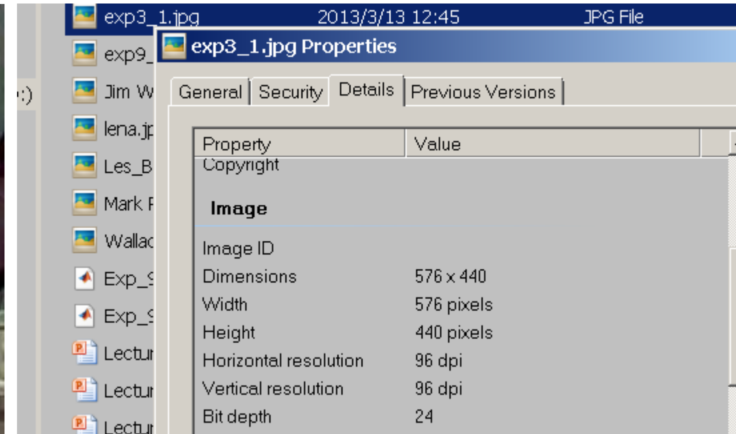
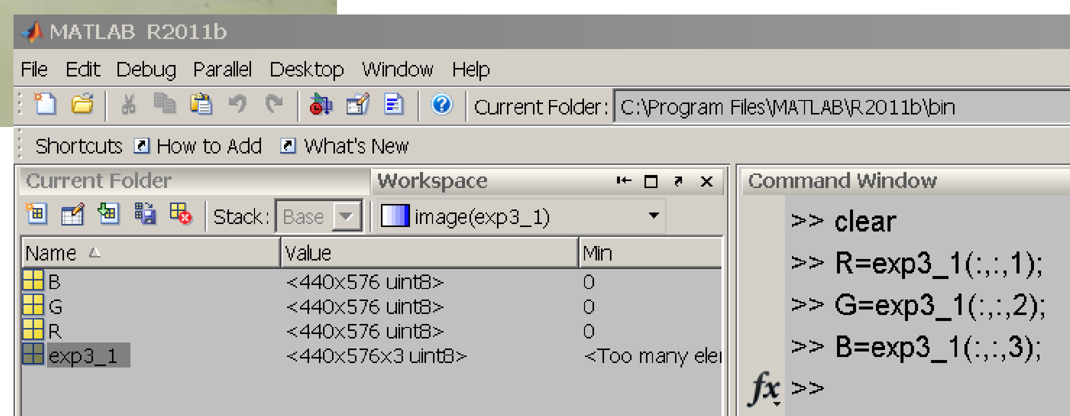

# MATLAB and Its Application in Engineering


**Assoc. Prof. Kirin Shi**

**Shanghai Jiao Tong University**

# Colour image vs 3D Matrix



# Colour image vs 3D Matrix

imshow(R)

imshow(G)

imshow(B)

# Colour image vs 3D Matrix

```
>>  I=exp3_1;
>> I(:,:,1)= I(:,:,1)+50;
>> imshow(I)
```

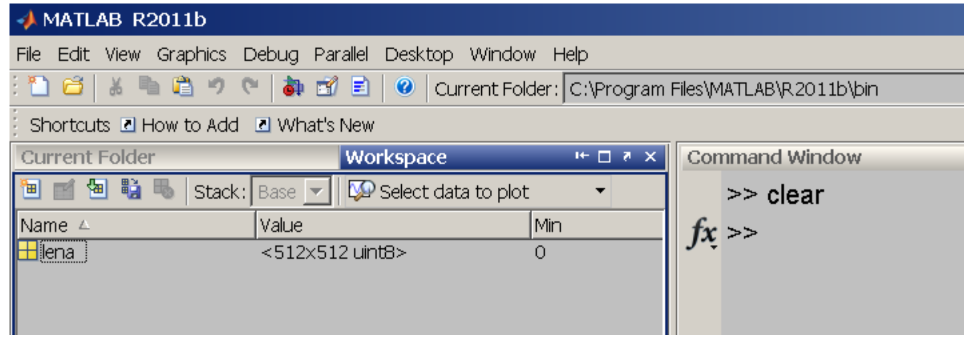
# Colour image vs 3D Matrix

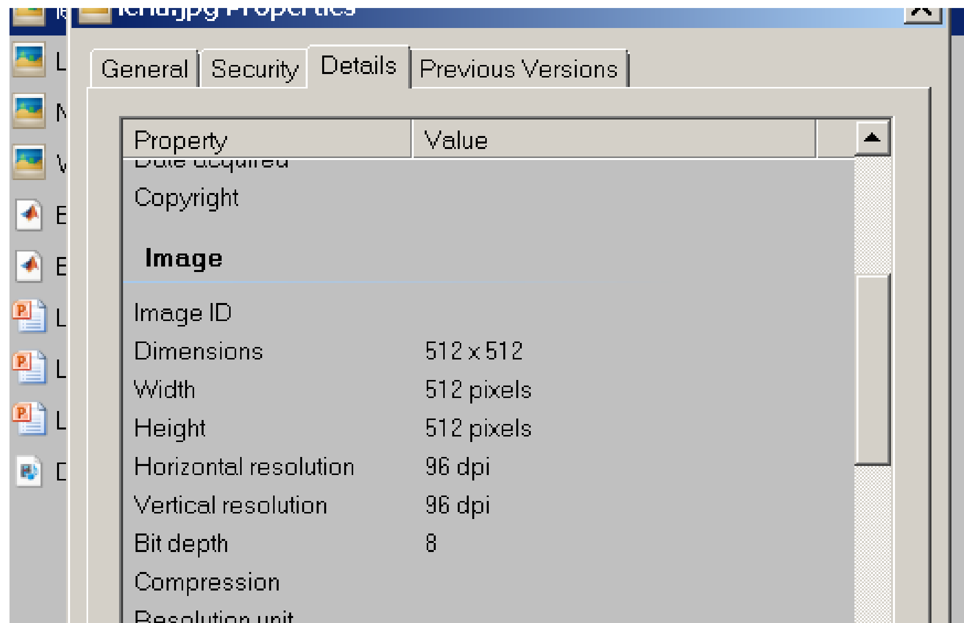
```
>>  I=exp3_1;
>> I(:,:,2)= I(:,:,2)+50;
>> imshow(I)
```

# Colour image vs 3D Matrix

```
>>  I=exp3_1;
>> I(:,:,3)= I(:,:,3)+50;
>> imshow(I)
```

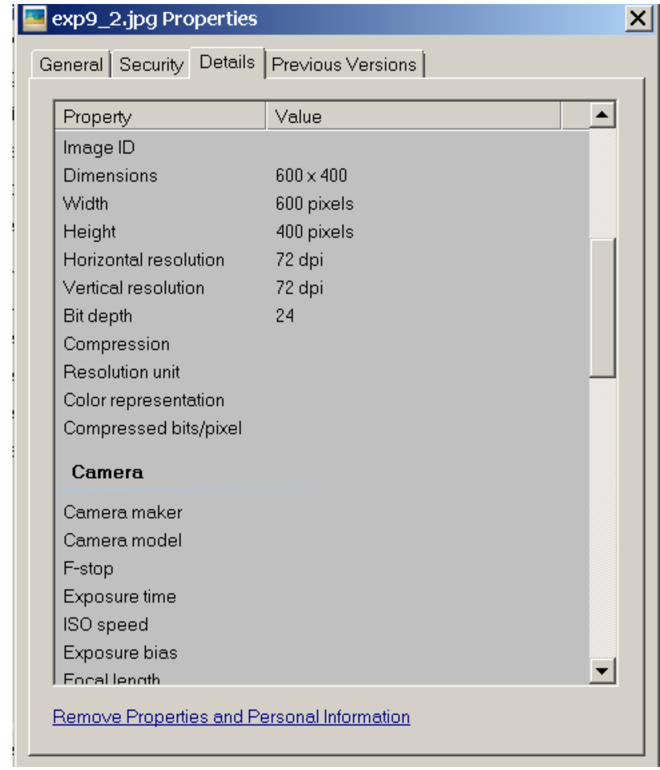
# Grey image vs 2D Matrix



# Resizing image

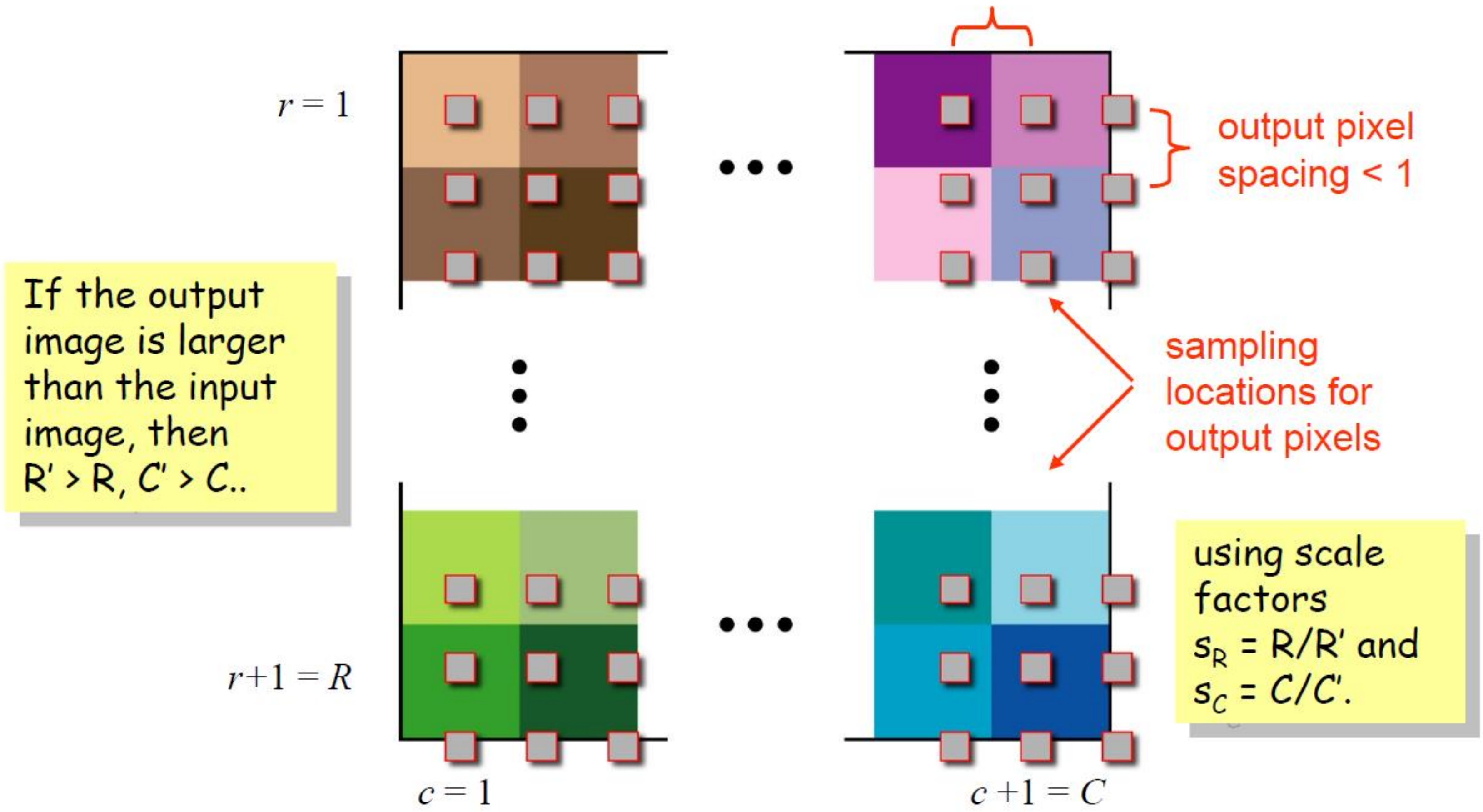
**Original size 600×400 → 2×larger**

# Resizing image

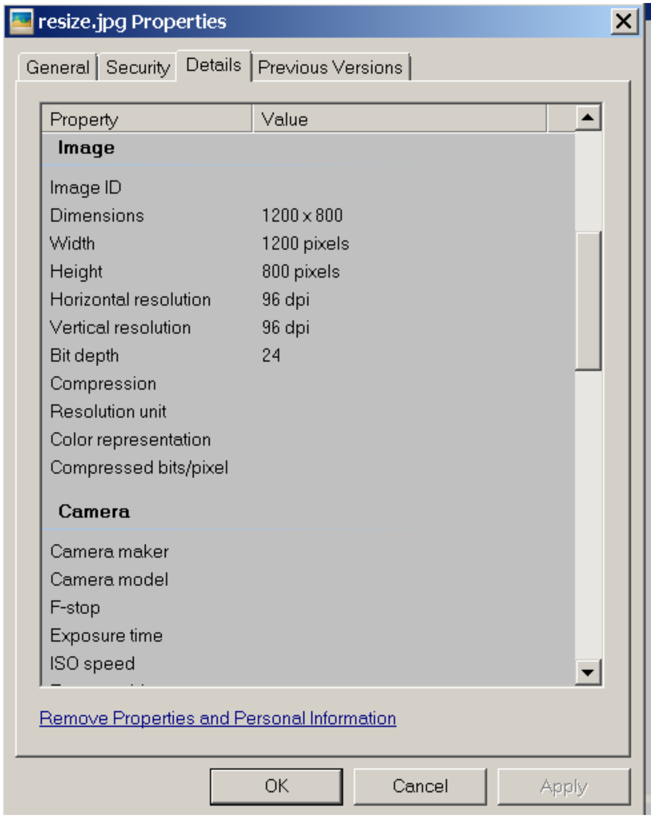
```
I=imread('exp9_2.jpg');
I=double(I);
I_new=zeros(size(I,1)*2,size(I,2)*2,3);
[r,c]=meshgrid(1:size(I,2),1:size(I,1));

rc=linspace(1, size(I,1), size(I,1)*2);
cc=linspace(1, size(I,2), size(I,2)*2);
[r_new,c_new]=meshgrid(cc,rc);

I_new(:,:,1)=interp2(r,c,I(:,:,1),r_new,c_new,'spline');
I_new(:,:,2)=interp2(r,c,I(:,:,2),r_new,c_new,'spline');
I_new(:,:,3)=interp2(r,c,I(:,:,3),r_new,c_new,'spline');

imwrite(uint8(I_new),'resize.jpg','jpg')
```

# Resizing image



# Resizing image

## Resized image 1200×800

# Image Segmentation

- Point, Line, and Edge Detection
- Line Detection Using the Hough Transform
- Thresholding
- Region-Based Segmentation
- Segmentation Using the Watershed Transform

# Point, Line, and Edge Detection

The most common way to look for discontinuities is to run a mask through the image.

For a 3 X 3 mask this involves computing the sum of products of the coefficients with the intensity levels contained in the region encompassed by the mask. The response, R, of the mask at any point in the image is given by

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \, ;$$

where $z_i$ is the intensity of the pixel associated with mask coefficient $w_i$, As before, the response of the mask is defined at its center.

# Point Detection

The detection of isolated points embedded in areas of constant or nearly constant intensity in an image is straightforward in principle. Using the mask shown in below,  we say that an isolated point has been detected at the location on which the mask is centered if

$$|R| \geq T$$

where T is a nonnegative threshold.

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

# Point Detection

If T is given, the following command implements the point-detection approach just discussed:

**>> g = abs ( imfilter ( double (f) , w ) ) >= T ;**

where **f** is the input image, **w** is an appropriate point-detection mask and **g** is an image containing the points detected.

# Point Detection

If T is not given, its value often is chosen based on the filtered result, in which case the previous command string is divided into three basic steps:

1. Compute the filtered image, abs ( imfilter ( double(f) , w ) ) ;
2. find the value for T using the data from the filtered image;
3. compare the filtered image against T.

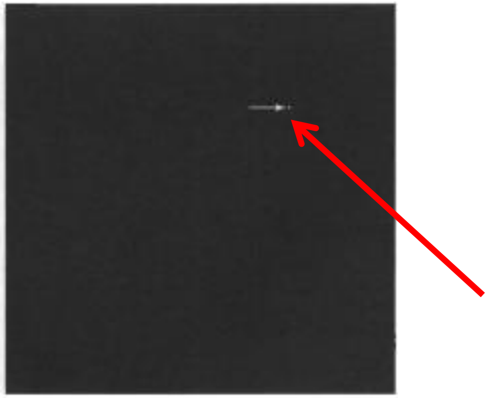
The following example illustrates this approach

# Point Detection

Following figure (a) shows an image, f, with a nearly invisible black point in the northeast quadrant of the sphere. We detect the point as follows:

```
>> w = [-1 -1 -1; -1 8 -1 ; -1 -1 -1] ;
>> g = abs ( imfilter(f , w ) );
>> T = max ( g ( : ) ) ;
>> g = g >= T ;
>> imshow ( g )
```

(a)



(b)

# Point Detection

By selecting T to be the maximum value in the filtered image, g , and then finding all points in g such that g>=T, we identify the points that give the largest response. The assumption is that these are isolated points embedded in a constant or nearly constant background. Because T was selected in this case to be the maximum value in g, there can be no points in g with values greater than T; we used the >= operator ( instead of =) for consistency in notation. As Fig (b) shows, there was a single isolated point that satisfied the condition g >= T with T set to max ( g ( : ) ) .

# Line Detection

If the mask in Fig (a) were moved around an image, it would respond more strongly to lines (one pixel thick) oriented horizontally. With a constant background, the maximum response results when the line passes through the middle row of the mask.

The second mask in fig responds best to lines oriented at +45 ° ;

The third mask to vertical lines;

The fourth mask to lines in the -45 ° direction.

The coefficients of each mask sum to zero, indicating a zero response in areas of constant intensity.

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 2 | 2 | 2 |
| −1 | −1 | −1 |

Horizontal

(a)

| | | |
|---|---|---|
| 2 | −1 | −1 |
| −1 | 2 | −1 |
| −1 | −1 | 2 |

+45°

(b)

| | | |
|---|---|---|
| −1 | 2 | −1 |
| −1 | 2 | −1 |
| −1 | 2 | −1 |

Vertical

(c)

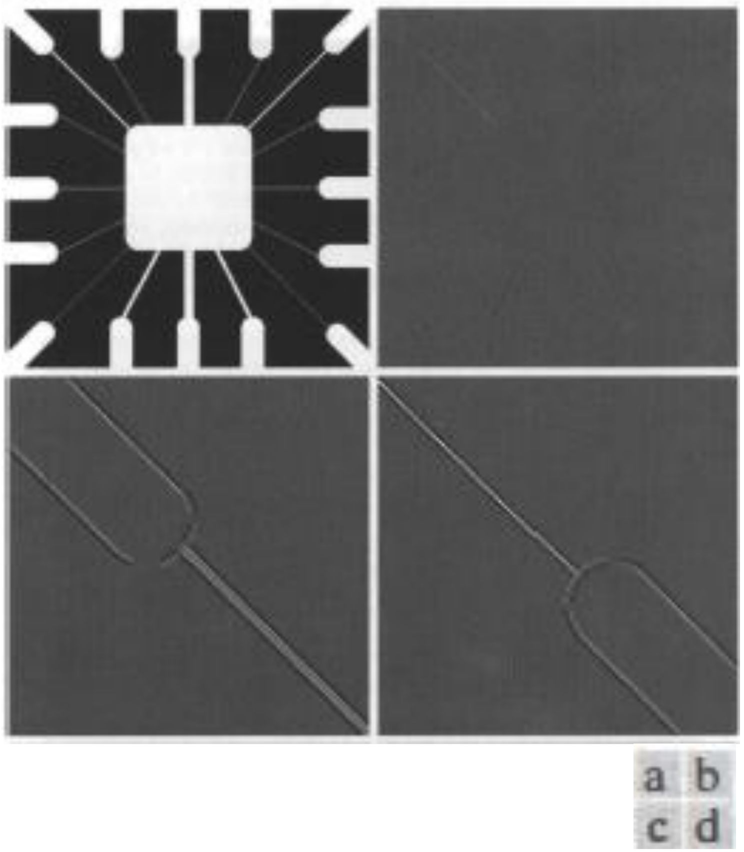
| | | |
|---|---|---|
| −1 | −1 | 2 |
| −1 | 2 | −1 |
| 2 | −1 | −1 |

−45°

(d)

# Line Detection

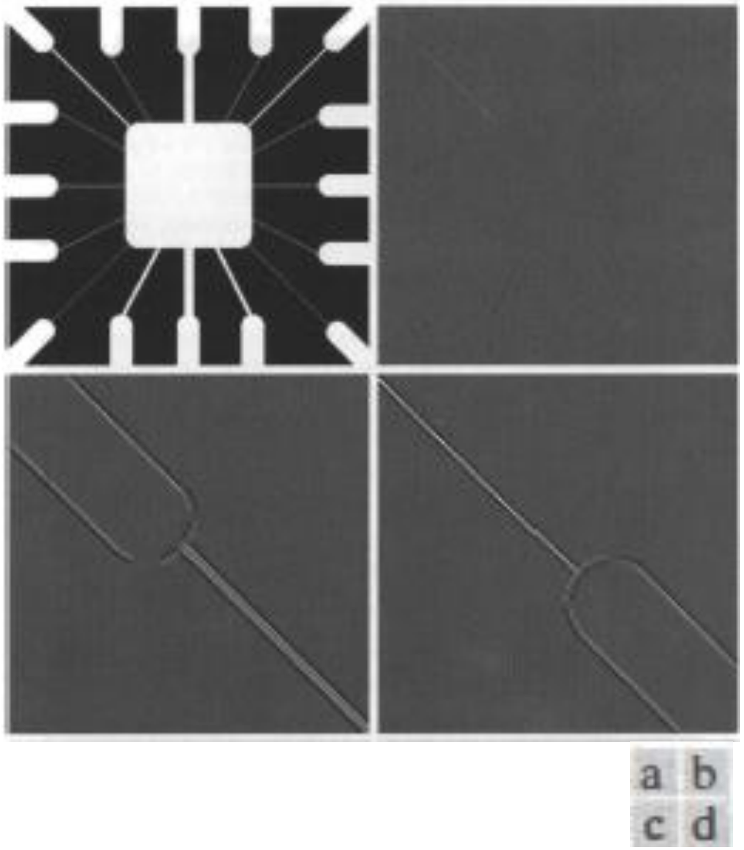
Figure (a) shows a digitized (binary) portion of a wire-bond template for an electronic circuit. The image size is 486 X 486 pixels. Suppose that we want to find all the lines that are one pixel thick, oriented at +45° . For this, we use the second mask in last Fig. Figures (b) through (d) were generated using the following commands, where f is the image in Fig (a):



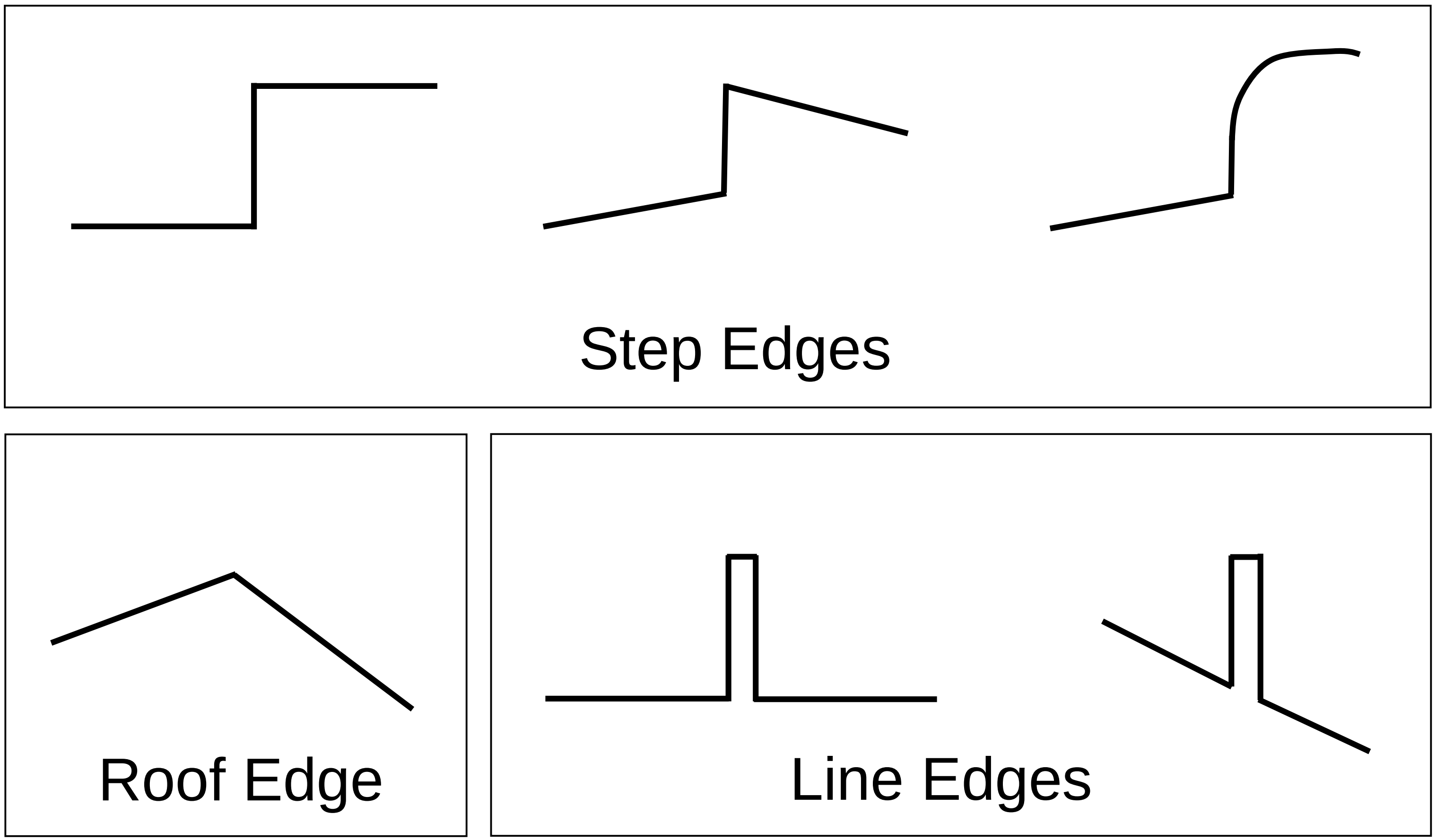

(a) Image of a wire-bond template.
(b) Result of processing with the +45 ° detector.
(c) Zoomed view of the top, left region of (b).
(d) Zoomed view of the bottom, right section of (b).

# Line Detection

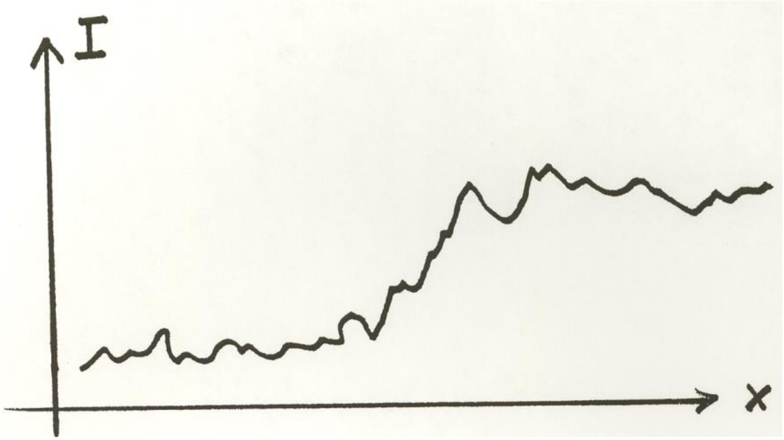
```
>> w = ( 2 - 1 - 1 ; - 1 2 - 1 ; - 1 - 1 2 ] ;
>> g = imfilter (double (f) , w) ;
>> imshow( g) % Fig . 1 1 . 4 ( b )
>> gtop = g(1:120,1:120); %Top, left section
>> figure , imshow(gtop) % Fig(c)
>> gbot = g(end-119:end, end-119:end) ;
>> figure , imshow (gbot) % Fig(d)
>> g = abs ( g ) ;
>> figure , imshow ( g) % Fig(e)
>> T = max ( g ( : ) ) ;
>> g = g >= T ;
>> figure, imshow ( g ) % Fig (f)
```



# Edge Types



# Real Edges



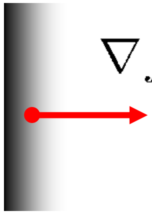

Noisy and Discrete!

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

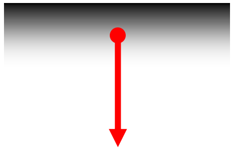
- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

# Gradient

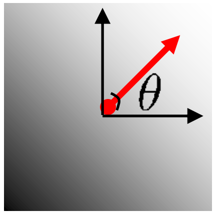
- Gradient equation: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$
- Represents direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

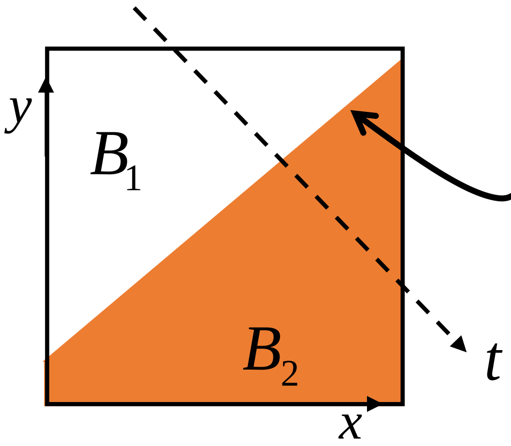



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- 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




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):

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

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

- Squared gradient:

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

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)

# 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):

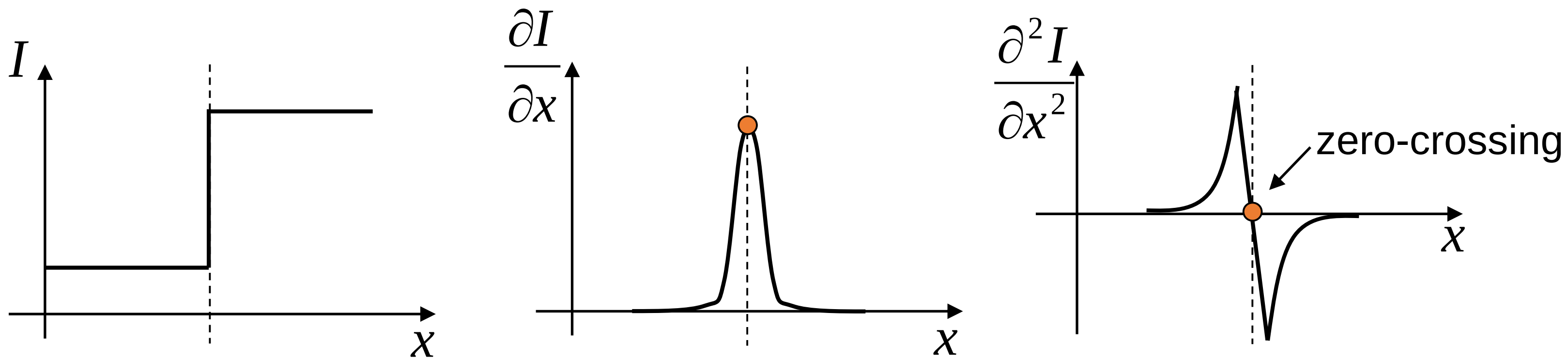
$$\frac{\partial I}{\partial x} = +\sin\theta (B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

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

- 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:

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

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

| | | |
|---|---|---|
| $I_{i,j+1}$ | $I_{i+1,j+1}$ | $\varepsilon$ |
| $I_{i,j}$ | $I_{i+1,j}$ | |

Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline\end{array} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & -1 \\ \hline\end{array}$$

# 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 :

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

# The Sobel Operators

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

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$s_x$ $\qquad$ $s_y$

# Comparing Edge Operators

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

Good Localization
Noise Sensitive
Poor Detection

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):

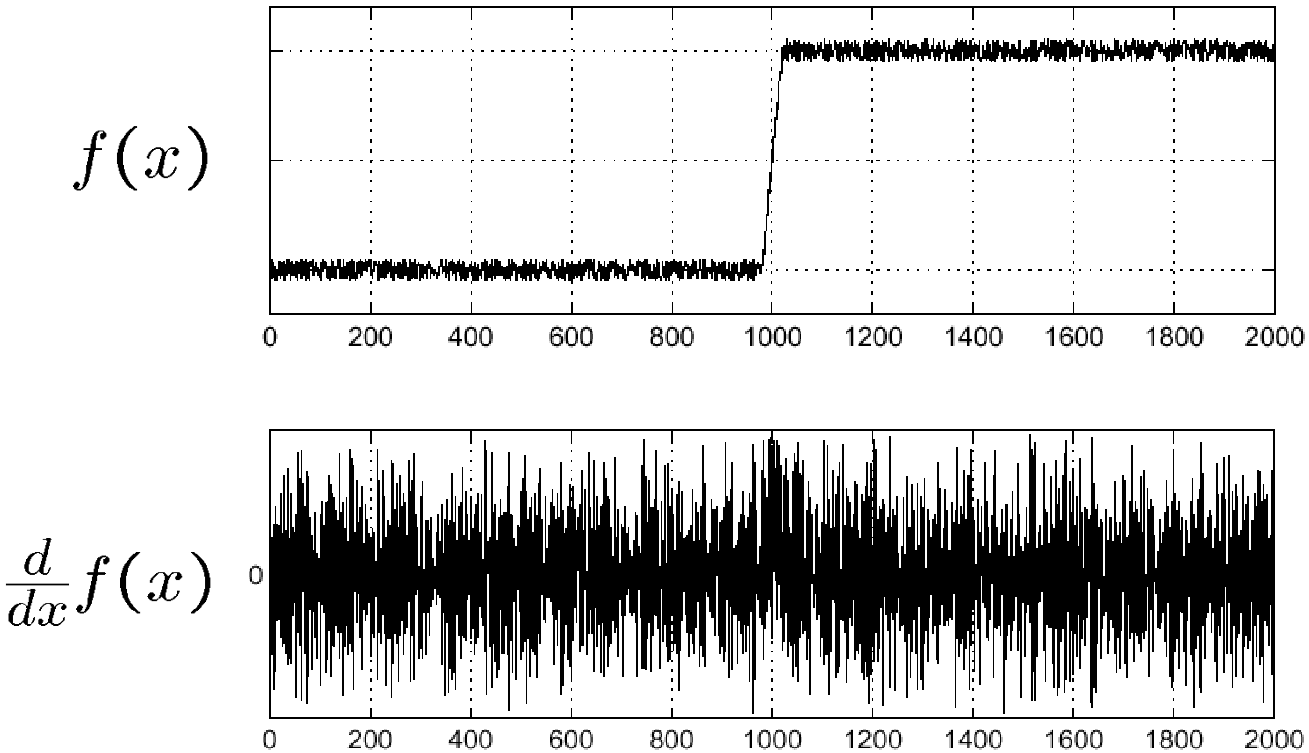
| | | | | |
|---|---|---|---|---|
| -1 | -2 | 0 | 2 | 1 |
| -2 | -3 | 0 | 3 | 2 |
| -3 | -5 | 0 | 5 | 3 |
| -2 | -3 | 0 | 3 | 2 |
| -1 | -2 | 0 | 2 | 1 |

| | | | | |
|---|---|---|---|---|
| 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 |

Poor Localization
Less Noise Sensitive
Good Detection

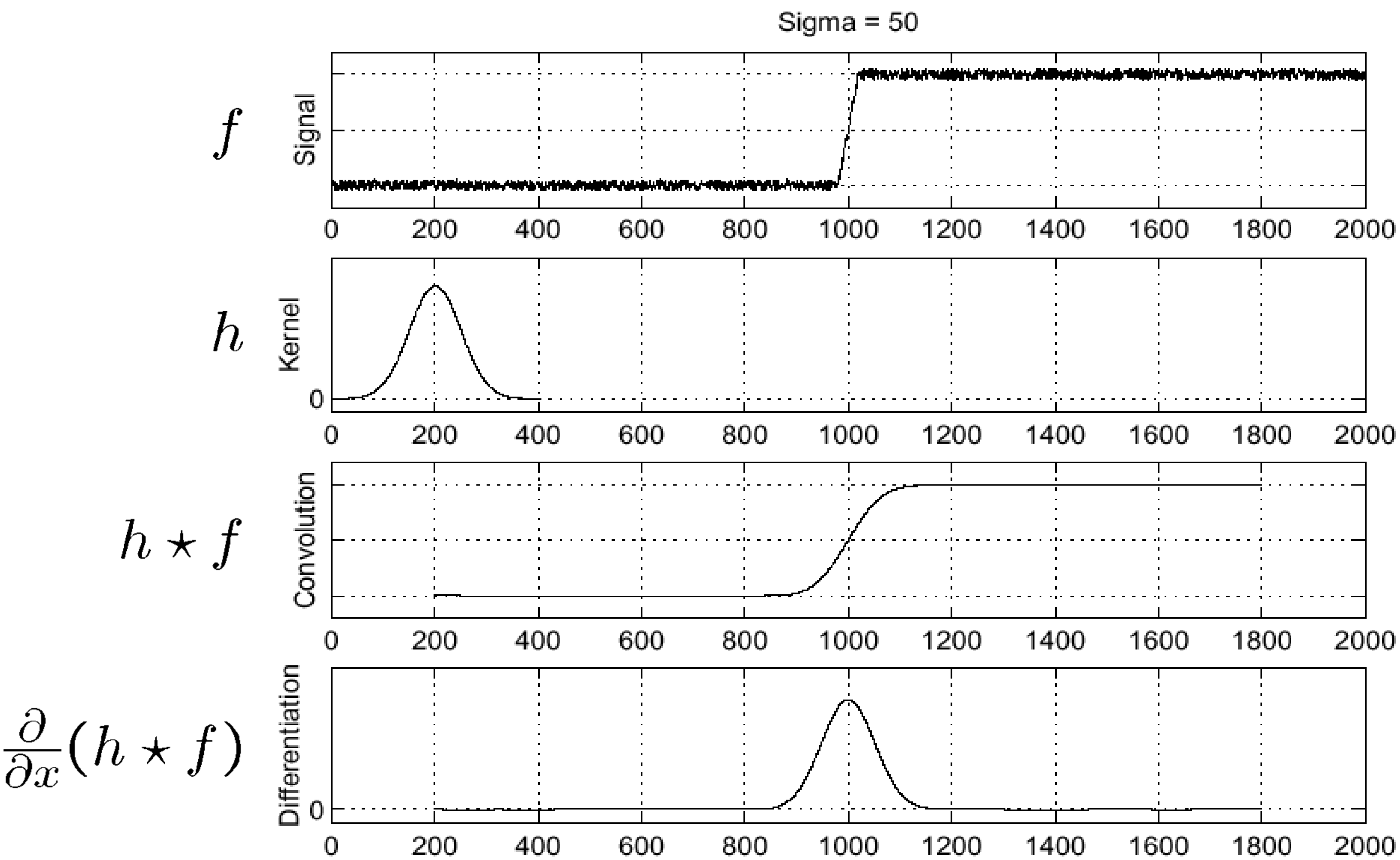
# Effects of Noise

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




Where is the edge??

# Solution:  Smooth First




Where is the edge?

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

# Derivative Theorem of Convolution

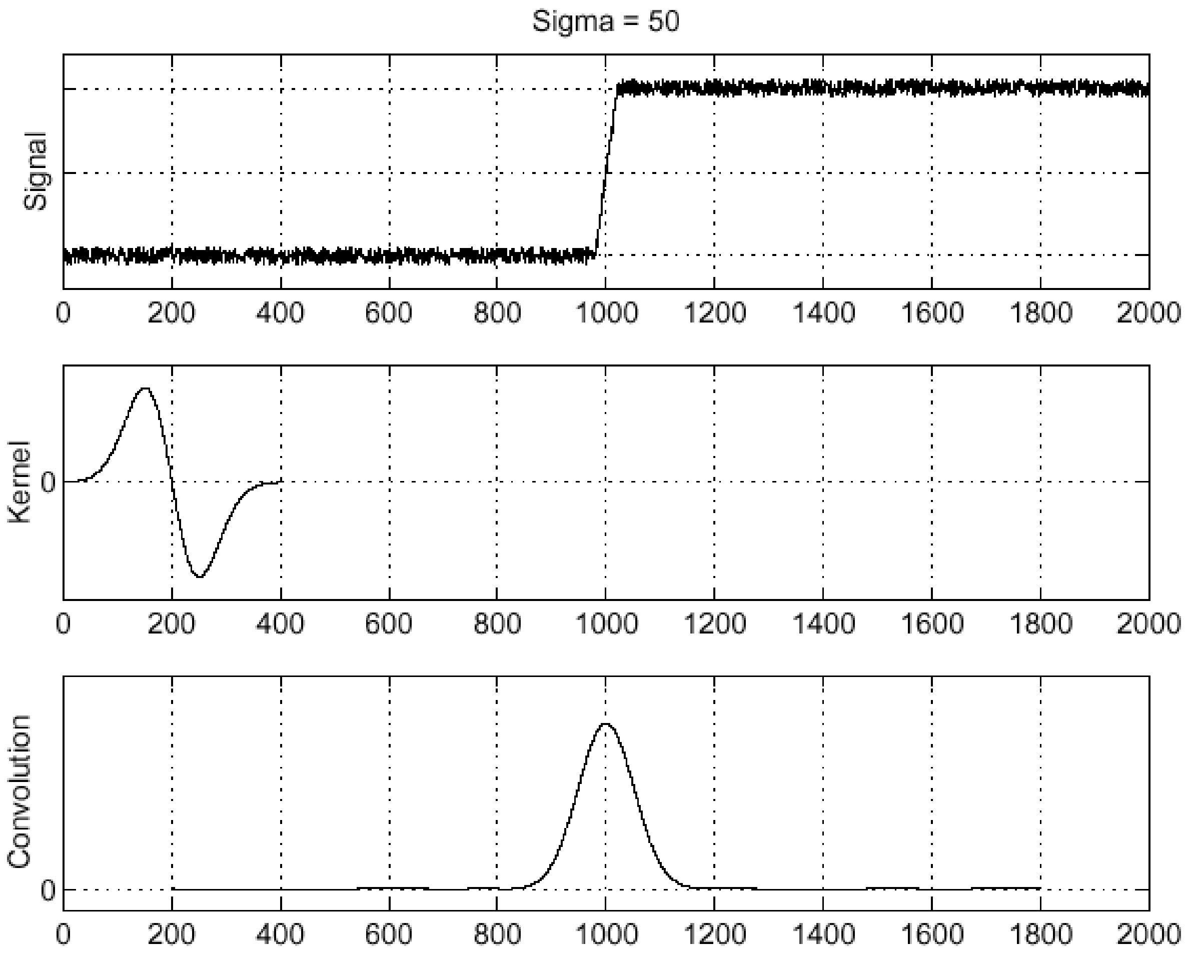
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

…saves us one operation.

$f$

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

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



# Laplacian of Gaussian (LoG)

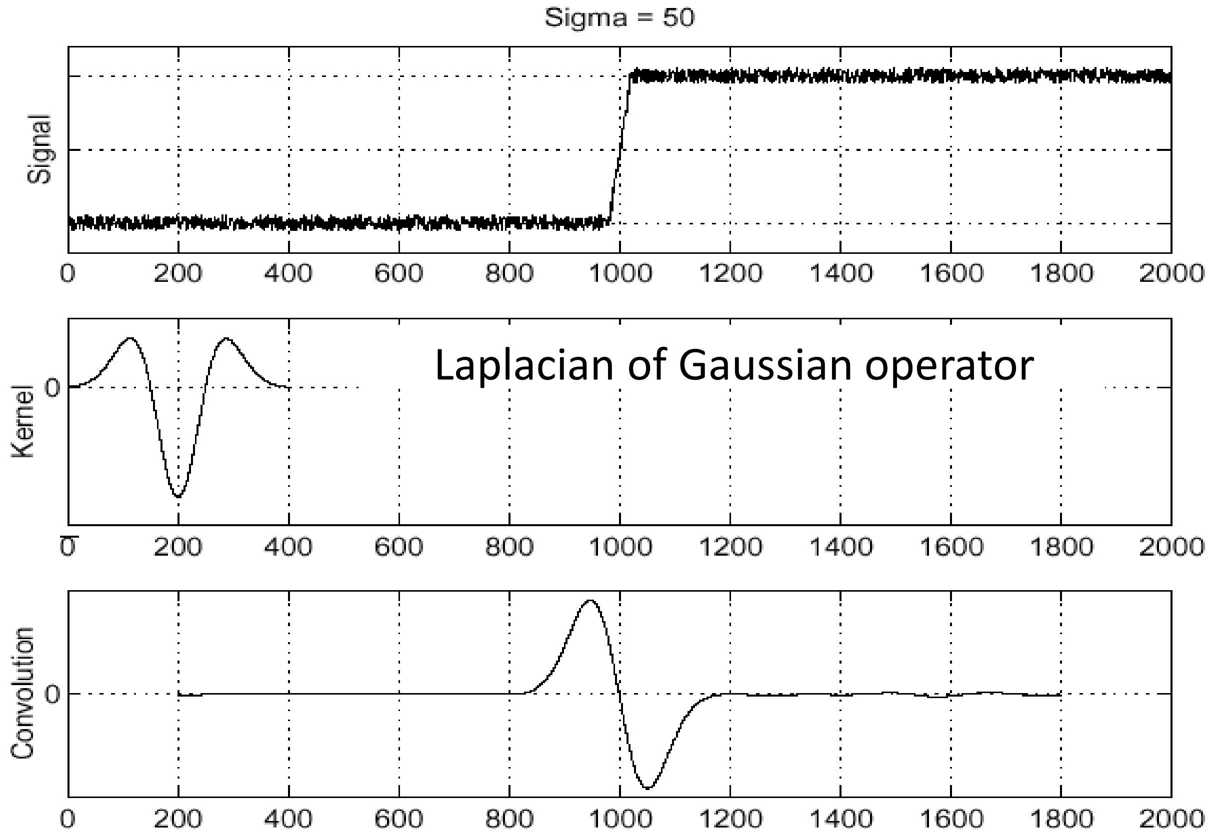
$$\frac{\partial^2}{\partial x^2}(h * f) = \boxed{\left(\frac{\partial^2}{\partial x^2} h\right)} * f$$

Laplacian of Gaussian

$f$

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

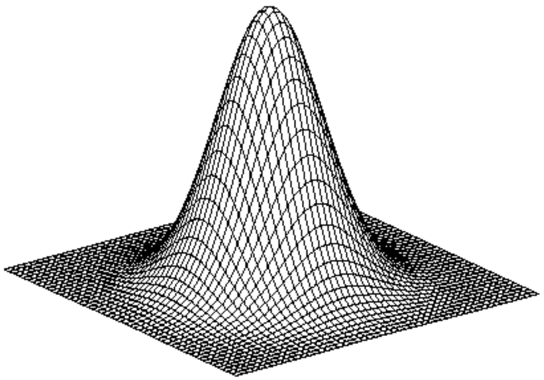
$\left(\frac{\partial^2}{\partial x^2} h\right) \star f$




Where is the edge?

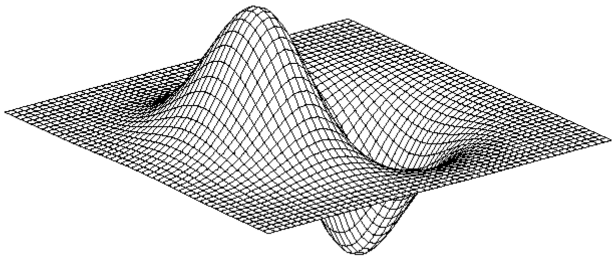
Zero-crossings of bottom graph !

# 2D Gaussian Edge Operators



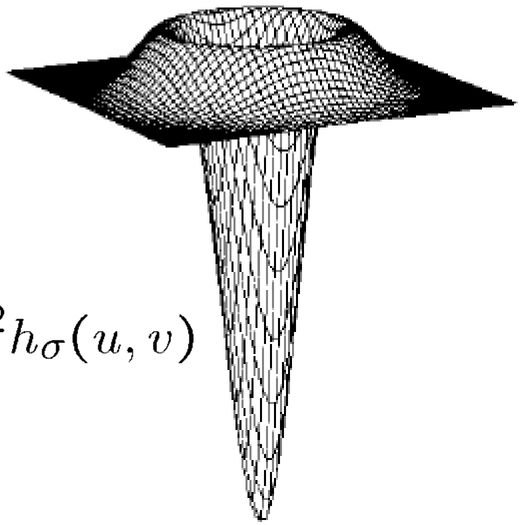
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

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)

- $\nabla^2$ is the **Laplacian** operator:

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

# Canny Edge Detector

1) Smooth image with a Gaussian
   - optimizes the trade-off between noise filtering and edge localization
2) Compute the Gradient magnitude using approximations of partial derivatives
   - 2x2 filters
3) Thin edges by applying non-maxima suppression to the gradient magnitude
4) Detect edges by double thresholding

# Gradient

- At each point convolve with

$$G_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

- magnitude and orientation of the Gradient are computed as

$$M[i,j] = \sqrt{P[i,j]^2 + Q[i,j]^2}$$

$$\theta[i,j] = \tan^{-1}(Q[i,j], P[i,j])$$

- Avoid floating point arithmetic for fast computation

# Non-Maxima Suppression

- Thin edges by keeping large values of Gradient
  - not always at the location of an edge
  - there are many thick edges

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 3 |
| 0 | 0 | 0 | 1 | 2 | 1 | 3 | 1 |
| 0 | 0 | 2 | 1 | 2 | 1 | 1 | 0 |
| 0 | 1 | 3 | 2 | 1 | 1 | 0 | 0 |
| 0 | 3 | 2 | 1 | 0 | 0 | 1 | 0 |
| 2 | 3 | 2 | 0 | 0 | 1 | 0 | 1 |
| 2 | 3 | 2 | 0 | 1 | 0 | 2 | 1 |

# Non-Maxima Suppression (2)

- Thin the broad ridges in M[i,j] into ridges that are only one pixel wide
- Find local maxima in M[i,j] by suppressing all values along the line of the Gradient that are not peak values of the ridge

$$
\begin{matrix}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 3 \\
3 & 0 & 0 & 1 & 2 & 1 & 3 & 1 \\
0 & 0 & 2 & 1 & 2 & 1 & 1 & 0 \\
0 & 1 & 3 & 2 & 1 & 1 & 0 & 0 \\
0 & 3 & 2 & 1 & 0 & 0 & 1 & 3 \\
2 & 3 & 2 & 0 & 0 & 1 & 0 & 1 \\
2 & 3 & 2 & 0 & 1 & 0 & 2 & 1
\end{matrix}
$$

false edges

gaps

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 2 | 1 | 3 | 0 |
| 0 | 0 | 2 | 1 | 2 | 0 | 0 | 0 |
| 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 3 | 0 | 0 | 1 | 0 | 2 | 0 |

false edges

- The suppressed magnitude image will contain many false edges caused by noise or fine texture

# Thresholding

- Reduce number of false edges by applying a threshold T
  - all values below T are changed to 0
  - selecting a good values for T is difficult
  - some false edges will remain if T is too low
  - some edges will disappear if T is too high
  - some edges will disappear due to softening of the edge contrast by shadows

# Double Thresholding

- Apply two thresholds in the suppressed image
  - $T_2 = 2T_2$
  - two images in the output
  - the image from $T_2$ contains fewer edges but has gaps in the contours
  - the image from $T_1$ has many false edges
  - combine the results from $T_1$ and $T_2$
  - link the edges of $T_2$ into contours until we reach a gap
  - link the edge from $T_2$ with edge pixels from a $T_1$ contour until a $T_2$ edge is found again

$T_2=2$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 2 | | 3 | 0 |
| 0 | 0 | 2 | | 2 | 0 | 0 | 0 |
| 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 0 | 0 | 0 | 0 | 2 | 0 |

gaps filled from $T_1$

$T_1=1$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 2 | 1 | 3 | 0 |
| 0 | 0 | 2 | 1 | 2 | 0 | 0 | 0 |
| 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 3 | 0 | 0 | 1 | 0 | 2 | 0 |

- A $T_2$ contour has pixels along the green arrows
- Linking: search in a 3x3 of each pixel and connect the pixel at the center with the one having greater value
- Search in the direction of the edge (direction of Gradient)

# The Canny Edge Detector

original image (Lena)

# The Canny Edge Detector

magnitude of the gradient

# The Canny Edge Detector

After non-maximum suppression

# Edge detection by subtraction

original

# Edge detection by subtraction

smoothed (5x5 Gaussian)

# DoG Edge Detection

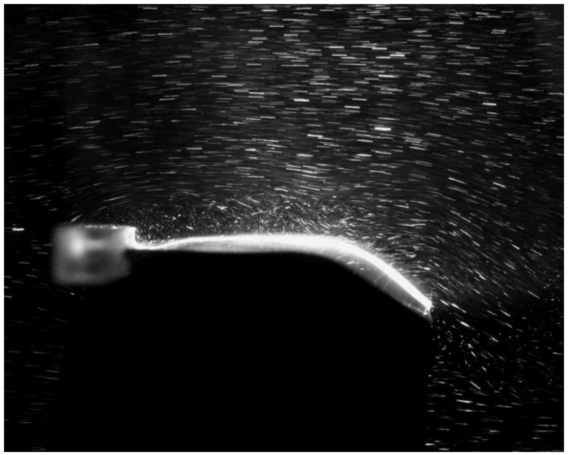
(a) $\sigma = 1$

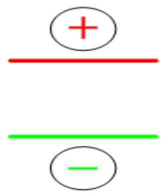
(b) $\sigma = 2$

(b)-(a)

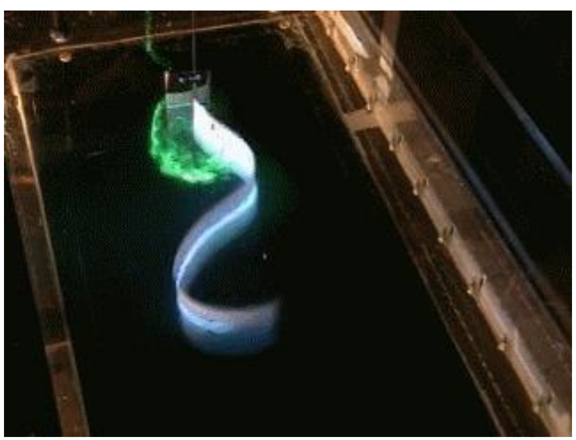
# Application

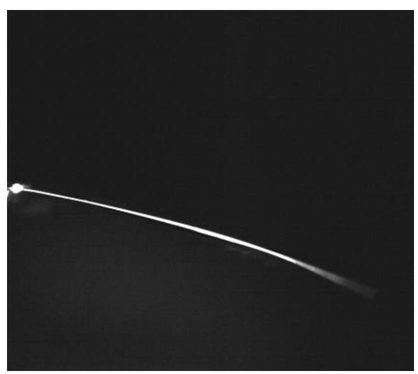
## Energy harvesting eel

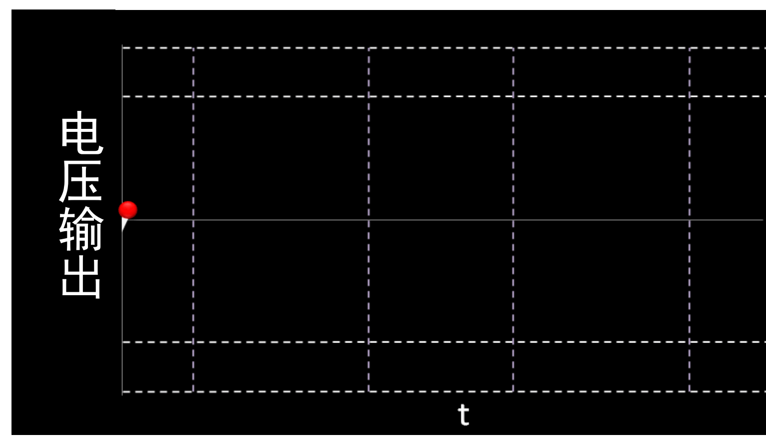




**PVDF membrane**
(SJTU Micro Tech. Inst.)



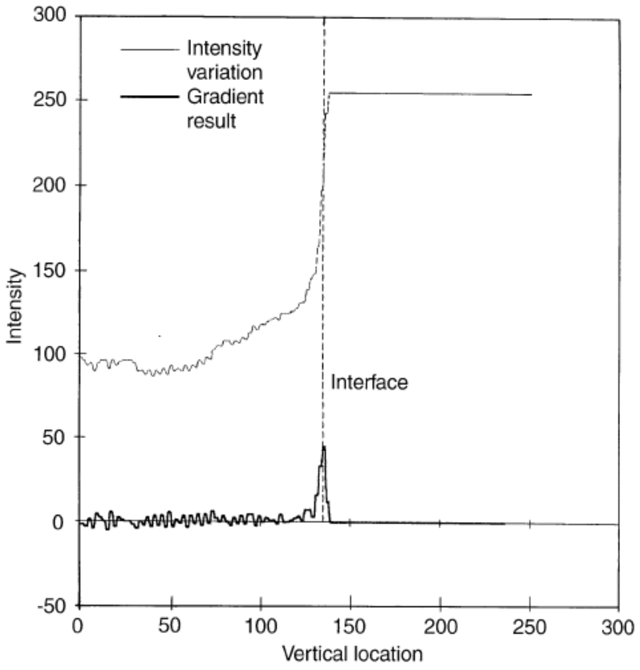


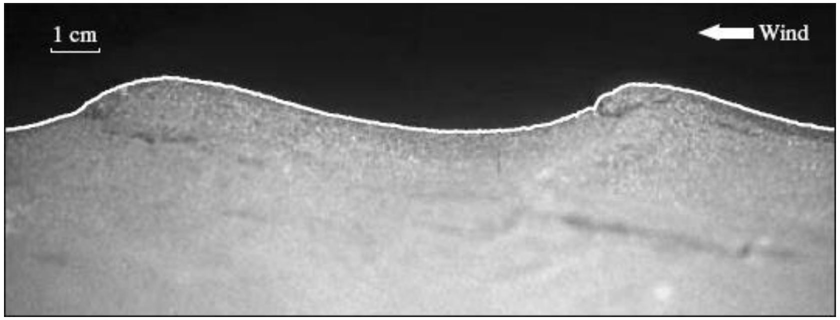


# Application

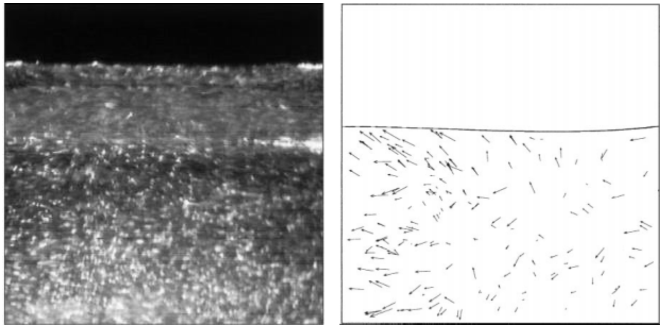
Interface in the PIV image

- Free surface detection.
  - Law *et al*. (1999), Zarruk (2005), Li *et al*. (2005), Mukto *et al*. (2007) .
- Maximum gradient location is defined as the interface.
- Mostly used technique.



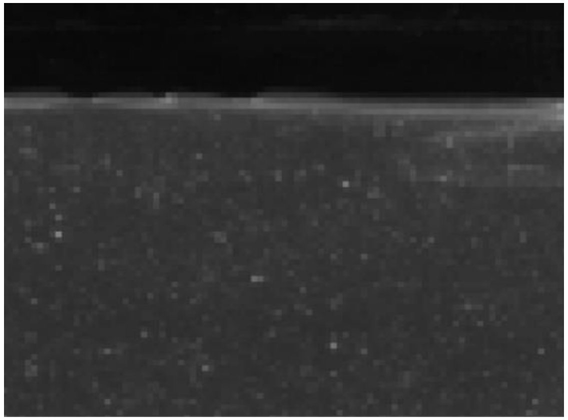

Mukto *et al.* (2007)



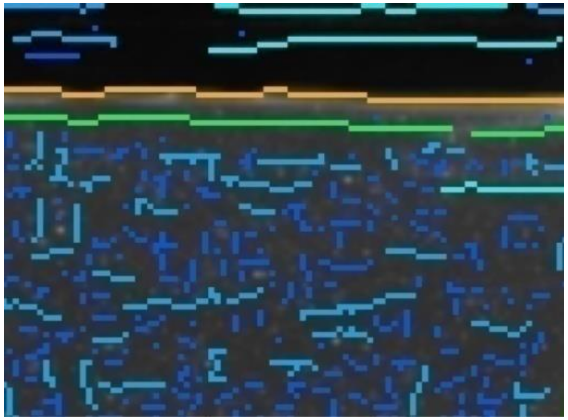
Law *et al.* (1999)

# Edge detection

- Edge detection
  - Image processing that extracts features from a digital image.
- Canny edge detection (Canny, 1986)
  - Linear filtering with a Gaussian kernel and then computes the vertical and horizontal intensity differences for each pixel.
  - Canny edge detection requires excessive computing time to the entire image.

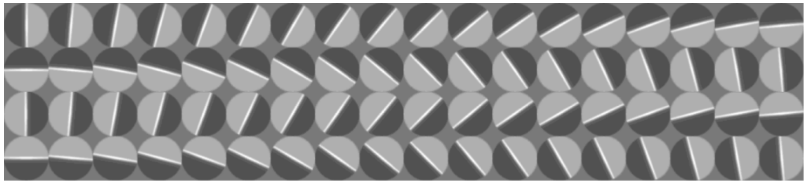


Original PIV image

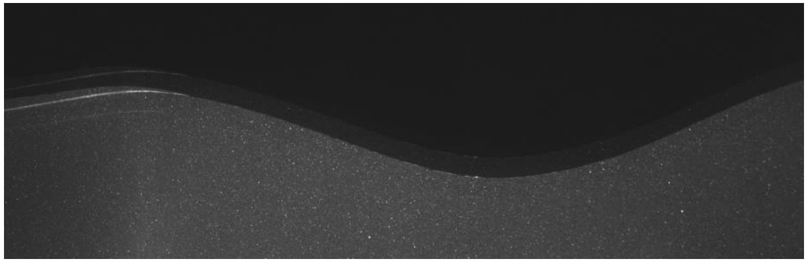


Edges by “Canny edge detection”

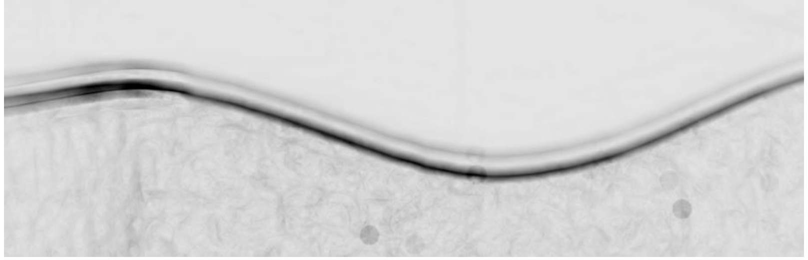
# Application



Textons



Original particle image



Probability histogram

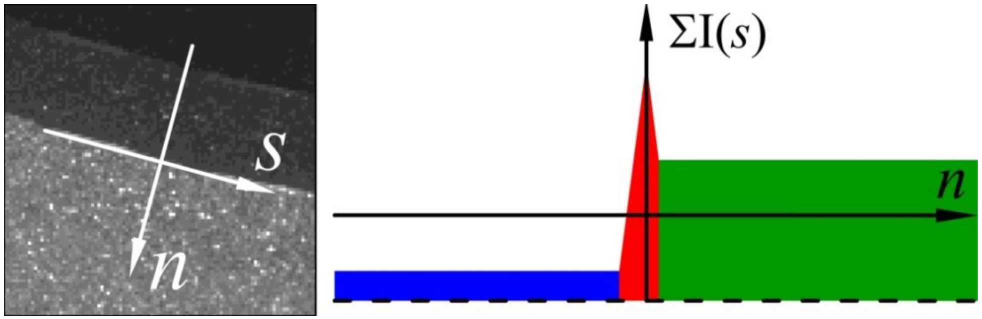
– 72 convolutions are performed and the maximum values are plotted.

– Interface tracking
  - To overcome this problem, the convolutions were only calculated for the interface points.
  - The computing time can be significantly reduced to a reasonable level.

# Define texton

## Definition of the textons,

$$T(i,\ j,\ \theta) = \frac{1}{t_w}\frac{1}{t_d}\frac{1}{4\pi}\exp\left[-\frac{i^2+j^2}{2t_w{}^2}-\frac{(i\sin\theta - j\cos\theta)^2}{2t_d{}^2}\right] + A(i,\ j,\ \theta) - A_{avg}$$




$$\text{where } A(i,\ j,\ \theta) = \begin{cases} A_p, & \text{for } i\cos(\theta+\pi/2) + j\sin(\theta+\pi/2) \geq 0 \\ A_b, & \text{for } i\cos(\theta+\pi/2) + j\sin(\theta+\pi/2) < 0 \end{cases},$$

$\theta$ : inclined angle of the texton $\qquad$ $t_w$ : size of the square texton

$t_d$ : width of the interface imposed by the Gaussian distribution

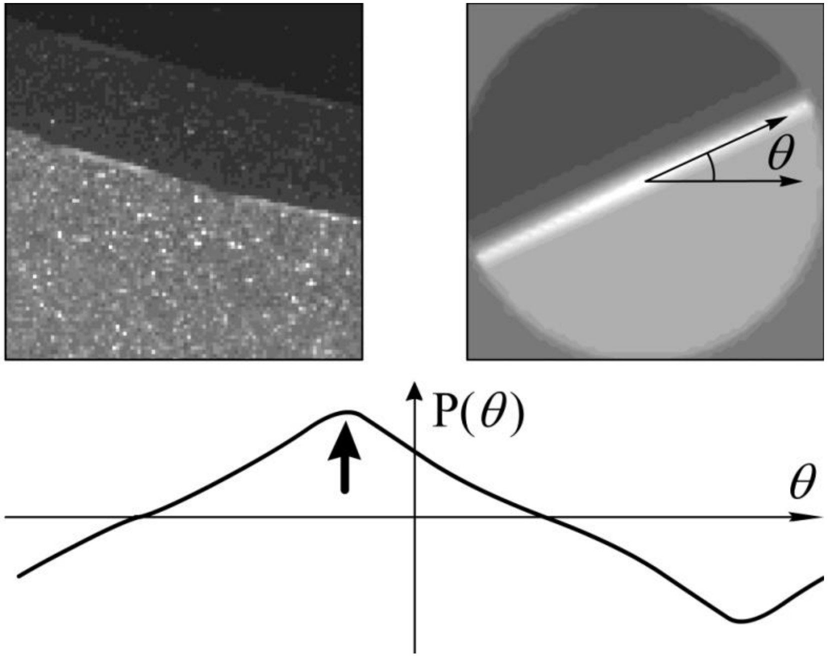
$A_b$ and $A_p$ : averaged intensities of the boundary and particle regions

$A_{avg}$ : offset value that adjust the sum of every pixel of $T$ to zero.

# Interface tracking

## Interface segments

- When the interface is tracked by using textons, the *x* and *y* locations of the interface and its inclined angle *q* are necessary.
- *q* can be determined according to the pre-described textons, which maximizes the integral of the equation. Here, I (*x*,*y*) is the convolution window.




$$\theta_k\ (x_k,\ y_k)$$
$$= \max \int_{-t_w/2}^{t_w/2} \int_{-t_w/2}^{t_w/2} \mathrm{I}\ (x_k + i,\ y_k + j)\ T\ (i,\ j,\ \theta)\ \mathrm{d}i\mathrm{d}j$$

# 3. PIV resutls