Interest Points

Computer Vision (CS0029)

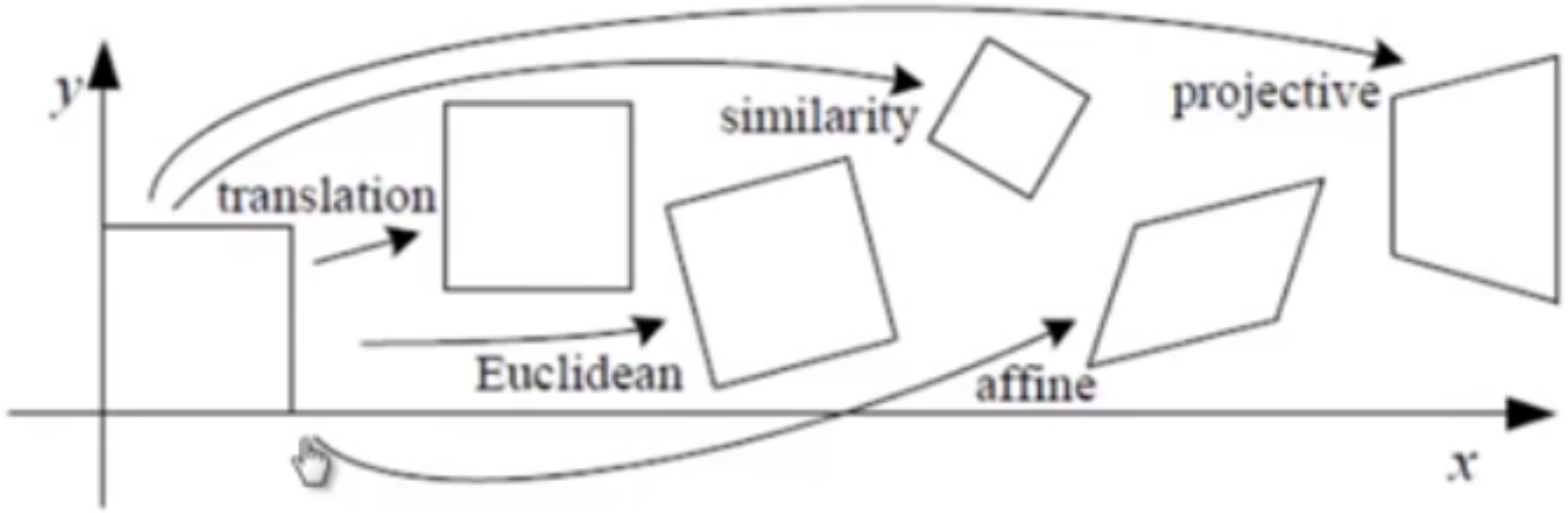
Outline

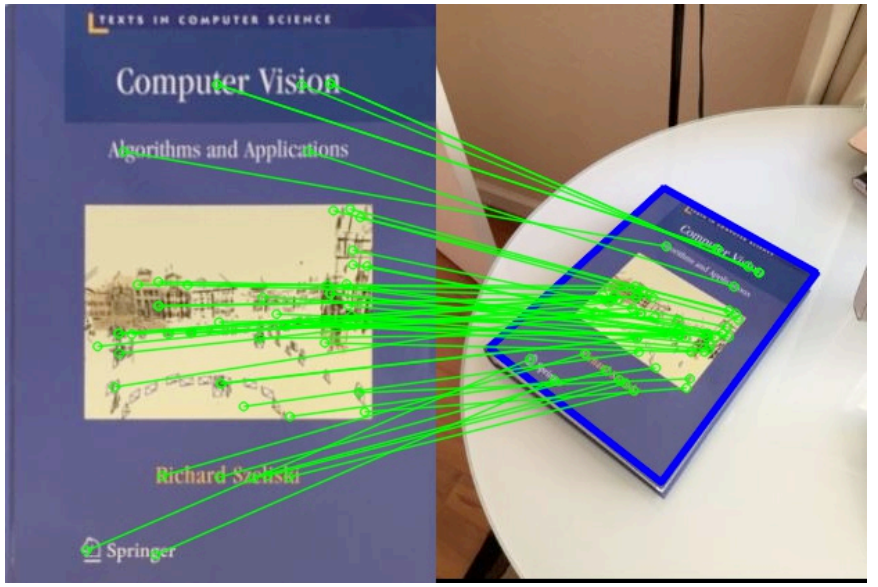
- **Local Invariant Features**
  - **Motivation**
  - **Requirements, Invariances**
- Keypoint Localization
  - Feature from Accelerated Segment Test (FAST)
  - Harris
  - Shi-Tomasi
- Scale Invariant Region Selection
  - Automatic scale selection
  - Laplacian-of-Gaussian detector
  - Difference-of-Gaussians detector
- Local Descriptors
  - Orientation normalization
  - SIFT

Consider Image Matching

Consider Image Matching

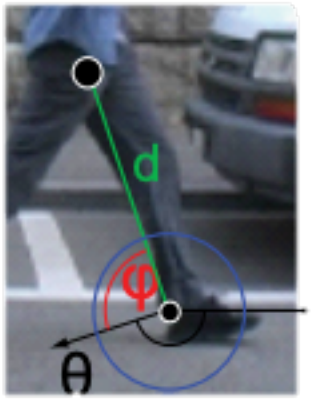
- Estimate the transformation between two images
  - We have to find the matching point between two images first
  - We need to find local features

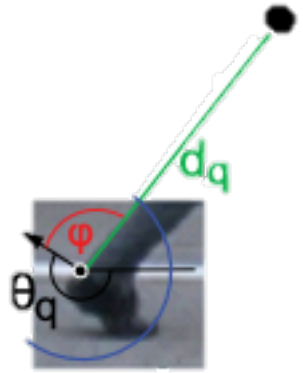


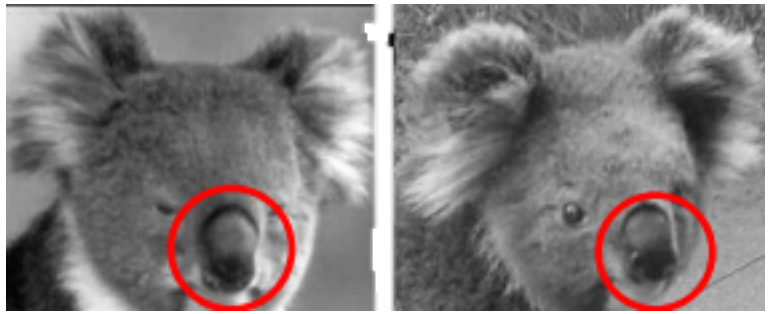


Global vs. Local Matching

- Global image representation can be easily corrupted (e.g., by even small occlusion)
- Instead, describe and match using multiple local regions
- Increased robustness to occlusions, articulation, within-category variation







Goal of Local Features

- Find points in an image that can be:
  - Found in other images
  - Found precisely (repeatability) – well localized
  - Found reliably – well matched

- Want to compute a fundamental matrix to recover geometry
- Robotics/vision: see how a bunch of points move from one frame to another. Allows computation of how camera moved -> depth -> moving objects
  - E.g. structure from motion, https://www.youtube.com/watch?v=i7ierVkXYa8
- Panorama

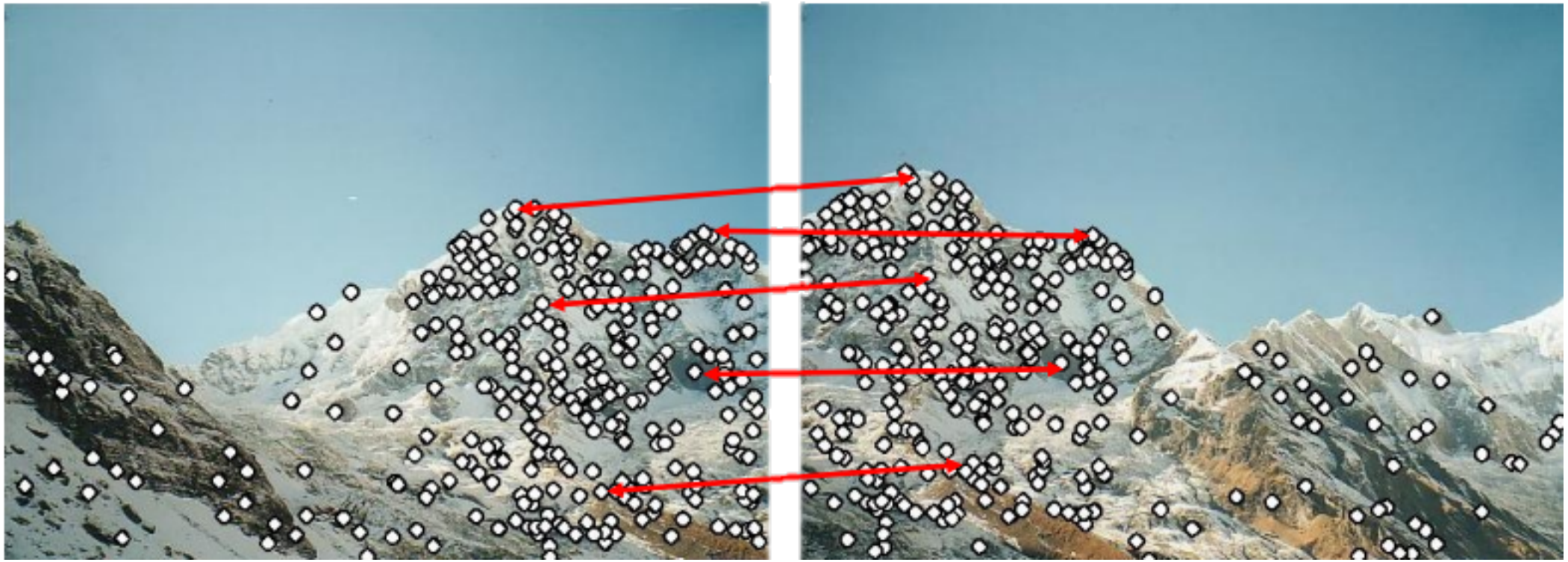
Application: Panorama

Application: Panorama

Detect features (feature points) in both images

Application: Panorama

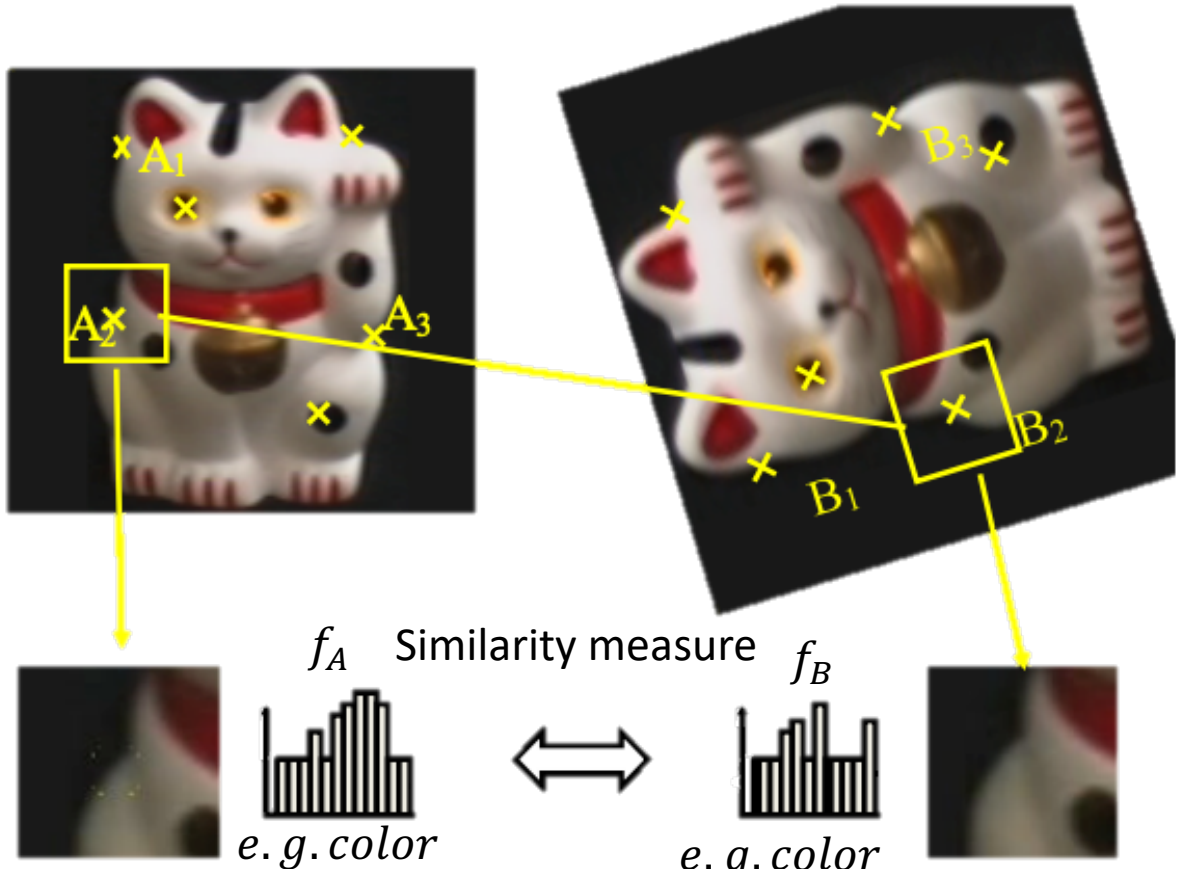
Match: find corresponding pairs



Application: Panorama

Use the pairs to transform/align images

General Approach



- 1. Find a set of distinctive keypoints
- Define a region around each keypoint
- Extract the region content
- Compute a local descriptor from the region
- Match local descriptors

Common Requirements

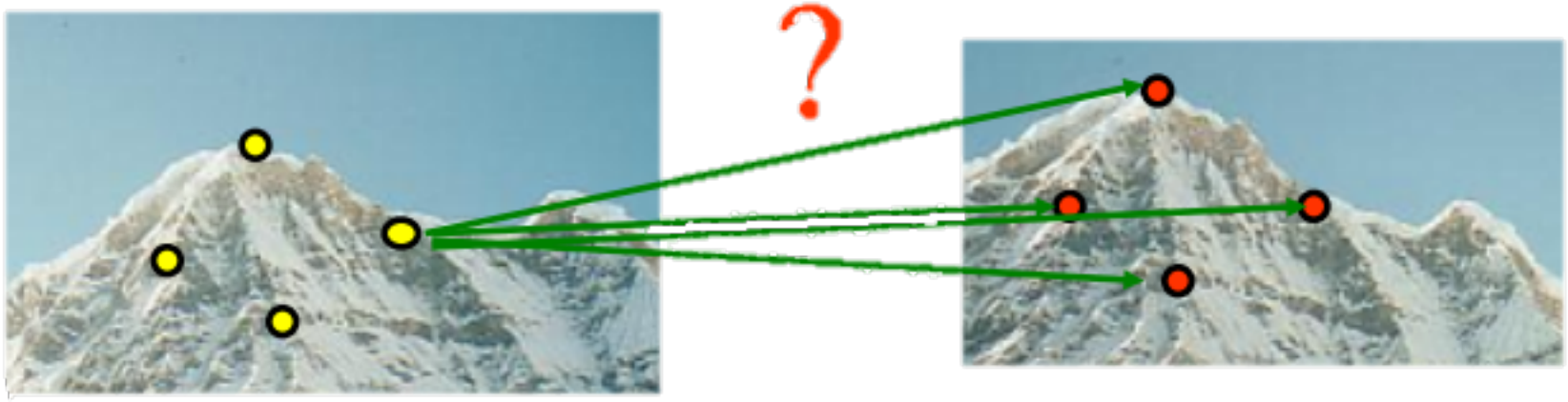
- Problem 1:
  - Detect the same points independently in both images

No chance to match!

We need a repeatable detector!

Common Requirements

- Problem 1:
  - Detect the same points independently in both images
- Problem 2:
  - For each point, correctly identify the corresponding point



We need a reliable and distinctive descriptor

More Motivation

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

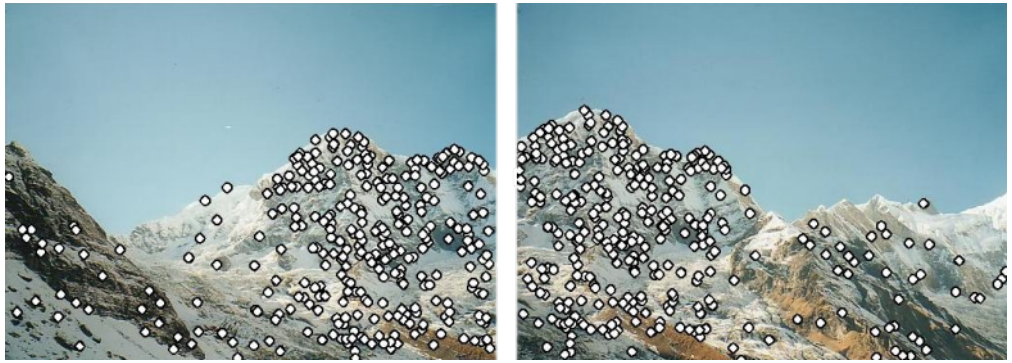
Practice

- Select the ones that can serve as good features

Invariance: Geometric Transformations

Invariance: Photometric Transformations

Key Takeaways

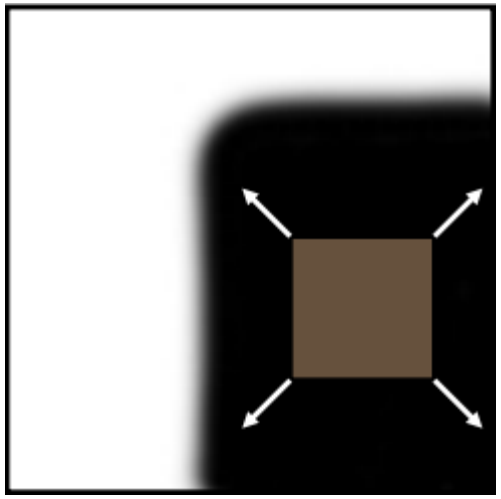


- Repeatable/precision:
  - The same feature can be found in several images despite geometric and photometric transformation
- Saliency/matchability
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer feature than image pixels
  - close to real-time performance
- Locality
  - A feature occupies a relatively small area of the image, robust to clutter and occlusion
- Invariance
  - Invariant to translation, rotation, scale changes, and other simple (affine) transformations
  - Robust to lighting variations, noise, blur, quantization

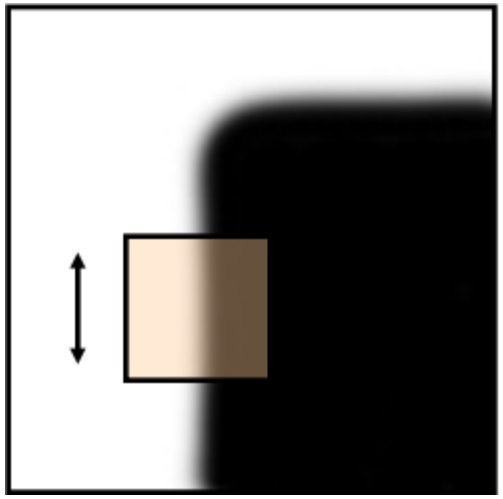
Outline

- Local Invariant Features
  - Motivation
  - Requirements, Invariances
- **Keypoint Localization**
  - **Feature from Accelerated Segment Test (FAST)**
  - **Harris**
  - **Shi-Tomasi**
- Scale Invariant Region Selection
  - Automatic scale selection
  - Laplacian-of-Gaussian detector
  - Difference-of-Gaussians detector
- Local Descriptors
  - Orientation normalization
  - SIFT

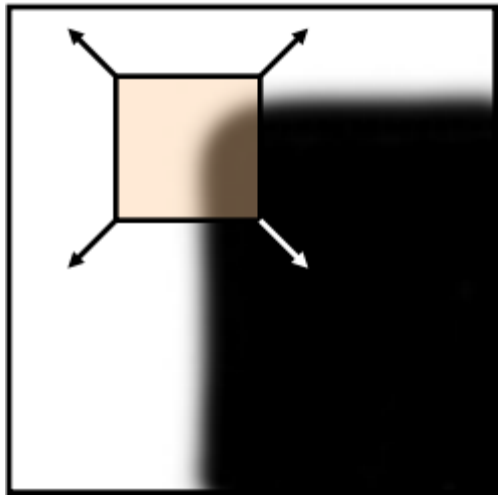
Flat vs. Edge vs. Corner (Precision)



“flat” region: no change in all directions



“edge”: no change along the edge direction



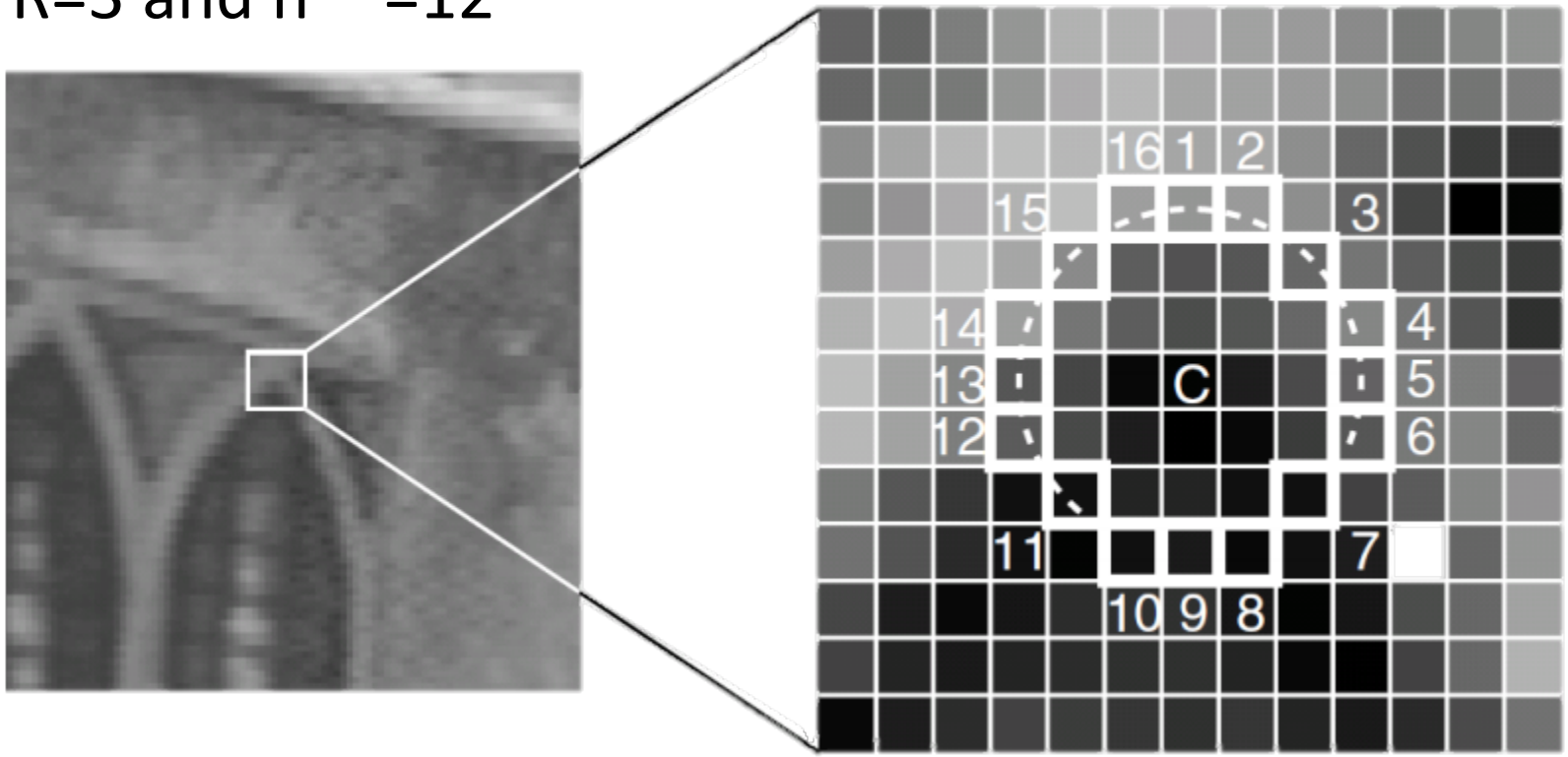
“corner”: significant change in all directions

Features from Accelerated Segment Test (FAST)

- Compare local neighborhood around each pixel to determine if the pixel is a good feature point
- For each pixel x
  - Look at the pixels on the border of a circle of radius r around x
  - Let n be the number of contiguous pixels (check for wrap-around!) whose intensities are either
    - $all\ >\ (I(x) + T)$ or $all\ <\ (I(x) - T\ )$
    - $I(x)$ is the intensity at pixel x and T is a threshold
  - If $n\ \geq\ n^*$then the pixel is considered a feature
    - Original paper suggest using $r = 3$ (yields 16 pixels on border) and $n^* = 12$
    - Later results suggest using $r = 3$ (yields 16 pixels on border) and $n^* = 9$

FAST Example

- R=3 and n^*=12



- Techniques can be employed to decrease computation time

FAST Example ($r = 3, n^* = 12$)

Input

T=10

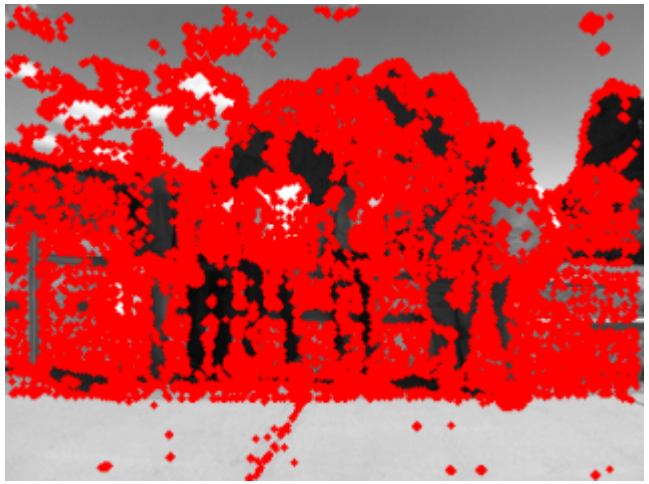
T=30

T=50

FAST Example ($r = 3, n^* = 9$)

Input

T=10



T=30

T=50

Harris Detector

- Based on the matrix M, which is a 2x2 correlation matrix from image derivatives around a pixel (recall good feature of KLT)

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Sum over image region-
the area we are checking for corner

Window/Weighting function

Gradients with respect to x and y

- To evaluate the "cornerness" of a particular pixel location from its local region

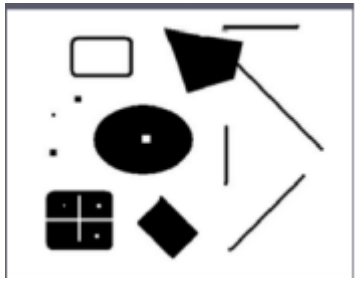
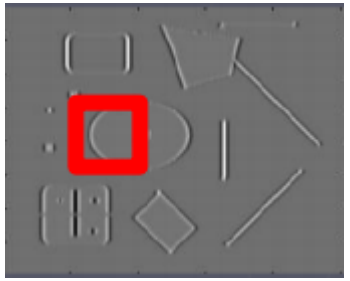
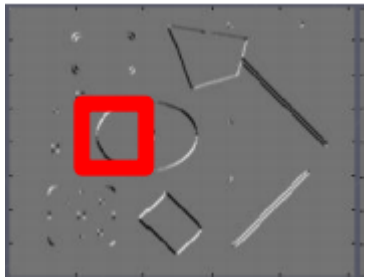


Image I



I_x

I_y



$I_x I_y$

Gradients

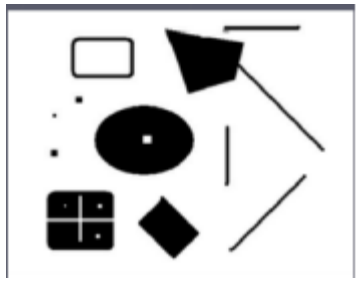
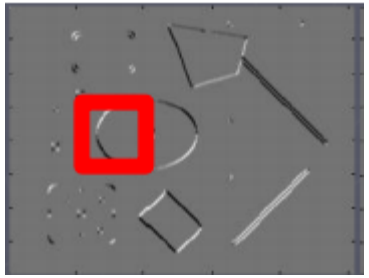


Image I

I_x

I_y



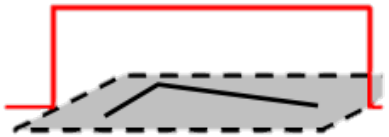
$I_x I_y$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Window/Weighting Function $w(x, y)$

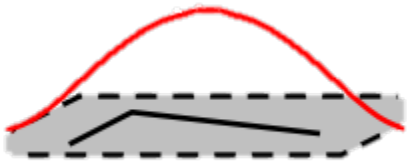
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Option 1: uniform window
  - Sum over square window
  - $M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$



1 in window, 0 outside

- Option 2: Gaussian window
  - Gaussian performs weighted sum
  - $M = \sum_{x,y} g(\sigma) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$



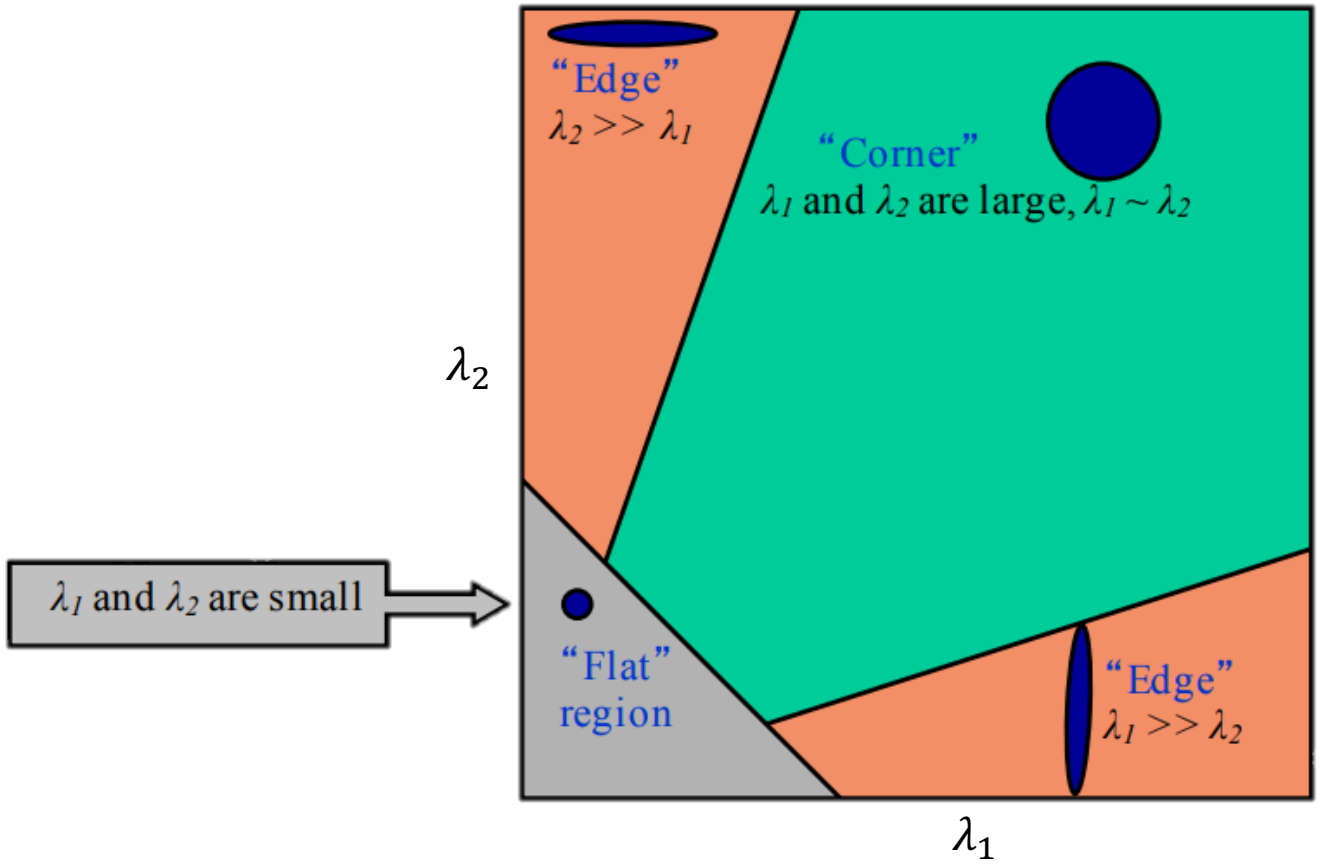
Gaussian

Eigenvalues of M

- Let $\lambda_1 \geq \lambda_2$ be the Eigenvalues of M
  - $\lambda_1 < e$: intensity is nearly constant over patch
  - $\lambda_1 >> \lambda_2$: edge was found
  - $\lambda_2 > T$: corner or textured patch found

Interpreting the Eigenvalues

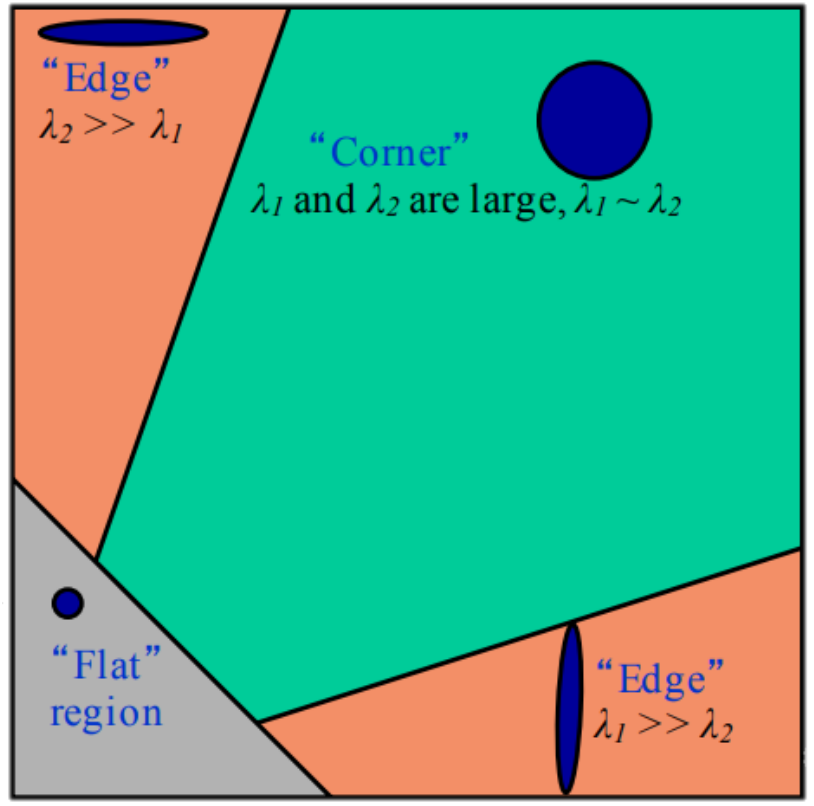
- Classification of image points using eigenvalues of M
  - No ordering of eigenvalues



Harris Corner Response Function

- $R = \det(M) - \alpha\, trace(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$
  - α:constant (0.04 to 0.06)

- Fast approximation
  - R depends on only eigenvalue of M
  - But, do not compute them (still fast in the 80's)



Harris Algorithm

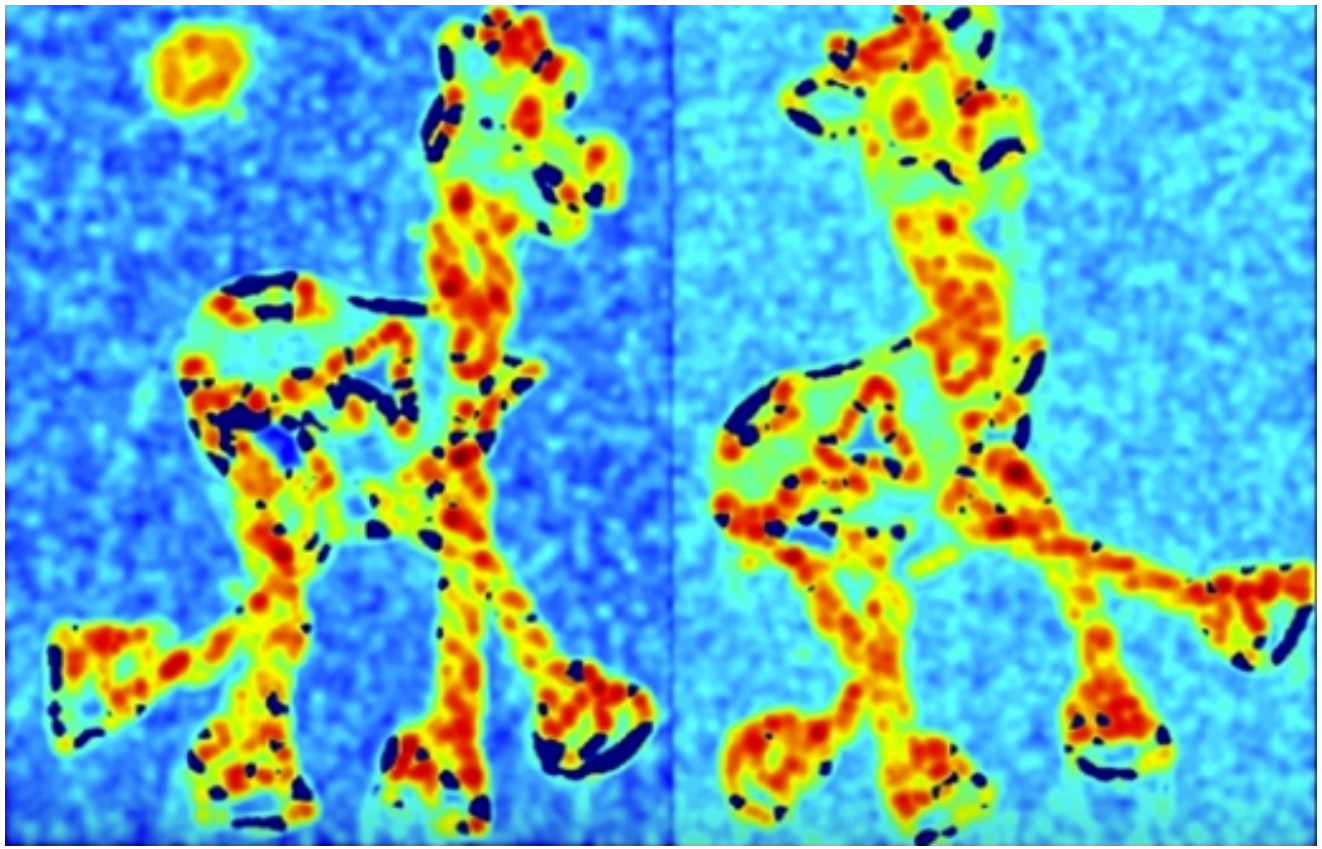
1. Compute Gaussian derivative at each pixel
2. Compute second moment matrix M in a Gaussian window around each pixel
3. Compute corner response function R
4. Threshold R
5. Find local maxima of response function (nonmaximum suppression)

Harris Detector Example

- Head has been rotated

Harris Detector Example

- R function

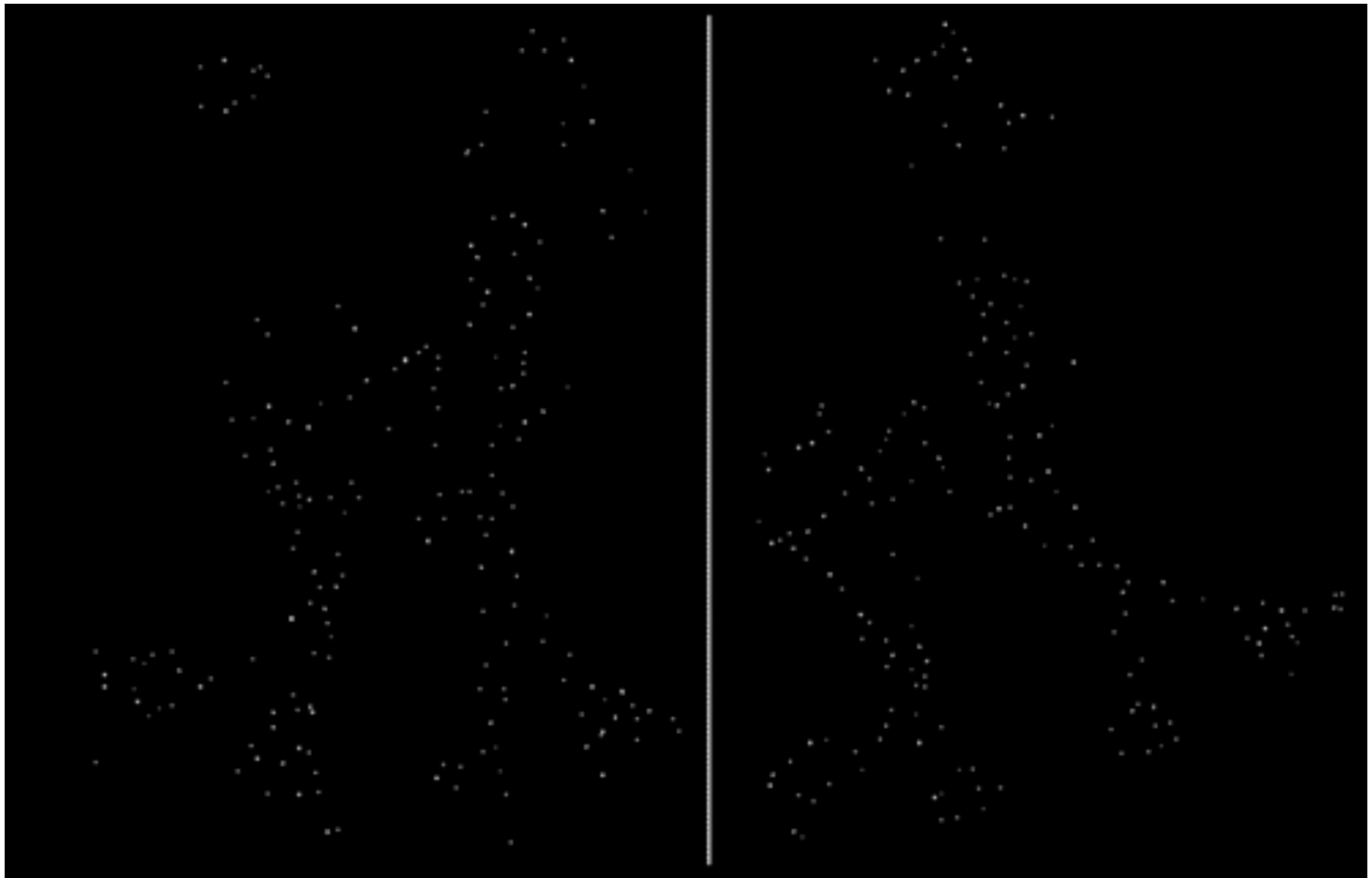


Harris Detector Example

- Thresholding

Harris Detector Example

- Take only the point of local maxima of R

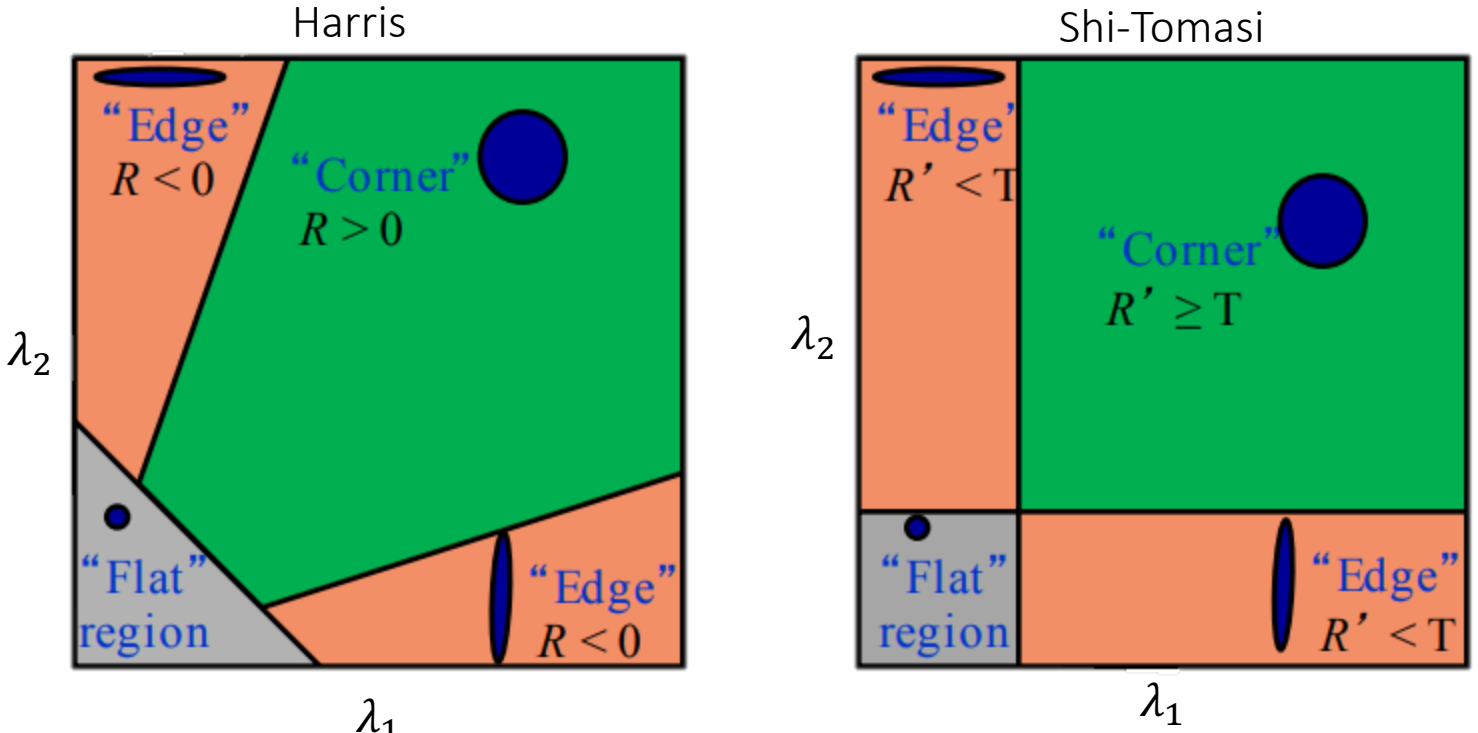


Harris Detector Example

Harris Detector - Responses

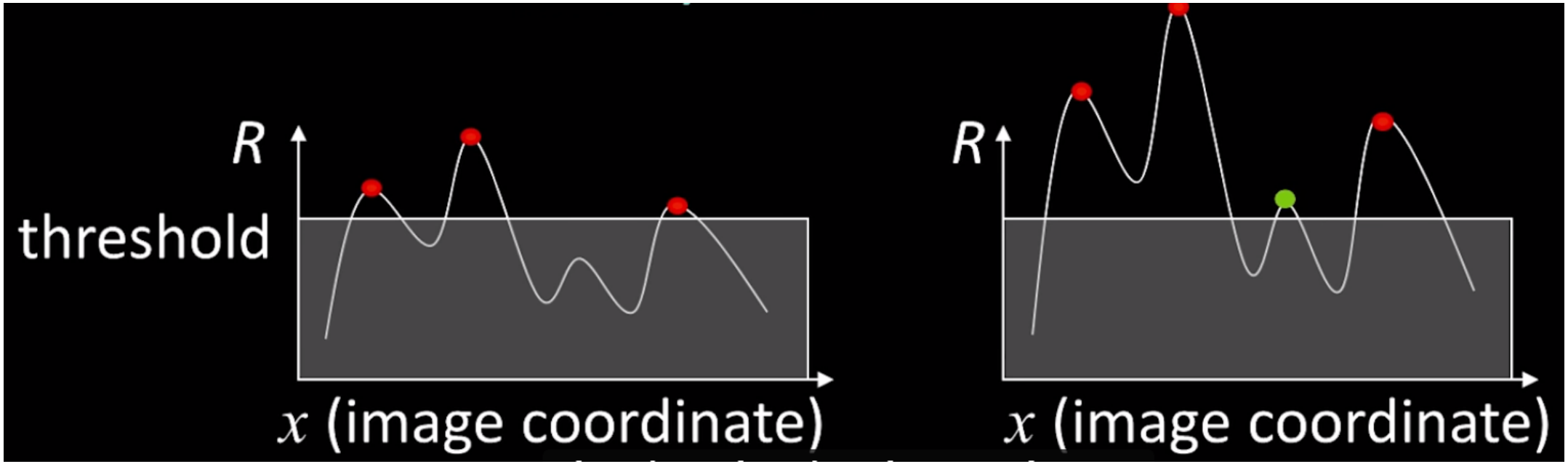
Shi-Tomasi Corner Detector

- Similar to Harris corner points
- Harris: $R = \det(M) - \alpha\, trace(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$
- Shi-Tomasi (KLT good feature): R'=min(λ_1, λ_2)



Harris and Shi-Tomasi Detector : Properties

- Intensity invariance?



Harris and Shi-Tomasi Detector : properties

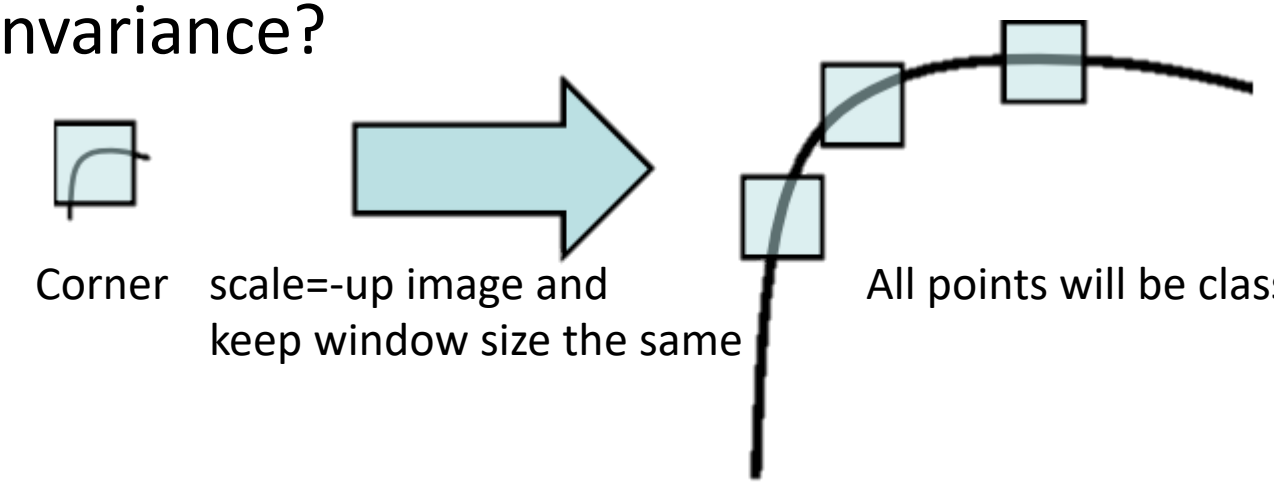
- Intensity invariance
- Rotation invariance?

Eigenvalues remain the same

Corner response R is invariant to image rotation

Harris and Shi-Tomasi Detector : properties

- Intensity invariance
- Rotation invariance
- Scale invariance?



Corner

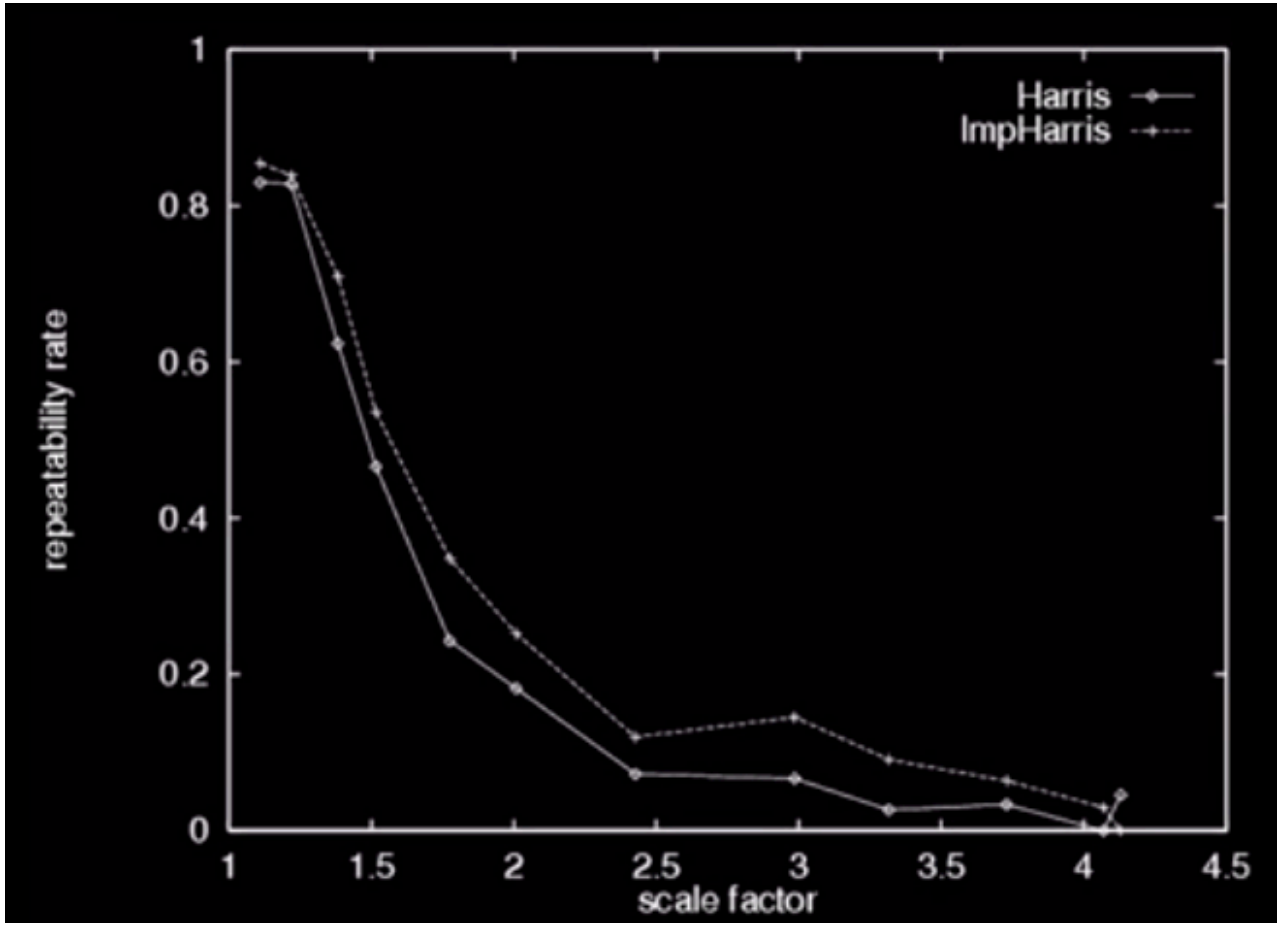
scale=-up image and keep window size the same

All points will be classified as edges!

Not invariant to image scale!

Harris and Shi-Tomasi Detector : Properties

- Repeatability rate v.s. scale factor



Key Takeaways

- Harris corner detection is similar to Shi-Tomasi good feature detector
  - Theoretically, use eigen values to detect the corners (a little bit different equation)
- They are intensity invariant, rotation invariant, but NOT scale invariant

Outline

- Local Invariant Features
  - Motivation
  - Requirements, Invariances
- Keypoint Localization
  - Feature from Accelerated Segment Test (FAST)
  - Harris
  - Shi-Tomasi
- **Scale Invariant Region Selection**
  - **Automatic scale selection**
  - **Laplacian-of-Gaussian detector**
  - **Difference-of-Gaussians detector**
- Local Descriptors
  - Orientation normalization
  - SIFT

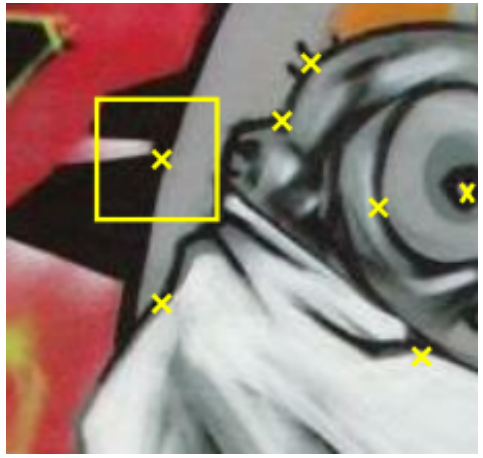
From Points to Regions

- Harris and Shi-Tomasi operators define interest points

- How can we detect scale invariant interest region?
  - In order to compare points, we need to compute a descriptor over a region
  - How can we define such a region in a scale invariant manner?

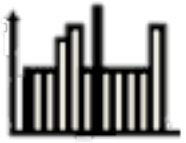
Naïve Approach: Exhaustive Search

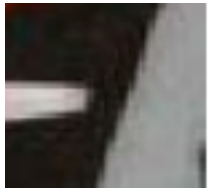
- Multi-scale procedure
  - Compare descriptors while varying the patch size in the other image



$f_A \quad similarity\ measure \quad f_B$

$\neq$

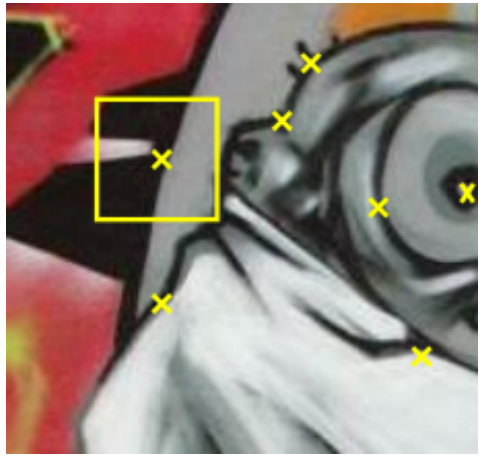




$d(f_A, f_B)$

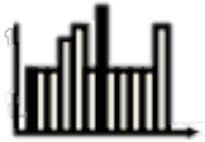
Naïve Approach: Exhaustive Search

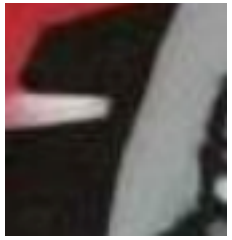
- Multi-scale procedure
  - Compare descriptors while varying the patch size in the other image



$f_A \quad similarity\ measure \quad f_B$

$\neq$

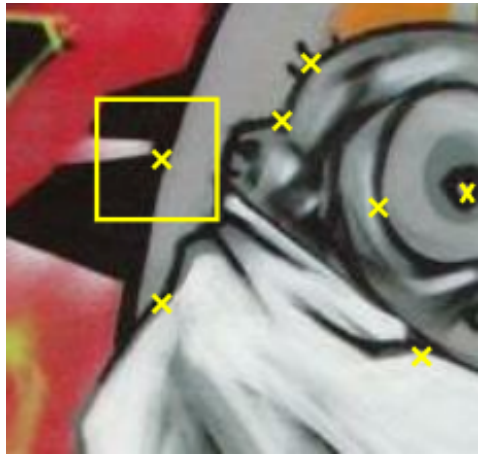




$d(f_A, f_B)$

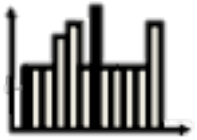
Naïve Approach: Exhaustive Search

- Multi-scale procedure
  - Compare descriptors while varying the patch size in the other image



$f_A \quad similarity\ measure \quad f_B$

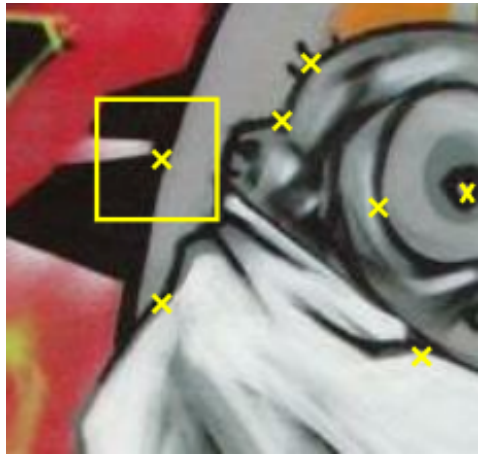
$\neq$



$d(f_A, f_B)$

Naïve Approach: Exhaustive Search

- Multi-scale procedure
  - Compare descriptors while varying the patch size in the other image



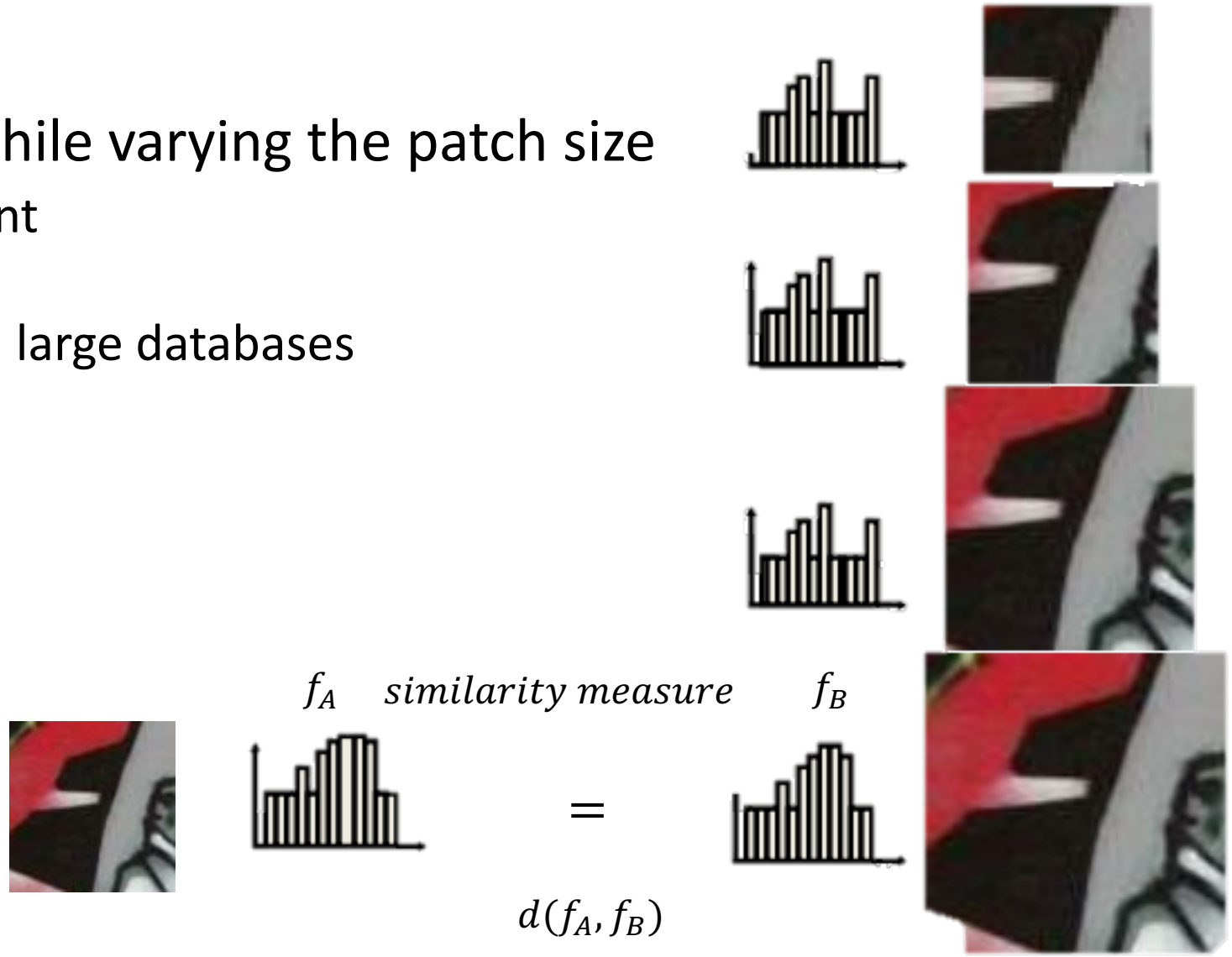
f_A $similarity\ measure$ f_B

$=$

$d(f_A, f_B)$

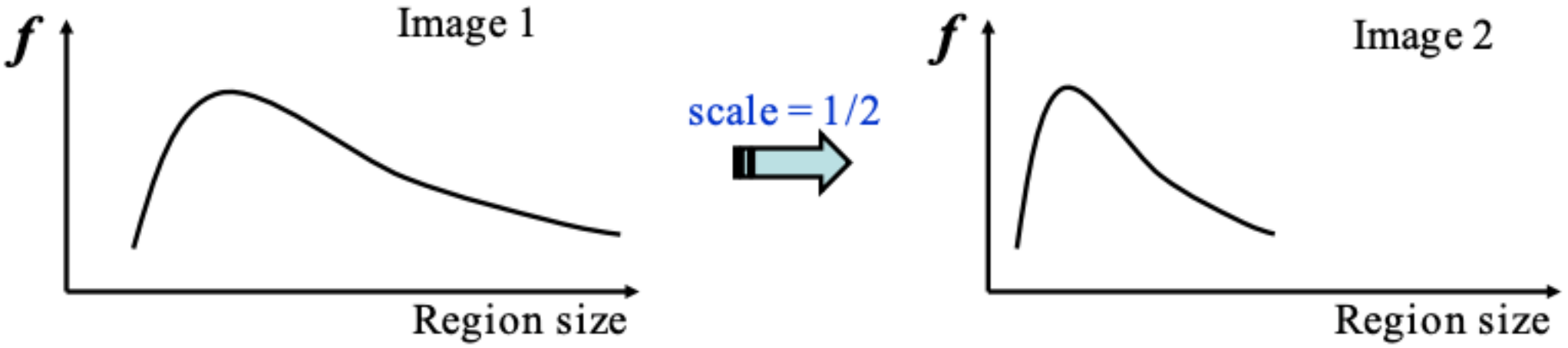
Naïve Approach: Exhaustive Search

- Comparing descriptors while varying the patch size
  - Computationally inefficient
    - But possible for matching
  - Prohibitive for retrieval in large databases



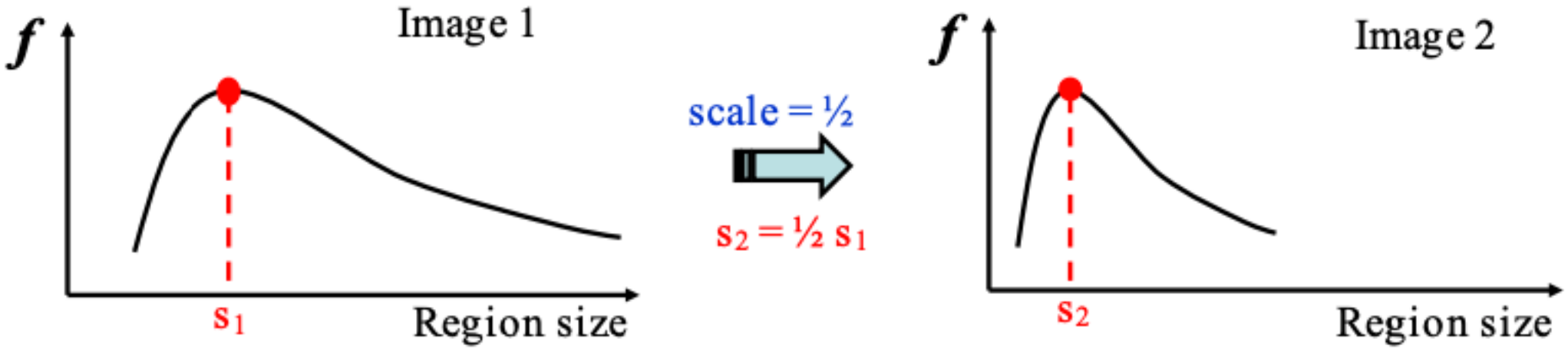
Automatic Scale Selection

- Solution:
  - Design a “magic” function f that is maximal at the correct region size for a given location (indicates the best region size)



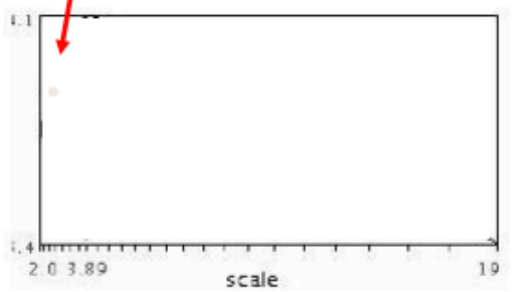
Automatic Scale Selection

- Choose the region size that is best
  - Important: this scale invariant region size (for a point) is found in each image independently!

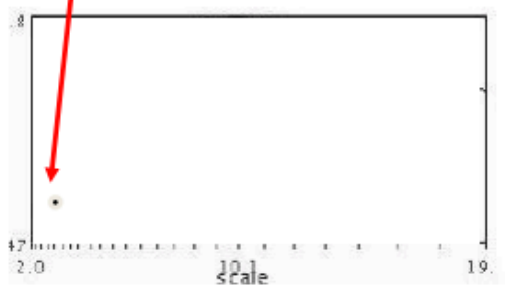


Automatic Scale Selection

- Function responses for increasing scale (scale signature)



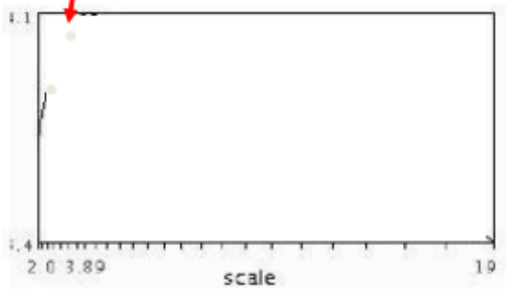
$f(I_{i_1 \ldots i_m}(x, \sigma))$



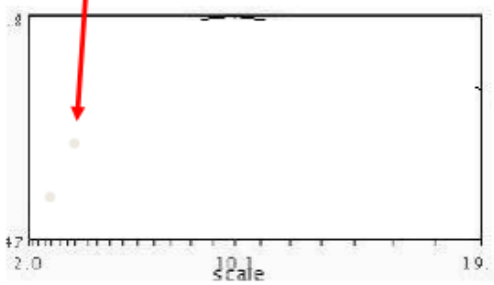
$f(I_{i_1 \ldots i_m}(x', \sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



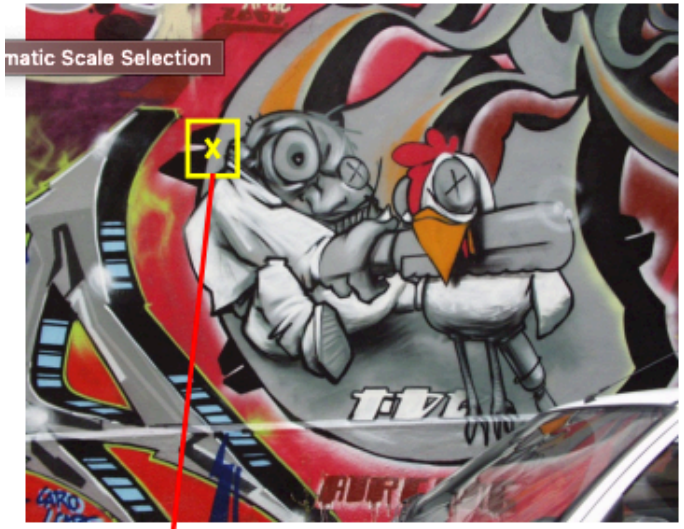
$f(I_{i_1 \ldots i_m}(x,\sigma))$

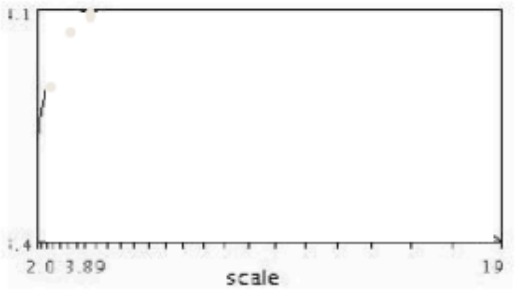


$f(I_{i_1 \ldots i_m}(x',\sigma))$

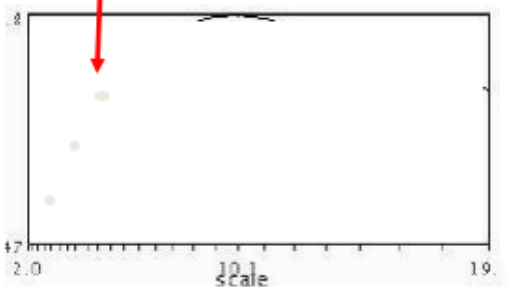
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





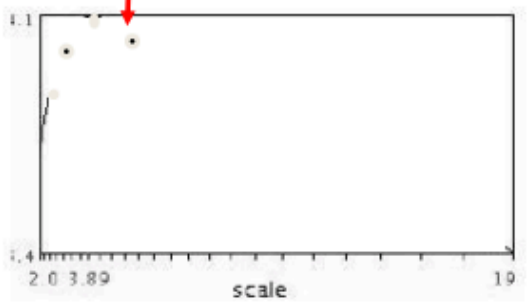
$f(I_{i_1 \ldots i_m}(x,\sigma))$



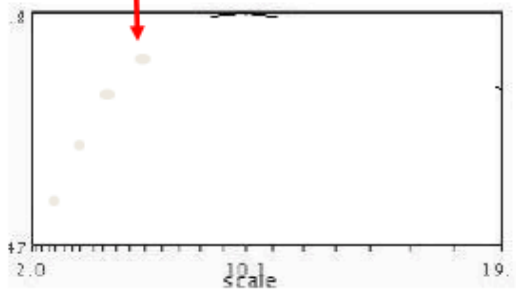
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



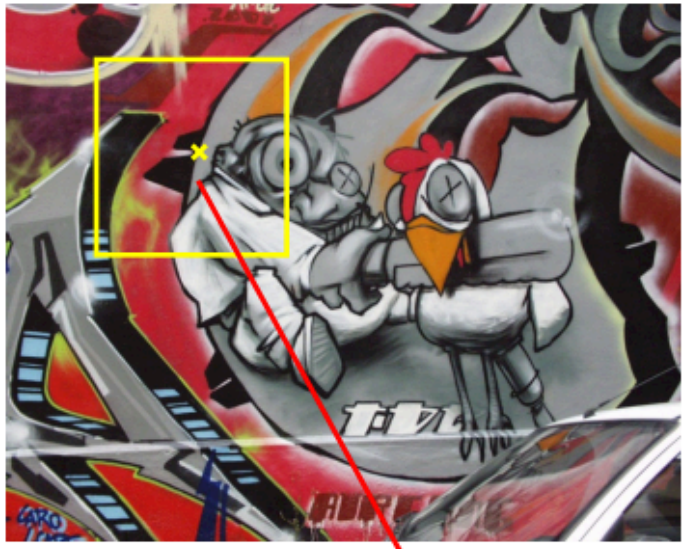
$f(I_{i_1 \ldots i_m}(x,\sigma))$

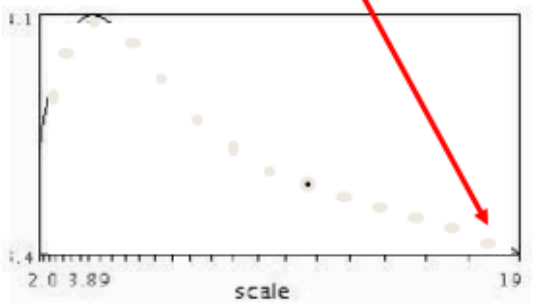


$f(I_{i_1 \ldots i_m}(x',\sigma))$

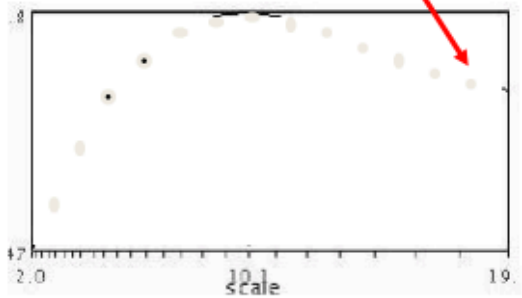
Automatic Scale Selection

- Function responses for increasing scale (scale signature)





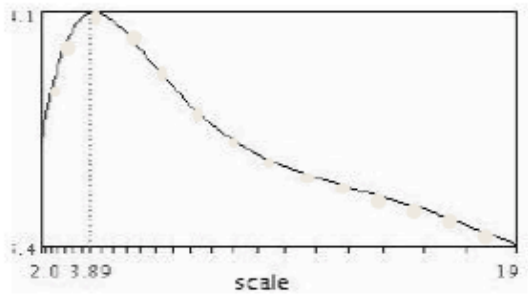
$f(I_{i_1 \ldots i_m}(x,\sigma))$



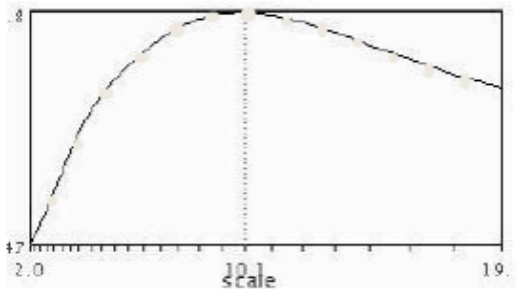
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



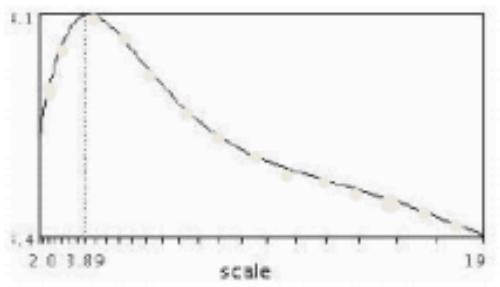
$f(I_{i_1 \ldots i_m}(x,\sigma))$



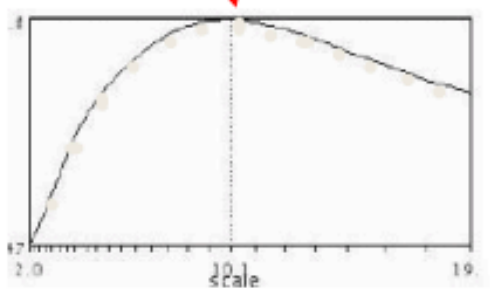
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Automatic Scale Selection

- Normalize: Rescale best region size to fixed size



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



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

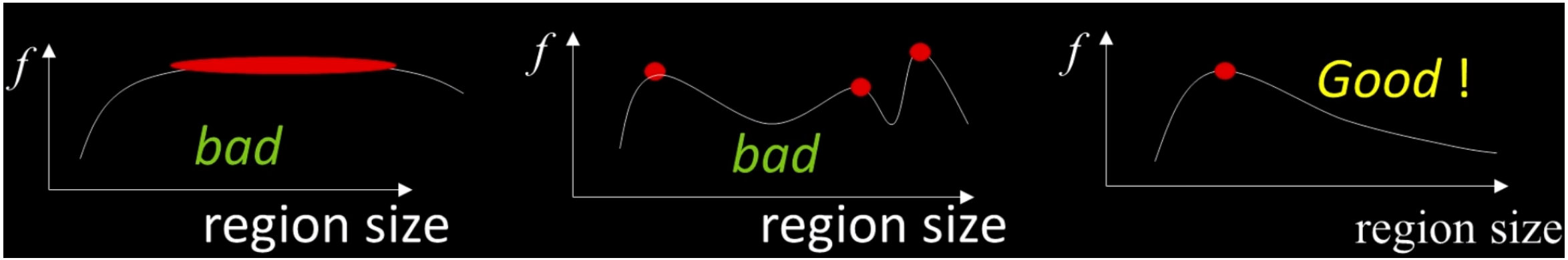
Image1

Image2

What could be a possible magic function for f ?

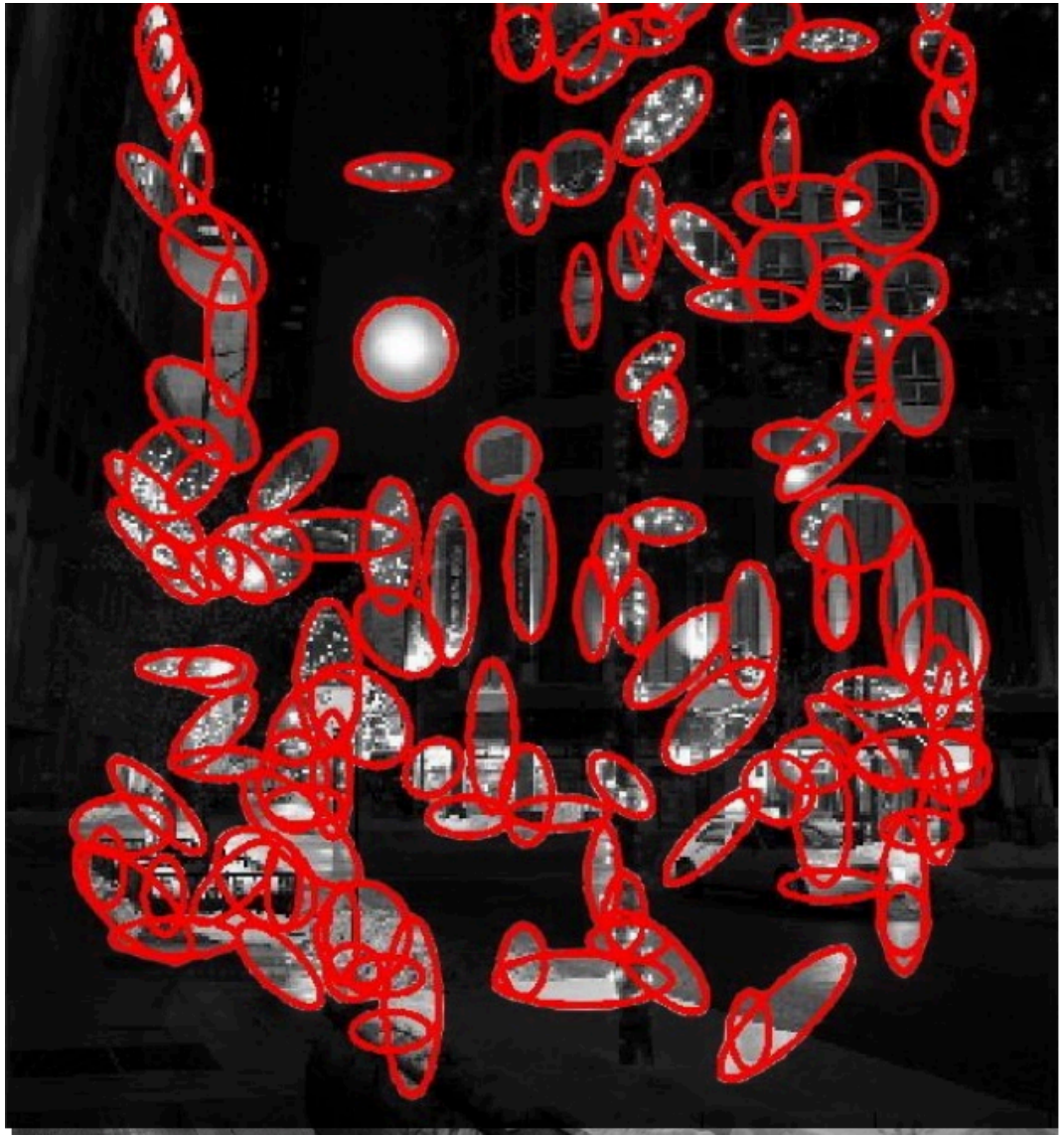
Scale Invariant Detection

- 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)

Blob Detector



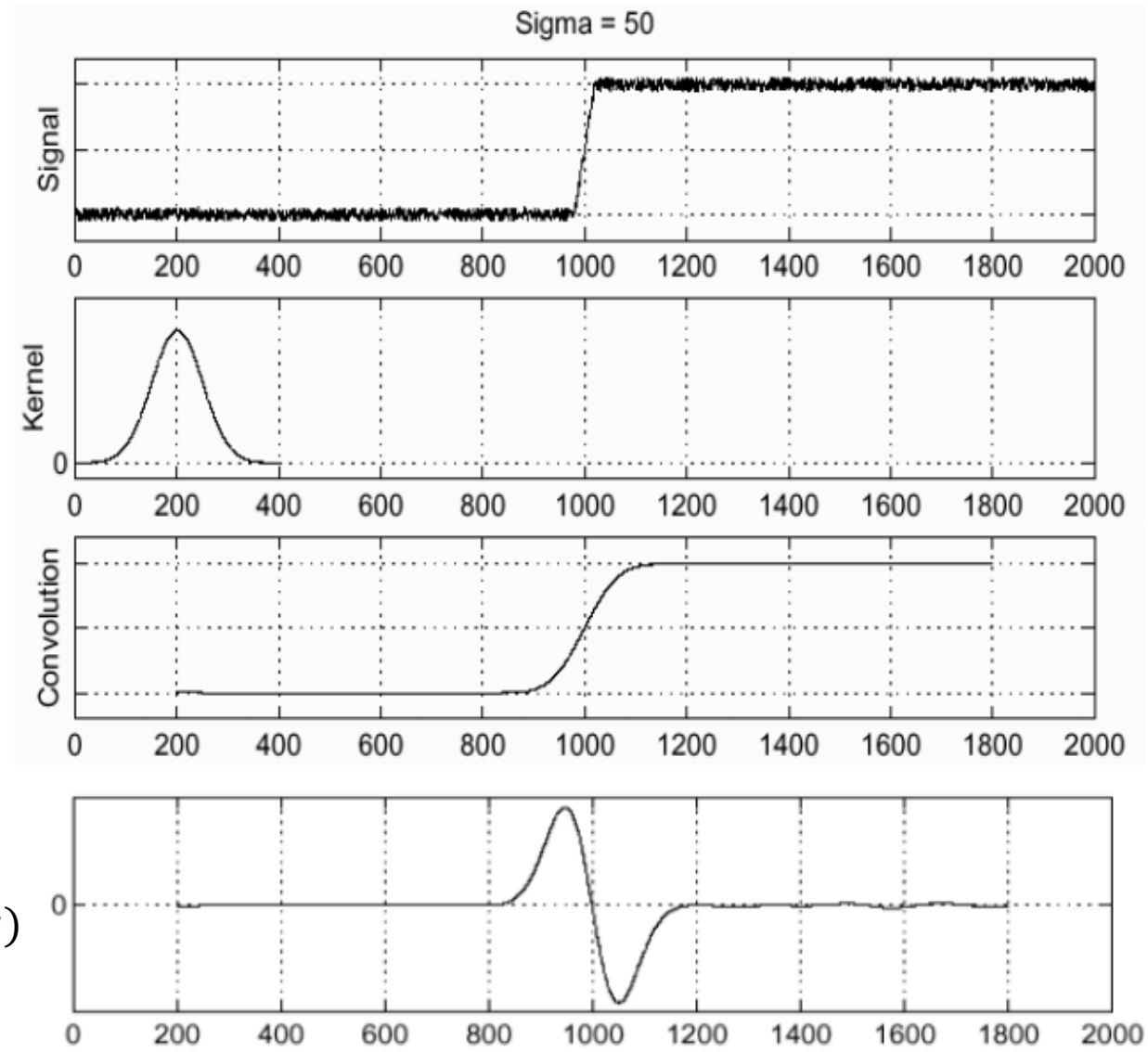
Edge Detector

f

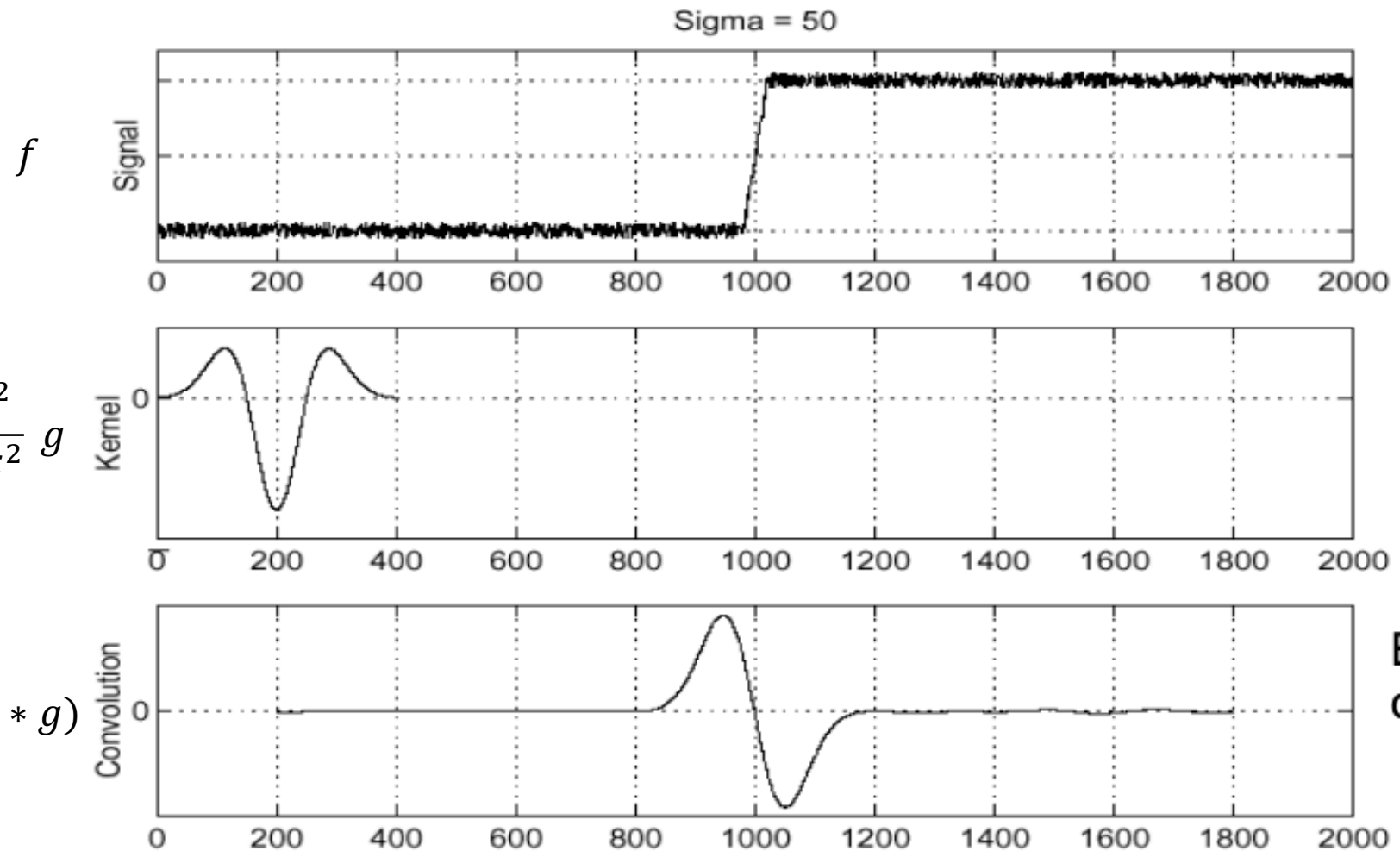
g

$f * g$

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



Edge Detector as Zero Crossing



f

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

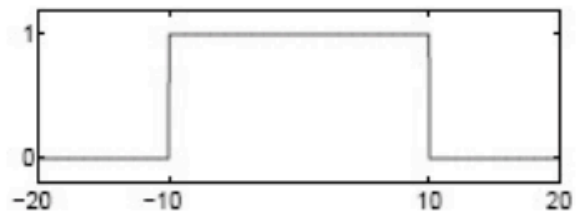
$\frac{d^2}{dx^2}\ (f * g)$

Edge

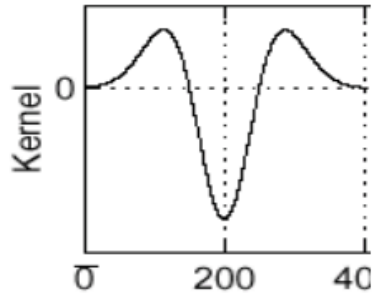
Laplacian

Edge = zero crossing of second derivative

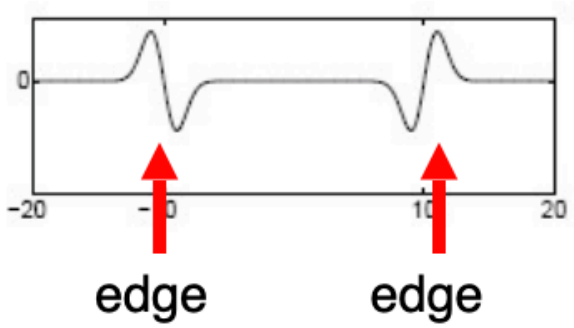
Edge Detector as Zero Crossing



*

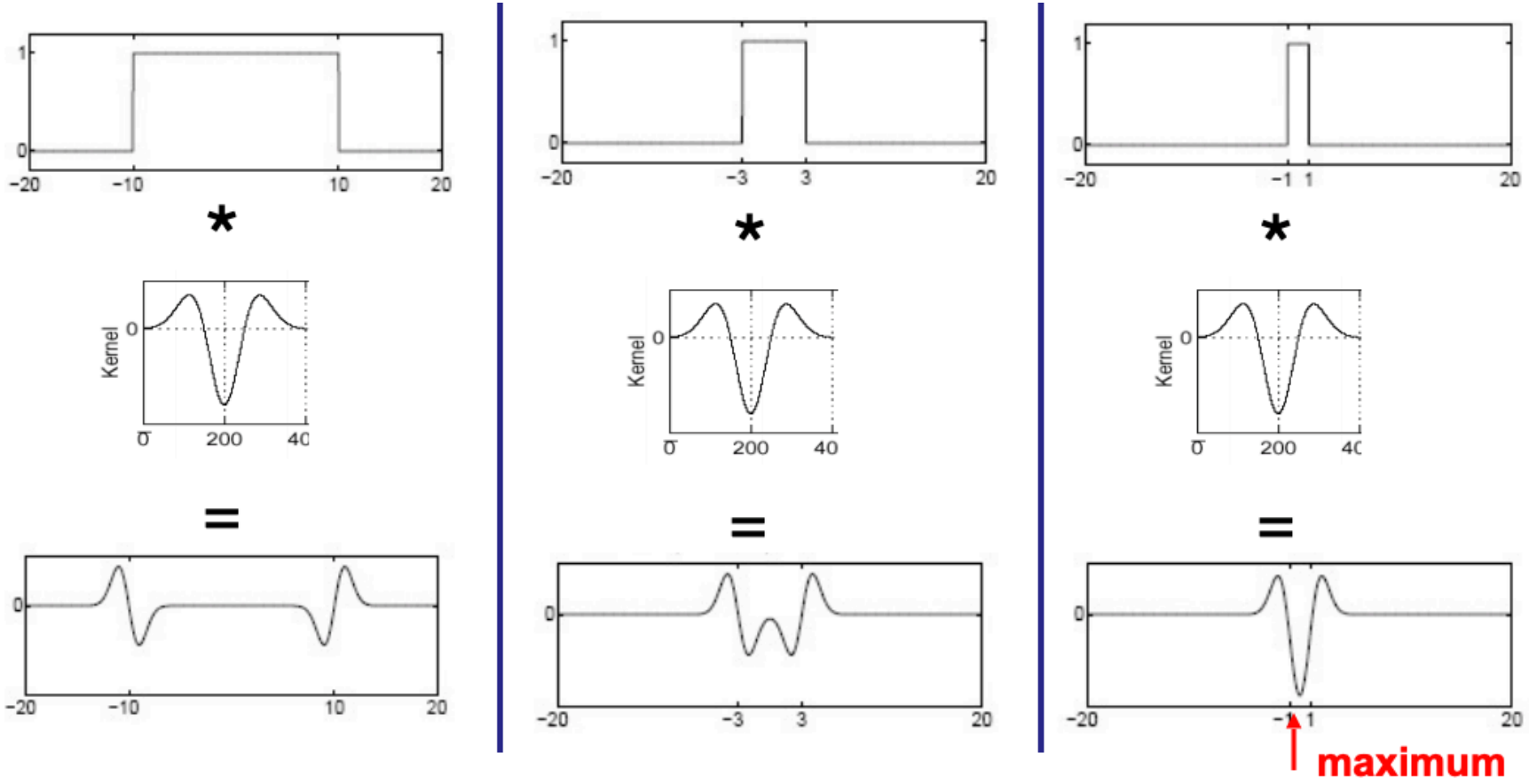


=



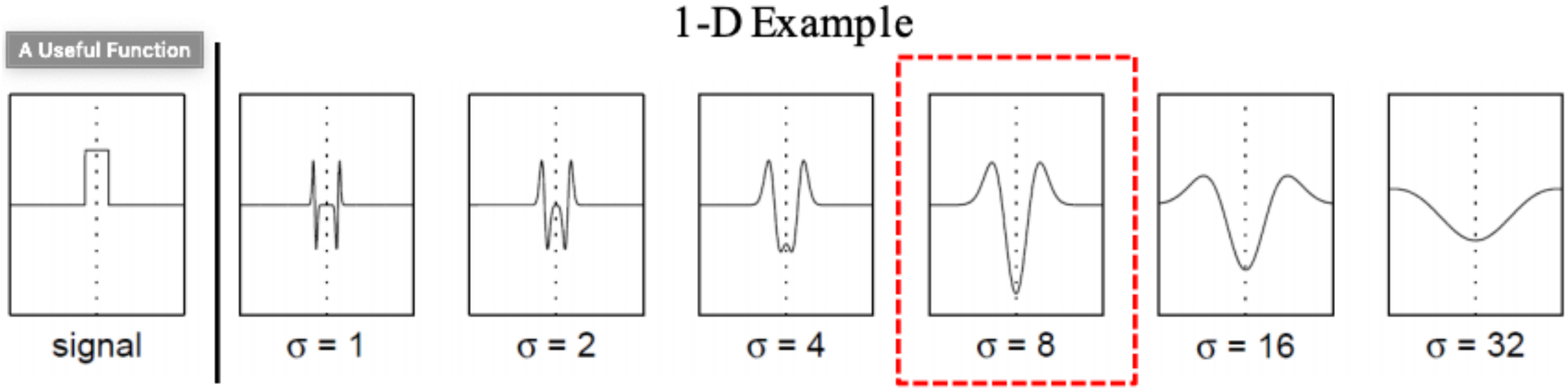
From Edge to Blob

- Can we use the Laplacian to find a blob (rect function in 1D)



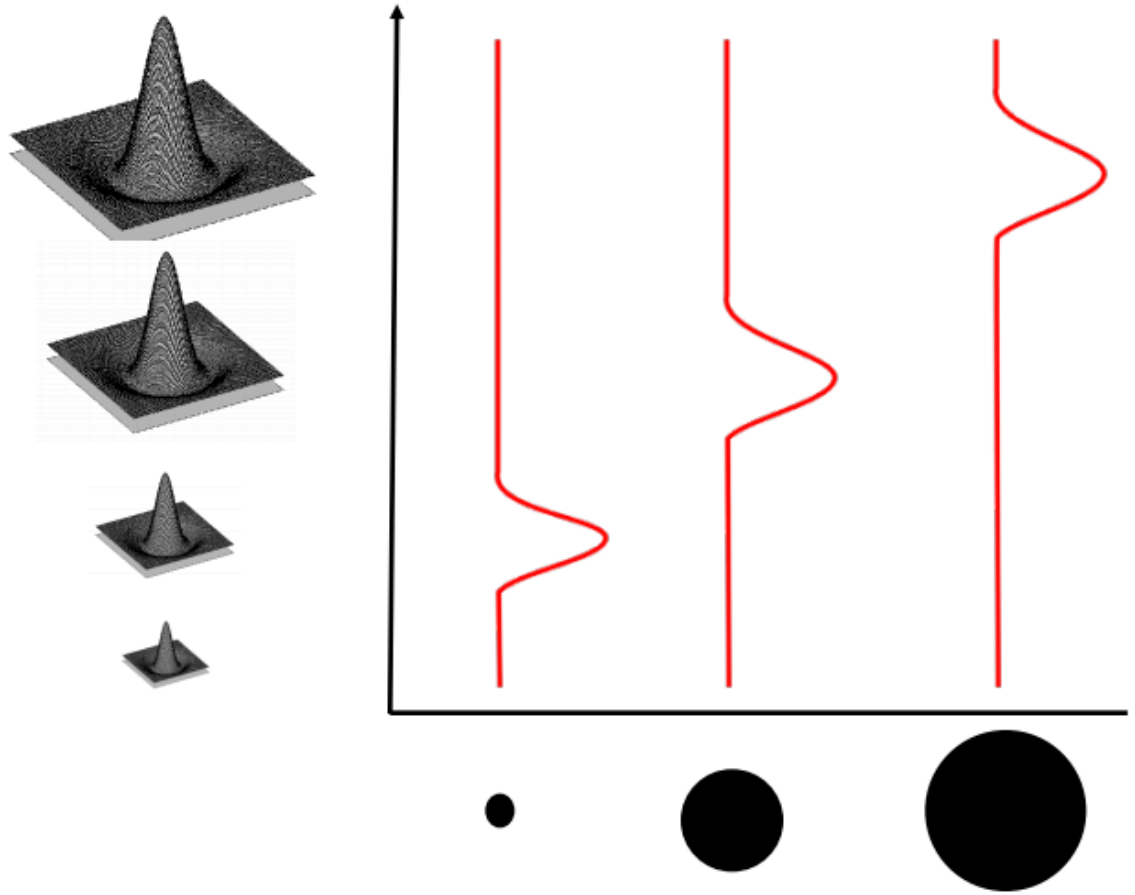
Scale-Normalized Laplacian-of-Gaussian

- $L = \sigma^2(G_{xx}(x, y; \sigma) + G_{yy}(x, y; \sigma))$
  - Scale-normalized Laplacian shows the strongest response at the characteristic scale of the original signal ($\sigma = 8, x = 0$)



A Useful Function

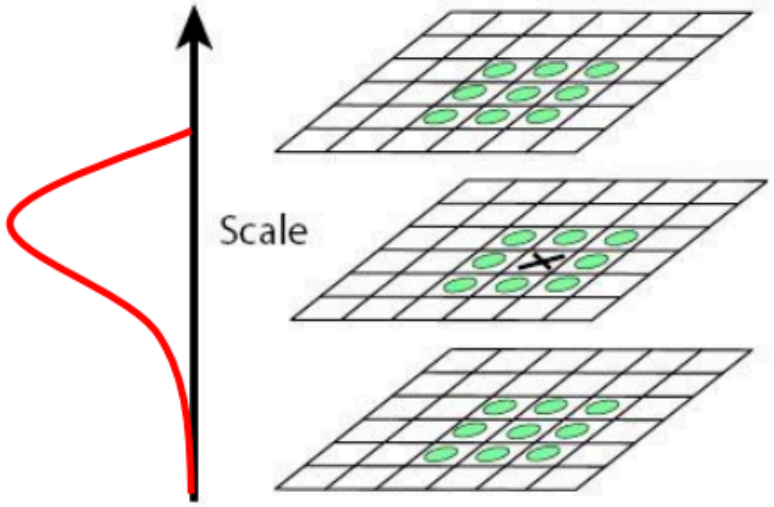
- “Scale-normalized” Laplacian-of-Gaussian = blob/spot detector



Scale Blob Detector

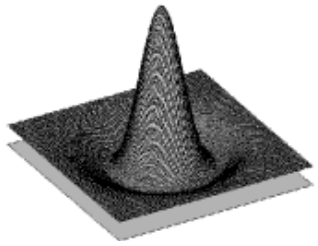
- Convolve image with scale-normalized Laplacian at several scales
- Find maximum of squared Laplacian response in scale-space

The maxima indicate that a blob has been detected and what's its intrinsic scale



Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space (above/below/around) of scale-normalized Laplacian-of-Gaussian



$\sigma^2(G_{xx}(\sigma) + G_{yy}(\sigma))$

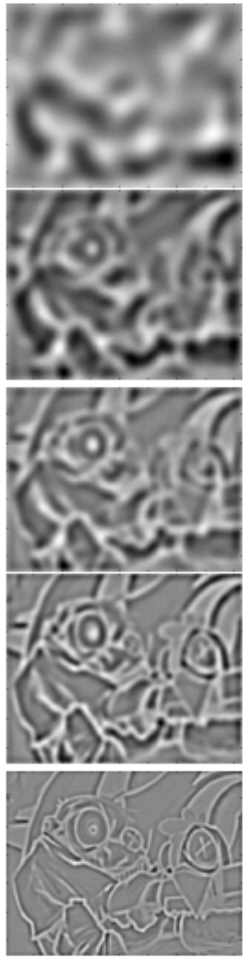
$\sigma = \sigma_0^5$

$\sigma = \sigma_0^4$

$\sigma = \sigma_0^3$

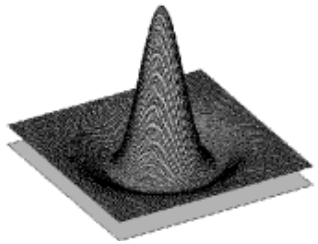
$\sigma = \sigma_0^2$

$\sigma = \sigma_0$



Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space (above/below/around) of scale-normalized Laplacian-of-Gaussian



$\sigma^2(G_{xx}(\sigma) + G_{yy}(\sigma))$

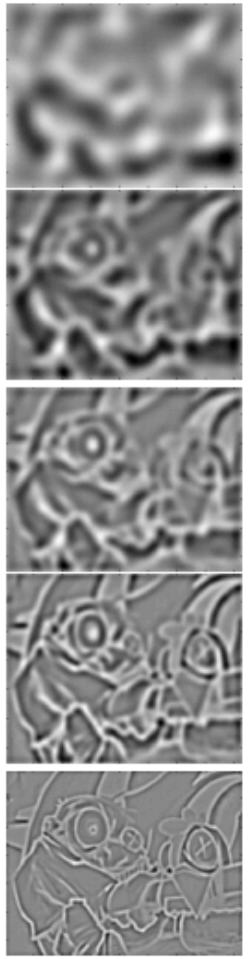
$\sigma = \sigma_0^5$

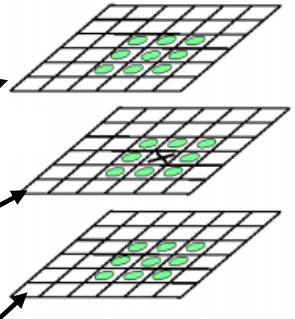
$\sigma = \sigma_0^4$

$\sigma = \sigma_0^3$

$\sigma = \sigma_0^2$

$\sigma = \sigma_0$

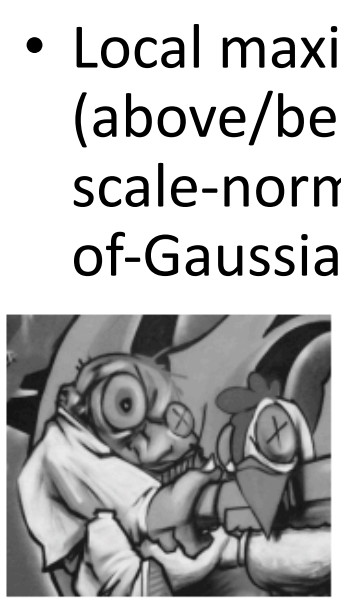


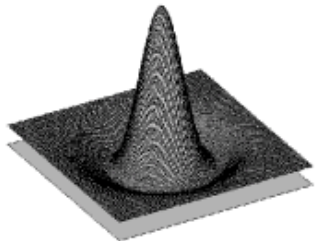


Scale

Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space (above/below/around) of scale-normalized Laplacian-of-Gaussian





$\sigma^2(G_{xx}(\sigma) + G_{yy}(\sigma))$

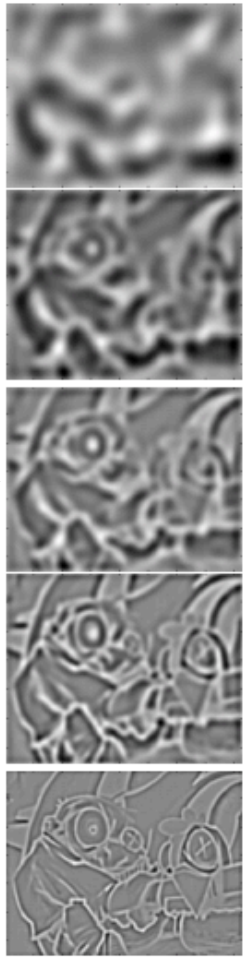
$\sigma = \sigma_0^5$

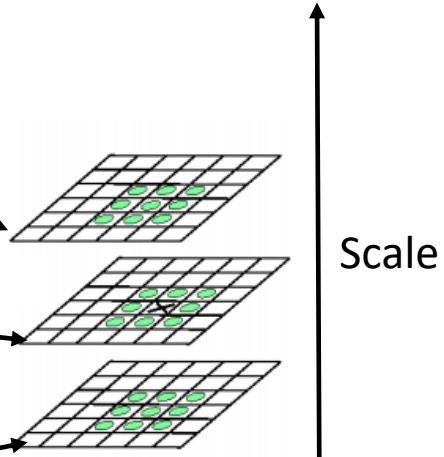
$\sigma = \sigma_0^4$

$\sigma = \sigma_0^3$

$\sigma = \sigma_0^2$

$\sigma = \sigma_0$

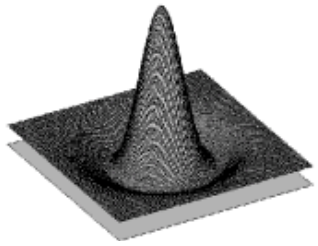




Scale

Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space (above/below/around) of scale-normalized Laplacian-of-Gaussian



$\sigma^2(G_{xx}(\sigma) + G_{yy}(\sigma))$

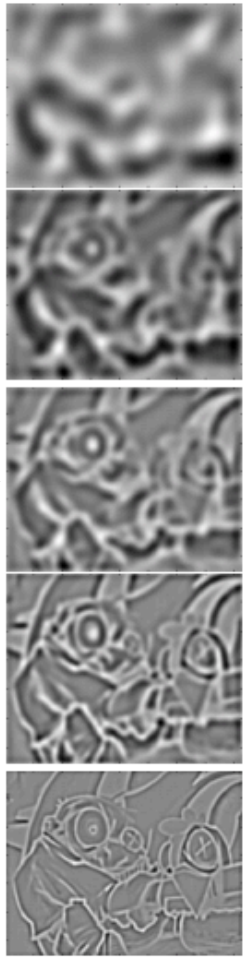
$\sigma = \sigma_0^5$

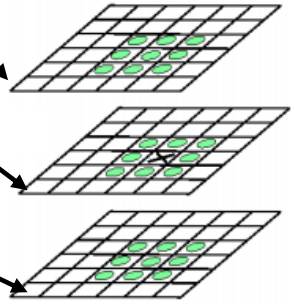
$\sigma = \sigma_0^4$

$\sigma = \sigma_0^3$

$\sigma = \sigma_0^2$

$\sigma = \sigma_0$



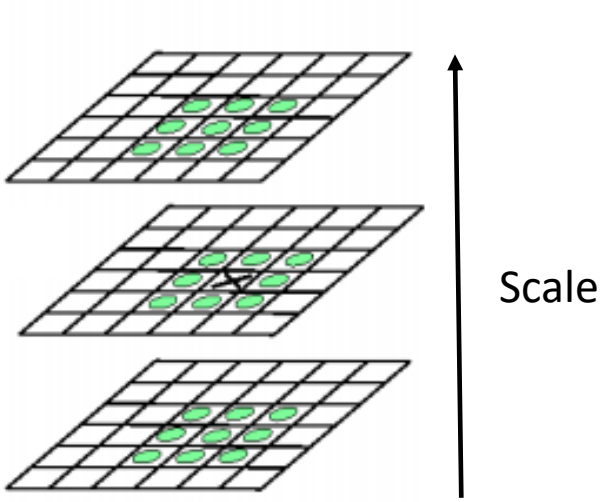


Scale

Save list of (x, y, σ)that are locally maximal

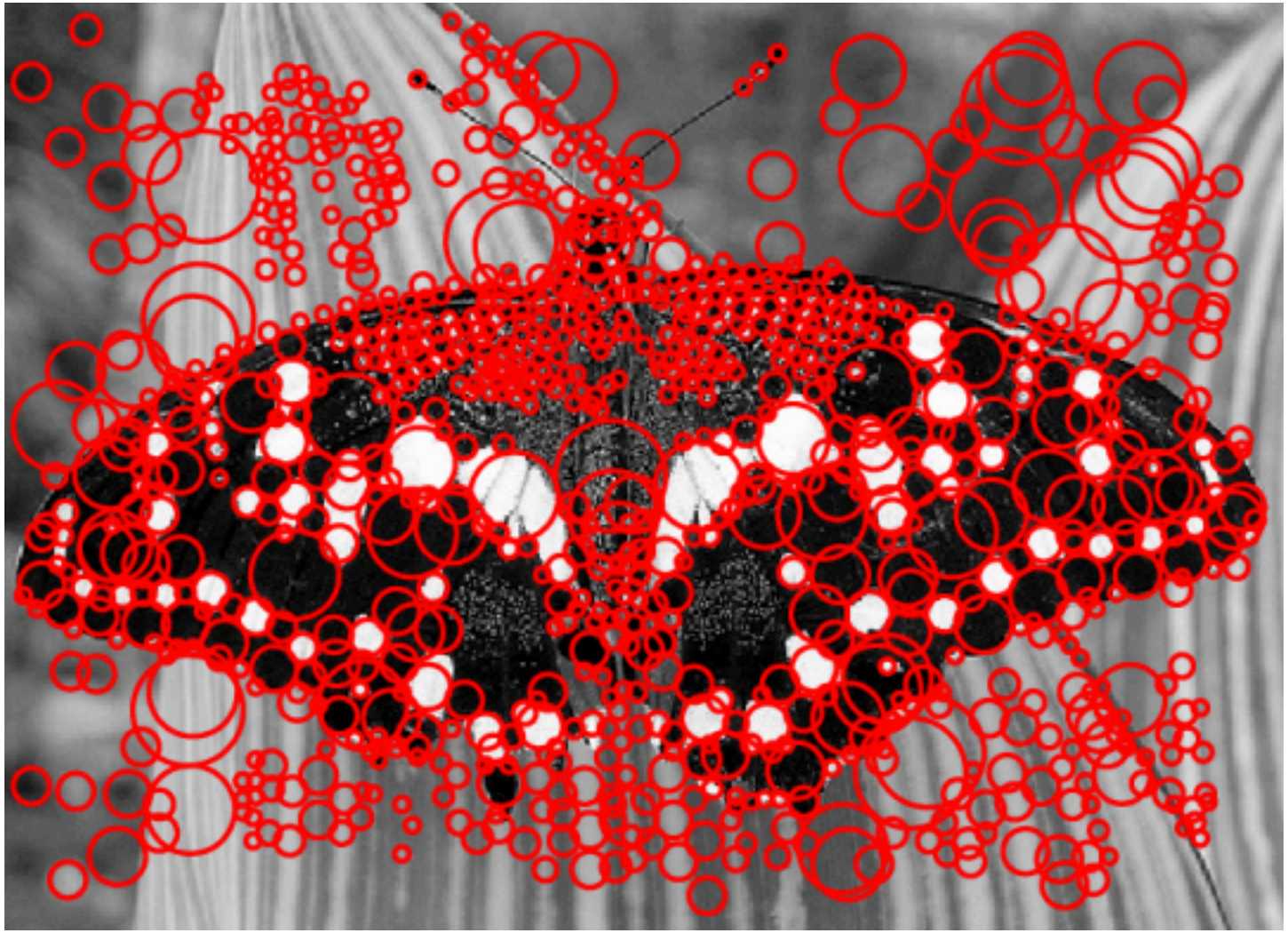
Keypoint Localization

- Detect maxima in scale space
- Then reject points with low image contrast (threshold)
- Eliminate edge responses



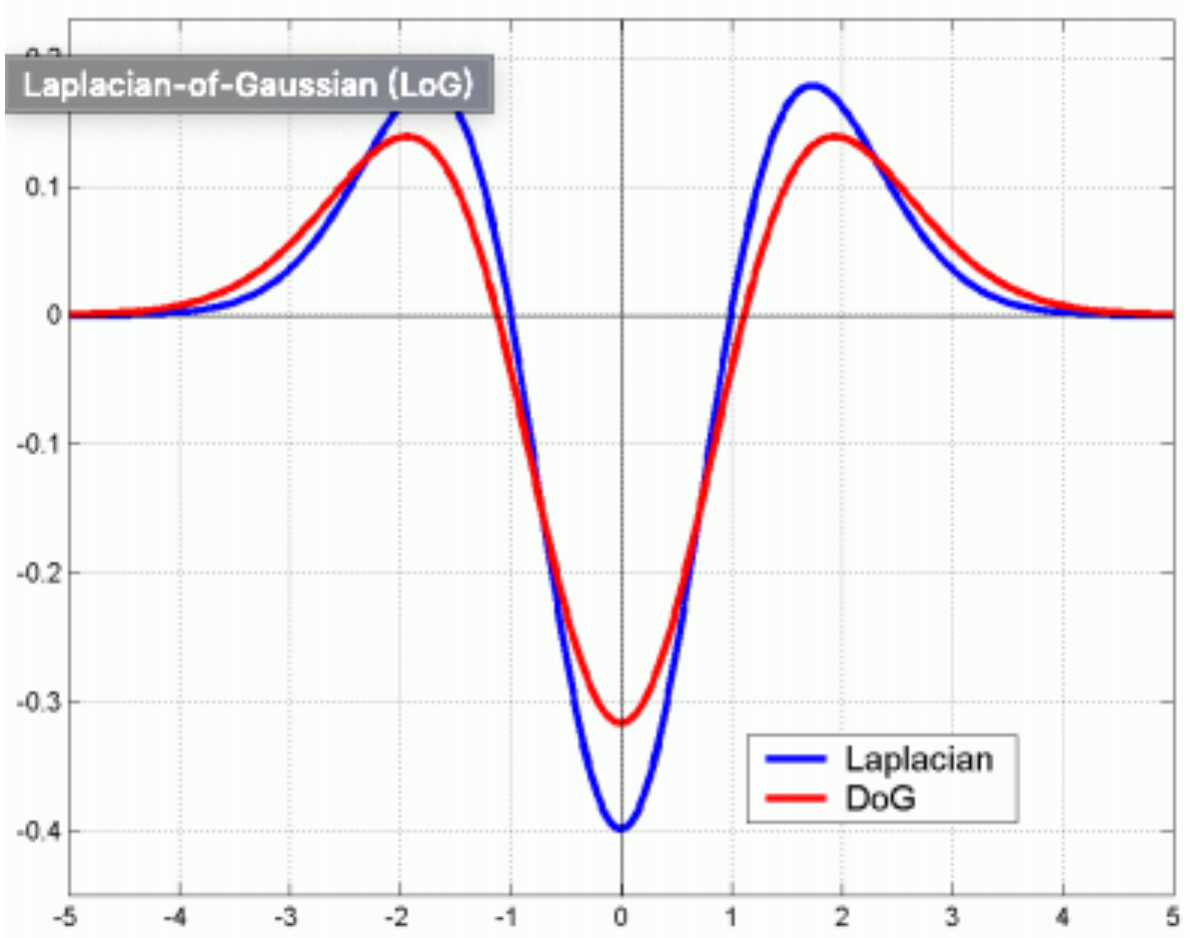
Candidate keypoints: list of (x, y, σ)

LoG Detector



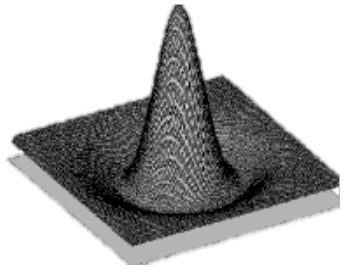
Technical Detail

- We can efficiently approximate the scale-normalized Laplacian with a "difference of Gaussian" (DoG)
  - $L = \sigma^2(G_{xx}(x, y; \sigma) + G_{yy}(x, y; \sigma))$
  - $DoG = G(x, y; k\sigma) - G(x, y; \sigma)$
- Since it is an approximation, DoG is generally slightly inferior to LoG



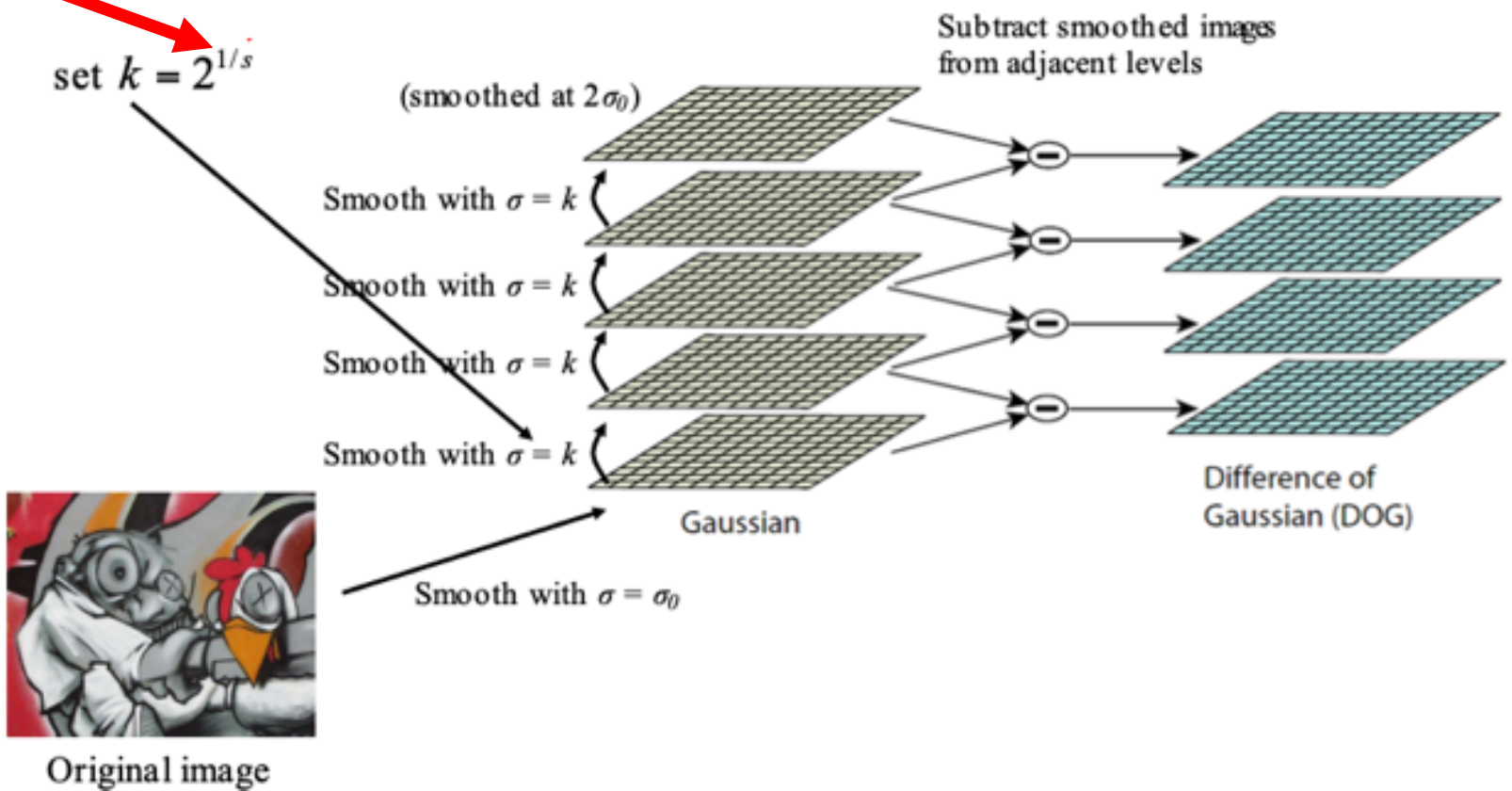
Difference-of-Gaussians (DoG)

- Difference-of-Gaussians as approximation of the scale-normalized LoG
- Advantage
  - No need to compute 2nd derivatives

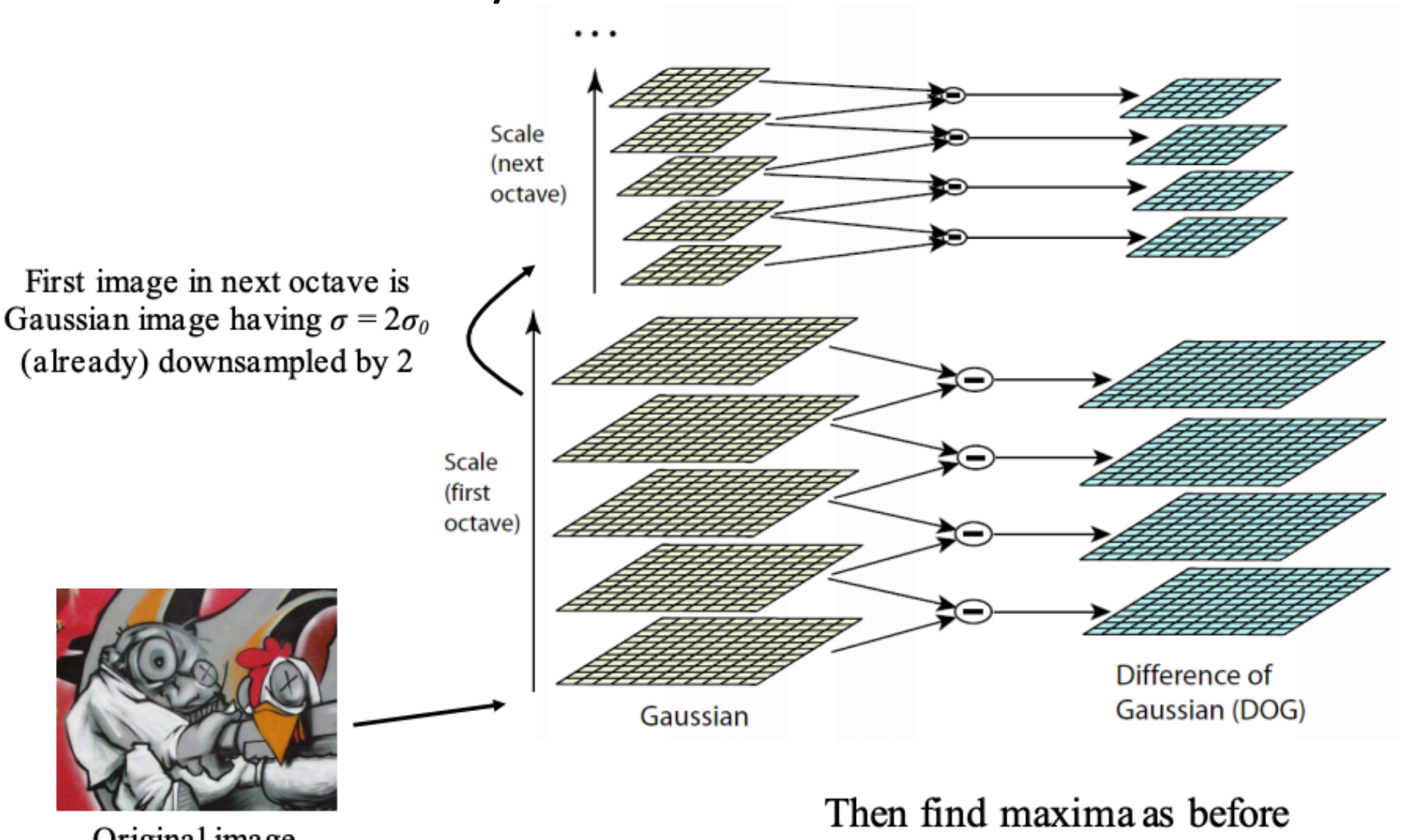


DoG “Octave”

- Octave=“set of images for which smoothing σ is doubled”
- s=desired # of images in DoG “octave”



DoG Octave Pyramid



Example of Keypoint Detection

DoG extrema

Points remaining after contrast threshold

Points remaining after removing edge responses

Key Takeaways

- Use a magic function to determine the scale of a feature
  - Laplacian-of-Gaussian (LoG)
- Implementation: use Difference of Gaussian (DoG) to create octave pyramid (approximation)
  - Find the local extreme in scale and space neighbors
- Remove low contrast points and edge points

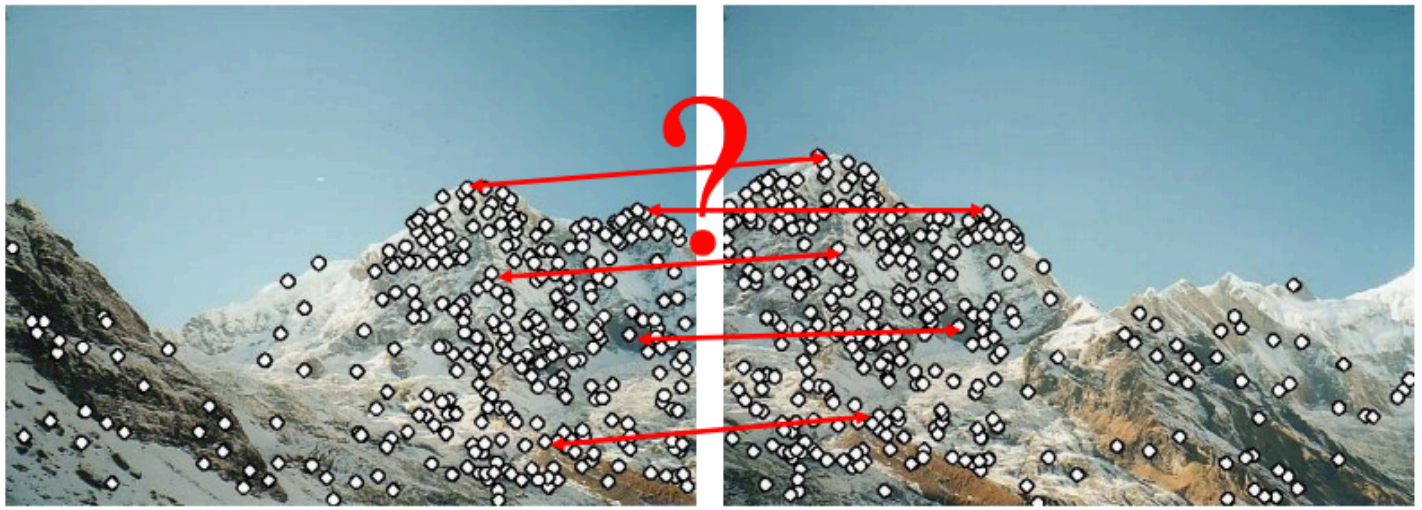
Outline

- Local Invariant Features
  - Motivation
  - Requirements, Invariances
- Keypoint Localization
  - Feature from Accelerated Segment Test (FAST)
  - Harris
  - Shi-Tomasi
- Scale Invariant Region Selection
  - Automatic scale selection
  - Laplacian-of-Gaussian detector
  - Difference-of-Gaussians detector
- **Local Descriptors**
  - **Orientation normalization**
  - **SIFT**

Local Descriptors

- We know how to detect points and appropriate scales
- Next question:

How to *describe* them for matching?



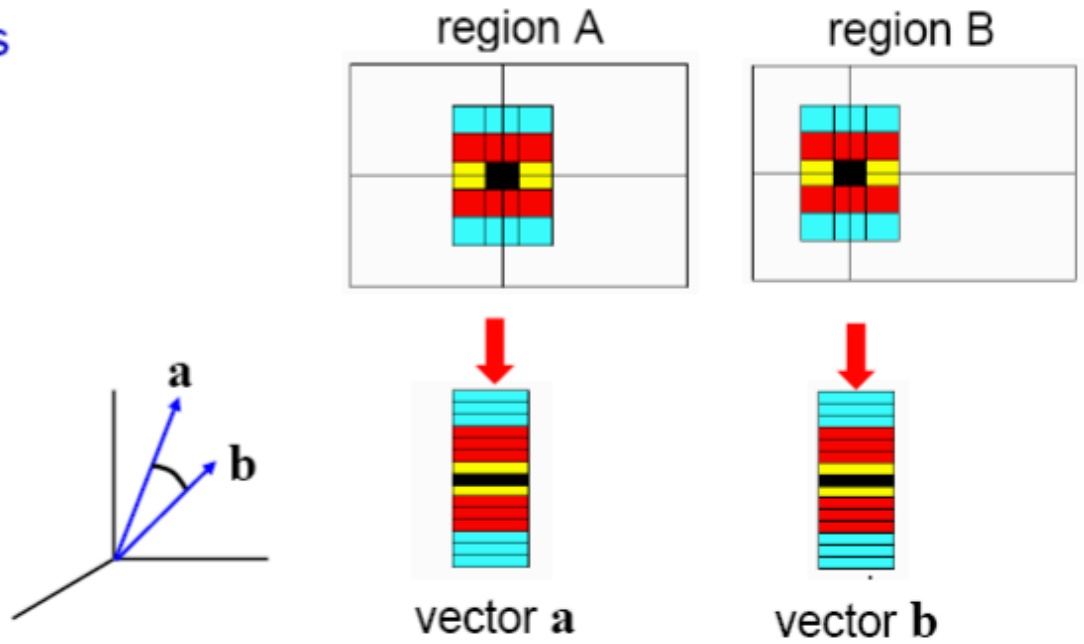
Point descriptor should be:
1. Invariant
2. Distinctive

Local Descriptors

- Simplest descriptor: rasterized list of intensities within a patch
- To what is this going to be invariant (for normalized patch sizes)?
  - Translation and scale only (not rotation)

Write regions as vectors

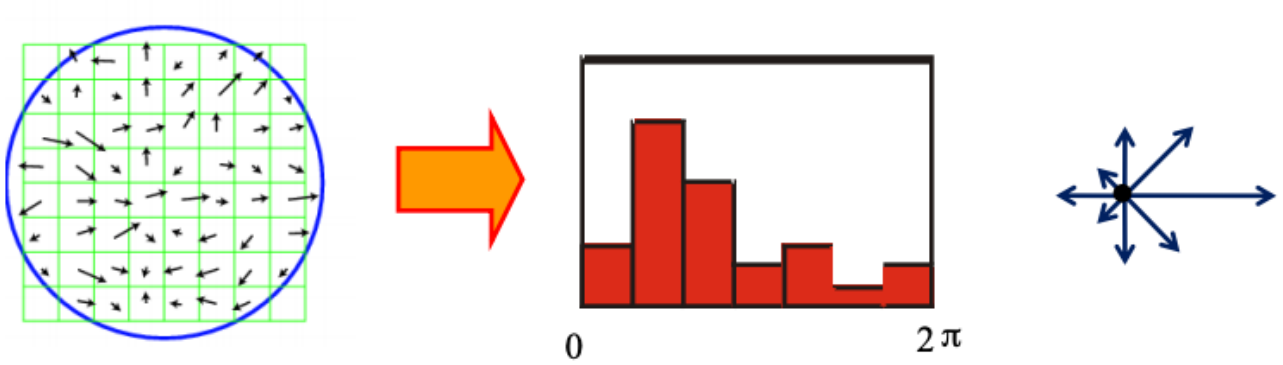
$$A \rightarrow \mathbf{a}, \quad B \rightarrow \mathbf{b}$$



Feature Descriptors

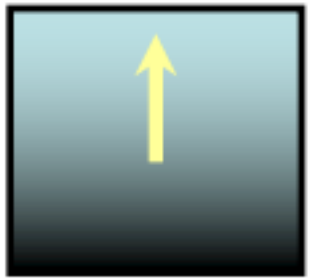
- Disadvantage of patches as descriptors:
  - Small deformations can greatly affect matching score

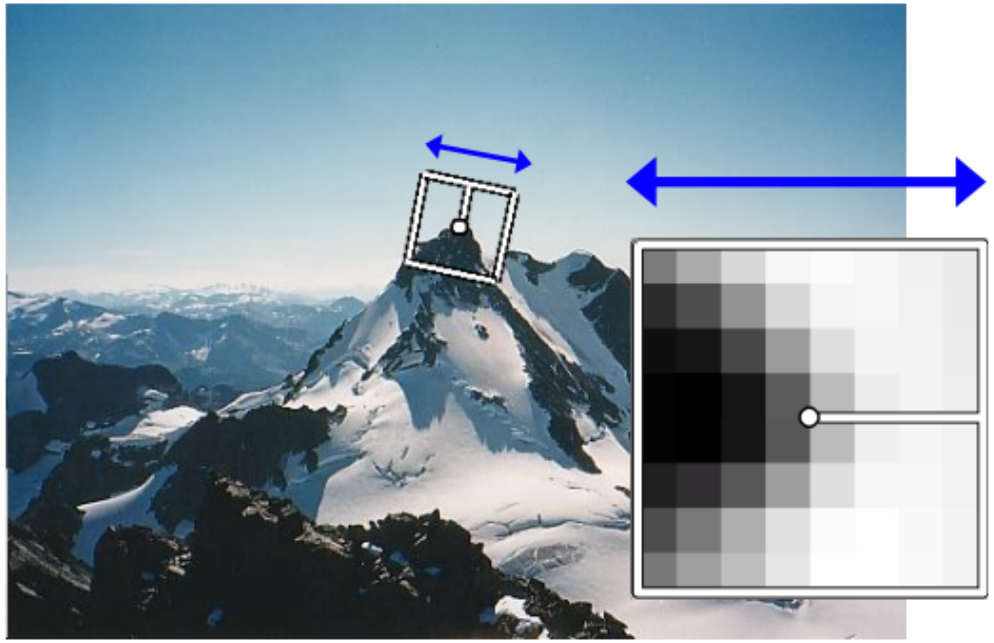
  - Solution: histogram of gradient direction



Rotation Invariant Descriptors

- Find local gradient directions in patch
- Find dominant direction of overall gradient for the patch
- Rotate patch according to this angle
  - This pus the patches into a scandalized direction





Feature Descriptors: SIFT

- **S**cale **I**nvariant **F**eature **T**ransform
- Descriptor computation:
  - Divide patch (for keypoint at chosen scale) after standardized direction alignment into 4x4 sub-patches: 16 cells
  - Compute Gaussian weighted histogram of gradient directions (8 reference angles) for all pixels inside each sub-patch
  - Resulting descriptor: 4x4x8=128 dimensions
  - Rasterize and normalize to unit vector (enhances invariance to affine changes in illumination)
  - Clip high values and renormalize vector to reduce effects of non-linear illumination

Working with SIFT Descriptors

- One image with n feature point yields:
  - n 128-dimensional descriptor: each one is a collection of histograms of the gradients orientations within a patch
    - [n x 128 matrix]
  - n scale parameters specifying the size of each patch
    - [n x 1 matrix]
  - n orientation parameters specifying the angle of the patch
    - [n x 1 matrix]
  - n 2-D points giving positions of the patches
    - [n x 2 matrix]

Matching SIFT Descriptors

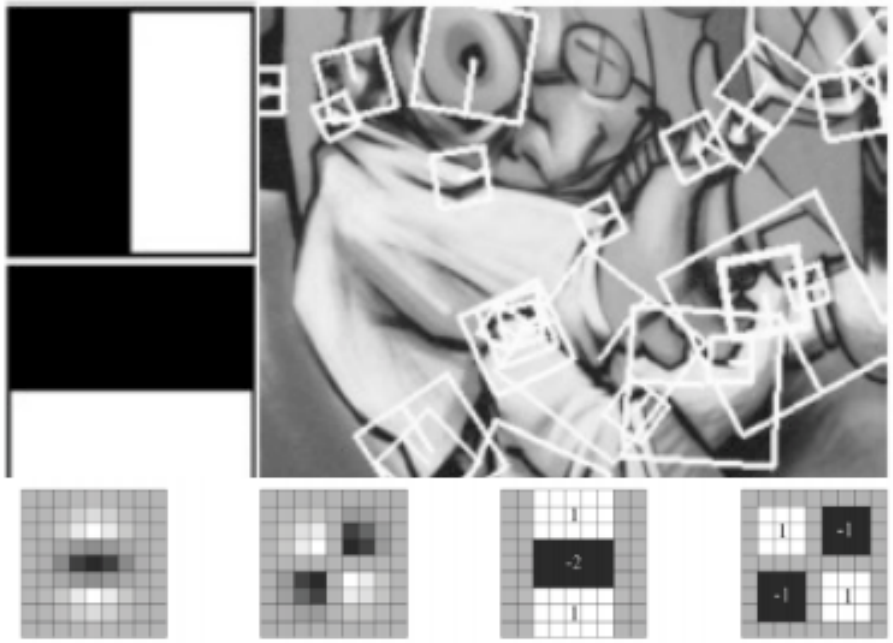
- Given two images
- Match each keypoint independently
  - Employ 128-D descriptors
  - Best candidate match is closest Euclidean neighbor
  - Discard matches using ratio of closest and second closest neighbors
    - Simple measure of how unique is the match
- Can employ more costly approaches to match complete sets of points (e.g., Hungarian algorithm)

Advantages of SIFT

- Fairly robust matching technique
  - Can handle significant change in viewpoint
  - Can handle significant changes in illumination
    - Sometimes even day vs night
  - Fast and efficient – can run in real time
    - Lots of code available

Variations and Implementations

- SURF: Fast approximation of SIFT (multiple times faster)
  - Efficient computation by 2-D box filters and integral images
  - Essentially equivalent quality for object identification
- GPU implementations of SIFT and SURF exist



Key Takeaways

- **SIFT** is a very popular and powerful local descriptor from 15-20 years ago
  - Translation, rotation and scale invariant
  - Handle significant illumination and viewpoint change well
- Translation invariant:
- Rotation invariant: define/find the dominant direction of overall gradient for the patch
- Scale: use “Octave” to define the scale of a patch

- A lot of open source code and various SIFT implementations