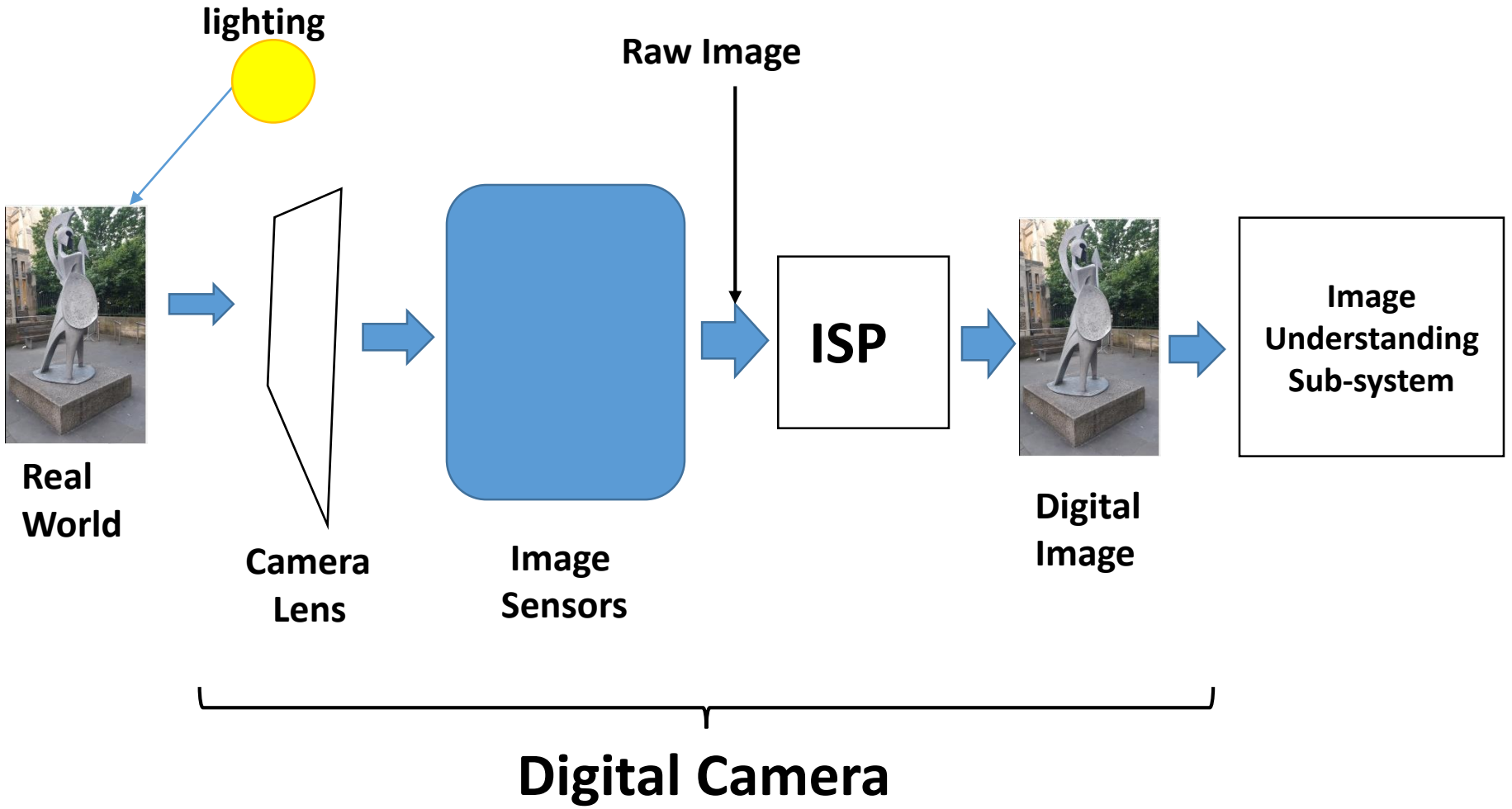
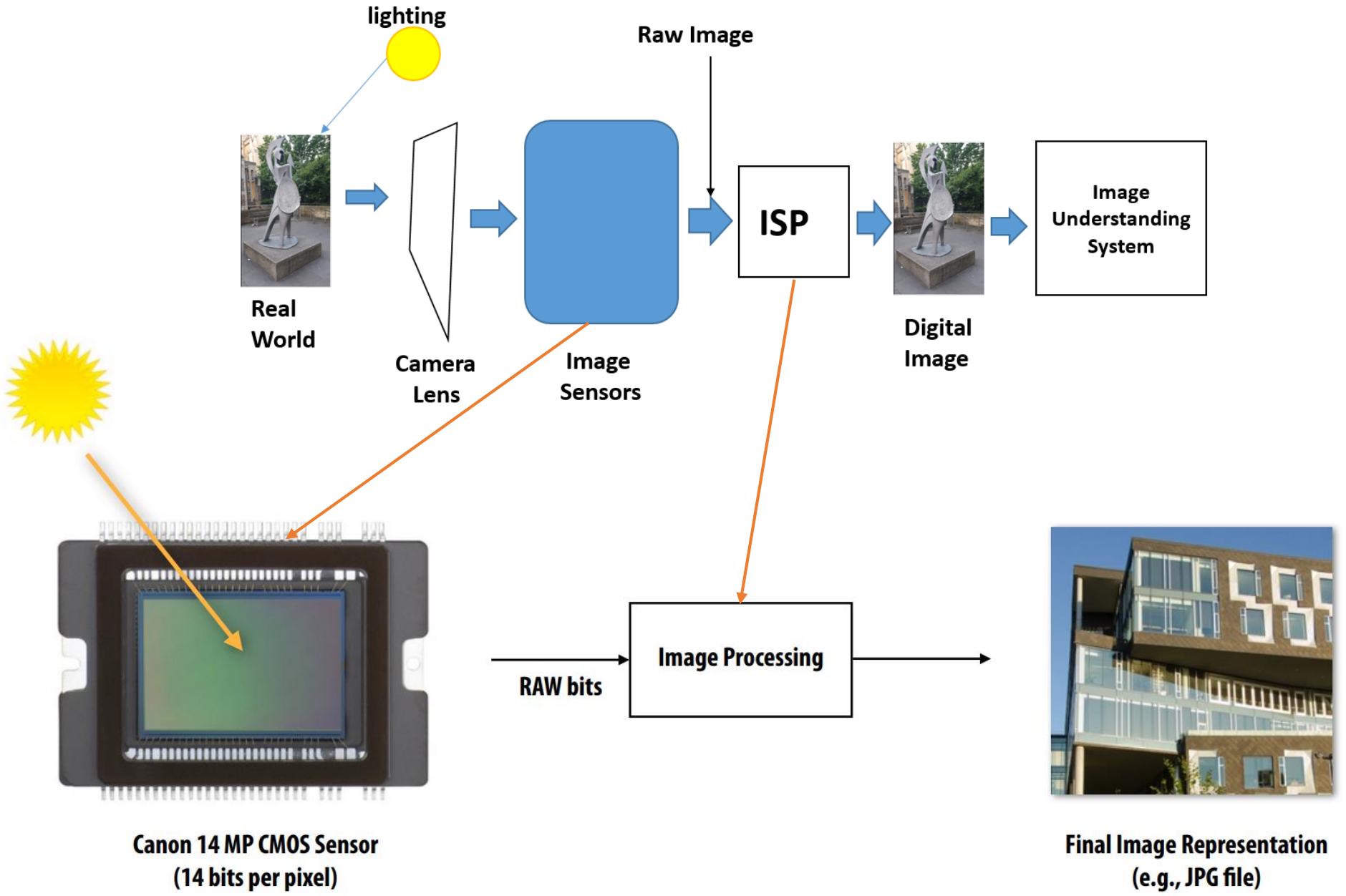
Edge Detection

Jianping Fan
Dept of Computer Science
UNC-Charlotte

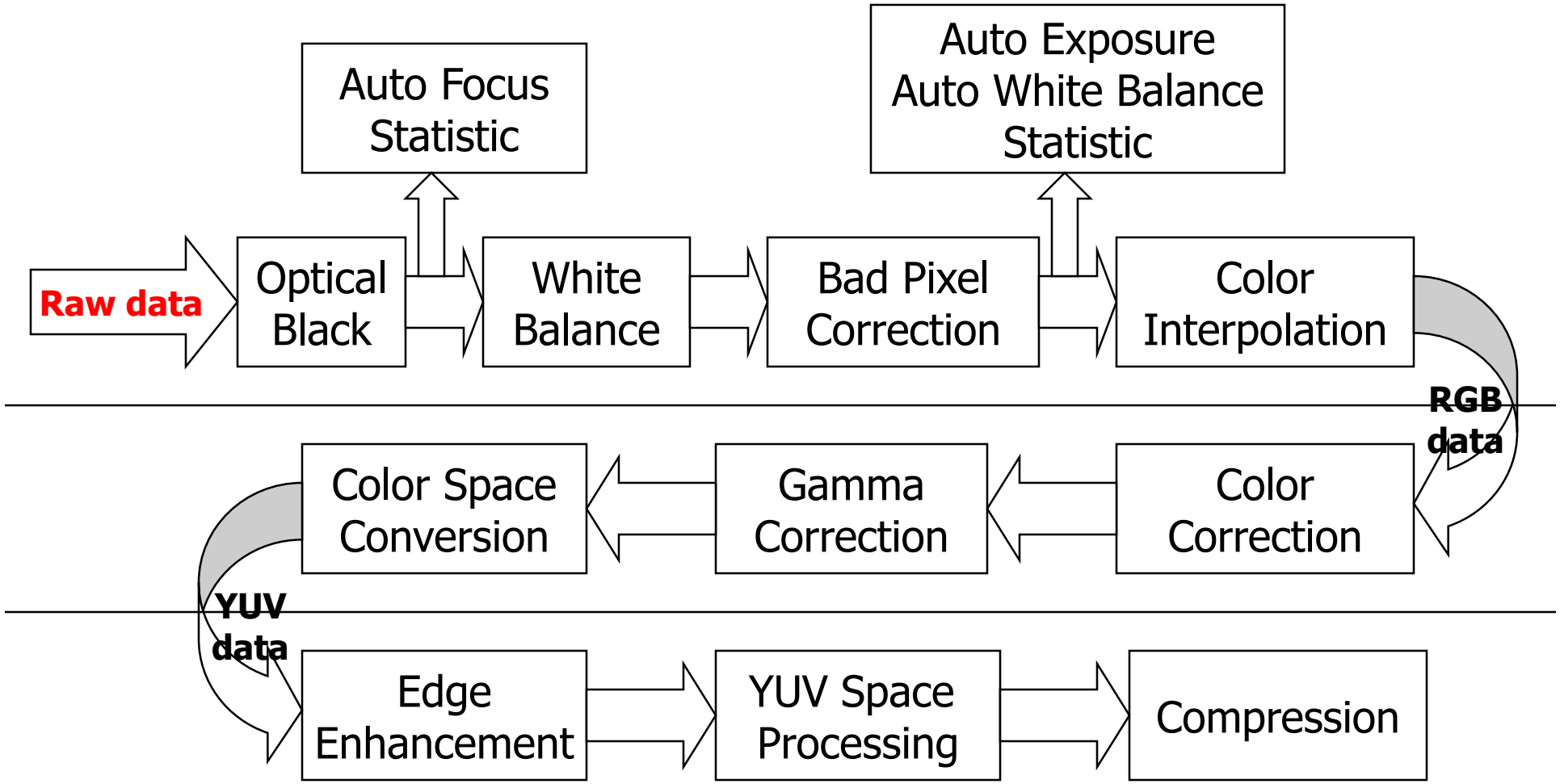
Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Pipeline for Computer Vision Systems

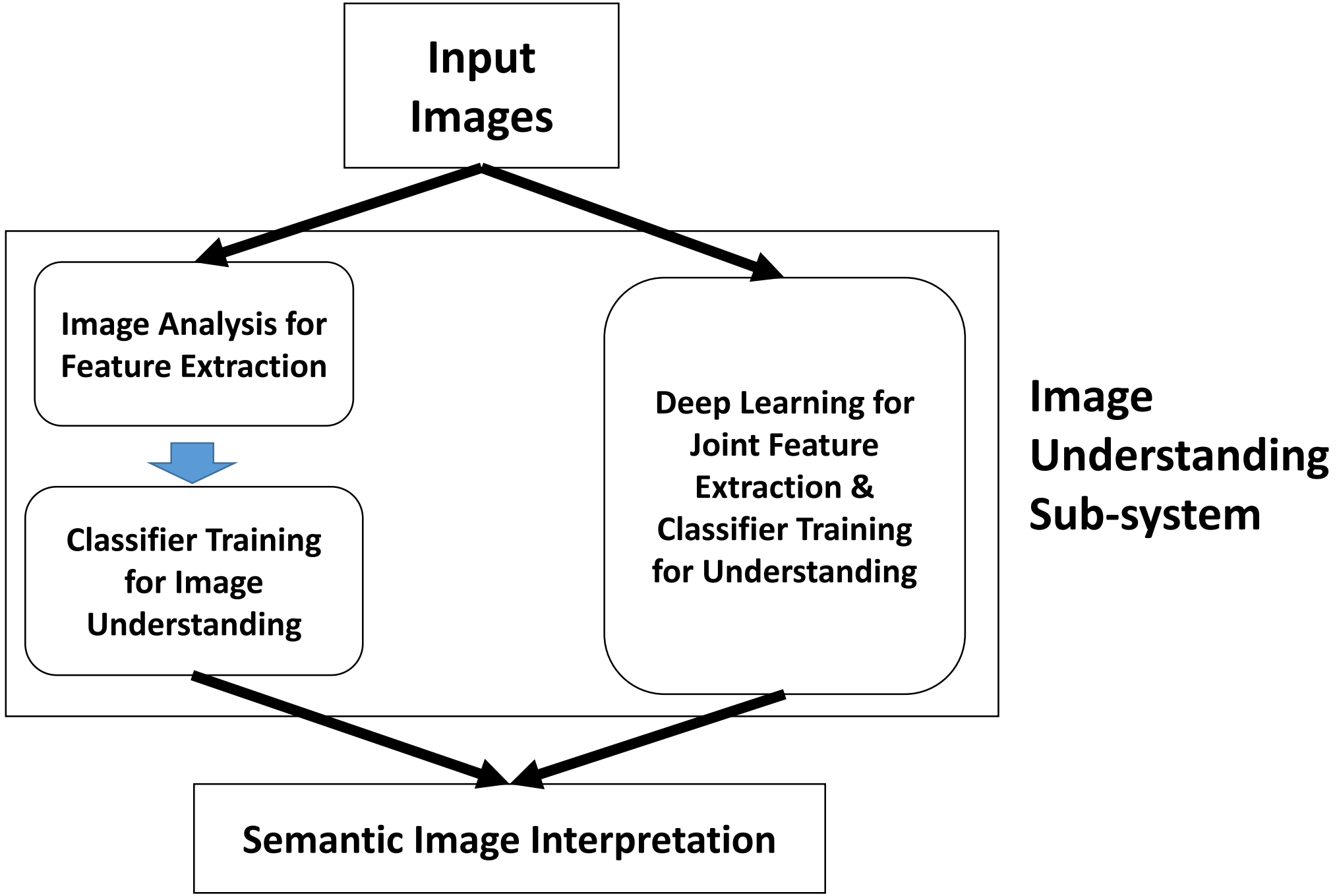



lighting
Raw Image
Real World
Camera Lens
Image Sensors
ISP
Digital Image
Image Understanding System
Canon 14 MP CMOS Sensor
(14 bits per pixel)
RAW bits
Image Processing
Final Image Representation
(e.g., JPG file)

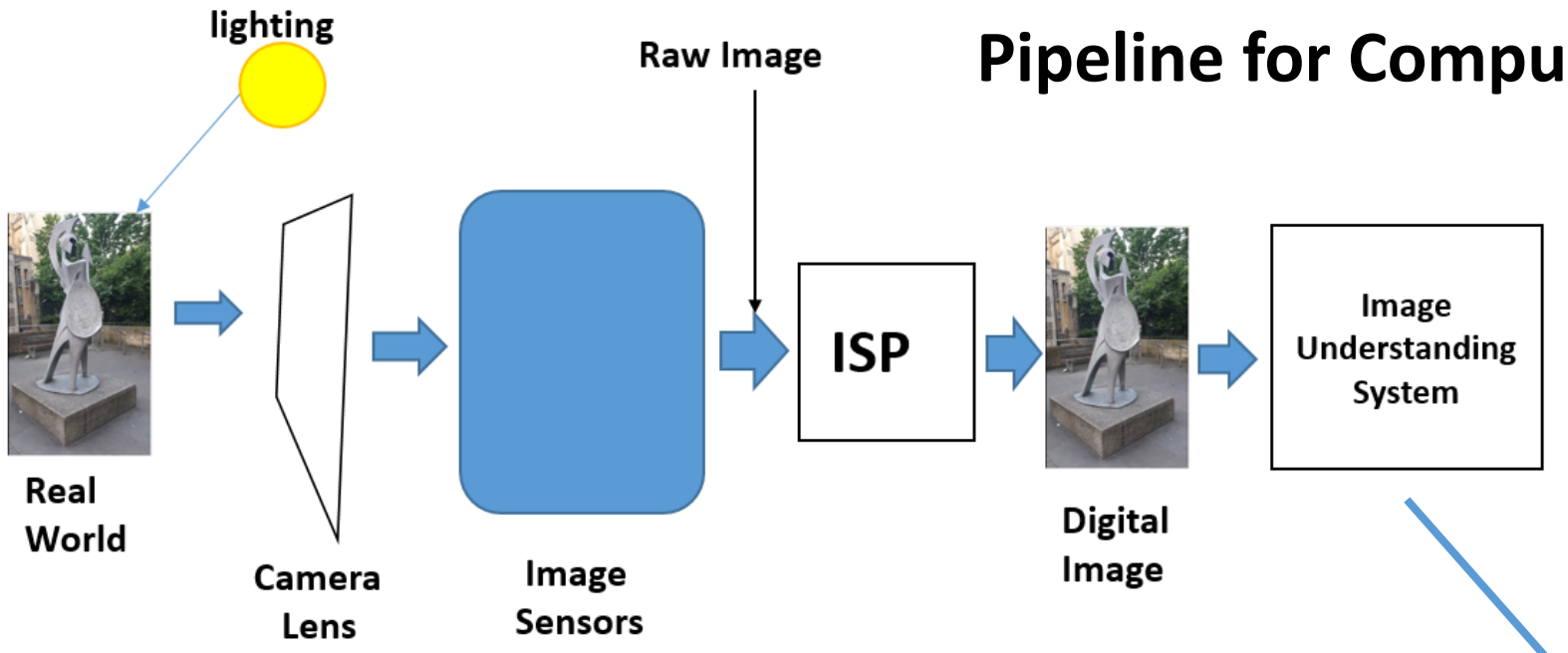
A Typical Image Pipeline for Digital Camera



www.csie.ntu.edu.tw › IntroductionToDigitalCameraSignalProcessor


Input Images
Image Analysis for Feature Extraction
Classifier Training for Image Understanding
Deep Learning for Joint Feature Extraction & Classifier Training for Understanding
Image Understanding Sub-system
Semantic Image Interpretation

Pipeline for Computer Vision Systems



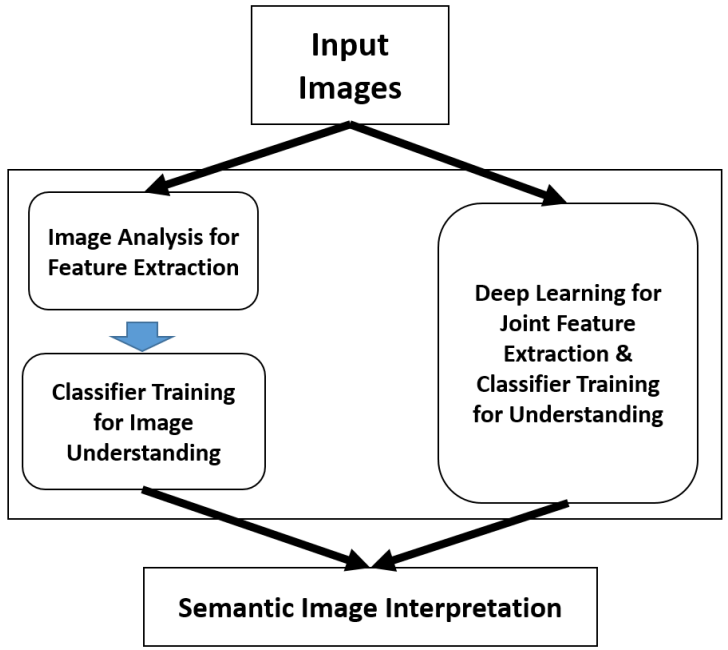


Image Understanding Sub-system

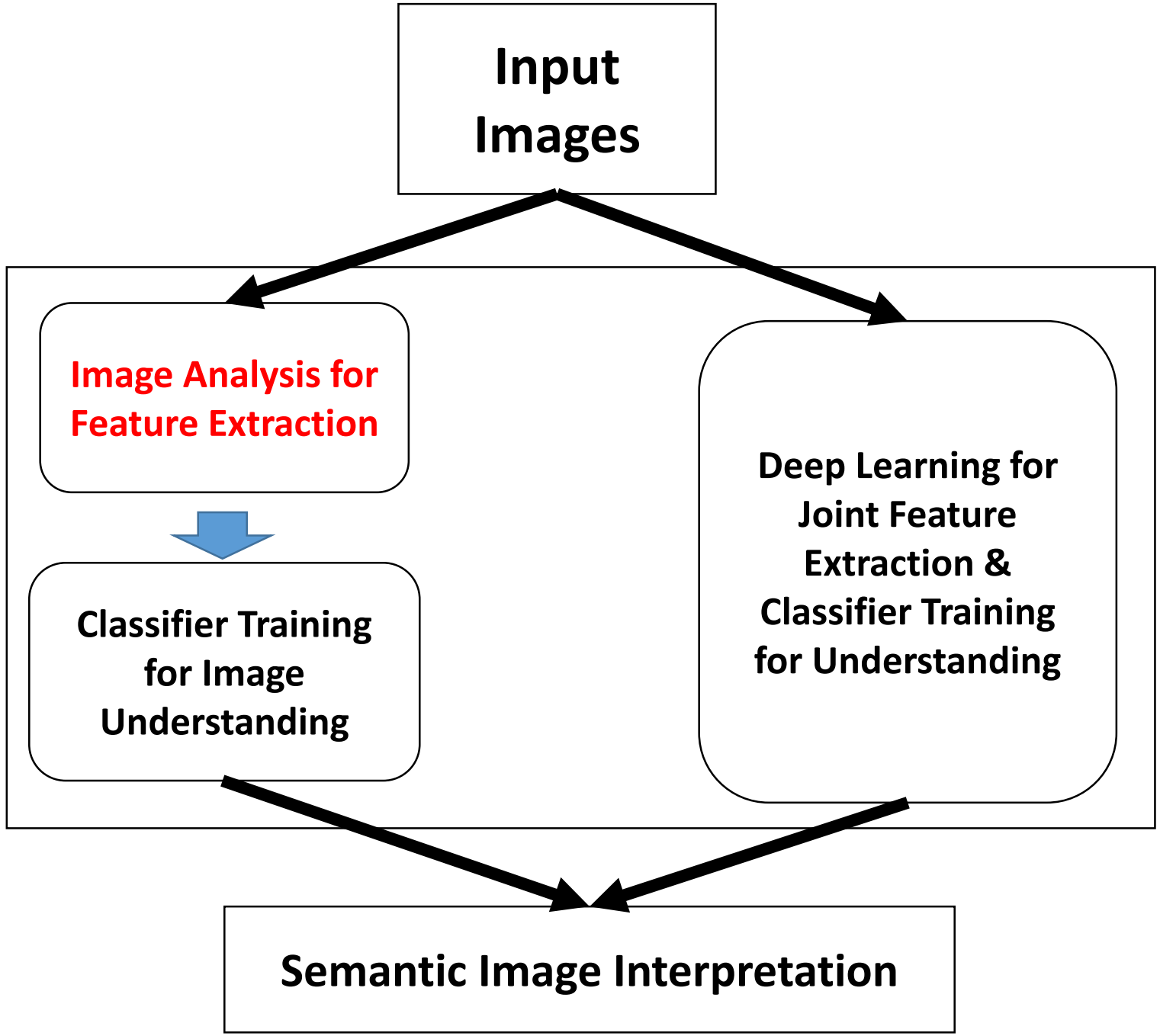


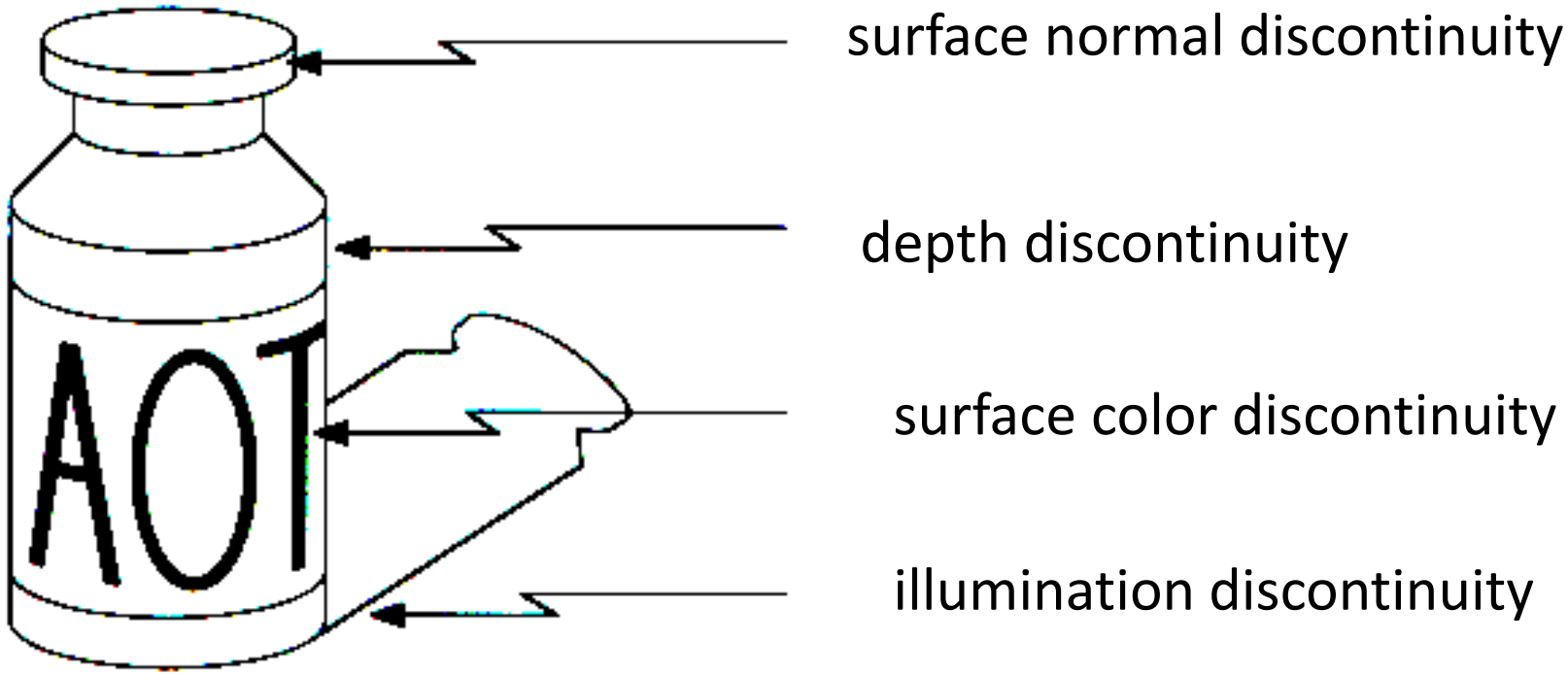
Image Understanding Sub-system

What are key characterizations in an image?

What are key characterizations in an image?

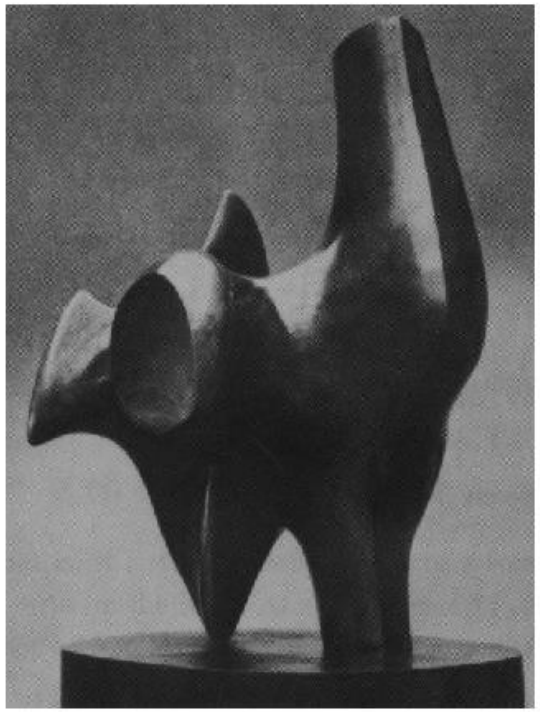
1. Global scene
2. Objects
3. Colors & textures
4. Object boundaries & edge
5. Object relationships
6. ……

What are Edges in an Image?



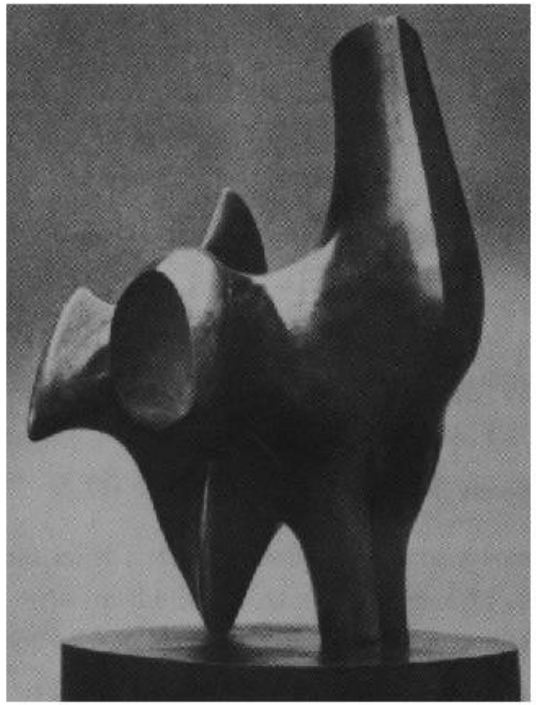
- **Edges are caused by the discontinuities (Differences) of a variety of factors: color, texture, illumination, depth, …….**

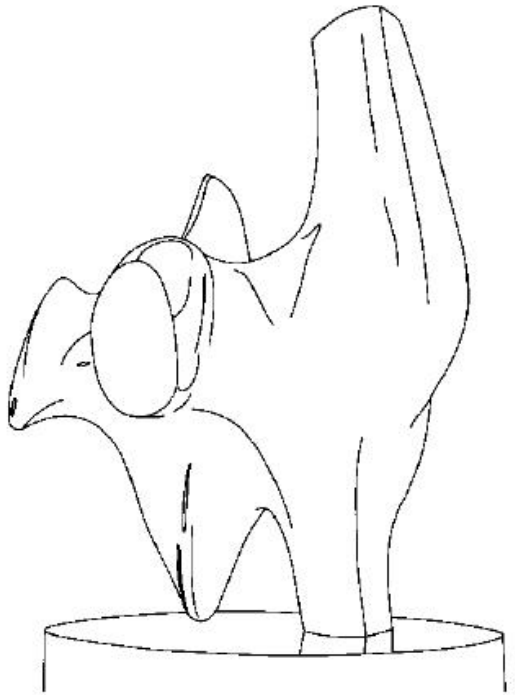
Goal of Edge Detection



- **Convert a 2D image into a set of curves**
  - Extracts salient features of the scene
  - More compact than pixels

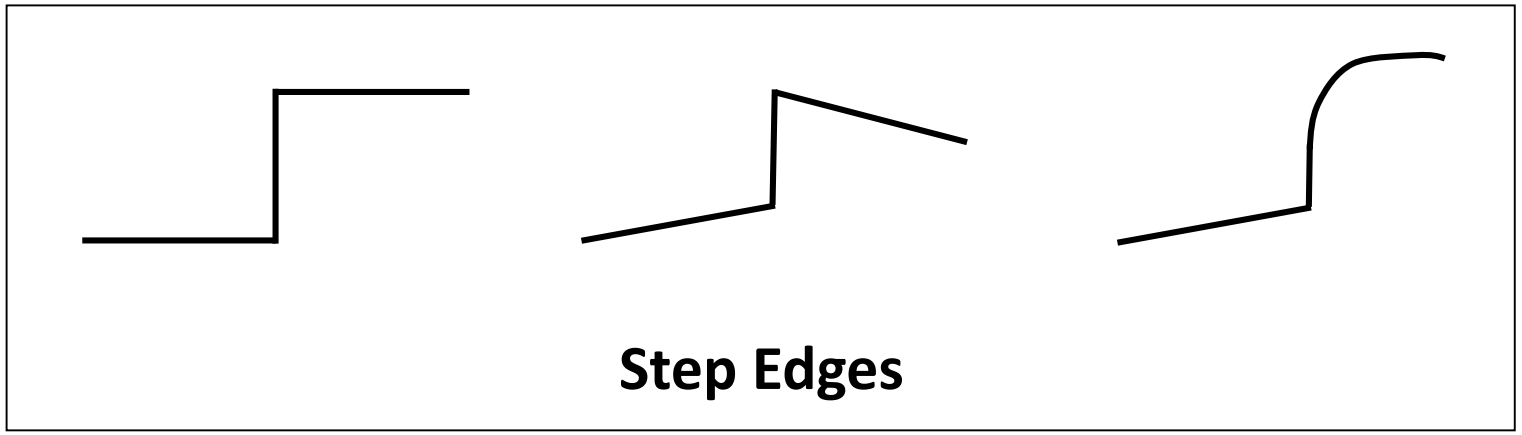
How can you tell that a pixel is on an edge?

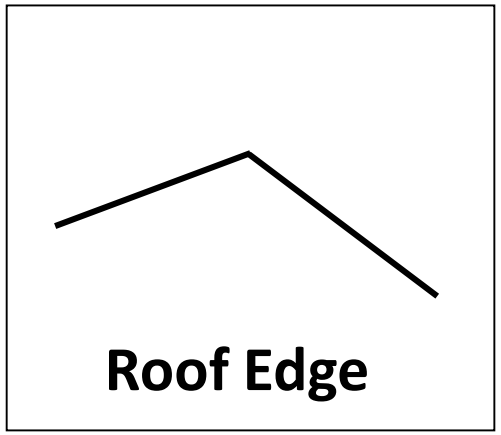


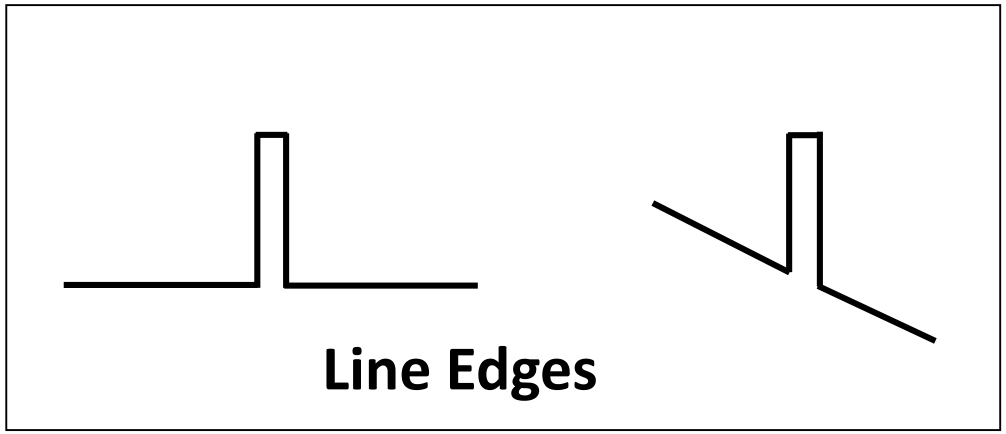


Discontinuity on illuminance, color, texture, depth, surface normal, ……

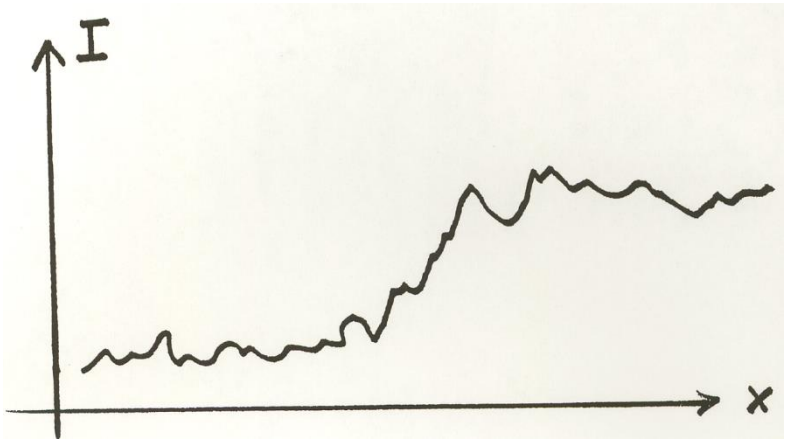
Edge Types







Real Edges



Noisy and Discrete!

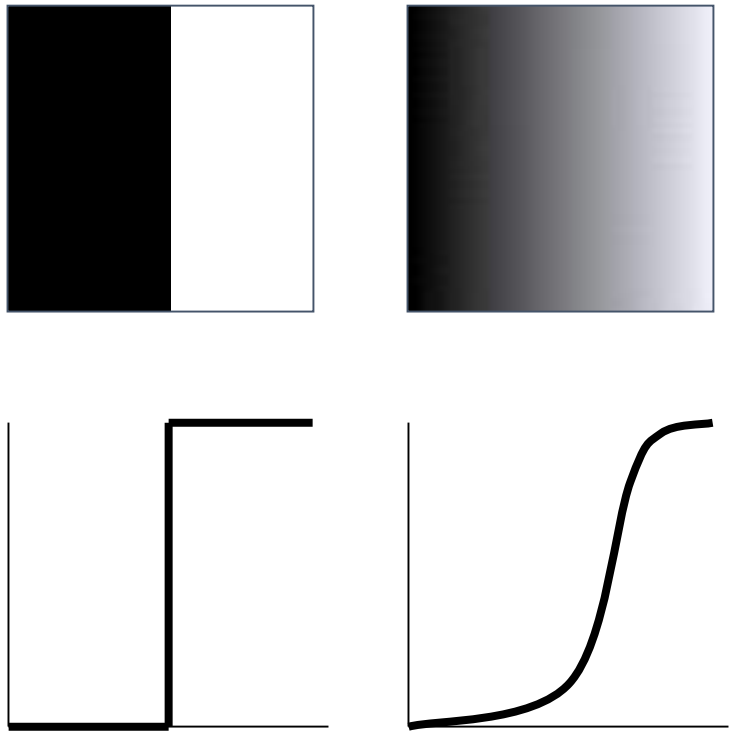
We want an **Edge Operator** that produces:

- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

Noise reduction for edge operator

What are edges in an image?

- Edges are those places in an image that correspond to **object boundaries**.
- Edges are pixels where image **brightness changes** abruptly.
- Edges are pixels where image **depth** changes.



Brightness vs. Spatial Coordinates

Discontinuities (Differences) on brightness, depth, color, surface normal …..

More About Edges: *weighted differences among neighboring pixels*

- An edge is a property attached to an individual pixel and is calculated from the image function behavior in a **neighborhood** of the pixel.

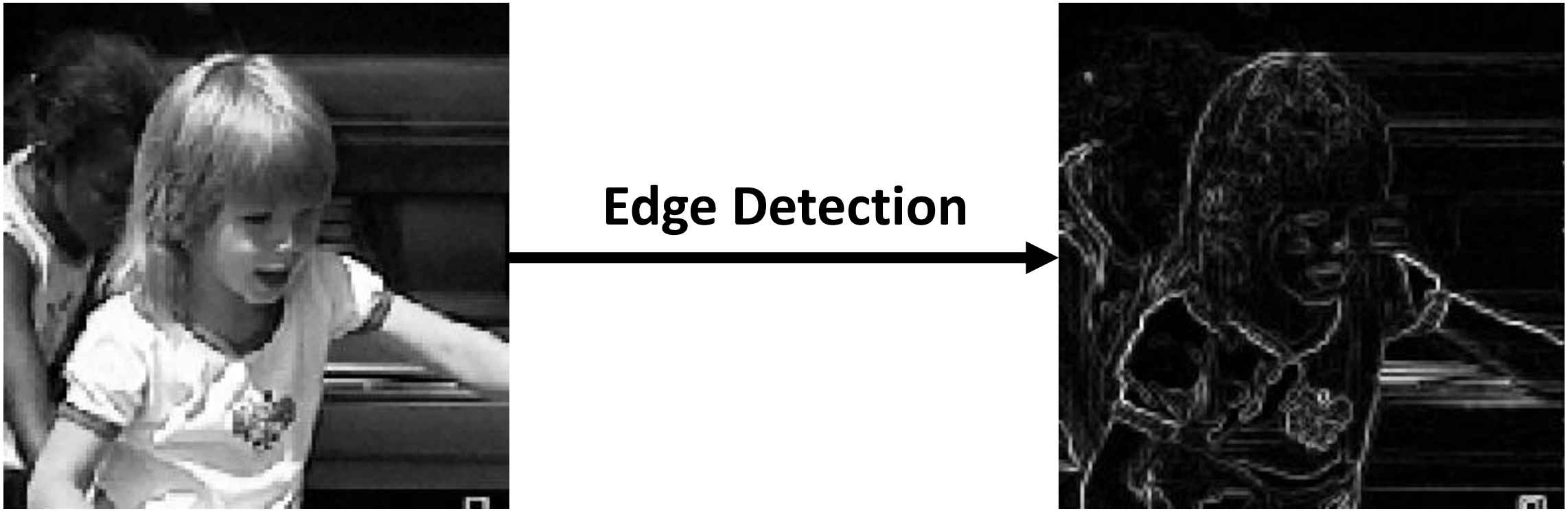
- **Weights** in a neighborhood pattern.

- It is a **vector variable** (**magnitude** of the gradient, **direction** of an edge**)** .

Four Issues for Edge Detection

- Neighborhood Patterns for Difference (Discontinuity) Calculation;
- Weights for Difference (Discontinuity) Calculation;
- Thresholds for Decision Making (edge ***vs.*** non-edge);
- Misleading Effects from Noise (Noise also causes discontinuities/differences)

From Image To Edge Map



1. Discontinuities (Differences)

2. Significance of Discontinuities (Differences)

Edge Detection Methods

- Many are implemented with convolution mask and based on discrete approximations to **differential operators**.
- Differential operations measure the rate of change in the image **brightness** function.
- Some operators return **orientation** information. Other only return information about the existence of an edge at each point.

Edge Detection Operators

- **Neighborhood Patterns for Difference (Discontinuity) Calculation;**
- **Weights on Neighborhood Patterns for Difference (Discontinuity) Calculation**

Roberts Operator

- Mark edge point only
- No information about edge orientation
- Work best with binary images
- Primary disadvantage:
  - High sensitivity to noise
  - Few pixels are used to approximate the gradient

Roberts Operator (Cont.)

(r-1, c-1)	(r, c-1)
(r-1,c)	**(r, c)**

- First form of Roberts Operator

$$\sqrt{\left[I(r,c)-I(r-1,c-1)\right]^2+\left[I(r,c-1)-I(r-1,c)\right]^2}$$

- Second form of Roberts Operator

$$|I(r,c)-I(r-1,c-1)|+|I(r,c-1)-I(r-1,c)|$$

$$h_1=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad h_2=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$$

Sobel Operator

(r-1, c-1)	(r, c-1)	(r+1, c-1)
(r-1, c)	**(r, c)**	(r+1, c)
(r-1, c+1)	(r, c+1)	(r+1, c+1)

- Looks for edges in both **horizontal** and **vertical** directions, then combine the information into a single metric.
- The masks are as follows:

$$y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Edge Magnitude = $\sqrt{x^2 + y^2}$ Edge Direction = $\tan^{-1}\left[\frac{y}{x}\right]$

Prewitt Operator

- Similar to the Sobel, with different mask coefficients:

$$y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

(r-1, c-1)	(r, c-1)	(r+1, c-1)
(r-1, c)	**(r, c)**	(r+1, c)
(r-1, c+1)	(r, c+1)	(r+1, c+1)

Edge Magnitude = $\sqrt{x^2 + y^2}$

Edge Direction = $\tan^{-1}\left[\frac{y}{x}\right]$

Kirsch Compass Masks

- Taking a single mask and rotating it to **8** major compass **orientations**: N, NW, W, SW, S, SE, E, and NE.

- The edge magnitude = The **maximum value** found by the convolution of each mask with the image.

$$S(x, y) = \max \{ N(x,y), NW(x,y), W(x,y), SW(x,y), S(x,y), SE(x,y), E(x,y), NE(x,y) \}$$

- The **edge direction** is defined by the mask that produces the maximum magnitude.

Kirsch Compass Masks (Cont.)

(r-1, c-1)	(r, c-1)	(r+1, c-1)
(r-1, c)	**(r, c)**	(r+1, c)
(r-1, c+1)	(r, c+1)	(r+1, c+1)

- The Kirsch masks are defined as follows:

$$N=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & -3 & 5\end{bmatrix} \quad W=\begin{bmatrix}-3 & 5 & 5\\ -3 & 0 & 5\\ -3 & -3 & -3\end{bmatrix} \quad S=\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix} \quad E=\begin{bmatrix}5 & 5 & -3\\ 5 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}$$

$$NW=\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix} \quad SW=\begin{bmatrix}-3 & -3 & -3\\ 5 & 0 & -3\\ 5 & 5 & -3\end{bmatrix} \quad SE=\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & -3\\ 5 & 5 & 5\end{bmatrix} \quad NE=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & 5 & 5\end{bmatrix}$$

- EX: If NE produces the maximum value, then the edge direction is Northeast

Robinson Compass Masks

(r-1, c-1)	(r, c-1)	(r+1, c-1)
(r-1, c)	**(r, c)**	(r+1, c)
(r-1, c+1)	(r, c+1)	(r+1, c+1)

- Similar to the Kirsch masks, with mask coefficients of 0, 1, and 2:

$$N = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad W = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad S = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad E = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$NW = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad SW = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \quad SE = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad NE = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Laplacian Operators

- **Edge magnitude** is approximated in digital images by a **convolution sum**.

$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$$

- The **sign of the result** (+ or -) from two adjacent pixels provide edge orientation and tells us which side of edge brighter

(r-1, c-1)	(r, c-1)	(r+1, c-1)
(r-1, c)	**(r, c)**	(r+1, c)
(r-1, c+1)	(r, c+1)	(r+1, c+1)

g(m)

Laplacian Operators (Cont.)

- Masks for 4 and 8 neighborhoods

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

(r-1, c-1)	(r, c-1)	(r+1, c-1)
(r-1, c)	**(r, c)**	(r+1, c)
(r-1, c+1)	(r, c+1)	(r+1, c+1)

g(m)

- Mask with stressed significance of the central pixel or its neighborhood

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$$

Edge Map In Matlab Program

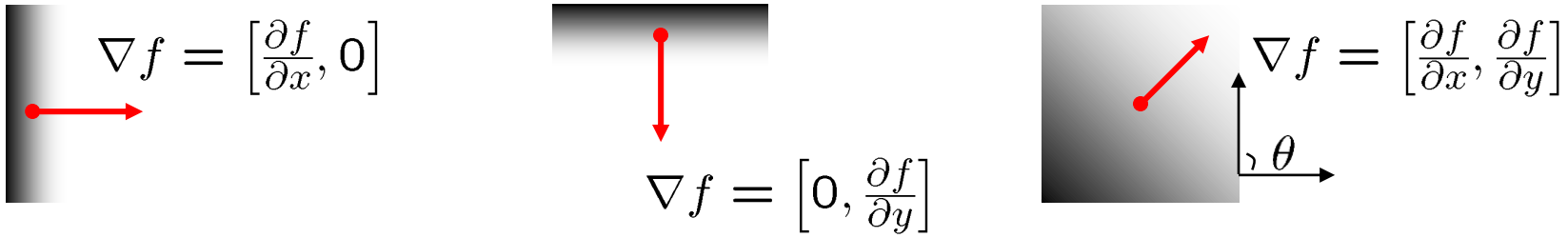
- Implement all methods in this presentation
- Set up edge detection mask(s)
- Use convolution method (filter2 function)
- Calculate edge magnitude
- Show the result of edge map
- No calculation of edge direction

Edge Detection Operators

- **Gradient as Difference (Discontinuity)**

Image Gradient

- Gradient equation: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Represents direction of most rapid change in intensity



- Gradient direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Theory of Edge Detection

- **Image intensity (brightness):**

$$I(x,y) = B_1 + (B_2 - B_1)u(x\sin\theta - y\cos\theta + \rho)$$

- Partial derivatives (**gradients**):

$$\frac{\partial I}{\partial x} = +\sin\theta(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

$$\frac{\partial I}{\partial y} = -\cos\theta(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

- **Squared gradient**:

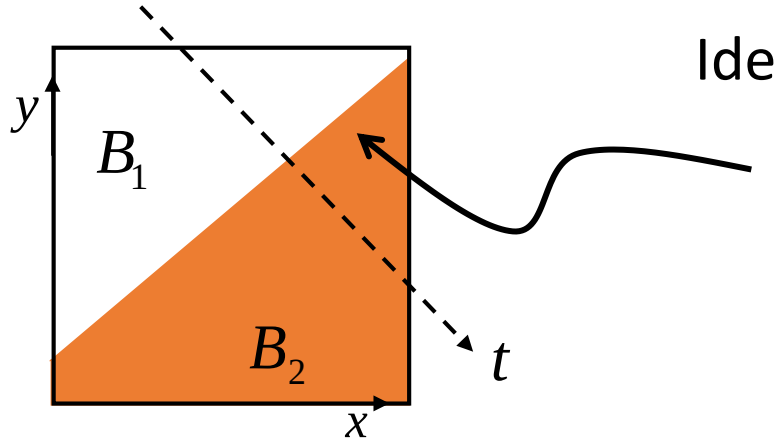
$$s(x,y) = \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2 = [(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)]^2$$

Edge Magnitude: $\sqrt{s(x,y)}$

Edge Orientation: $\arctan\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$ (normal of the edge)

Rotationally symmetric, non-linear operator

Theory of Edge Detection



Ideal edge

$$L(x,y) = x\sin\theta - y\cos\theta + \rho = 0$$

$$B_1 : L(x,y) < 0$$

$$B_2 : L(x,y) > 0$$

Unit step function:

$$u(t) = \begin{cases} 1 & \text{for } t > 0 \\ 1/2 & \text{for } t = 0 \\ 0 & \text{for } t < 0 \end{cases} \qquad u(t) = \int_{-\infty}^{t} \delta(s)\,ds$$

Image intensity (brightness):

$$I(x,y) = B_1 + (B_2 - B_1)\,u(x\sin\theta - y\cos\theta + \rho)$$

Theory of Edge Detection

- Image intensity (brightness):

$$I(x,y) = B_1 + (B_2 - B_1)u(x\sin\theta - y\cos\theta + \rho)$$

- Partial derivatives (gradients):

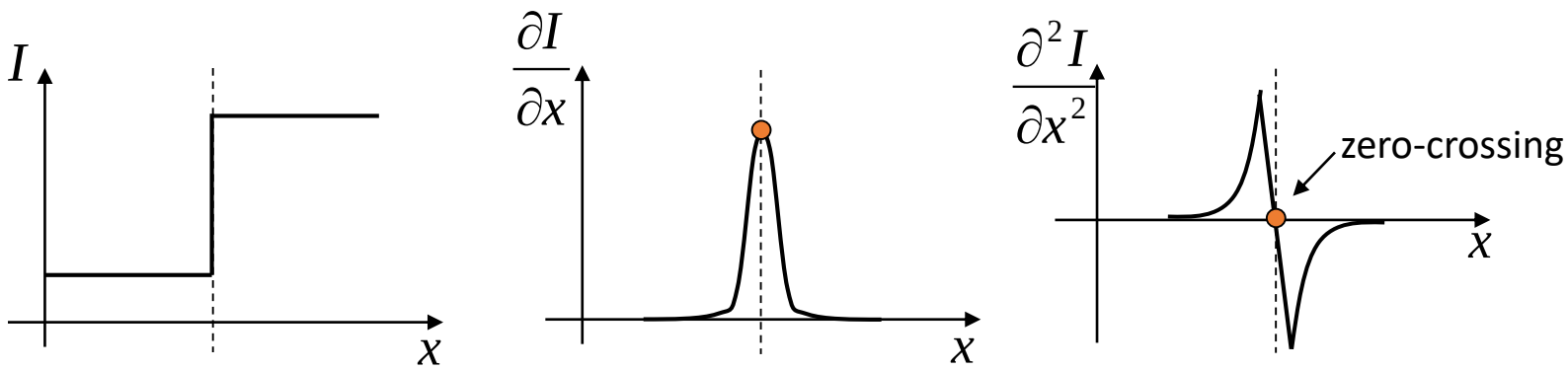
$$\frac{\partial I}{\partial x} = +\sin\theta(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

$$\frac{\partial I}{\partial y} = -\cos\theta(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

- **Laplacian**:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = (B_2 - B_1)\delta'(x\sin\theta - y\cos\theta + \rho)$$

Rotationally symmetric, linear operator



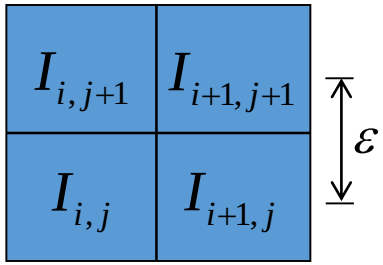
Discrete Edge Operators

- How can we differentiate a ***discrete*** image?

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$



Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

Discrete Edge Operators

- Second order partial derivatives:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

- **Laplacian** :

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Convolution masks :

$$\nabla^2 I \approx \frac{1}{\varepsilon^2}\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array} \quad \text{or} \quad \frac{1}{6\varepsilon^2}\begin{array}{|c|c|c|}\hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline\end{array} \quad \text{(more accurate)}$$

The Sobel Operators

- Better approximations of the gradients exist
  - The *Sobel* operators below are commonly used

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad s_x$$

$$\frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad s_y$$

Comparing Edge Operators

Gradient: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

-1	0	1
-1	0	1
-1	0	1

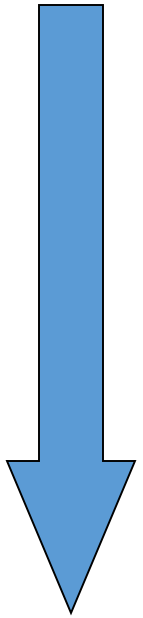
1	1	1
0	0	0
-1	-1	1

Sobel (5 x 5):

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Good Localization
Noise Sensitive
Poor Detection



Poor Localization
Less Noise Sensitive
Good Detection

Edge Detection Operators

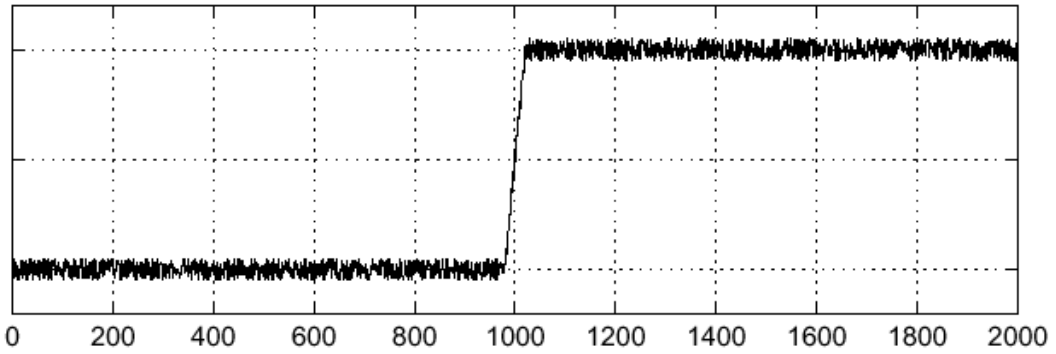
- **Noise's Effects on Difference (Discontinuity) Calculation**
- **Filter can first be performed for noise reduction**

Spatial domain *vs.* Frequency domain

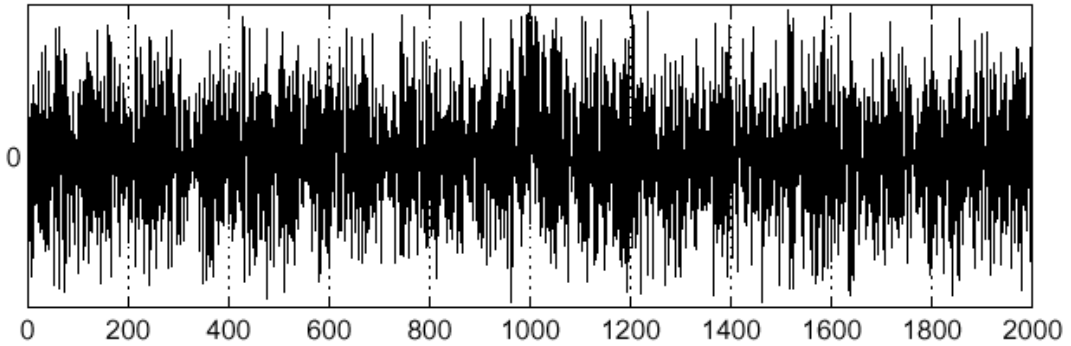
Effects of Noise

- 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??

Noise and Edge Detection

- Noise is a bad thing for edge-detection
  - Usually assume that noise is white Gaussian noise (not likely in reality!)
- Introduces many spurious edges
- Low-pass filtering is a simple way of reducing the noise
  - For the Laplacian of Gaussian Method, it is integrated into the edge detection

Why does filtering with a Gaussian reduce noise?

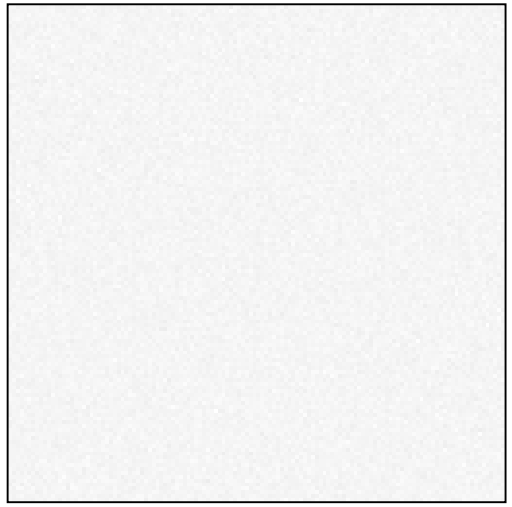
- Forsyth and Ponce's explanation:

"Secondly, pixels will have a greater tendency to look like neighboring pixels — if we take stationary additive Gaussian noise, and smooth it, the pixel values of the resulting signal are no longer independent. In some sense, this is what smoothing was about — recall we introduced smoothing as a method to predict a pixel's value from the values of its neighbors. However, if pixels tend to look like their neighbors, then derivatives must be smaller (because they measure the tendency of pixels to look different from their neighbors)."
– from Computer Vision – A Modern Approach

How I like to see it

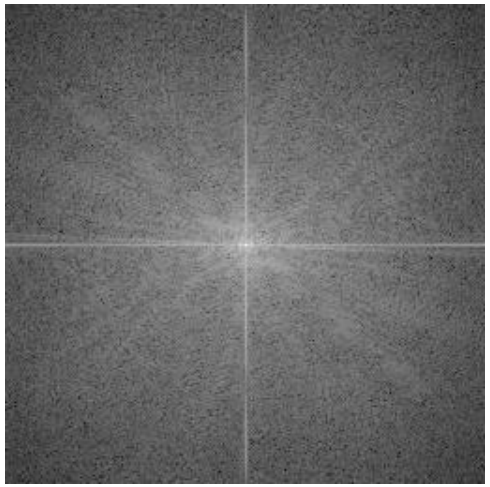
- If F is the DFT of an image, then $|F|^2$ is the power at each frequency

```
for i=1:10000
cimg=randn(128);
cum=cum+abs(fft2(cimg)).^2;
end
```



- In the average power image, the power is equal at every frequency

Remember: Energy in Images is concentrated in low-frequencies

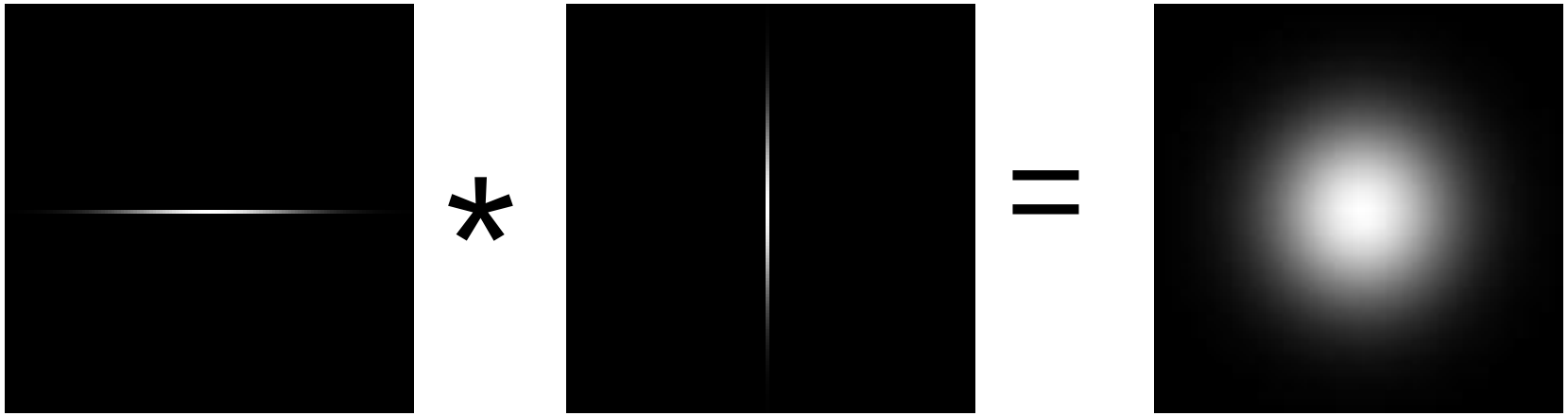


Log-Spectrum of the Einstein Image

- When you low-pass filter, you're retaining the relatively important low-frequencies and eliminating the noise in the high frequencies, where you don't expect much image energy

Smoothing Fast

- The Gaussian Filter is a separable filter

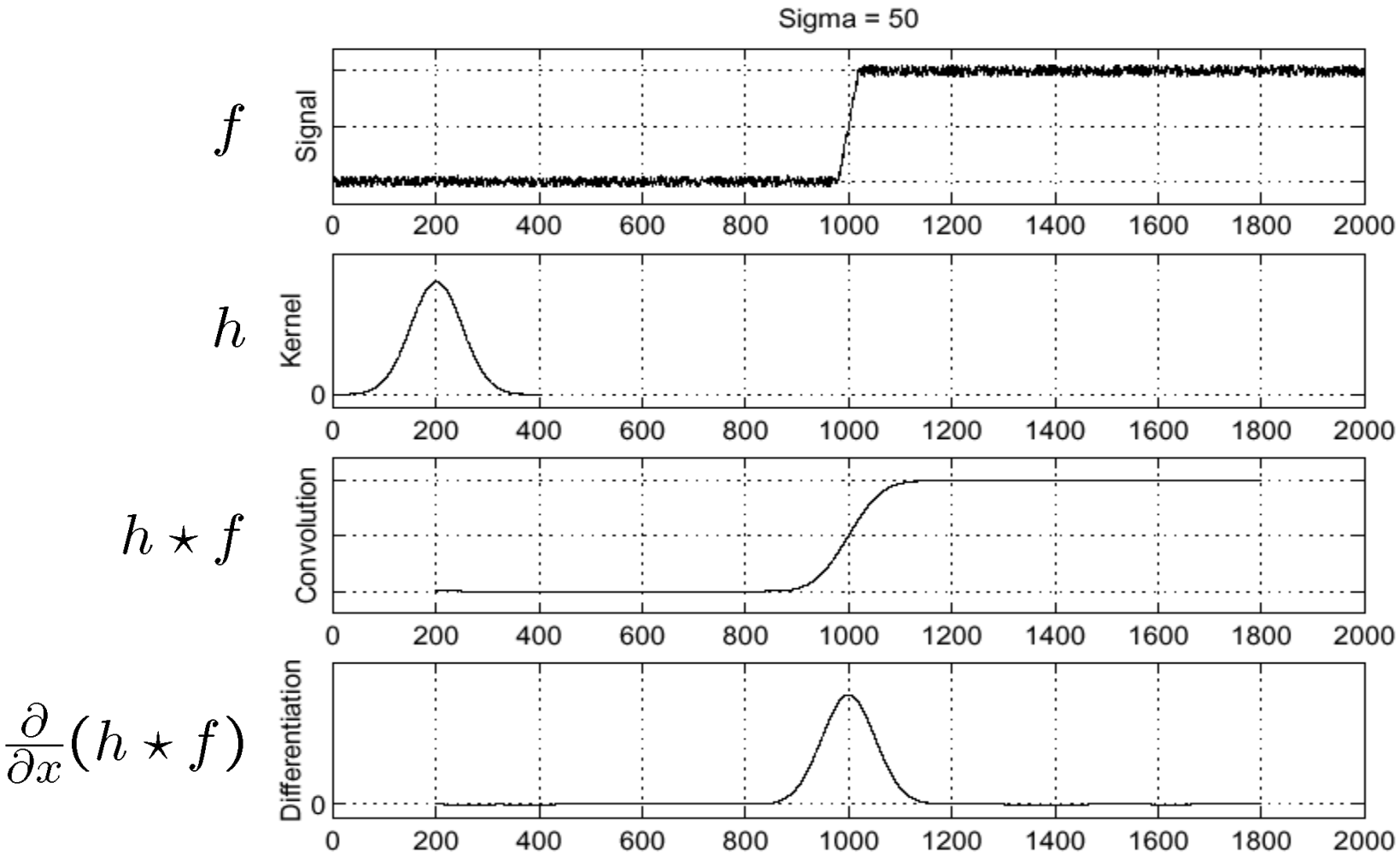


- The 2D filter can be expressed as a series of 2 1D convolutions
- For a 9x9 filter, a separable filter requires 18 computations versus 81 for the 2D filter

Further Improvements to the Gradient Edge-Detector

- Fundamental Steps in Gradient-Based Edge Detection that we have talked about:
  - Smooth to reduce noise
  - Compute Gradients
  - Do Non-maximal suppression
- There is one more heuristic that we can advantage of.
- Edges tend to be continuous

Solution: Smooth First



Where is the edge?

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

Derivative Theorem of Convolution

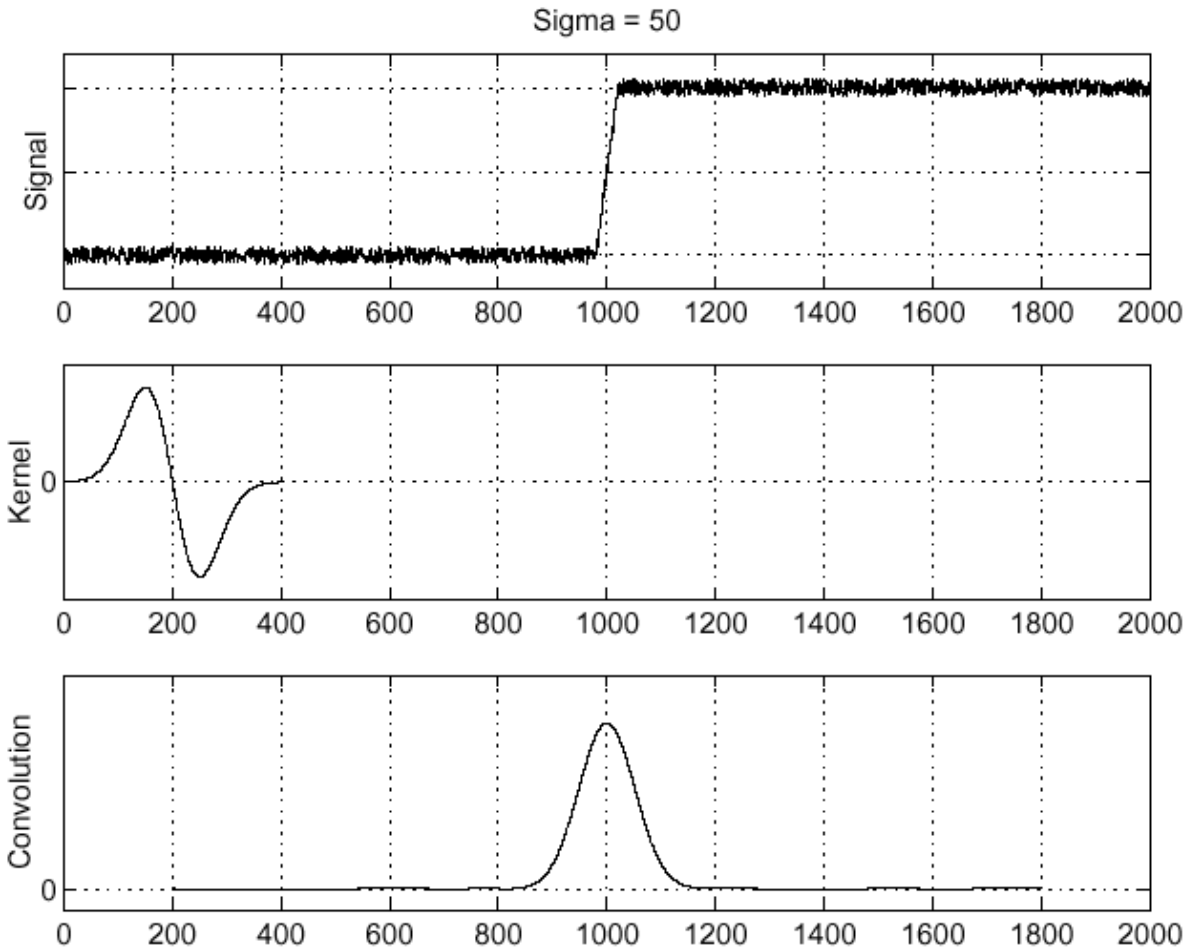
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

…saves us one operation.

f

$\frac{\partial}{\partial x}h$

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



Laplacian of Gaussian (LoG)

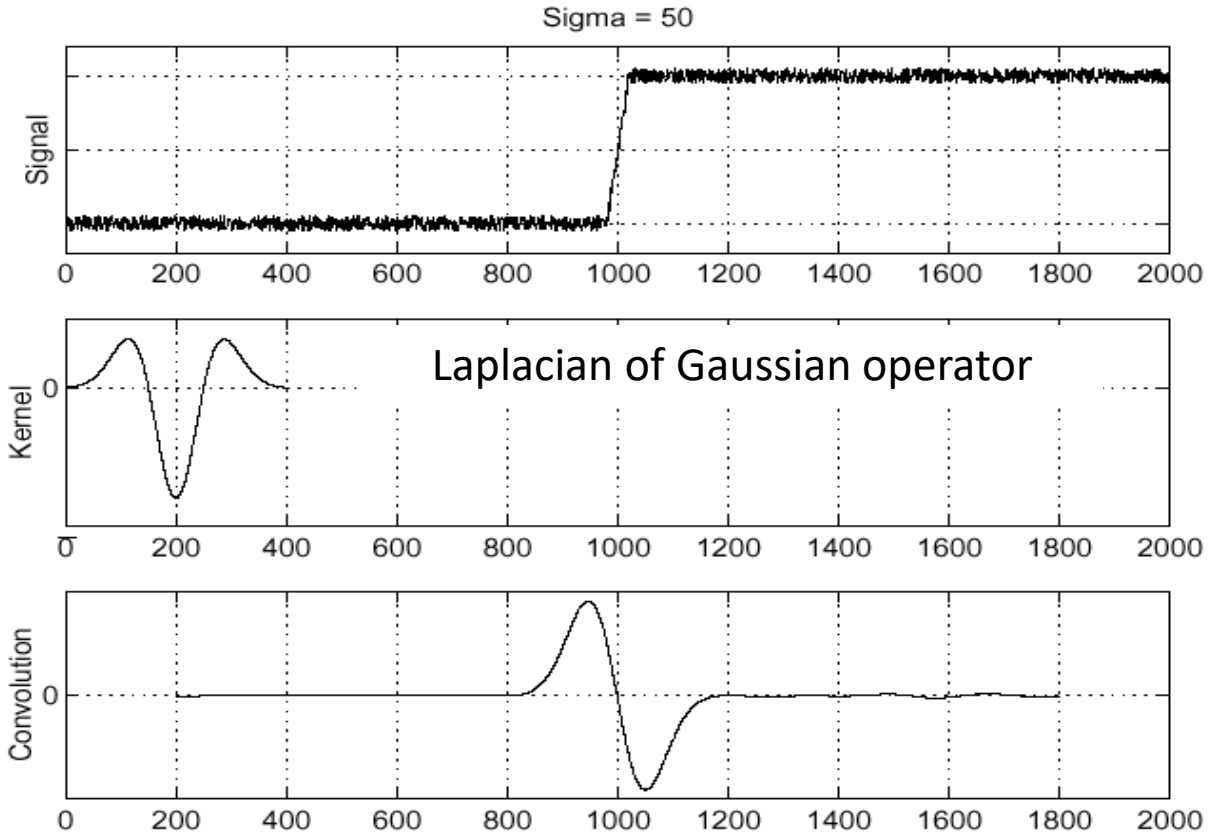
$$\frac{\partial^2}{\partial x^2}(h * f) = \left(\frac{\partial^2}{\partial x^2} h\right) * f$$

Laplacian of Gaussian

f

$\frac{\partial^2}{\partial x^2} h$

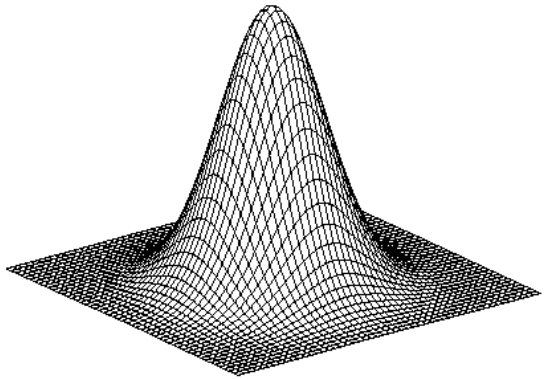
$(\frac{\partial^2}{\partial x^2} h) \star f$



Where is the edge?

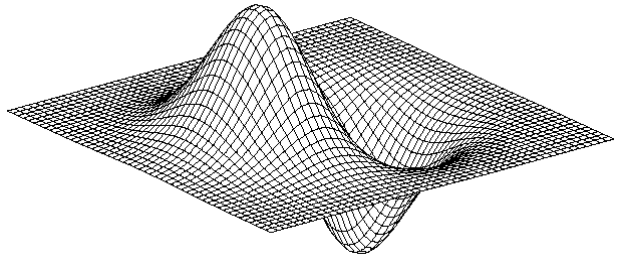
Zero-crossings of bottom graph !

2D Gaussian Edge Operators



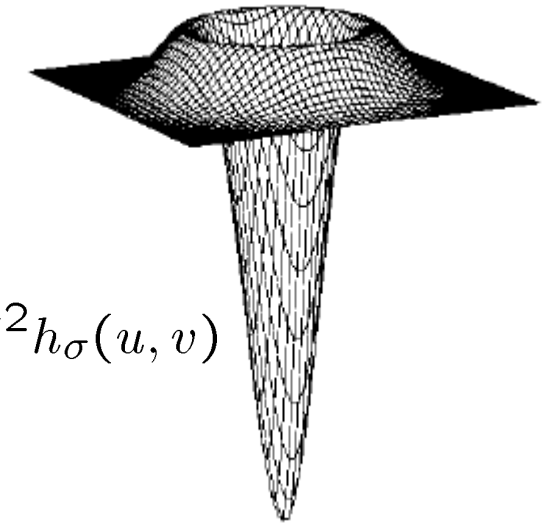
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Gaussian



$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Derivative of Gaussian (DoG)



$$\nabla^2 h_\sigma(u,v)$$

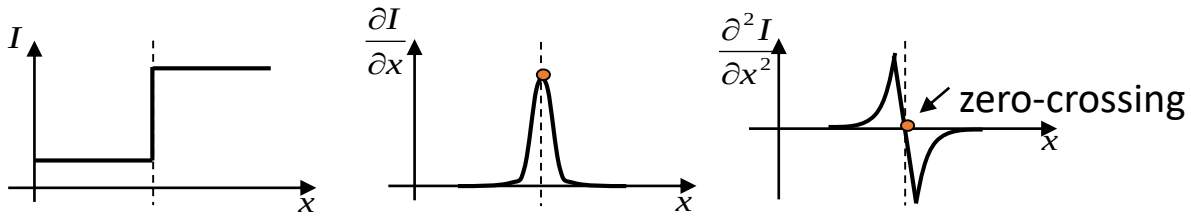
Laplacian of Gaussian

Mexican Hat (Sombrero)

- ∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Canny Edge Operator



- Smooth image *I* with 2D Gaussian:

$$G * I$$

- Find local edge normal directions for each pixel

$$\bar{\mathbf{n}} = \frac{\nabla(G * I)}{|\nabla(G * I)|}$$

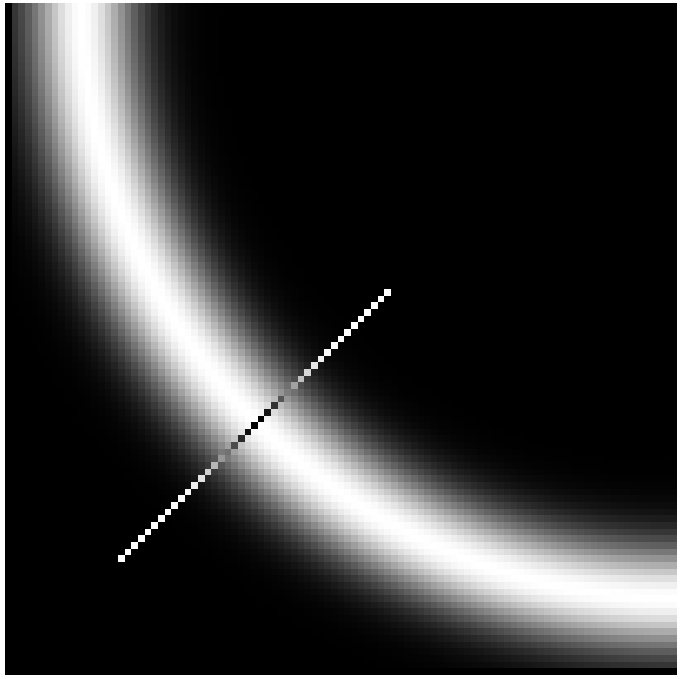
- Compute edge magnitudes

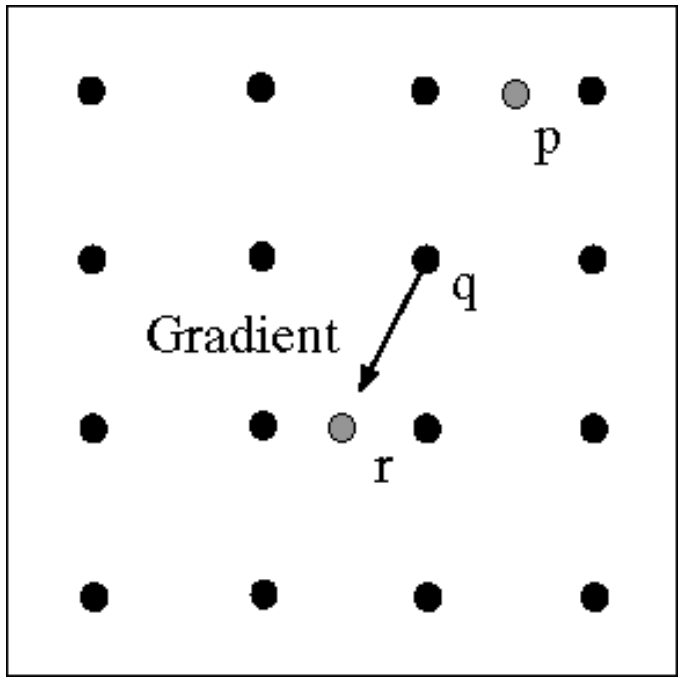
$$|\nabla(G * I)|$$

- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2(G * I)}{\partial \bar{\mathbf{n}}^2} = 0$$

Non-maximum Suppression





- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r

The Canny Edge Detector

original image (Lena)

The Canny Edge Detector

magnitude of the gradient

The Canny Edge Detector

After non-maximum suppression

Canny Edge Operator

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

- **The choice of σ depends on desired behavior**
  - large σ detects large scale edges
  - small σ detects fine features

Difference of Gaussians (DoG): Difference between two different Gaussians

- Laplacian of Gaussian can be approximated by the difference between two different Gaussians

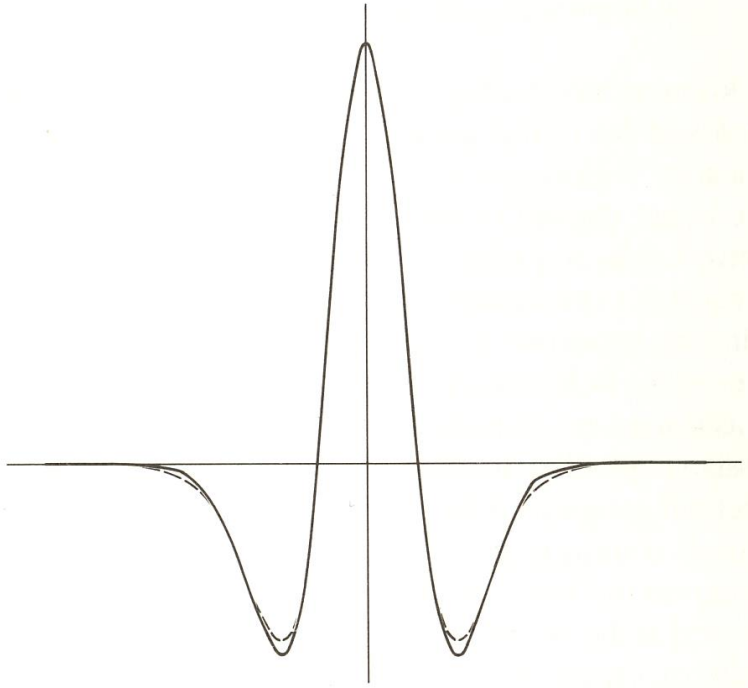
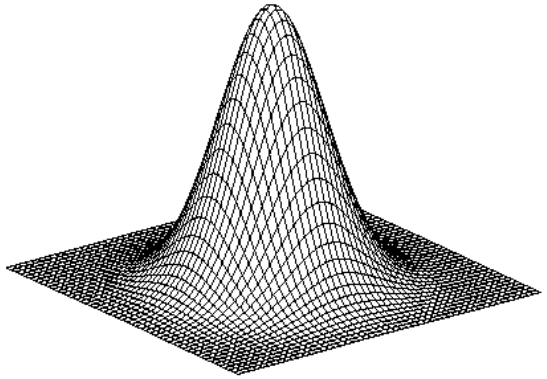


Figure 2–16. The best engineering approximation to $\nabla^2 G$ (shown by the continuous line), obtained by using the difference of two Gaussians (DOG), occurs when the ratio of the inhibitory to excitatory space constraints is about 1:1.6. The DOG is shown here dotted. The two profiles are very similar. (Reprinted by permission from D. Marr and E. Hildreth, "Theory of edge detection, " *Proc. R. Soc. Lond. B 204,* pp. 301–328.)

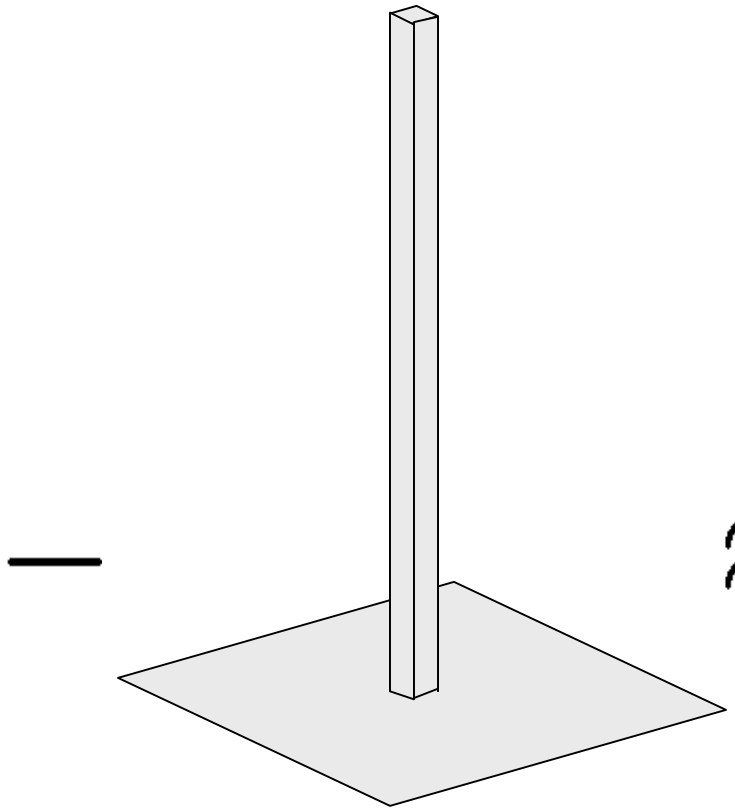
DoG Edge Detection

(a) $\sigma = 1$ (b) $\sigma = 2$ (b)-(a)

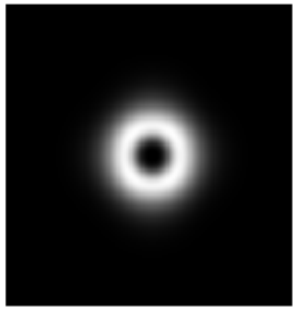
Gaussian – Image filter



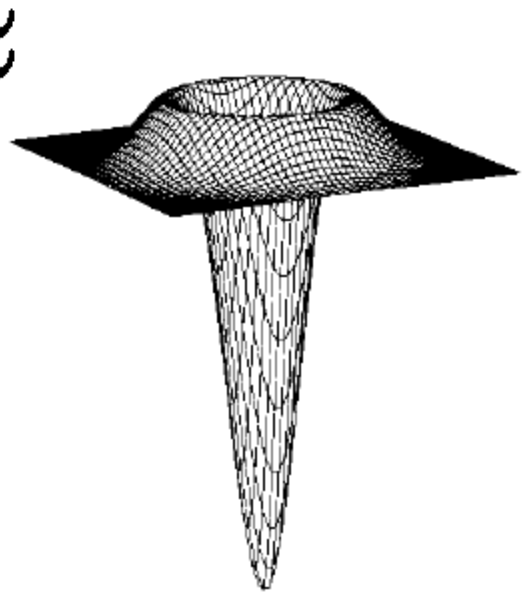
Gaussian



delta function



Fourier Transform



Unsharp Masking

MATLAB demo

```
g = fspecial('gaussian',15,2);
imagesc(g)
surfl(g)
gclown = conv2(clown,g,'same');
imagesc(conv2(clown,[-1 1],'same'));
imagesc(conv2(gclown,[-1 1],'same'));
dx = conv2(g,[-1 1],'same');
imagesc(conv2(clown,dx,'same');
lg = fspecial('log',15,2);
lclown = conv2(clown,lg,'same');
imagesc(lclown)
imagesc(clown + .2*lclown)
```

Edge Detection Operators

- **Thresholding for Decision Making (Significance of Differences)**

Edge Thresholding

- Standard Thresholding:

$$E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

  - Can only select “strong” edges.
  - Does not guarantee “continuity”.

- Hysteresis based Thresholding (use two thresholds)

$$\begin{array}{rll} & \|\nabla f(x,y)\| \geq t_1 & \text{definitely an edge} \\ t_0 \geq & \|\nabla f(x,y)\| < t_1 & \text{maybe an edge, depends on context} \\ & \|\nabla f(x,y)\| < t_0 & \text{definitely not an edge} \end{array}$$

Example: For “maybe” edges, decide on the edge if neighboring pixel is a strong edge.

Edge Detection

- Two Key Steps for Edge Detection
  - Difference Calculation between neighboring pixels;
  - Threshold for decision making.

Significant changes of illuminations between neighboring pixels!

Edge Detection

- Pixel Difference Calculation

1	2	1
0	0	0
-1	-2	-1

1	0	-1
2	0	-2
1	0	-1

2	1	0
1	0	-1
0	-1	-2

0	1	2
-1	0	1
-2	-1	0

Horizontal edge, Vertical edge, Northeastern, Southwestern

$$HOE(x,y) = | I(x-1,y-1) + 2I(x,y-1) + I(x+1,y-1)$$
$$- I(x-1,y+1) - 2I(x,y+1) - I(x+1,y+1) |$$

Edge Detection

- Decision Making

  (1) Local Maximum

$$MOE(x,y) = \max\{HOE(x,y), VOE(x,y), NOE(x,y), SOE(x,y)\}$$

  (2) Binary Classification

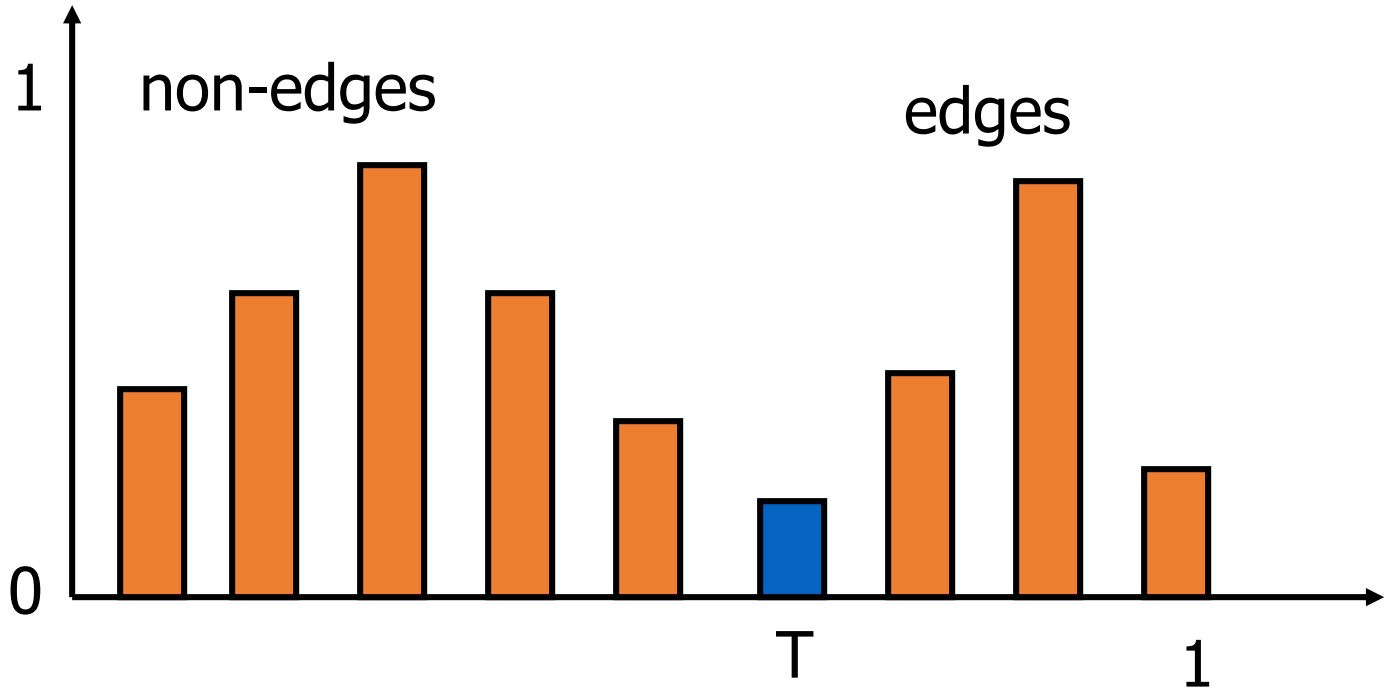
$$Y_E(x,y) = \begin{cases} 1, edge\_pixel, MOE(x,y) \geq T \\ 0, non\_edge\_pixel, MOE(x,y) < T \end{cases}$$

Edge Detection

- Threshold Determination for Decision Making

Relationships among neighboring pixels can defined as:

edges versus non-edge



Edge Detection

- Threshold Determination for Decision Making

Entropy for non-edge and edge pixels:

$$H(\overline{T}) = \max_{T=0,1,\ldots,M} \{H_{nsc}(T) + H_{sc}(T)\}.$$

$$P_{nsc}(i) = \frac{f_i}{\sum_{h=0}^{T} f_h}, \quad 0 \leq i \leq T,$$

Edge Detection

- Threshold Determination for Decision Making

$$
\begin{aligned}
H_{\mathrm{nsc}}(T+1) &= -\sum_{i=0}^{T+1} \frac{f_i}{P_0(T+1)} \log \frac{f_i}{P_0(T+1)} \\
&= -\frac{P_0(T)}{P_1(T+1)} \sum_{i=0}^{T+1} \frac{f_i}{P_0(T)} \log \left\{ \frac{f_i}{P_0(T)} \frac{P_0(T)}{P_0(T+1)} \right\} \\
&= \frac{P_0(T)}{P_0(T+1)} H_{\mathrm{nsc}}(T) - \frac{f_{T+1}}{P_0(T+1)} \log \frac{f_{T+1}}{P_0(T+1)} \\
&\quad - \frac{P_0(T)}{P_0(T+1)} \log \frac{P_0(T)}{P_0(T+1)},
\end{aligned}
$$

$$
\begin{aligned}
H_{\mathrm{sc}}(T+1) &= -\sum_{i=T+2}^{M} \frac{f_i}{P_1(T+1)} \log \frac{f_i}{P_1(T+1)} \\
&= -\frac{P_1(T)}{P_1(T+1)} \sum_{i=T+2}^{M} \frac{f_i}{P_1(T)} \log \left\{ \frac{f_i}{P_1(T)} \frac{P_1(T)}{P_1(T+1)} \right\} \\
&= \frac{P_1(T)}{P_1(T+1)} H_{\mathrm{sc}}(T) + \frac{f_{T+1}}{P_1(T+1)} \log \frac{f_{T+1}}{P_1(T+1)} \\
&\quad - \frac{P_1(T)}{P_1(T+1)} \log \frac{P_1(T)}{P_1(T+1)}.
\end{aligned}
$$

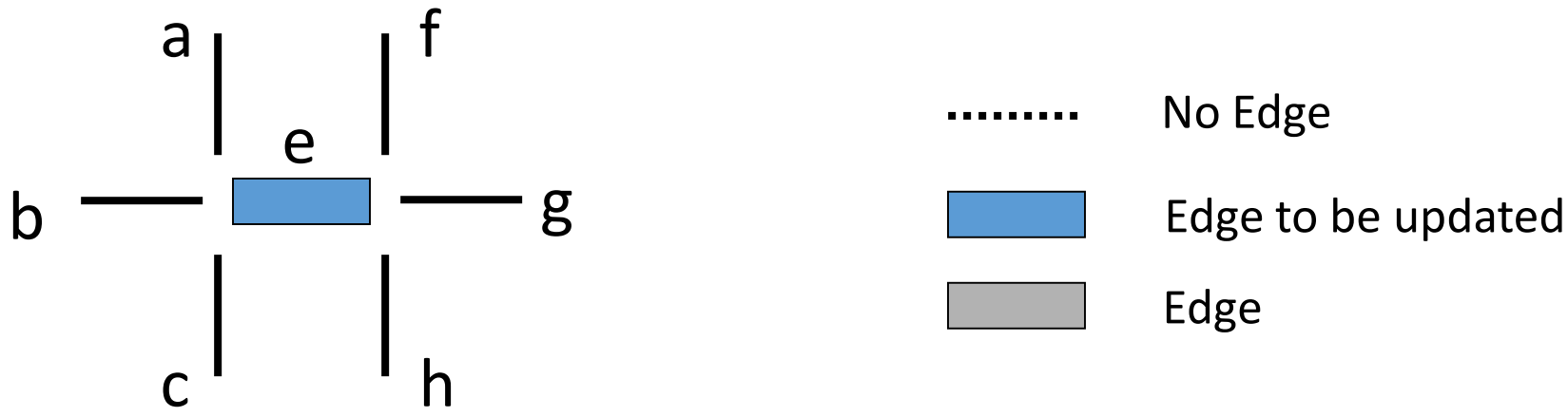
Binary Clustering for Edge Detection

Edge Detection Operators

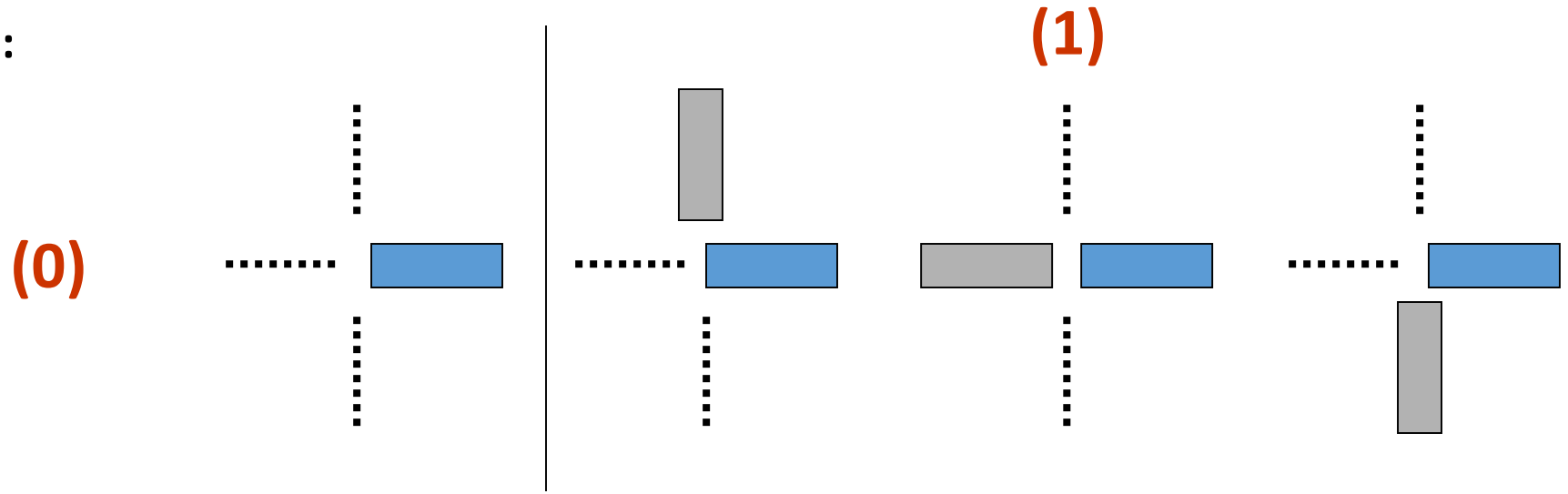
- **Edge Relaxation**

Edge Relaxation

- **Parallel – Iterative method to adjust edge values on the basis of neighboring edges.**



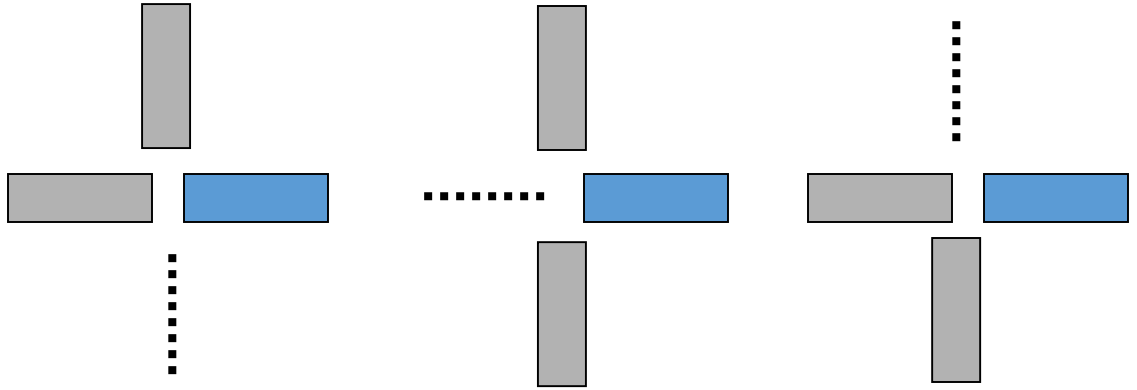
- **Vertex Types:**



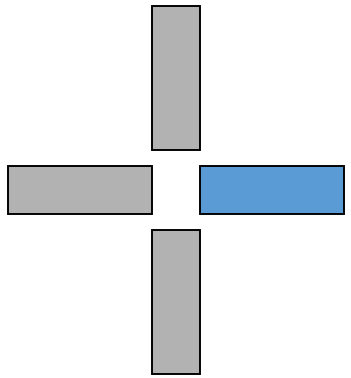
Edge Relaxation

- **Vertex Types (continued):**

(2)



(3)



Edge Relaxation Algorithm

- Action Table:

Edge Type	Decrement	Increment	Leave as is
	0 - 0	1 - 1	0 - 1
	0 - 2	1 - 2	2 - 2
	0 - 3	1 - 3	2 - 3
			3 - 3

- Algorithm:

Step 0: Compute Initial Confidence of each edge e:

$$C^0(e) = \frac{Magnitude\ of\ \ e}{Maximum\ Gradient\ in\ image}$$

Step 1: Initialize $k = 1$

Step 2: Compute Edge Type of each edge e

Step 3: Modify confidence $C^k(e)$ based on $C^{k-1}(e)$ and Edge Type

Step 4: Test to see if all $C^k(e)'s$ have CONVERGED to either 1 or 0. Else go to Step 2.

Edge Relaxation

Fig. 3.22 Edge relaxation results. (a) Raw edge data. Edge strengths have been thresholded at 0.25 for display purposes only. (b) Results after five iterations of relaxation applied to (a). (c) Different version of (a). Edge strengths have been thresholded at 0.25 for display purposes only. (d) Results after five iterations of relaxation applied to (c).

Performance

- **Sobel and Prewitt methods** are very effectively providing good edge maps.
- **Kirsch and Robinson methods** require more time for calculation and their results are not better than the ones produced by Sobel and Prewitt methods.
- **Roberts and Laplacian methods** are not very good as expected.

A Quick Note

- Matlab’s image processing toolbox provides edge function to find edges in an image.
- Edge function supports six different edge-finding methods: Sobel, Prewitt, Roberts, Laplacian of Gaussian, Zero-cross, and Canny.
- Edge is a powerful edge-detection method