# Interest Point Detectors

## Sift Invariant Feature Transform (SIFT)
## and
## Histogram of Gradient (HOG)

Some slides were adapted/taken from various sources, including 3D Computer Vision of Prof. Hee, NUS, Air Lab Summer School, The Robotic Institute, CMU, Computer Vision of Prof. Mubarak Shah, UCF, Computer Vision of Prof. William Hoff, Colorado School of Mines and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and **NOT** to distribute it.

# Finding the “same” thing across images

**Categories** Find a bottle:

Can’t do
unless you do not
care about few errors…

**Instances** Find these two objects

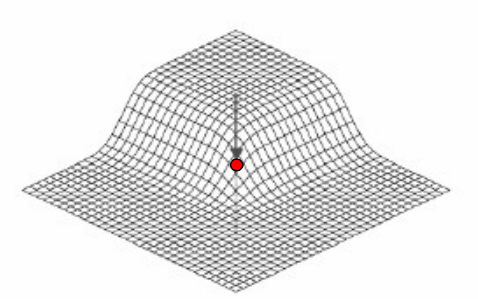

Can nail it

# Building a Panorama

M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003

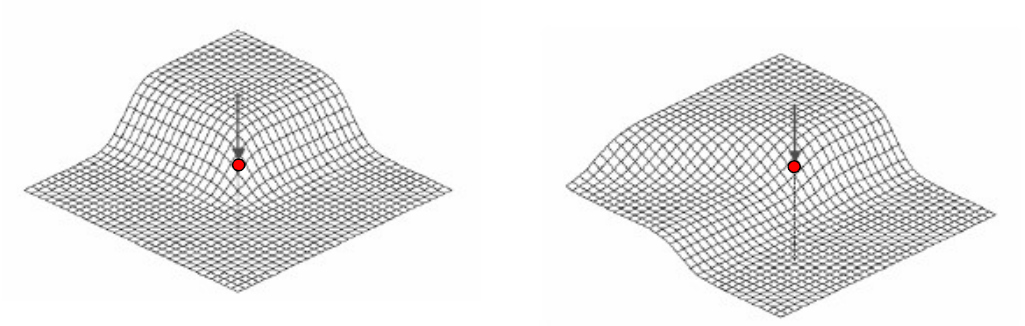
# What is an interest point

- Expressive texture
  - The point at which the direction of the boundary of object changes abruptly
  - Intersection point between two or more edge segments

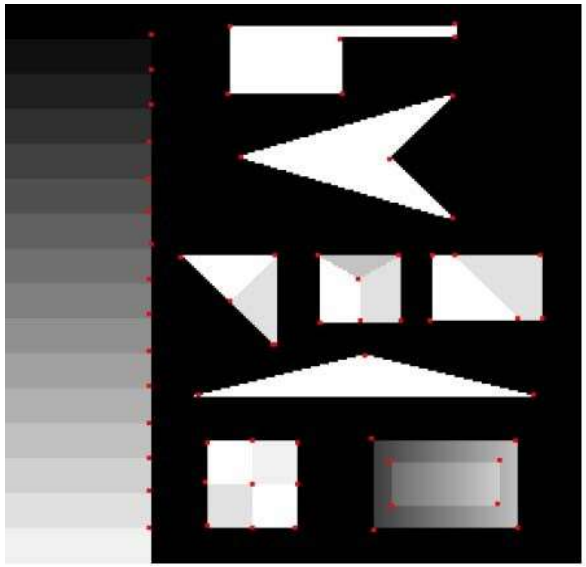


# What is an interest point

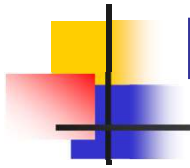
- Expressive texture
  - The point at which the direction of the boundary of object changes abruptly
  - Intersection point between two or more edge segments



# Synthetic & Real Interest Points

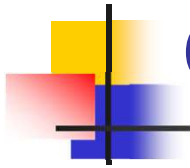


Corners are indicated in red



# Properties of Interest Point Detectors

- Detect all (or most) true interest points
- No false interest points
- Well localized.
- Robust with respect to noise.
- Efficient detection



# Possible Approaches to Corner Detection

- Based on brightness of images
  - Usually image derivatives
- Based on boundary extraction
  - First step edge detection
  - Curvature analysis of edges

# Image Derivative and Average

- Derivative: Rate of change
  - *Speed* is a rate of change of a *distance*
  - *Acceleration* is a rate of change of *speed*
- Average (Mean)
  - Dividing the sum of $N$ values by $N$

# Derivative

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

$$v = \frac{ds}{dt} \text{ speed} \qquad a = \frac{dv}{dt} \text{acceleration}$$

# Discrete Derivative

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

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

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

# Discrete Derivative

## Finite Difference

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

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

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

# Example: 1D
## (Backward Difference)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $f(x) =$ | 10 | 15 | 10 | 10 | 25 | 20 | 20 | 20 |
| $f'(x) =$ | 0 | 5 | −5 | 0 | 15 | −5 | 0 | 0 |
| $f''(x) =$ | 0 | 5 | −10 | 5 | 15 | 20 | 5 | 0 |

Derivative Masks

| | |
|---|---|
| Backward difference | [-1 1] |
| Forward difference | [1 -1] |
| Central difference | [-1 0 1] |

# Derivative in 2D

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_x}{f_y}$$

# Derivatives of Images

Derivative masks

$$f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

X direction Centre difference

$$f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

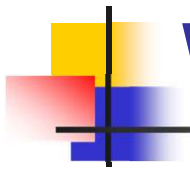
Y direction Centre difference

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Derivatives of Images

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

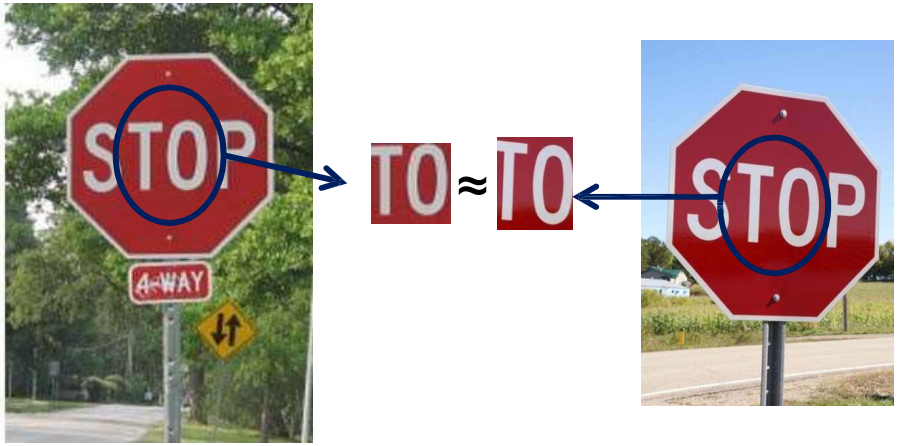


# Where can we use it?

- Automate object tracking
- Point matching for computing disparity
- Stereo calibration
  - Estimation of fundamental matrix
- Motion based segmentation
- Recognition
- 3D object reconstruction
- Robot navigation
- Image retrieval and indexing

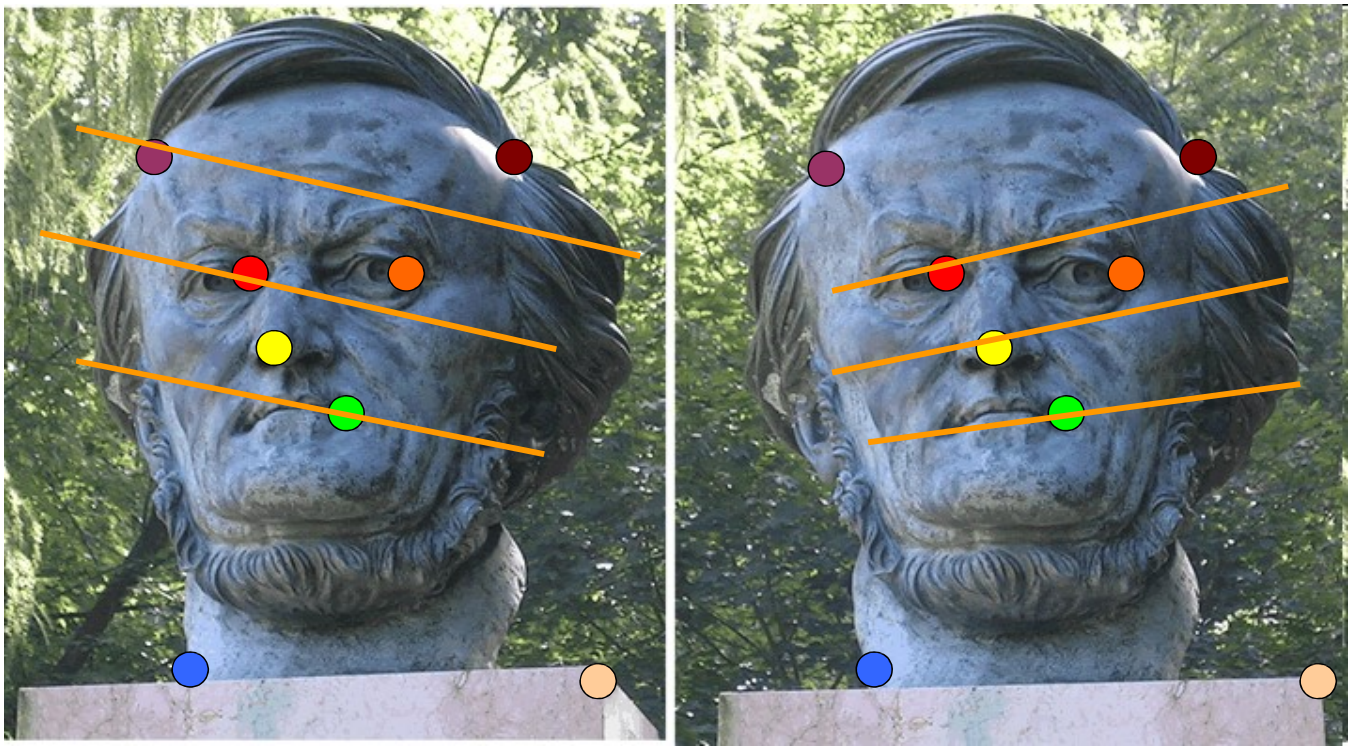
# Correspondence across views

- Correspondence: matching points, patches, edges, or regions across images



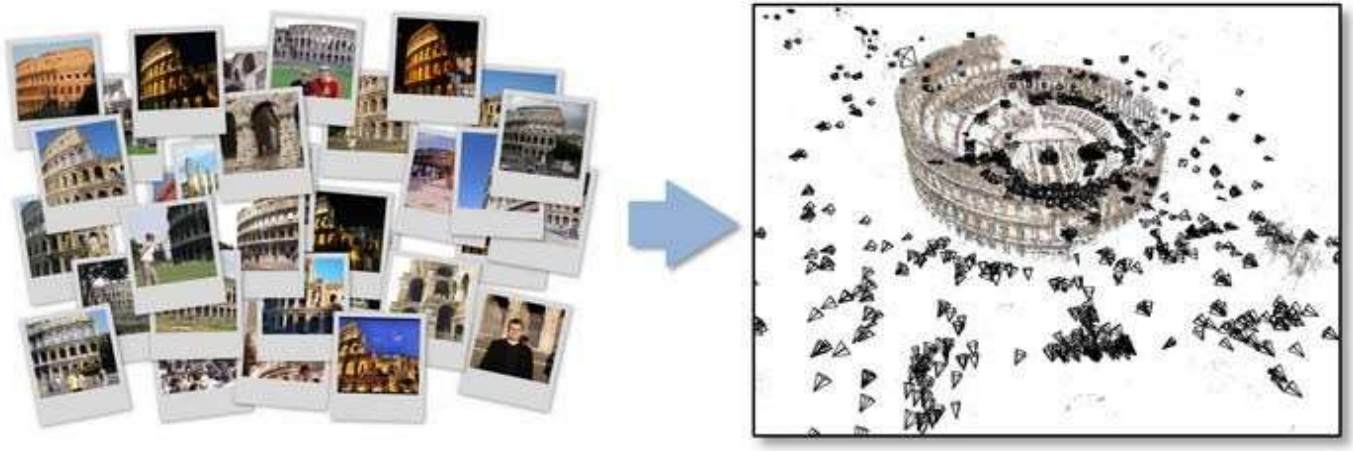

Slide Credit: James Hays

# Example: estimating “fundamental matrix” that corresponds two views



Slide from Silvio Savarese

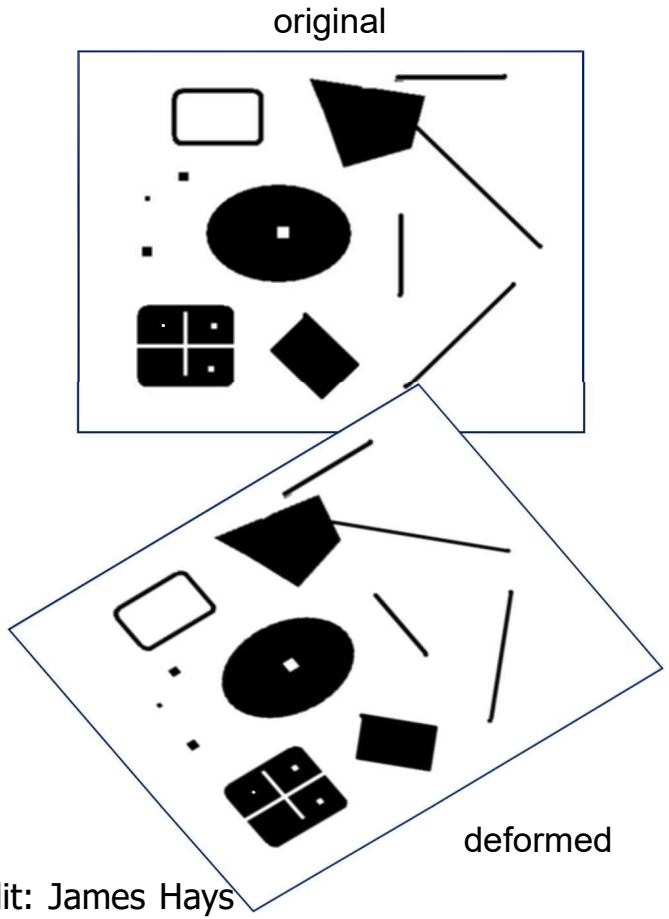
# Example: structure from motion



Slide Credit: James Hays

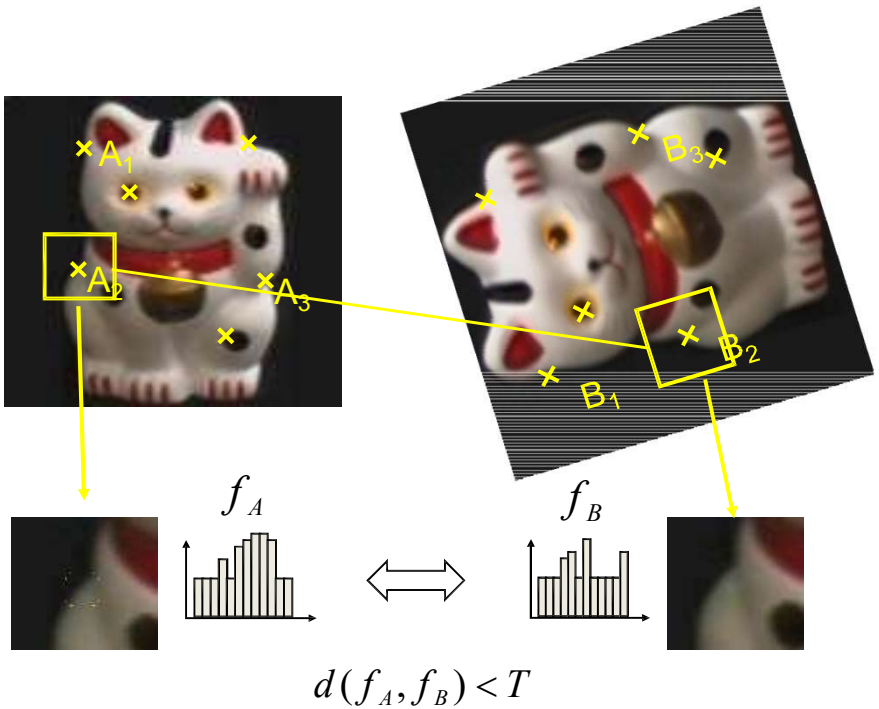
# This class: interest points

- Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
  - Which points would you choose?




Slide Credit: James Hays

# Overview of Keypoint Matching

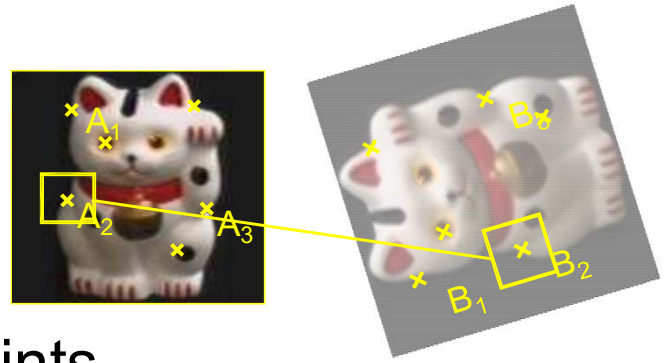



1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

# Goals for Keypoints

Detect points that are *repeatable* and *distinctive*

# Key trade-offs




## Detection of interest points

More Repeatable
- Robust detection
- Precise localization

More Points
- Robust to occlusion
- Works with less texture

## Description of patches

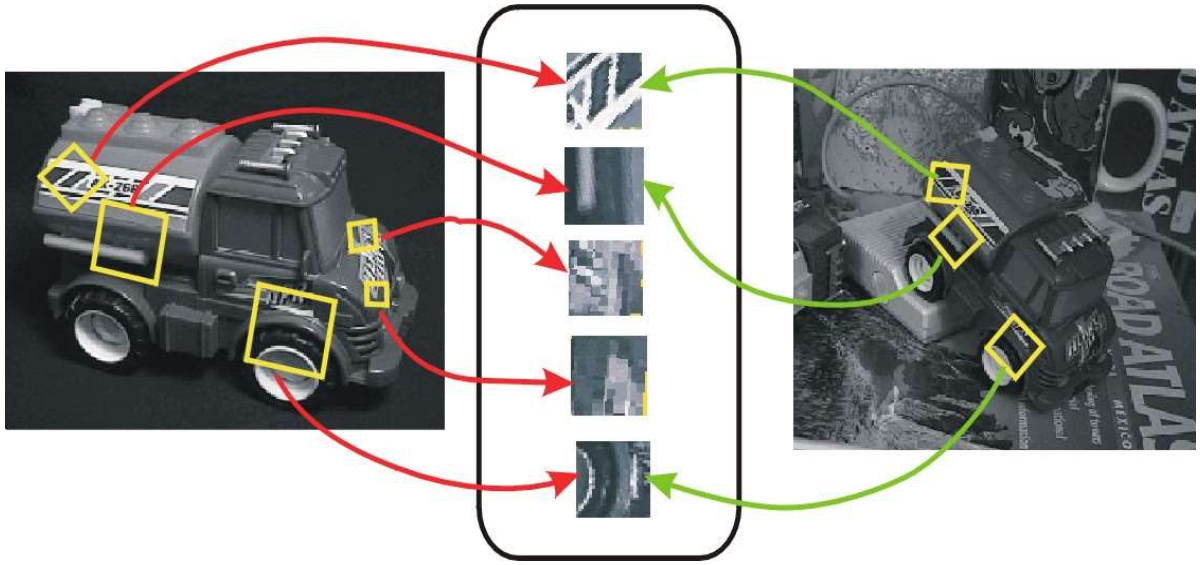
More Distinctive
- Minimize wrong matches

More Flexible
- Robust to expected variations
- Maximize correct matches

Slide Credit: James Hays

# Invariant Local Features

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



**Features Descriptors**

# Choosing interest points

Where would you tell your friend to meet you?

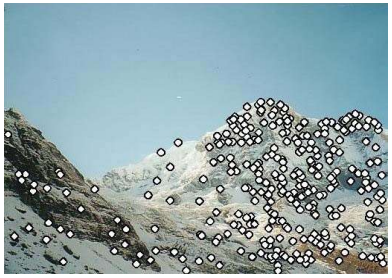
Slide Credit: James Hays

# Feature extraction: Corners


Slides from Rick Szeliski, Svetlana Lazebnik, and Kristin Grauman

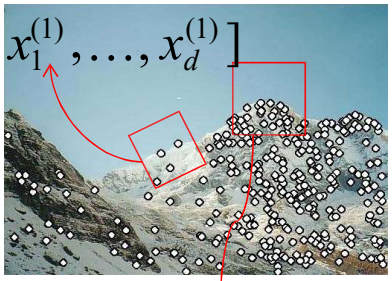
# Local features: main components

1) Detection: Identify the interest points



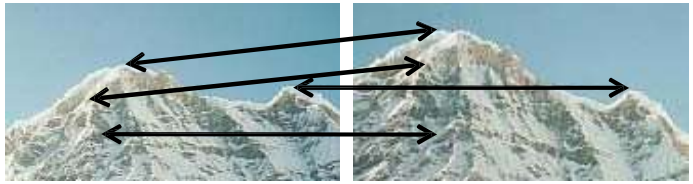
2) Description :Extract feature vector descriptor surrounding each interest point.

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



3) Matching: Determine correspondence between descriptors in two views

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



# Goal: interest operator repeatability

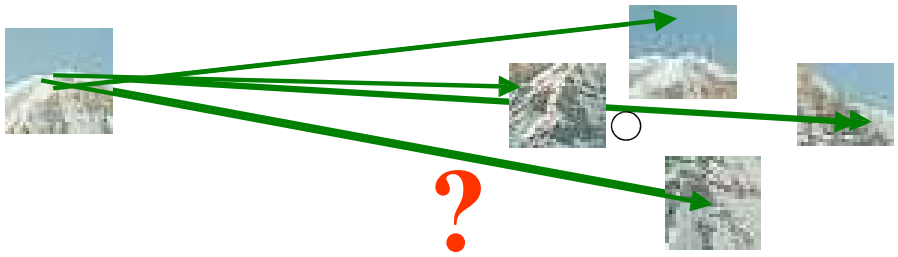
- We want to detect (at least some of) the same points in both images.

No chance to find true matches!

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

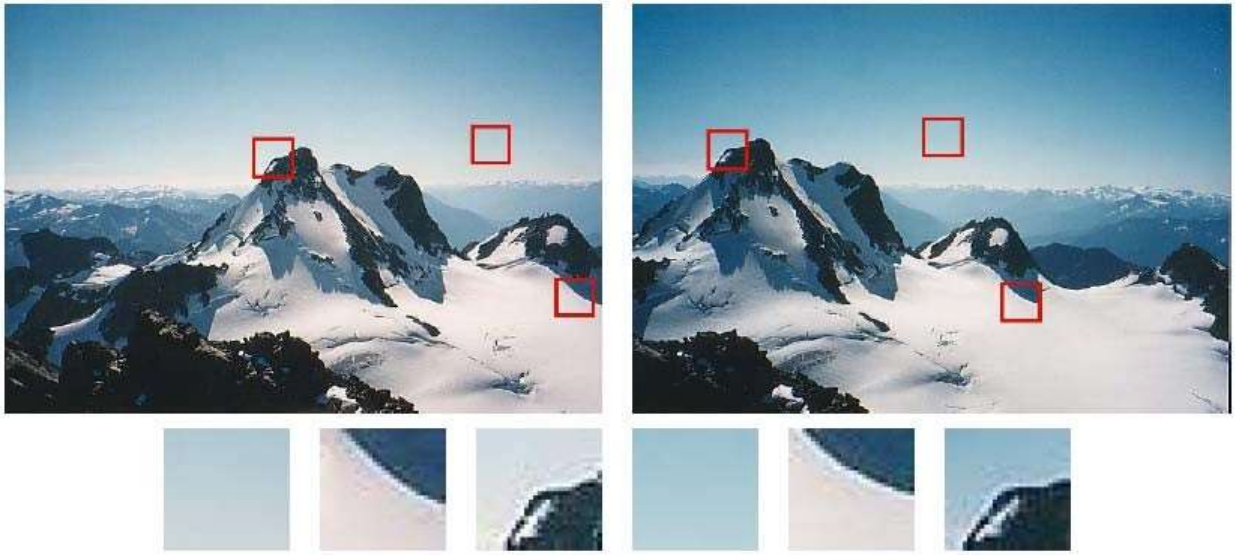
# Goal: descriptor distinctiveness

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

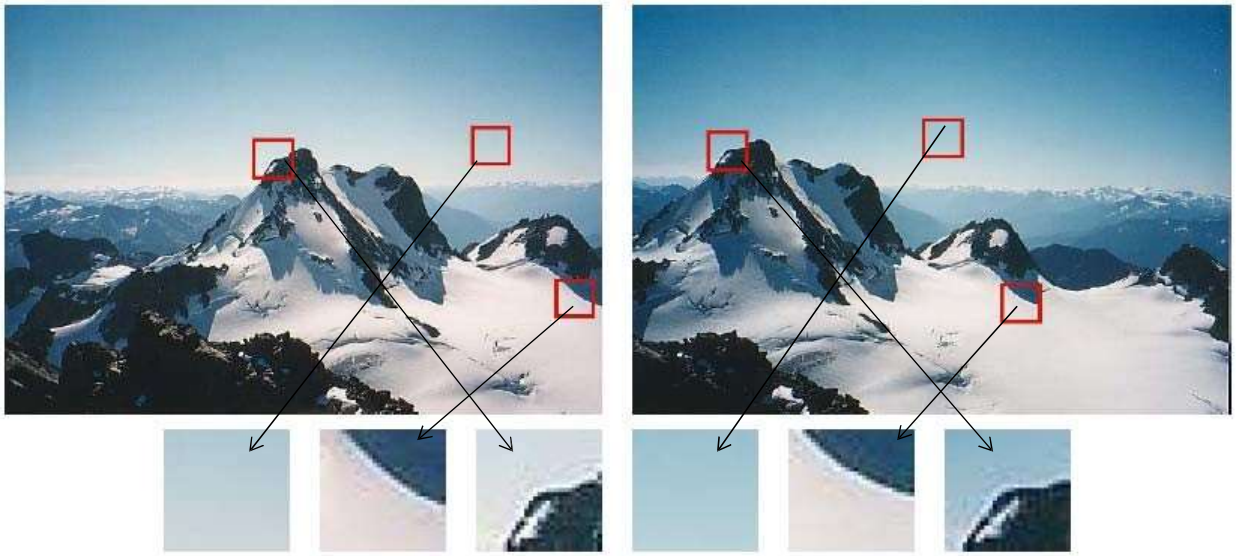



- Must provide some invariance to geometric and photometric differences between the two views.

Some patches can be localized or matched with higher accuracy than others.



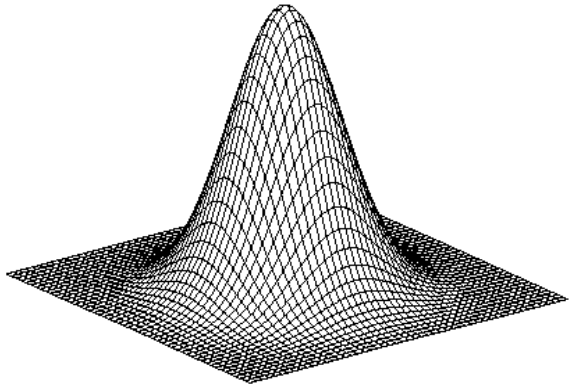
Some patches can be localized or matched with higher accuracy than others.



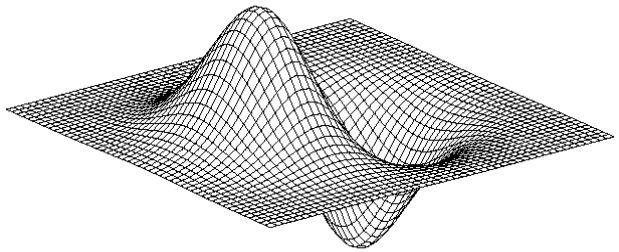
To continue…

# Some Mathematical Preliminaries

# Derivative of Gaussian filter

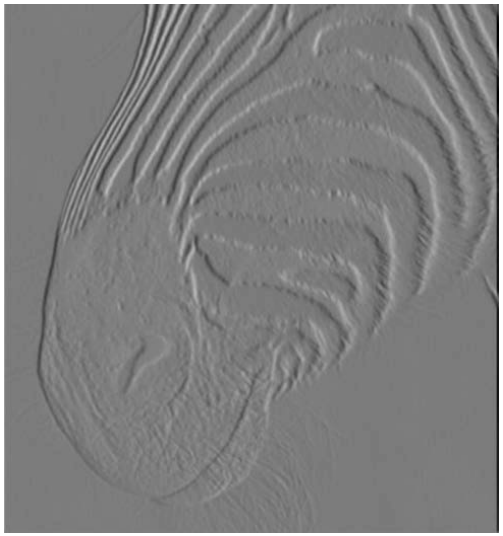


* [1 -1] =



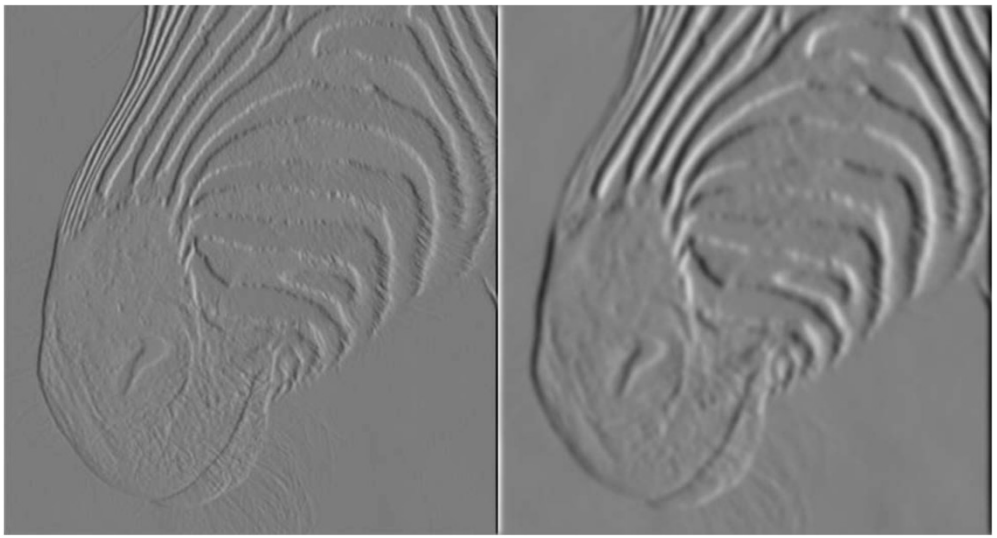
# Tradeoff between smoothing and localization

# Tradeoff between smoothing and localization



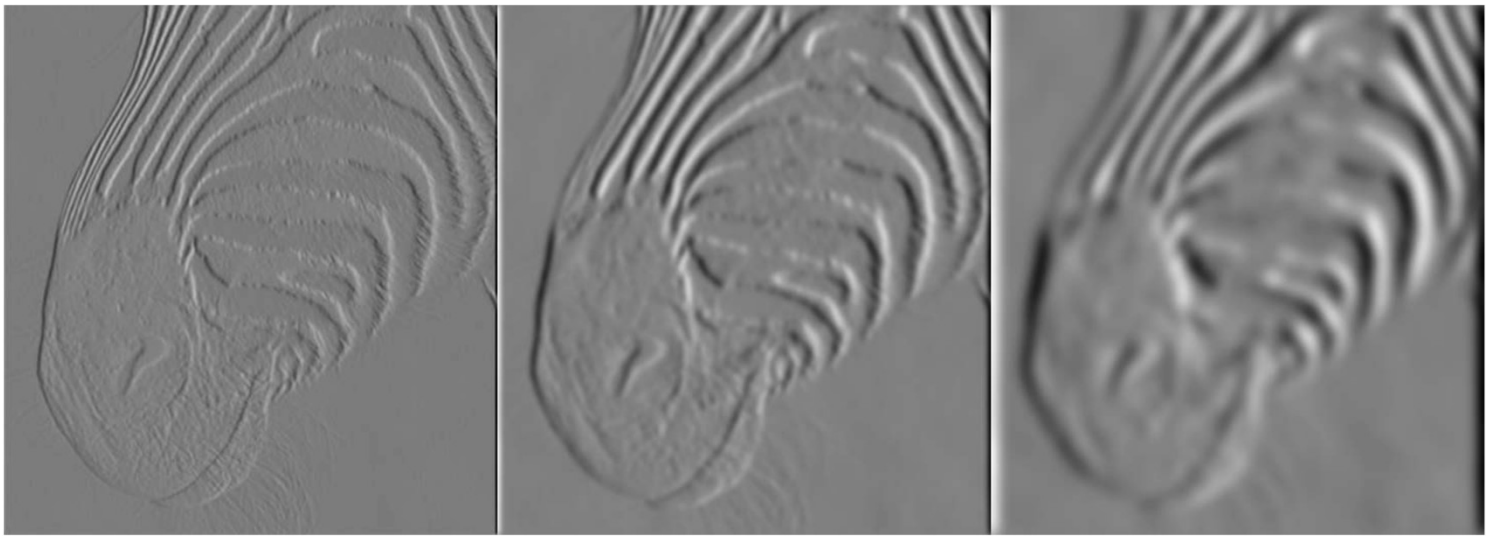
1 pixel

# Tradeoff between smoothing and localization



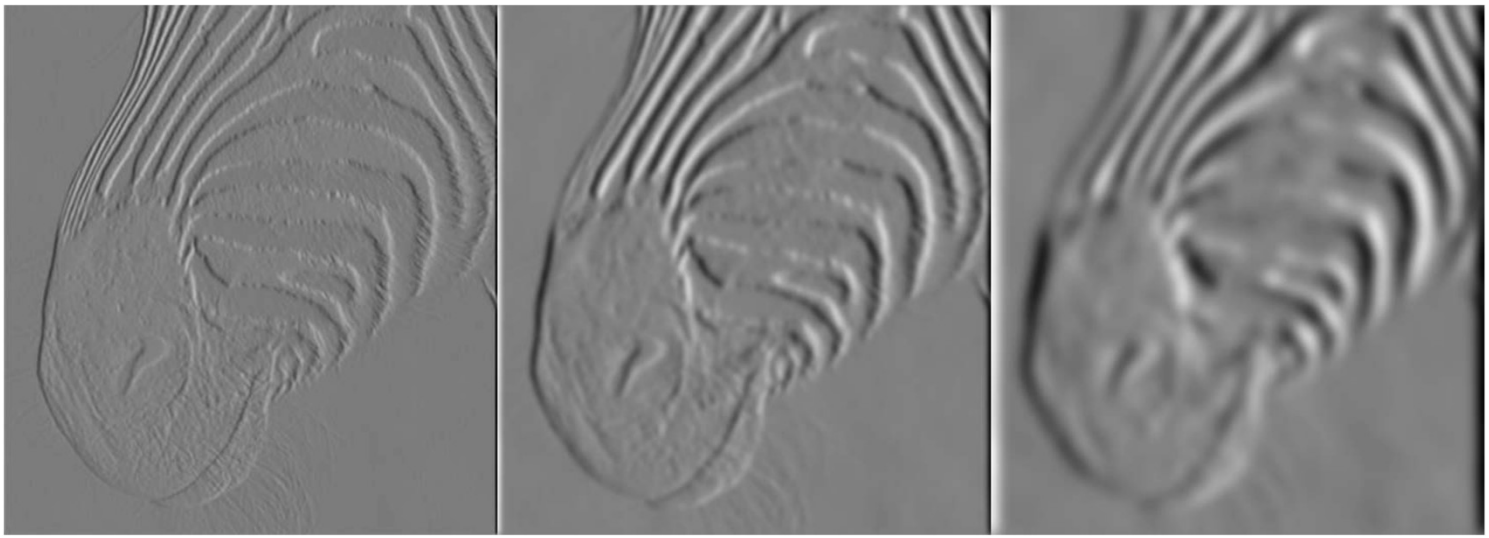
1 pixel 3 pixels

# Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

# Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

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

# Detecting Discontinuities

- Image derivatives

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon) - f(x)}{\varepsilon}\right) \Rightarrow \frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}) - f(x)}{\partial x}$$

- Convolve image with derivative filters

# Detecting Discontinuities

- Image derivatives

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon) - f(x)}{\varepsilon}\right) \Rightarrow \frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}) - f(x)}{\partial x}$$

- Convolve image with derivative filters

| | |
|---|---|
| Backward difference | [-1 1] |
| Forward difference | [1 -1] |
| Central difference | [-1 0 1] |

# Derivative in Two-Dimensions

- Definition
- Approximation
- Convolution kernels

# Derivative in Two-Dimensions

- Definition

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y) - f(x,y)}{\varepsilon}\right)$$

- Approximation

- Convolution kernels

# Derivative in Two-Dimensions

- Definition

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y) - f(x,y)}{\varepsilon}\right) \qquad \frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0}\left(\frac{f(x,y+\varepsilon) - f(x,y)}{\varepsilon}\right)$$

- Approximation

- Convolution kernels

# Derivative in Two-Dimensions

- Definition

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y)-f(x,y)}{\varepsilon}\right) \qquad \frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0}\left(\frac{f(x,y+\varepsilon)-f(x,y)}{\varepsilon}\right)$$

- Approximation

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x_{n+1},y_m)-f(x_n,y_m)}{1}$$

- Convolution kernels

# Derivative in Two-Dimensions

- Definition

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y)-f(x,y)}{\varepsilon}\right) \qquad \frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0}\left(\frac{f(x,y+\varepsilon)-f(x,y)}{\varepsilon}\right)$$

- Approximation

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x_{n+1},y_m)-f(x_n,y_m)}{1} \qquad \frac{\partial f(x,y)}{\partial y} \approx \frac{f(x_n,y_{m+1})-f(x_n,y_m)}{1}$$

- Convolution kernels

# Derivative in Two-Dimensions

- Definition

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y)-f(x,y)}{\varepsilon}\right) \qquad \frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0}\left(\frac{f(x,y+\varepsilon)-f(x,y)}{\varepsilon}\right)$$

- Approximation

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x_{n+1},y_m)-f(x_n,y_m)}{1} \qquad \frac{\partial f(x,y)}{\partial y} \approx \frac{f(x_n,y_{m+1})-f(x_n,y_m)}{1}$$

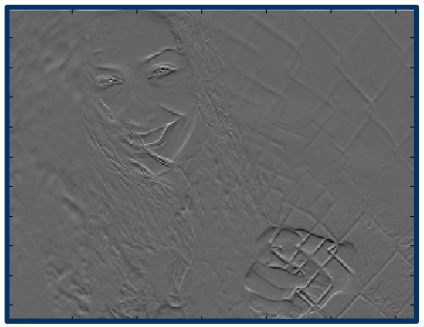
- Convolution

$$f_x = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad f_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

# Image Derivatives

Image $I$

$$I_x = I * \begin{bmatrix} 1 & -1 \end{bmatrix}$$



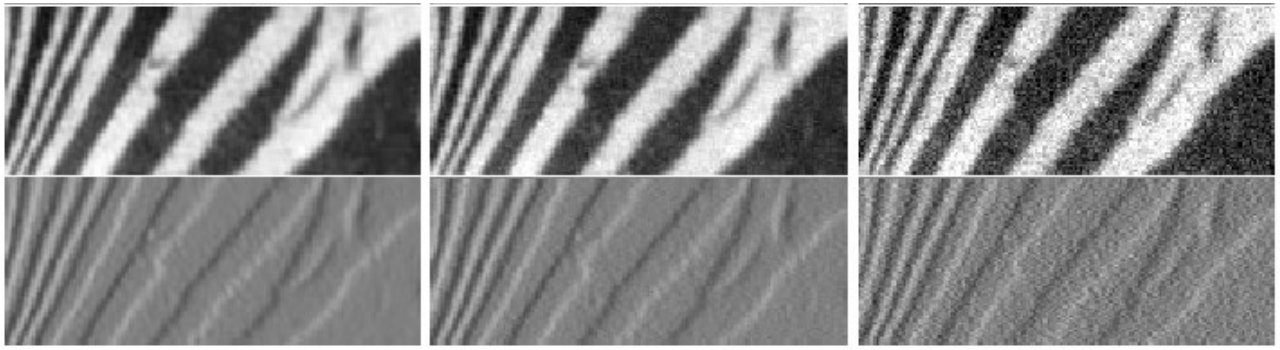
$$I_y = I * \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

# Derivatives and Noise

- Strongly affected by noise
  - obvious reason: image noise results in pixels that look very different from their neighbors
- The larger the noise is the stronger the response

- What is to be done?
  - Neighboring pixels look alike
  - Pixel along an edge look alike
  - Image smoothing should help
    - Force pixels different from their neighbors (possibly noise) to look like neighbors

# Derivatives and Noise



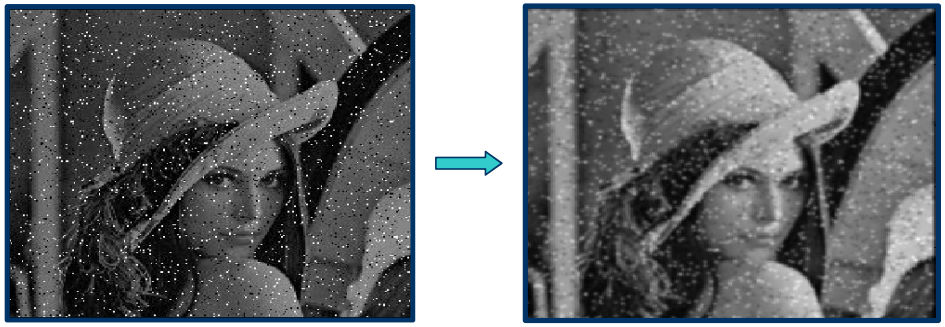
Increasing noise →

Zero mean additive gaussian noise

# Image Smoothing

- Expect pixels to "**be like**" their neighbors
  - Relatively few reflectance changes
- Generally expect noise to be independent from pixel to pixel
  - Smoothing suppresses noise

# Gaussian Smoothing



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

- Scale of Gaussian σ
  - As σ increases, more pixels are involved in average
  - As σ increases, image is more blurred
  - As σ increases, noise is more effectively suppressed

# Marr Hildreth Edge Detector

- Smooth image by Gaussian filter ➔ $S$
- Apply Laplacian to $S$
  - Used in mechanics, electromagnetics, wave theory, quantum mechanics and Laplace equation
- Find zero crossings
  - Scan along each row, record an edge point at the location of zero-crossing.
  - Repeat above step along each column

# Spatial Filters – sharpening

- The *Laplacian*
  - second derivative of a two dimensional function $f(x,y)$

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

$$= [f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)]-4f(x,y)$$

# Spatial Filters – sharpening

- The *Laplacian*
  - use a convolution mask to approximate

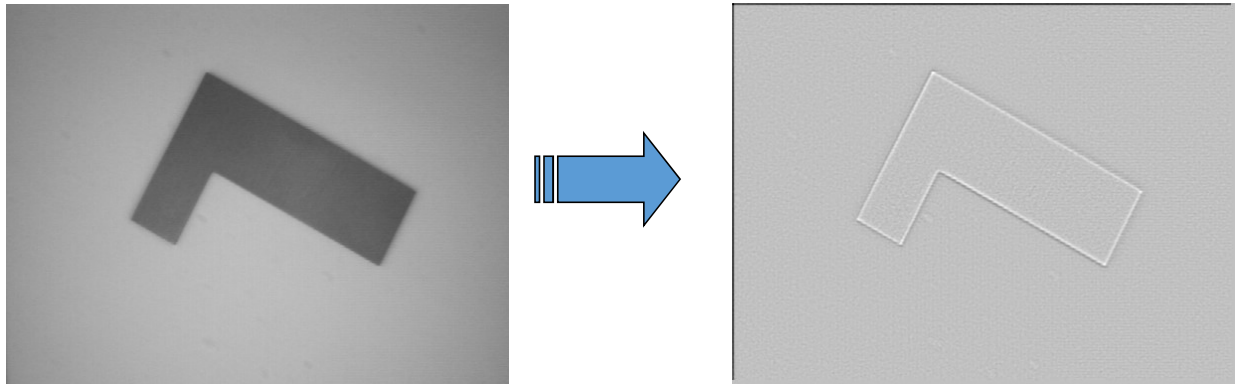
| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

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

| | | |
|---|---|---|
| -1 | 2 | -1 |
| 2 | -4 | 2 |
| -1 | 2 | -1 |

# Spatial Filters – sharpening

- The *Laplacian*
  - example:



# Marr Hildreth Edge Detector

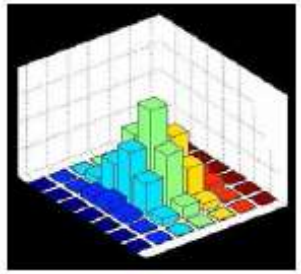
- Smooth image by Gaussian filter ➔ $S$
- Apply Laplacian to $S$
  - Used in mechanics, electromagnetics, wave theory, quantum mechanics and Laplace equation
- Find zero crossings
  - Scan along each row, record an edge point at the location of zero-crossing.
  - Repeat above step along each column

# Marr Hildreth Edge Detector

- Gaussian smoothing

$$\underset{\text{smoothed image}}{\vec{S}} = \underset{\text{Gaussian filter}}{\vec{g}} * \underset{\text{image}}{\vec{I}}$$

$$g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

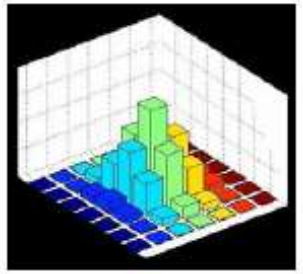


- Find Laplacian

# Marr Hildreth Edge Detector

- Gaussian smoothing

$$\underbrace{\vec{S}}_{\text{smoothed image}} = \underbrace{\vec{g}}_{\text{Gaussian filter}} * \underbrace{\vec{I}}_{\text{image}} \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



- Find Laplacian

$$\Delta^2 S = \overbrace{\frac{\partial^2}{\partial x^2} S}^{\text{second order derivative in } x} + \overbrace{\frac{\partial^2}{\partial y^2} S}^{\text{second order derivative in } y}$$

- $\nabla$ is used for gradient (first derivative)
- $\Delta^2$ is used for Laplacian (Second derivative)

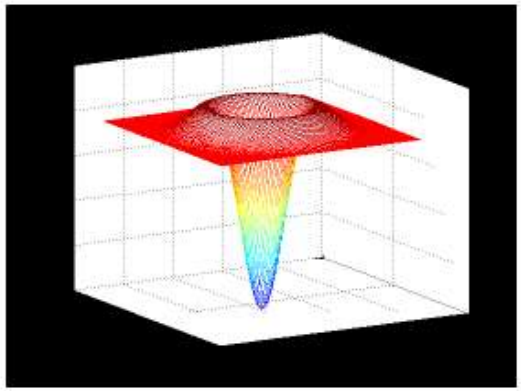
# Marr Hildreth Edge Detector

- Deriving the Laplacian of Gaussian (LoG)

$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$g_x = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \left( -\frac{2x}{2\sigma^2} \right)$$

$$\Delta^2 g = -\frac{1}{\sqrt{2\pi}\sigma^3} \left( 2 - \frac{x^2+y^2}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



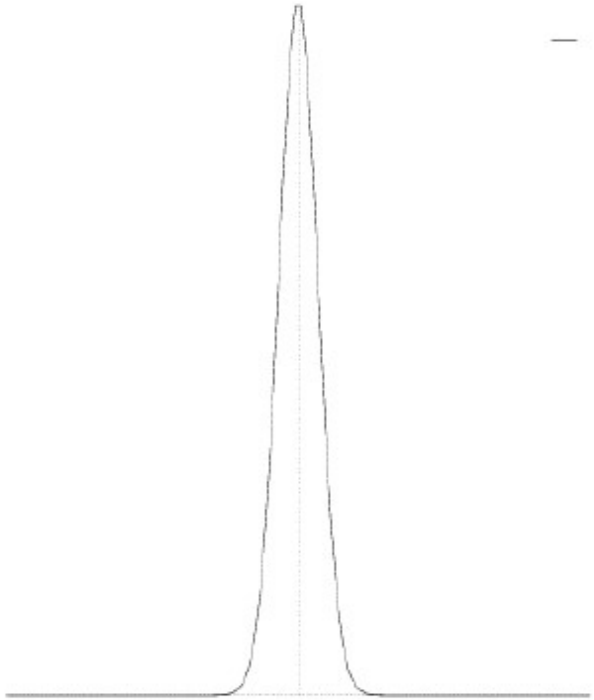
# Gaussian

$$g(x) = e^{\frac{-x^2}{2o^2}}$$

Standard deviation

| x | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| g(x) | .011 | .13 | .6 | 1 | .6 | .13 | .011 |

# Gaussian



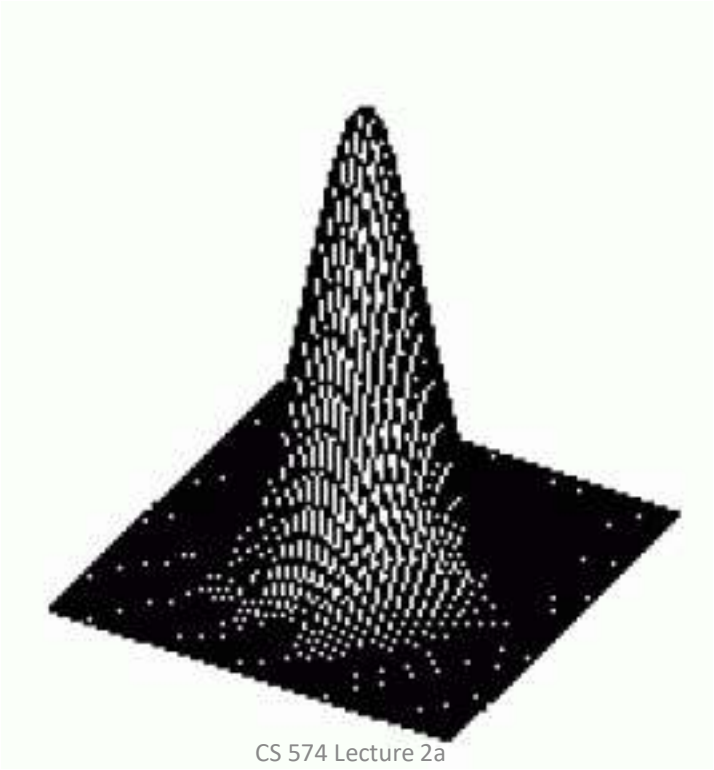
# 2-D Gaussian

$$g(x,y) = e^{\frac{-(x^2+y^2)}{2o^2}}$$

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 3 | 6 | 9 | 11 | 9 | 6 | 3 | 1 | 0 | 0 |
| 0 | 1 | 4 | 11 | 20 | 30 | 34 | 30 | 20 | 11 | 4 | 1 | 0 |
| 0 | 3 | 11 | 26 | 50 | 73 | 82 | 73 | 50 | 26 | 11 | 3 | 0 |
| 1 | 6 | 20 | 50 | 93 | 136 | 154 | 136 | 93 | 50 | 20 | 6 | 1 |
| 2 | 9 | 30 | 73 | 136 | 198 | 225 | 198 | 136 | 73 | 30 | 9 | 2 |
| 2 | 11 | 34 | 82 | 154 | 225 | 255 | 225 | 154 | 82 | 34 | 11 | 2 |
| 2 | 9 | 30 | 73 | 136 | 198 | 225 | 198 | 136 | 73 | 30 | 9 | 2 |
| 1 | 6 | 20 | 50 | 93 | 136 | 154 | 136 | 93 | 50 | 20 | 6 | 1 |
| 0 | 3 | 11 | 26 | 50 | 73 | 82 | 73 | 50 | 26 | 11 | 3 | 0 |
| 0 | 1 | 4 | 11 | 20 | 30 | 34 | 30 | 20 | 11 | 4 | 1 | 0 |
| 0 | 0 | 1 | 3 | 6 | 9 | 11 | 9 | 6 | 3 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 1 | 0 | 0 | 0 | 0 |

$$\sigma = 2$$

# 2-D Gaussian



# LoG Filter

$$\Delta^2 G_\sigma = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2-\frac{x^2+y^2}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

| | | | Y | | | |
|---|---|---|---|---|---|---|
| 0.0008 | 0.0066 | 0.0215 | 0.031 | 0.0215 | 0.0066 | 0.0008 |
| 0.0066 | 0.0438 | 0.0982 | 0.108 | 0.0982 | 0.0438 | 0.0066 |
| 0.0215 | 0.0982 | 0 | -0.242 | 0 | 0.0982 | 0.0215 |
| 0.031 | 0.108 | -0.242 | -0.7979 | -0.242 | 0.108 | 0.031 X |
| 0.0215 | 0.0982 | 0 | -0.242 | 0 | 0.0982 | 0.0215 |
| 0.0066 | 0.0438 | 0.0982 | 0.108 | 0.0982 | 0.0438 | 0.0066 |
| 0.0008 | 0.0066 | 0.0215 | 0.031 | 0.0215 | 0.0066 | 0.0008 |

# Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}
- Slope of zero-crossing {a, -b} is |a+b|.
- To mark an edge
  - compute slope of zero-crossing
  - Apply a threshold to slope

# On the Separability of Gaussian

- Two-dimensional Gaussian can be separated into 2 one-dimensional Gaussians

$$h(x,y) = I(x,y) * g(x,y) \qquad n^2 \text{ multiplications}$$

$$h(x,y) = \left(I(x,y) * g_1(x)\right) * g_2(y) \qquad 2n \text{ multiplications}$$

$$g(x) = e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

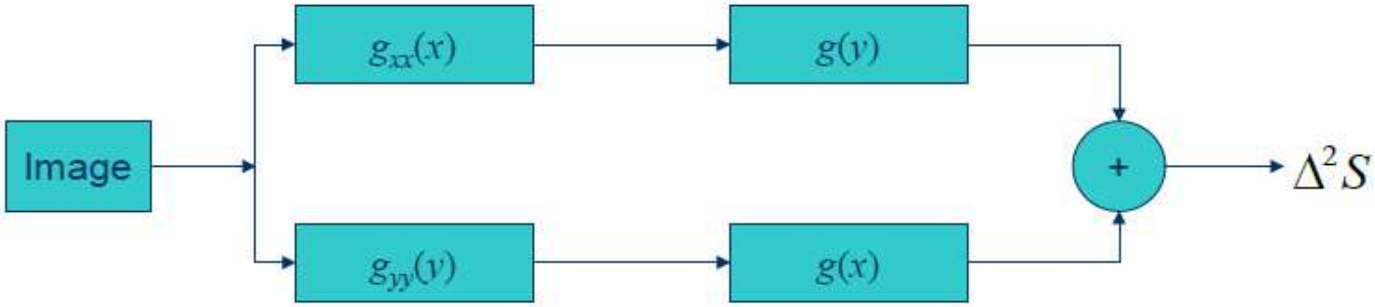
$$g_1 = g(x) = [.011 \quad .13 \quad .6 \quad 1 \quad .6 \quad .13 \quad .011]$$

$$g_2 = g(y) = \begin{bmatrix} .011 \\ .13 \\ .6 \\ 1 \\ .6 \\ .13 \\ .011 \end{bmatrix}$$

# Seperability

Gaussian Filtering


Laplacian of Gaussian Filtering

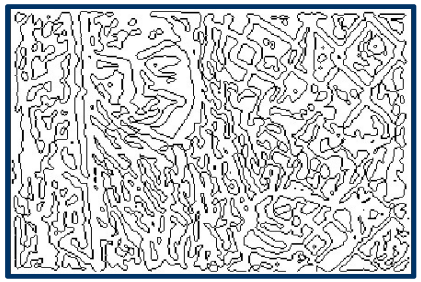


# Example

$I$

$I * \left(\Delta^2 g\right)$

Zero crossings of $\Delta^2 S$



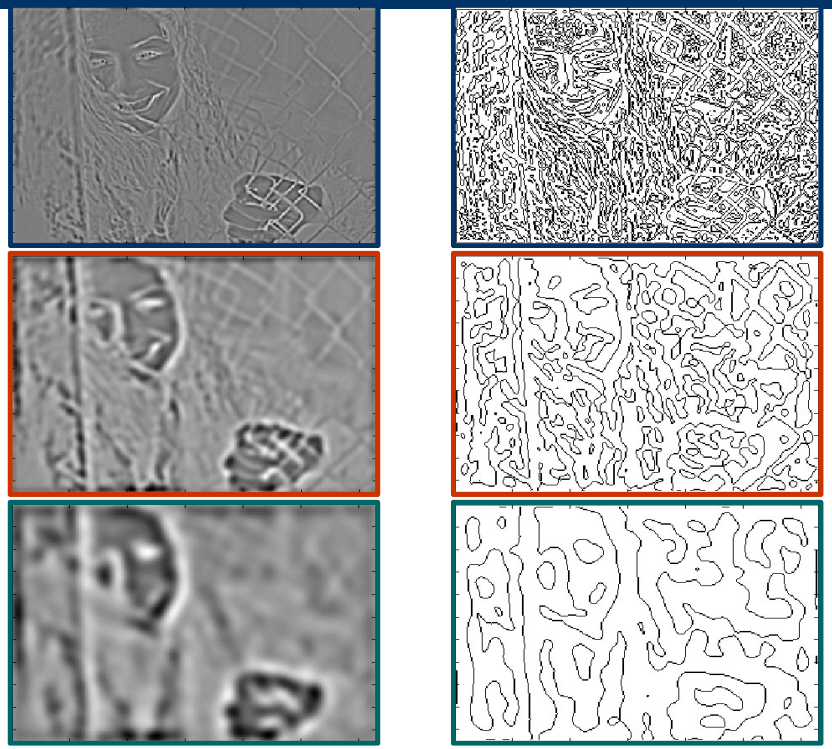
# Example

$$\Delta^2 G_\sigma = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2 - \frac{x^2+y^2}{\sigma^2}\right)e^{\frac{x^2+y^2}{2\sigma^2}}$$

$\sigma = 1$

$\sigma = 3$

$\sigma = 6$

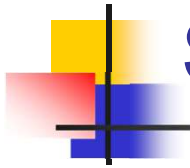


# LoG Algorithm

- Apply LoG to the Image: Either
  - Use 2D filter $\Delta^2 g(x, y)$
  - Or Use 4 1D filters $g(x), g_{xx}(x), g(y), g_{yy}(y)$
- Find zero-crossings from each row
- Find slope of zero-crossings
- Apply threshold to slope and mark edges

# Interest Point Detectors

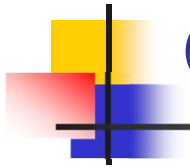
Sift Invariant Feature Transform (SIFT)
and
Histogram of Gradient (HOG)



# SIFT - Key Point Extraction

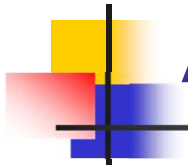
- Stands for Scale Invariant Feature Transform
- Patented by university of British Columbia
- Similar to the one used in primate visual system (human, ape, monkey, etc.)
- Transforms image data into scale-invariant coordinates

D. Lowe. Distinctive image features from scale-invariant key points., International Journal of Computer Vision 2004.



# Goal

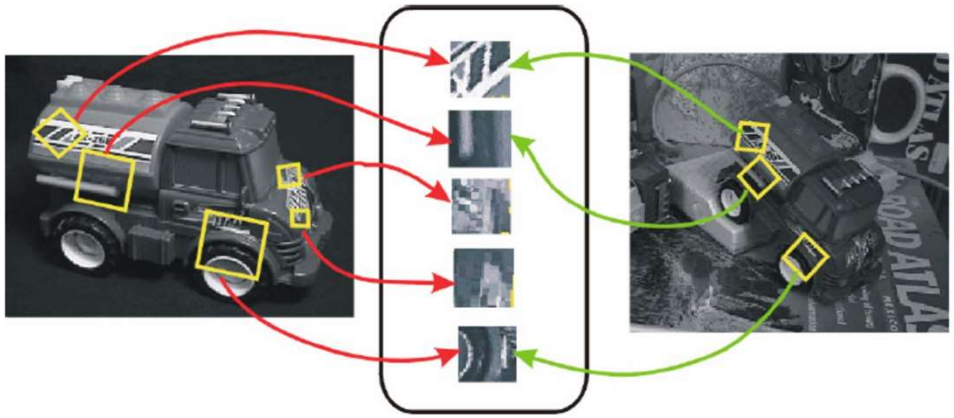
- Extract distinctive invariant features
  - Correctly matched against a large of database features from many images
- Invariance to image scale and rotation
- Robustness to
  - Affine (rotation, scale, shear) distortion,
  - Change in 3D viewpoint,
  - Addition of noise,
  - Change in illumination.



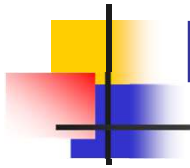
# Advantages

- **Locality:** features are local, so robust to occlusion and clutter
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance

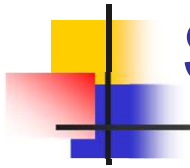
# Invariant Local Features



# Steps for Extracting Key Points (SIF Points)



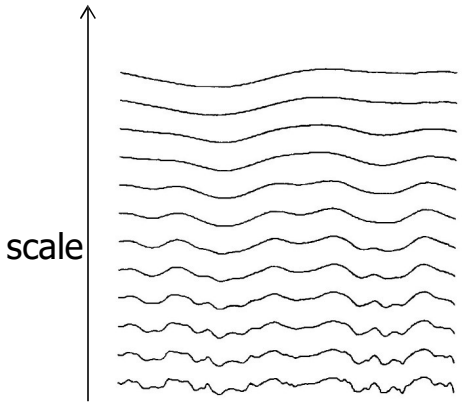
- Scale space peak selection
  - Potential locations for finding features
- Key point localization
  - Accurately locating the feature key points
- Orientation Assignment
  - Assigning orientation to the key points
- Key point descriptor
  - Describing the key point as a high dimensional vector (128) (SIFT Descriptor)



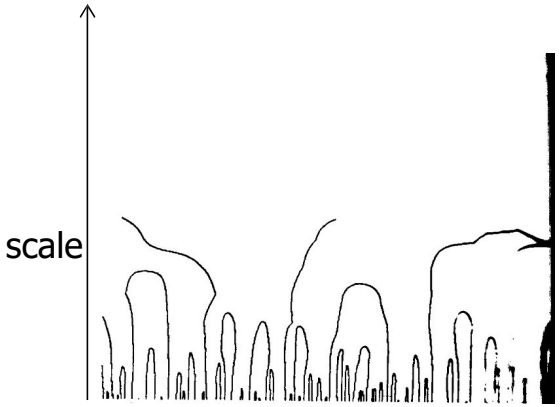
# Scales

- What should be sigma value for Canny and LoG edge detection?
- If use multiple sigma values (scales), how do you combine multiple edge maps?
- Marr-Hildreth:
  - *Spatial Coincidence* assumption:
    - Zerocrossings that coincide over scales several are physically significant.

# Scale Space

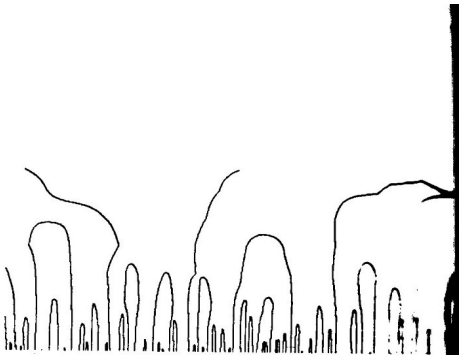



Multiple smooth versions of a signal

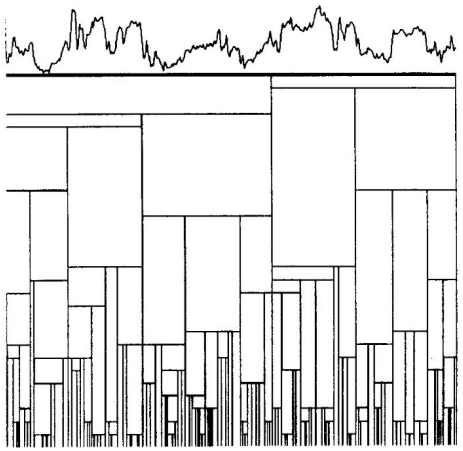



Zerocrossings at multiple scale

# Scale Space



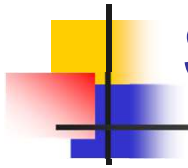
Scale Space



Interval Tree

# Scale Space (Witkin, IJCAI 1983)

- Apply whole spectrum of scales
- Plot zerocrossings vs scales in a scale-space
- Interpret scale space contours
  - Contours are arches, open at the bottom, closed at the top
  - Interval tree
    - Each interval corresponds to a node in a tree,
      - whose parent node represents larger interval, from which interval emerged, and
      - whose off springs represent smaller intervals.
    - Stability of a node is a scale range over which the interval exits.



# Scale Space

- Top level description
  - Iteratively remove nodes from the tree,
    - splicing out nodes that are less stable than any of their parents and off springs

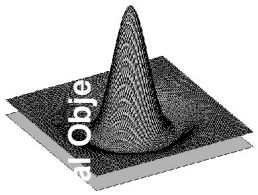
# Scale Space

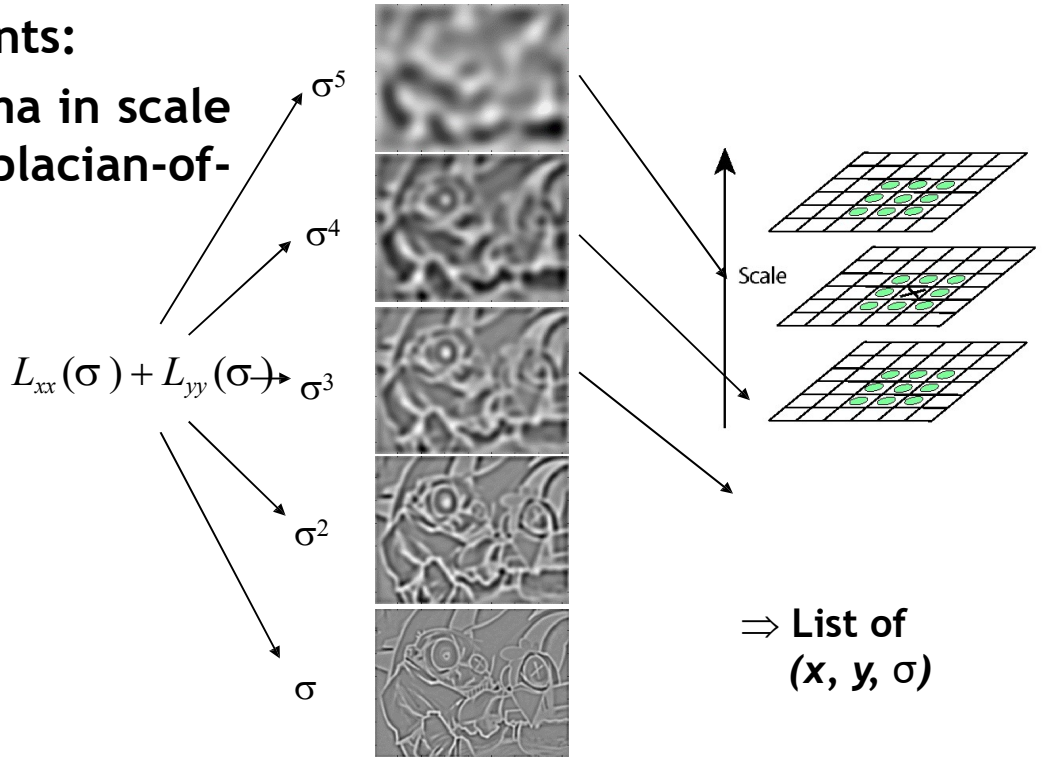
A top level description of several signals using stability criterion.

# Laplacian-of-Gaussian (LoG)

- **Interest points:**

  **Local maxima in scale space of Laplacian-of-Gaussian**



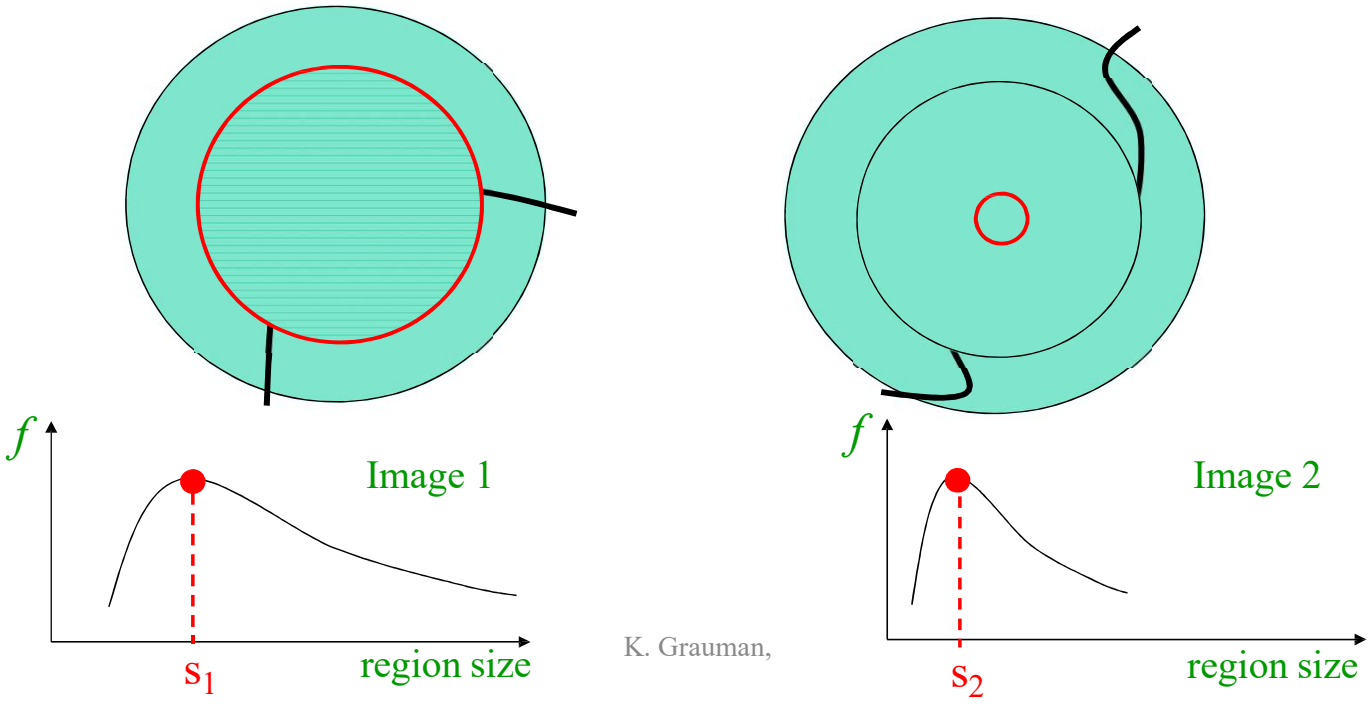



⇒ **List of** ***(x, y, σ)***

# Automatic scale selection

**Intuition:**

- Find scale that gives local maxima of some function *f* in both position and scale.




K. Grauman,

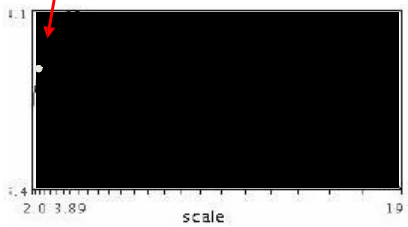
# Automatic Scale Selection

$$f(I_{i_1 \ldots i_m}(x,\sigma)) \quad = \quad f(I_{i_1 \ldots i_m}(x',\sigma'))$$

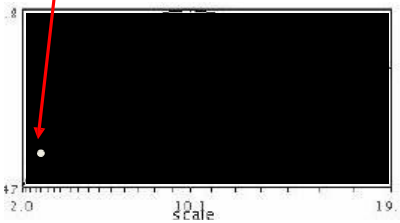
How to find corresponding patch sizes?

# Automatic Scale Selection

- Function responses for increasing scale (scale signature)




$f(I_{i_1 \ldots i_m}(x,\sigma))$

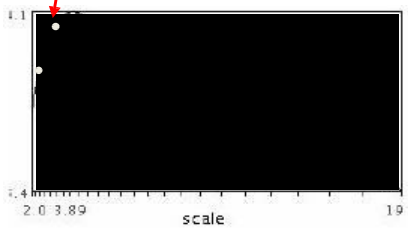



$f(I_{i_1 \ldots i_m}(x',\sigma))$

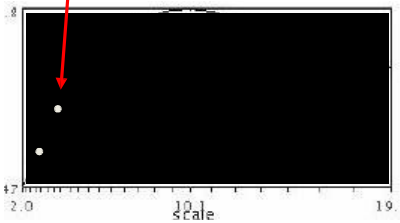
K. Grauman, B. Leibe

# Automatic Scale Selection

- Function responses for increasing scale (scale signature)




$f(I_{i_1 \ldots i_m}(x,\sigma))$

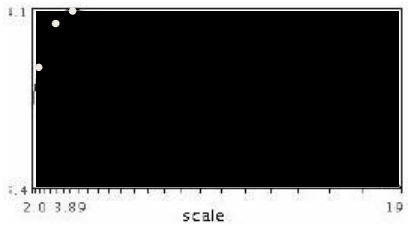



$f(I_{i_1 \ldots i_m}(x',\sigma))$

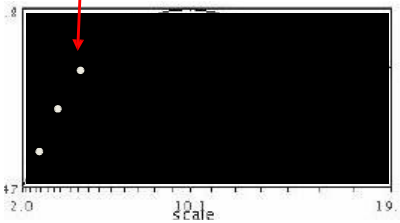
K. Grauman, B. Leibe

# Automatic Scale Selection

- Function responses for increasing scale (scale signature)




$f(I_{i_1 \ldots i_m}(x,\sigma))$

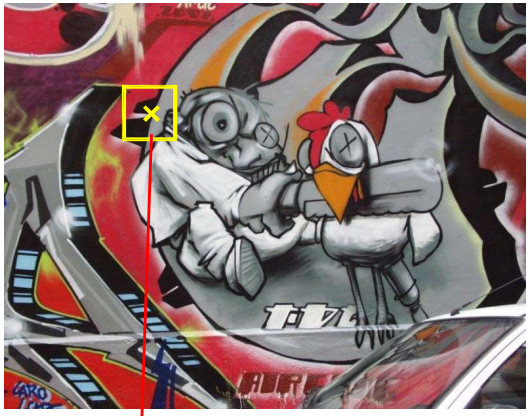


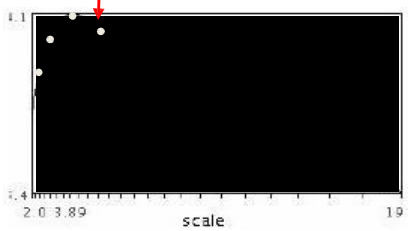

$f(I_{i_1 \ldots i_m}(x',\sigma))$

K. Grauman, B. Leibe

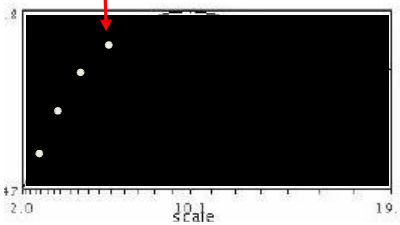
# Automatic Scale Selection

- Function responses for increasing scale (scale signature)






$f(I_{i_1 \ldots i_m}(x,\sigma))$

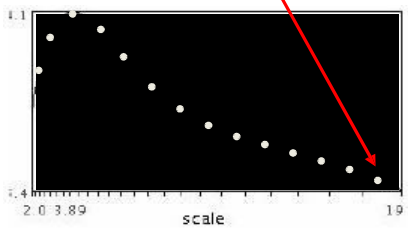



$f(I_{i_1 \ldots i_m}(x',\sigma))$

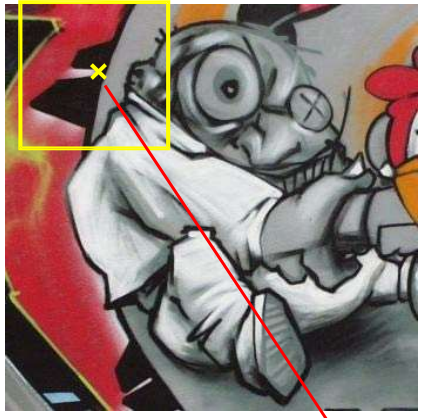
K. Grauman, B. Leibe

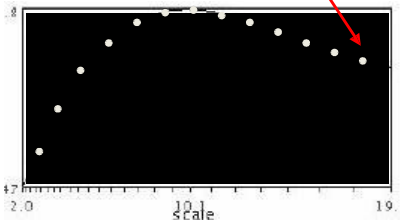
# Automatic Scale Selection

- Function responses for increasing scale (scale signature)




$f(I_{i_1 \ldots i_m}(x,\sigma))$



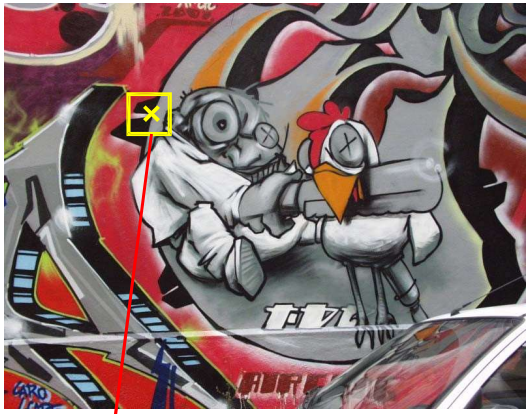


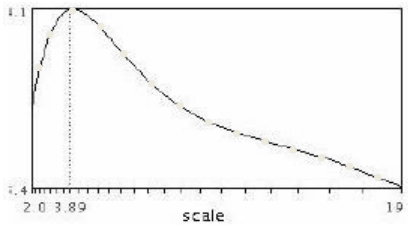

$f(I_{i_1 \ldots i_m}(x',\sigma))$

K. Grauman, B. Leibe

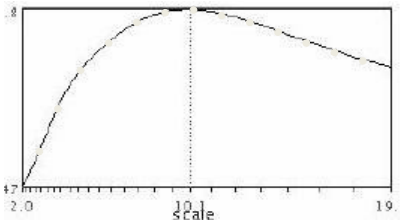
# Automatic Scale Selection

- Function responses for increasing scale (scale signature)






$f(I_{i_1 \ldots i_m}(x,\sigma))$

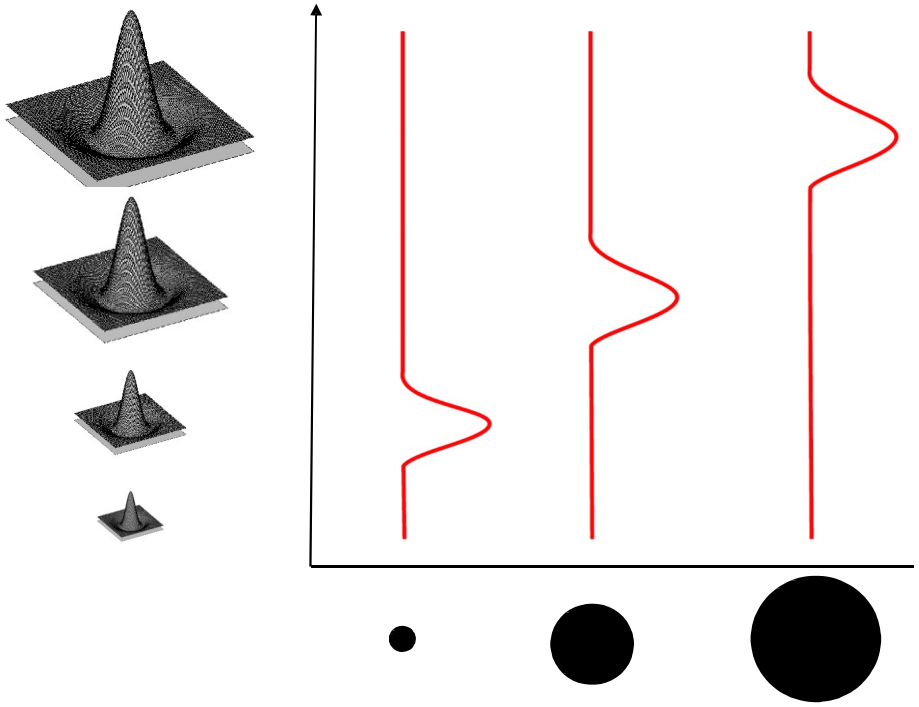



$f(I_{i_1 \ldots i_m}(x',\sigma'))$

K. Grauman, B. Leibe

# What Is A Useful Signature Function?

- Laplacian-of-Gaussian = “blob” detector



K. Grauman, B. Leibe

# Scale-space blob detector: Example

Source: Lana Lazebnik

Sigma = 2

Sigma = 2.5

Sigma = 3.1

Sigma = 3.9

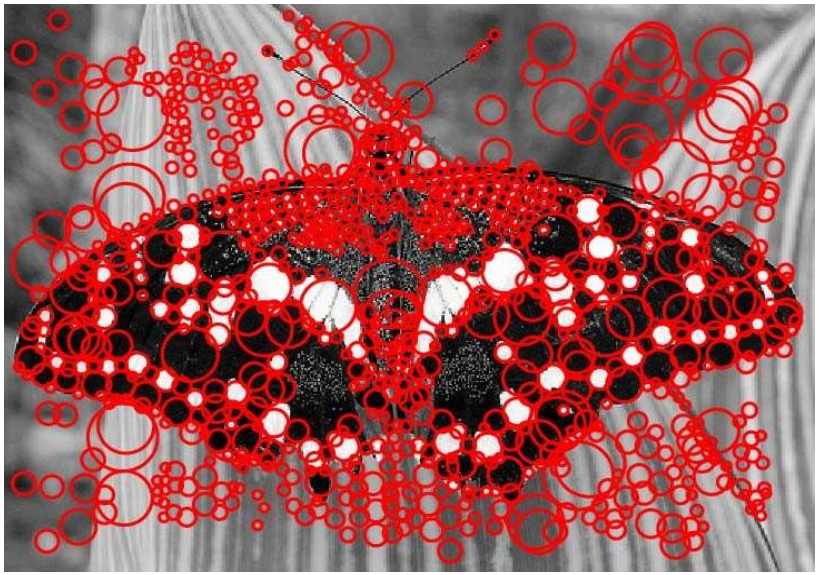
Sigma = 6.1

# Scale-space blob detector: Example

sigma = 11.9912

Source: Lana Lazebnik

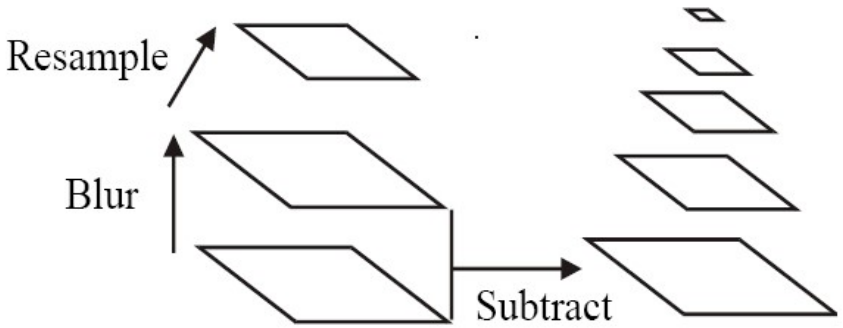
# Scale-space blob detector: Example



Source: Lana Lazebnik

# Building a Scale Space

- All scales must be examined to identify scale-invariant features
- An efficient function is to compute the Laplacian Pyramid (Difference of Gaussian) (Burt & Adelson, 1983)



# Approximation of LoG by Difference of Gaussians

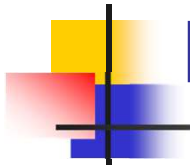
$$\frac{\partial G}{\partial \sigma} = \sigma \Delta^2 G$$

Heat Equation

$$\sigma \Delta^2 G = \frac{\partial G}{\partial \sigma} = \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \Delta^2 G$$

$$\text{Typical values}: \sigma = 1.6; \; k = \sqrt{2}$$



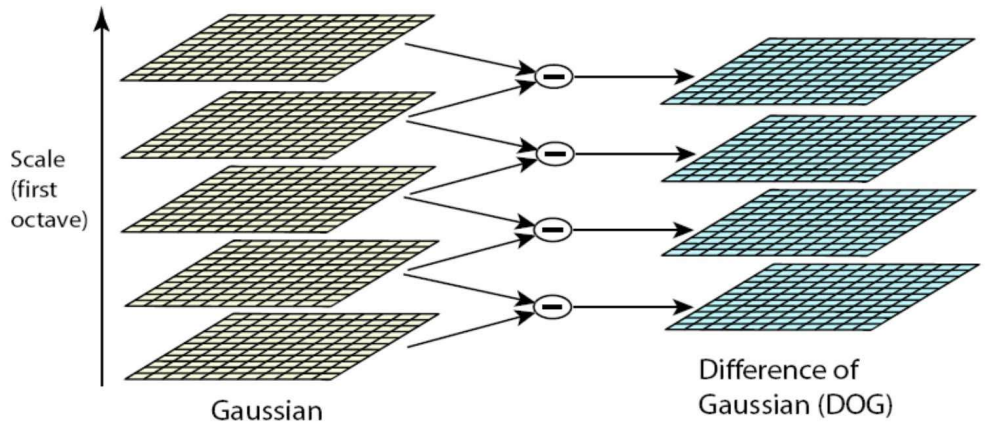
# Steps for Extracting Key Points (SIF Points)

- Scale space peak selection
  - Potential locations for finding features
- Key point localization
  - Accurately locating the feature key points
- Orientation Assignment
  - Assigning orientation to the key points
- Key point descriptor
  - Describing the key point as a high dimensional vector (128) (SIFT Descriptor)

# Building a Scale Space

...

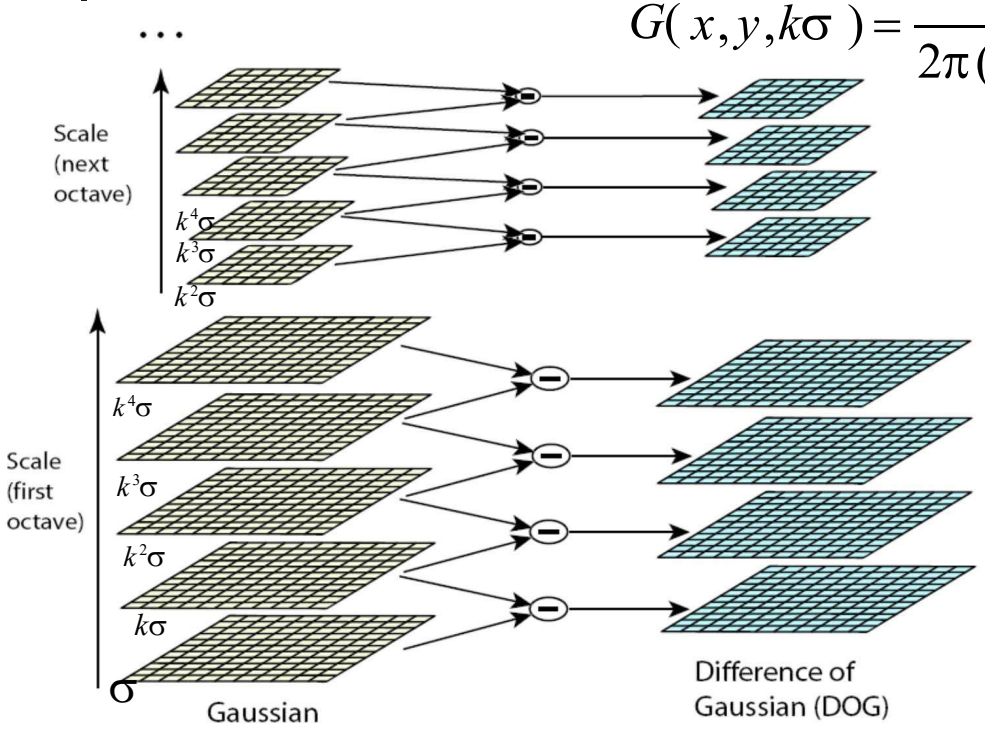
$$G(x, y, k\sigma) = \frac{1}{2\pi(k\sigma)^2} e^{-(x^2+y^2)/2k^2\sigma^2}$$




$$k = \sqrt{2}$$

# Building a Scale Space

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

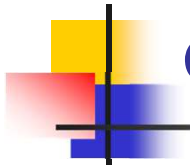



$$k = \sqrt{2}$$

| octave \ scale → | | | | | |
|---|---|---|---|---|---|
| | 0.707107 | 1.000000 | 1.414214 | 2.000000 | 2.828427 |
| | 1.414214 | 2.000000 | 2.828427 | 4.000000 | 5.656854 |
| | 2.828427 | 4.000000 | 5.656854 | 8.000000 | 11.313708 |
| | 5.656854 | 8.000000 | 11.313708 | 16.000000 | 22.627417 |

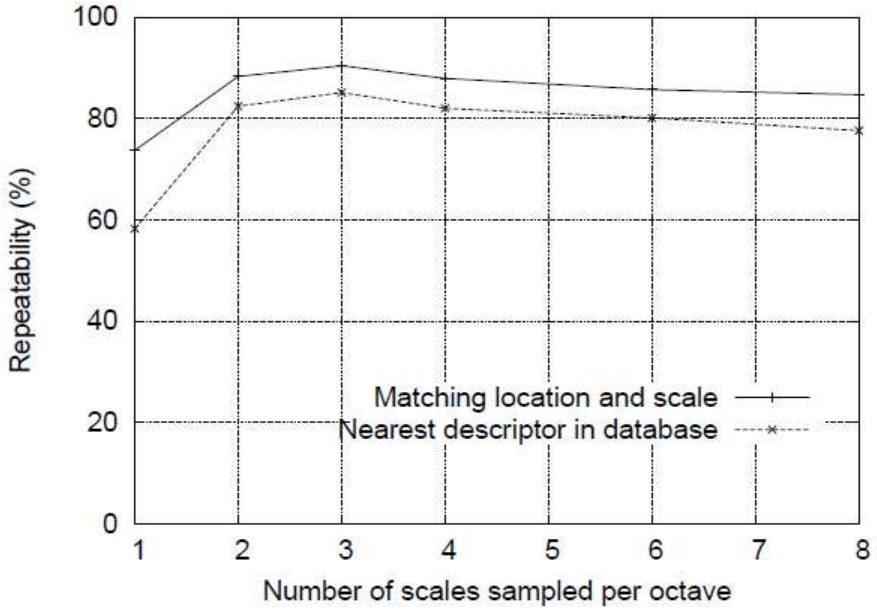
$$\sigma = .707187.6; \;\; k = \sqrt{2}$$

# How many scales per octave?



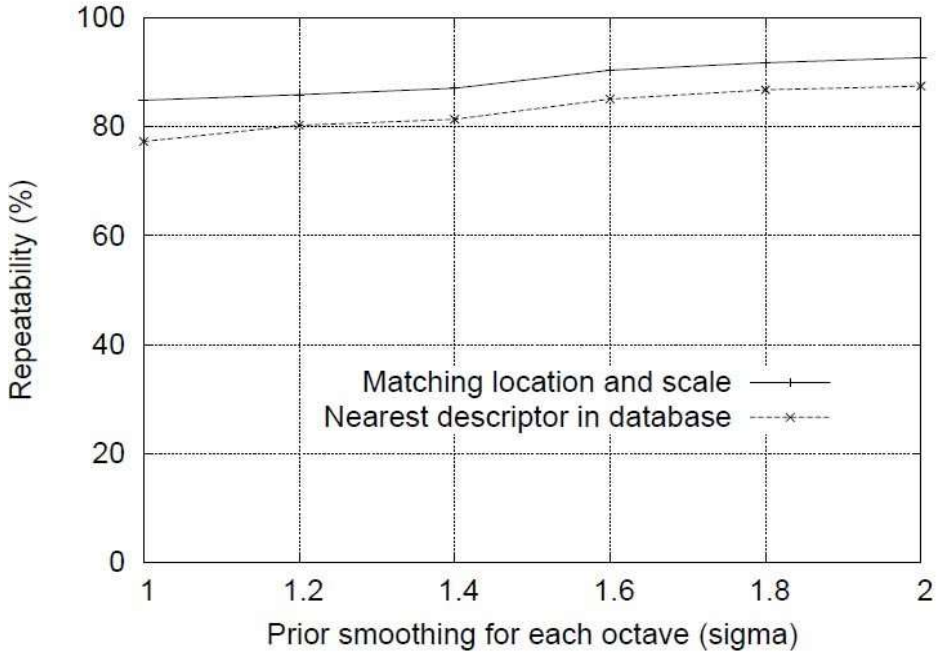
- A collection of 32 real images drawn from a diverse range, including
  - outdoor scenes, human faces, aerial photographs, and industrial
- Each image was then subject to a range of transformations:
  - rotation, scaling, affine stretch, change in brightness and
  - contrast, and addition of image noise.

# How many scales per octave?




3 Scales

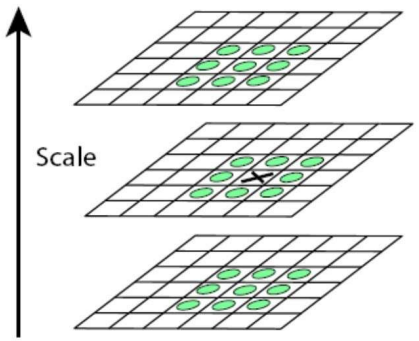
# Initial value of sigma




1.6

# Scale Space Peak Detection

- Compare a pixel (**X**) with 26 pixels in current and adjacent scales (Green Circles)
- Select a pixel (**X**) if larger/smaller than all 26 pixels
- Large number of extrema, computationally expensive
  - Detect the most stable subset with a coarse sampling of scales



# Key Point Localization

- Candidates are chosen from extrema detection

original image

extrema locations

# Initial Outlier Rejection

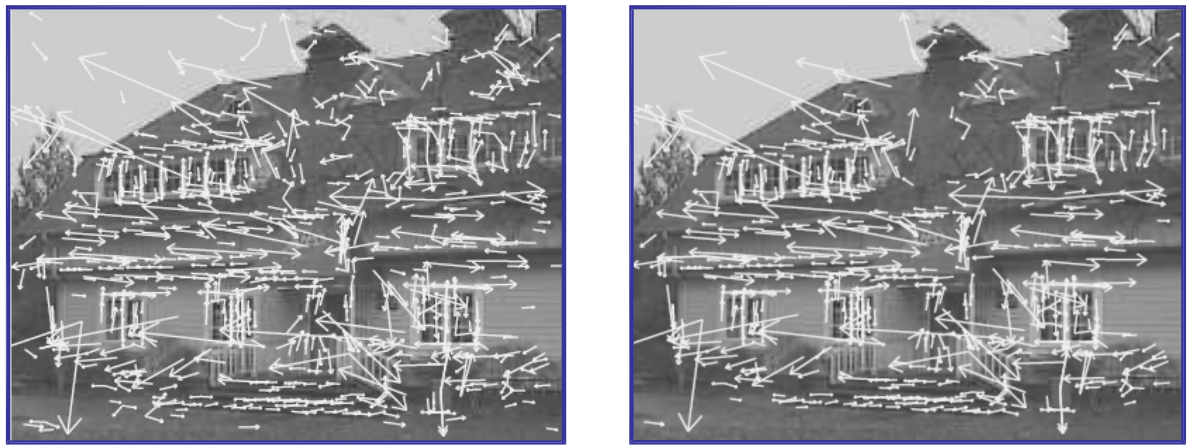
1. Low contrast candidates
2. Poorly localized candidates along an edge

- Taylor series expansion of DOG, $\mathbf{D}$.

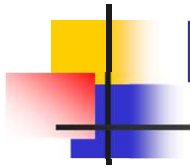
$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^\mathbf{T}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \qquad \mathbf{x} = (x, y, \sigma)^T$$

- Minima or maxima is located at $\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$
- Value of $\mathrm{D}(\mathrm{x})$ at minima/maxima must be large, $|D(x)| > th$.

# Initial Outlier Rejection

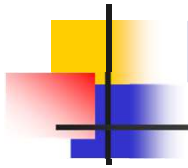


from 832 key points to 729 key points, $th=0.03$.



# Further Outlier Rejection

- DOG has strong response along edge
- Assume DOG as a surface
  - Compute principal curvatures (PC)
  - Poorly defined peak will have very low curvature along the edge, high across the edge



# Further Outlier Rejection

- Use Trace and det similar to Harris corner detector
- Compute Hessian of D (principal curvature)

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$Tr(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$

$$Det(H) = D_{xx}D_{yy} - D_{xy}^2 = \lambda_1\lambda_2$$

- Remove outliers by evaluating

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r}$$

$$r = \frac{\lambda_1}{\lambda_2}$$

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1\lambda_2} = \frac{(r\lambda_2 + \lambda_2)^2}{r\lambda_2^2} = \frac{(r+1)^2}{r}$$

# Further Outlier Rejection

- Following quantity is minimum (eigen values are equal) when $r$=1
- It increases with $r$

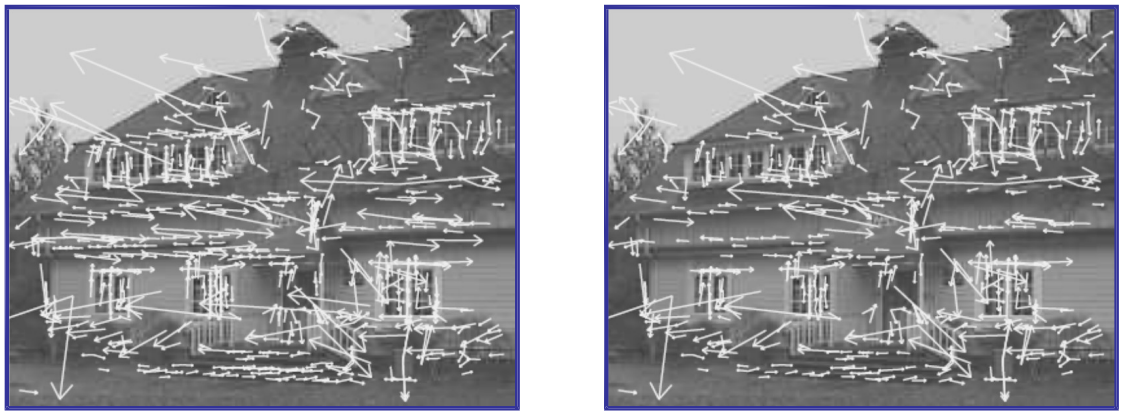
$$r = \frac{\lambda_1}{\lambda_2}$$

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r}$$

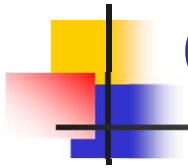
- Eliminate key points if

$$\frac{Tr(H)^2}{Det(H)} \prec \frac{(r+1)^2}{r} \qquad r > 10$$

# Further Outlier Rejection



from 729 key points to 536 key points.



# Orientation Assignment

- To achieve rotation invariance
- Compute central derivatives, gradient magnitude and direction of *L (*smooth image) at the scale of key point (x,y)

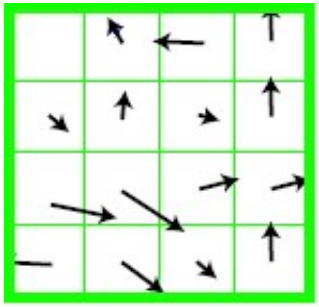
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

L is Laplacian of Gaussian (LoG)

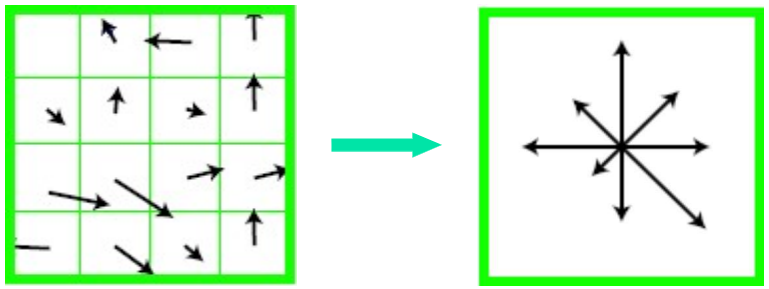
# Orientation Assignment

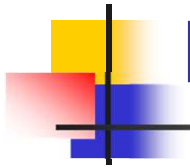
- Create a weighted direction histogram in a neighborhood of a key point (36 bins)
- Weights are
  - Gradient magnitudes
  - Spatial gaussian filter with $\sigma$=1.5 x <scale of key point>



# Orientation Assignment

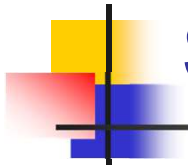
- Select the peak as direction of the key point
- Introduce additional key points at same location
  - if another peak is within 80% of max peak of the histogram with different directions





# Local Image Descriptors at Key Points

- Possible descriptor
  - Store intensity samples in the neighborhood
  - Sensitive to lighting changes, 3D object transformation
- Use of gradient orientation histograms
  - Robust representation



# Similarity to IT cortex

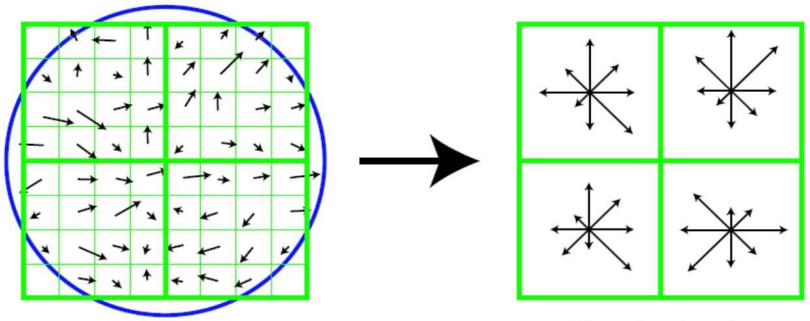
- Complex neurons respond to a gradient at a particular orientation.
- Location of the feature can shift over a small receptive field.

# Tomaso Poggio, MIT

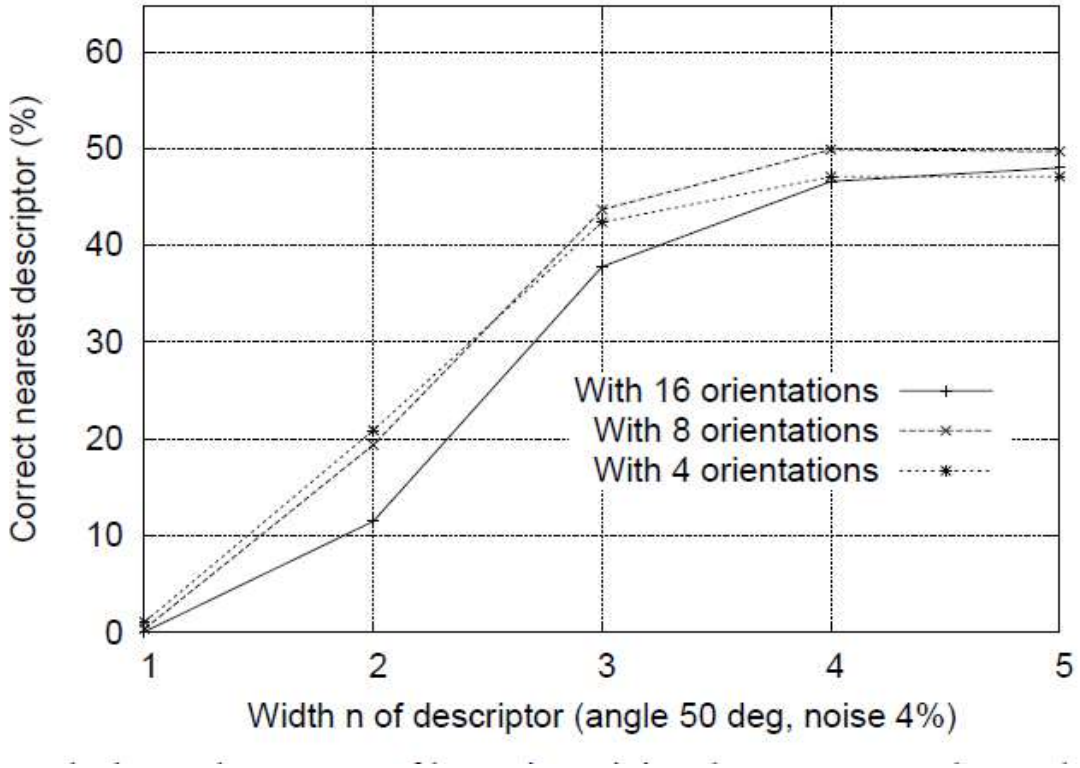
- Edelman, Intrator, and Poggio (1997)
  - The function of the cells allow for matching and recognition of 3D objects from a range of view points.
- Experiments show better recognition accuracy for 3D objects rotated in depth by up to 20 degrees

# Extraction of Local Image Descriptors at Key Points

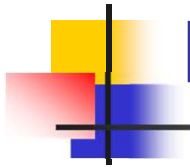
- Compute relative orientation and magnitude in a 16x16 neighborhood at key point
- Form weighted histogram (8 bin) for 4x4 regions
  - Weight by magnitude and spatial Gaussian
  - Concatenate 16 histograms in one long vector of 128 dimensions
- Example for 8x8 to 2x2 descriptors



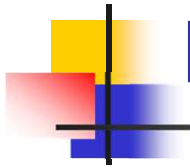
# Descriptor Regions (n by n)



# Extraction of Local Image Descriptors at Key Points



- Store numbers in a vector
- Normalize to unit vector (**UN**)
  - Illumination invariance (affine changes)
- For non-linear intensity transforms
  - Bound **Unit Vector** items to maximum 0.2 (remove gradients larger than 0.2)
  - Renormalize to unit vector



# Key point matching

- Match the key points against a database of that obtained from training images.
- Find the nearest neighbor i.e. a key point with minimum Euclidean distance.
  - Efficient Nearest Neighbor matching
    - Looks at ratio of distance between best and 2nd best match (.8)

# Matching local features

# Matching local features

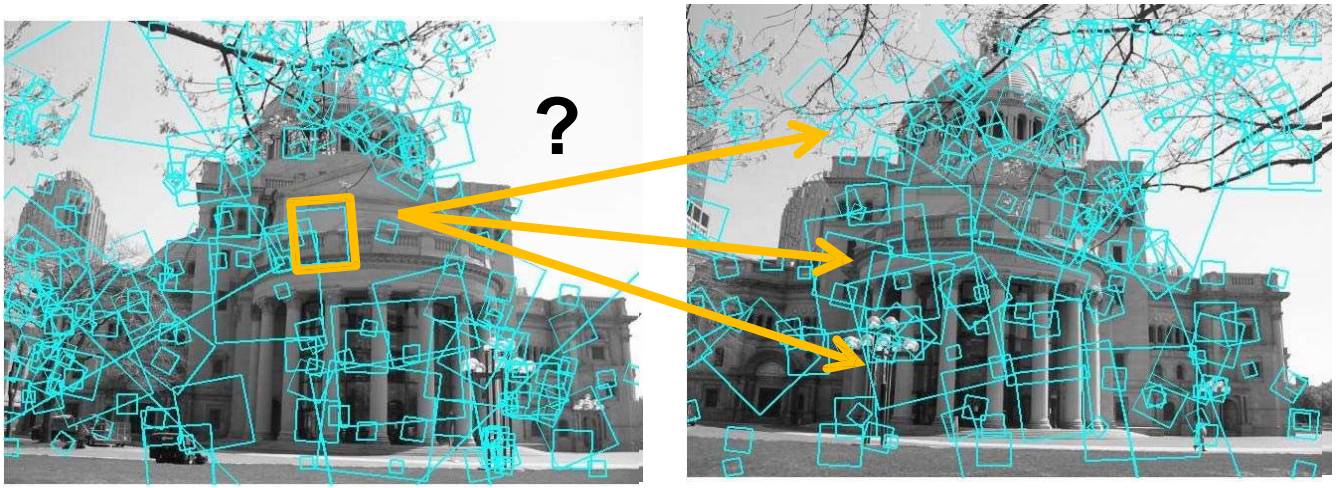
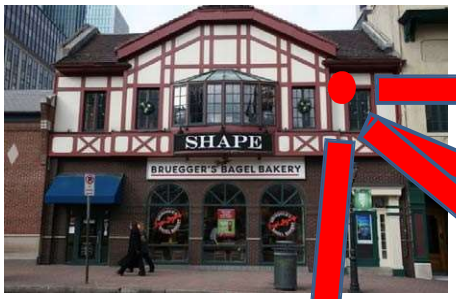



Image 1 Image 2

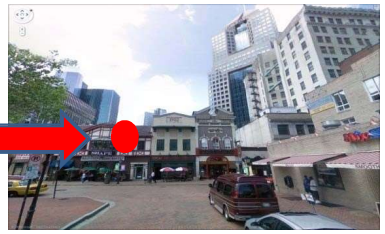
- To generate **candidate matches**, find that have the most similar appearance or SIFT descriptor patches
- Simplest approach: compare them all, take the closest
  (or closest k, or within a thresholded distance)


**Query Image**

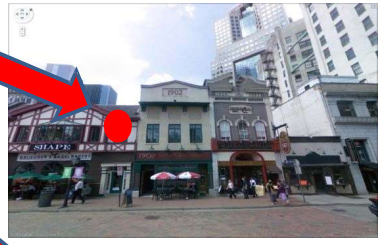



Query Image

1st NN

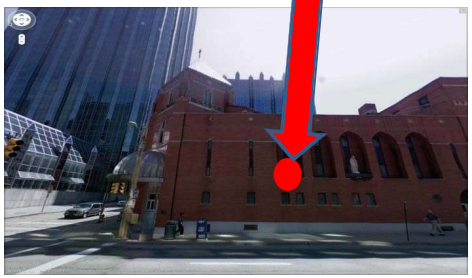


2nd NN



3rd NN


4th NN



$$\text{if} \quad \frac{\text{Distance to first match}}{\text{Distance to seond match}} < .8 \quad \text{Good match}$$

# Ambiguous matches

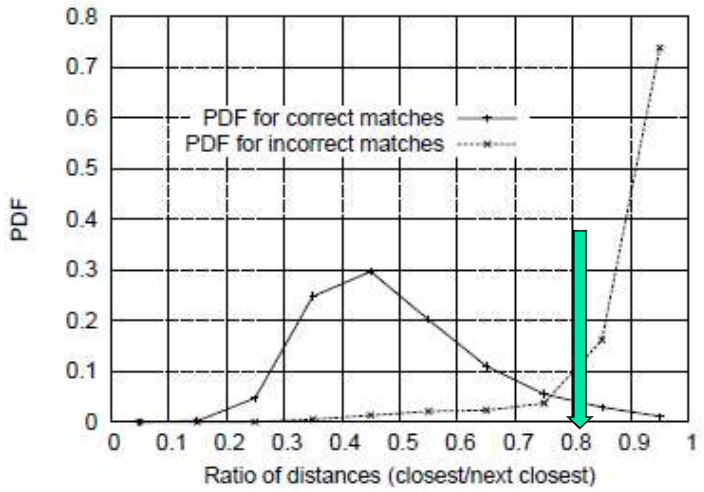
Image 1

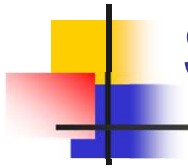

Image 2

- At what distance do we have a good match?
- To add robustness to matching, can consider **ratio** : distance to best match / distance to second best match
- If low, first match looks good.
- If high, could be ambiguous match.

Kristen Grauman

# The ratio of distance from the closest to the distance of the second closest



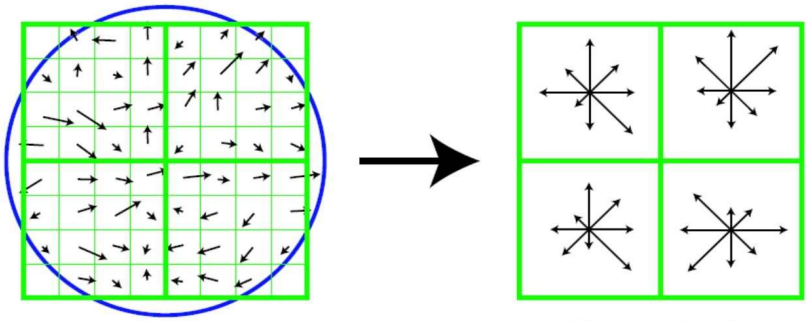


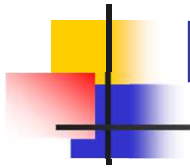
# SIFT Detector

- Generate Scale Space of an Image
- Detect Peaks in Scale Space (extrema)
- Localize Interest Points (Taylor Series)
- Remove outliers (remove response along edges)
- Assign Orientation

# SIFT Descriptor

- Compute relative orientation and magnitude in a 16x16 neighborhood at key point
- Form weighted histogram (8 bin) for 4x4 regions
  - Weight by magnitude and spatial Gaussian
  - Concatenate 16 histograms in one long vector of 128 dimensions
- Example for 8x8 to 2x2 descriptors





# Reference

D. Lowe. Distinctive image features from scale-invariant key points., International Journal of Computer Vision 2004.

# Histograms of Oriented Gradients for Human Detection
# (HOG)

N. Dalal and B. Triggs

CVPR 2005

# Histograms of Oriented Gradients for Human Detection

**Navneet Dalal and Bill Triggs**
INRIA Rhône-Alps, 655 avenue de l'Europe, Montbonnot 38334, France
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, http://lear.inrialpes.fr

## Abstract

*We study the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient based descriptors, we show experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. We study the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so we introduce a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds.*

## 1 Introduction

Detecting humans in images is a challenging task owing to their variable appearance and the wide range of poses that they can adopt. The first need is a robust feature set that allows the human form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination. We study the issue of feature sets for human detection, showing that locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets including wavelets [17, 22]. The proposed descriptors are reminiscent of edge orientation histograms [4, 5], SIFT descriptors [12] and shape contexts [1], but they are computed on a dense grid of uniformly spaced cells and they use overlapping local contrast normalizations for improved performance. We make a detailed study of the effects of various implementation choices on detector performance, taking "pedestrian detection" (the detection of mostly visible people in more or less upright poses) as a test case. For simplicity and speed, we use linear SVM as a baseline classifier throughout the study. The new detectors give essentially perfect results on the MIT pedestrian test set [18, 17], so we have created a more challenging set containing over 1800 pedestrian images with a large range of poses and backgrounds. Ongoing work suggests that our feature set performs equally well for other shape-based object classes.

We briefly discuss previous work on human detection in §2, give an overview of our method §3, describe our data sets in §4 and give a detailed description and experimental evaluation of each stage of the process in §5–6. The main conclusions are summarized in §7.

## 2 Previous Work

There is an extensive literature on object detection, but here we mention just a few relevant papers on human detection [18, 17, 22, 16, 20]. See [6] for a survey. Papageorgiou *et al* [18] describe a pedestrian detector based on a polynomial SVM using rectified Haar wavelets as input descriptors, with a parts (subwindow) based variant in [17]. Depoortere *et al* give an optimized version of this [2]. Gavrila & Philomen [8] take a more direct approach, extracting edge images and matching them to a set of learned exemplars using chamfer distance. This has been used in a practical real-time pedestrian detection system [7]. Viola *et al* [22] build an efficient moving person detector, using AdaBoost to train a chain of progressively more complex region rejection rules based on Haar-like wavelets and space-time differences. Ronfard *et al* [19] build an articulated body detector by incorporating SVM based limb classifiers over 1st and 2nd order Gaussian filters in a dynamic programming framework similar to those of Felzenszwalb & Huttenlocher [3] and Ioffe & Forsyth [9]. Mikolajczyk *et al* [16] use combinations of orientation-position histograms with binary-thresholded gradient magnitudes to build a parts based method containing detectors for faces, heads, and front and side profiles of upper and lower body parts. In contrast, our detector uses a simpler architecture with a single detection window, but appears to give significantly higher performance on pedestrian images.

## 3 Overview of the Method

This section gives an overview of our feature extraction chain, which is summarized in fig. 1. Implementation details are postponed until §6. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. Similar features have seen increasing use over the past decade [4, 5, 12, 15]. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or



**Cited by** 8908

# HOG Steps

- HOG feature extraction
  - Compute centered horizontal and vertical gradients with no smoothing
  - Compute gradient orientation and magnitudes
    - For color image, pick the color channel with the highest gradient magnitude for each pixel.
  - For a 64x128 image,
  - Divide the image into 16x16 blocks of 50% overlap.
    - 7x15=105 blocks in total
  - Each block should consist of 2x2 cells with size 8x8.
  - Quantize the gradient orientation into 9 bins
    - The vote is the gradient magnitude
    - Interpolate votes between neighboring bin center.
    - The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block.
  - Concatenate histograms (Feature dimension: 105x4x9 = 3,780)

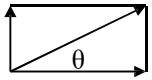
# Computing Gradients

- Centered: $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$
- Filter masks in x and y directions
  - Centered:

| -1 | 0 | 1 |
|---|---|---|

| -1 |
|---|
| 0 |
| 1 |

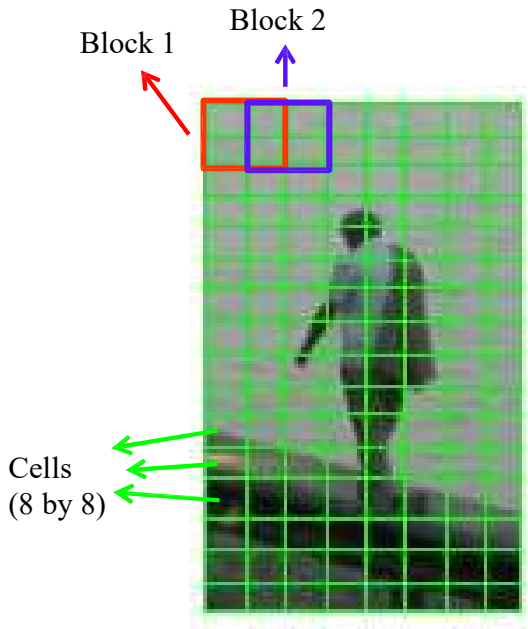
- Gradient
  - Magnitude: $s = \sqrt{s_x^2 + s_y^2}$




  - Orientation: $\theta = \arctan(\frac{s_y}{s_x})$

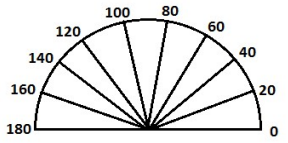
# Blocks, Cells

- 16x16 blocks of 50% overlap.
  - 7x15=105 blocks in total
- Each block should consist of 2x2 cells with size 8x8.

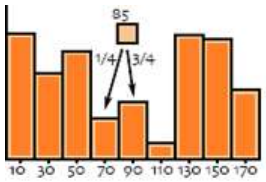


# Votes

- Each block consists of 2x2 cells with size 8x8
- Quantize the gradient orientation into 9 bins (0-180)



9 Bins

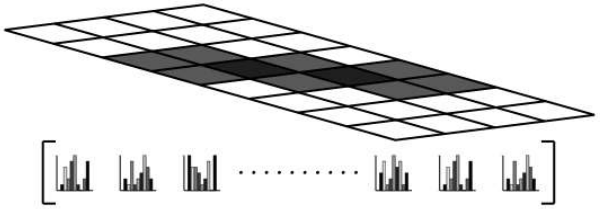


Bin centers

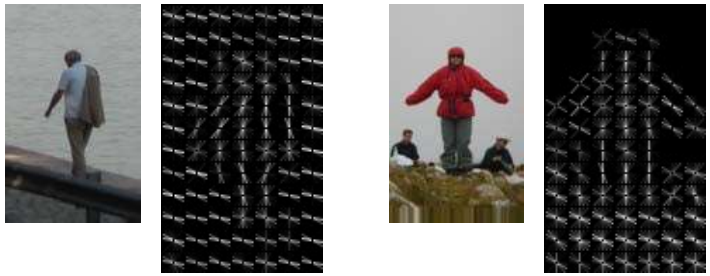
  - The vote is the gradient magnitude
  - Interpolate votes linearly between neighboring bin centers.
    - Example: if θ=85 degrees.
    - Distance to the bin center Bin 70 and Bin 90 are 15 and 5 degrees, respectively.
    - Hence, ratios are 5/20=1/4, 15/20=3/4.
  - The vote can also be weighted with Gaussian to down weight the pixels near the edges of the block.

# Final Feature Vector

- Concatenate histograms
  - Make it a 1D vector of length 3780.



- Visualization



# Results

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

# SIFT Vs HOG

**SIFT**

- 128 dimensional vector
- 16 by 16 window
- 4x4 sub-window (16 total)
- 8 bin histogram

**HOG**

- 3,780 dimensional vector
- 64 by 128 window
- 16 by 16 blocks with overlap
- Each block consists of 2 by 2 cells each of 8 by 8
- Overlapping
- 9 bin histogram

To continue…