Scale-Invariant Feature Transform
CS-477 Computer Vision

Dr. Mohsin Kamal
Associate Professor
dr.mohsinkamal@seecs.edu.pk

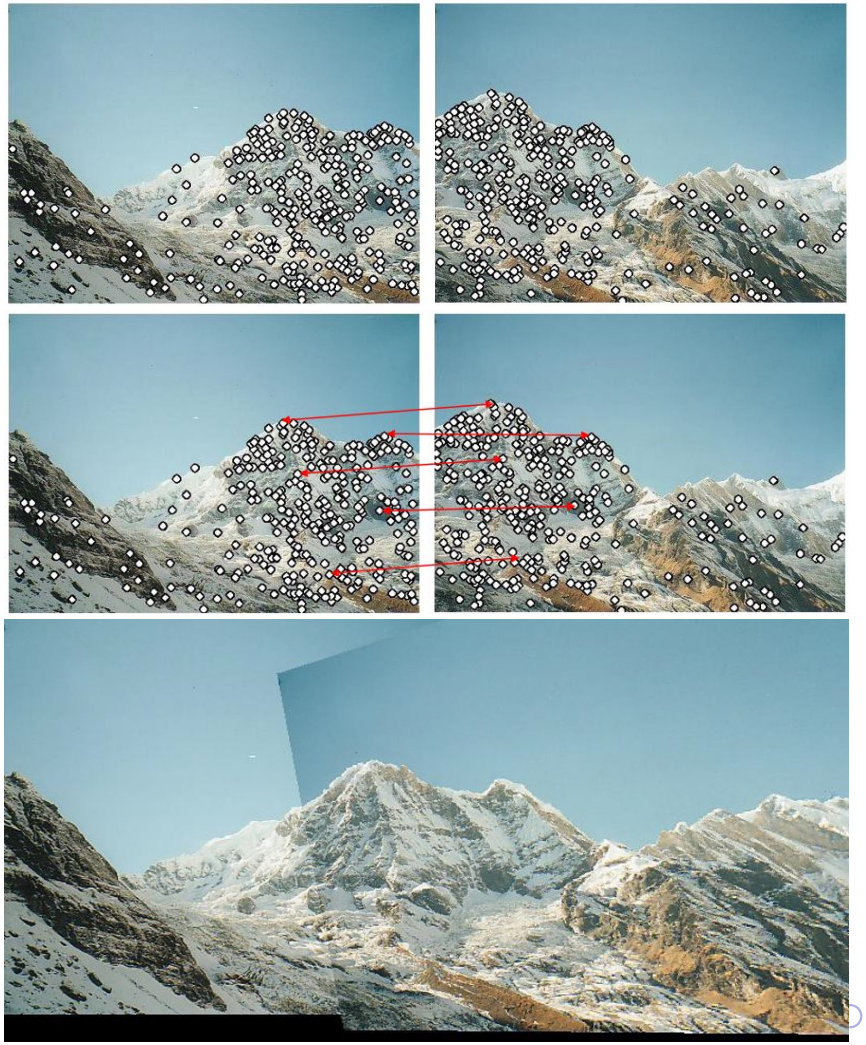
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST), Pakistan

How do we build a panorama?

We need to match (align) images

How do we build a panorama?

- Detect feature points in both images
- Find corresponding pairs
- Use these matching pairs to align images - the required mapping is called a homography.



Matching with features

Problem 1: Detect the *same* point independently in both images

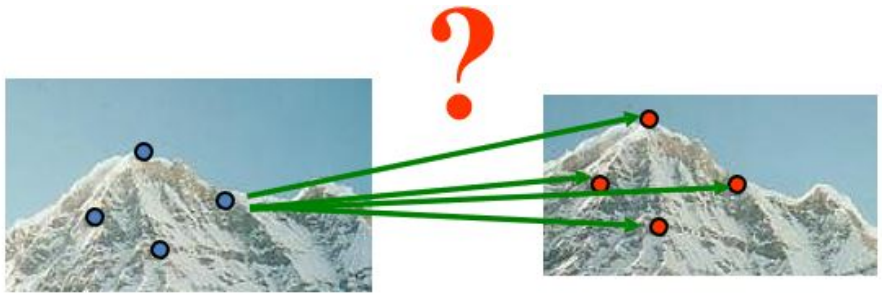
counter-example:

no chance to match!

We need a repeatable **detector**

Matching with features

Problem 2: For each point correctly recognize the corresponding one



We need a reliable and distinctive **descriptor**

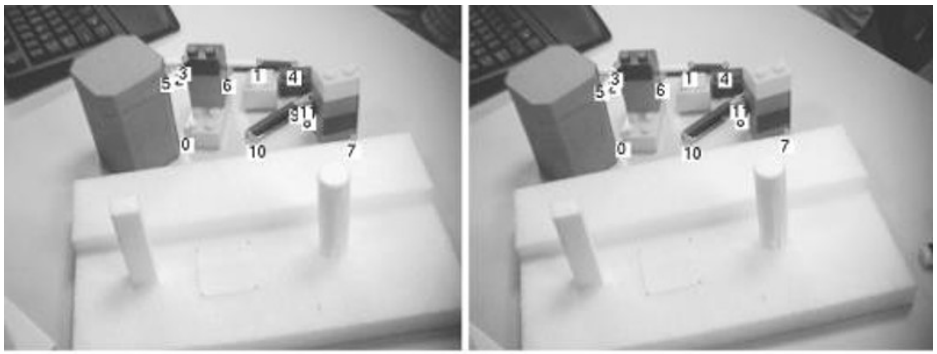
Matching with features

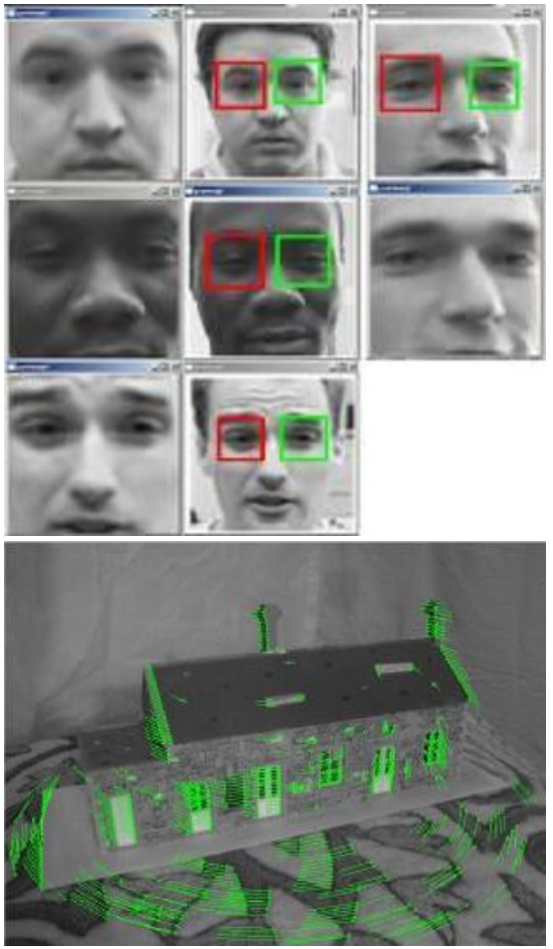
Among all my matches, how do I know which ones are good?

Finding the "same" thing across images

Why do we care about matching features?

- Object recognition
- Wide baseline matching
- Tracking





Matching with features

We want invariance!

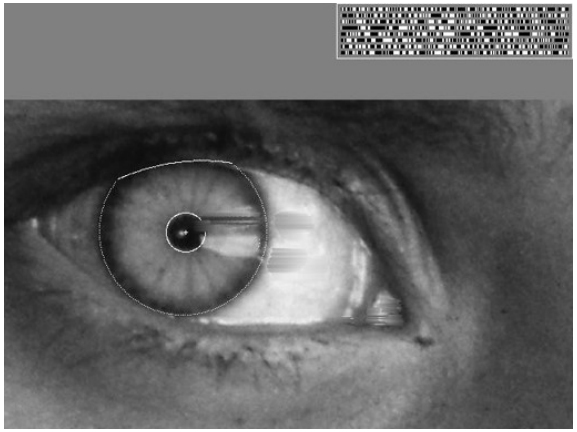
Good features should be robust to all sorts of nastiness that can occur between images.

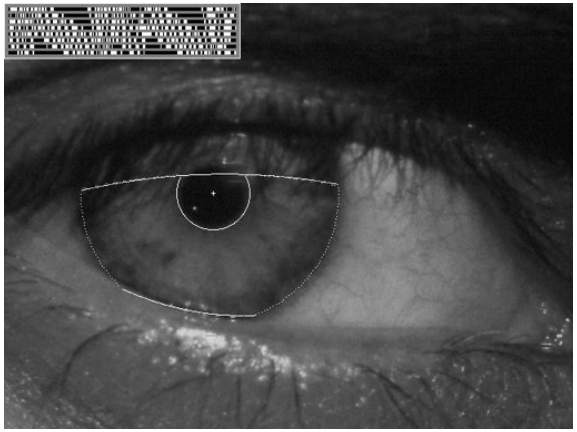
Image matching

Harder case

Ever harder case

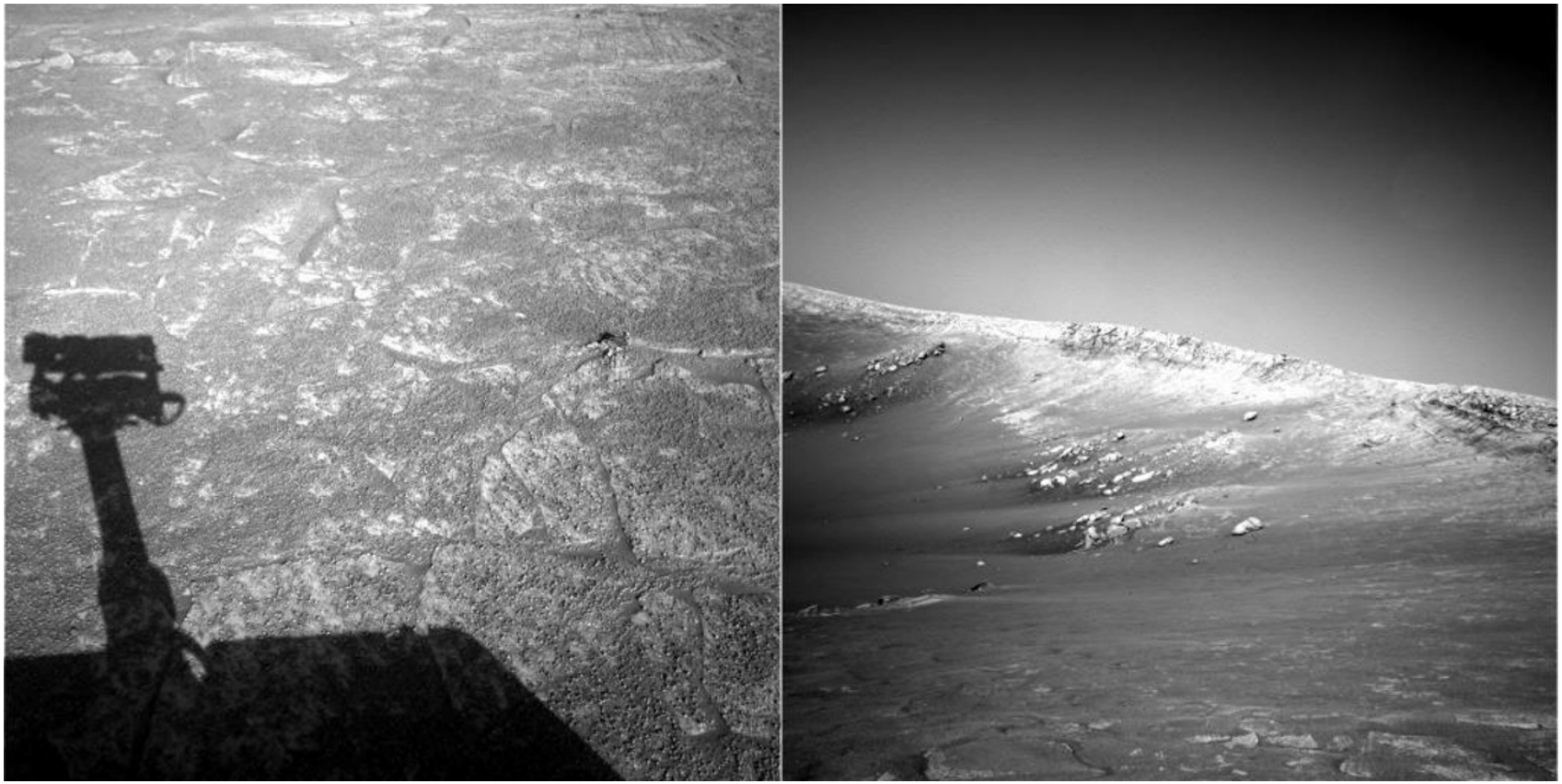
"How the Afghan Girl was Identified by Her Iris Patterns[1]"





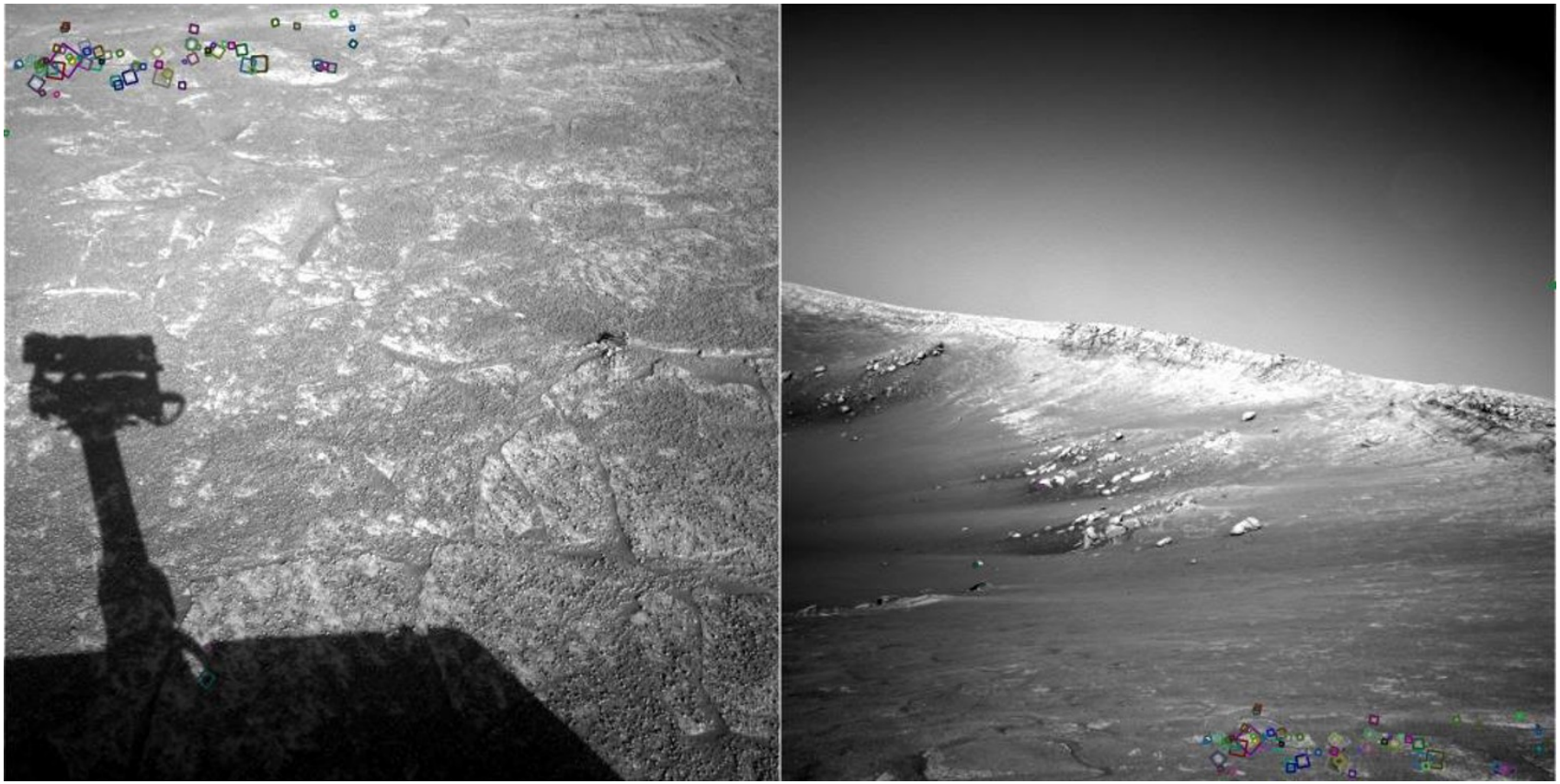
[1] https://www.cl.cam.ac.uk/~jgd1000/afghan.html

Still a harder case



NASA Mars Rover images

Answer below (look for tiny colored squares...)



NASA Mars Rover images with SIFT feature matches (Figure by Noah Snavely)

Image matching

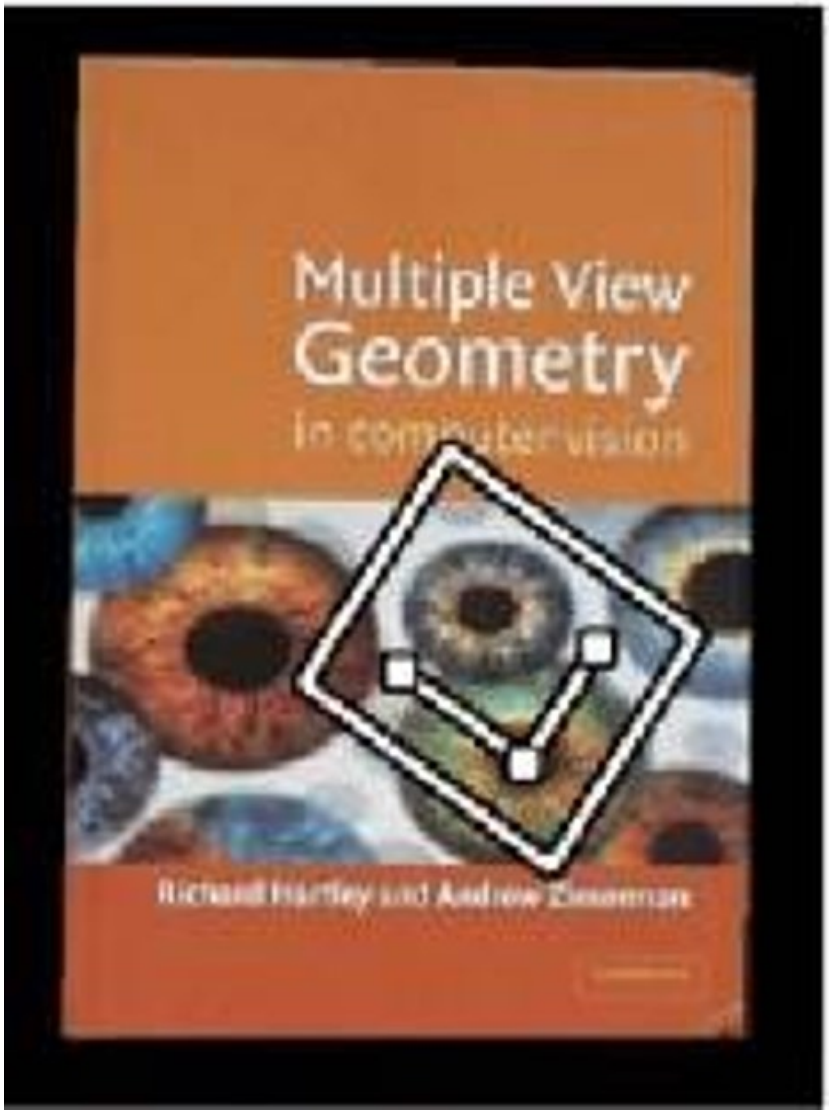
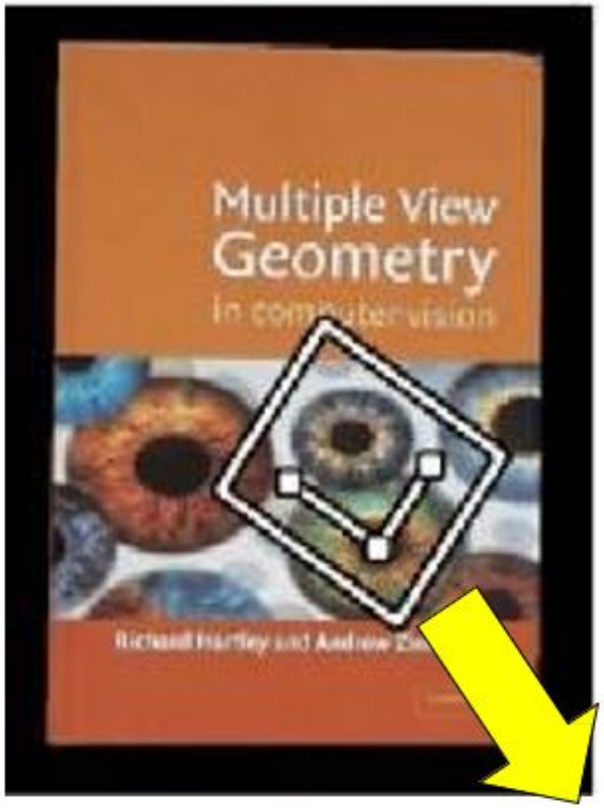
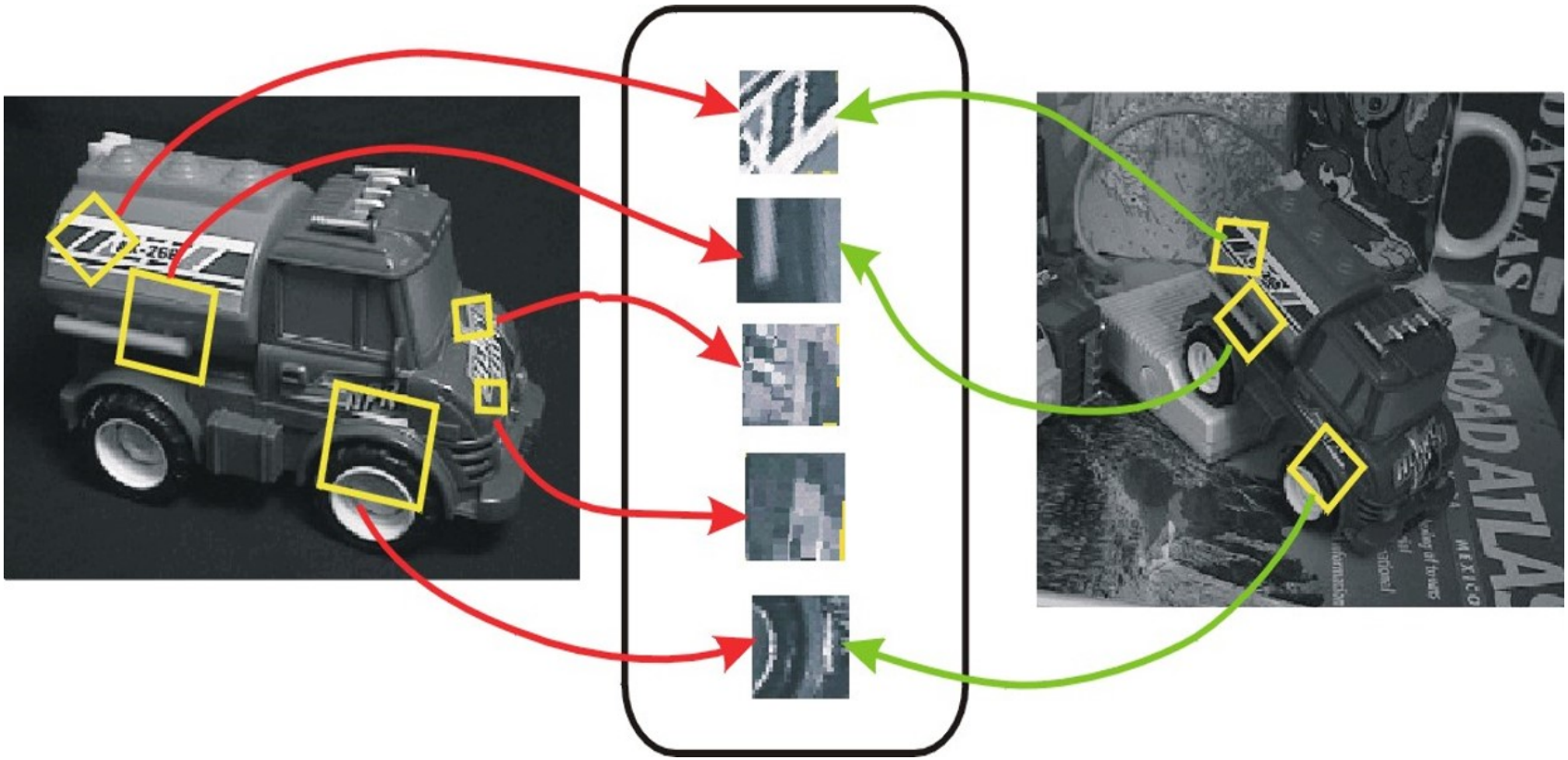


Image matching



Invariant local features

- Find features that are invariant to transformations
  - geometric invariance: translation, rotation, scale
  - photometric invariance: brightness, exposure, ...



Advantages of local features

Locality: features are local, so robust to occlusion and clutter
Distinctiveness: can differentiate a large database of objects
Quantity: hundreds or thousands in a single image
Efficiency: real-time performance achievable
Generality: exploit different types of features in different situations

More motivation

Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Robot navigation, and
- many more

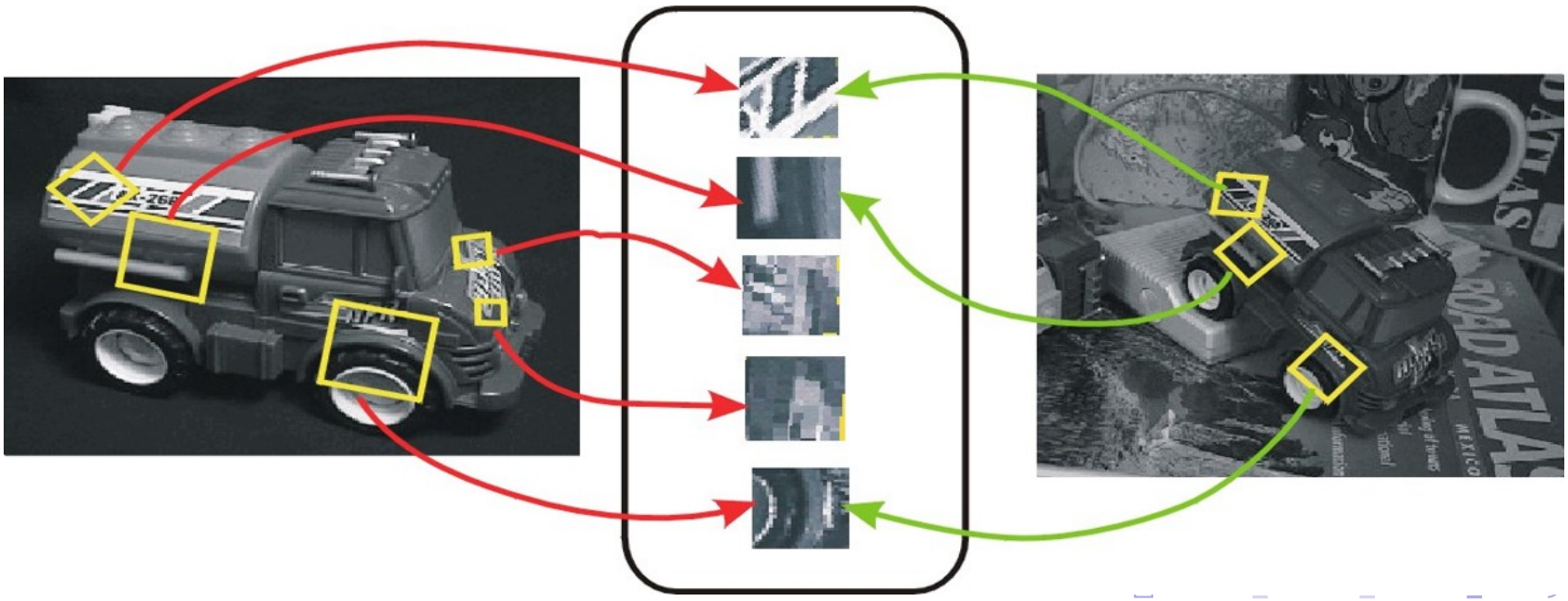
Uniqueness

Look for image regions that are unusual

- Lead to unambiguous matches in other images
- How to define "unusual"?

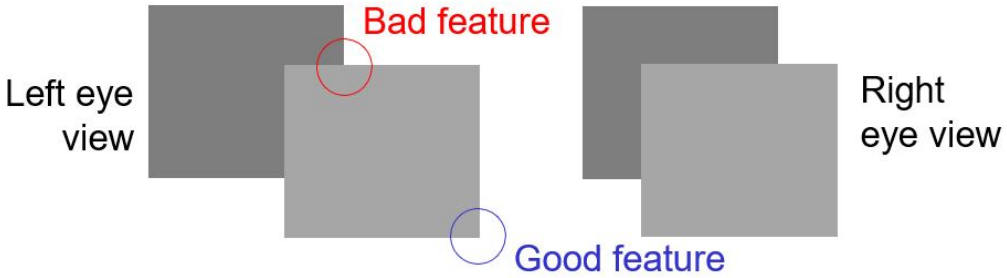
Detector, Descriptor and Correspondence

- **Detector:** detect same scene points independently in both images
- **Descriptor:** encode local neighboring window
  - Note how scale and rotation of window are the same in both image (but computed independently)
- **Correspondence:** find most similar descriptor in other image



Selecting Good features

- What is a "good feature"?
  - Satisfies brightness constancy - looks the same in both images
  - Has sufficient texture variation
  - Does not have too much texture variation
  - Corresponds to a "real" surface patch.



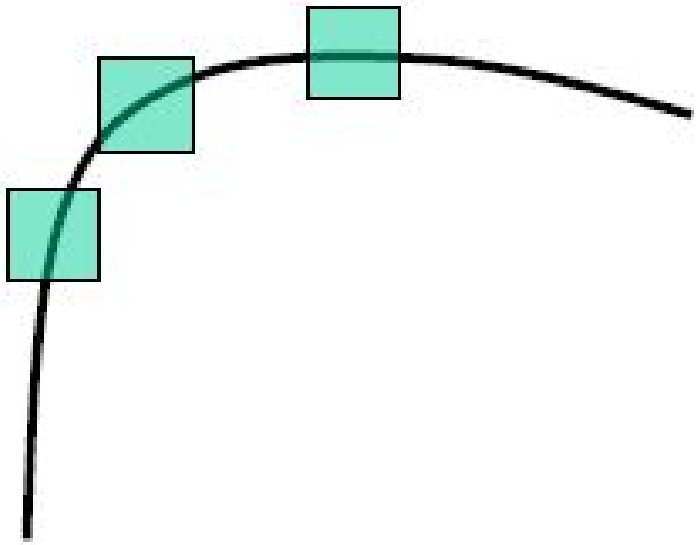
  - Does not deform too much over time

Harris detector: Some properties

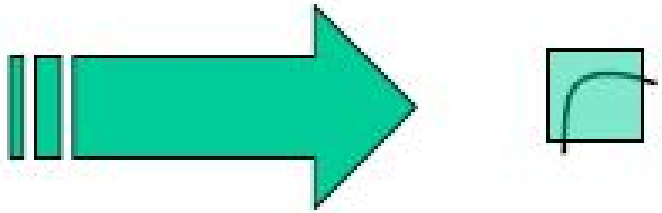
Invariant image scale

Harris detector: Some properties

Not invariant to image scale!



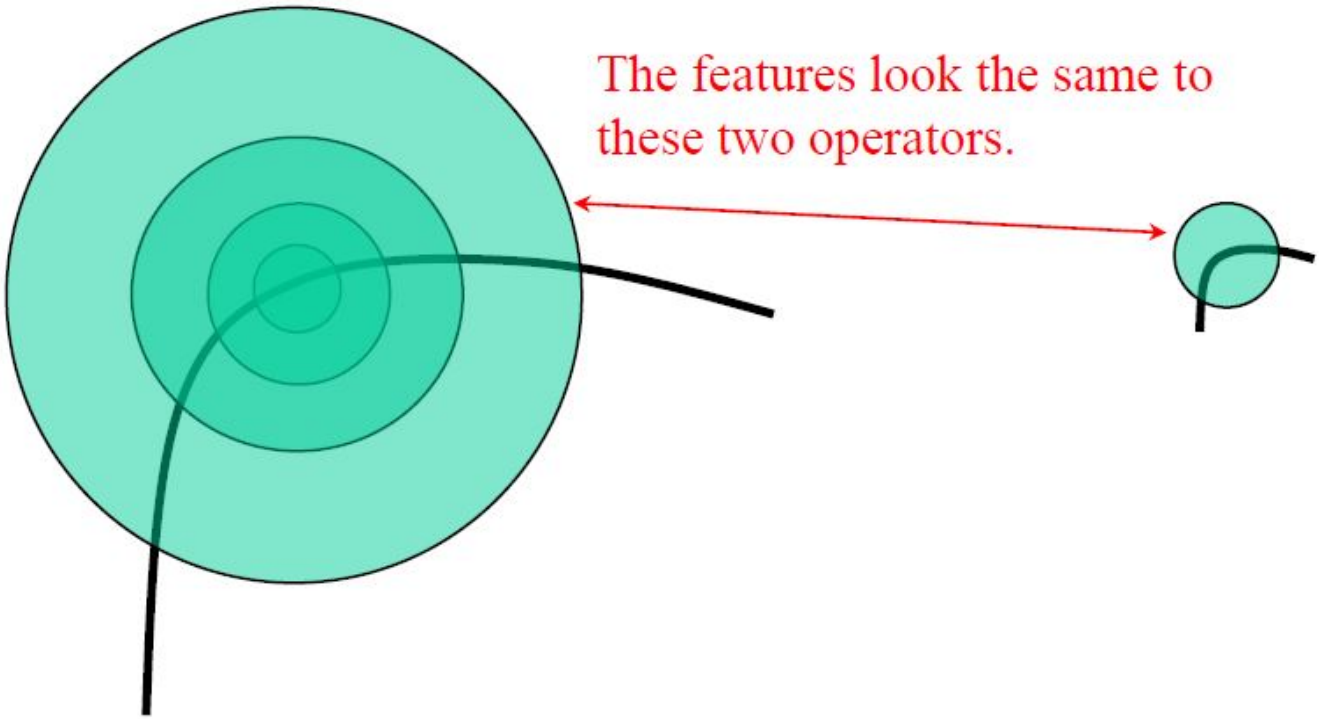
All points will be classified as edges



Corner !

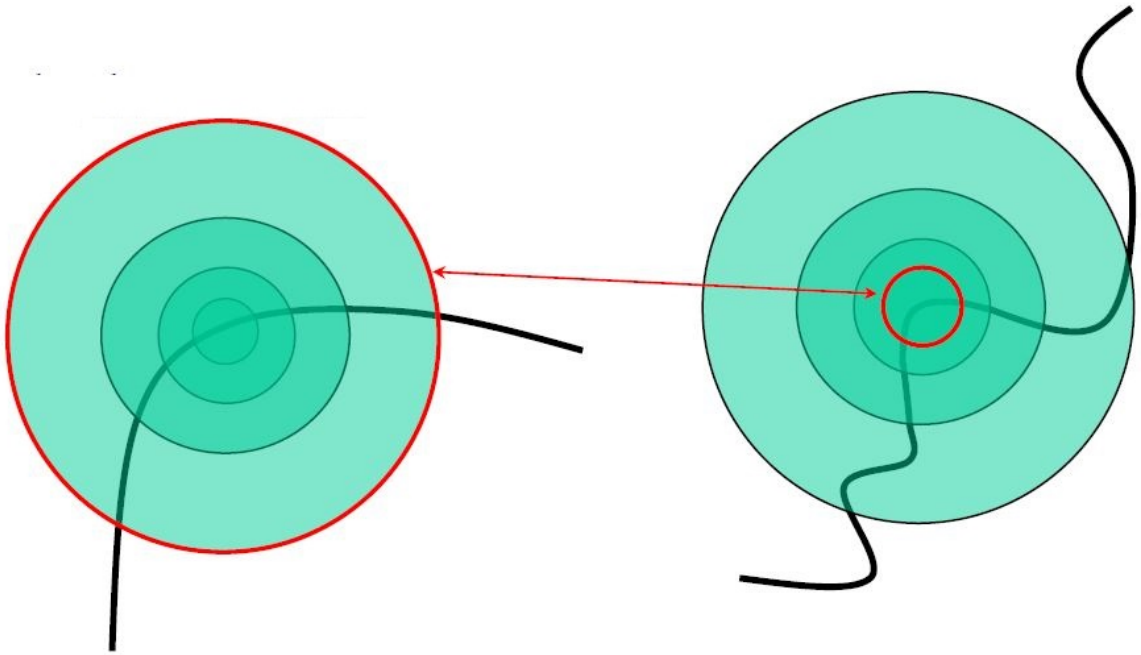
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



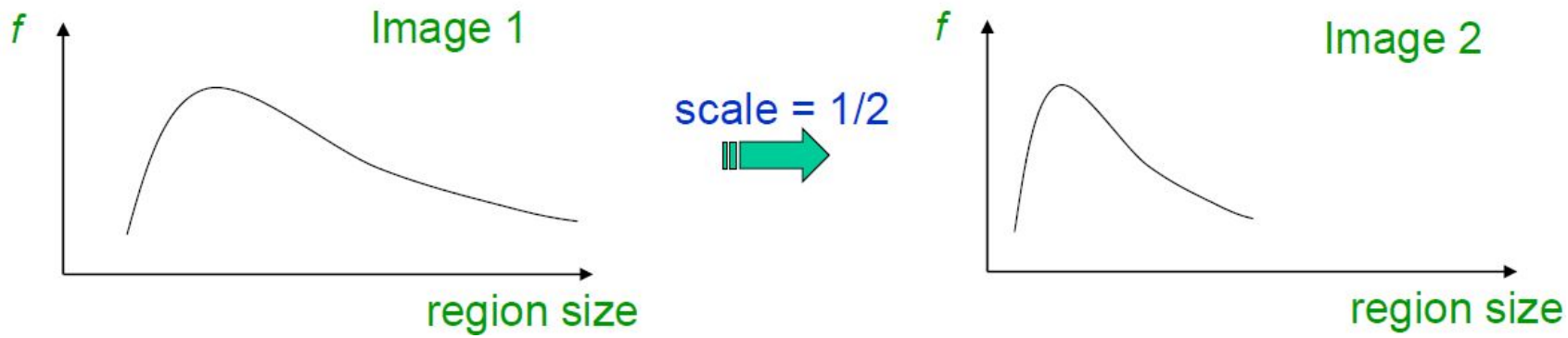
The problem

- How do we choose corresponding circles independently in each image?
- Do objects in the image have a characteristic scale that we can identify?



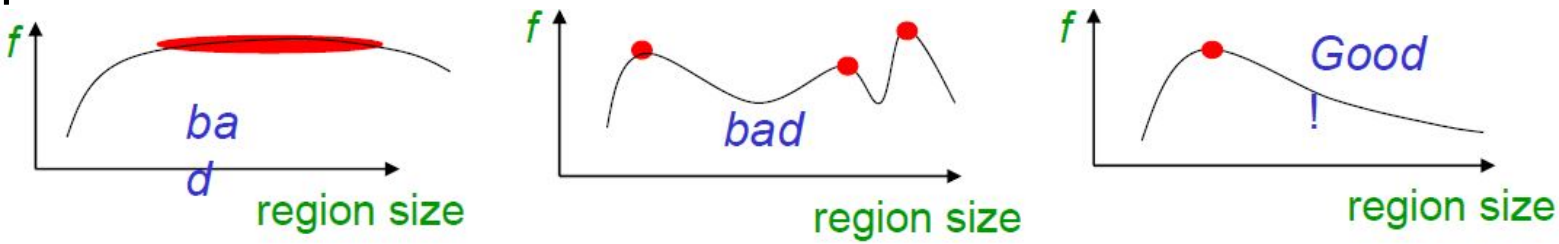
Solution

- Design a function on the region (circle), which is "scale invariant" (the same for corresponding regions, even if they are at different scales)
  - Example: average intensity. For corresponding regions (even of different sizes) it will be the same.
- For a point in one image, we can consider it as a function of region size (circle radius)



Scale Invariant Detection

- A "good" function for scale detection has one stable sharp peak



- For usual images: a good function would be a one which responds to contrast (sharp local intensity change)

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

Goal

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

Properties of SIFT

- Highly distinctive
  - A single feature can be correctly matched with high probability against a large database of features from many images.
- Scale and rotation invariant.
- Partially invariant to 3D camera viewpoint
  - Can tolerate up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
- Sometimes even day vs. night (below)
- Fast and efficient - can run in real time
- Lots of code available


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

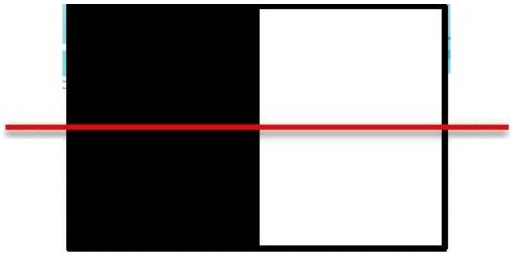
Steps for Extracting Key Points

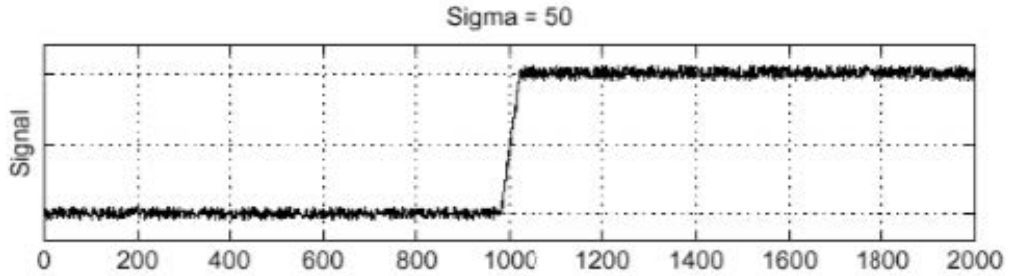
- Scale space peak selection
  - Potential locations for finding features
- Key point localization
  - Accurately locating the feature key points
- Orientation Assignment
  - Assigning orientation to the key points
- Key point descriptor
  - Describing the key point as a high dimensional vector

Scales

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

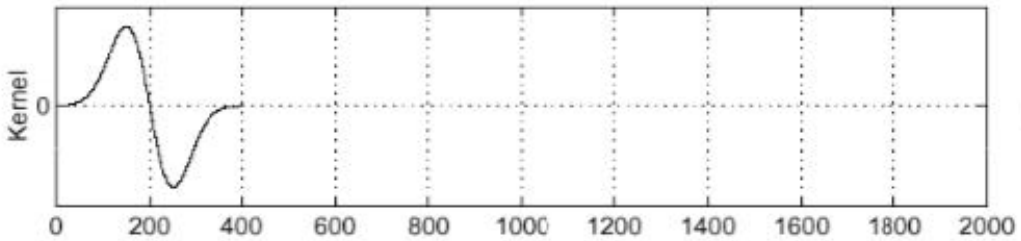
Edge detection - Rewind





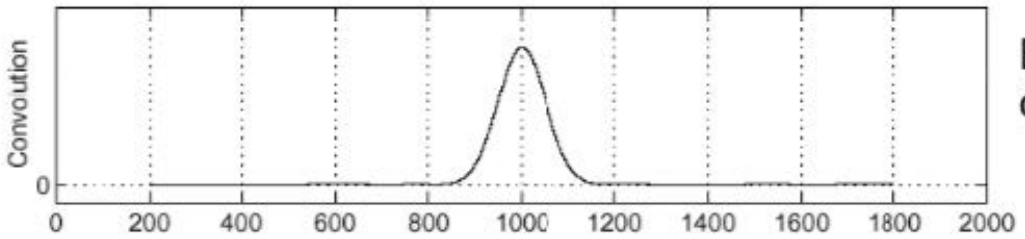
f

Edge



$\frac{d}{dx} g$

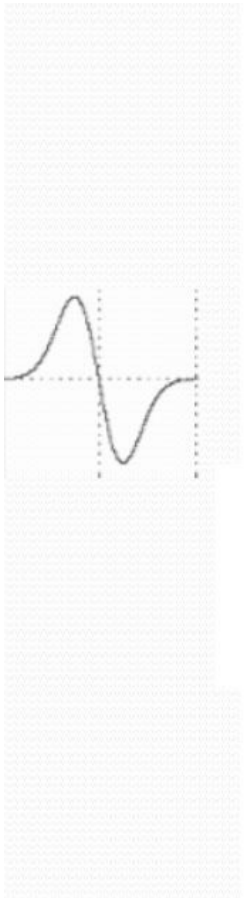
Derivative of Gaussian



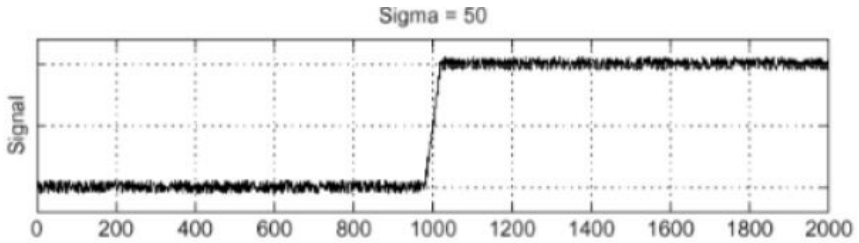
$f * \frac{d}{dx} g$

Edge = maximum of derivative

Edge detection - Rewind

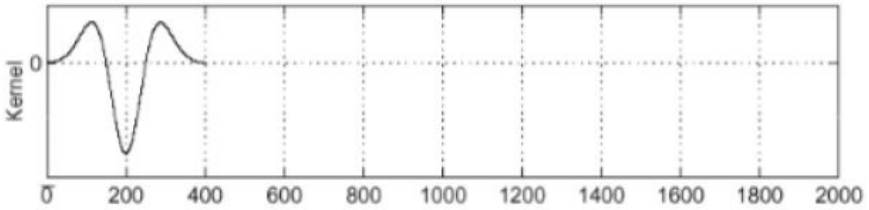


f



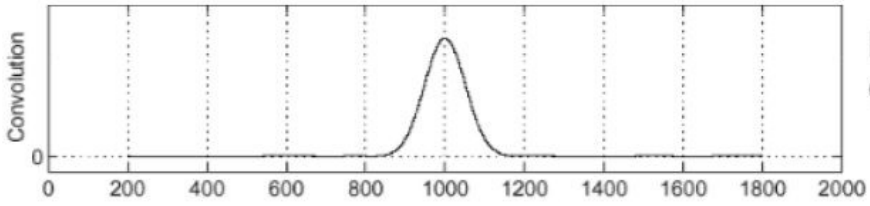
Edge

$\frac{d^2}{dx^2} g$



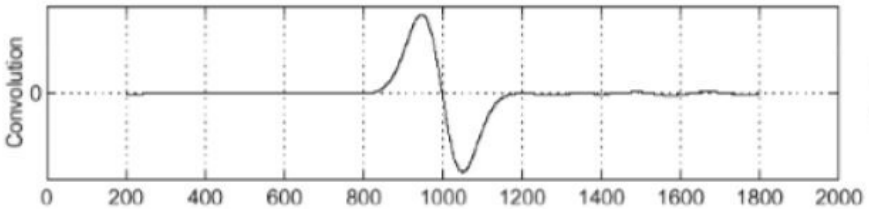
Second derivative of Gaussian (Laplacian)

$f * \frac{d}{dx} g$



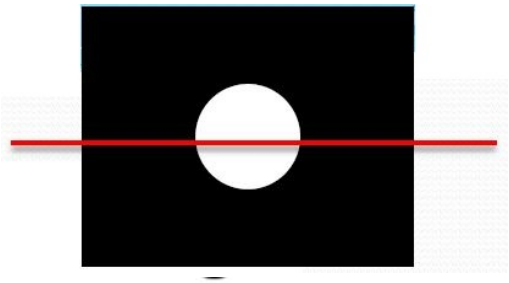
Edge = maximum of derivative

$f * \frac{d^2}{dx^2} g$

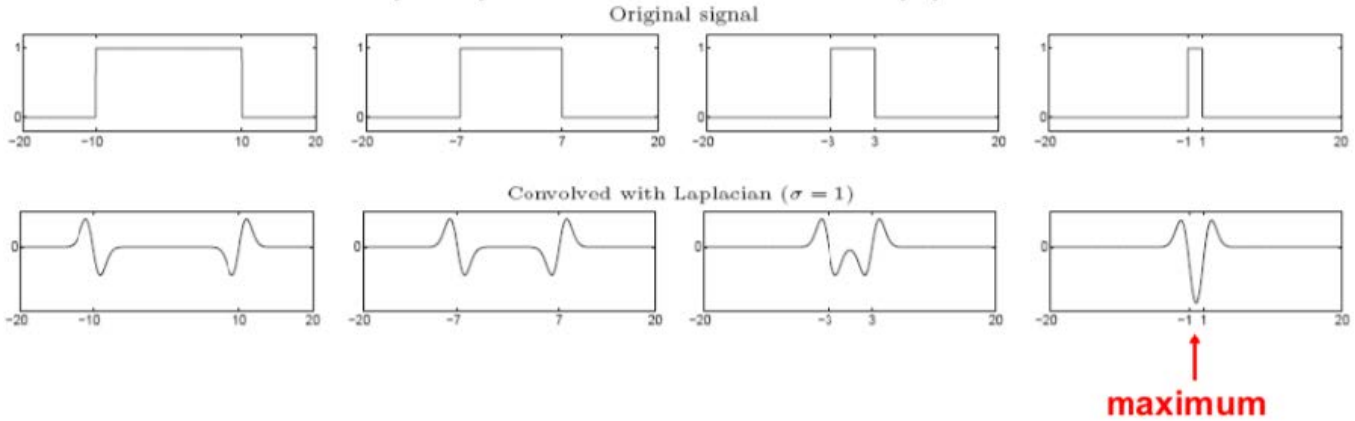


Edge = zero crossing of second derivative

Edge detection - Rewind



- Edge = ripple
- Blob = superposition of two ripples

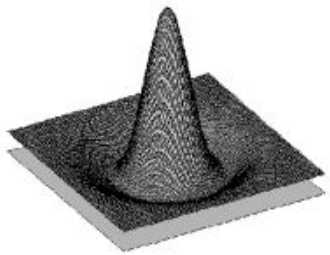


Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

Laplacian-of-Gaussian (LoG)

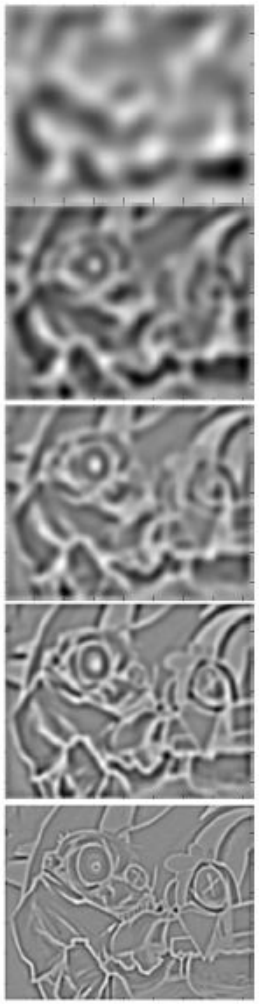
Interest points:

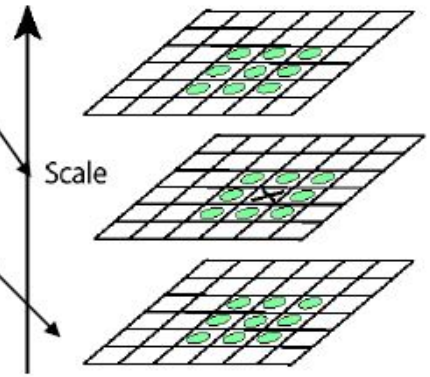
Local maxima in scale space of Laplacian-of-Gaussian



$$L_{xx}(\sigma) + L_{yy}(\sigma)$$

σ^5 σ^4 σ^3 σ^2 σ



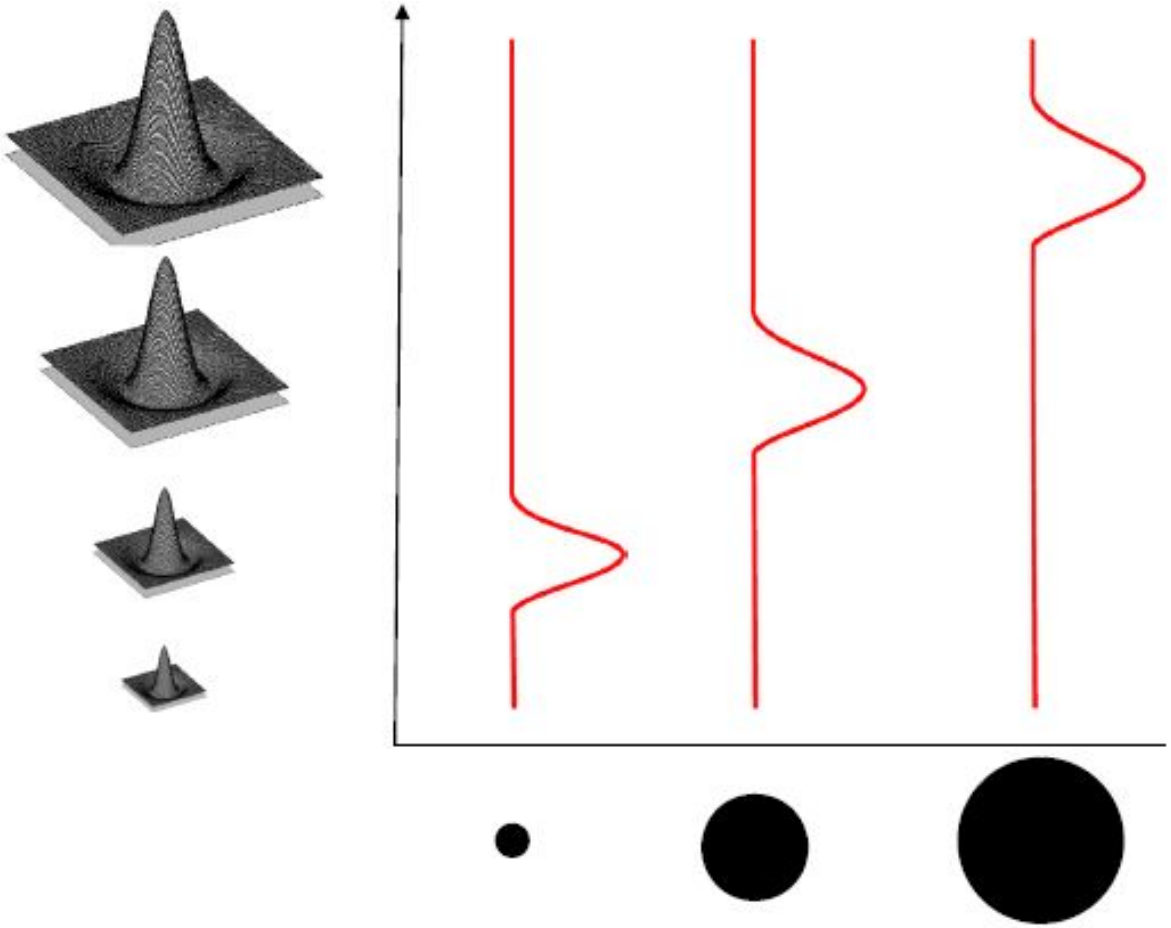


⇒ List of
(x, y, σ)

K. Grauman, B. Leibe

What is a useful Signature function?

Laplacian-of-Gaussian = "Blob" detector

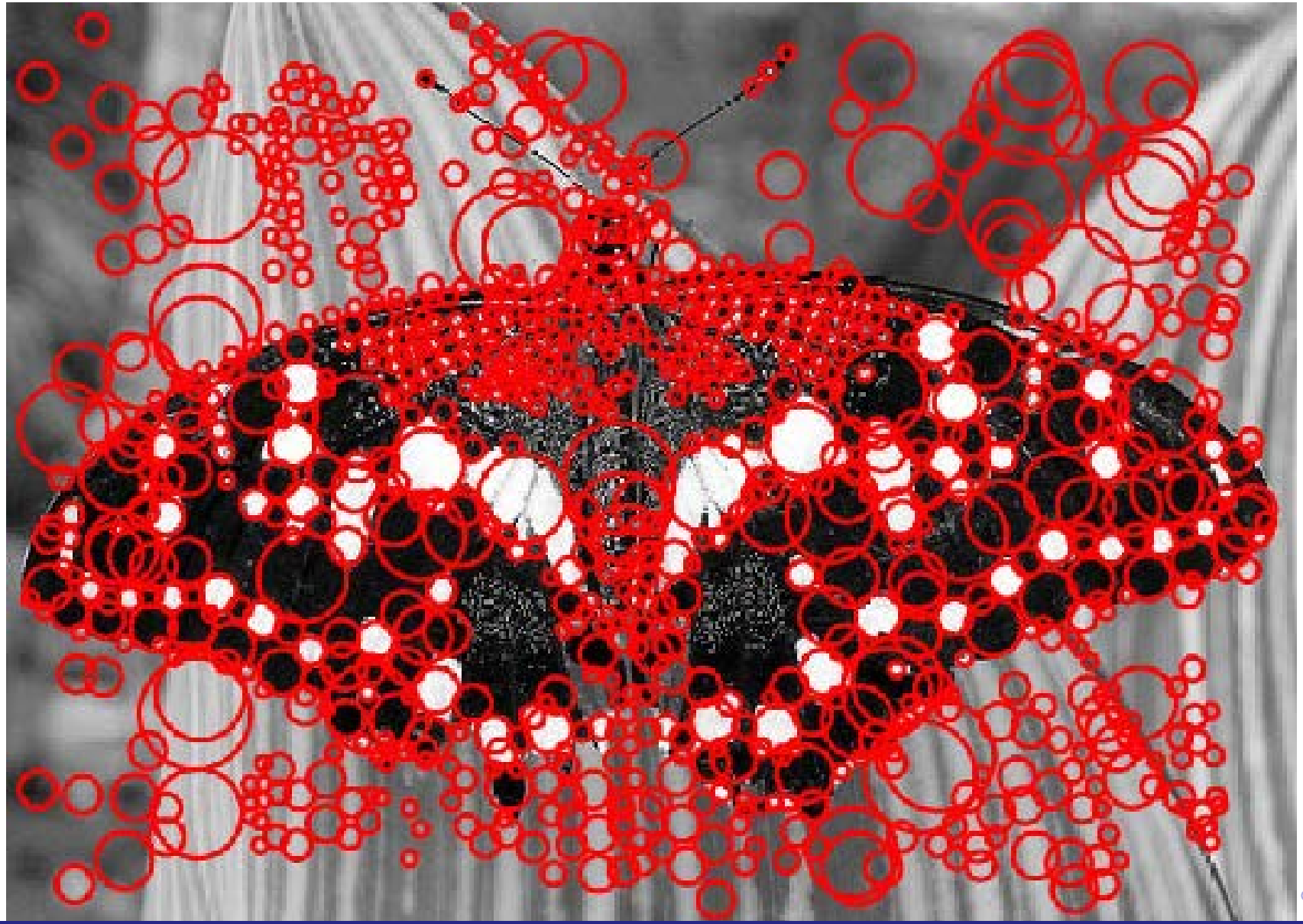


Scale-space blob detector: Example

Scale-space blob detector: Example

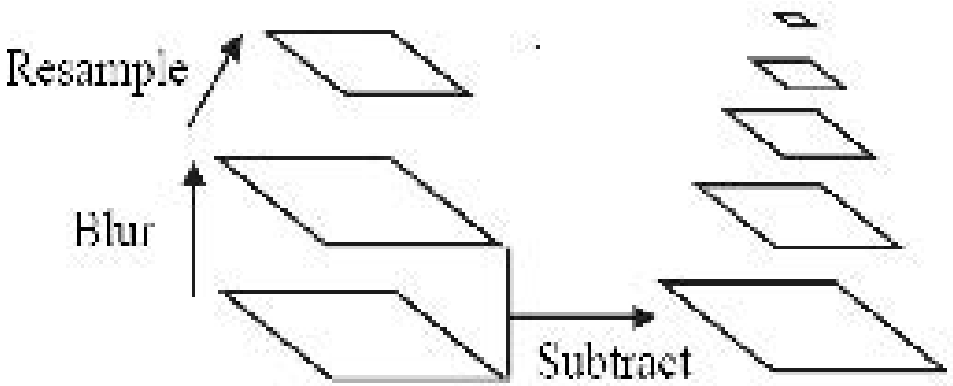
sigma = 11.9912

Scale-space blob detector: Example



Building a Scale Space

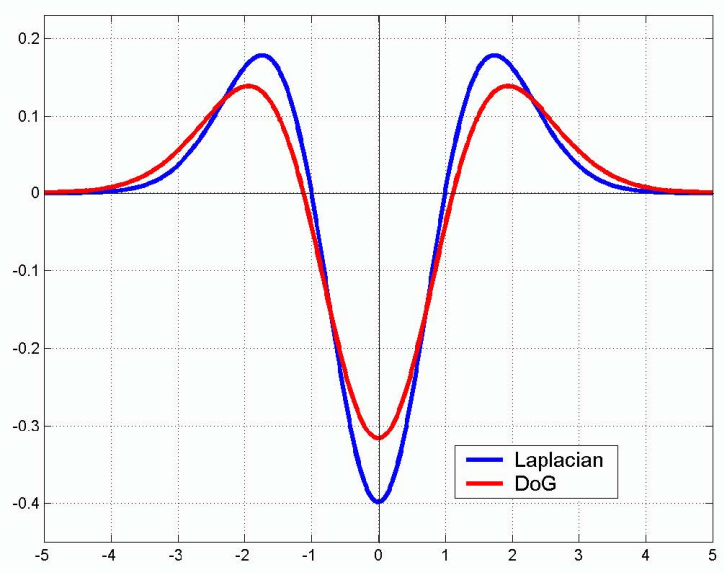
- All scales must be examined to identify scale invariant features
- An efficient function is to compute the Laplacian Pyramid (Difference of Gaussian)



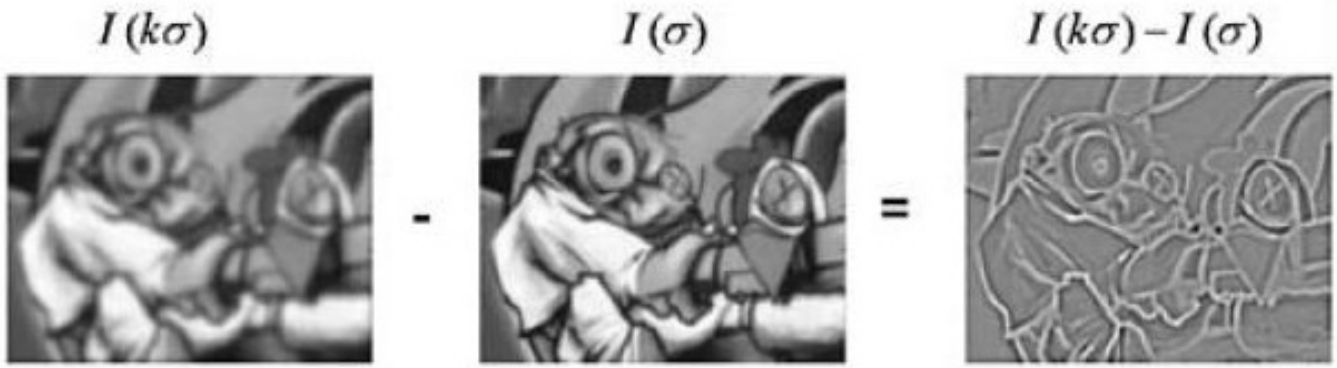
Approximation of LoG by Difference of Gaussians

We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$
(Laplacian: 2nd derivative of Gaussian)

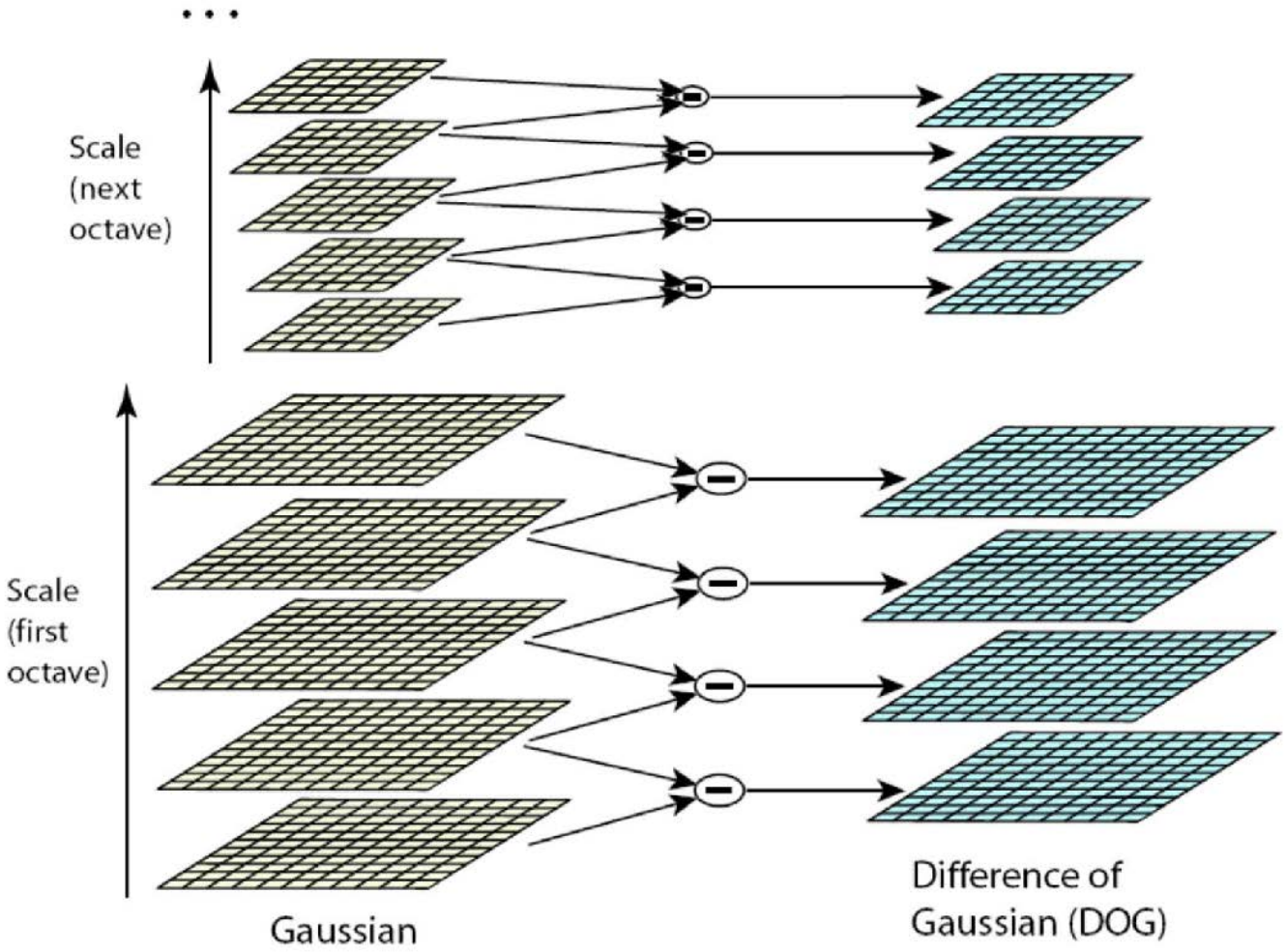


$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$
Difference of Gaussians



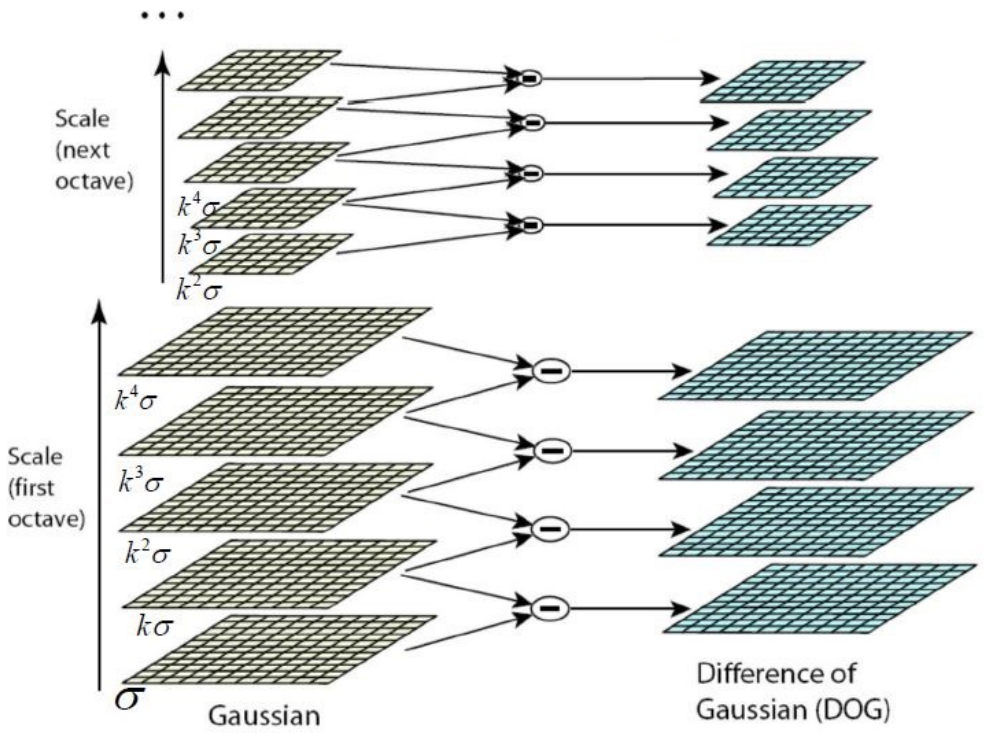
Building a Scale Space

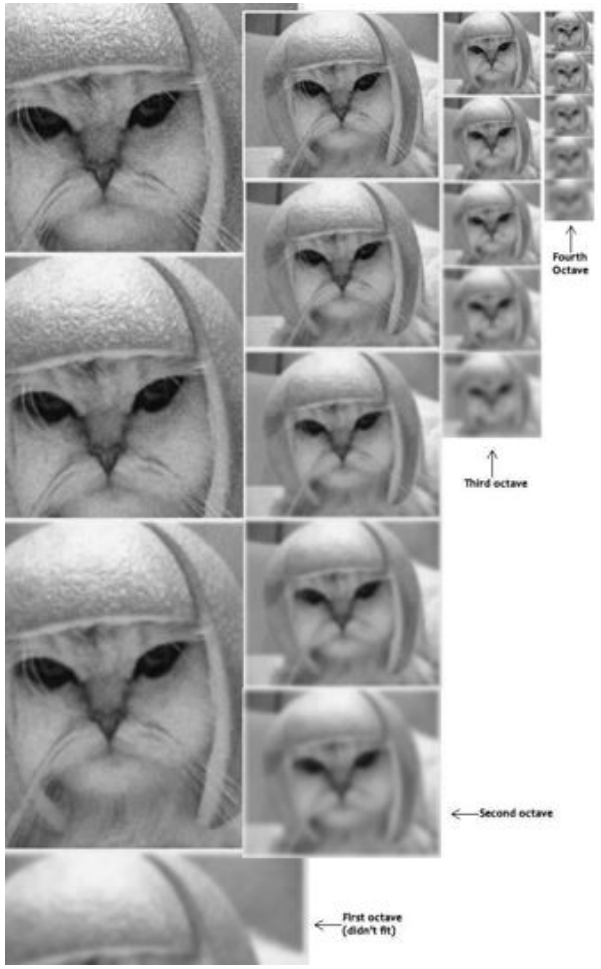
Using, $G(x, y, k\sigma) = \frac{1}{2\pi(k\sigma)^2} e^{-(x^2+y^2)/2k^2\sigma^2}$



Building a Scale Space

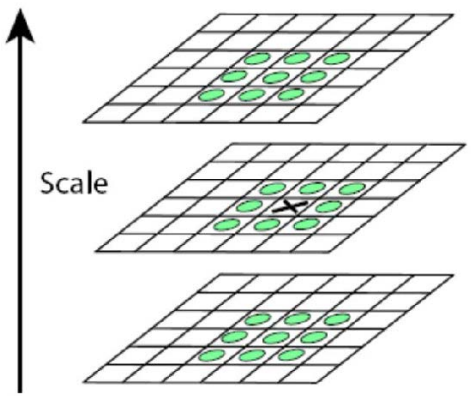
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$
$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$





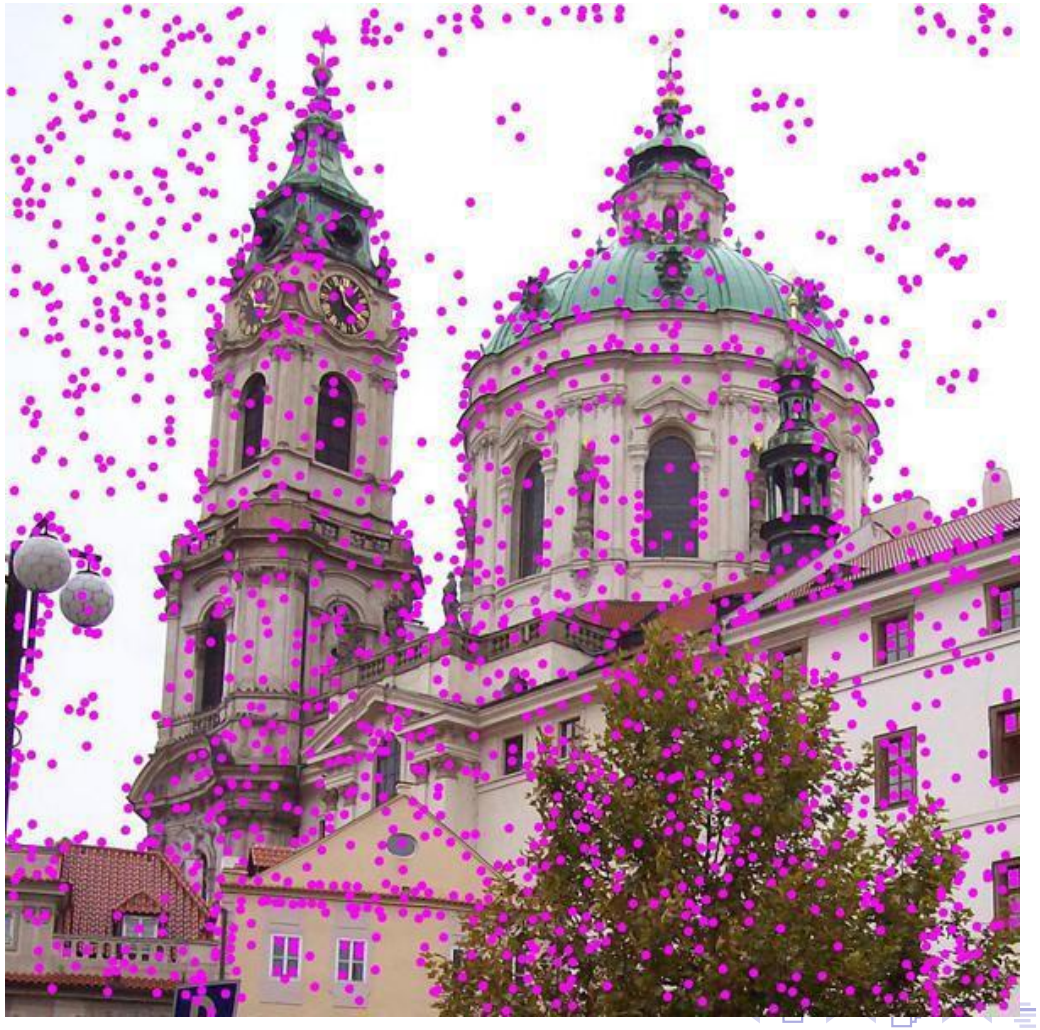
Scale Space peak detection

- 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



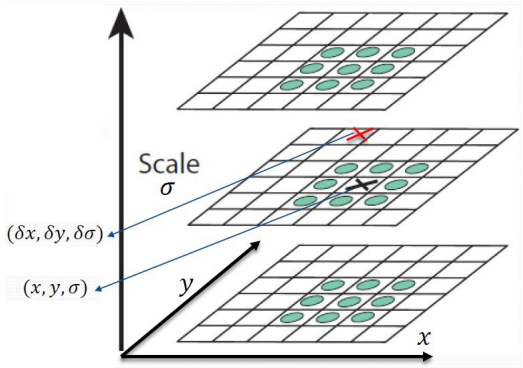
Key Point Localization

Candidates are chosen from extrema detection



Feature Point Localization - Initial outlier rejection

1. Low contrast candidates
2. Poorly localized candidates along an edge
   - let $x_o = [x, y, \sigma]^T$ and $z = [\delta x, \delta y, \delta\sigma]^T$
   - Taylor series expansion of DoG, D.
     $$D(x_o + x) = D(X_o) + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$
   - Minima or maxima is located at
     $$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}$$
   - Value of $D(x)$ at minima/maxima must be large, $|D(x)| > th$.



Feature Point Localization - Initial outlier rejection

Solution(1/3)

$$D(x_o + x) \approx D(X_o) + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \tag{1}$$

To find maxima or minima, we know that

$$\frac{\partial D(x_o + x)}{\partial x} = 0 \tag{2}$$

Derive 1 w.r.t. x

$$\frac{\partial D(x_o+x)}{\partial x} = 0 + \frac{\partial D}{\partial x} + \frac{1}{2} \times 2 \times \frac{\partial^2 D}{\partial x^2} \hat{x}$$

Put in 2, which becomes

$$\frac{\partial D}{\partial x} + \frac{\partial^2 D}{\partial x^2} \hat{x} = 0 \tag{3}$$

By solving 3, we get

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x} \tag{4}$$

Solution(2/3)

1 can be modified as

$$D(x_o + \hat{x}) \approx D(X_o) + \frac{\partial D^T}{\partial x}\hat{x} + \frac{1}{2}\hat{x}^T\frac{\partial^2 D}{\partial x^2}\hat{x} \tag{5}$$

Put the value of $\hat{x}$ computed in 4. 5 becomes

$$D(x_o + \hat{x}) \approx D(X_o) + \frac{\partial D^T}{\partial x}\hat{x} + \frac{1}{2}\left(-\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}\right)^T\frac{\partial^2 D}{\partial x^2}\hat{x} \tag{6}$$

$$D(x_o + \hat{x}) \approx D(X_o) + \frac{\partial D^T}{\partial x}\hat{x} - \frac{1}{2}\left(\frac{\partial D}{\partial x}\right)^T\left(\frac{\partial^2 D}{\partial x^2}^{-1}\right)^T\frac{\partial^2 D}{\partial x^2}\hat{x} \tag{7}$$

In 7, $\left(\frac{\partial^2 D}{\partial x^2}^{-1}\right)^T$ represents the Hessian matrix which is symmetric in nature.

Solution(3/3)

7 is solved to

$$D(x_o + \hat{x}) \approx D(X_o) + \frac{\partial D^T}{\partial x}\hat{x} - \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \tag{8}$$

$$D(x_o + \hat{x}) \approx D(X_o) + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \tag{9}$$

If the result of (9) is less than 0.03, it is discarded saying it's a low contrast point.
All pixels are normalized between 0 and 1.
Note that $\frac{\partial D}{\partial z} = \begin{bmatrix} \frac{\partial D}{\partial x} & \frac{\partial D}{\partial y} & \frac{\partial D}{\partial \sigma} \end{bmatrix}$

After Feature Point Localization and thresholding

Removing points on the edges - Further outlier rejection

- DoG has strong response along edge
- The prinicpal curvatures can be computed from a 2×2 Hessian matrix, H, computed at the location and scale of the key point

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

let α and β be the eigen values of the matrix

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta}$$

let r represents the ratio of the eigen values i.e., $r = \frac{\alpha}{\beta}$, then $\alpha = r\beta$

Putting in equation above, we get

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

Keep only those values which satisfies the following condition

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \text{ for } r = 10$$

Further explaination (for reading)

- Analogous to Harris corner detector
- Compute Hessian of D

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

where,

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$
$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

- Remove outliers by evaluating

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

where $r = \frac{\alpha}{\beta}$

Removing points on the edges - Further outlier rejection

- Following quantity is minimum when $r = 1$
- It increases with r

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

  where $r = \frac{\alpha}{\beta}$
- Eliminate key points if $r > 10$

Removing points on the edges - Further outlier rejection

After removal of edges points

Descriptor - Why do we need it?

Orientation Assignment

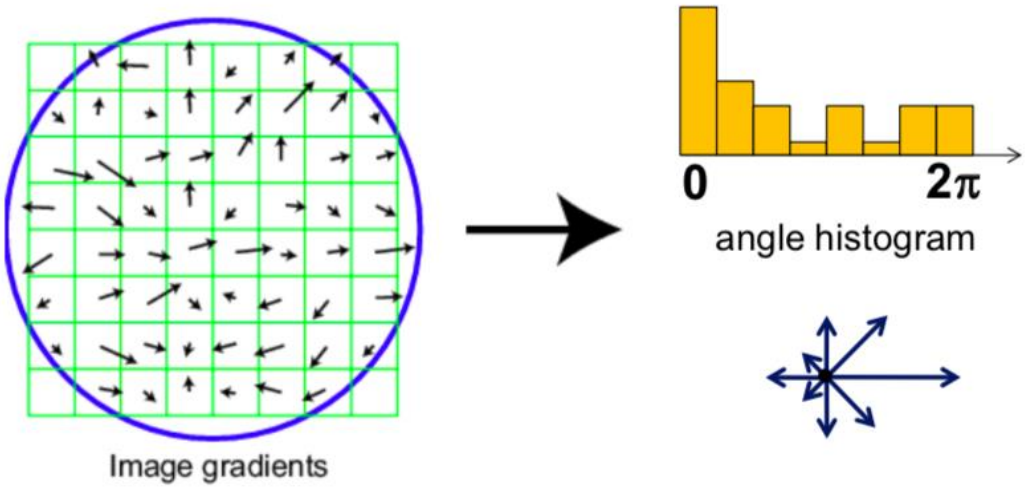
- To achieve rotation invariance
- Compute central derivatives, gradient magnitude and direction of L (smooth image) at the scale of key point (x, y)

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

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

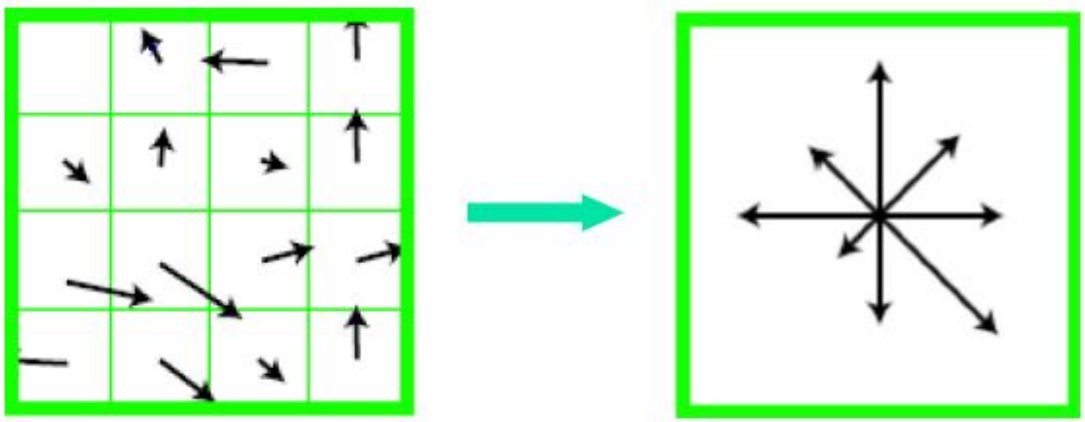
Orientation Assignment

- Create a weighted direction histogram in a neighborhood of a key point (36 bins i.e., from $360^o/10$)



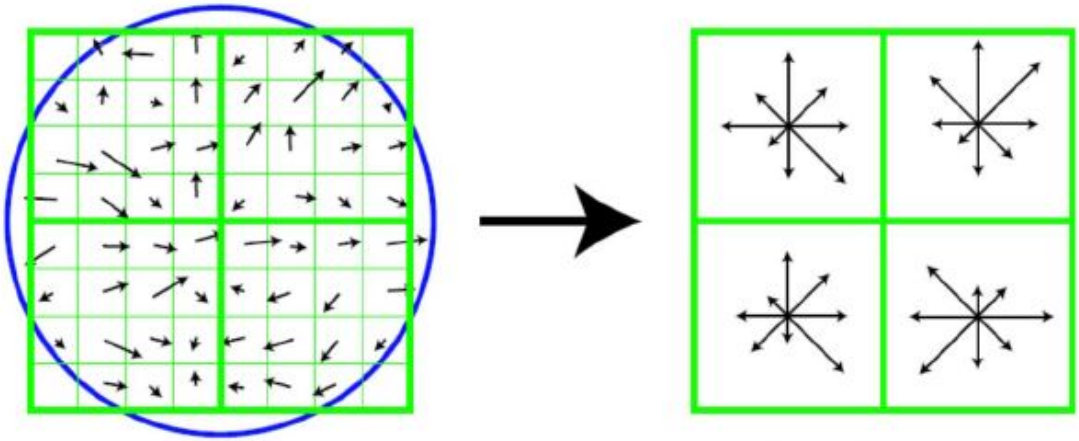
Orientation Assignment

- Select the peak as direction of the key point
- Introduce additional key points (same location) at local peaks (within 80% of max peak) of the histogram with different directions



Extraction of Local Image Descriptors at Key Points

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

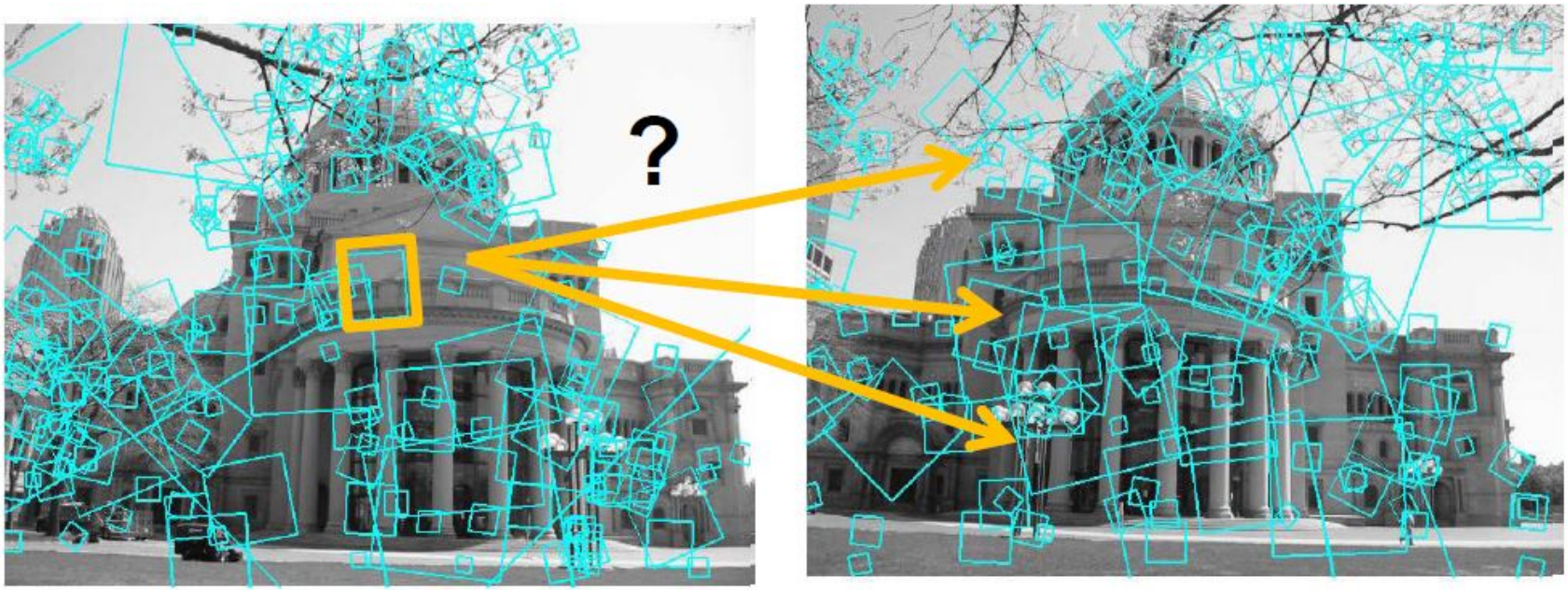


Key point matching

- 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. What if you get ratio = 0.8? It is ambiguous.

Matching local features

Matching local features



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

Ambiguous matches

Image 1

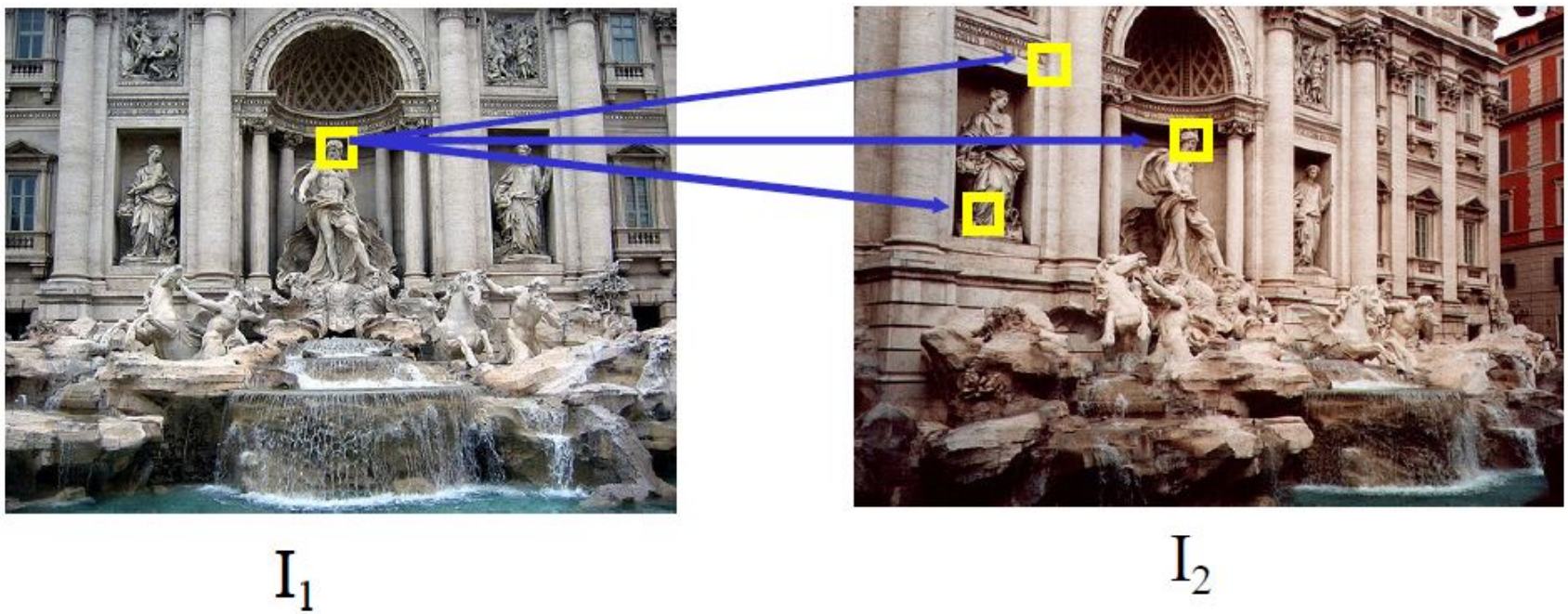
Image 2

- 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.

Matching local features

Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors.
2. Test all the features in I2, find the one with min distance.



L2 Distance

- Essentially, comparing two arrays of data
- Let $H_1(k)$ and $H_2(k)$ be two arrays of data f length n

$$d(H_1, H_2) = \sqrt{\sum_k (H_1(k) - H_2(k))^2}$$

- Smaller the distance metric, better the match
- Perfect match is achieved when $d(H_1, H_2) = 0$

Normalized Correlation

$$d(H_1, H_2) = \frac{\sum_k [(H_1(k) - \bar{H}_1)(H_2(k) - \bar{H}_2)]}{\sqrt{\sum_k (H_1(k) - \bar{H}_1)^2} \sqrt{\sum_k (H_2(k) - \bar{H}_2)^2}} \quad (10)$$

- where $\hat{H}_i = \frac{1}{N} \sum_{k=1}^{N} H_i(k)$
- Larger the distance metric, better the match
- Perfect match when $d(H_1, H_2) = 1$