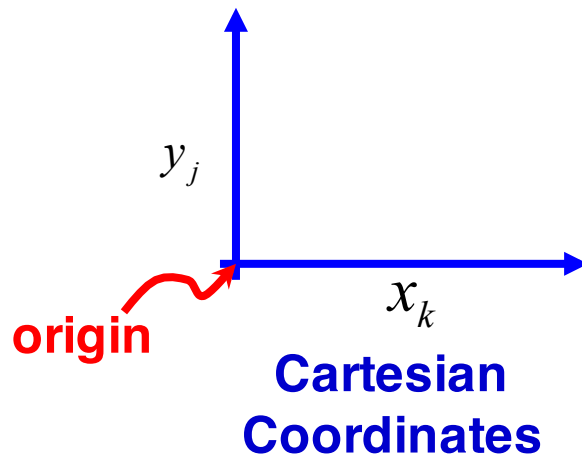
## ■ Coordinates

### ○ Geometrical transformations

#### ■ Cartesian coordinates

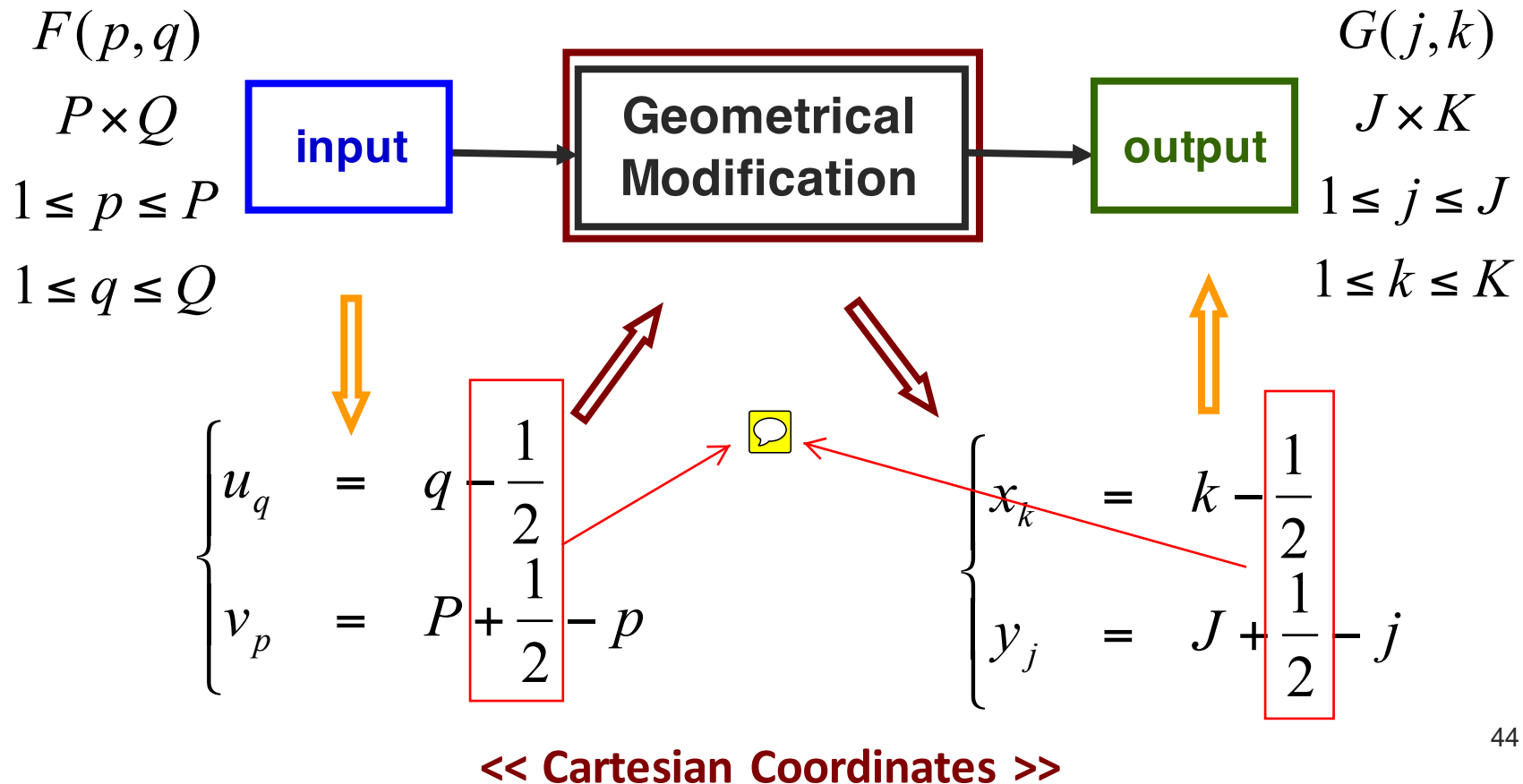
### ○ Discrete image

#### ■ Cartesian coordinates v.s. Image coordinates



# Geometrical Modification

## Linear/Affine coordinates transformation

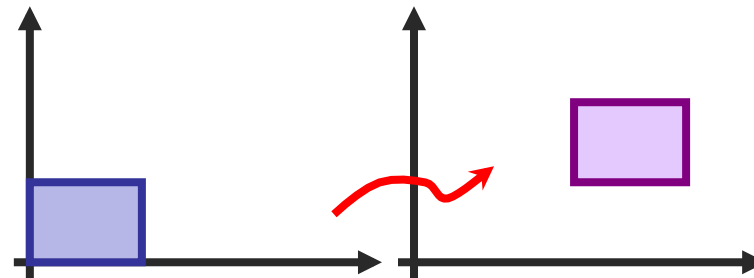


# Geometrical Modification

## Translation (Shift)

$$\begin{cases} x_k &= u_q + t_x \\ y_j &= v_p + t_y \end{cases}$$

substitute  $\begin{cases} u_q &= q - \frac{1}{2} \\ v_p &= P + \frac{1}{2} - p \end{cases}$



and  $\begin{cases} x_k &= k - \frac{1}{2} \\ y_j &= J + \frac{1}{2} - j \end{cases}$

$$\Rightarrow \begin{cases} k' &= q + t_x \\ j' &= p - (P - J) - t_y \end{cases}$$

Forward treatment

$$\begin{cases} k &= q' + t_x \\ j &= p' - (P - J) - t_y \end{cases}$$

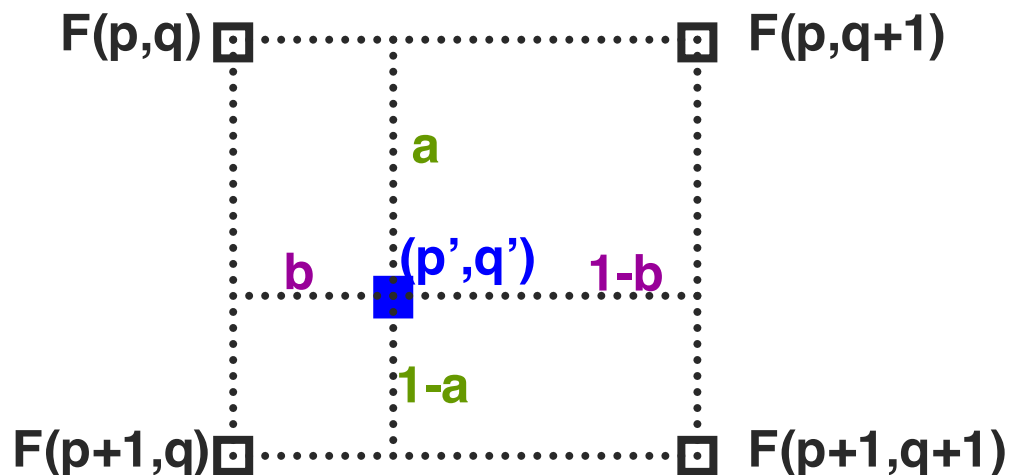
Backward treatment

**Better**

# Geometrical Modification

## ■ Translation (Shift)

- Non-integer pixel positions
- i.e. How to compute  $p'$  and  $q'$ ?
  - Bilinear interpolation

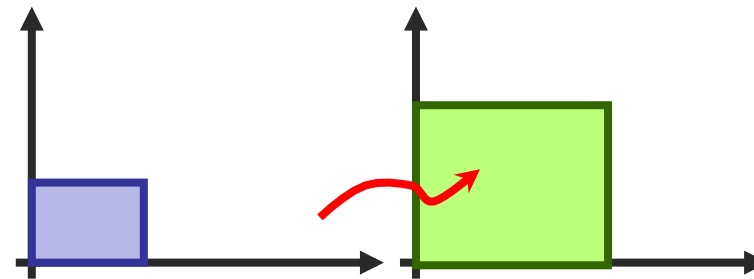


$$\begin{aligned} F(p',q') &= (1-a)(1-b)F(p,q) \\ &+ (1-a)bF(p,q+1) \\ &+ a(1-b)F(p+1,q) \\ &+ abF(p+1,q+1) \end{aligned}$$

# Geometrical Modification

## ■ Scaling

$$\begin{cases} x_k &= s_x u_q \\ y_j &= s_y v_p \end{cases}$$



where  $s_x$  &  $s_y$  are scaling parameters, and  $s_x$  &  $s_y > 0$

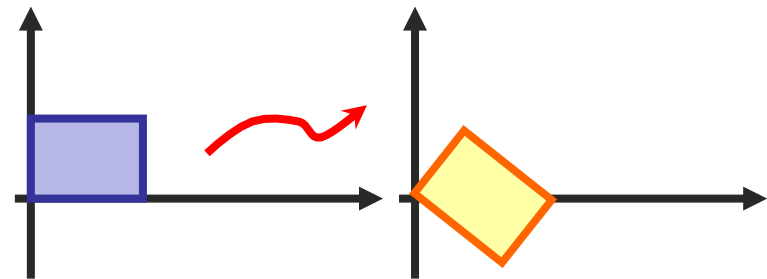
$$\begin{cases} s_x \text{ & } s_x > 1: & \text{magnification} \\ s_x \text{ & } s_x < 1: & \text{minification} \end{cases}$$

$$\Rightarrow \begin{cases} p' &= \frac{1}{s_y} \left( j - J - \frac{1}{2} \right) + P + \frac{1}{2} \\ q' &= \frac{1}{s_x} \left( k - \frac{1}{2} \right) + \frac{1}{2} \end{cases}$$

# Geometrical Modification

## ■ Rotation

$$\begin{cases} x_k &= u_q \cos \theta - v_p \sin \theta \\ y_j &= u_q \sin \theta + v_p \cos \theta \end{cases}$$



Rotate by an angle with respect to the origin of the Cartesian coordinates

**What if the reference point is not the origin of the Cartesian coordinate?**



# Geometrical Modification

## ■ Generalized Linear Geometrical Transformations

➤ translation

$$\begin{bmatrix} x_k \\ y_j \end{bmatrix} = \begin{bmatrix} u_q \\ v_p \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

➤ scaling

$$\begin{bmatrix} x_k \\ y_j \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} u_q \\ v_p \end{bmatrix}$$

➤ rotation

$$\begin{bmatrix} x_k \\ y_j \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} u_q \\ v_p \end{bmatrix}$$

# Geometrical Modification

## ■ Generalized Linear Geometrical Transformations

### ○ Compound operator

$$\begin{bmatrix} u_q \\ v_p \end{bmatrix} \rightarrow \text{translation} \rightarrow \text{scaling} \rightarrow \text{rotation} \rightarrow \begin{bmatrix} x_k \\ y_j \end{bmatrix}$$

$$\begin{bmatrix} x_k \\ y_j \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \left[ \begin{bmatrix} u_q \\ v_p \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \right]$$

**How to convert the above affine system to a linear one?**

# Geometrical Modification

## ■ Generalized Linear Geometrical Transformations

- Expand the system from 2D to 3D

$$T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_k \\ y_j \\ 1 \end{bmatrix} = R(\theta) S(s_x, s_y) T(t_x, t_y) \begin{bmatrix} u_q \\ v_p \\ 1 \end{bmatrix} \xrightarrow{\text{□}} \begin{bmatrix} u_q \\ v_p \\ 1 \end{bmatrix} = T^{-1}(t_x, t_y) S^{-1}(s_x, s_y) R^{-1}(\theta) \begin{bmatrix} x_k \\ y_j \\ 1 \end{bmatrix}$$

# Geometrical Modification

## ■ Exercise

- Write down a **linear system** which represents the following operation:

- Rotate an image by an angle of  $\theta$   
w.r.t. a **pivot point**  $(x_c, y_c)$

$$H = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix}$$



# **Geometrical Modification**

## **Part II**

# Geometrical Modification

## ■ Non-linear Coordinates Transformation and Spatial Warping

### ○ Non-linear address mapping

■ Forward 
$$\begin{cases} x &= X\{u, v\} \\ y &= Y\{u, v\} \end{cases}$$

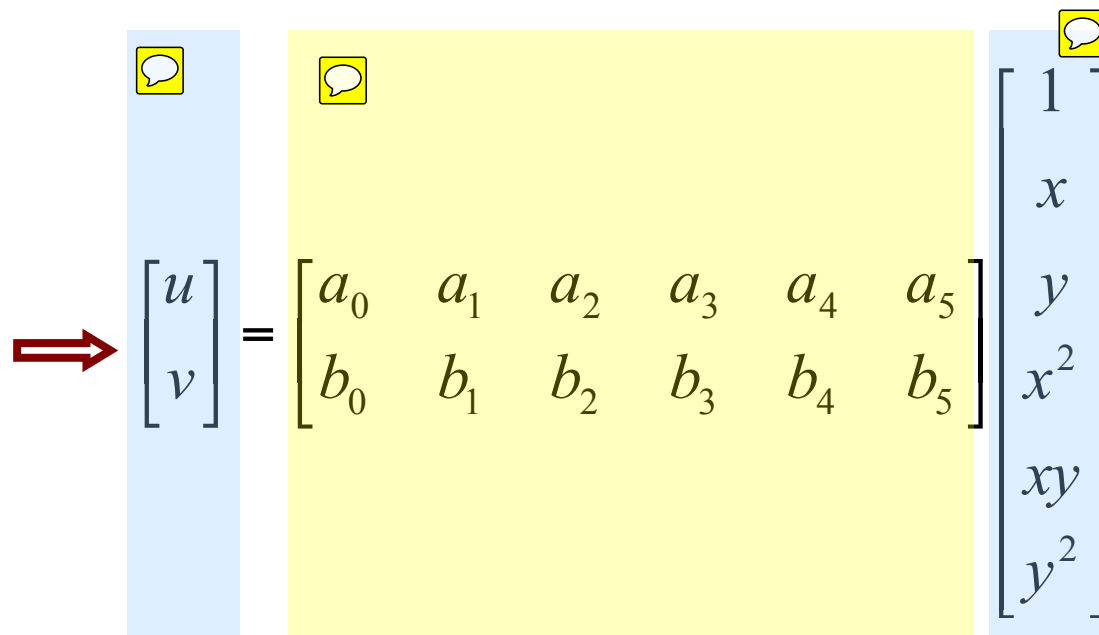
■ Backward (reverse) 
$$\begin{cases} u &= U\{x, y\} \\ v &= V\{x, y\} \end{cases}$$

$$\begin{cases} u_q &= q - \frac{1}{2} \\ v_p &= P + \frac{1}{2} - p \end{cases} \boxed{\text{input}} \Rightarrow \boxed{\text{output}} \begin{cases} x_k &= k - \frac{1}{2} \\ y_j &= J + \frac{1}{2} - j \end{cases}$$

# Geometrical Modification

## ■ Polynomial Warping (2<sup>nd</sup>-order)

$$\begin{cases} u = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 \\ v = b_0 + b_1x + b_2y + b_3x^2 + b_4xy + b_5y^2 \end{cases}$$


$$\Rightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{bmatrix}$$

# [ Geometrical Modification ]

- **Polynomial Warping**

- Rubber-sheet stretching

- Identify spatial distortion

- Calibration → test patterns

- Two steps:

- Based on 'known' input and output pairs (control points), compute the coefficients 'a' and 'b' (either exact or least squares solution)

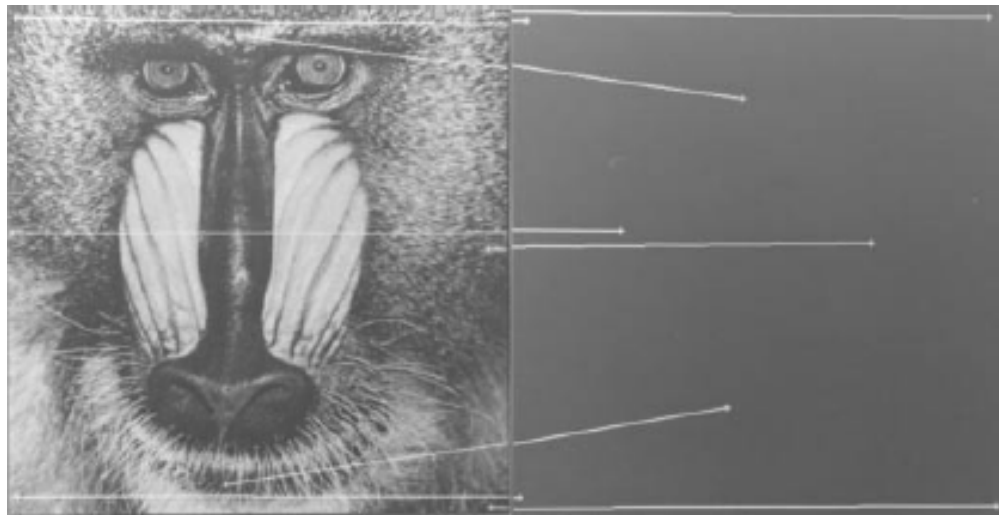
- Use the spatial warping matrix to compute all the output points from their corresponding input points

→ proper interpolation is necessary



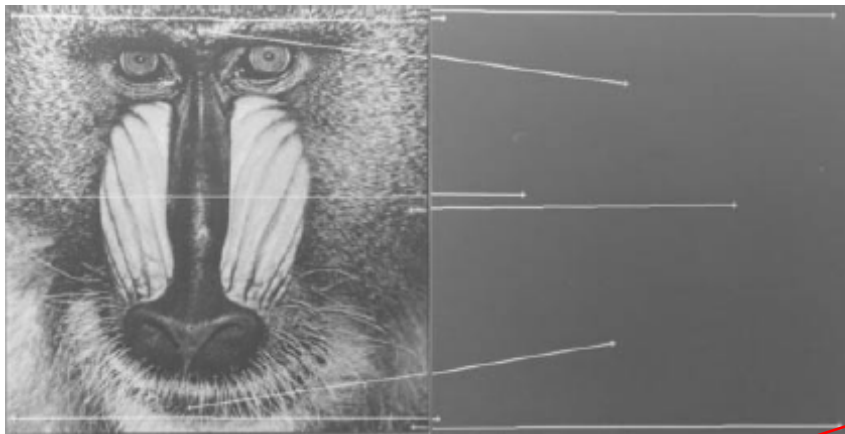
# Geometrical Modification

- Polynomial Warping
  - Example



# Geometrical Modification

## ○ Example



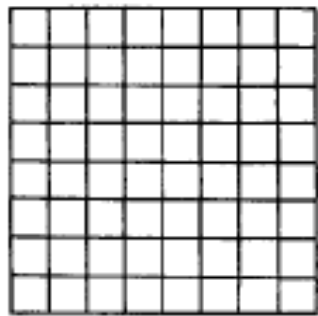
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{bmatrix}$$

$$\begin{bmatrix} u_1 & u_2 & \cdots & u_k \\ v_1 & v_2 & \cdots & v_k \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_k \\ y_1 & y_2 & \cdots & y_k \\ x_1^2 & x_2^2 & \cdots & x_k^2 \\ x_1 y_1 & x_2 y_2 & \cdots & x_k y_k \\ y_1^2 & y_2^2 & \cdots & y_k^2 \end{bmatrix}$$

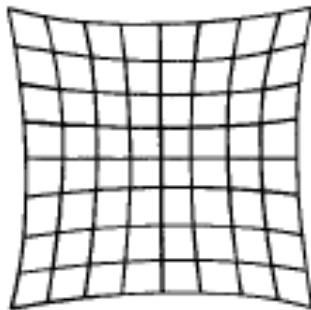
# Geometrical Modification

## ■ Polynomial Warping

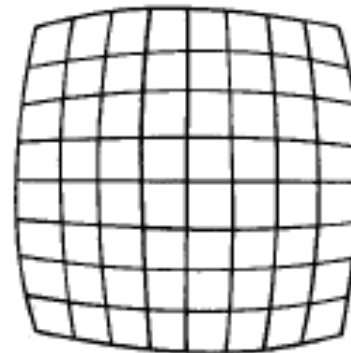
- Useful to compensate the spatial distortion caused by the limitation of a physical imaging system



Original



Pincusion  
distortion



Barrel  
distortion

# [ Geometrical Modification ]

## ■ Examples



Original



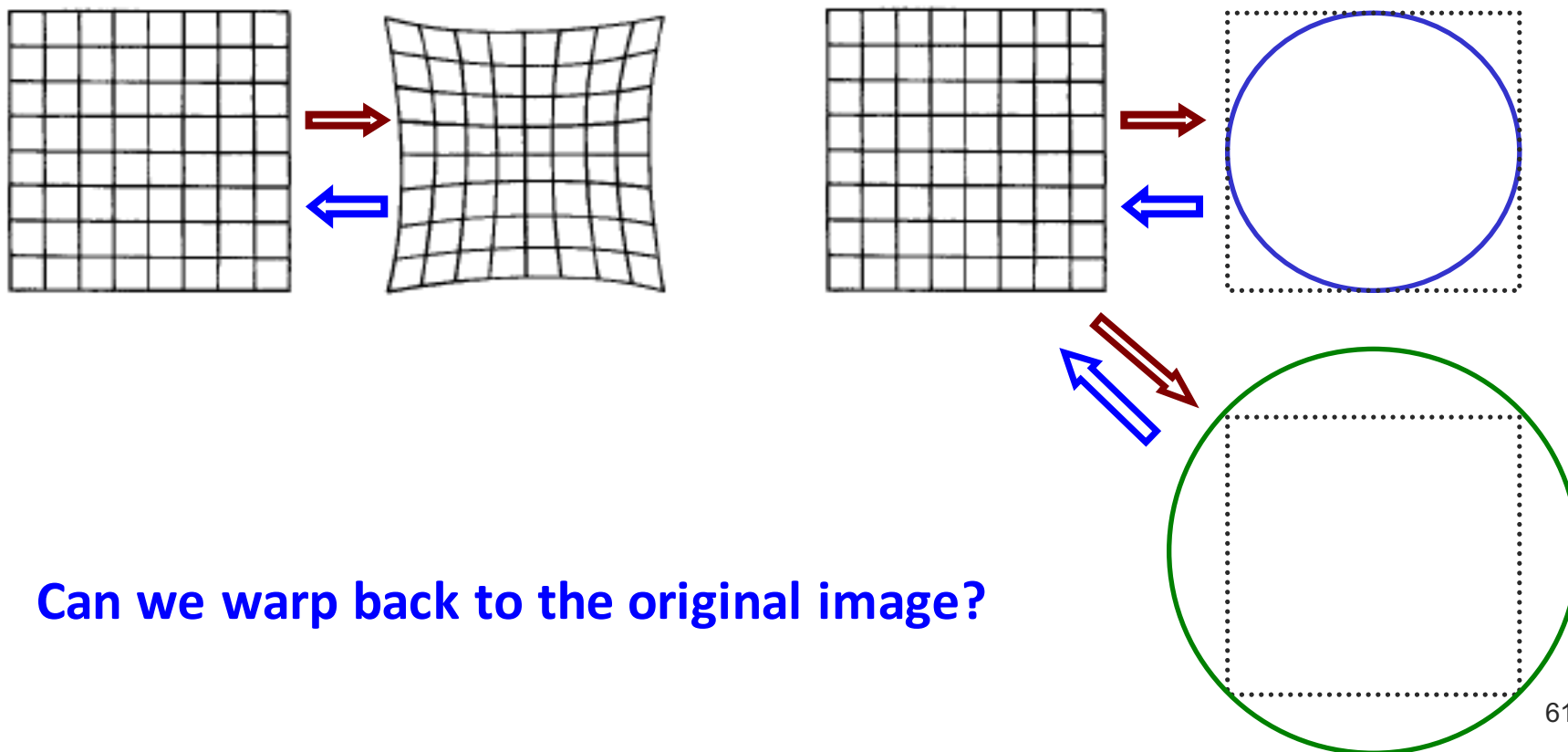
Barrel  
distortion



Pincusion  
distortion

# Geometrical Modification

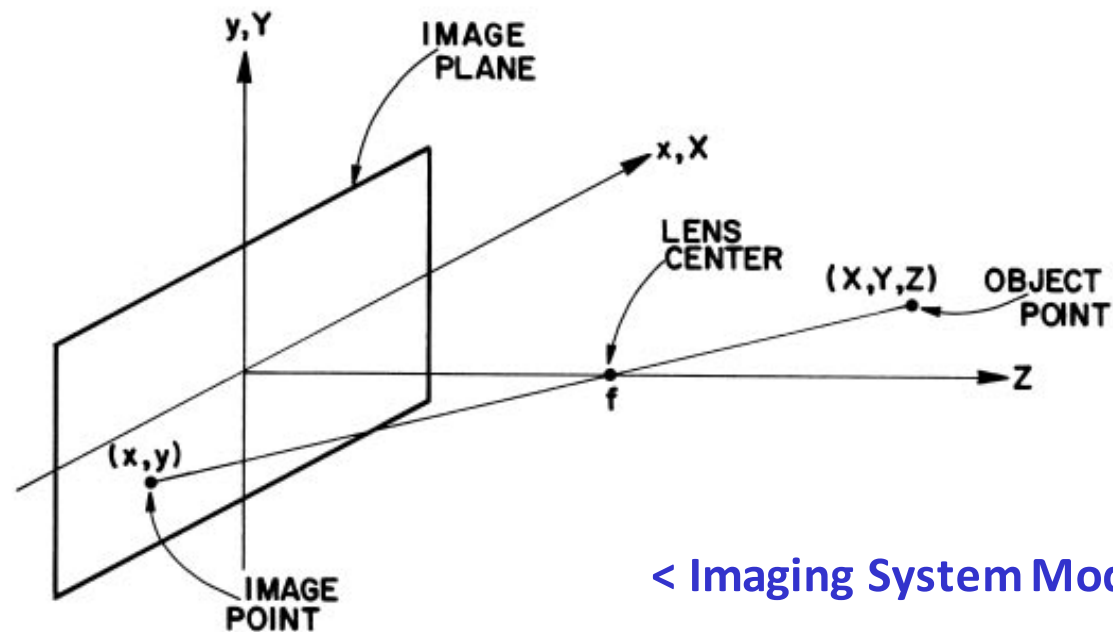
## ■ Example



Can we warp back to the original image?

# Geometrical Modification

- Perspective Transformation
  - Imaging in the 3D space
    - Fundamentals of computer graphics



< Imaging System Model >

# Geometrical Modification

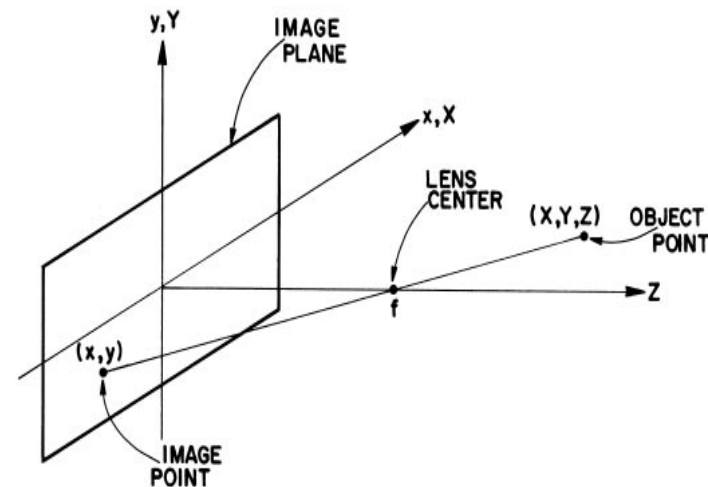
## ■ Perspective Transformation

### ○ Cartesian to image coordinates

#### ■ Similar triangle property

$$\frac{X}{-x} = \frac{Z - f}{f} \Rightarrow x = \frac{fX}{f - Z};$$
$$y = \frac{fY}{f - Z}$$

→ Many-to-one mapping



# Geometrical Modification

## ■ Perspective Transformation

### ○ Image to Cartesian coordinates

#### ■ Need another degree of freedom

$$X = \frac{fx_i}{f + z_i}; \quad Y = \frac{fy_i}{f + z_i}; \quad Z = \frac{fz_i}{f + z_i} \quad z_i \text{ is a free variable}$$

#### ■ Given $Z$ , we may compute $z_i$ and then $X$ & $Y$ via

$$X = \frac{x_i}{f}(f - Z) \quad Y = \frac{y_i}{f}(f - Z)$$



# Geometrical Modification

## ■ Perspective Transformation

$P$  is a perspective transformation matrix,

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix}$$


$$\tilde{v} = s \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \begin{array}{l} \text{homogeneous vector} \\ \text{3D object} \\ \text{s: scaling factor} \end{array}$$

$$\tilde{w} = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \begin{array}{l} \text{homogeneous image} \\ \text{position vector} \end{array}$$

$$\tilde{w} = P\tilde{v} = \begin{bmatrix} sX \\ sY \\ sZ \\ s - sz/f \end{bmatrix} \Rightarrow s = \frac{f}{f - z}$$

# Geometrical Modification

## ■ Camera Imaging Model

- Camera is supported by a gimbal ( $X_G, Y_G, Z_G$ )
- Gimbal can do 3D movements
  - panning ( $\theta$ ) /tilting ( $\phi$ ) 
- Offset between the gimbal support and the image plane center is ( $X_0, Y_0, Z_0$ )
- The complete camera imaging model can be derived by sequentially operating on the homogeneous vector

$$\tilde{w} = PT_c RT_G \tilde{v}$$

