OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Dr Waqar Shahid Qureshi
Waqarshahid.qureshi@universityofgalway.ie

Feature Detection and Matching

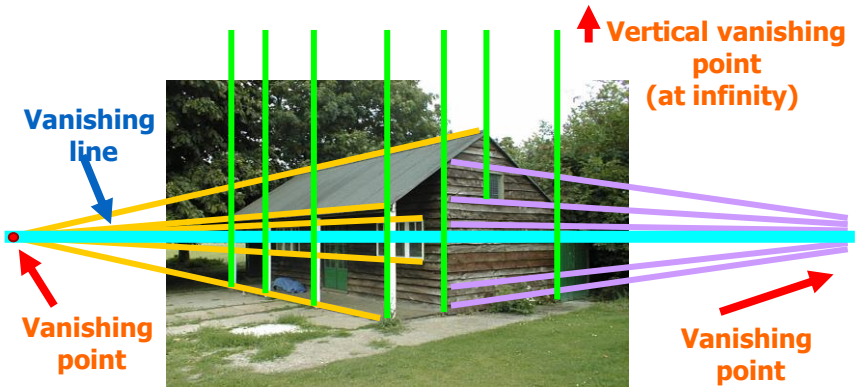
Module 2

Edge detection

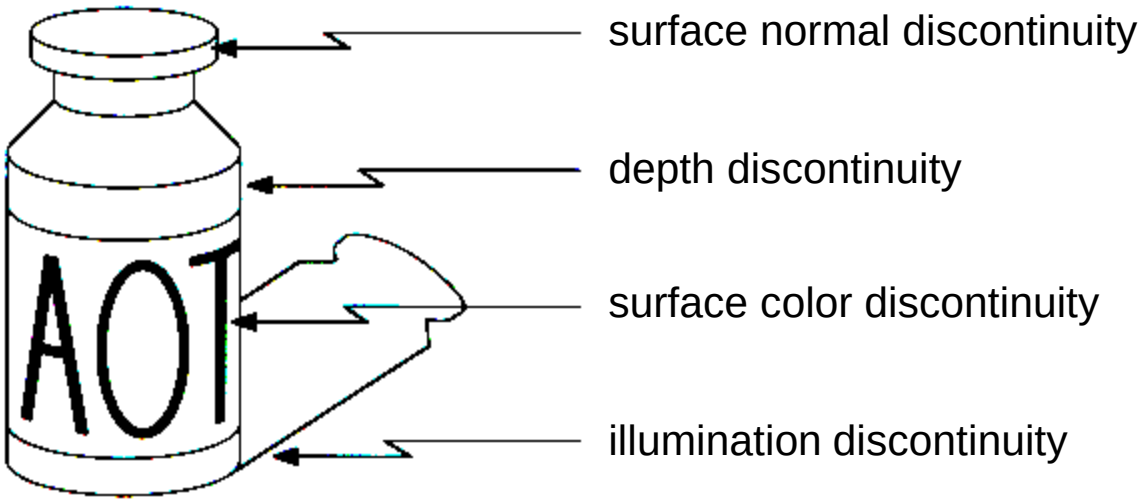
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Recover geometry and viewpoint



Origin of Edges

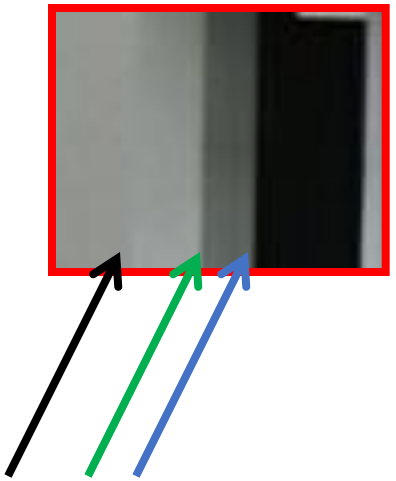


- Edges are caused by a variety of factors

Source: Steve Seitz

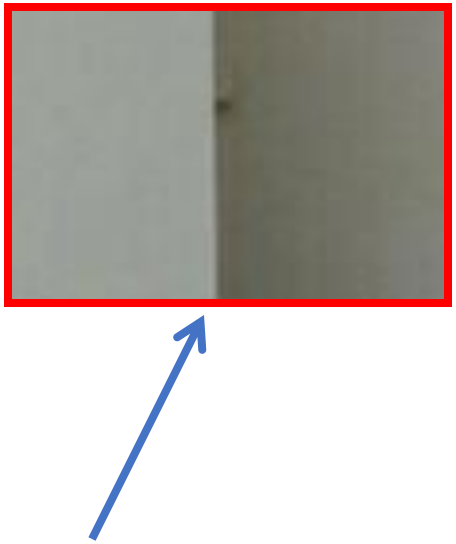
Adapted from James Hays slides

Closeup of edges



Source: D. Hoiem

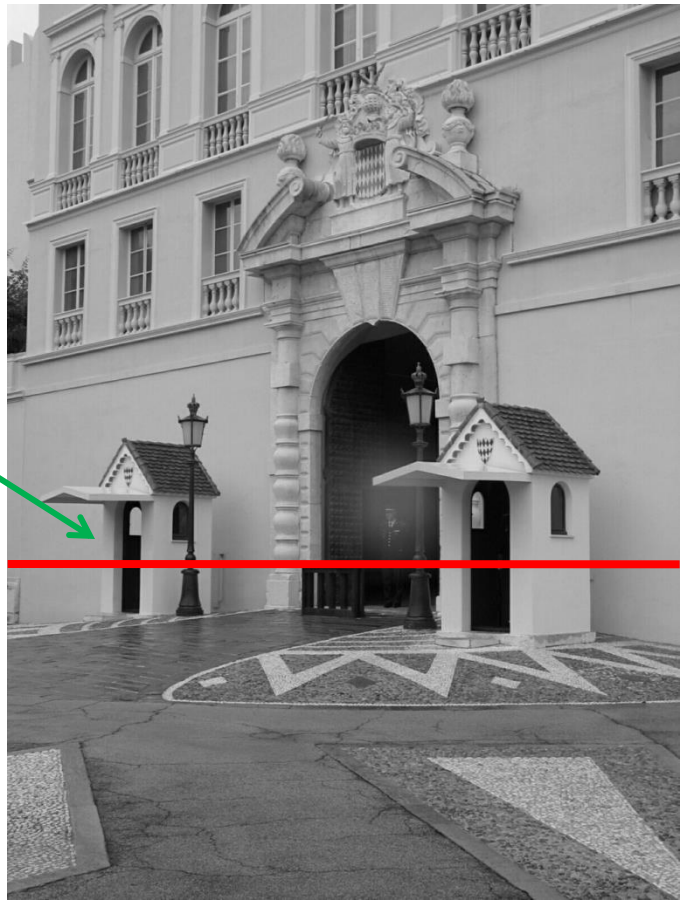
Closeup of edges

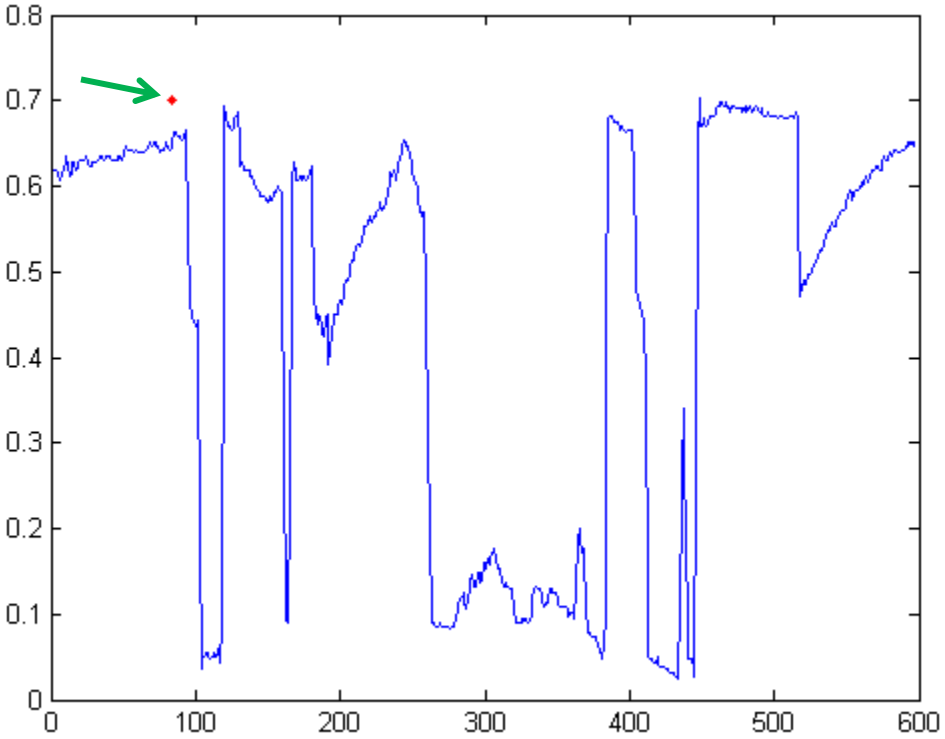


Source: D. Hoiem

Adapted from James Hays slides

Intensity profile





Source: D. Hoiem

Perception of Edges in Humans

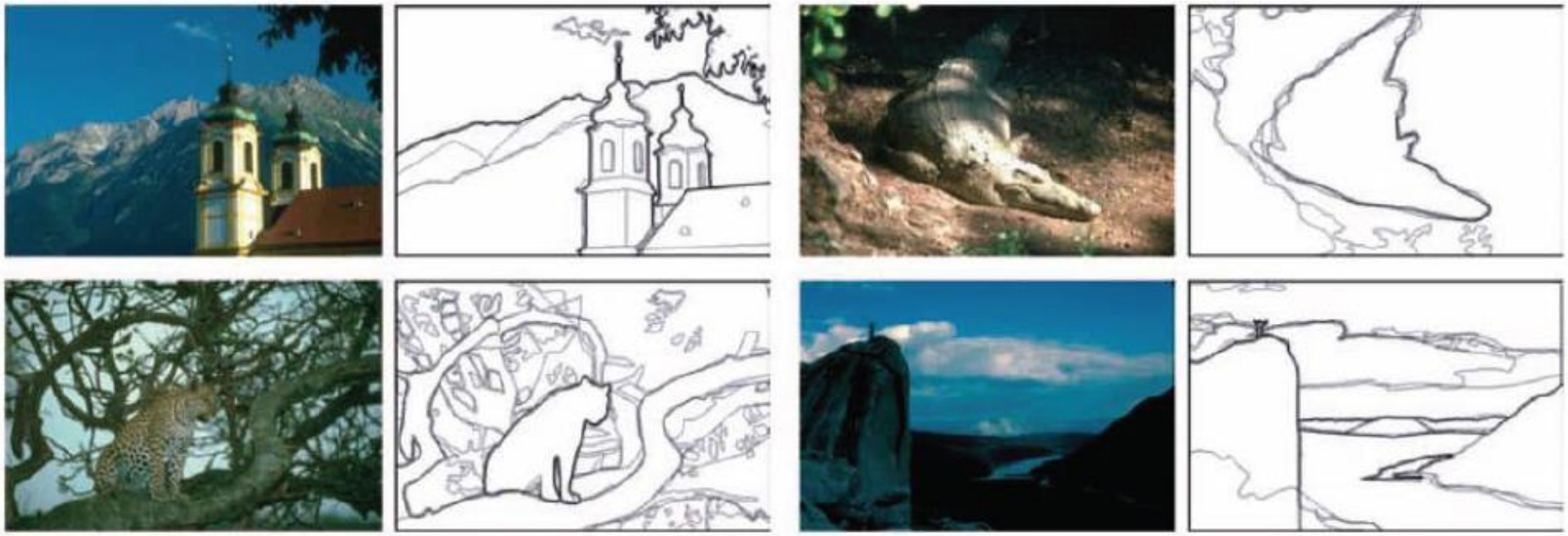
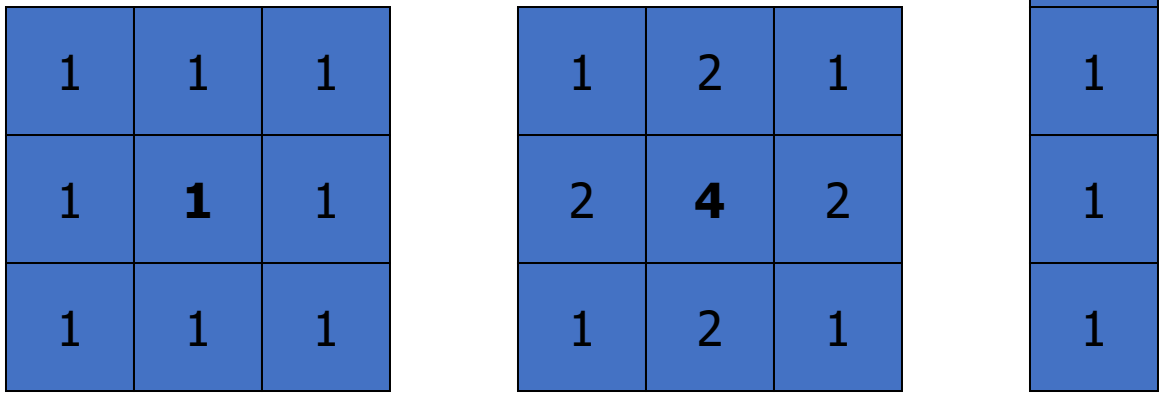


Figure 4.31 Human boundary detection (Martin, Fowlkes, and Malik 2004) © 2004 IEEE. The darkness of the edges corresponds to how many human subjects marked an object boundary at that location.

Applying Masks to Images

- Convolution Operation
- Mask
  - Set of pixel positions and weights
  - Origin of mask



Applying Masks to Images

- $I_1 \otimes \text{mask} = I_2$
- Convention: I_2 is the same size as I_1
- Mask Application:
  - For each pixel
  - Place mask origin on top of pixel
  - Multiply each weight with pixel under it
  - Sum the result and put in location of the pixel

Applying Masks to Images

40	40	40	80	80	80
40	40	40	80	80	80
40	1/9 40	1/9 40	1/9 80	80	80
40	1/9 40	1/9 40	1/9 80	80	80
40	1/9 40	1/9 40	1/9 80	80	80
40	40	40	80	80	80

1/9 x

1	1	1
1	1	1
1	1	1

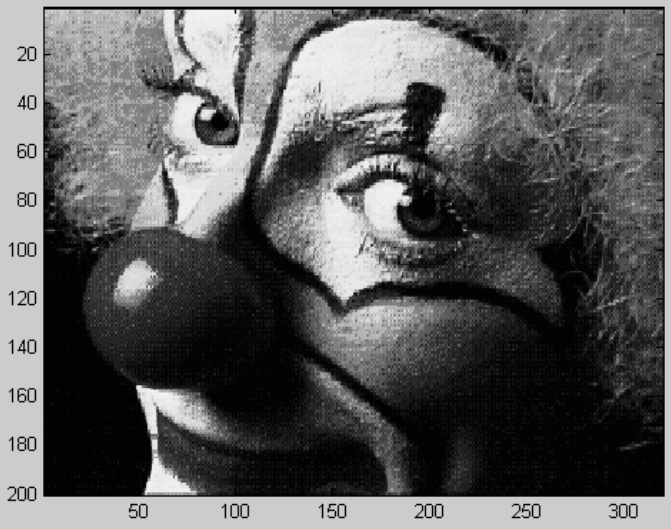
6*(1/9*40)+3*(1/9*80) = 53

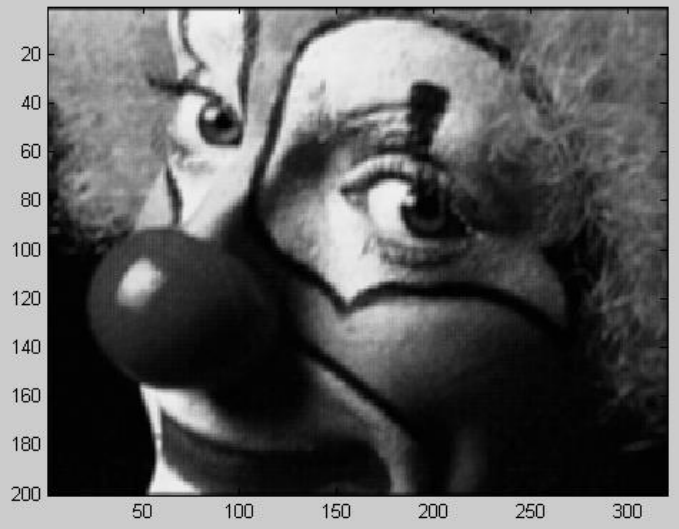
Applying Masks to Images

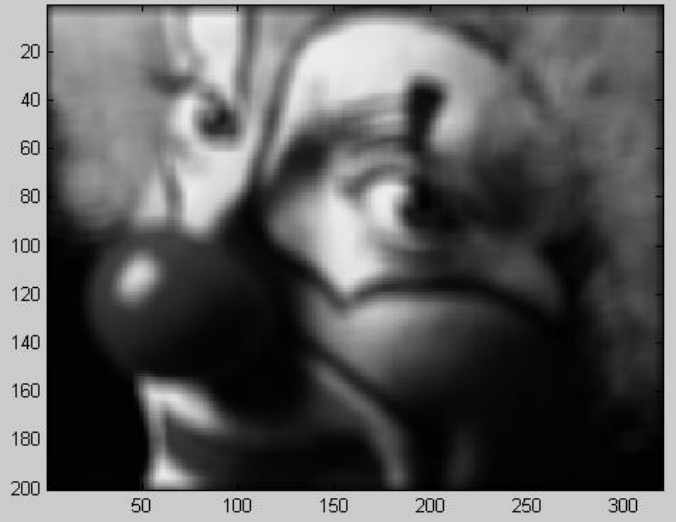
40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80

- Overall effect of this mask?

- Smoothing filter


20
40
60
80
100
120
140
160
180
200
50
100
150
200
250
300


20
40
60
80
100
120
140
160
180
200
50
100
150
200
250
300

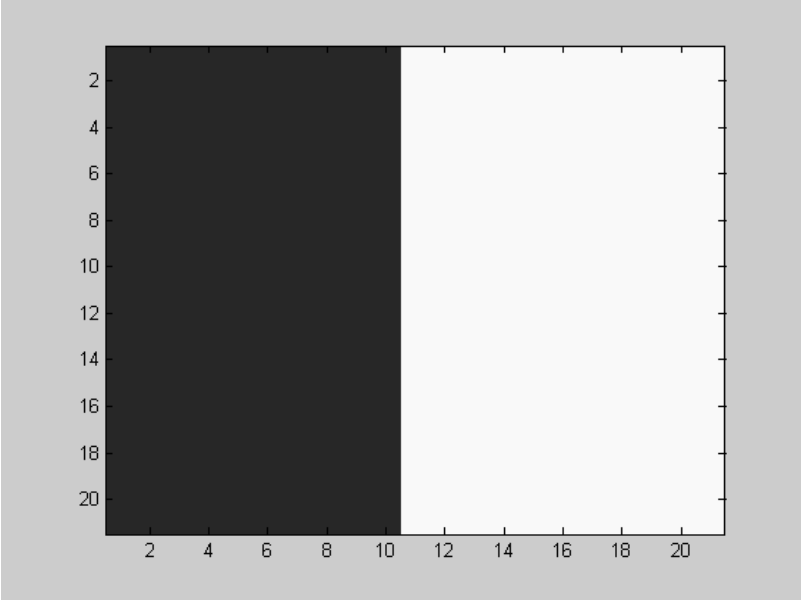

20
40
60
80
100
120
140
160
180
200
50
100
150
200
250
300

What about corner pixels

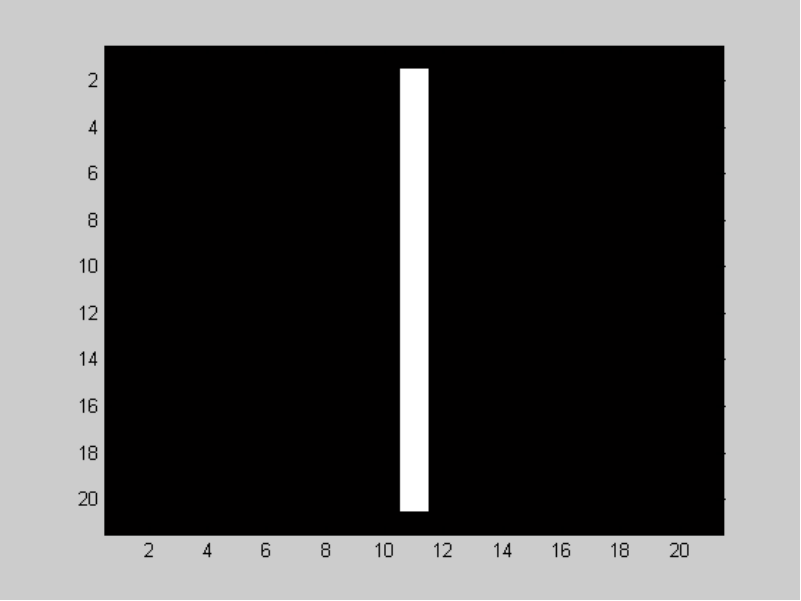
- Expand image with virtual pixels
- Options
  - Fill with a particular value, e.g. zeros
  - Fill with nearest pixel value
- Or just ignore them

Edge Detection

Input



Output



Choice of Mask?

- Should give a zero on smooth output
- Should give a high value on non-smooth regions

-1	0	1
-1	**0**	1
-1	0	1

Original Image, I

$$e = I \otimes \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

abs(e)

Finding Edges

- Edges are locations where intensity variation is high
- OR rate of change of intensity is high
- How do we find rate of change of intensity?
- DIFFERENTIATION

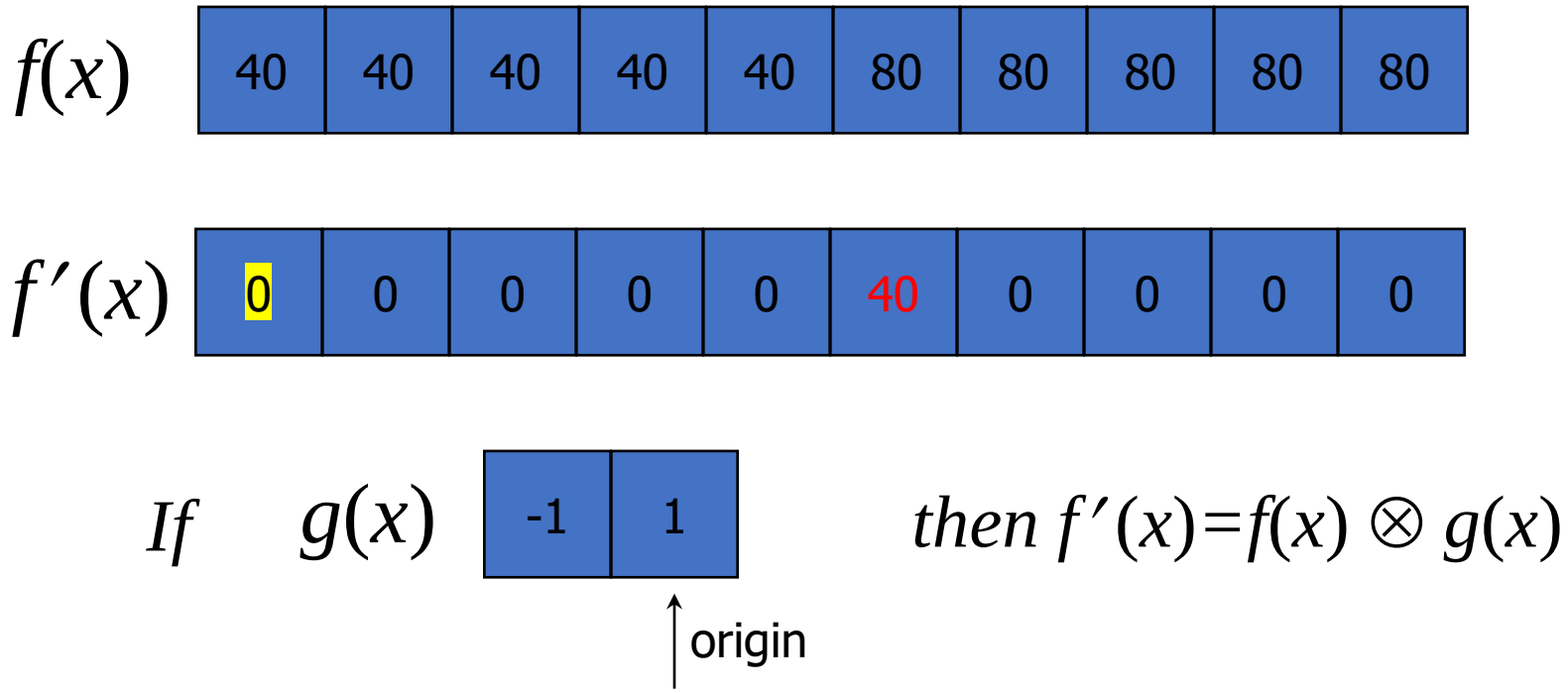
- Continuous form

$$f' = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

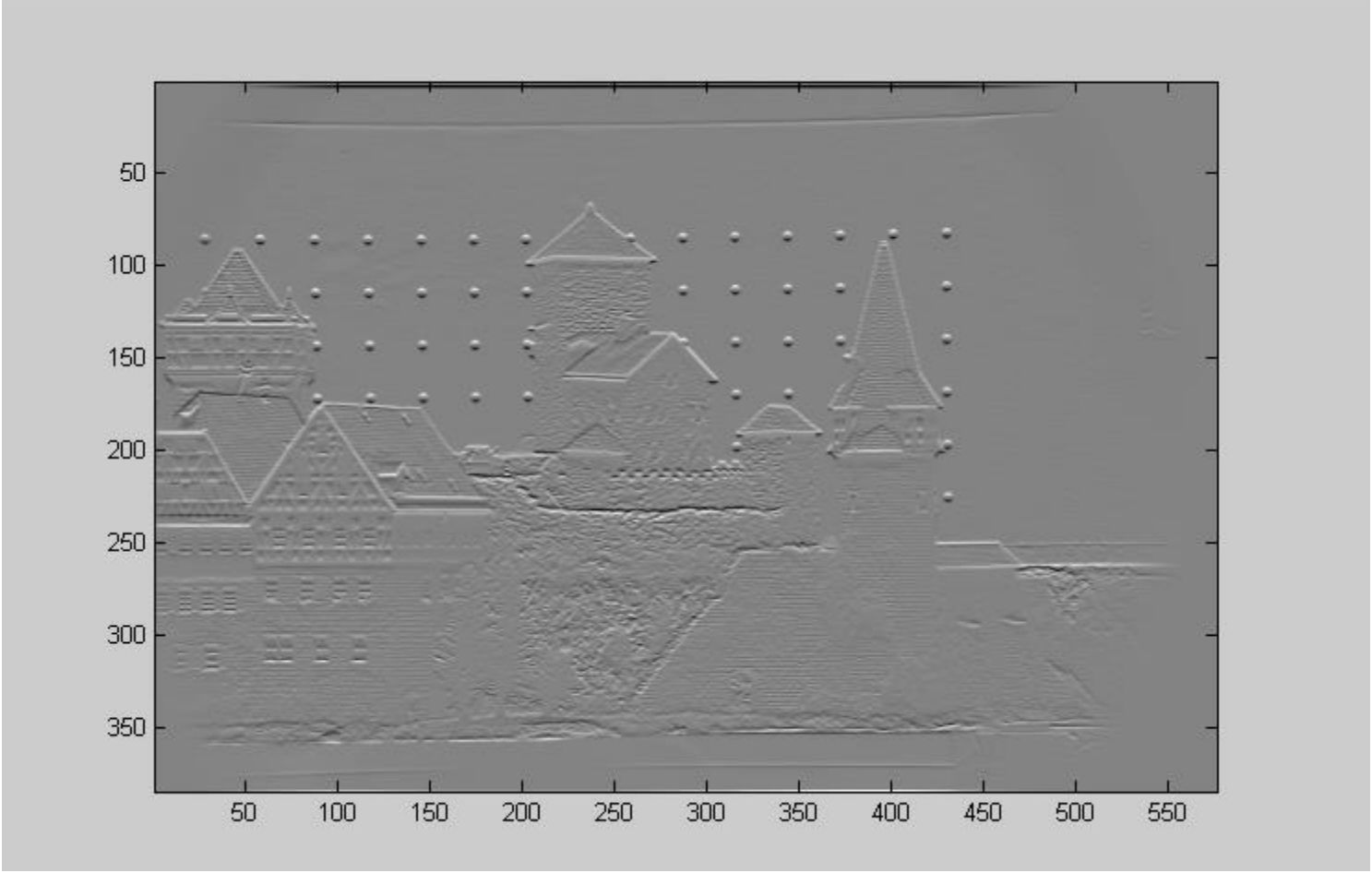
- Discrete form

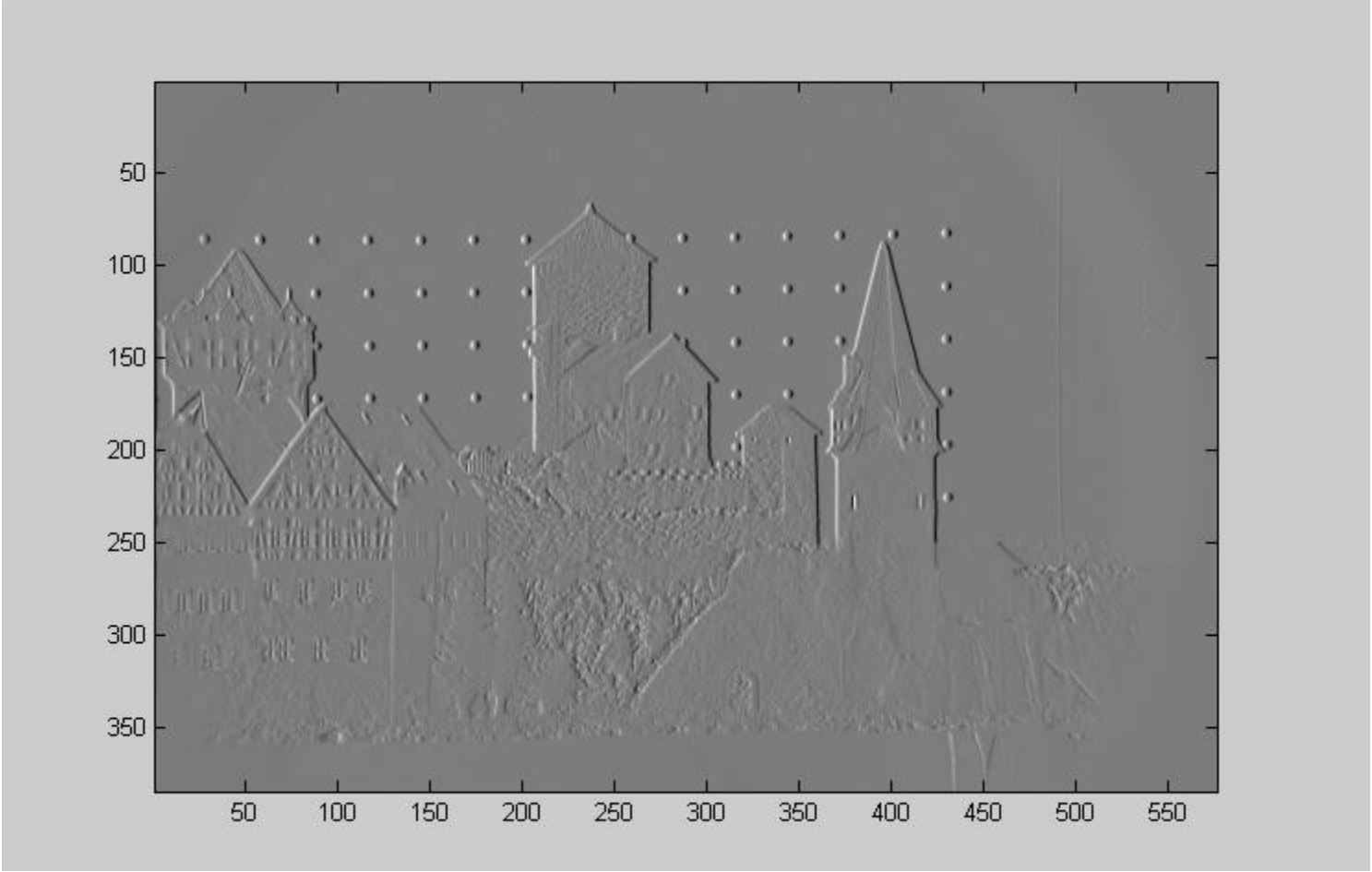
$$f' = \frac{df}{dx} = f(x) - f(x - 1)$$

Discrete Derivatives



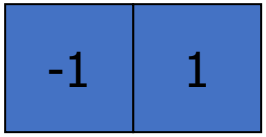

50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
450
500
550

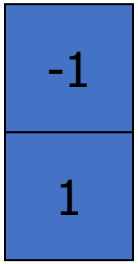

50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
450
500
550


50
100
150
200
250
300
350
50
100
150
200
250
300
350
400
450
500
550

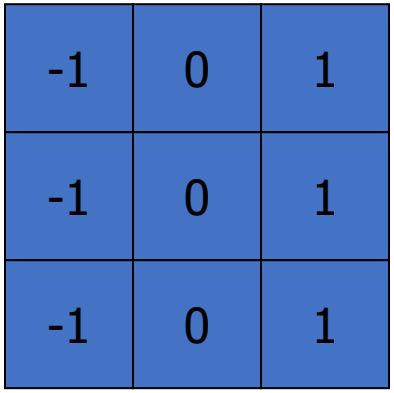
Derivative Masks

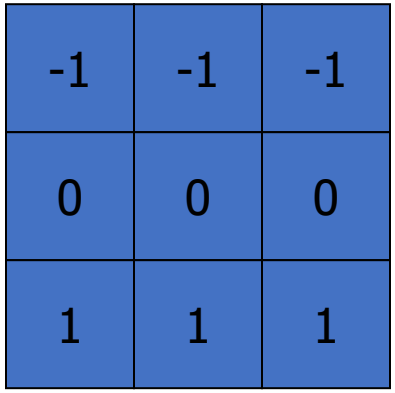
From definition of derivatives





Prewitt Operator





- Sobel Operator

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

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

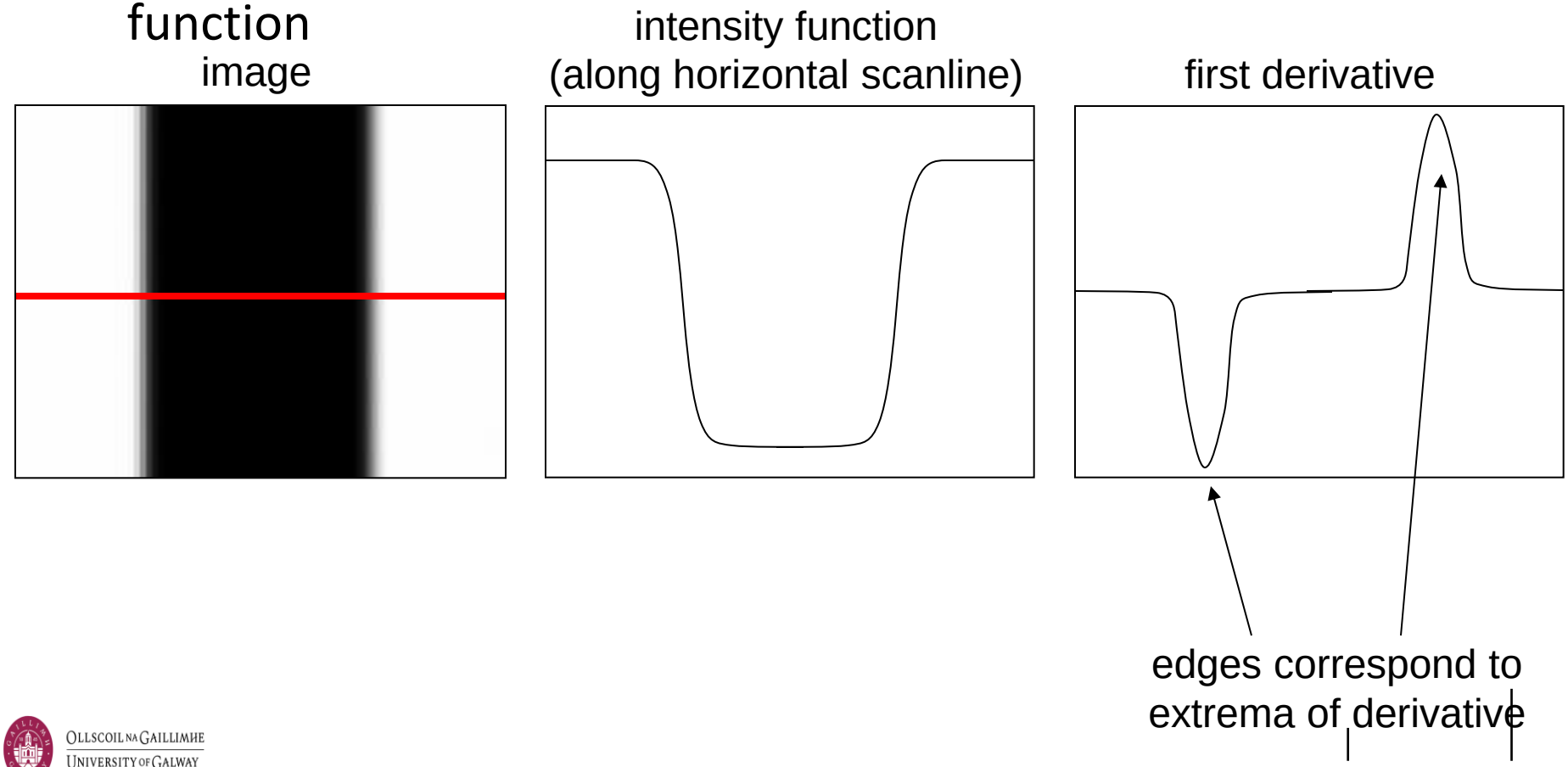
- Robert's Operator

1	0
0	-1

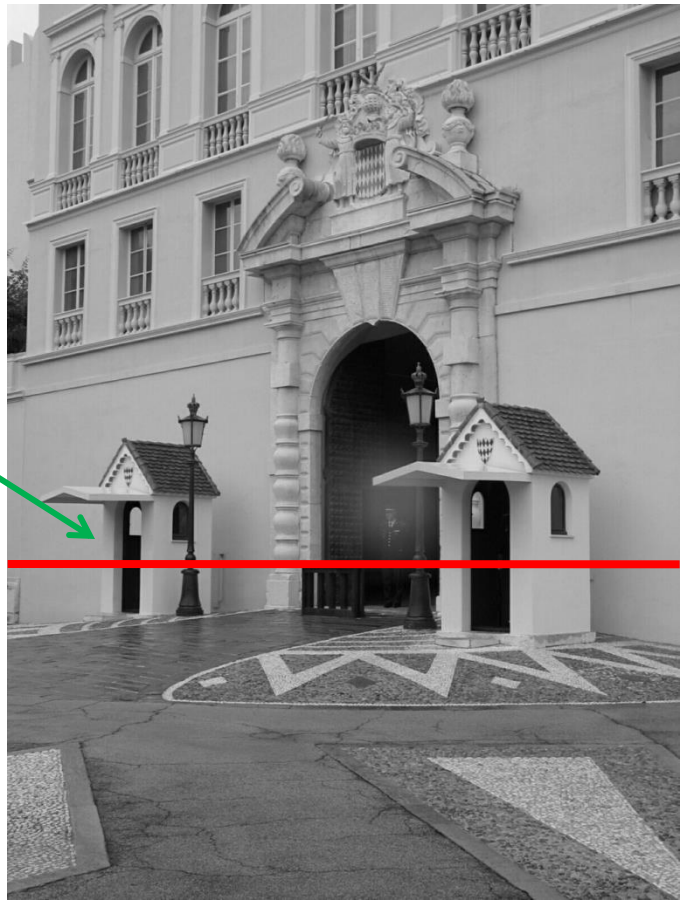
0	1
-1	0

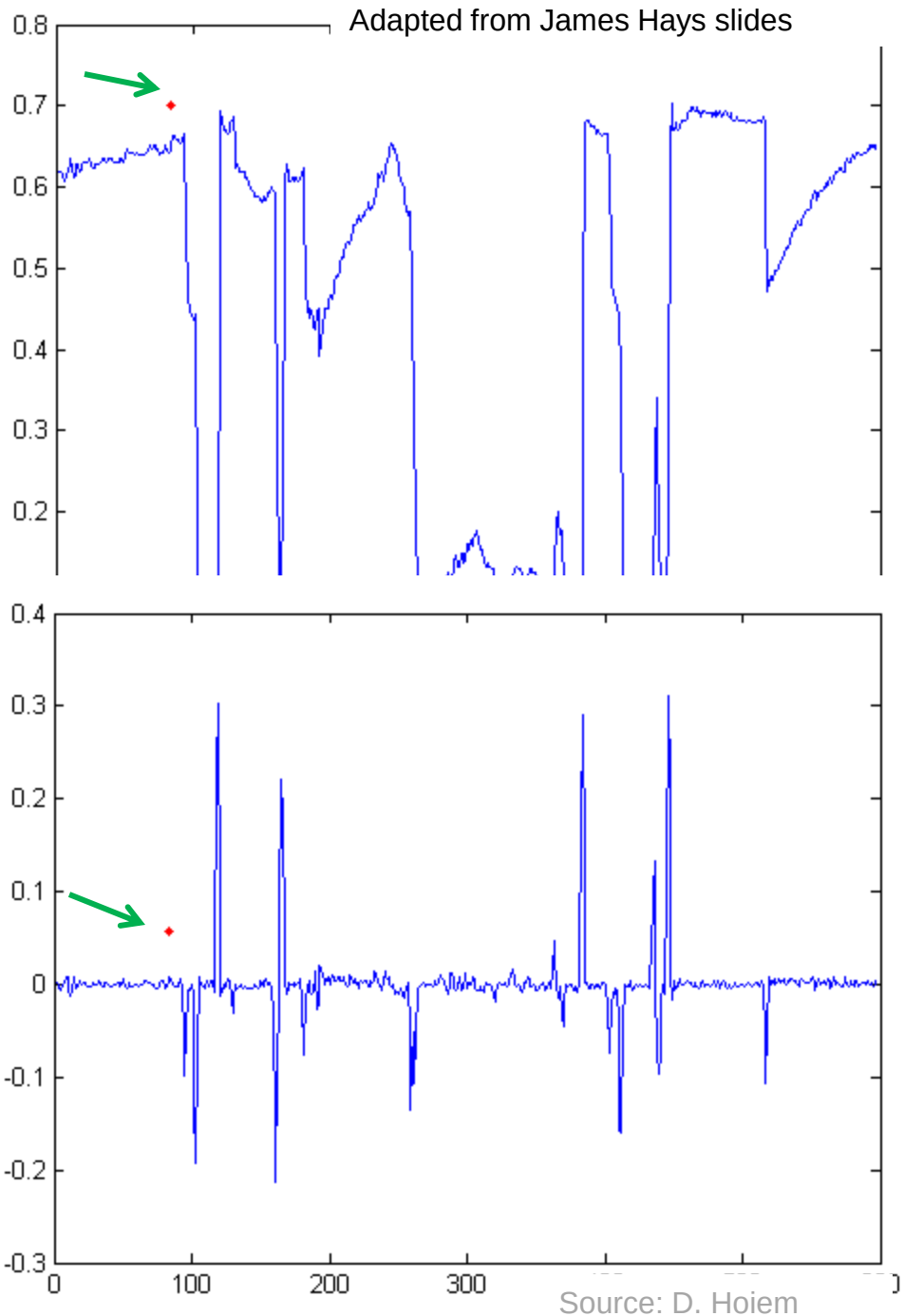
Characterizing edges

- An edge is a place of rapid change in the image intensity function



Intensity profile

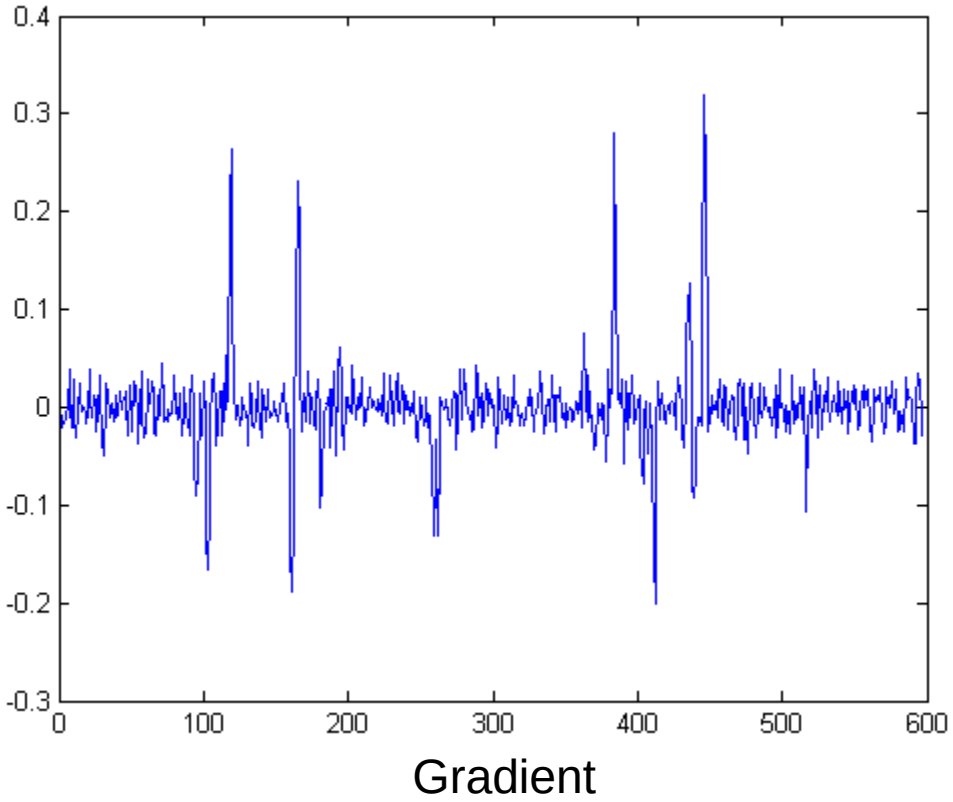




Source: D. Hoiem

Adapted from James Hays slides

With a little Gaussian noise

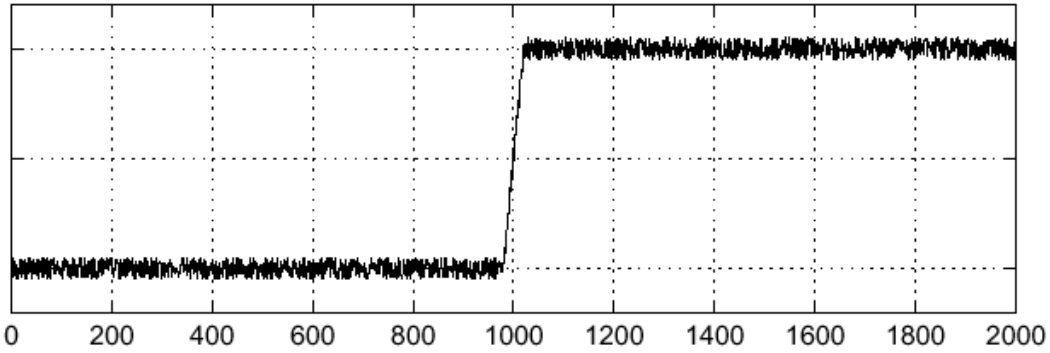


Source: D. Hoiem

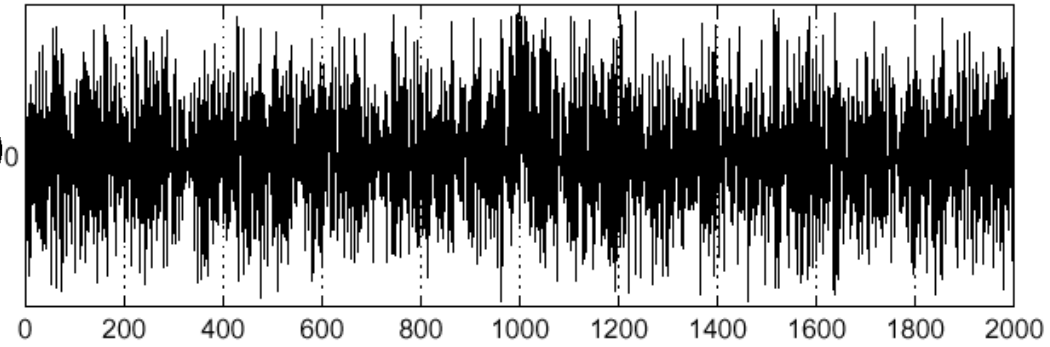
Effects of noise

- 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?

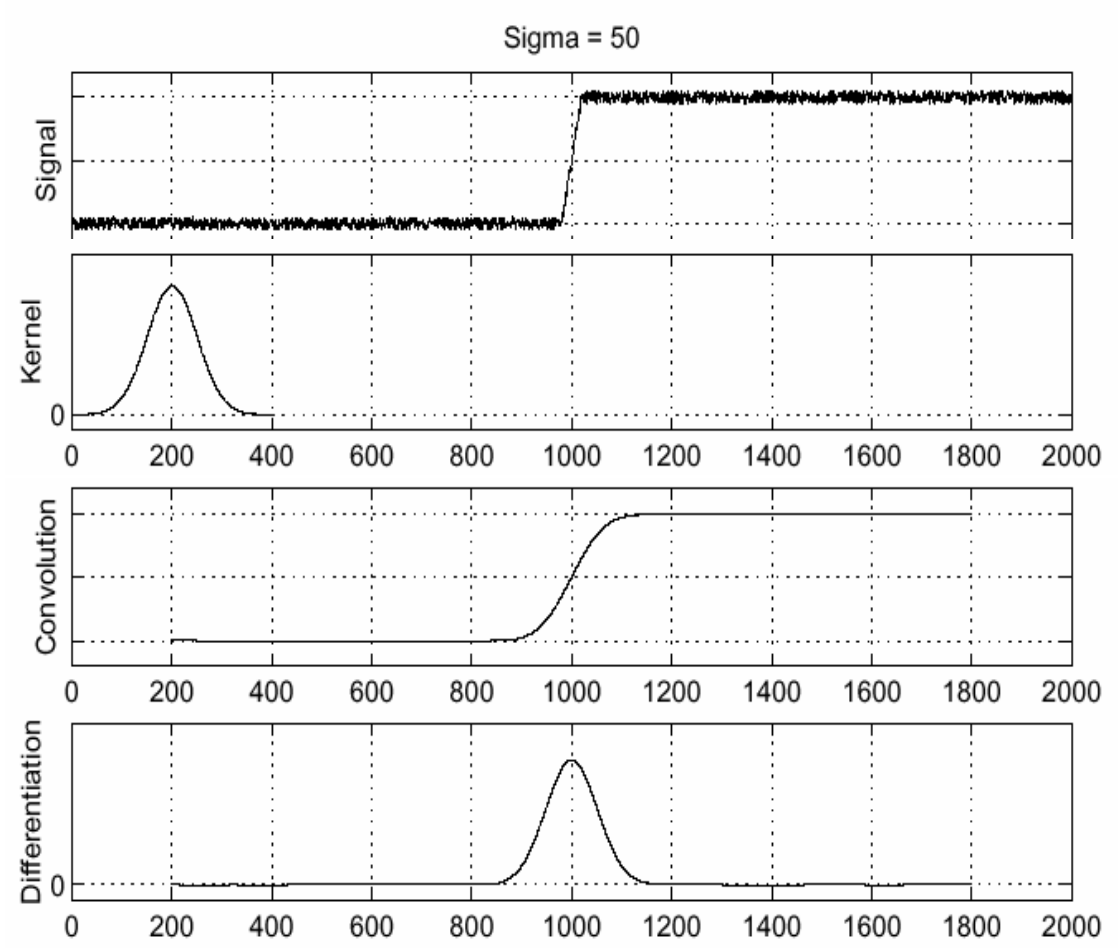
Source: S. Seitz

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

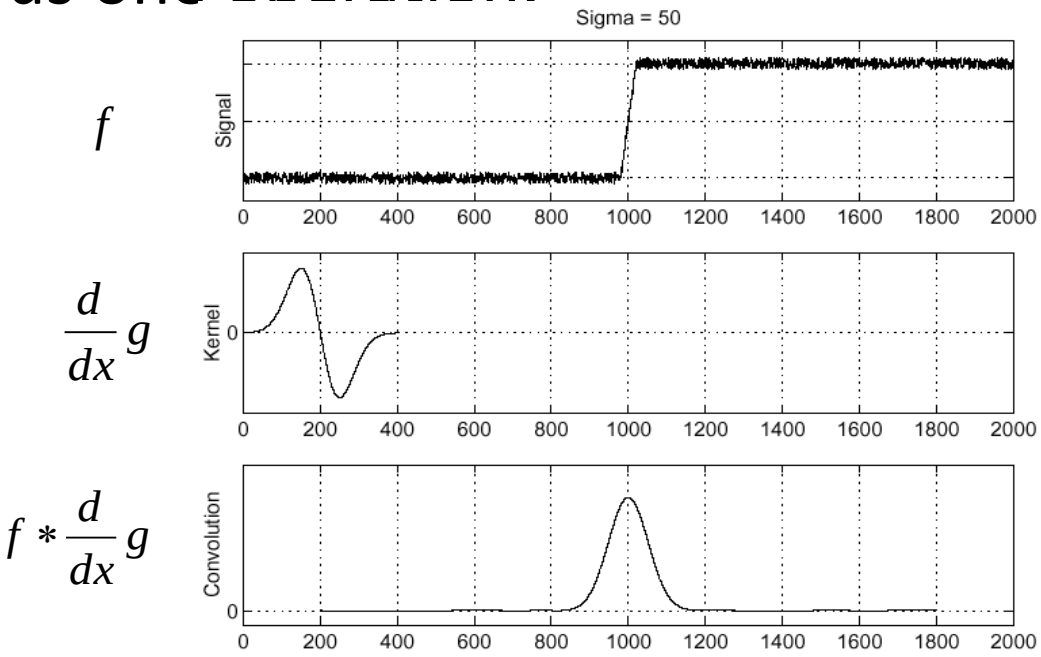
Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

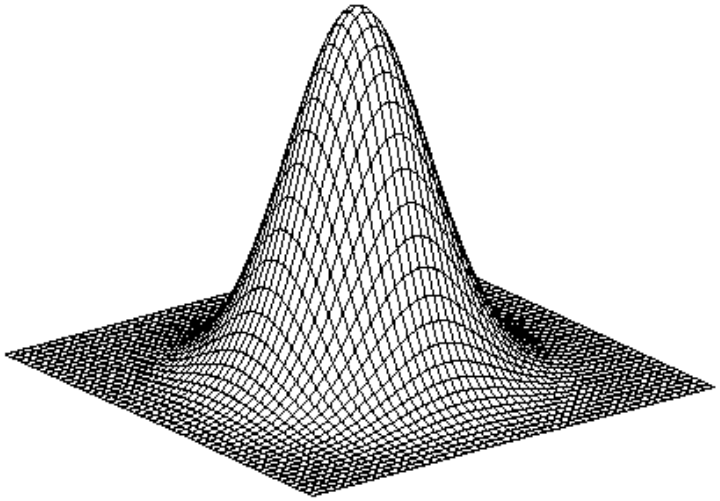
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

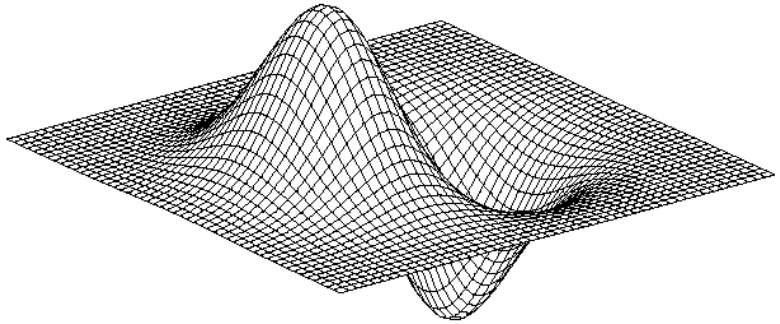


Source: S. Seitz

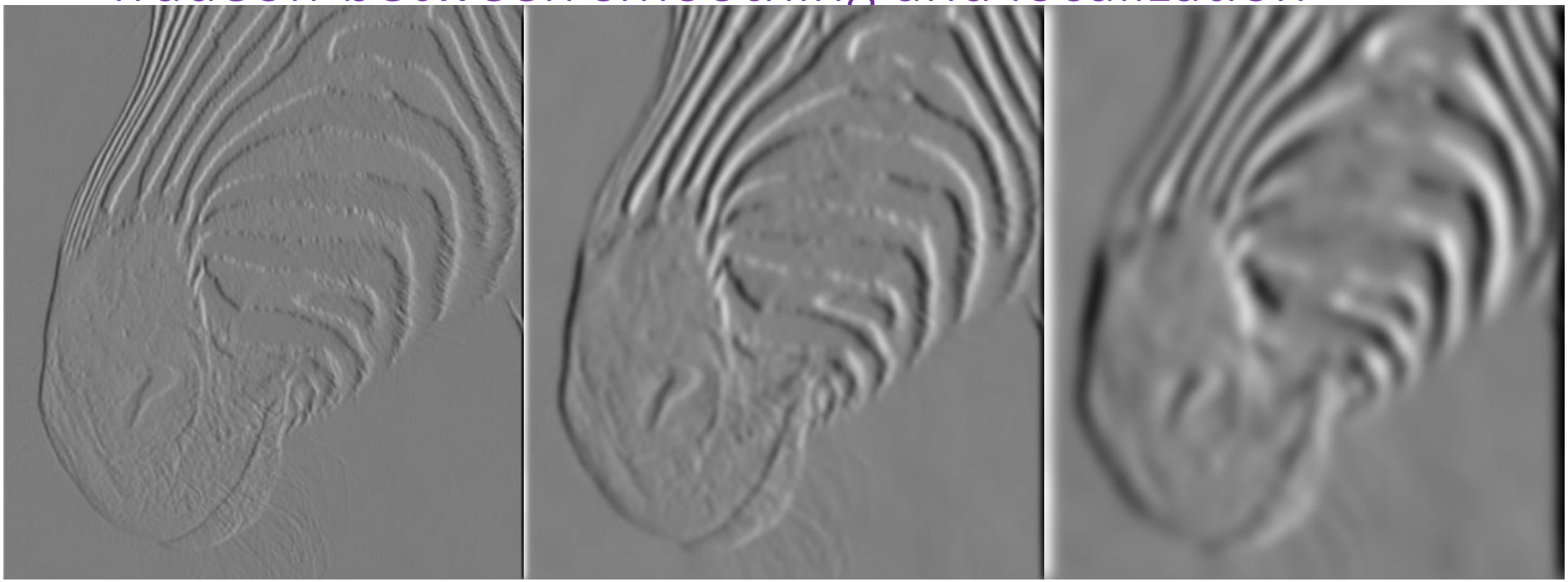
Derivative of Gaussian filter



* [1 -1] =



Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

Implementation issues

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

Application of 2-D Masks

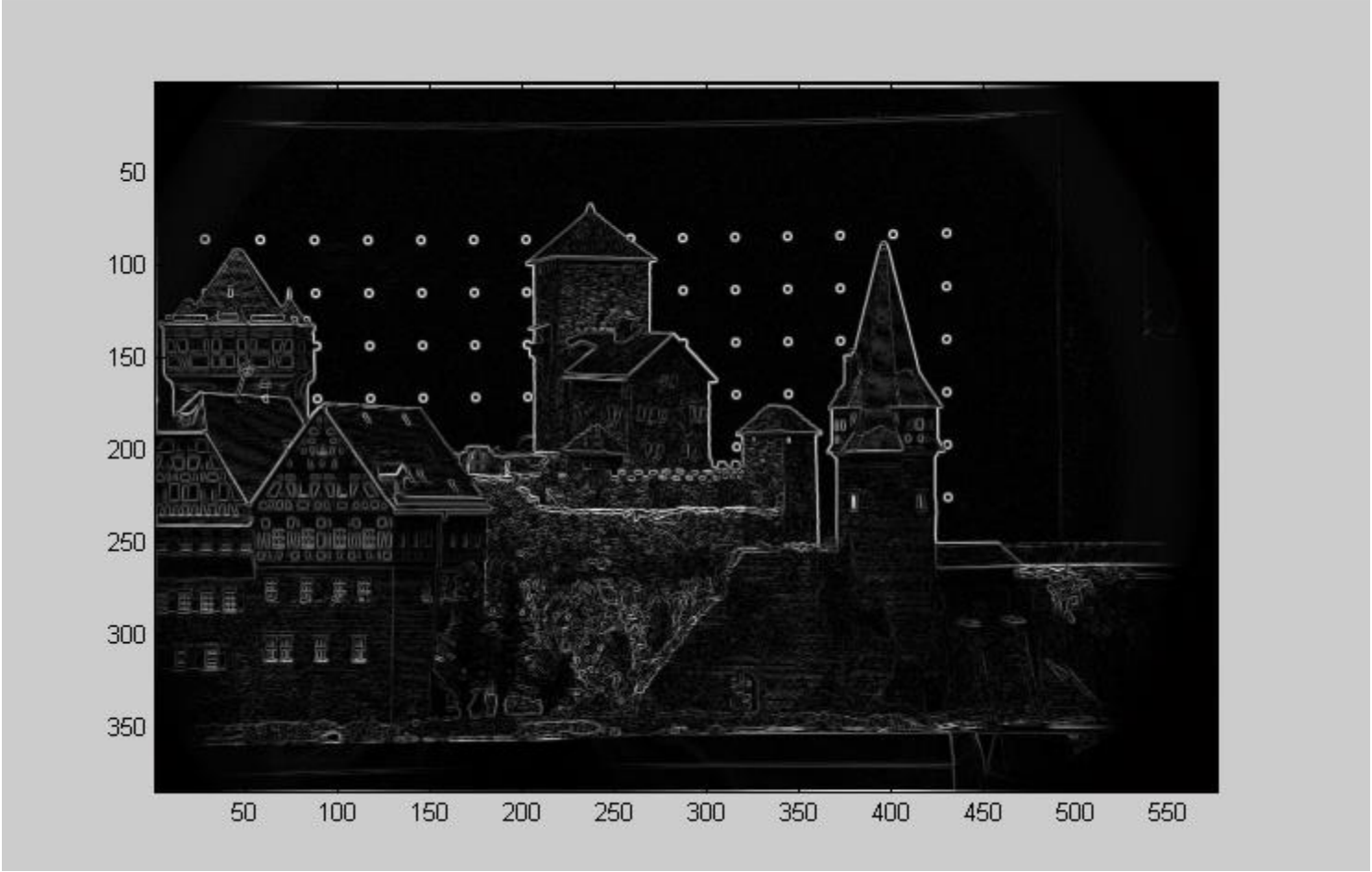
- If f_x is derivative in x-direction, f_y is derivative in y-direction
- Gradient Magnitude

$$M = \sqrt{f_x^2 + f_y^2}$$

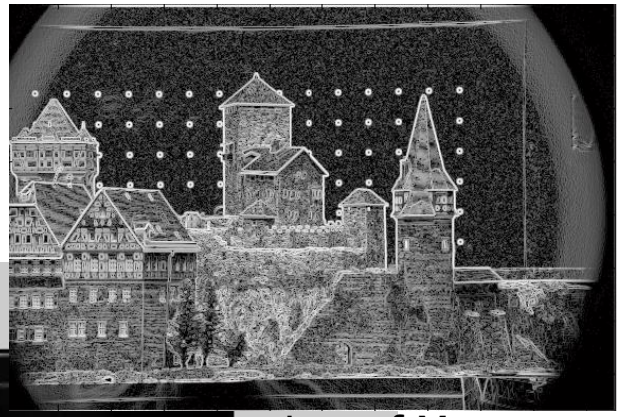
- Gradient Direction

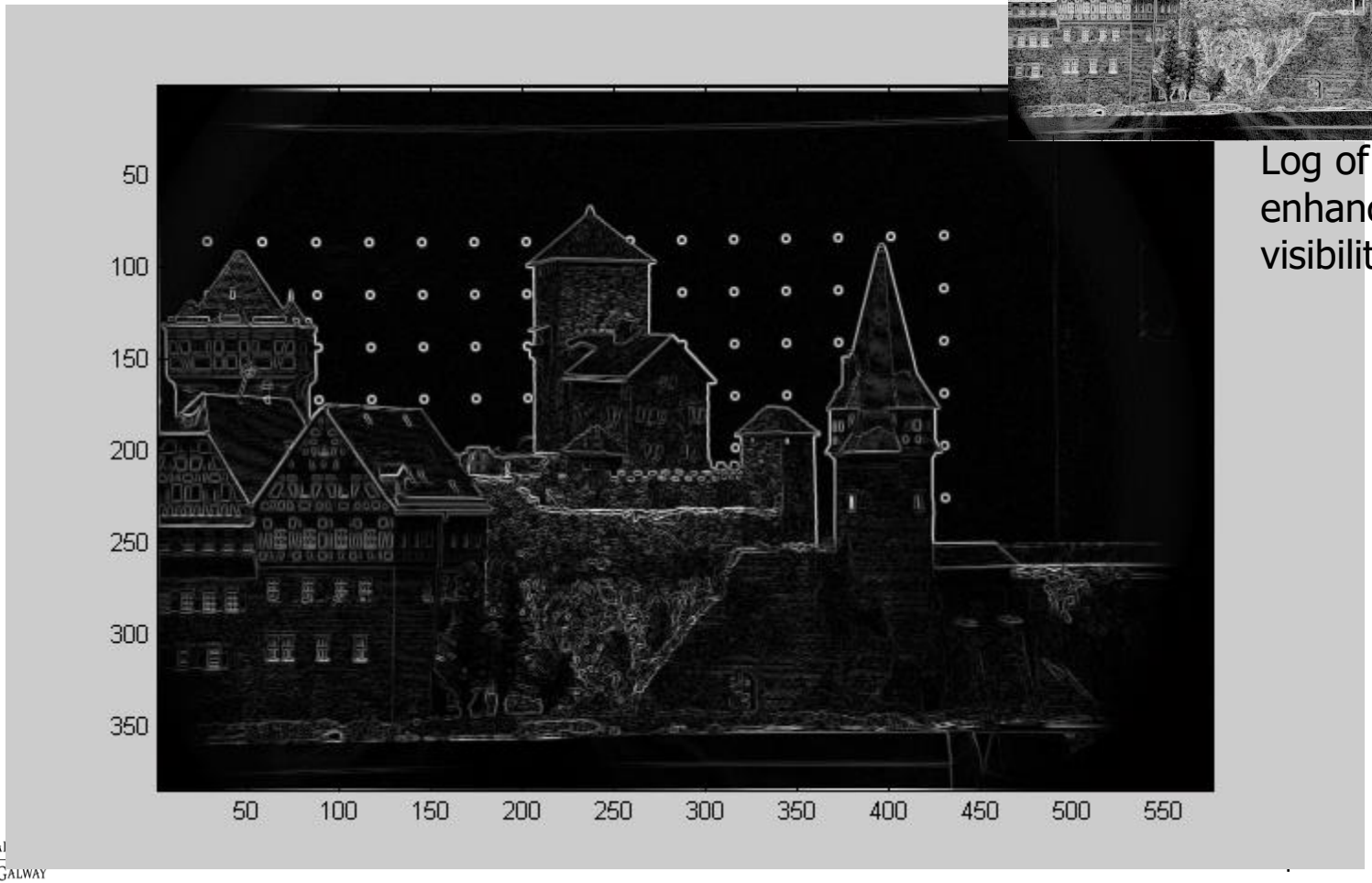
$$\theta = \arctan \frac{f_y}{f_x}$$

Gradient Magnitude



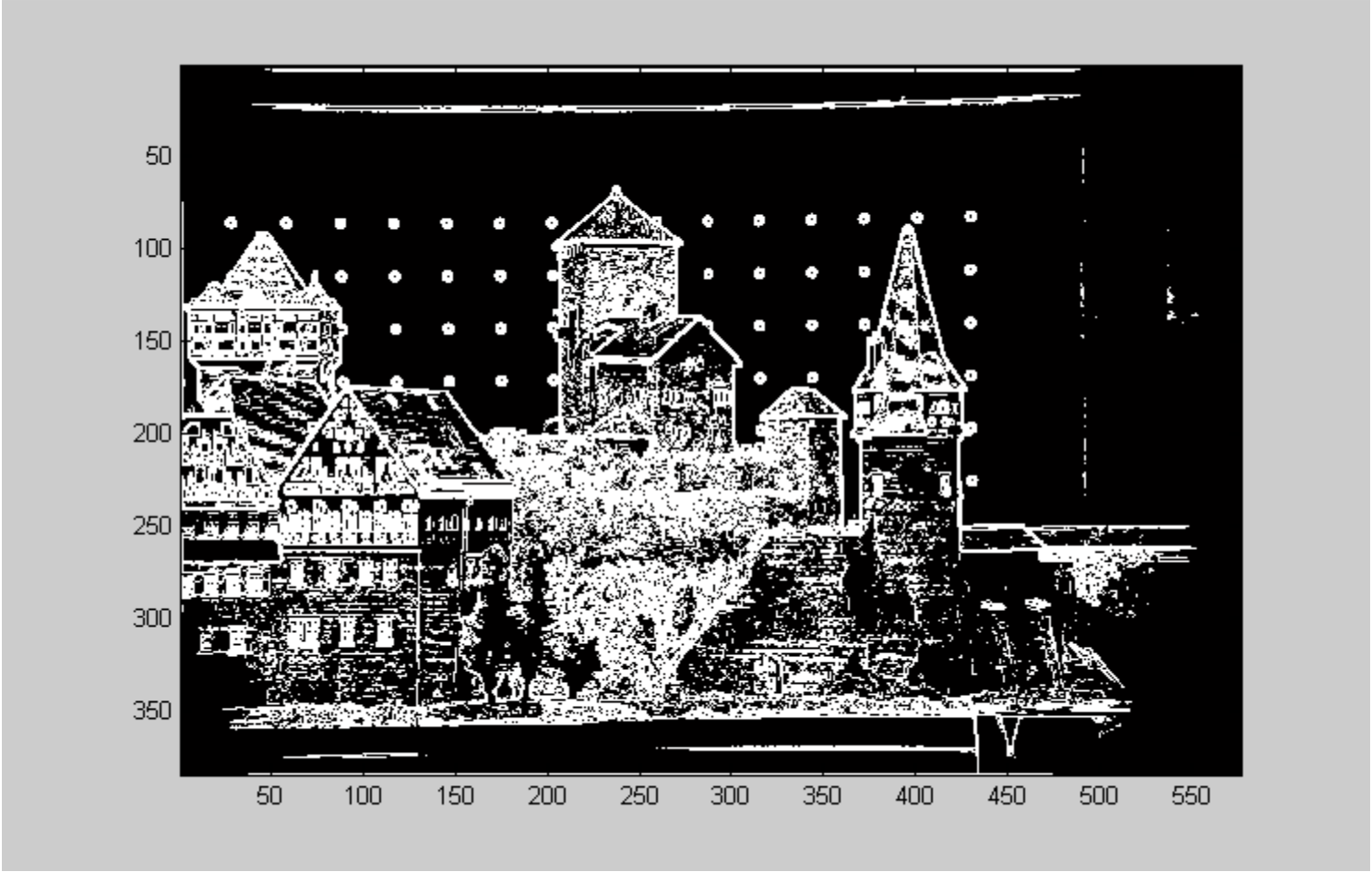
Gradient Magnitude





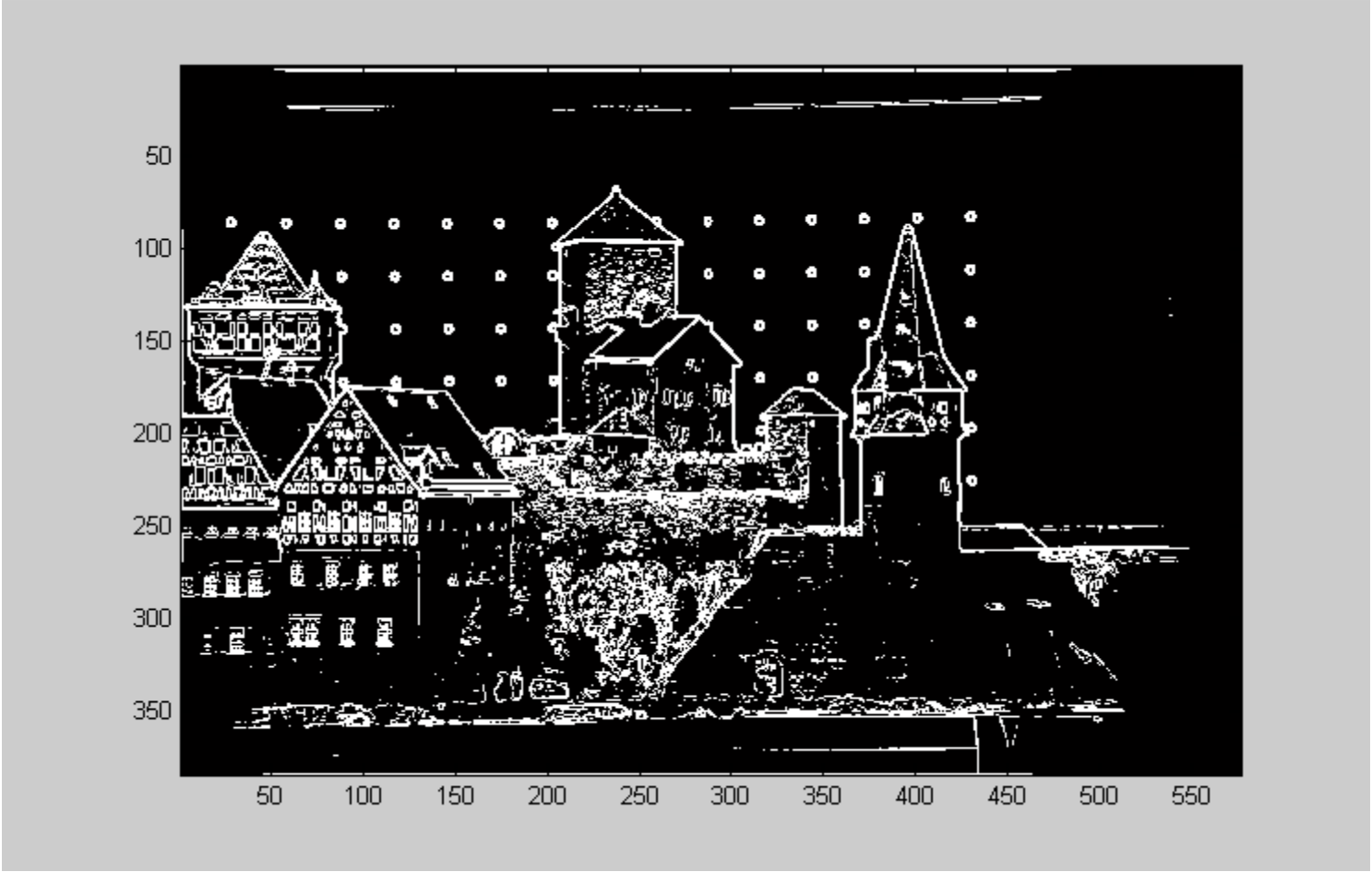
Log of M to enhance visibility

Gradient Magnitude – Thresholding T==10



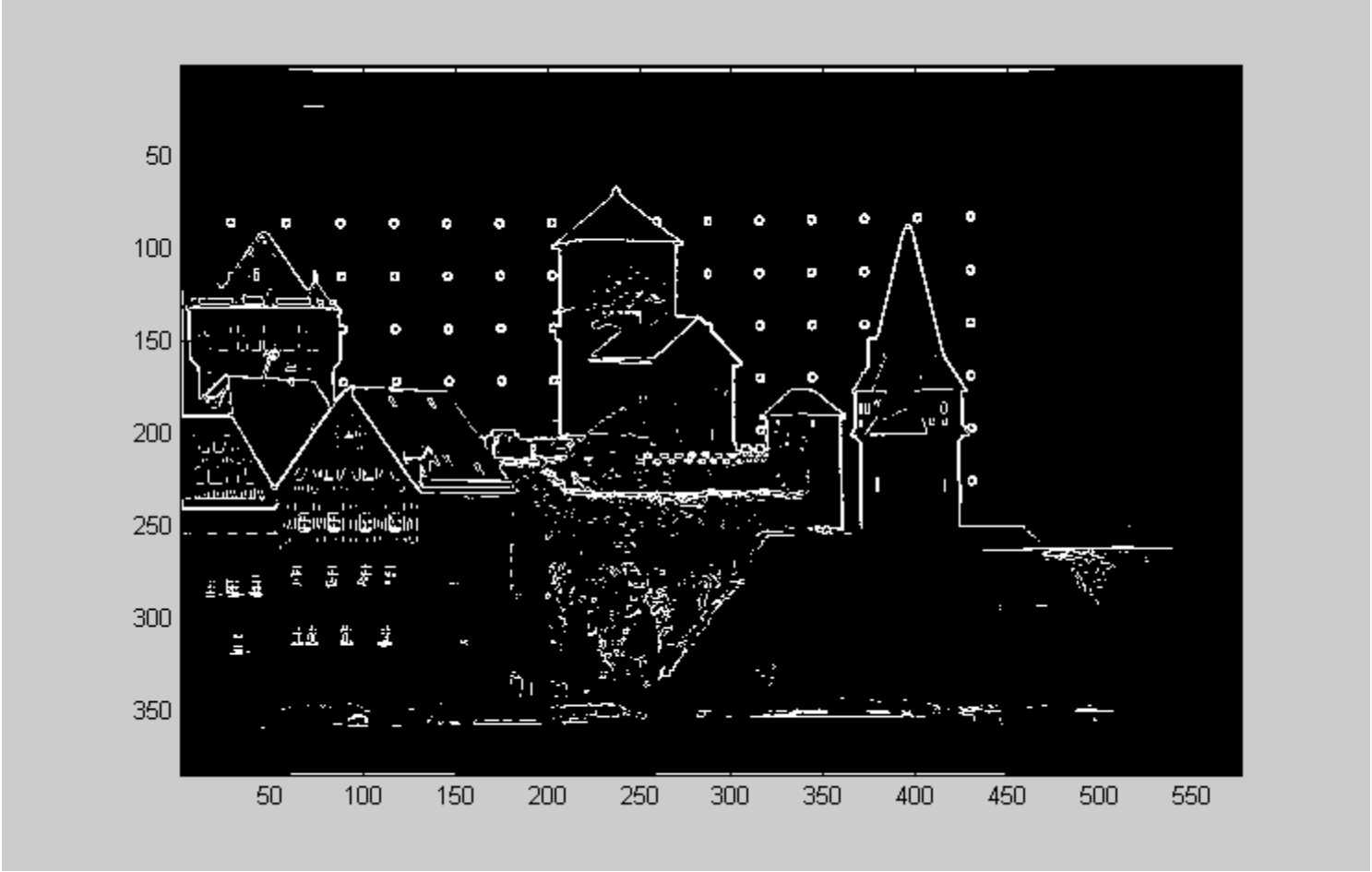
T = 10

Gradient Magnitude – Thresholding T==20



T = 20

Gradient Magnitude – Thresholding T==40



T = 40

What about Gradient Direction?

- Gradient Direction is always perpendicular to edge
- Direction of most change of gray levels
- Thick edges can be eliminated using gradient direction
- Weak edges also captured in this manner

Image Gradient

- Gradient vector

$$\nabla I(\mathbf{x}) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$

where

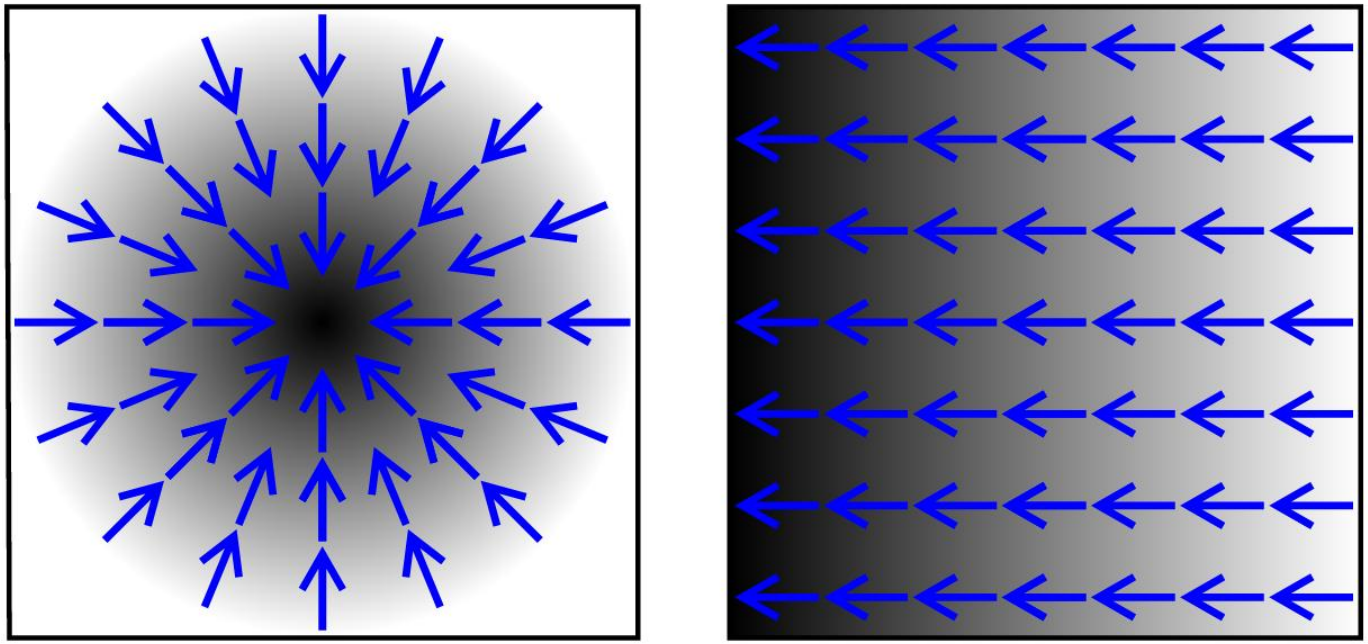
$$\frac{\partial I(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x}$$

$$\frac{\partial I(x,y)}{\partial y} = \lim_{\Delta y \to 0} \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y}$$

- Gradient vector points in the direction of maximum increase

Image Gradient

- Example



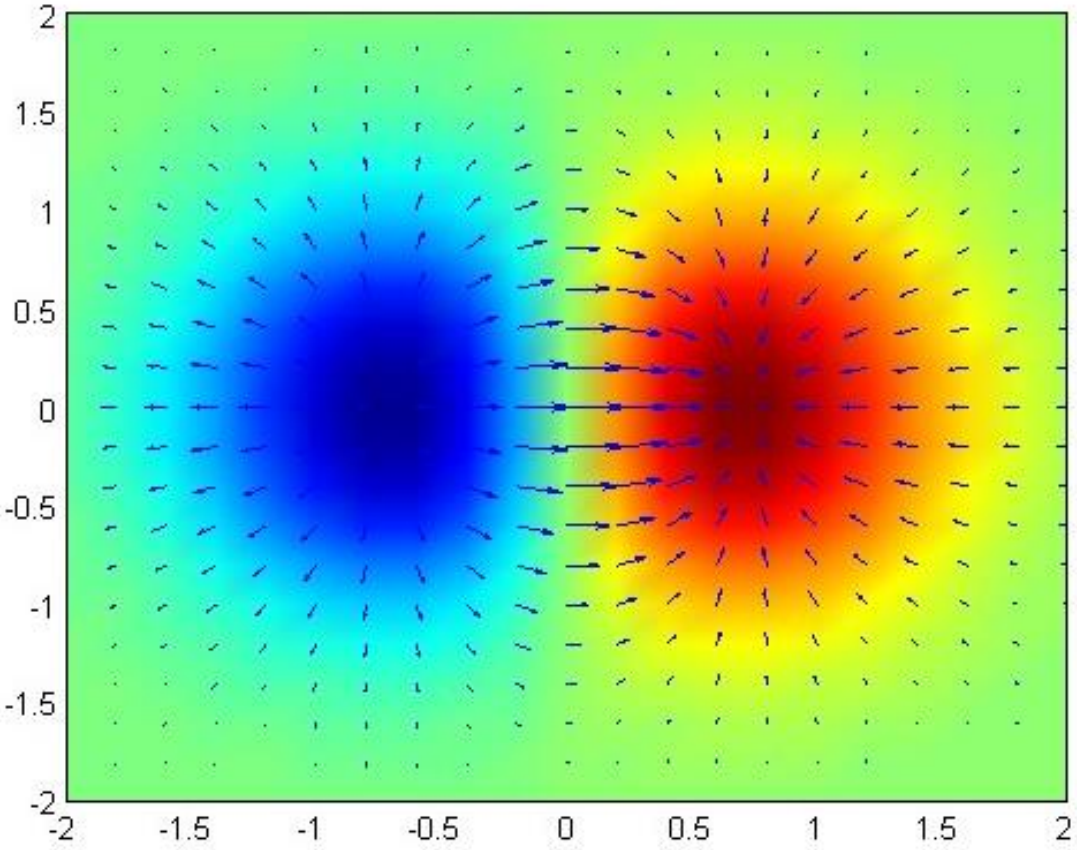
(*Black represents higher value of function*)

https://en.wikipedia.org/wiki/Gradient

Image Gradient

$$f(x,y) = xe^{-(x^2+y^2)}$$

- Example

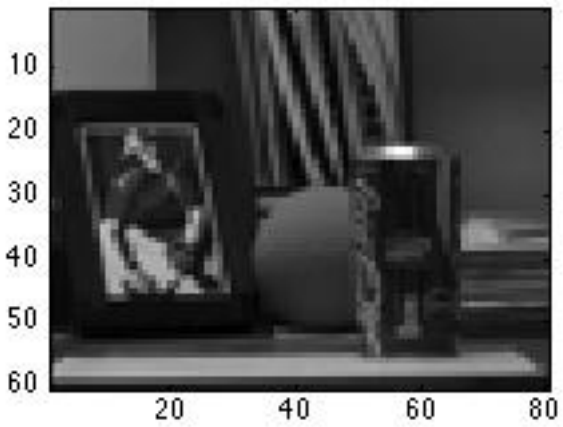


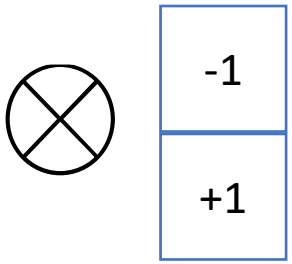
https://en.wikipedia.org/wiki/Gradient

Image Gradient

- Example

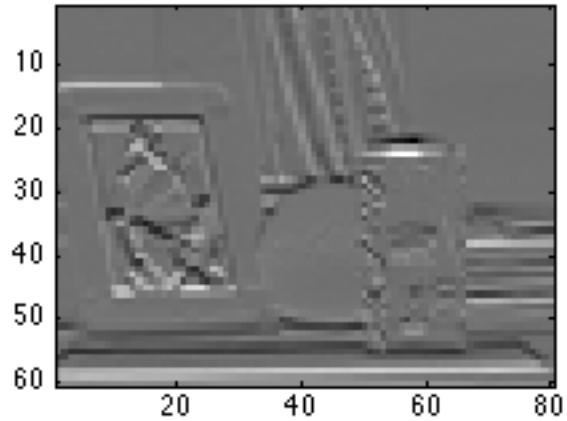
$I(x,y)$

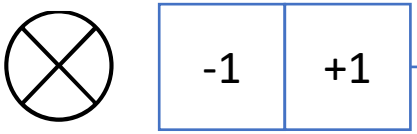




$I(x+1,y) - I(x,y)$

$\frac{\partial I(x,y)}{\partial x}$





$I(x,y+1) - I(x,y)$

$\frac{\partial I(x,y)}{\partial y}$

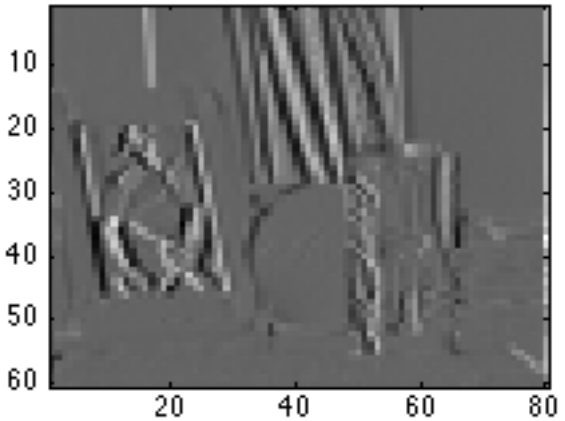
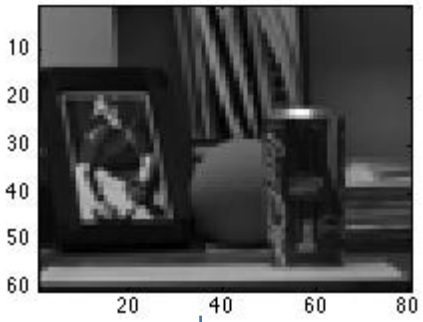


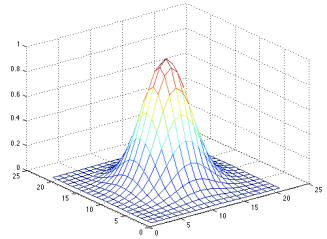
Image Gradient

- Derivatives are sensitive to noise
  so we can smooth before differentiating

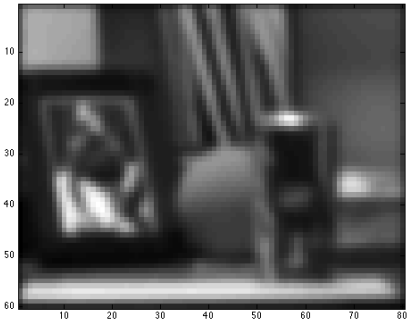
$I(x,y)$



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



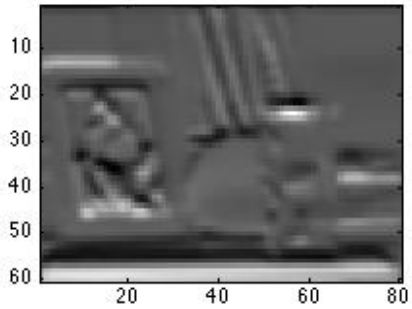
$I(x,y) \otimes G(x,y)$



$\nabla \left[I(\mathbf{x}) \otimes G_\sigma(\mathbf{x}) \right]$

$\frac{\partial (I(x,y) \otimes G(x,y))}{\partial x}$

-1
+1



$\frac{\partial (I(x,y) \otimes G(x,y))}{\partial y}$

-1 +1

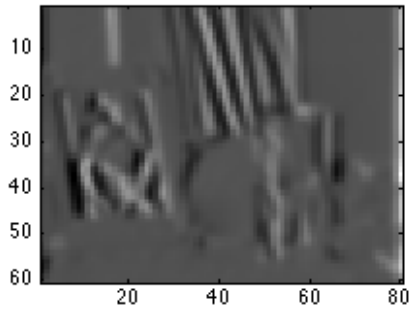
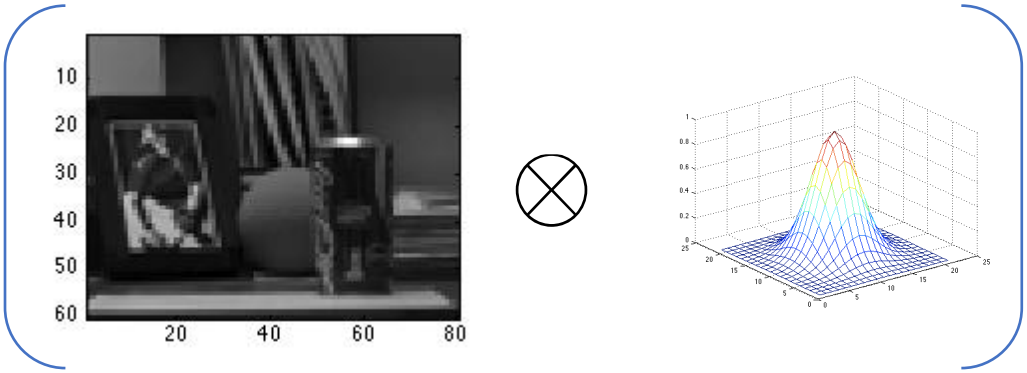


Image Gradient

- But convolution is associative operation.

$$\nabla \left[I(x,y) \right) \otimes G_\sigma(x,y)]$$



is equivalent to

$$I(x,y) \otimes \nabla G_\sigma(x,y)$$

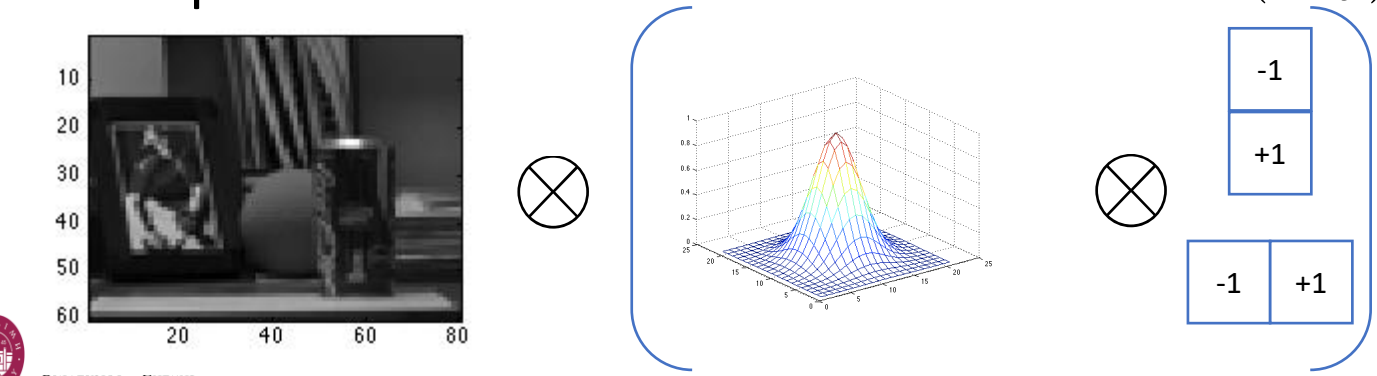
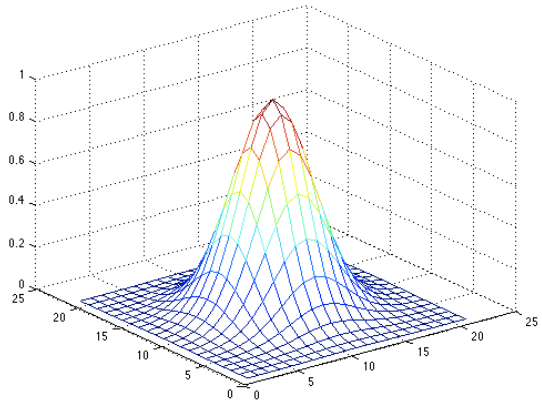


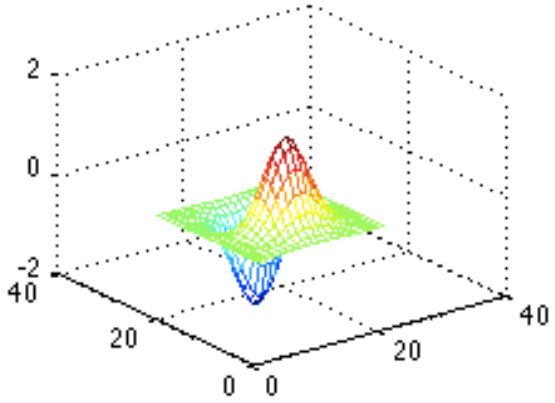
Image Gradient

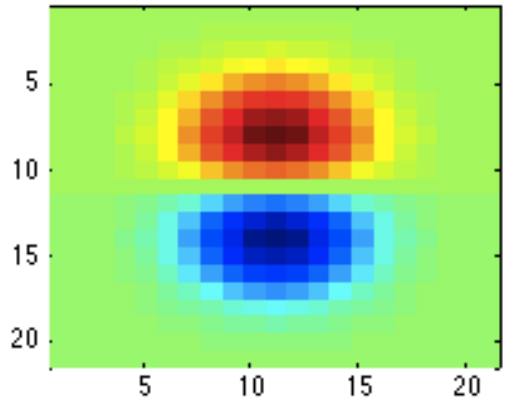
$$G(x,y) = \frac{1}{2\pi\sigma} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$



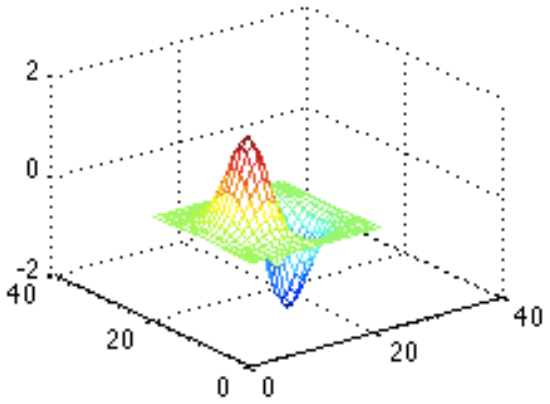
$$\nabla G(x,y) = \begin{bmatrix} \frac{\partial G(x,y)}{\partial x} \\ \frac{\partial G(x,y)}{\partial y} \end{bmatrix}$$

$$\frac{\partial G(x,y)}{\partial x} = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





$$\frac{\partial G(x,y)}{\partial y} = -\frac{y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



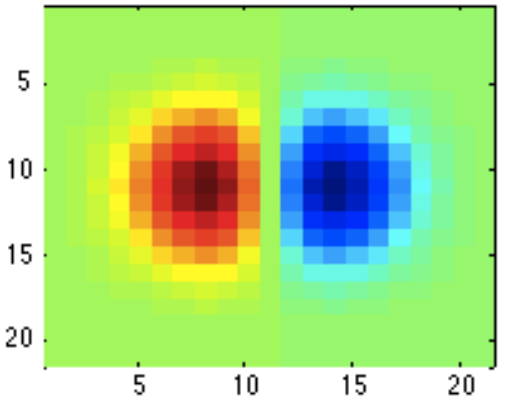
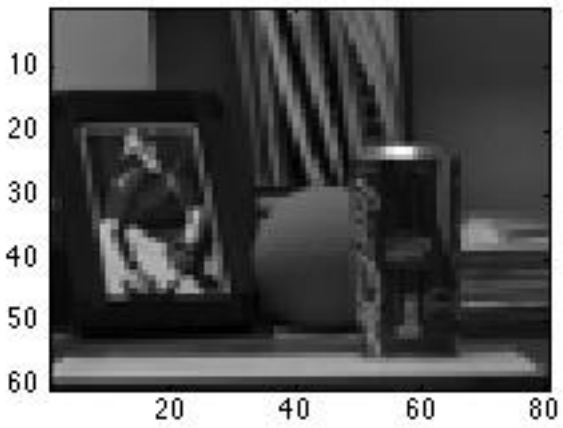
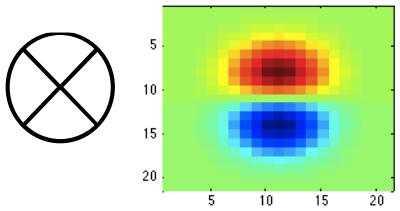


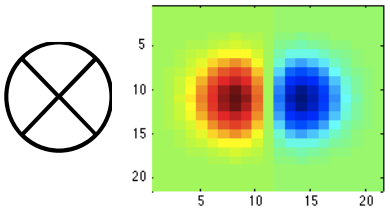
Image Gradient

- Computing Derivatives

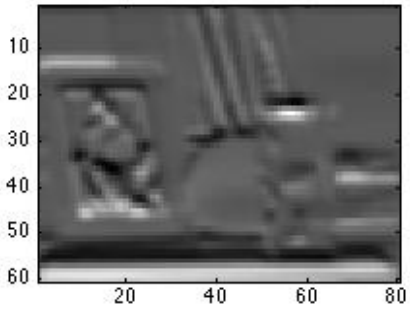
$I(x,y)$



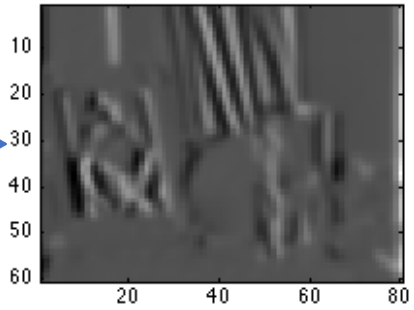




$\frac{\partial(I(x,y)\otimes G(x,y))}{\partial x}$



$\frac{\partial(I(x,y)\otimes G(x,y))}{\partial y}$



$$I(x,y) \otimes \nabla G_\sigma(x,y)$$

Image Gradient

- Gradient magnitude

$$\| \nabla \left[I(x,y)) \otimes G_\sigma(x,y) \right] \| = \sqrt{\left(\frac{\partial (I(x,y) \otimes G(x,y))}{\partial x} \right)^2 + \left(\frac{\partial (I(x,y) \otimes G(x,y))}{\partial x} \right)^2}$$

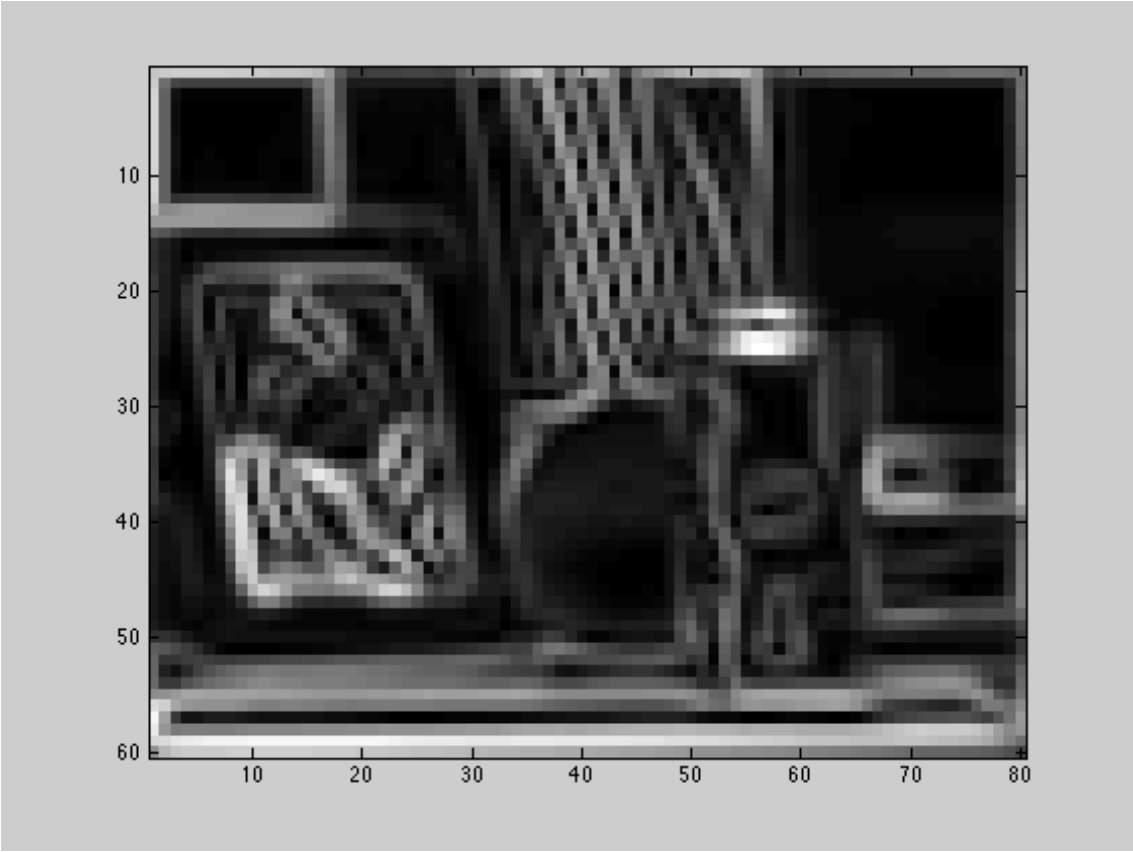


Image Gradient

- Effect of σ

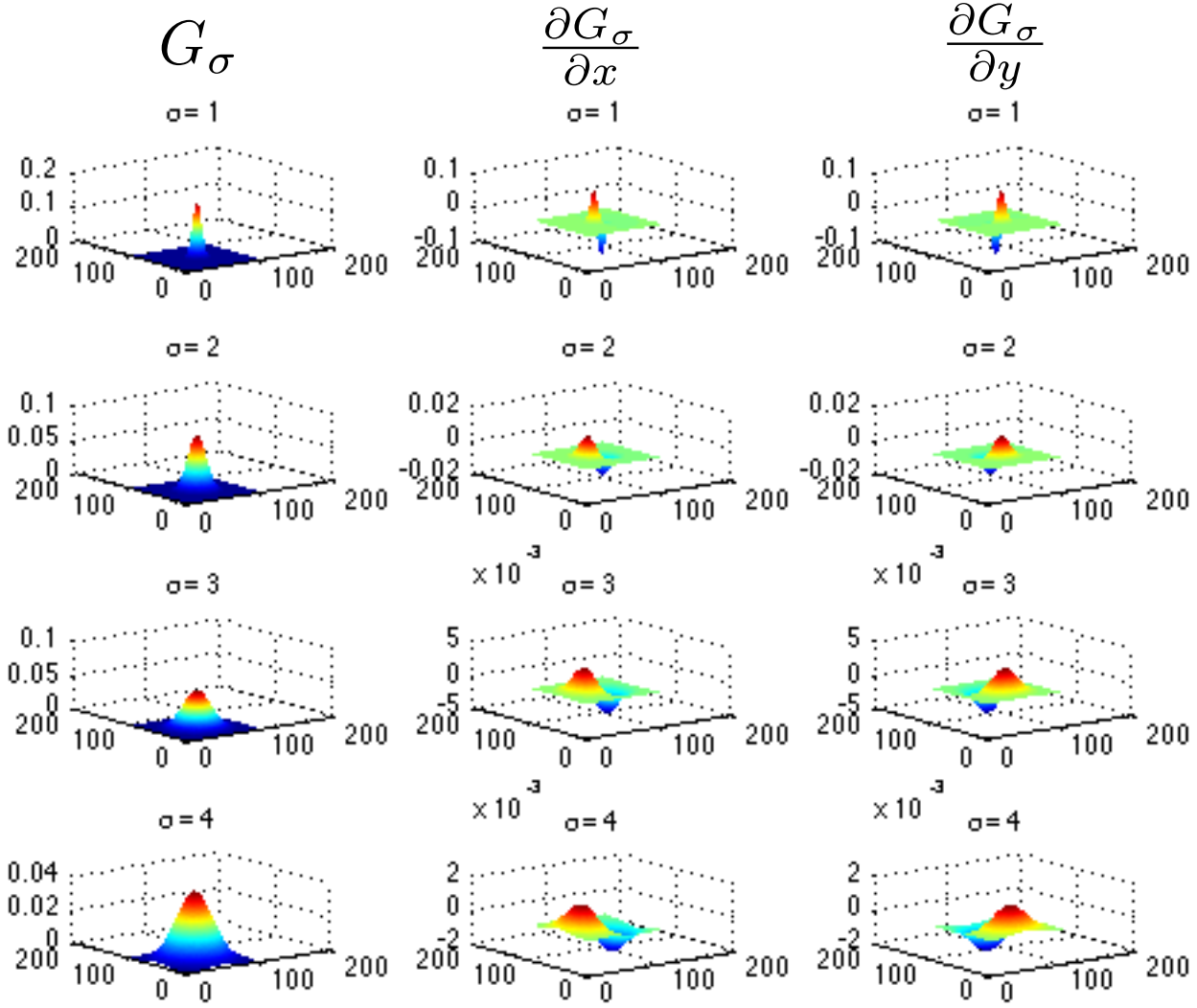
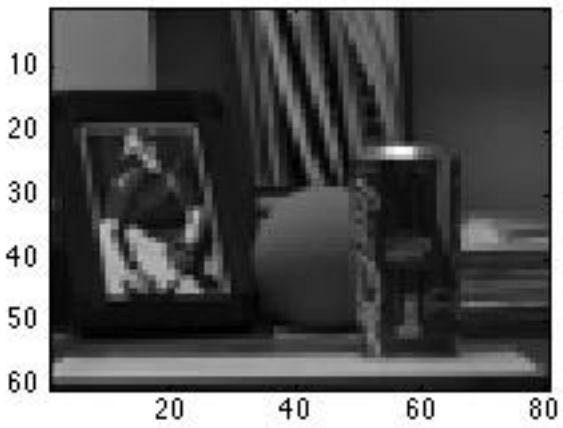


Image Gradient

- Effect of σ



$\frac{\partial(I(x,y)\otimes G(x,y))}{\partial x}$ $\frac{\partial(I(x,y)\otimes G(x,y))}{\partial y}$ $\|\nabla(.)\|$

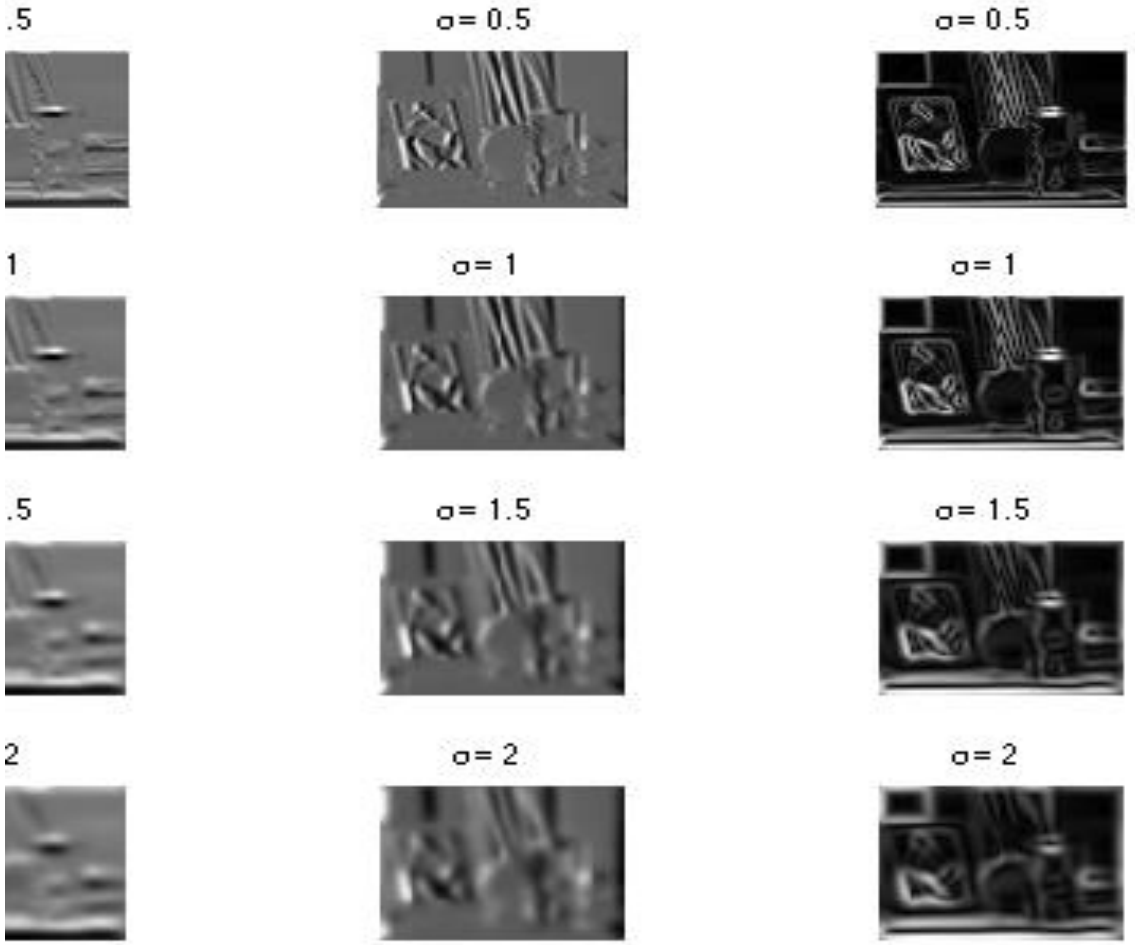


Image Gradient

- Gradient Vectors

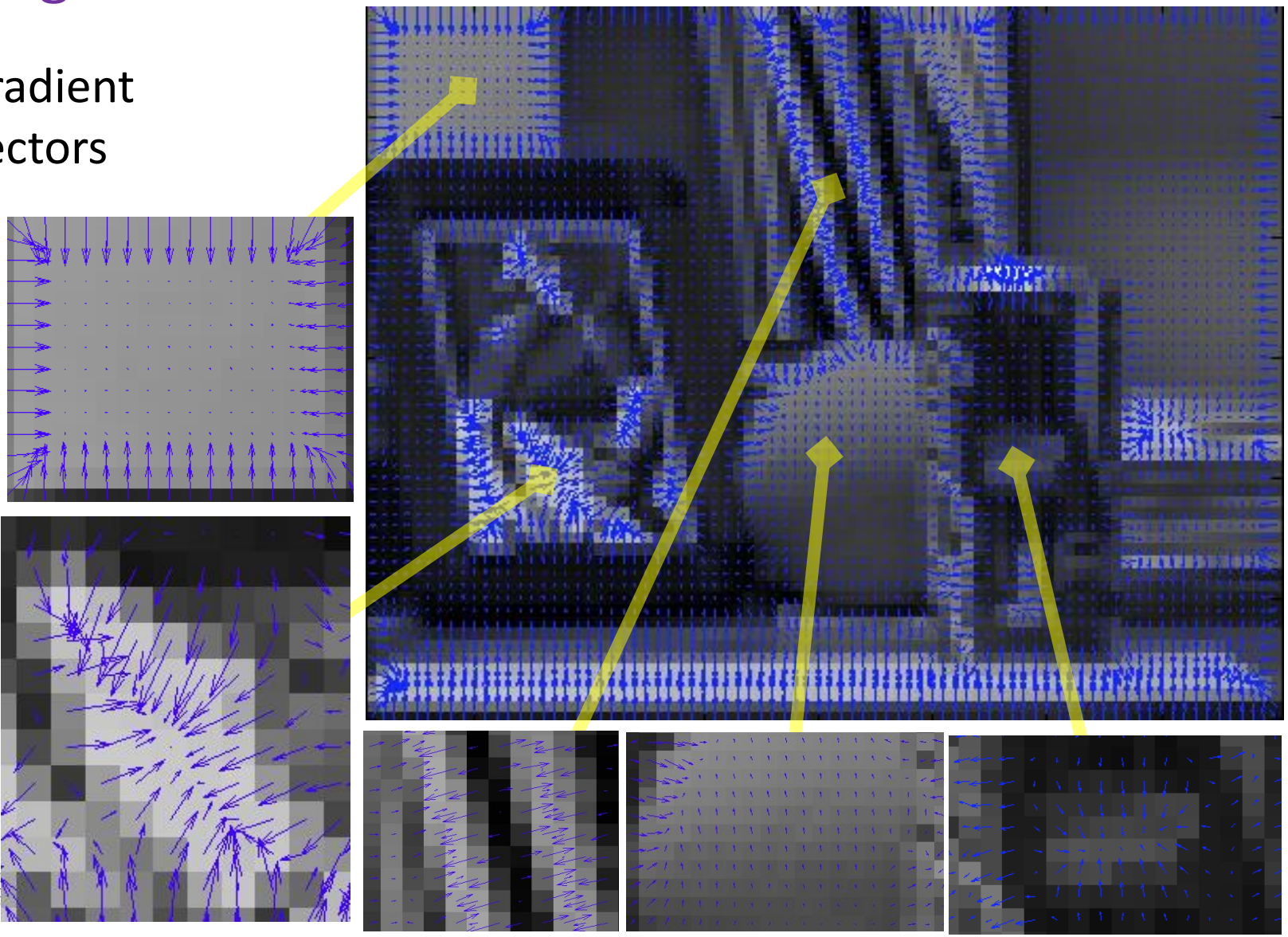
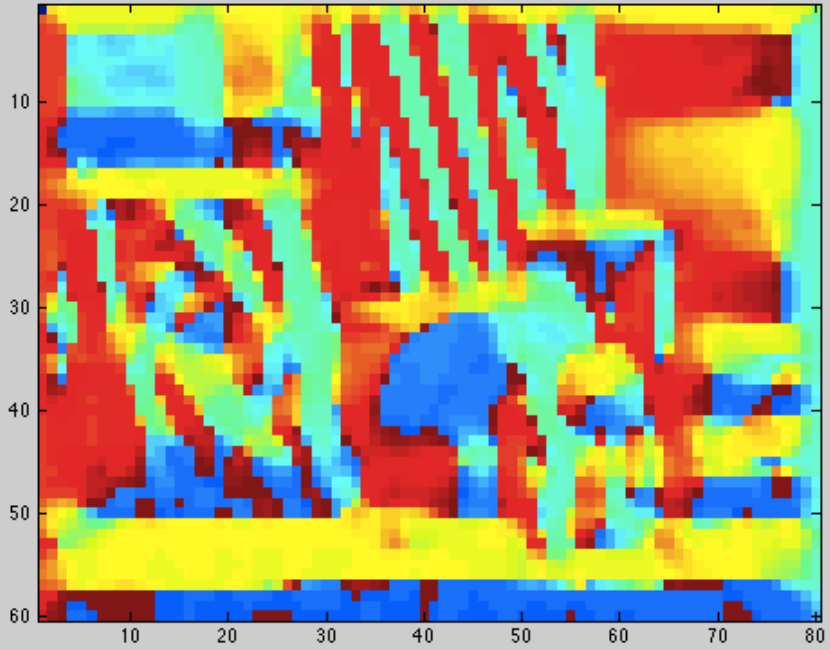


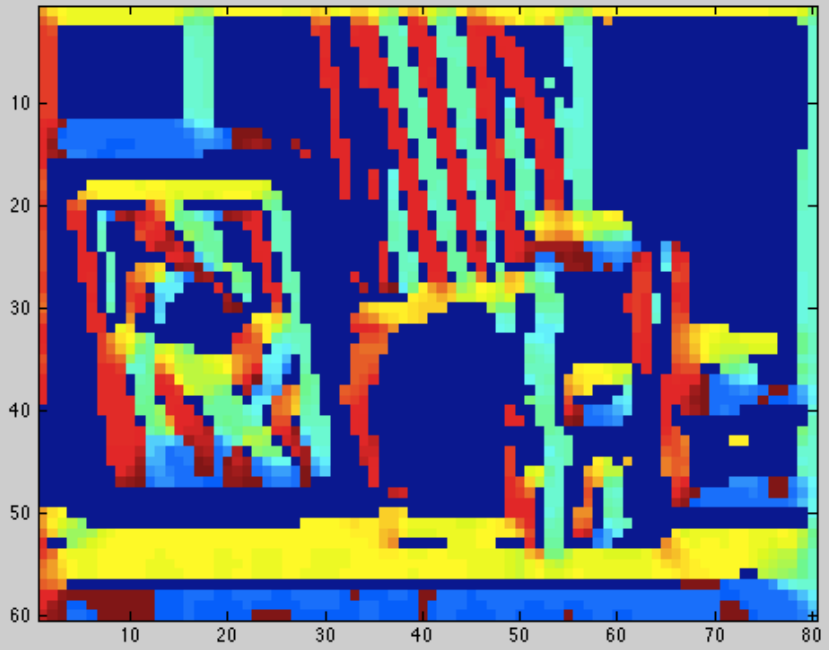
Image Gradient

- Gradient Direction $\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$

Gradient Direction at all locations



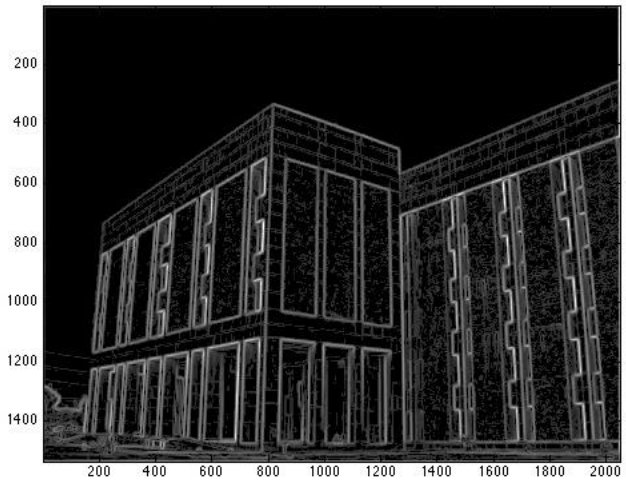
at locations of significant magnitude only



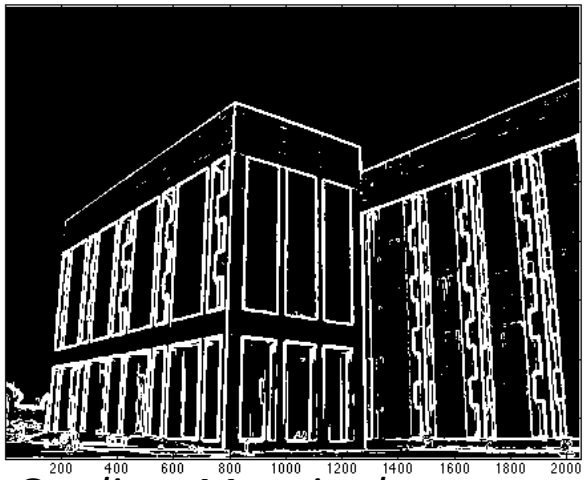
Line Detection

Line Detection

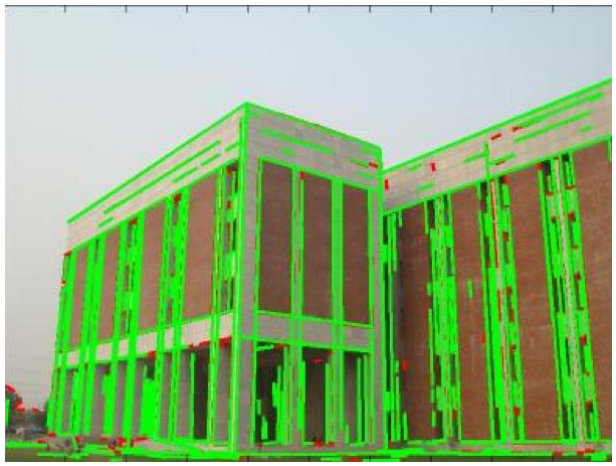
- We can use binary edge mask to detect lines in images



Gradient Magnitude



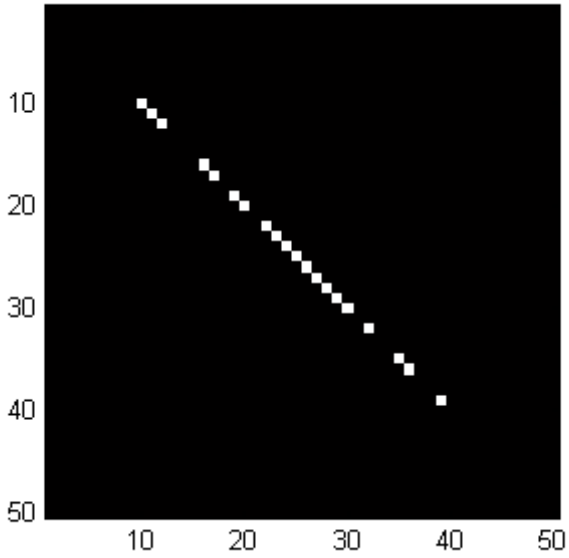
Gradient Magnitude Thresholded



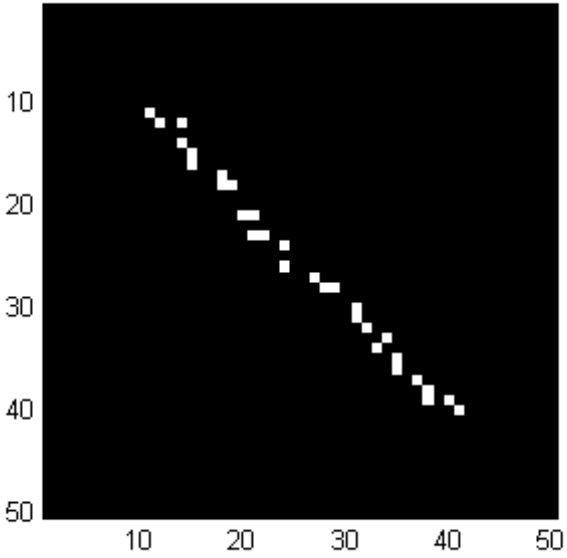
Output of State of Art Line Detector

Problems in Finding Lines

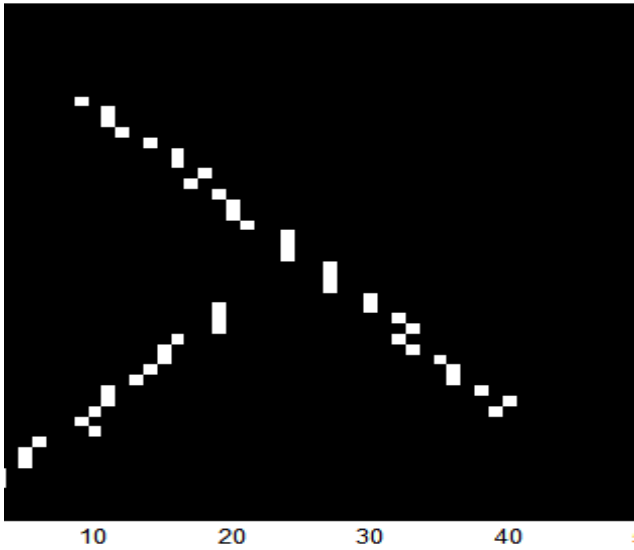
Missing Data



Noisy Data



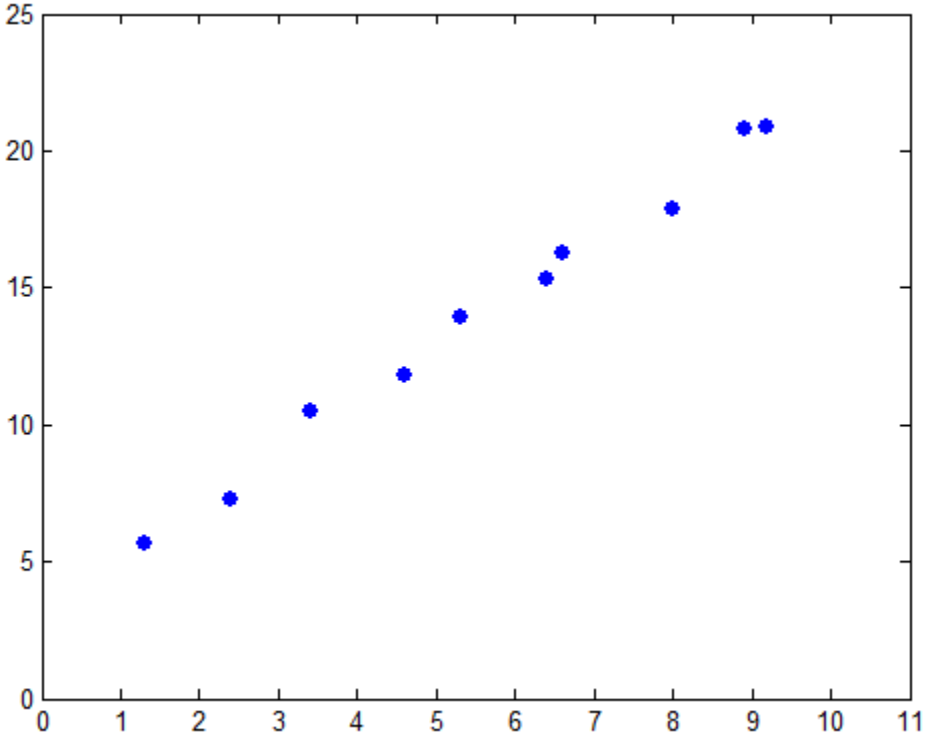
Multiple Lines



Parameter Optimization: Least Squared Error Solutions

- Fitting a line to a set of data points…

x	y
1.3	5.7
2.4	7.3
3.4	10.5
4.6	11.8
5.3	13.9
6.6	16.3
6.4	15.3
8.0	17.9
8.9	20.8
9.2	20.9



- Equation of best fit line ?

Line Fitting: Least Squared Error Solution

- Step 1: Identify the model
  - Equation of line: $y = mx + c$
- Step 2: Set up an error term which will give the goodness of every point with respect to the (unknown) model
  - Error induced by i^{th} point:
  - $e_i = mx_i + c - y_i$
  - Error for whole data: $E = \sum_i e_i^2$
  - $E = \sum_i (mx_i + c - y_i)^2$
- Step 3: Differentiate Error w.r.t. parameters, put equal to zero and solve for minimum point

Line Fitting: Least Squared Error Solution

$$E = \sum_i (mx_i + c - y_i)^2$$

$$\frac{\partial E}{\partial m} = \sum_i (mx_i + c - y_i)x_i = 0$$

$$\frac{\partial E}{\partial c} = \sum_i (mx_i + c - y_i) = 0$$

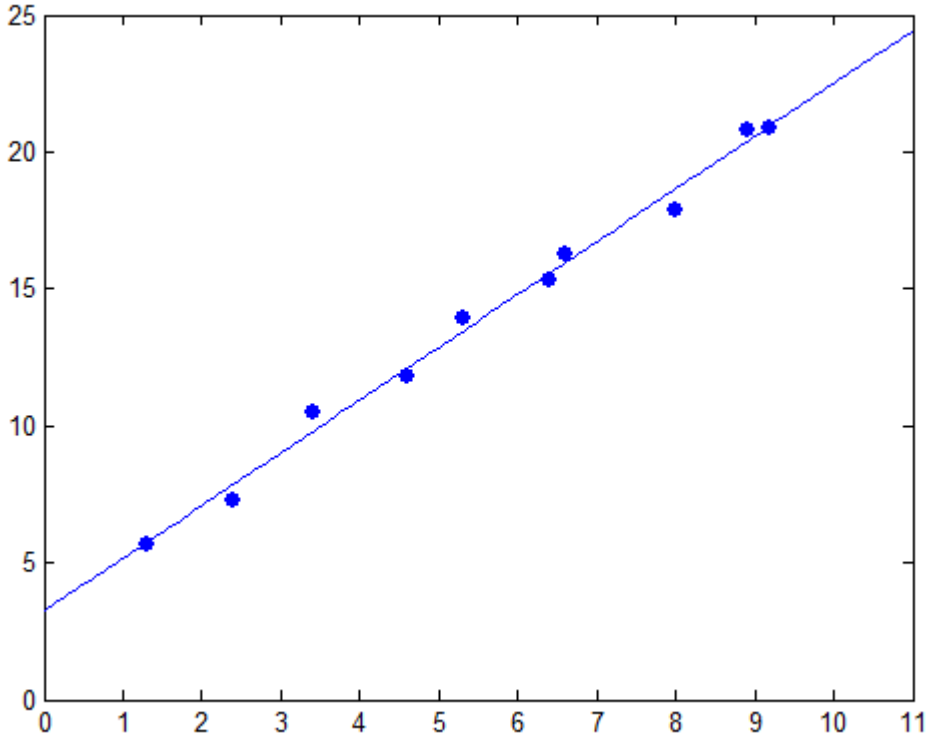
$$\begin{bmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & \sum_i 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} = \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix}$$

x	y
1.3	5.7
2.4	7.3
3.4	10.5
4.6	11.8
5.3	13.9
6.6	16.3
6.4	15.3
8.0	17.9
8.9	20.8
9.2	20.9

$$\begin{pmatrix} 380.63 & 56.1 \\ 56.1 & 10 \end{pmatrix} \begin{pmatrix} m \\ c \end{pmatrix} = \begin{pmatrix} 914.68 \\ 140.4 \end{pmatrix}$$

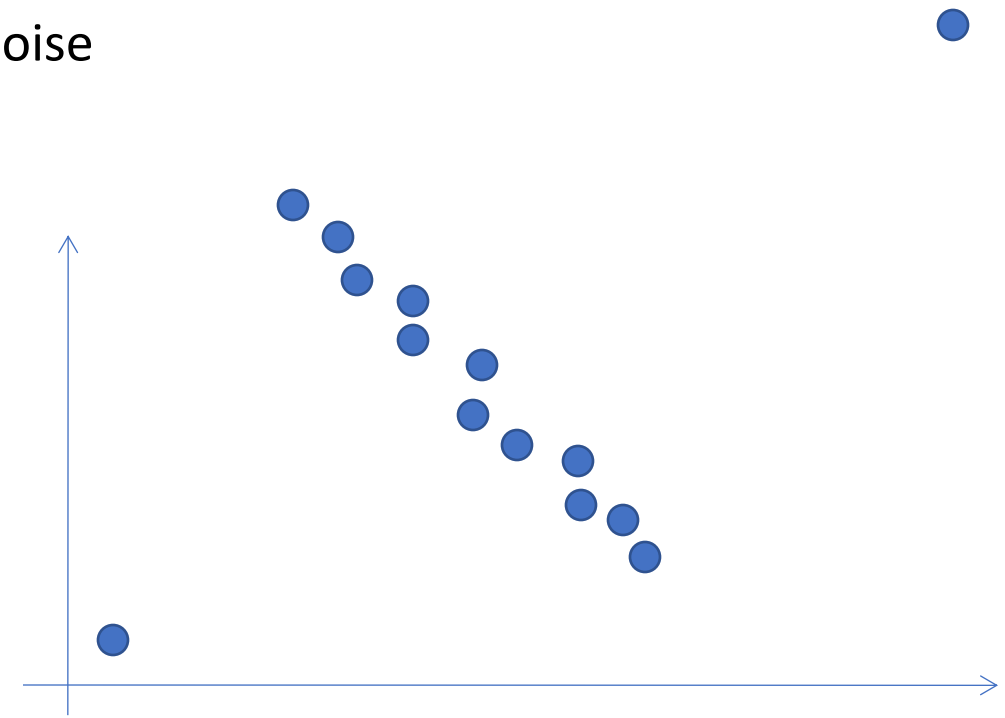
Solution: m = 1.9274 c = 3.227

Line Fitting: Least Squared Error Solution



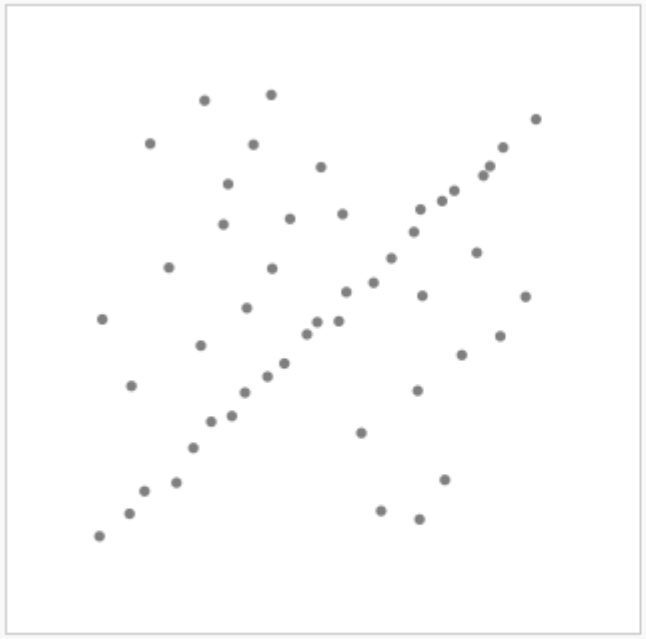
Least Squared Error Solution

- Disadvantages?
  - Multiple Lines…
  - Not robust to noise
- Example

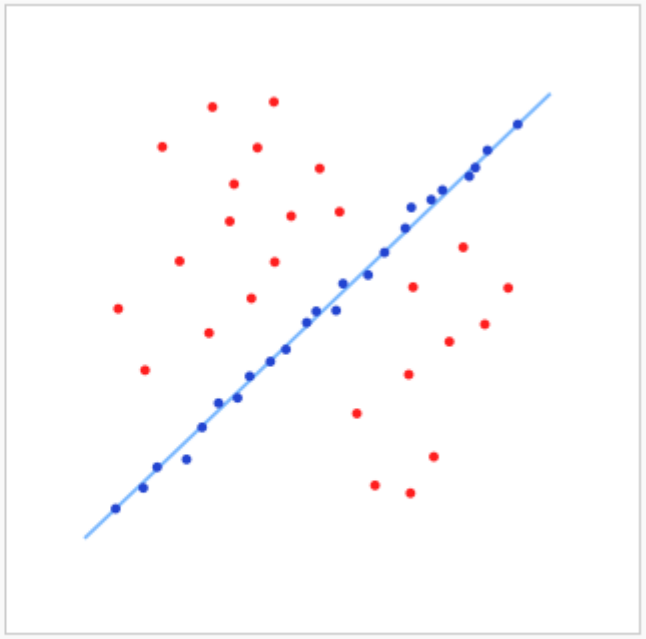


Another Approach to Line Fitting

- RANSAC (RANdom SAmple Consensus)
- Very robust to outliers



A data set with many outliers for which a line has to be fitted.



Fitted line with RANSAC, outliers have no influence on the result.

http://en.wikipedia.org/wiki/RANSAC

RANSAC… General Approach

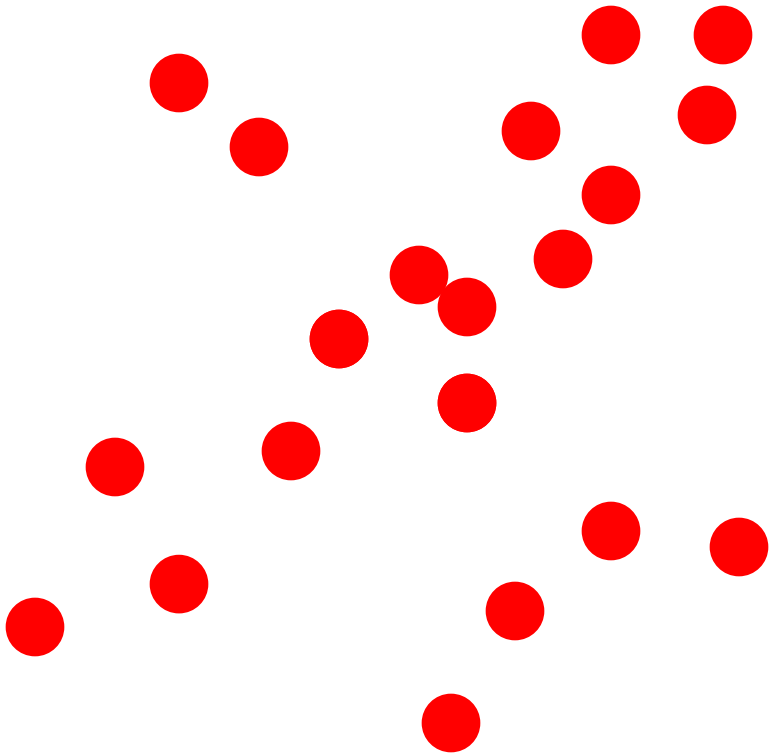
- Select minimum number of random points from data needed to estimate the model
- Estimate the model from selected random points
- Check how many other points are consistent with the fitted model (Consistent Set)
  - If consistent set is large enough, estimate model from all points in the consistent set
  - If the consistent set is larger than the previous best model, make the current model as the best model
- Repeat a number of times

RANSAC

Adapted from James Hays slides

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



Algorithm:

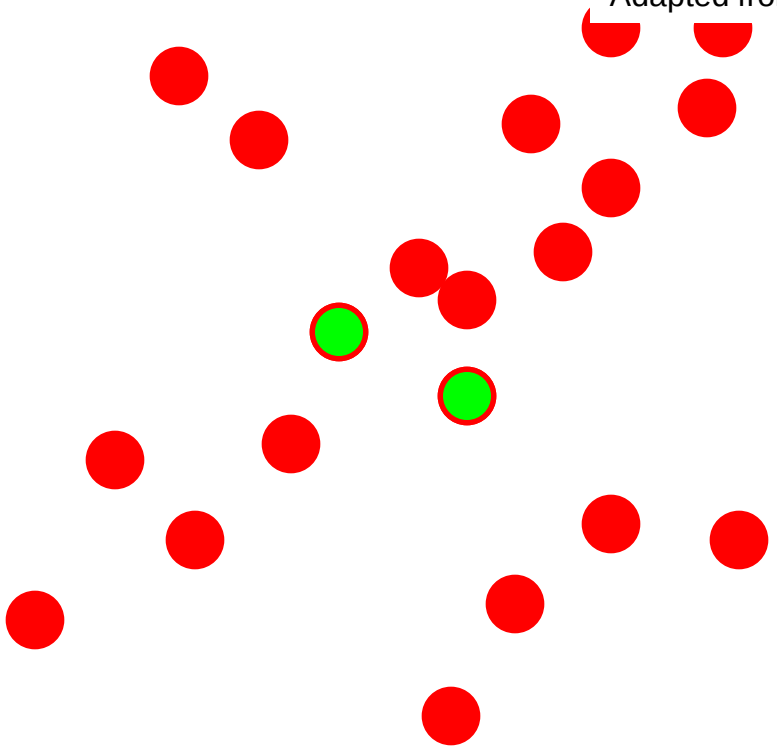
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Adapted from James Hays slides

RANSAC

Line fitting example



Algorithm:

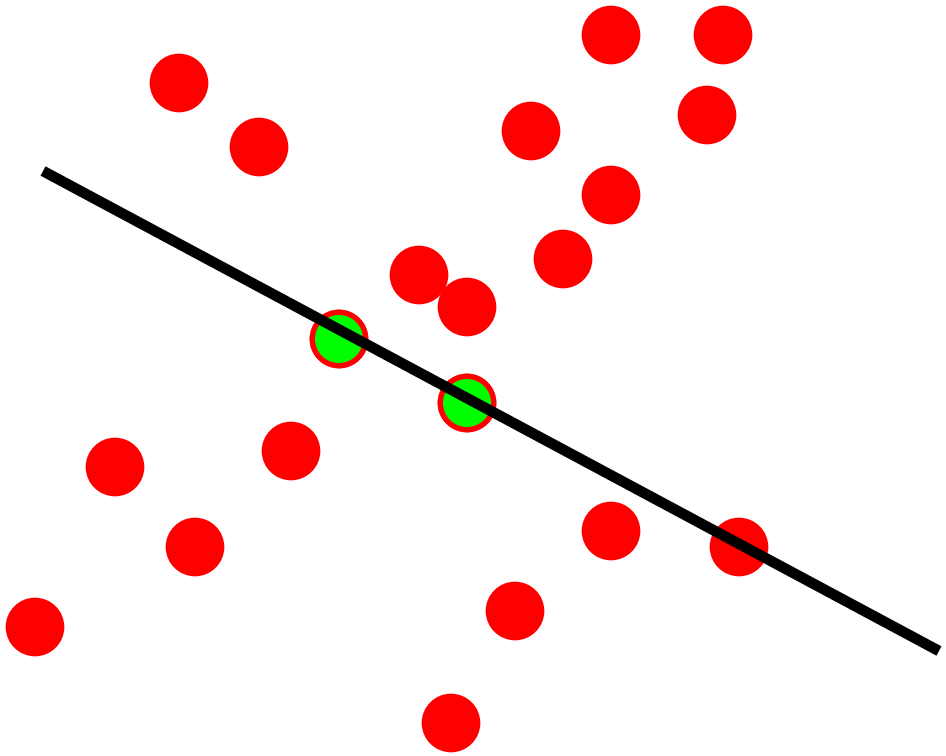
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



Algorithm:

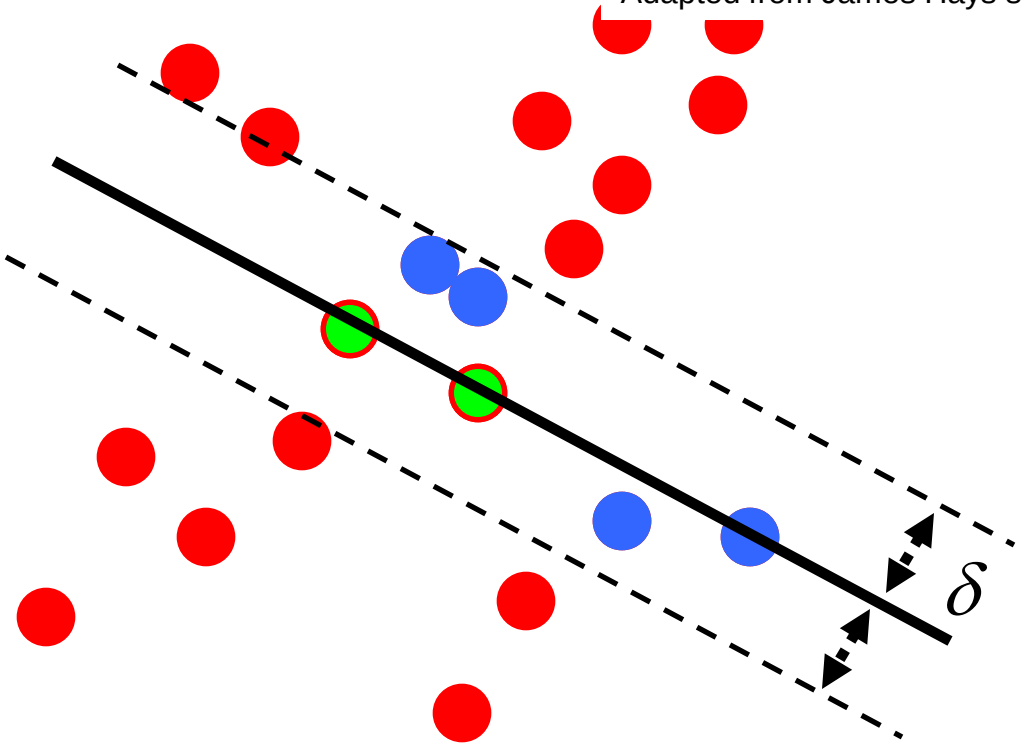
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Adapted from James Hays slides

RANSAC

Line fitting example



$$N_I = 6$$

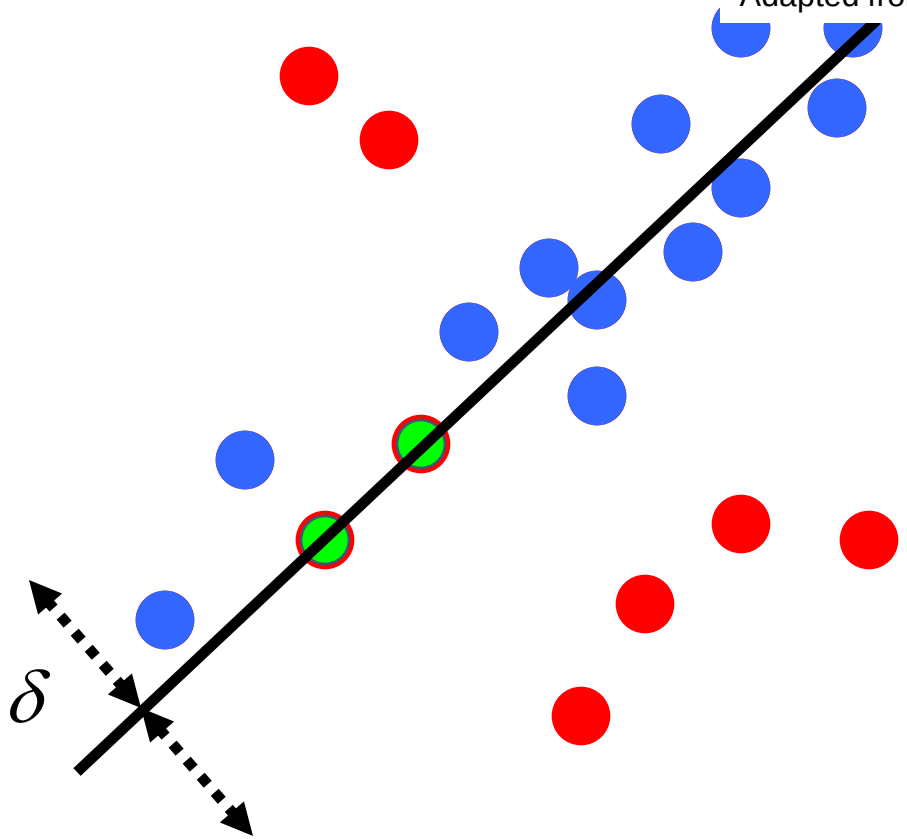
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Adapted from James Hays slides

RANSAC



$$N_I = 14$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of objective function parameters
- Optimization parameters are easier to choose than other approaches

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Needs to be adapted further to allow multiple fits

Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

Line Segment Detector (LSD)

http://www.ipol.im/pub/art/2012/gjmr-lsd/

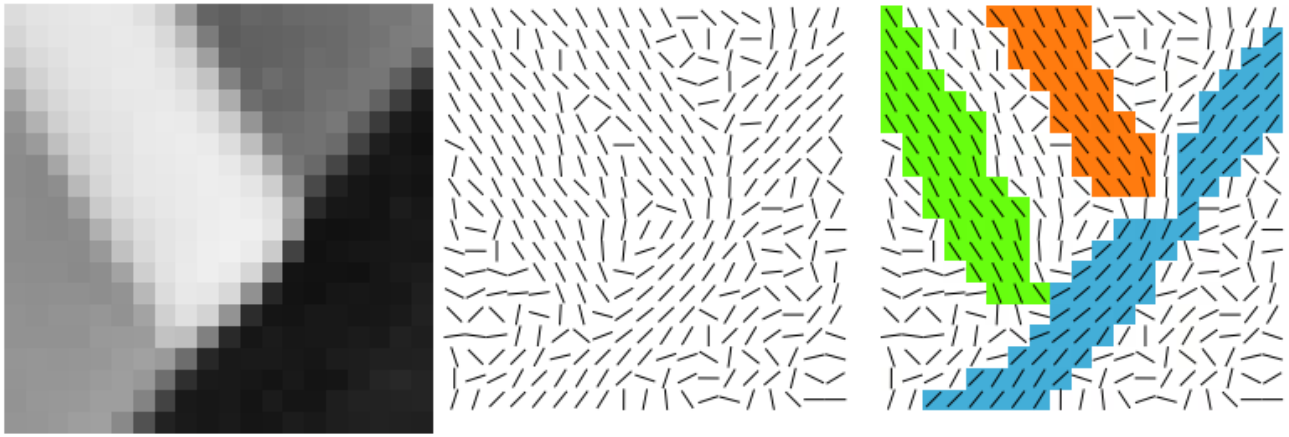


Image | Level-line Field | Line Support Regions

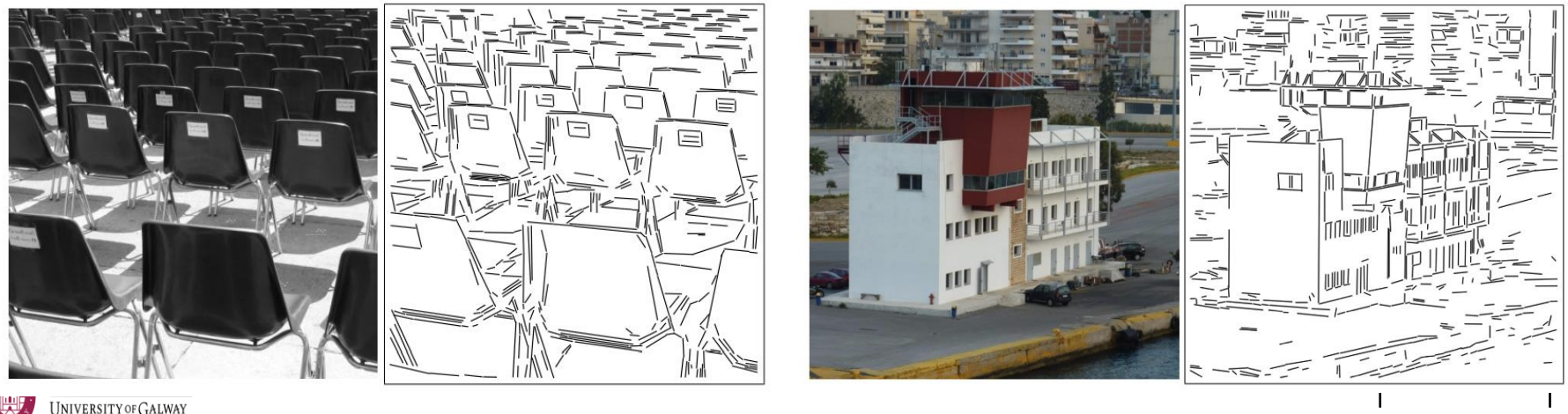


Image Features

What are Image Features?

- A feature is some piece of information about an image or about a portion of an image.
- Features are statistics of an image or a portion of it

- **Global Features**: Statistic of the image as a whole
- **Regional Features**: Statistic of a portion of an image
- **Local Features**: Statistic of a point, often based on a small neighborhood around it

Local Image Features
Interest Points and Corners

Why?

- Essential for making panoramas

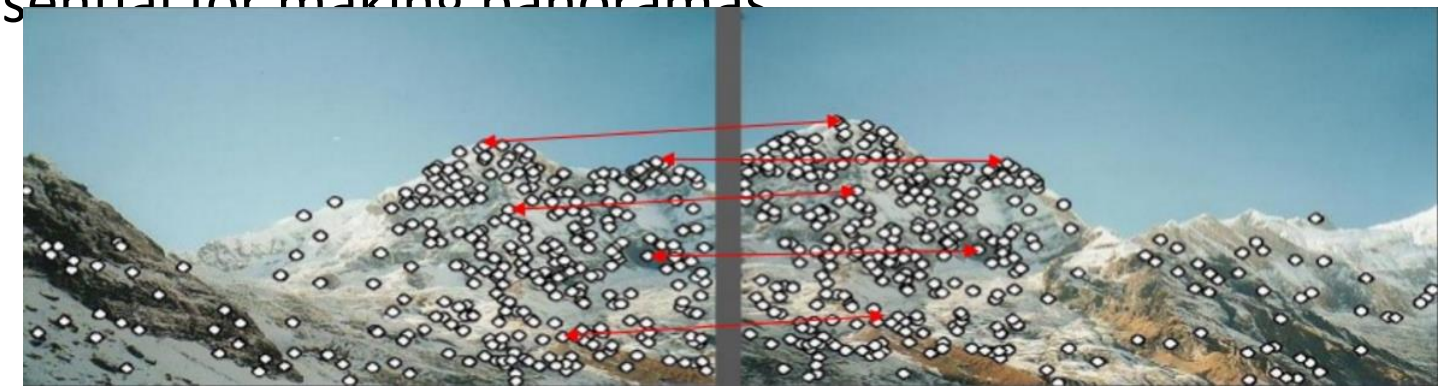


Image Credit: David Lowe

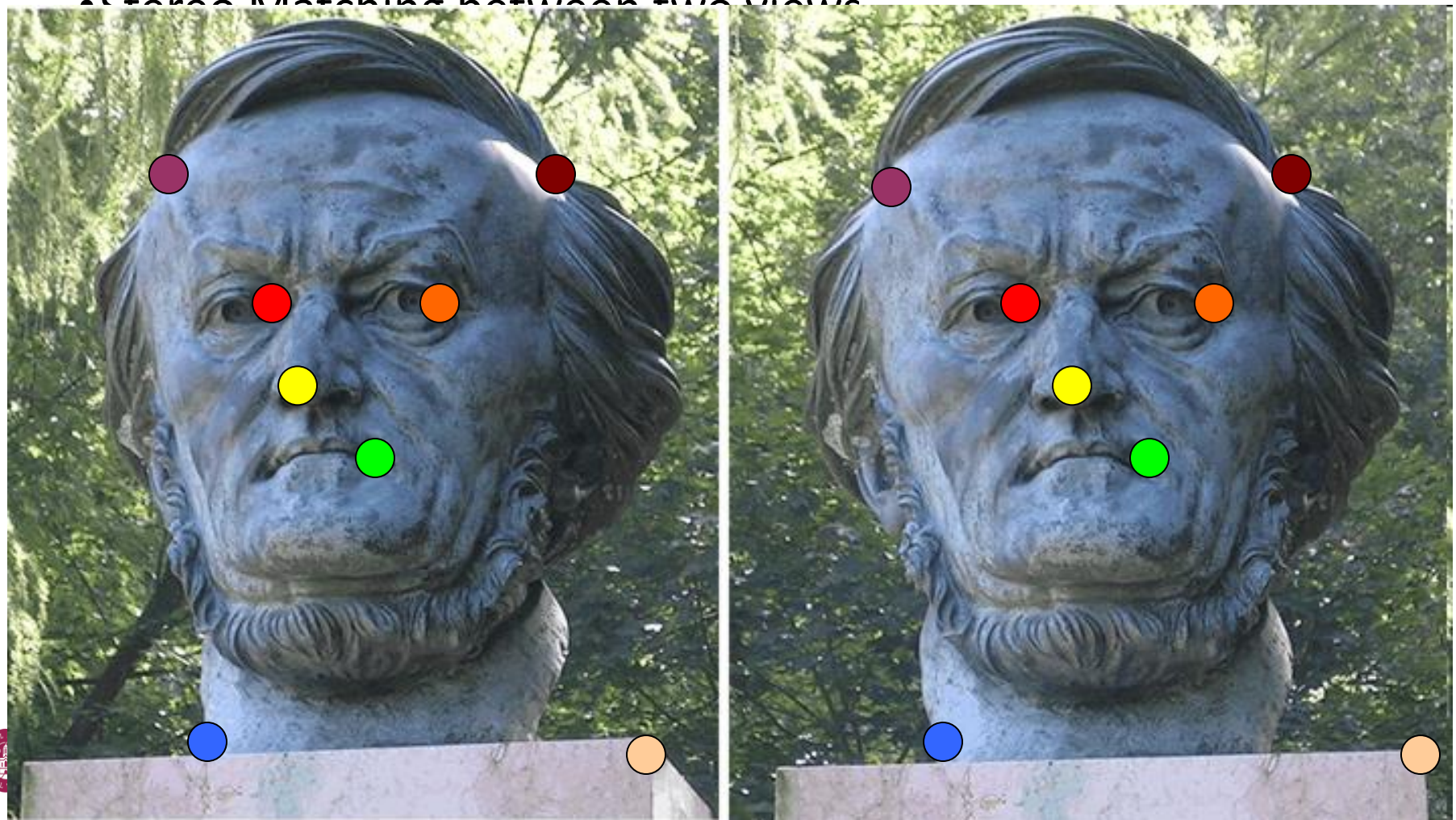
Why?

- Structure from Motion

Adapted from James Hays slides

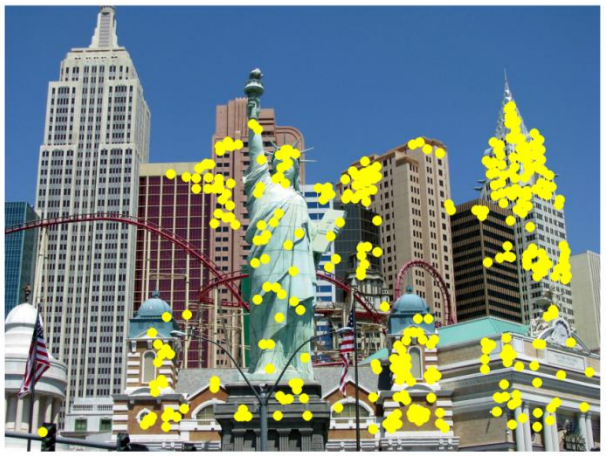
Why?

- Stereo Matching between two views



Applications

- Feature points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition

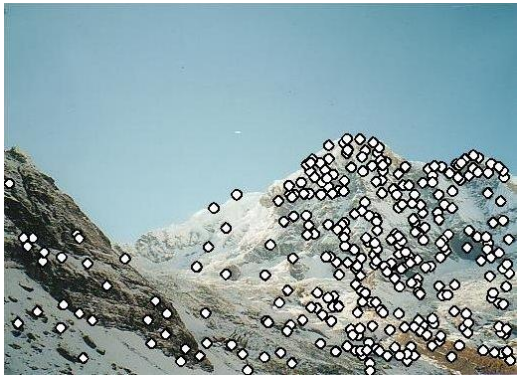


Panorama Stitching

- We have two images – how do we combine them?

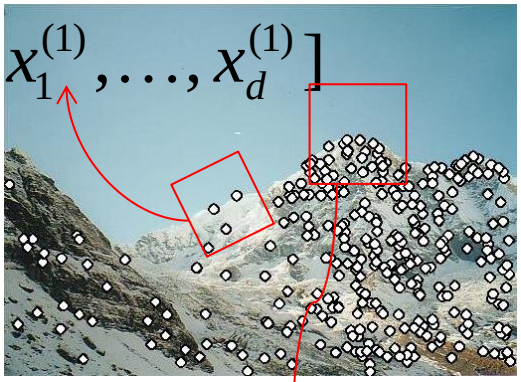
Local features: main components

1) **Detection:** Identify the interest points



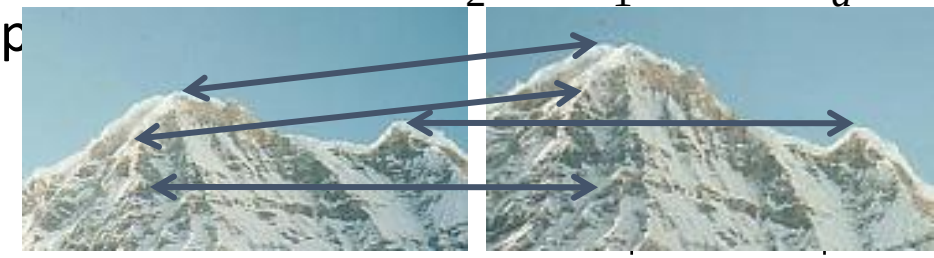
2) **Description:** Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

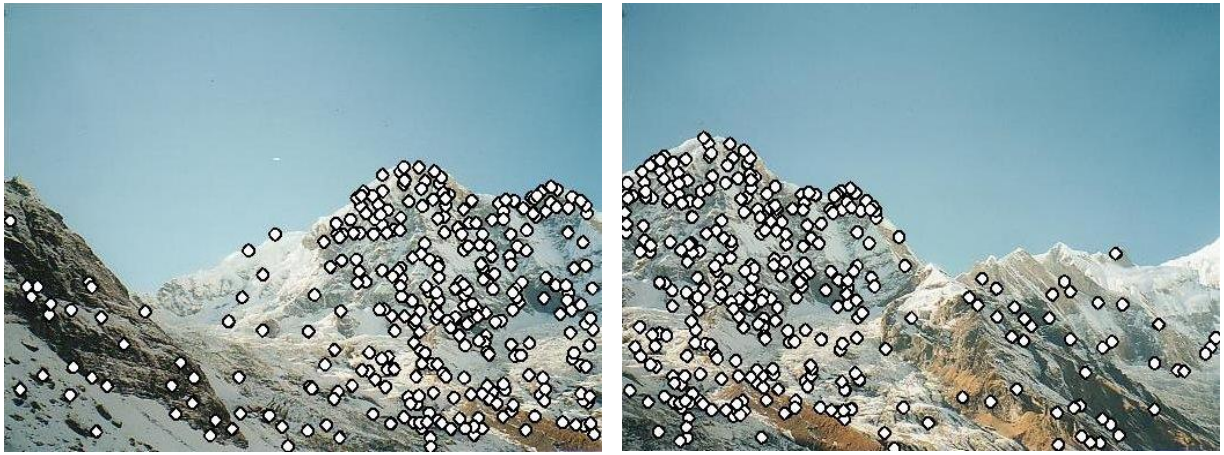


3) **Matching:** Determine correspondence between descrip in two views

$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$



Characteristics of good local features



- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Goal: interest operator repeatability

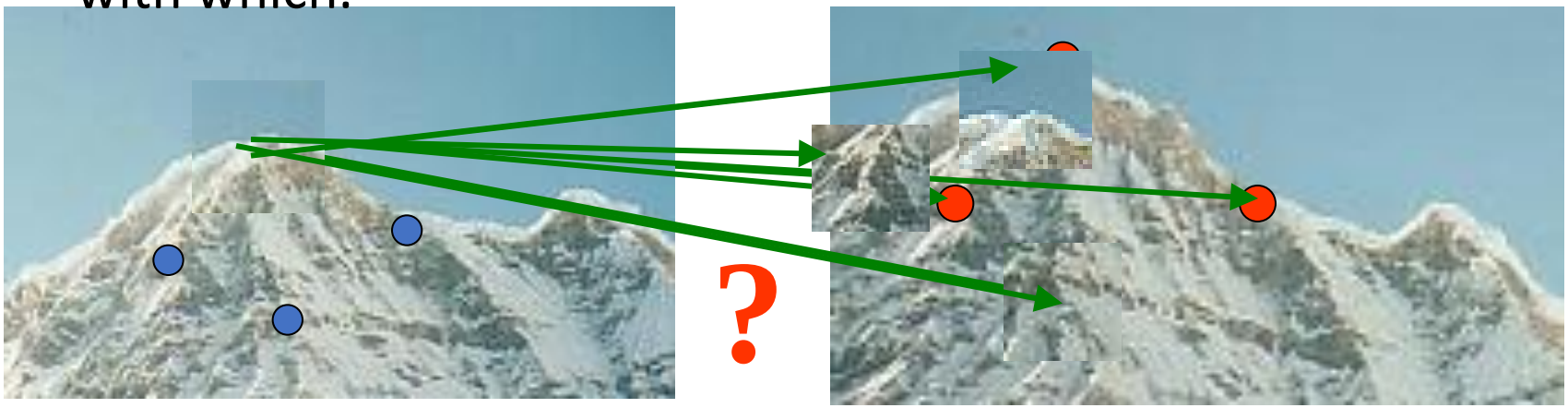
- We want to detect (at least some of) the same points in both images.

No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Goal: descriptor distinctiveness

- We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

Detecting Local Features (Corners)

corners detected

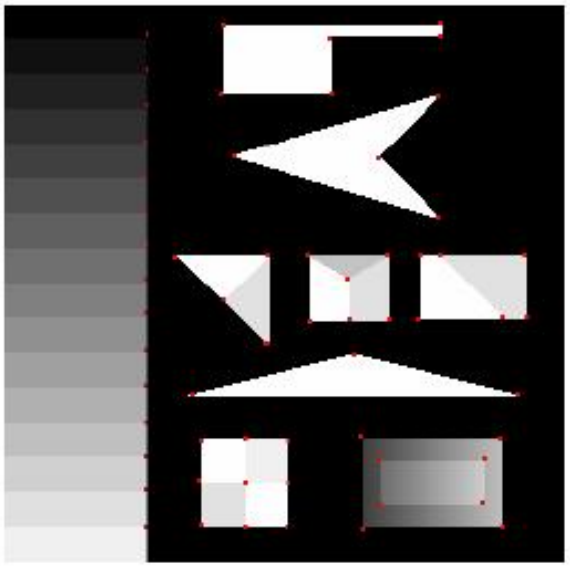
Corners

- How do we define a corner?
- Qs: How does a corner differ from an edge?

- A corner may be interpreted as an intersection of two edges, or as a region where strong derivatives exist in more than one direction

What Is a Corner?

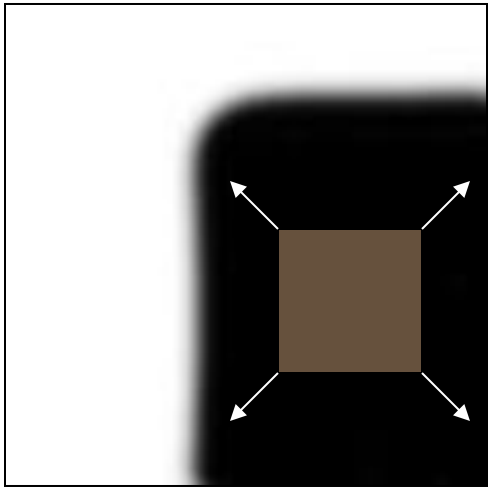
- Corners are local image features characterized by locations where variations of intensity function f(x, y) in both X and Y directions are high
- Intersection points between two or more edge segments



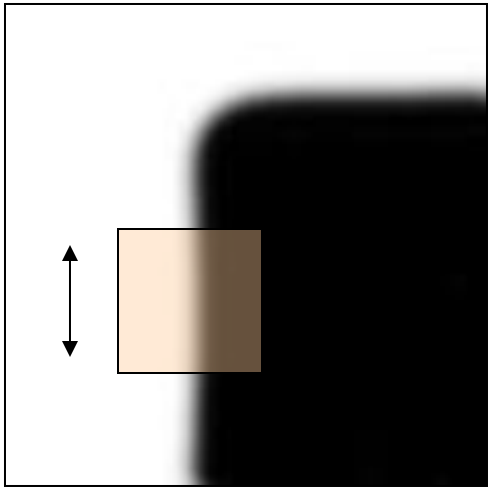
www.cse.cuhk.edu.hk/~lyu/seminar/05fall/Wyman.ppt

Corner Detection: Basic Idea

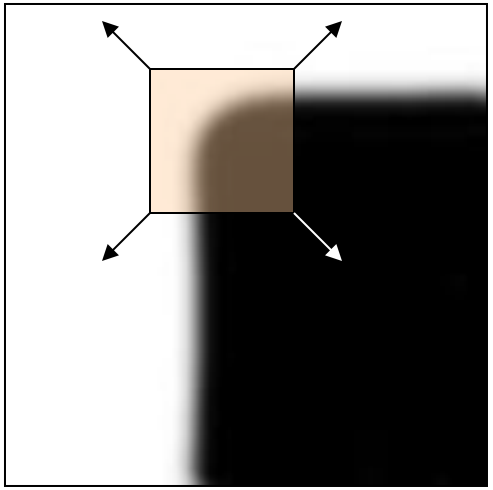
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



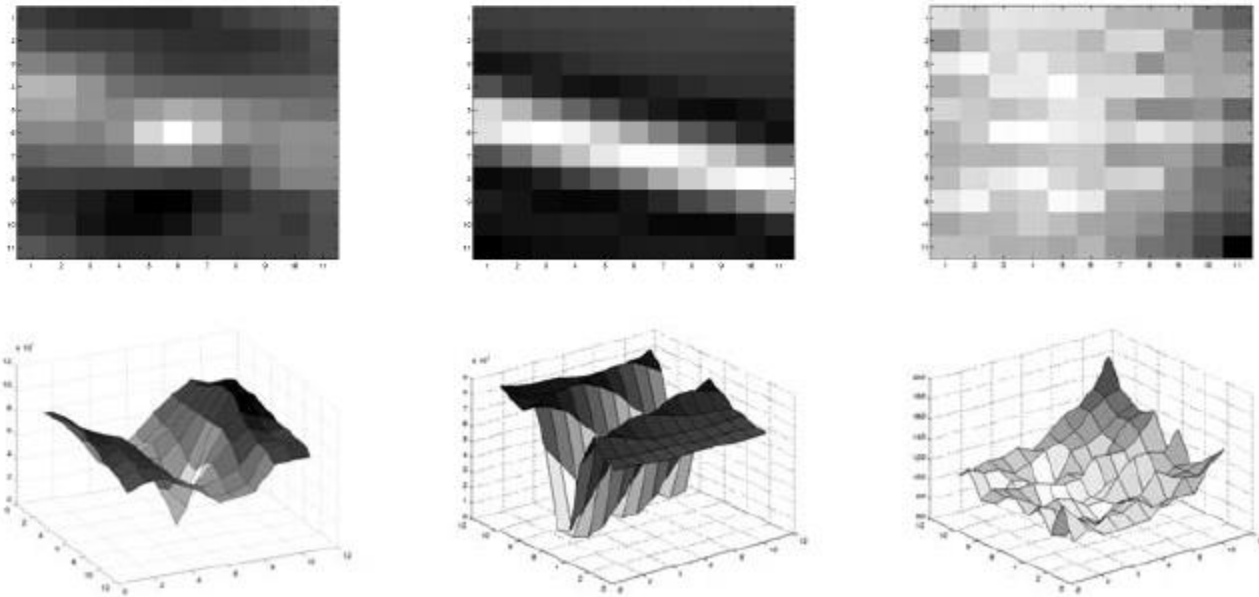
"edge":
no change along the edge direction



"corner":
significant change in all directions

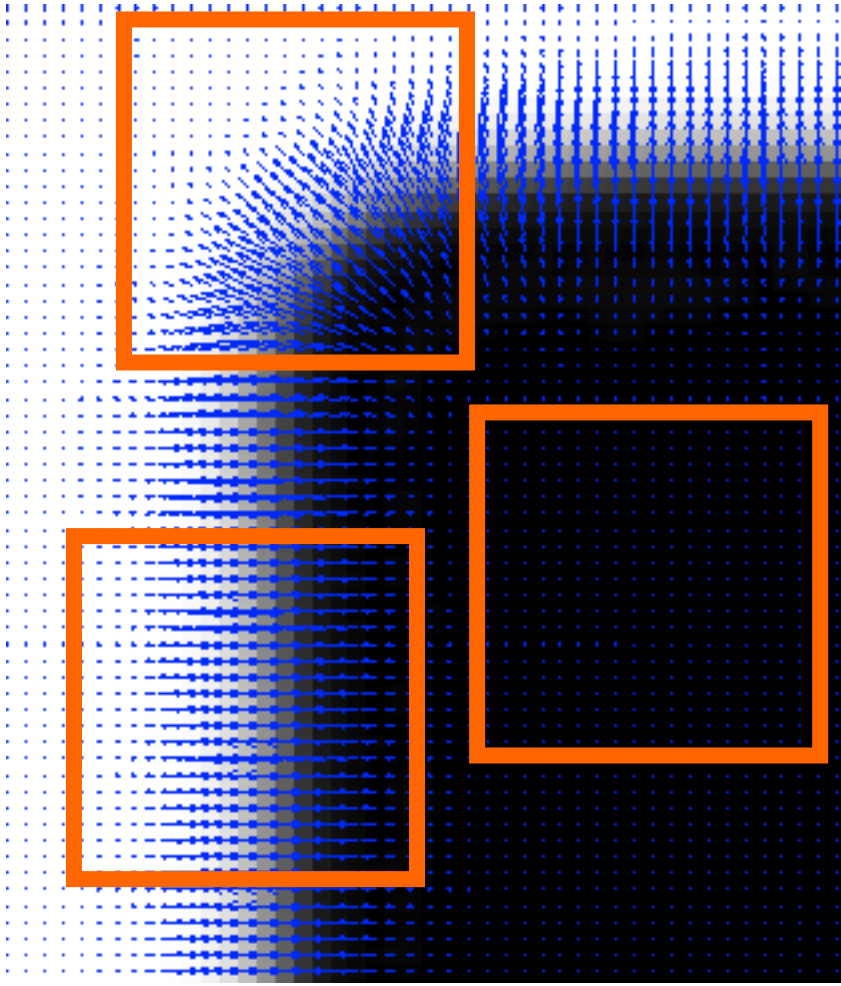
Source: A. Efros

(a)



Building a Simple Corner Detector

- How would the gradient vectors look like in the vicinity of a corner?
- Gradient at smooth region is close to zero
- Gradient vectors at edge are all pointed in one direction
- Gradient vectors at corner are in more than one direction
- Hence, the nature of variation of gradient vectors in a small neighborhood about a point can be an indicator of 'cornerness' of that point



Building a Simple Corner Detector

- The nature of variations in gradient in a small window is an indicator of "cornerness" at a point
- How can we measure "variations" of gradient vector?

- For a one dimensional entity, variation can be measured by variance

$$\mathrm{Var}[X] = \mathrm{E}\left[(X - \mathrm{E}[X])^2\right]$$

- This is mathematically equal to

$$\mathrm{Var}[X] = \mathrm{E}[X^2] - (\mathrm{E}[X])^2$$

- But Gradient is a two-dimensional vector entity
- How can we talk about the variation of a vector entity?

Building a Simple Corner Detector

- Variation of a vector entity can be described by its covariance matrix

$$\mathrm{Cov}[\mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^\top] - \mathrm{E}[\mathbf{x}]\mathrm{E}[\mathbf{x}]^\top$$

- What is likely to be the mean of a gradient vector

$$\mathrm{E}(\nabla(I(x,y)) = ?$$

- For zero-mean vectors, covariance can be written as

$$\mathrm{Cov}[\mathbf{x}] = \mathrm{E}[\mathbf{x}\mathbf{x}^\top]$$

- Hence, covariance of gradient vector is given by

$$\mathrm{Cov}[\nabla I(x,y)] = \begin{bmatrix} \Sigma(\frac{\partial I}{\partial x})^2 & \Sigma(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}) \\ \Sigma(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}) & \Sigma(\frac{\partial I}{\partial y})^2 \end{bmatrix}$$

For a **zero-mean gradient vector** at each pixel, where we assume that the mean of the gradients within a local neighborhood is zero

Building a Simple Corner Detector

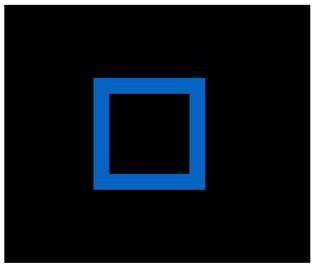
For a **zero-mean gradient vector** at each pixel, where we assume that the mean of the gradients within a local neighborhood is zero

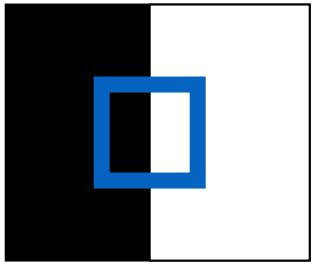
$$\mathrm{Cov}[\nabla I(x,y)] = \begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y^2 \end{bmatrix}$$

- This matrix tells us about the variation of gradient vectors in a window

Building a Simple Corner Detector

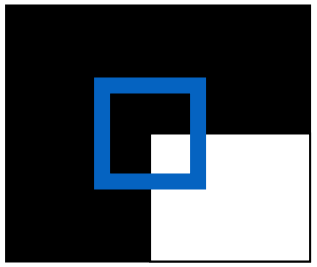
What form will $\begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y^2 \end{bmatrix}$ take in each case?

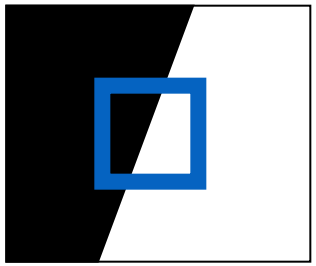


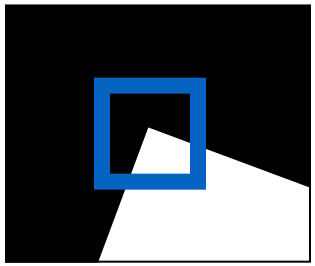


$$\mathbf{\Sigma} \approx \begin{bmatrix} \mathbb{E}\left[\left(\frac{\partial I}{\partial x}\right)^2\right] & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{\Sigma} \approx \begin{bmatrix} 0 & 0 \\ 0 & \mathbb{E}\left[\left(\frac{\partial I}{\partial y}\right)^2\right] \end{bmatrix}$$







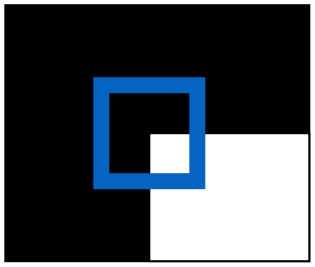
Can we somehow distinguish between the covariance matrices of these two cases?

Some Properties of this Covariance Matrix

- Is positive-semidefinite
  - Consequently, its eigen values will not be negative

$$\begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y^2 \end{bmatrix}$$

- If the x values do not tell me anything about the y values, then the off-diagonal terms are zero



- Rotation of the image by R changes covariance A by **RAR**$^\mathsf{T}$

Proof: Before rotation, covariance $\mathtt{A} = \mathrm{E}[\nabla I \nabla I^\top]$

After rotation covariance

$$\begin{aligned} &= \mathrm{E}[(\mathtt{R}\nabla I)(\mathtt{R}\nabla I)^\top] \\ &= \mathtt{R}\mathrm{E}[(\nabla I)(\nabla I)^\top]\mathtt{R}^\top \\ &= \mathtt{RAR}^\top \end{aligned}$$

Some Properties of this Covariance Matrix

- Consider the eigenvectors of A

$$\mathtt{A}\phi_1 = \lambda_1\phi_1$$
$$\mathtt{A}\phi_2 = \lambda_2\phi_2$$

- Both equations can be combined as

$$\mathtt{A}\Phi = \Phi\Lambda \qquad (1)$$

where

$\Phi = [\phi_1\phi_2]$ is an orthonormal matrix of eigenvectors

$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ is a diagonal matrix of eigenvalues

- From (1) $\Phi^\top \mathtt{A}\Phi = \Lambda$

- Implies: rotating the image by $\Phi^\top$ diagonalizes covariance

Building a Simple Corner Detector

- Hence, we do not need to rotate the image in any case
- We just need to find the eigen values of **A**, which, when arranged as Λ give us the covariance of the rotated image!

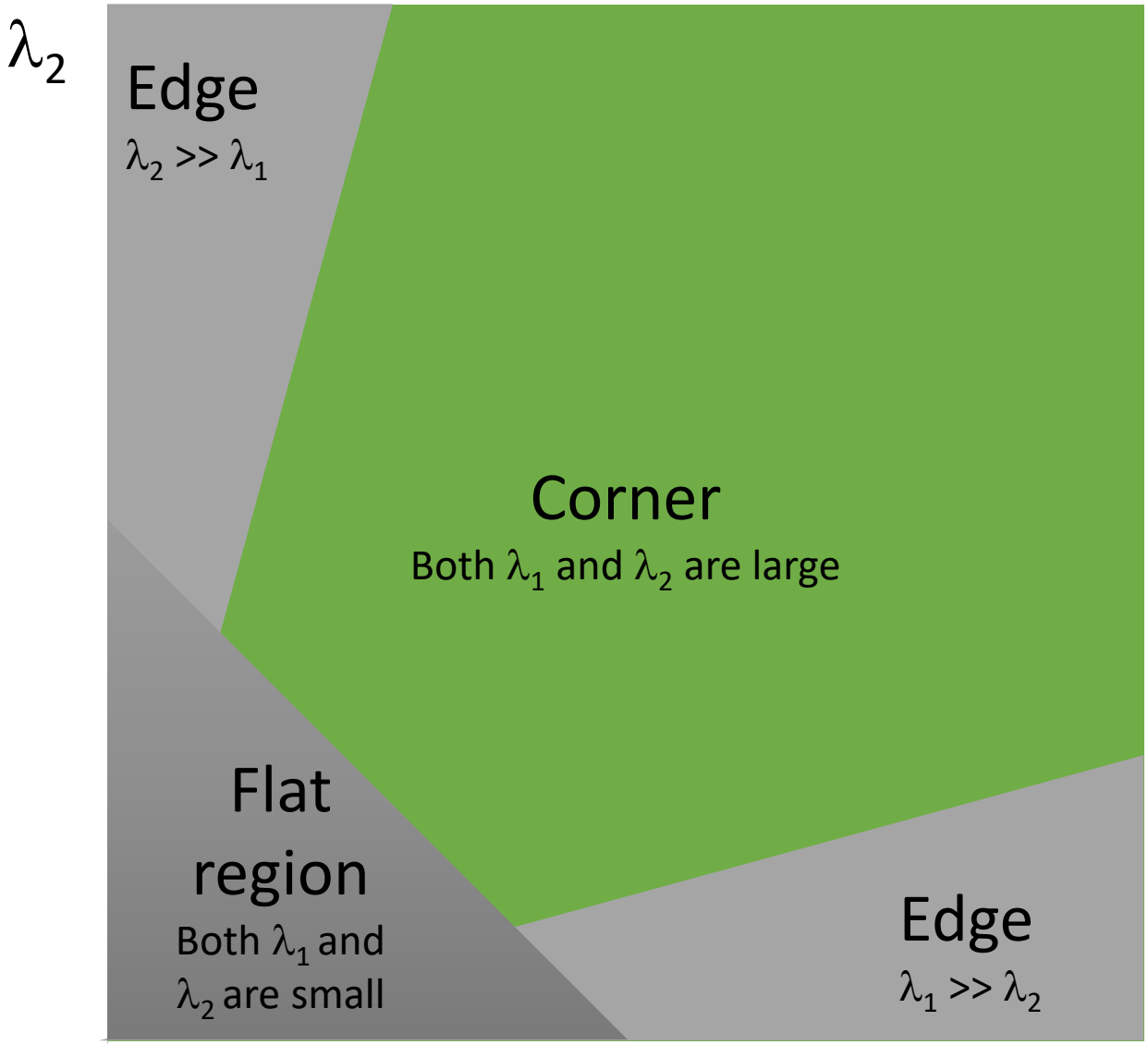
- This leads us to the KLT corner detector algorithm

KLT Corner Detector

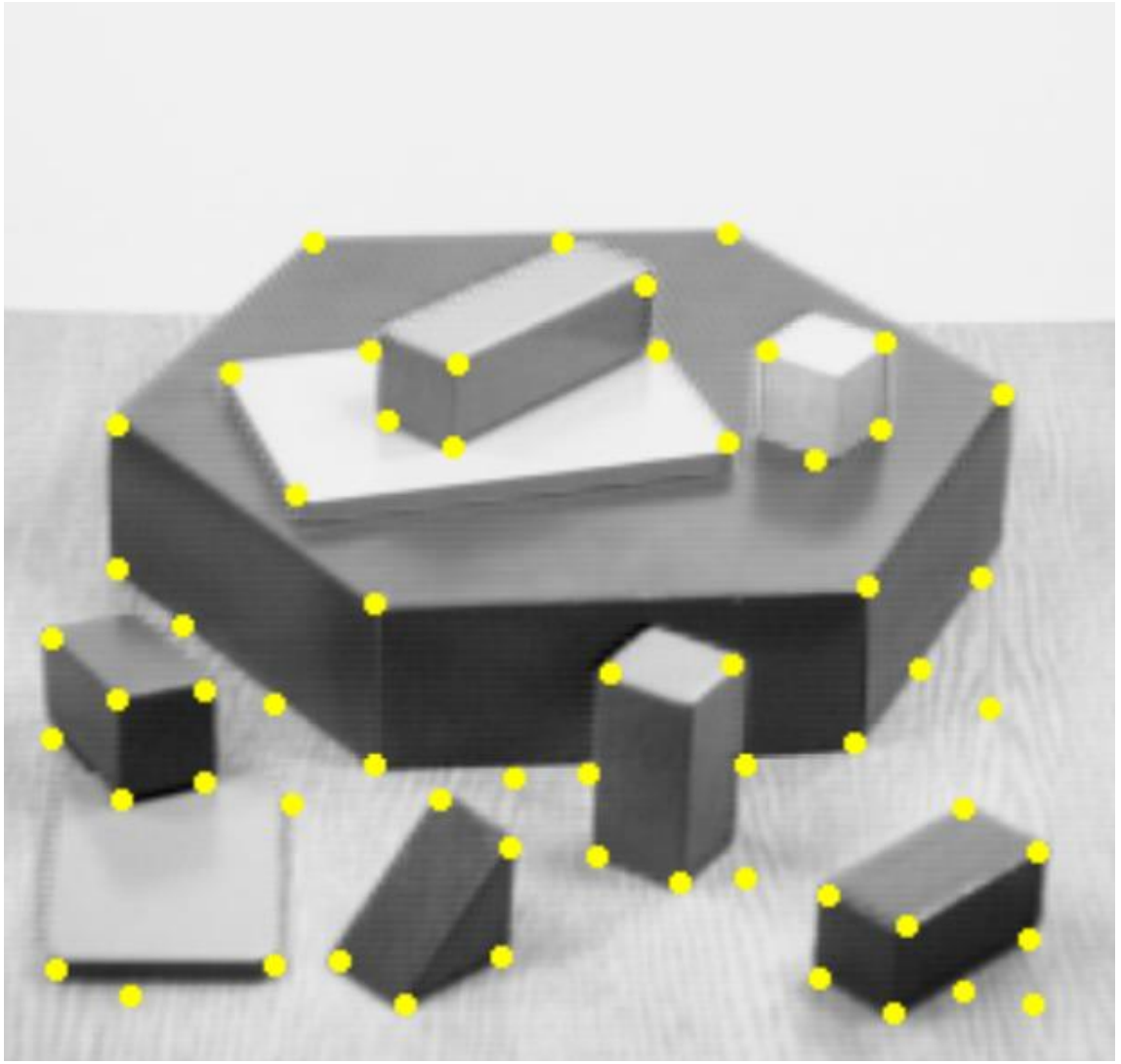
- Let λ_1 , λ_2 be eigenvalues of the gradient covariance matrix
- For a smooth patch:
  $\lambda_1 = \lambda_2 = 0$
- Edge
  $\lambda_1 > 0, \lambda_2 = 0$
- Corner
  $\lambda_1 > 0, \lambda_2 > 0$

$$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

- KLT Criterion
  $\min[\lambda_1 , \lambda_2] > T$


λ_2
Edge
$\lambda_2 >> \lambda_1$
Corner
Both λ_1 and λ_2 are large
Flat region
Both λ_1 and λ_2 are small
Edge
$\lambda_1 >> \lambda_2$
λ_1

Results




LINK
OX
2


OX
LORO
2


OX
LORO
2


ROX
2


2


ULINX
ROX
2


ROX
SPLAS


ROX
CLOR
2
PLAS


ROX

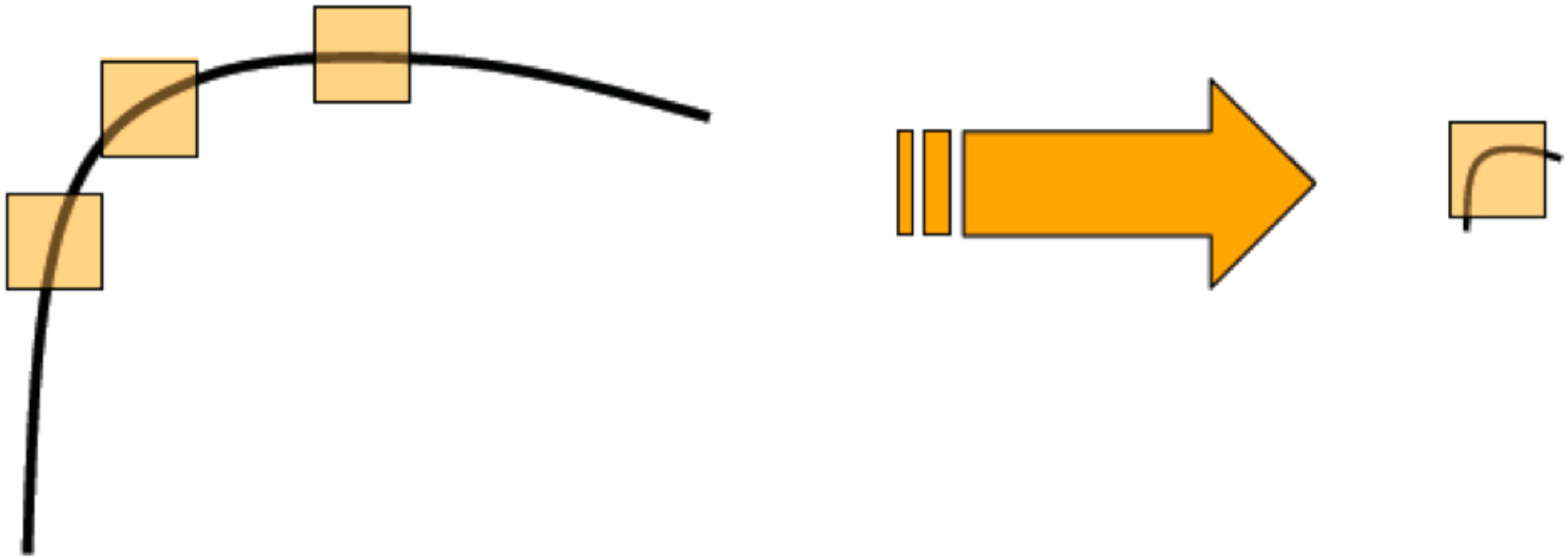

ROX
CLOR
2
PLASH

KLT Corner Detector - Properties

- Invariant to intensity shift $I' = I + b$
  - Because only gradient vectors are used, not actual intensity values
- Relatively invariant to intensity scaling $I' = aI$
  - Because comparison of eigenvalues is used
- Therefore it is invariant to affine intensity changes $I' = aI + b$

KLT Corner Detector - Properties

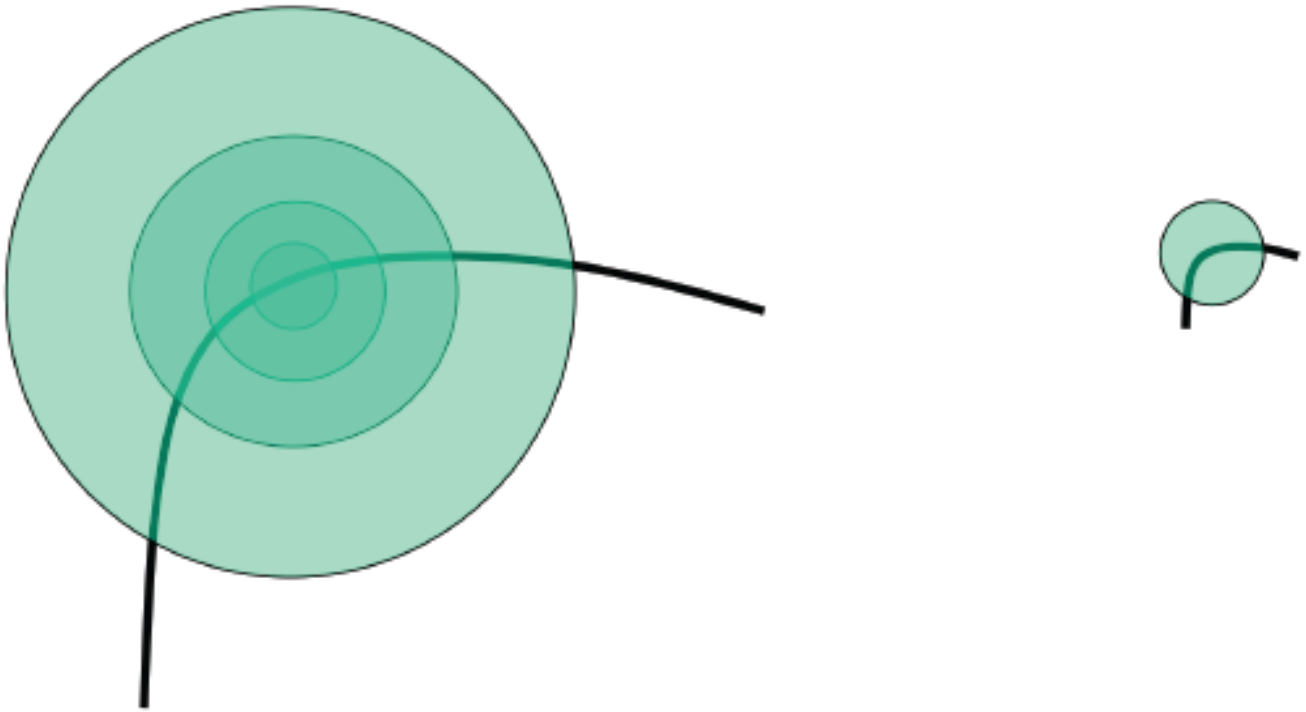
- But it is not invariant to image scale



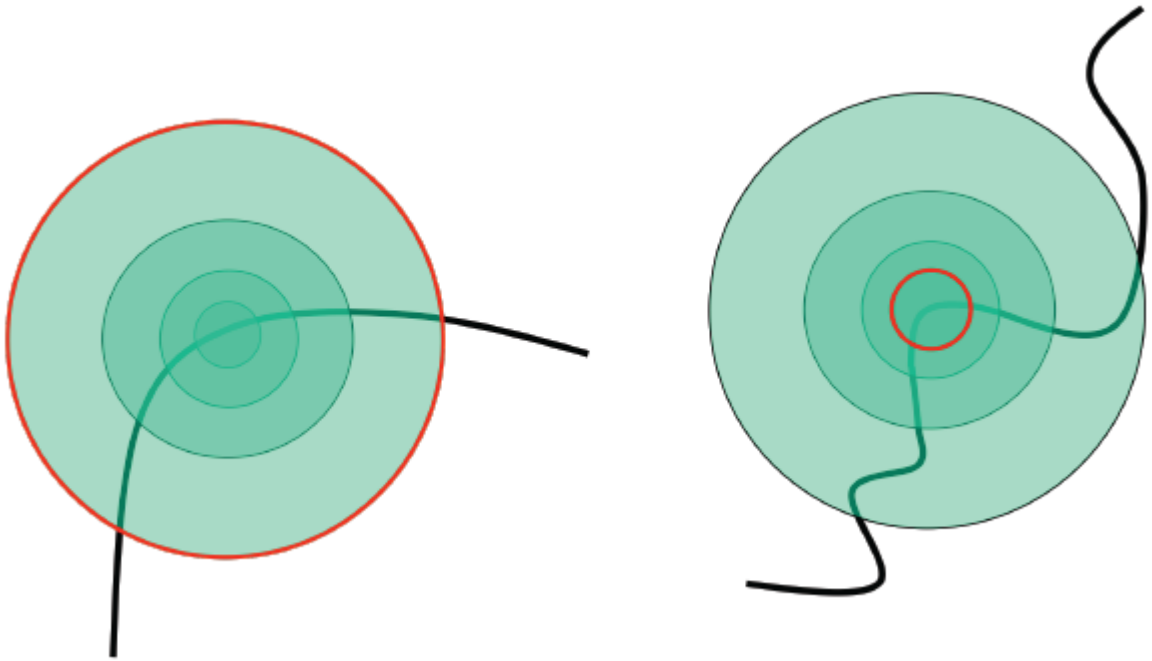
All points will be classified as edges

Corner !

The problem of Scale



The Problem of Scale

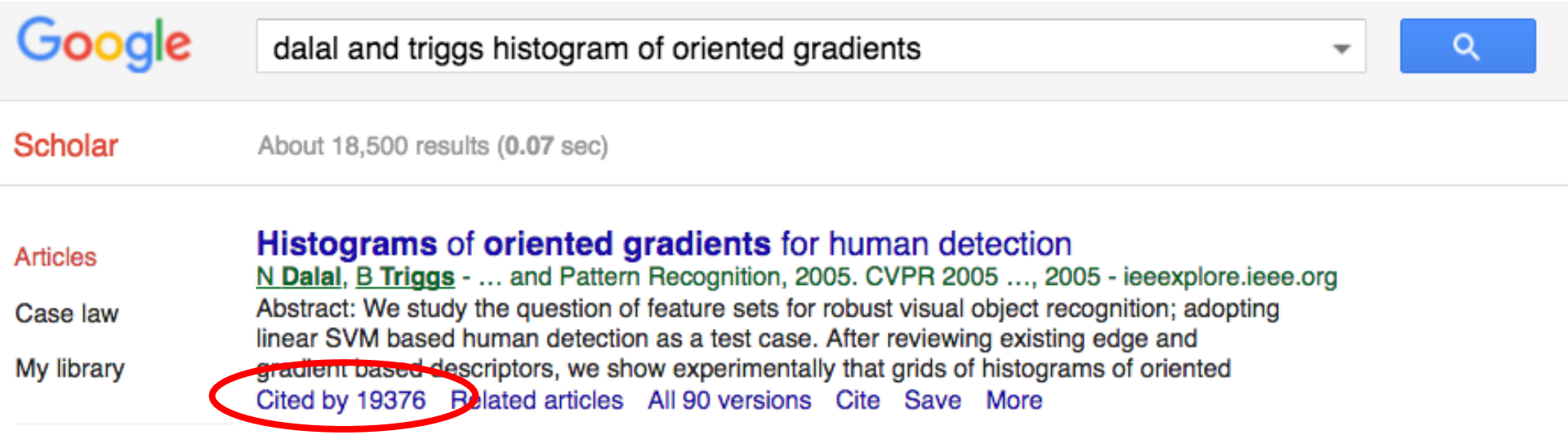


Non-Local Features

Features computed over a larger region or over the whole image

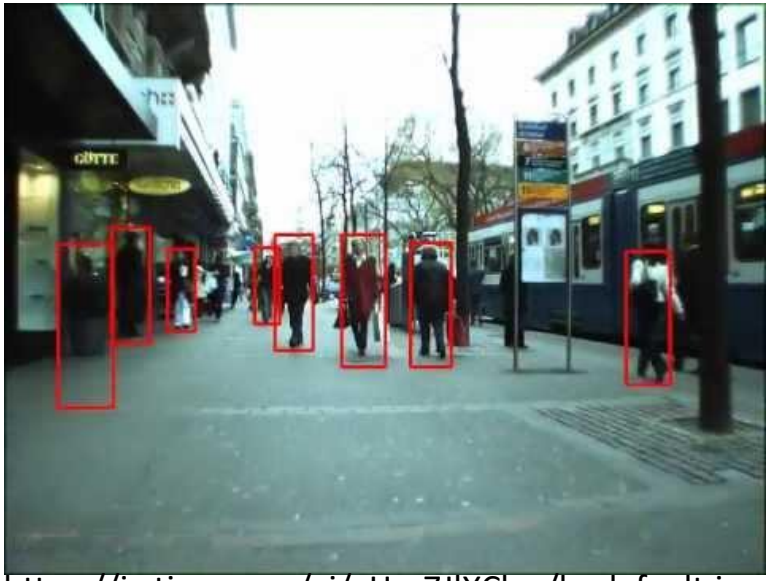
Histogram of Oriented Gradients - HOG

Histogram of Oriented Gradients (HoG)



Histogram of Oriented Gradients (HOG)

- HOG is not a local feature, computed as a descriptor of a point. Its purpose is not to aid in homography estimation or point-by-point image matching
- Instead, it is motivated by the object detection problem
- The original HOG paper was focused on the pedestrian detection problem



https://i.ytimg.com/vi/aHxs7JlXCkw/hqdefault.jpg

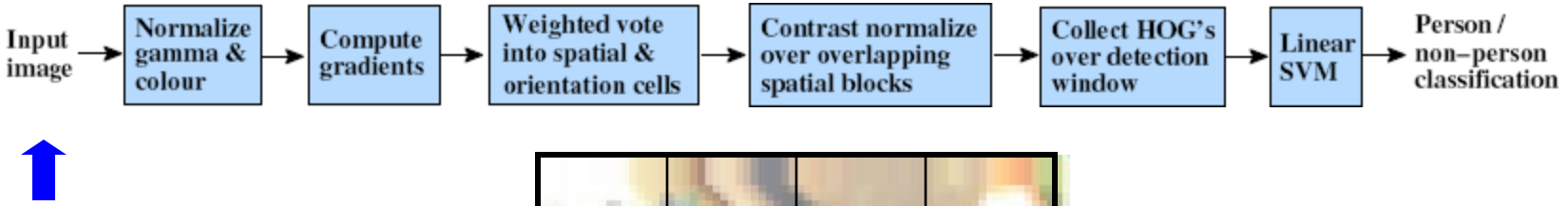
http://www.svcl.ucsd.edu/~ehsan/figs/Pedestrian_detection.jpg

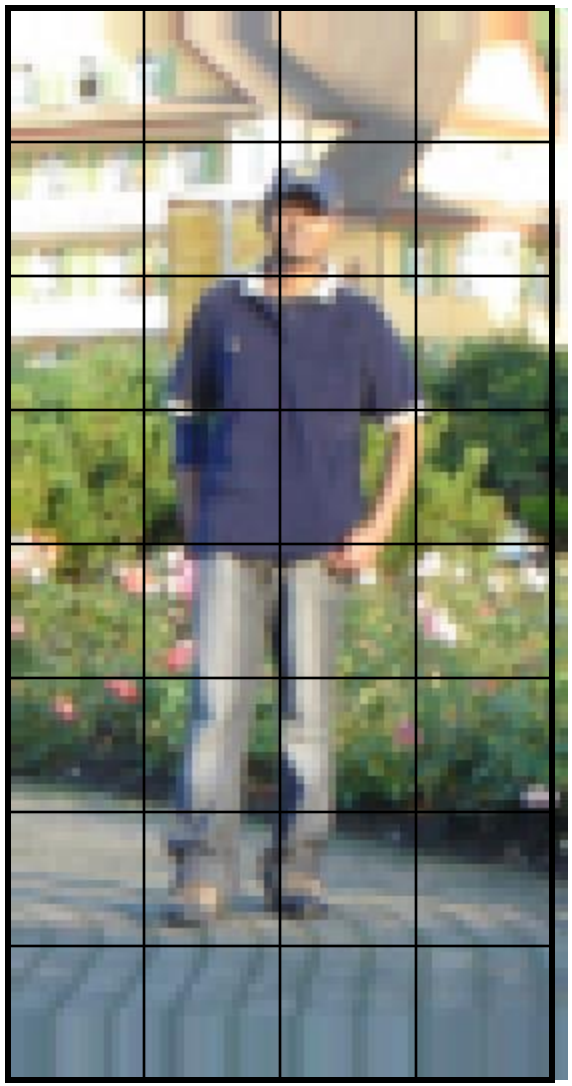
Histogram of Oriented Gradients (HOG)

- HOG Descriptor works on fixed size window (64x128 pixels)
- It is used in a sliding window fashion over the image, to compute this descriptor at all locations
- Image is resized at different scales, so that persons of different sizes can be extracted
- The computed feature vector is passed to a classifier to decide whether it is a person or not.

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Adapted from James Hays slides

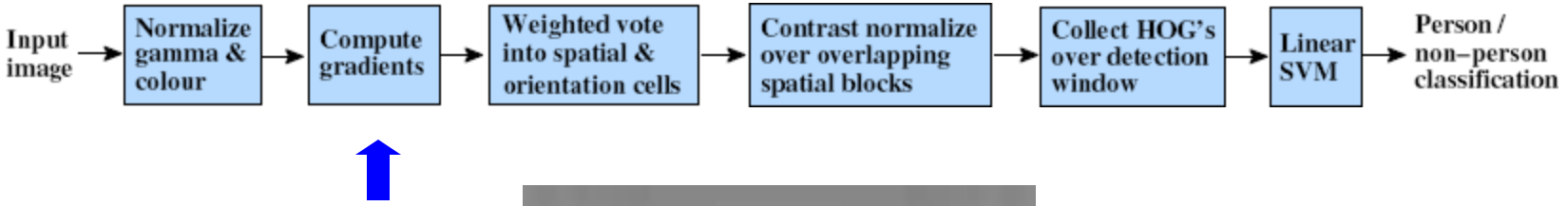




Steps to Calculate HOG Descriptor

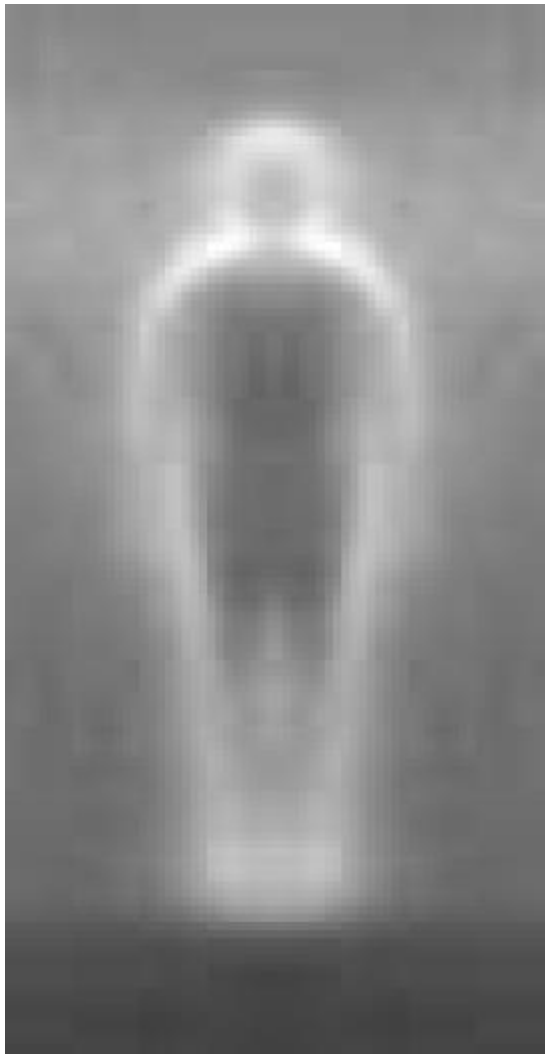
- Image Loading and Preprocessing:
  - Convert the input image to grayscale (if it's colored) and resize it to a standard dimension.
  - Apply gamma normalization to standardize brightness and contrast.
- Calculate Gradients:
  - Use the Sobel filter to calculate horizontal (Gx) and vertical (Gy) gradients for each pixel.
  - Compute the gradient magnitude (edge strength) and orientation (edge direction) from Gx and Gy.

Adapted from James Hays slides



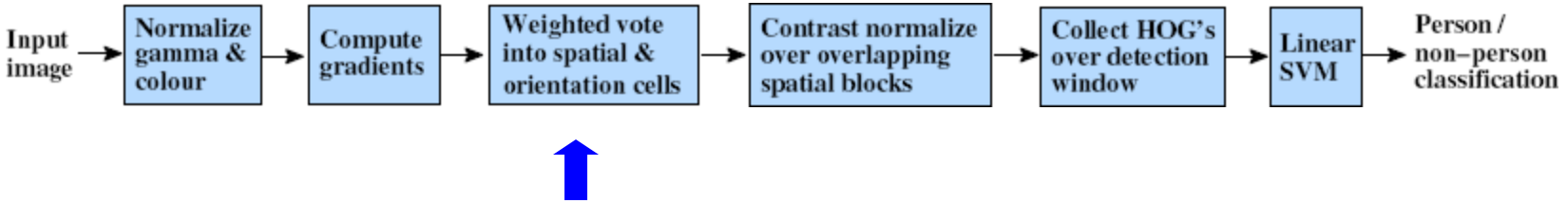
-1	0	1

centered



Steps to Calculate HOG Descriptor

- **Construct Cell Histograms:**
  - Divide the image into non-overlapping 4x4 pixel cells.
  - For each cell, create a histogram with 9 orientation bins (spanning 0° to 180°).
  - Populate each bin by summing the magnitudes of pixels whose gradient orientations fall within the bin's range, capturing the cell's main edge directions.
- **Normalize in Blocks**:
  - Group cells into 2x2 cell blocks (covering 8x8 pixels) to improve robustness to contrast and lighting variations.
  - Concatenate the histograms of the 4 cells in each block and normalize the resulting vector.

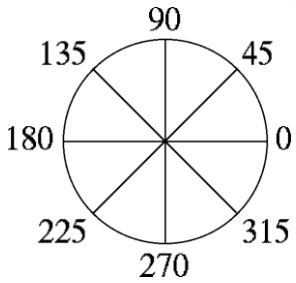


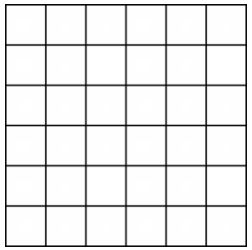
- Histogram of gradient orientations

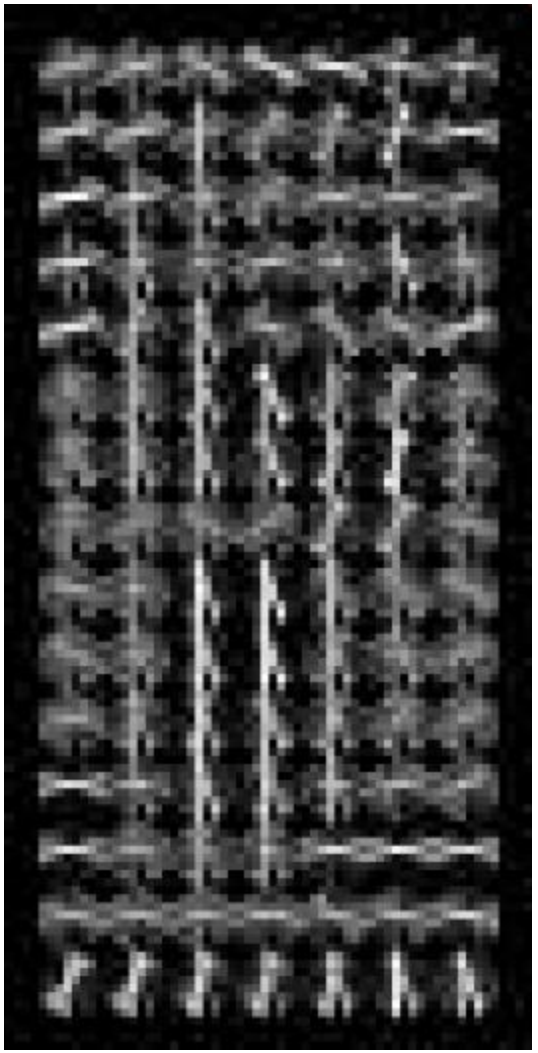
Orientation: 9 bins (for unsigned angles 0 -180)

Histograms in k x k pixel cells

- Votes weighted by magnitude



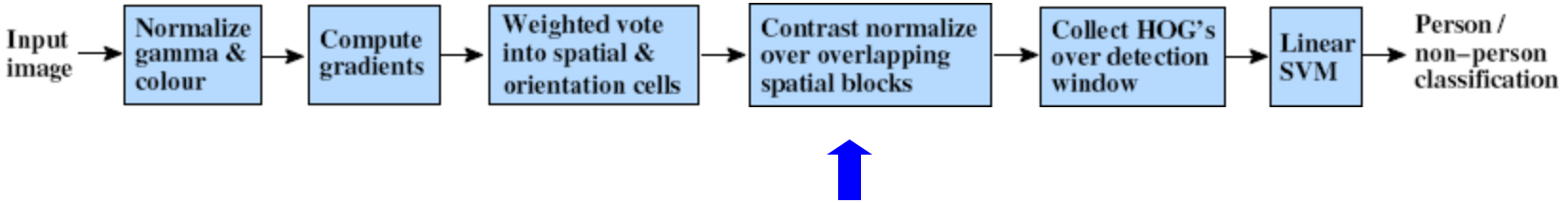




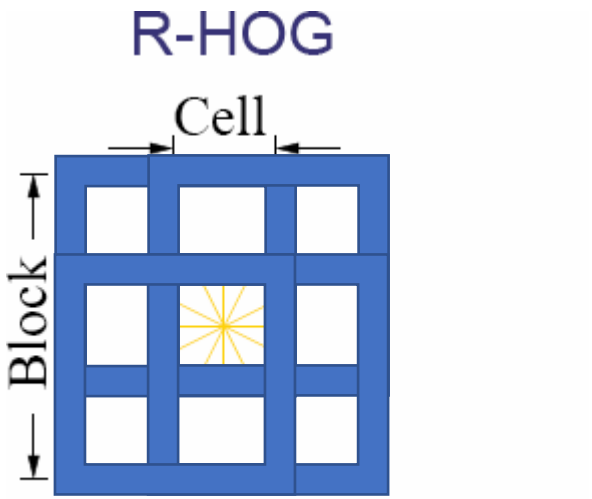
Steps to Calculate HOG Descriptor

- **Form the HOG Descriptor**:
  - Gather all normalized block histograms into a single feature vector, forming the final HOG descriptor, which represents the image's texture and shape.

Adapted from James Hays slides

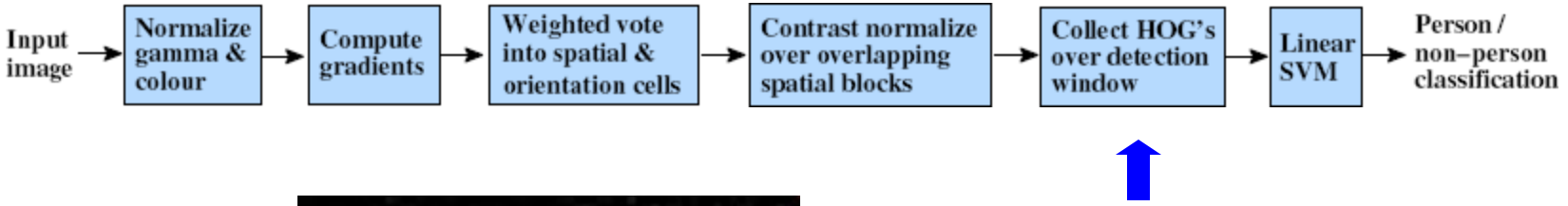


Normalize with respect to surrounding cells

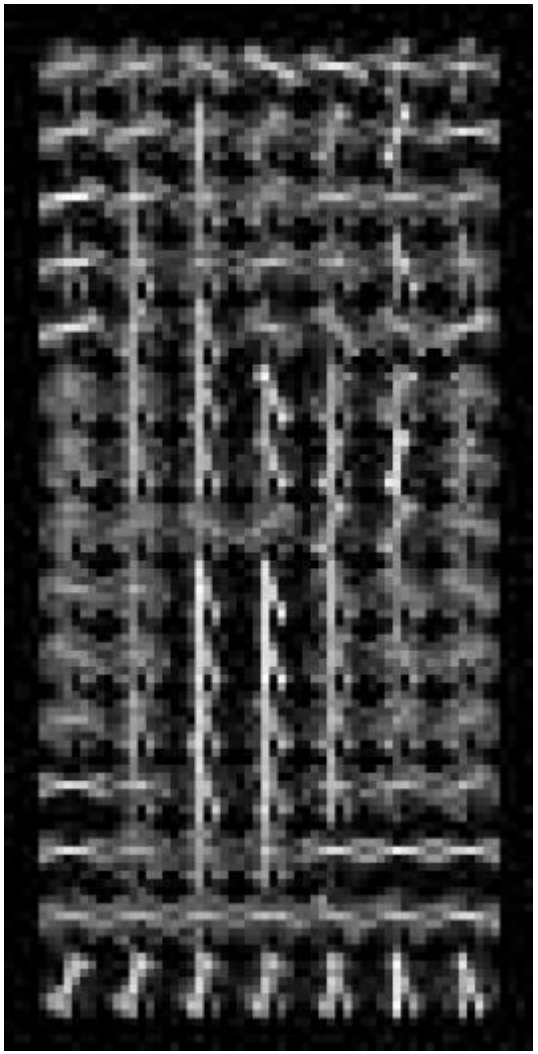


$$L2-norm: v \longrightarrow v/\sqrt{||v||_2^2+\epsilon^2}$$

Adapted from James Hays slides



X=



Original Formulation

orientations

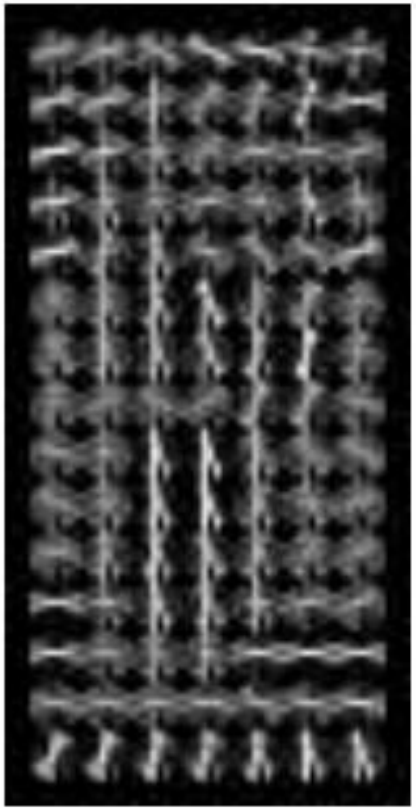
features = 15 x 7 x 9 x 4 = 3780

cells

normalizations by neighboring cells

Histogram of Oriented Gradients (HOG)

Original Image



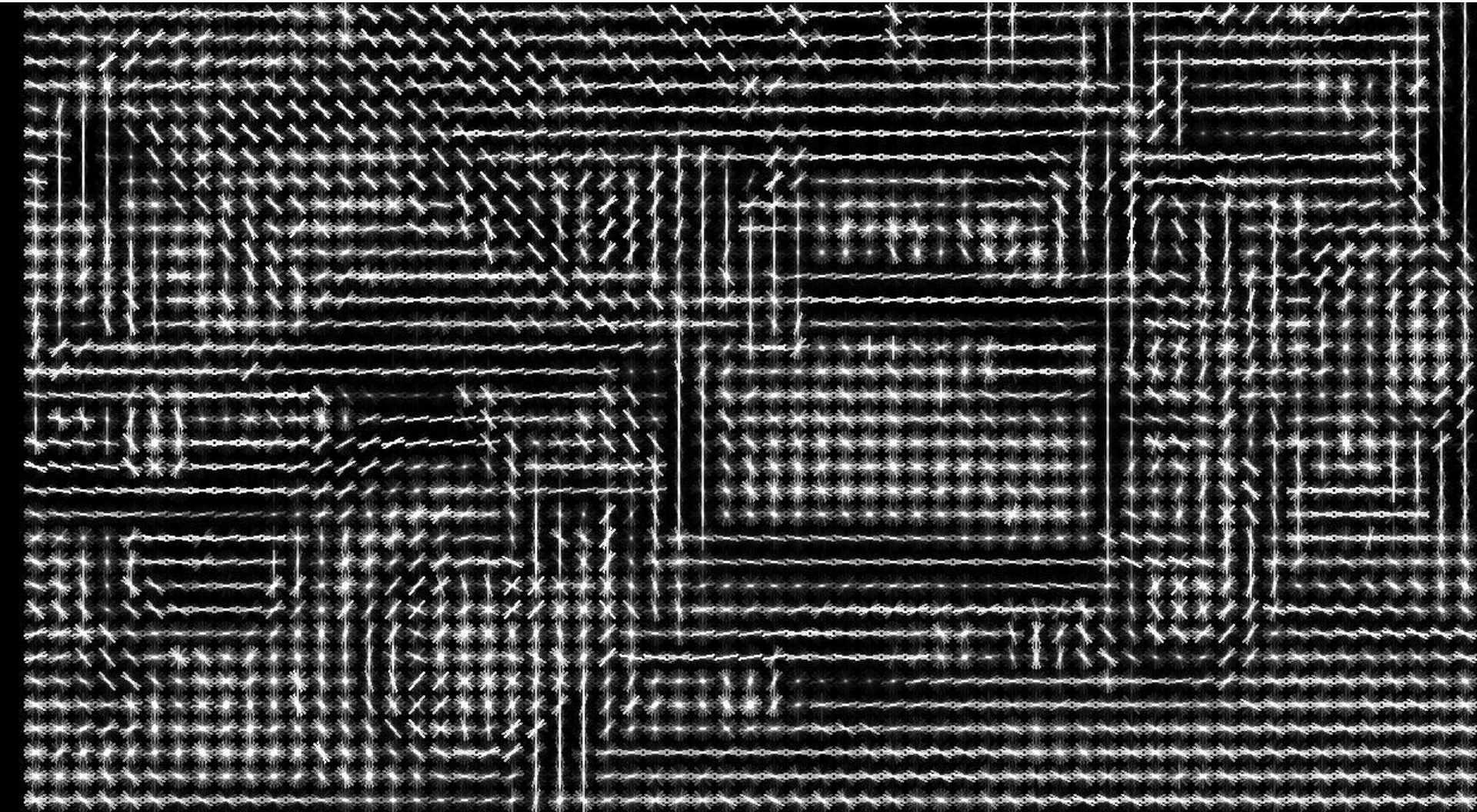
HOG Descriptor

HOG Descriptor weighted by +ve SVM weights

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Histogram of Oriented Gradients (HOG)

HOGgles: http://carlvondrick.com/ihog/



Histogram of Oriented Gradients (HOG)

Ollscoil na Gaillimhe
University of Galway

Sulman Baig: https://www.youtube.com/watch?v=b_DSTuxdGDw

HOG Features

- Are suitable for ‘learning’ the overall shape of an object
- They describe the spatial distribution of image gradients of a shape
- Given lots of examples of a particular object (e.g. pedestrians), we can learn the similarity between their HOG features

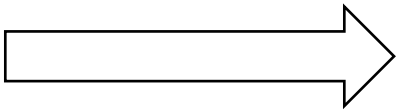
- This is an example of a regional or global feature, which are used for object detection
- General Strategy:
  Find appropriate features + train a classifier

Features Computed by Linear Basis Transformation

Linear Basis

- We are going to consider images as vectors

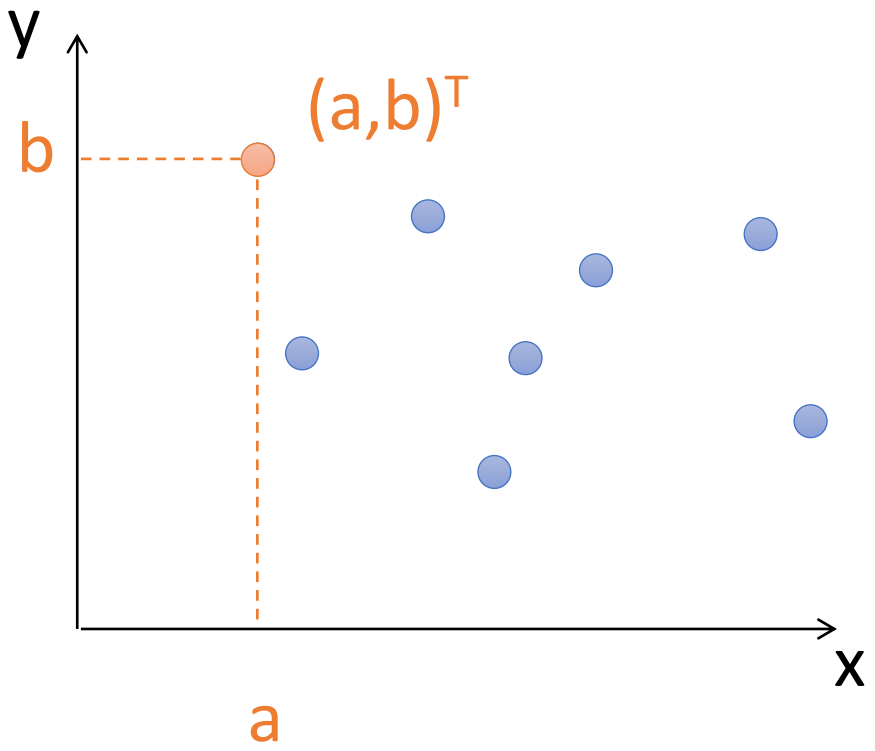
16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1



16
5
9
4
2
11
7
14
3
10
6
15
13
8
12
1

Vector Basis

- A vector is defined as a linear combination of basis vectors



Basis Vectors

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Any vector is written as a linear combination of basis vectors

$$\begin{bmatrix} a \\ b \end{bmatrix} = a \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

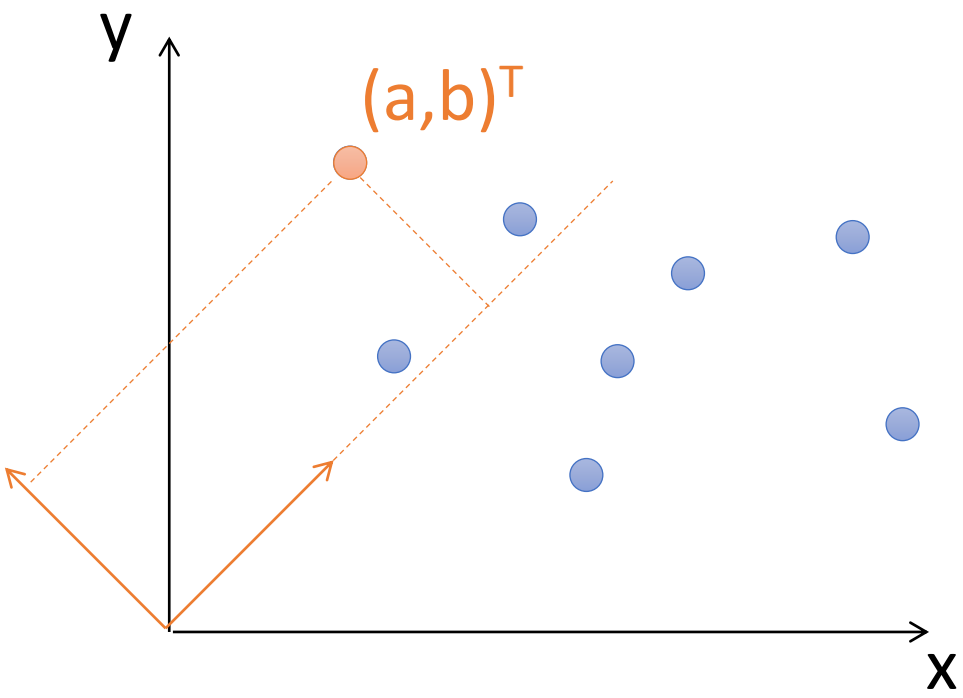
Vector basis

- A set of vectors (v_1, ..., v_k) forms a *basis* in some vector space W if:
  - (1) (v_1, ..., v_k) are linearly independent
  - (2) (v_1, ..., v_k) span W

- The identity matrix is taken to be the standard basis

Change of Basis

- I can represent my vector on some other basis



New Basis Vectors

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \qquad \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

New coordinates

$$a' = \begin{bmatrix} a \\ b \end{bmatrix} \bullet \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$b' = \begin{bmatrix} a \\ b \end{bmatrix} \bullet \begin{bmatrix} \frac{1}{-\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Optimal Basis

- Given a data set of images, I_1, I_2, … I_n as vectors
- Compute the mean image vector μ_I
- Let A be a matrix that contains each image as a column vector
- Form covariance matrix $L = AA^T - \mu_I \mu_I^T$
- Find eigenvectors of L, arranged as a matrix Φ
- Project data onto the basis Φ i.e. rotate by Φ^T
- This will be the most compact representation of data
- Typically, a lot of eigenvalues will be close to zero and can be discarded
- This transformation is called Karhunen-Loève Transform, and the process is called Principal Component Analysis

Basic Face Recognition Example

Face Recognition

- 10 different images of 40 different subjects
- Taken at different times, varying lighting, facial expressions, open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses)

ORL face database: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Face Recognition by Principal Component Analysis

- Training Concept
  - Treat each image as a vector
  - Project vectors into a low-dimensional subspace, by taking only the most significant eigenvectors of the covariance matrix
  - The vector of coefficients for each image are its features
- Recognition Concept
  - Find the mean vector of each person in this low-dimensional subspace
  - For every unknown image, use a nearest mean classifier

Eigen Vectors of Covariance Matrix

- There are 10 images of each person. We will use 9 for training, and then test face recognition performance on the 10^{th} (unseen) image.
- Each image is 112 x 92 pixels. Arrange it into a vector (size 10304 x 1);

1. Arrange all 9 training images of each of 40 persons as columns into a matrix A. Size of A will be 10304 x 360

```
load faceData;

A = [];

%use only 9 faces out of 10 per subject
for i = 1:40
    for j = 1:9
        A(:, end+1) = faceData(i).image(j).im(:);
    end;
end;
```

Eigenvectors of Covariance Matrix

1. Find the mean image of all training images

```
m = (mean(A'))';
figure
imagesc(reshape(m, 112, 92));
```

2. Subtract mean from all vectors, to make A zero mean

```
A = A - m * ones(1, size(A,2));
```

- Lets check mean of A now in the column direction

```
>> norm(mean(A')', 2)
ans =
    2.5633e-12
```

Eigenvectors of Covariance Matrix

- Now we need to find the eigenvalues and eigenvectors of $L = AA^T$
- Size of L: 10304 x 10304
- Smart Computation Trick: Instead of computing eigenvectors of $L = AA^T$, let's consider the eigenvectors of $C = A^TA$
- Claim: Let v_i be an eigenvector of C. Then $\phi_i = Av_i$ is an eigenvector of L
- Proof: If v_i is an eigenvector of C, then

$$\begin{aligned} C\nu_i &= \lambda_i \nu_i \\ A^\top A\nu_i &= \lambda_i \nu_i \\ AA^\top A\nu_i &= \lambda_i (A\nu_i) \\ L(A\nu_i) &= \lambda_i (A\nu_i) \end{aligned}$$

Eigenvectors of Covariance Matrix

4. Compute the eigenvectors C

```
C = A' * A;
[v,d] = eig(C);
```

5. Compute eigenvectors of L from eigenvectors of C

```
vL = A * v;
```

Top 20 eigenvectors of L

Projection and Reconstruction

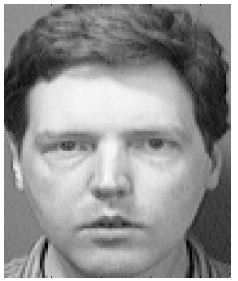
- Finding Coefficients
  - The i th coefficient is computed by taking the projection of the original vector on the i th eigenvector

$$a_i = I \cdot \phi_i$$

- Reconstruction
  - Each face image can be regenerated by a linear combination of eigenvectors weighted by coefficients

$$I' = \Sigma_i a_i \phi_i$$

Projection



- Consider one face image
- Convert to column vector
- Project onto first K eigenvectors (e.g. 20)

$$a_i = I \cdot \phi_i$$

$$\mathbf{a} = [\Phi^\top]_\mathrm{K} I$$

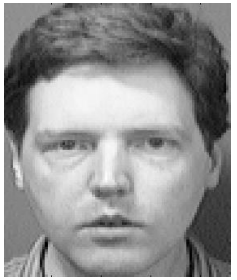
- Subscript K denotes first K rows of Φ^T
- We now have a compact representation of this image, given only by K numbers

```
1e6 x
       177.98
       222.63
      -2.0764
      -27.749
       40.304
       14.775
       19.308
      -14.086
       26.112
      -4.9478
      -21.023
      0.61836
       5.8531
       2.5611
     -0.66498
      -3.1729
     -0.58037
       1.9752
       3.0335
      -3.4609
```

Reconstruction

- Image can be reconstructed by multiplying coefficients with eigenvectors

$$I' = \Sigma_i a_i \phi_i$$



1e6 x

```
  177.98
  222.63
 -2.0764
 -27.749
  40.304
  14.775
  19.308
 -14.086
  26.112
 -4.9478
 -21.023
 0.61836
  5.8531
  2.5611
-0.66498
 -3.1729
-0.58037
  1.9752
  3.0335
 -3.4609
```

Face Recognition by Principal Component Analysis

- Training Concept
  - Treat each image as a vector
  - Project vectors into a low-dimensional subspace, by taking only the most significant eigenvectors of the covariance matrix
  - The vector of coefficients for each image are its features
- Recognition Concept
  - Find the mean vector of each person in this low-dimensional subspace
  - For every unknown image, use a nearest mean classifier

Face Recognition

Training

- Arrange training images in matrix A
- Compute C from A as $C = A^T A$
- Compute eigenvectors of C
- Compute eigenvectors of L from eigenvectors of C
- Select few most significant eigenvectors of L
- Compute coefficients of vectors corresponding to each training image
- For each person, coefficients vectors should form a cluster. Find mean of this vector as a representation of that person

Recognition

- Create a vector of the image to be recognized
- Compute coefficients of this vector by projecting it onto the selected eigenvectors
- Decide which person this image belongs to based on the distance from the nearest mean vector of persons in training data

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Dr Waqar Shahid Qureshi
Waqarshahid.qureshi@universityofgalway.ie

Thanks