CS 2770: Computer Vision
Local Feature Detection, Description and Matching

Prof. Adriana Kovashka
University of Pittsburgh
January 22, 2019

Plan for this lecture

- Feature detection / keypoint extraction
  - Corner detection
  - Properties
  - Blob detection
- Feature description (of detected features)
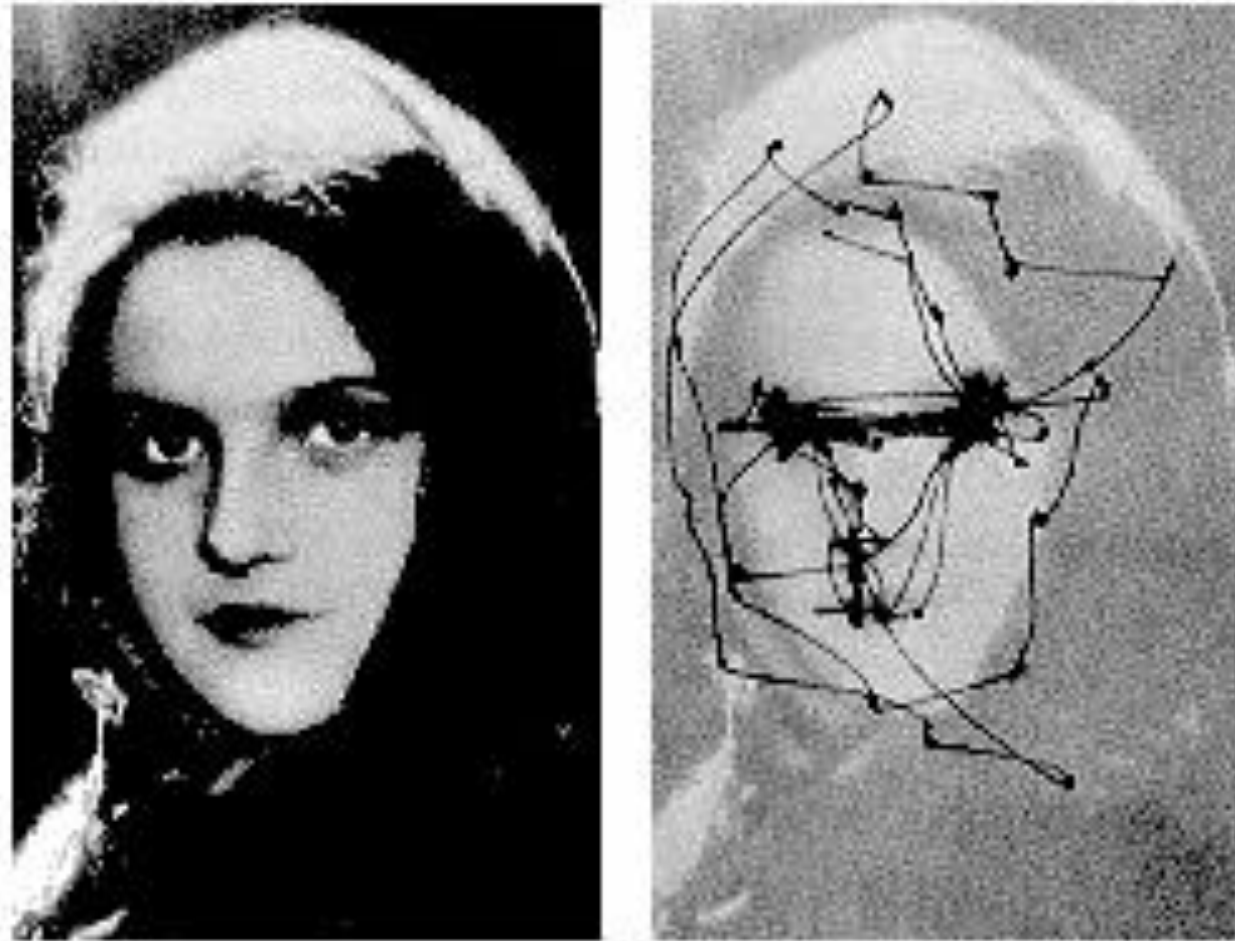- Matching features across images

An image is a set of pixels…

Problems with pixel representation

- Not invariant to small changes
  - Translation
  - Illumination
  - etc.
- Some parts of an image are more important than others
- What do we want to represent?

Human eye movements

Yarbus eye tracking

D. Hoiem

Local features

- *Local* means that they only cover a small part of the image

- There will be many local features detected in an image

- Later we'll talk about how to use those to compute a representation of the whole image

- Local features usually exploit image gradients, ignore color

Local features: desired properties

- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

- Repeatability and flexibility
  - Robustness to expected variations: the same feature can be found in several images despite geometric/photometric transformations
  - Maximize correct matches

- Distinctiveness
  - Each feature has a distinctive description
  - Minimize wrong matches

- Compactness and efficiency
 - Many fewer features than image pixels

Adapted from K. Grauman and D. Hoiem
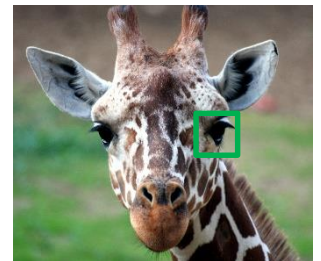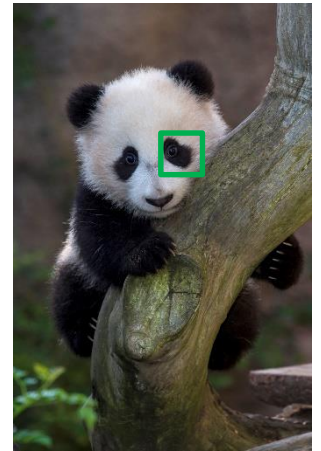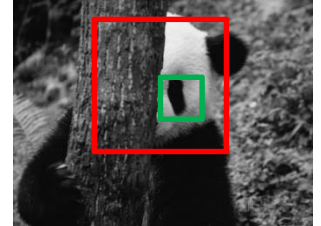
Interest(ing) points

- Note: "interest points" = "keypoints", also sometimes called "features"

- Many applications
  - Image search: which points would allow us to match images between query and database?

  - Recognition: which patches are likely to tell us something about the object category?

  - 3D reconstruction: how to find correspondences across different views?

  - Tracking: which points are good to track?

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?

original

deformed
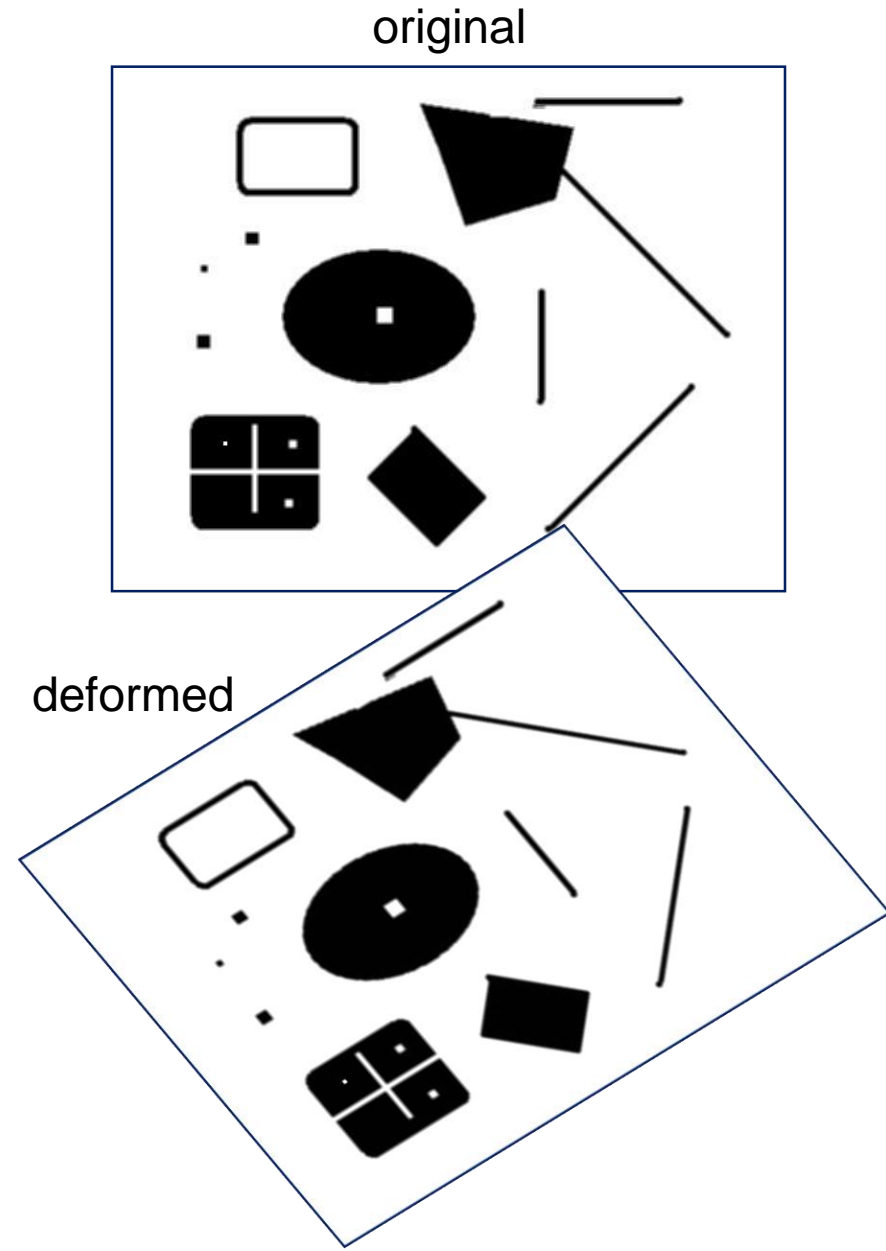
Choosing interest points
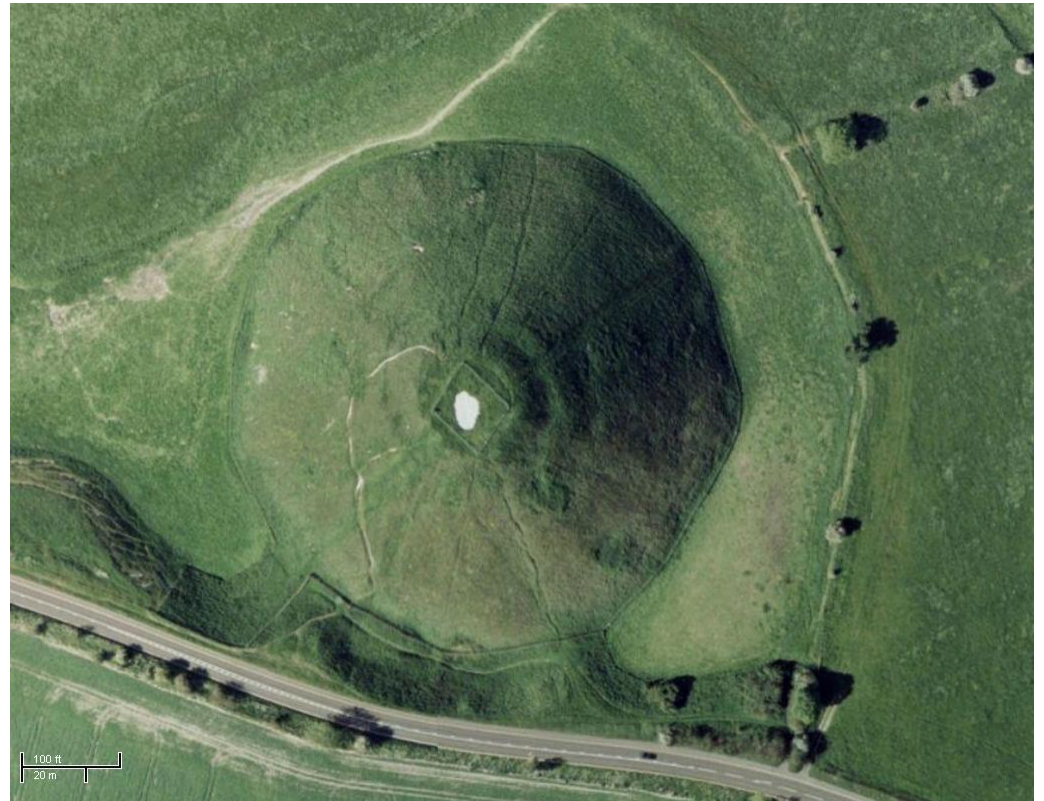
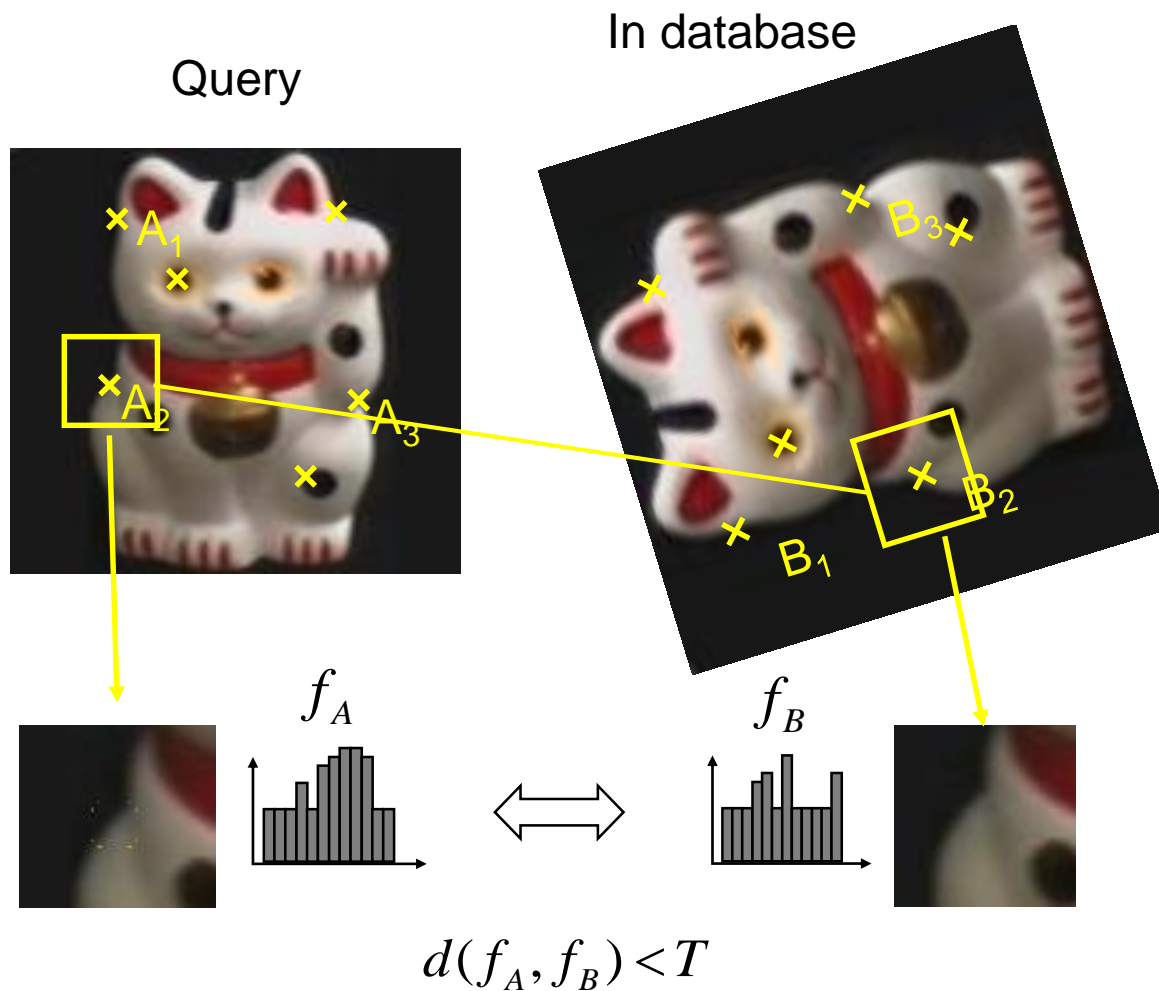Where would you tell your friend to meet you?

→ Corner detection

Choosing interest points

Where would you tell your friend to meet you?

→ Blob detection

Application 1: Keypoint Matching for Search

Query

In database

f_A

f_B

$$d(f_A, f_B) < T$$

1. Find a set of distinctive key-points

2. Define a region around each keypoint (window)

3. Compute a local descriptor from the region

4. Match descriptors

In database

Query

Goal:

We want to detect *repeatable* and *distinctive* points

- *Repeatable:* so that if images are slightly different, we can still retrieve them

- *Distinctive:* so we don't retrieve irrelevant content

Application 2: Panorama stitching

We have two images – how do we combine them?

Application 2: Panorama stitching

We have two images – how do we combine them?
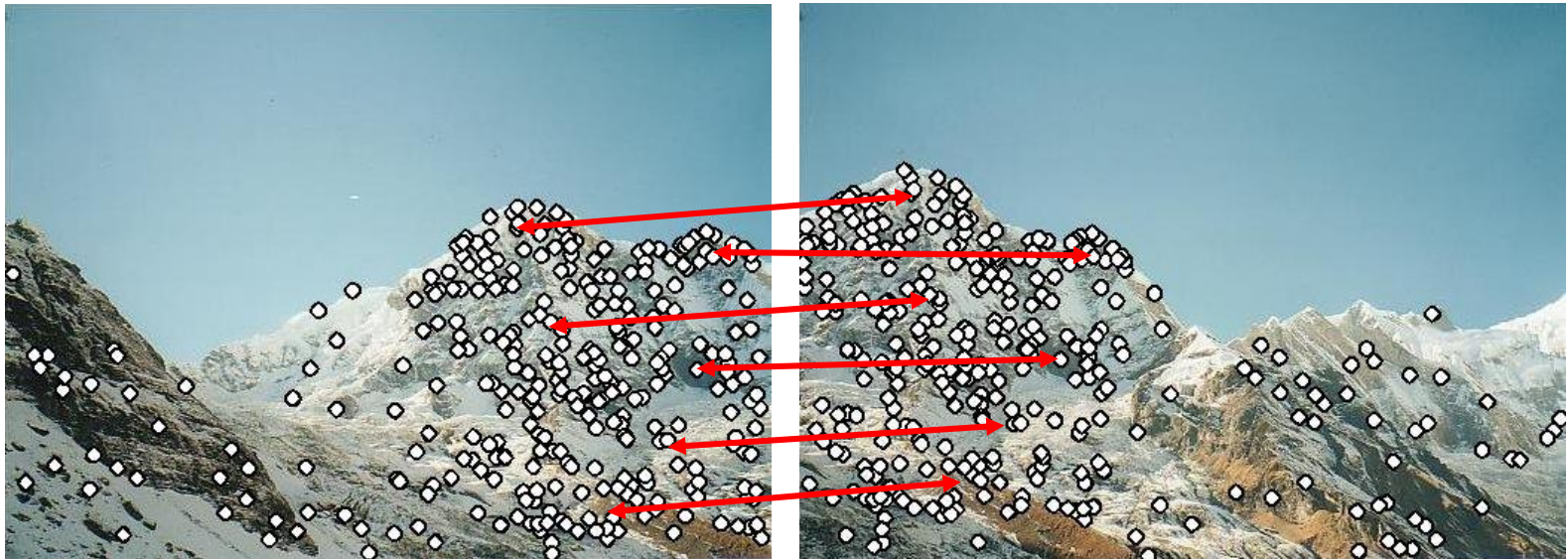

Step 1: extract features
Step 2: match features

Application 2: Panorama stitching

We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images

Desired properties of local features

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

- We want to detect (at least some of) the same points in both images → want *repeatability* of the interest operator
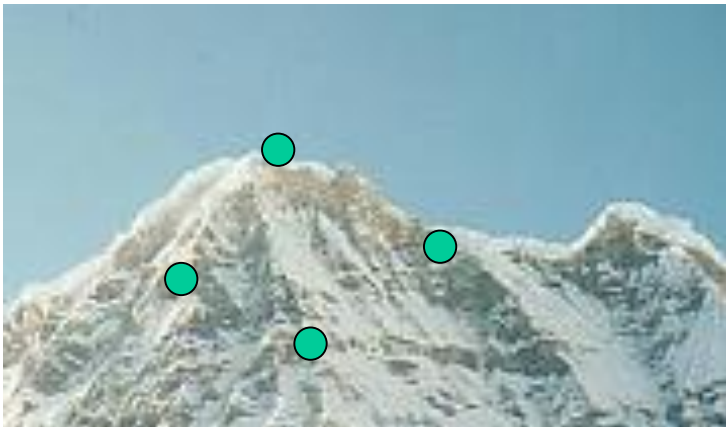
Desired properties of local features

- We want to be able to reliably determine which point goes with which → want feature *distinctiveness*

- In brief, want some invariance to geometric and photometric differences between the two views, without finding many false matches
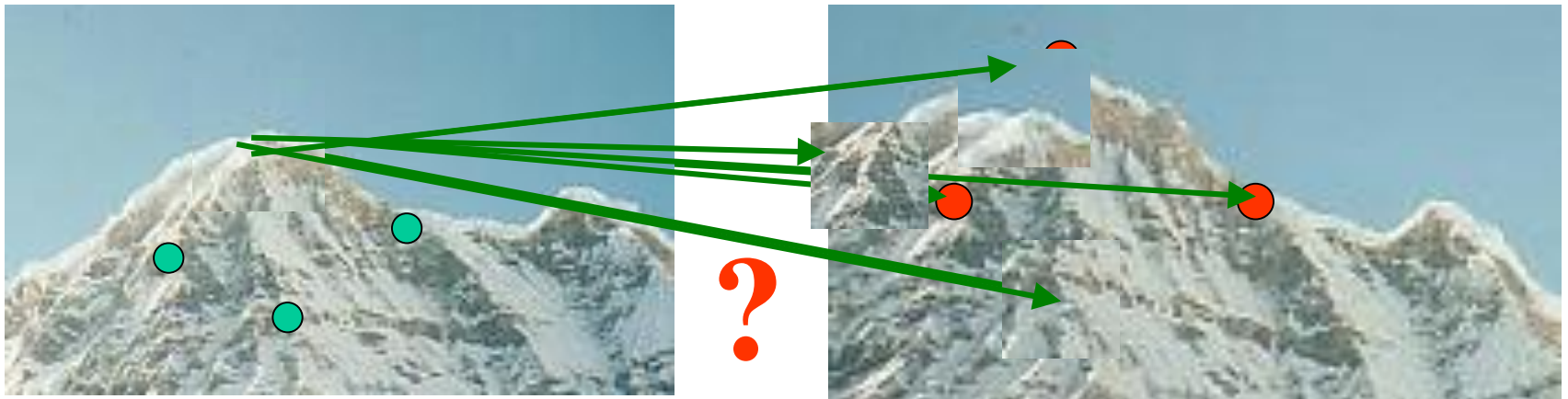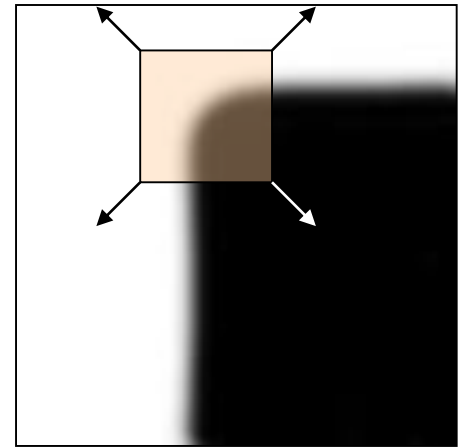
Corners as distinctive interest points

- We should easily recognize the keypoint by looking through a small window

- Shifting a window in *any direction* should give *a large change* in intensity

🔴 Candidate keypoint
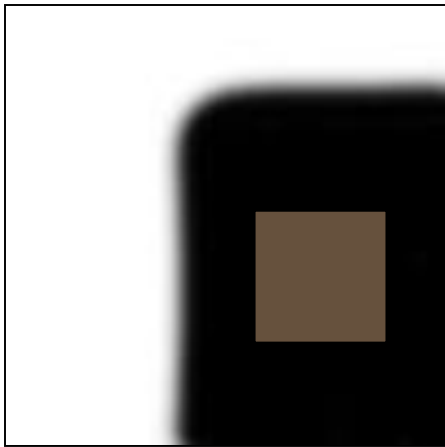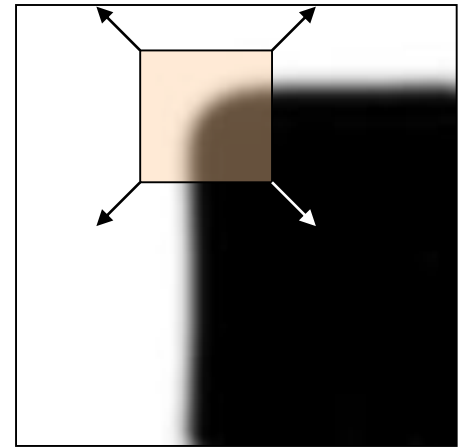
"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Corners as distinctive interest points

- We should easily recognize the keypoint by looking through a small window

- Shifting a window in *any direction* should give *a large change* in intensity
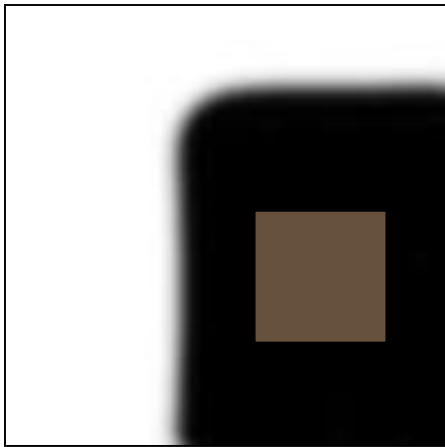
"flat" region:
no change in
all directions

"edge":
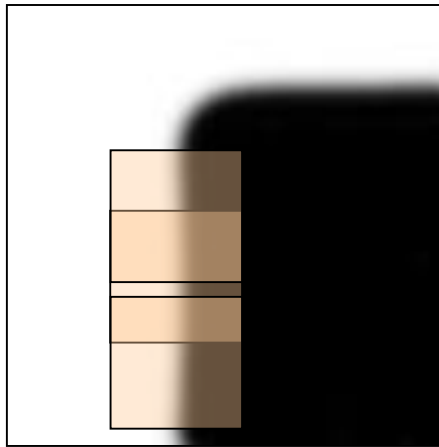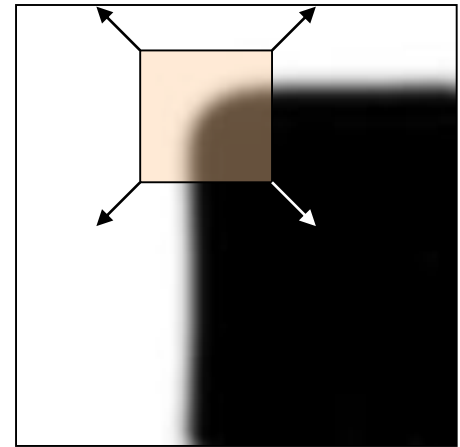no change along
the edge direction

"corner":
significant change
in all directions

Corners as distinctive interest points

- We should easily recognize the keypoint by looking through a small window

- Shifting a window in *any direction* should give *a large change* in intensity
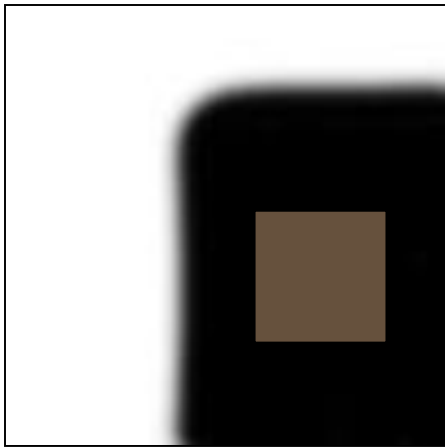
"flat" region:
no change in
all directions

"edge":
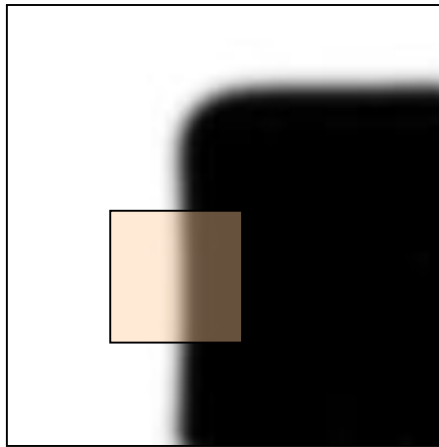no change along
the edge direction

"corner":
significant change
in all directions

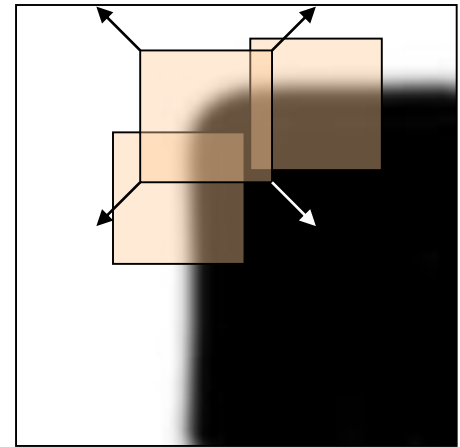# **Corners** as distinctive interest points

- We should easily recognize the keypoint by looking through a small window

- Shifting a window in *any direction* should give *a large change* in intensity

"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Corners as distinctive interest points

- We should easily recognize the keypoint by looking through a small window

- Shifting a window in *any direction* should give *a large change* in intensity

"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

What points would you choose?

Harris Detector: Mathematics

Window-averaged squared change of intensity induced by shifting the patch for a fixed candidate keypoint by $[u, v]$:

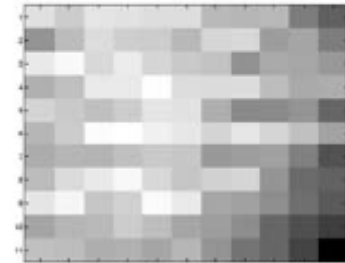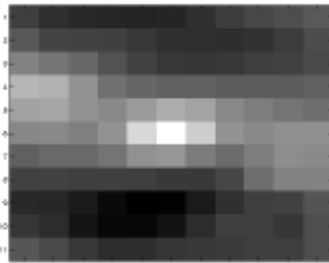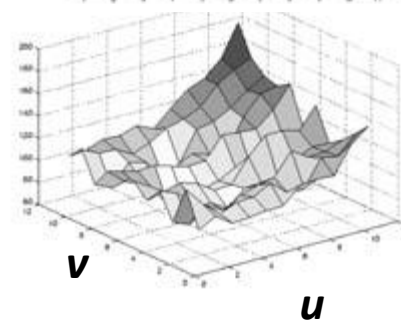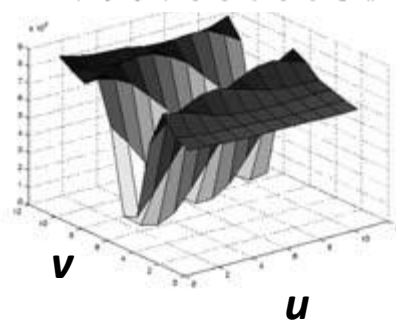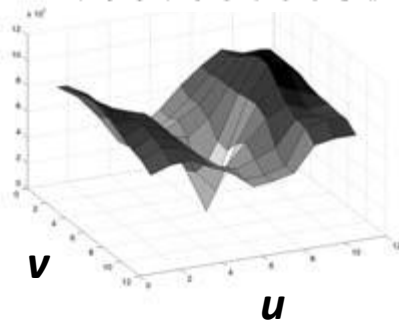$$E(u, v) = \sum_{x,y} \left[I(x+u, y+v) - I(x, y) \right]^2$$

Window function over neighbors (x, y)

Shifted intensity

Intensity

Candidate keypoint

Window function $w(x, y) =$
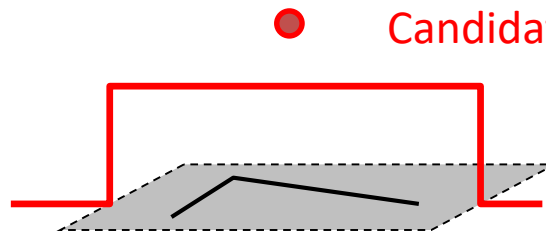
1 in window, 0 outside
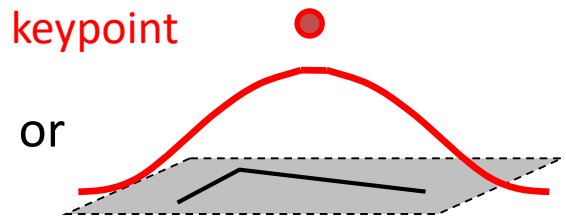
or

Gaussian

D. Frolova, D. Simakov

Harris Detector: Mathematics

Window-averaged squared change of intensity induced by shifting the patch for a fixed candidate keypoint by $[u,v]$:
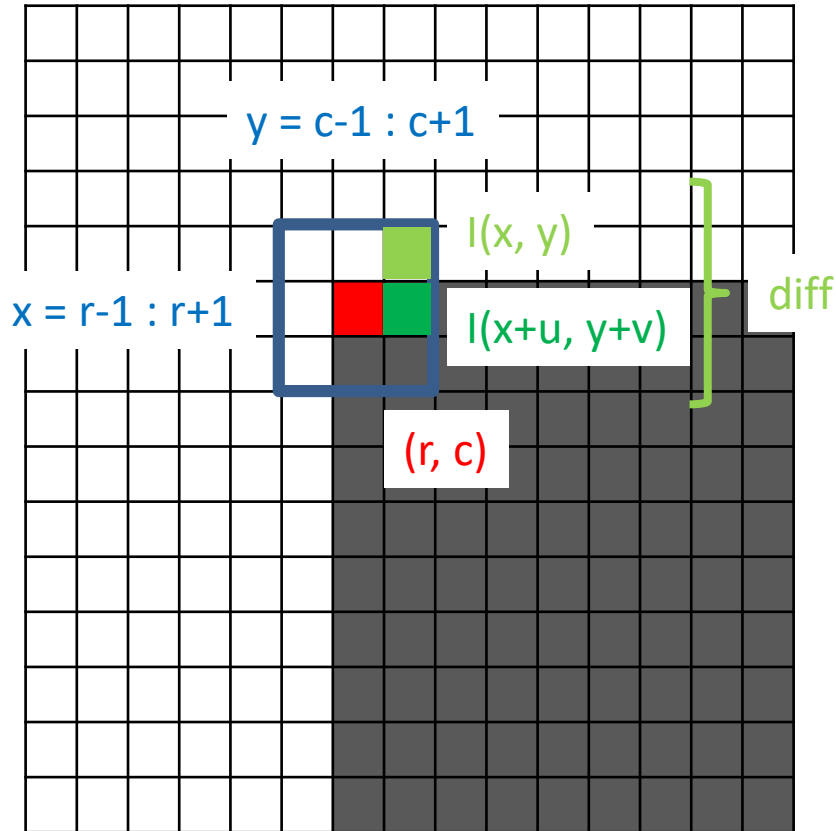
$$E(u,v) = \sum_{x,y} w(x,y) \left[I(x+u, y+v) - I(x,y) \right]^2$$

Win
over n

$E(u, v)$

v u v u v u

Harris Detector: Mathematics

y = c-1 : c+1

x = r-1 : r+1

I(x, y)

I(x+u, y+v)

diff

(r, c)

For every pixel (r, c) as candidate keypoint

Initialize E = zeros(max_offset, max_offset)

For each offset (u, v)

Initialize *sum* to 0

For each neighbor (x, y) of (r, c)

$sum = sum + [I(x, y) - I(x+u, y+v)]^2$

E(u, v) = *sum*

Plot E(u, v)

Here u = 1, v = 0

Harris Detector: Mathematics

We can approximate the autocorrelation surface between a patch and itself, shifted by [u,v], as:

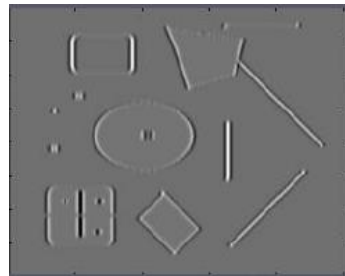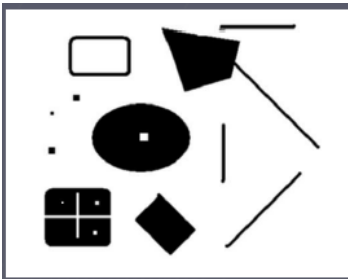$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a 2 × 2 matrix computed from image derivatives:

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

Harris Detector: Mathematics

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_h^2 & I_h I_v \\ I_h I_v & I_v^2 \end{bmatrix}$$
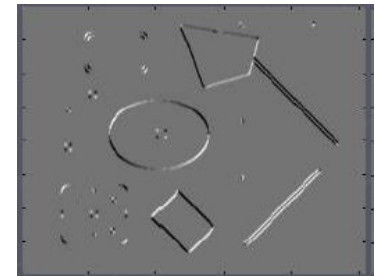
Notation:

$$I_h \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_v \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_h I_v \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Harris Detector: Mathematics

y = c-1 : c+1

x = r-1 : r+1

(r, c)
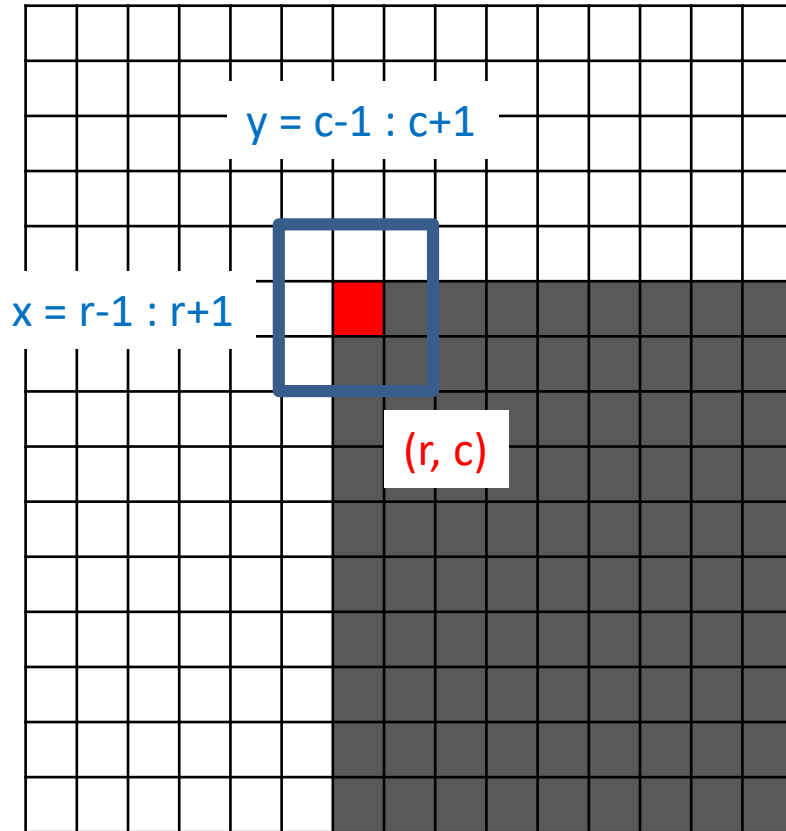
Let I_h (of size width x height of the image) be the image derivative in the horizontal direction, I_y be derivative in the vertical direction.
(Both require one correlation op to compute.)

For every pixel (r, c) as candidate keypoint

    Initialize M = zeros(2, 2)

    For x = r-1 : r+1

        For y = c-1 : c+1

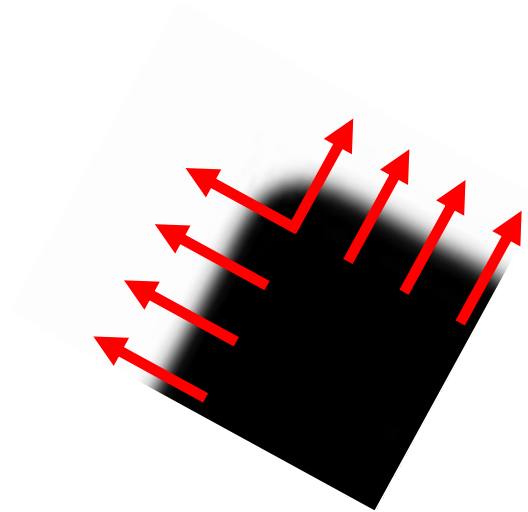 $M(1, 1) = M(1, 1) + I_h(x, y)^2$

            $M(1, 2) = ?$

            $M(2, 1) = ?$

            $M(2, 2) = ?$

What does the matrix M reveal?

Since *M* is symmetric, we have

$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

$$M x_i = \lambda_i x_i$$

The *eigenvalues* of *M* reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.
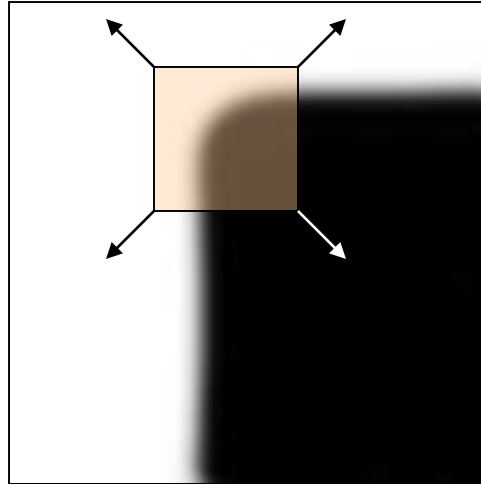
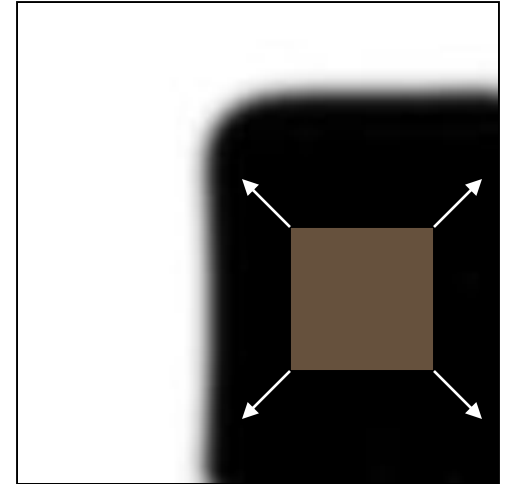# Corner response function

"edge":

$\lambda_1 \gg \lambda_2$

$\lambda_2 \gg \lambda_1$
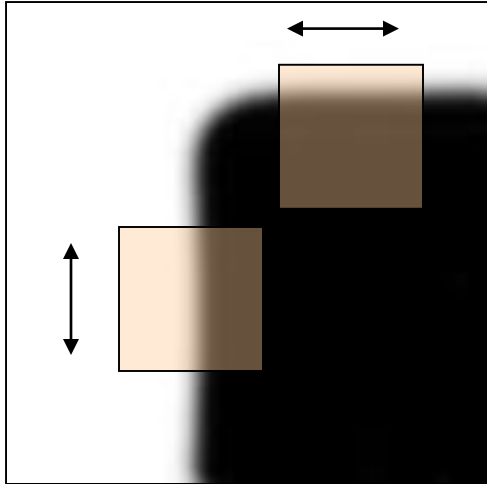
"corner":

λ_1 and λ_2 are large,

$\lambda_1 \sim \lambda_2$

"flat" region:

λ_1 and λ_2 are small

Harris Detector: Mathematics

Measure of corner response:

$$R = \det M - k \left(\text{trace } M \right)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$\text{trace } M = \lambda_1 + \lambda_2$$

Because M is symmetric

(k – empirical constant, k = 0.04-0.06)

Harris Detector: Algorithm

- Compute image gradients I_h and I_v for all pixels
- For each pixel
  - Compute

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

  by looping over neighbors x, y

  - compute

  $$R = \det M - k \left(\operatorname{trace} M \right)^2$$

  (k : empirical constant, $k = 0.04\text{-}0.06$)

- Find points with large corner response function R (R > threshold)
- Non-max suppression: Take the points of locally maximum R as the detected feature points (i.e., pixels where R is bigger than for all the 4 or 8 neighbors)

34

Example of Harris application

Example of Harris application

- Corner response at every pixel
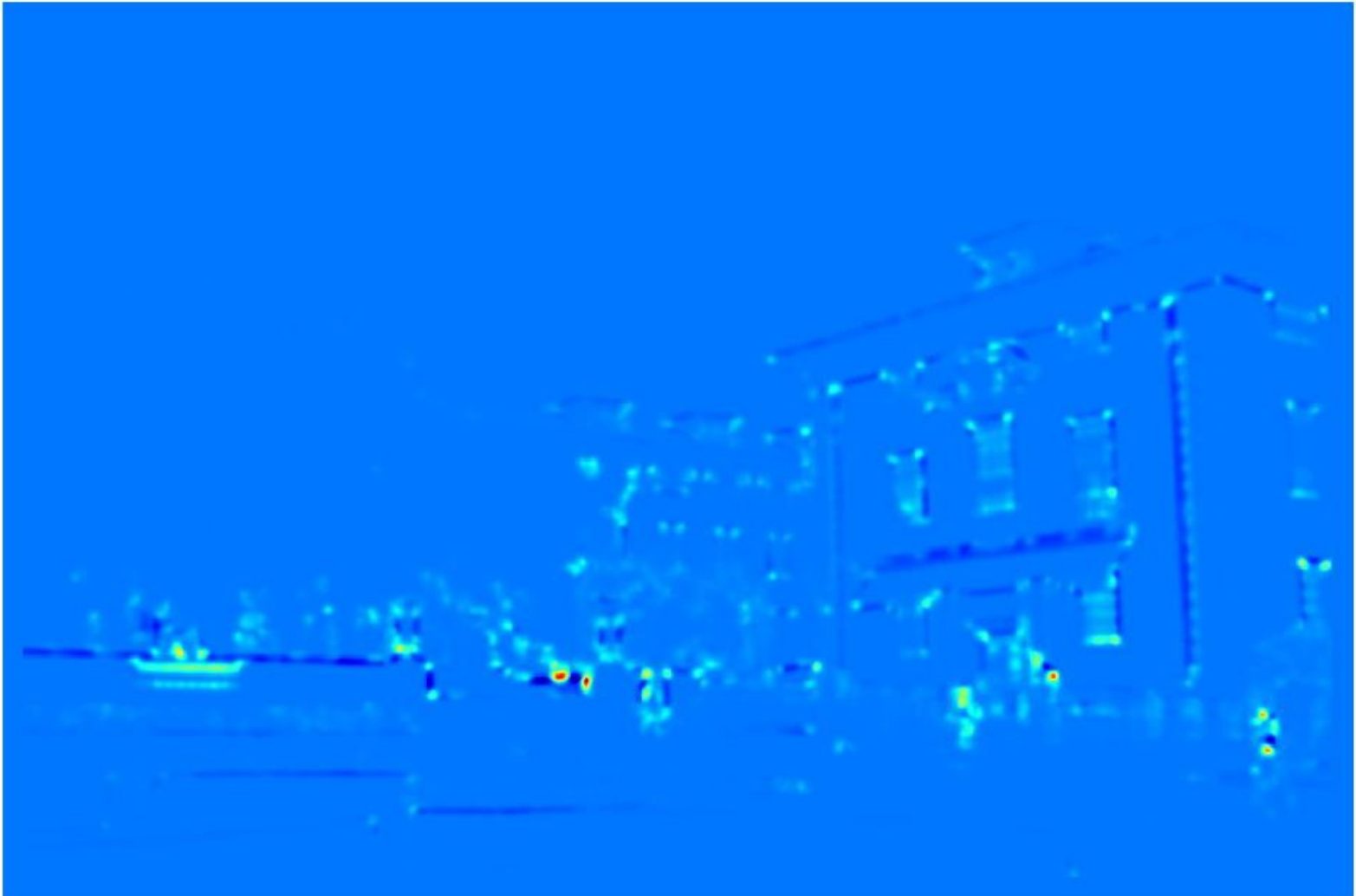
More Harris responses
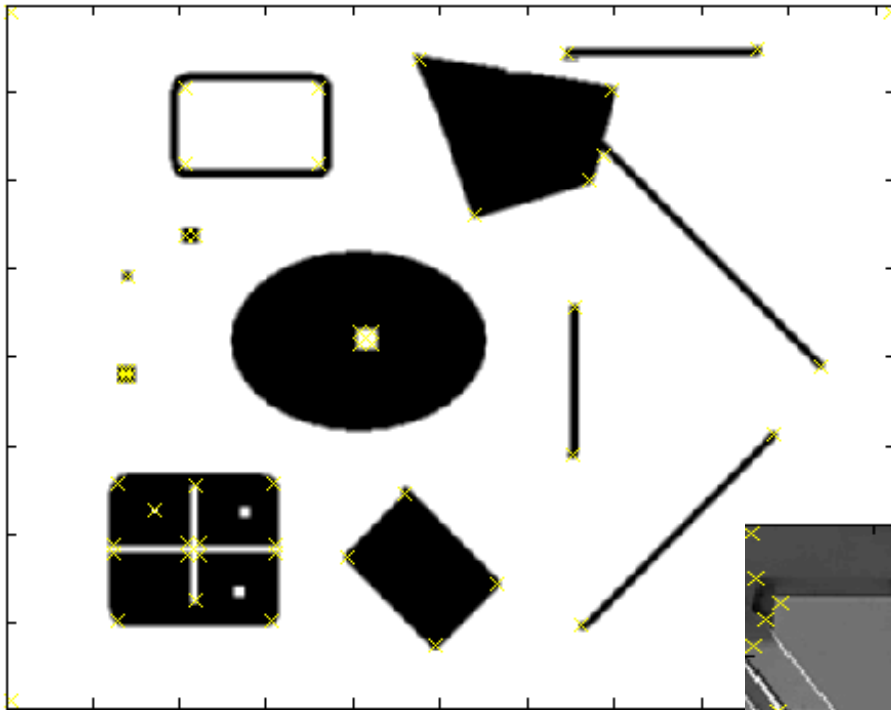
Effect: A very precise corner detector.

D. Hoiem
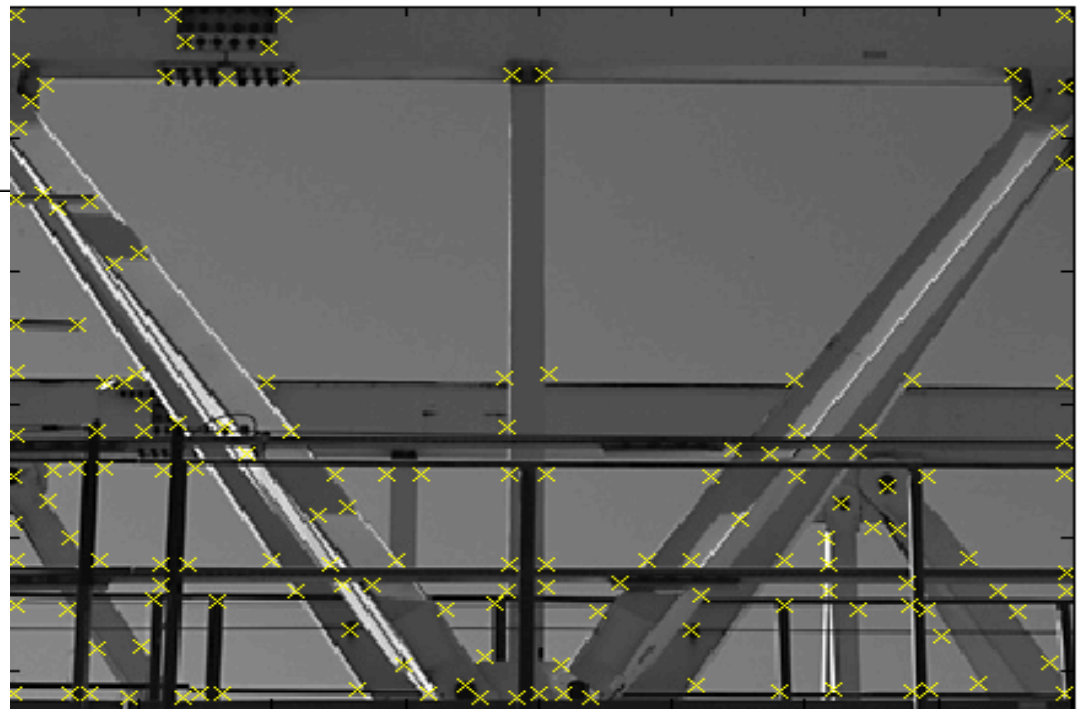
More Harris responses

Properties: Invariance vs covariance

"A function is *invariant* under a certain family of transformations if its value does not change when a transformation from this family is applied to its argument. A function is *covariant* when it commutes with the transformation, i.e., applying the transformation to the argument of the function has the same effect as applying the transformation to the output of the function. […]
[For example,] the area of a 2D surface is invariant under 2D rotations, since rotating a 2D surface does not make it any smaller or bigger.
But the orientation of the major axis of inertia of the surface is covariant under the same family of transformations, since rotating a 2D surface will affect the orientation of its major axis in exactly the same way."

What happens if: Affine intensity change
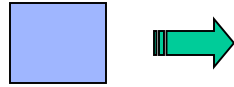
$$I \rightarrow a\,I + b$$

- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

- Intensity scaling: $I \rightarrow a\,I$

R

threshold

x (image coordinate)

R

x (image coordinate)

Partially invariant to affine intensity change

What happens if: Image translation

- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation (on image level), corner response is invariant (on patch level)

What happens if: Image rotation
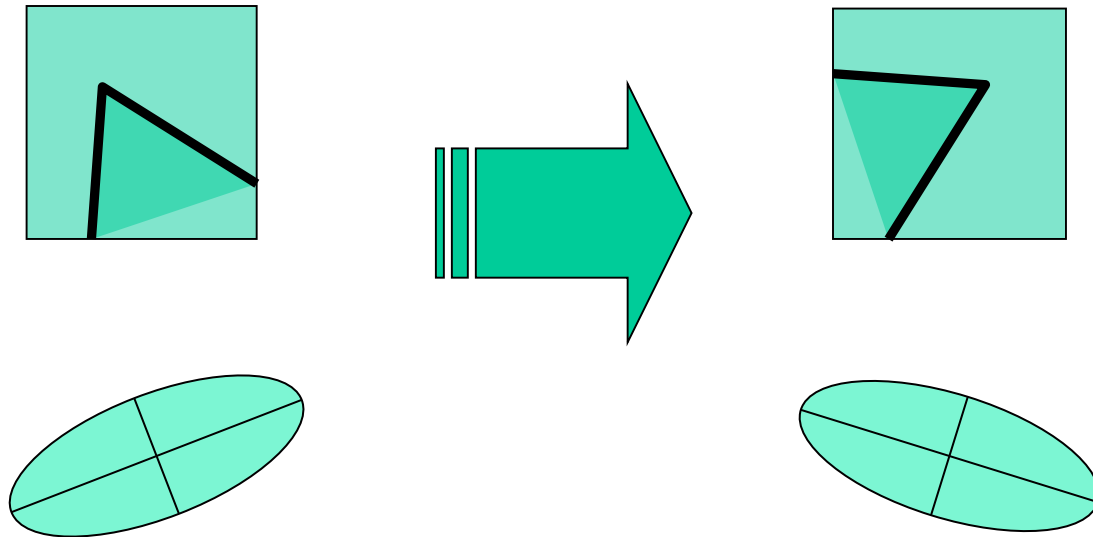
Second moment ellipse rotates but its shape
(i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation (on image level),
corner response is invariant (on patch level)

What happens if: Scaling

Invariant to image scale?

image

zoomed image

What happens if: Scaling

Corner

All points will
be classified
as edges
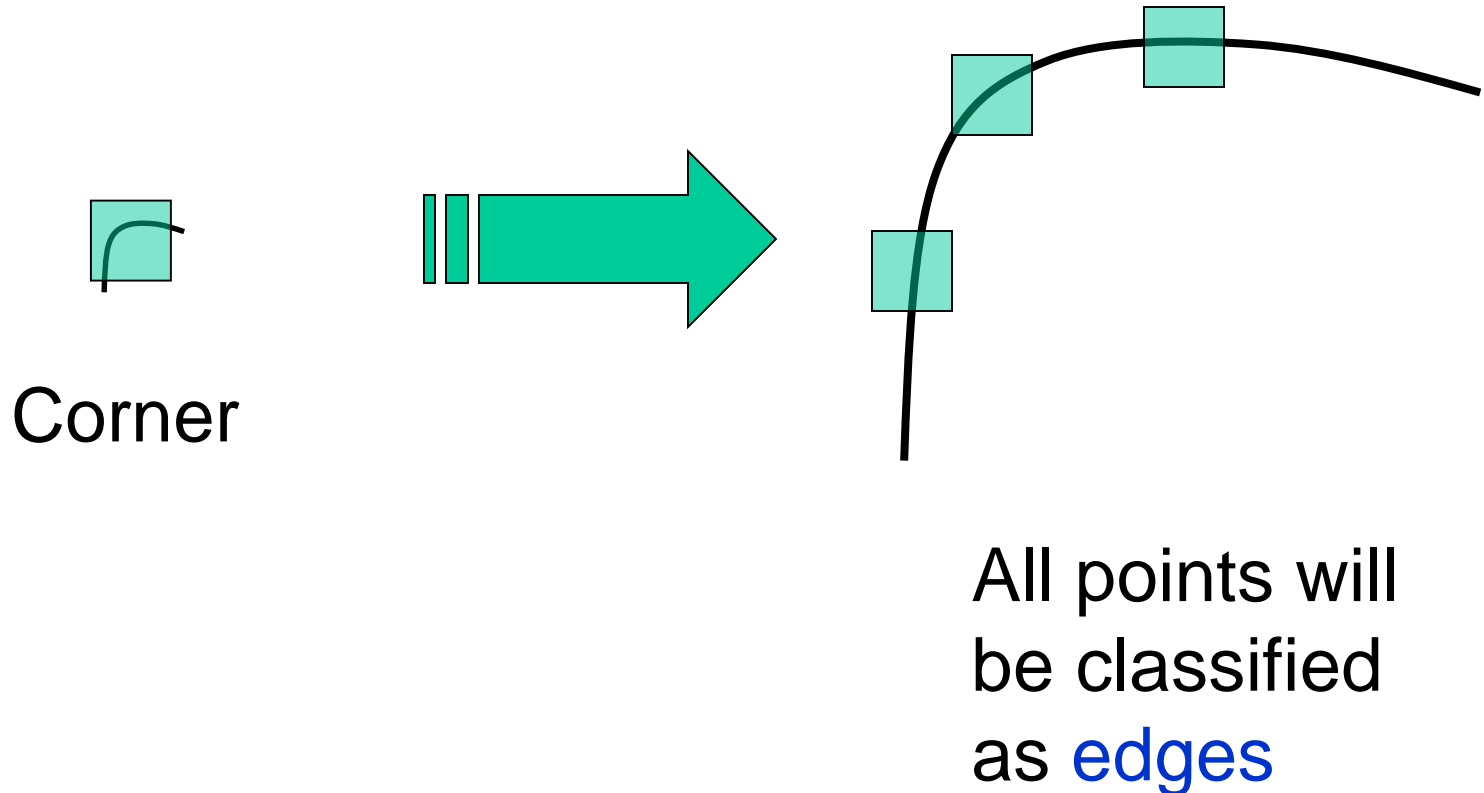
Corner location is not covariant to scaling!

Scale invariant detection

- Problem:
  - How do we choose corresponding windows *independently* in each image?
  - Do objects have a <u>characteristic scale</u> that we can identify?
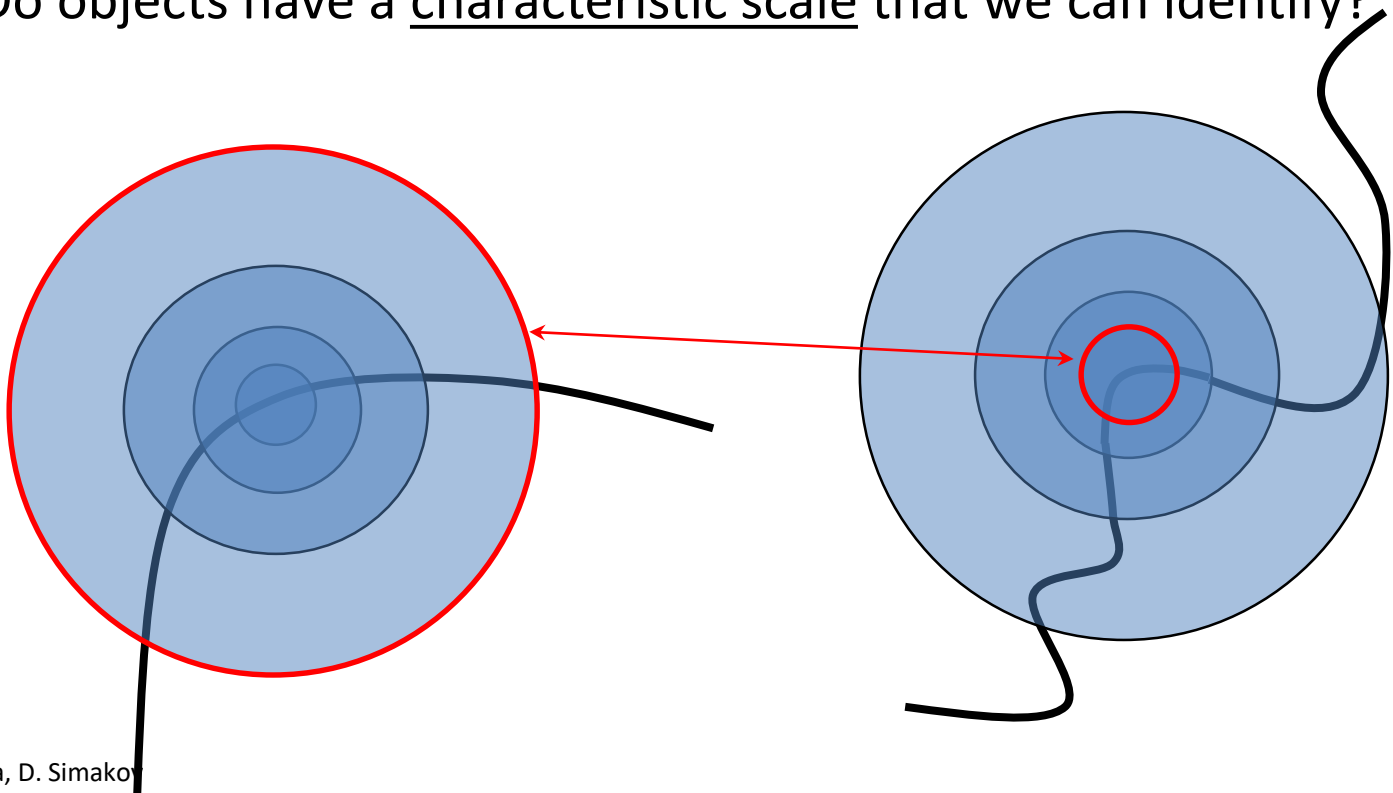
Scale invariant detection

- Solution:
  - Design a function on the region which has the same shape even if the image is resized
  - Take a local maximum of this function

Image 1

f

region size

s_1

scale = 1/2

Image 2

f

region size

s_2
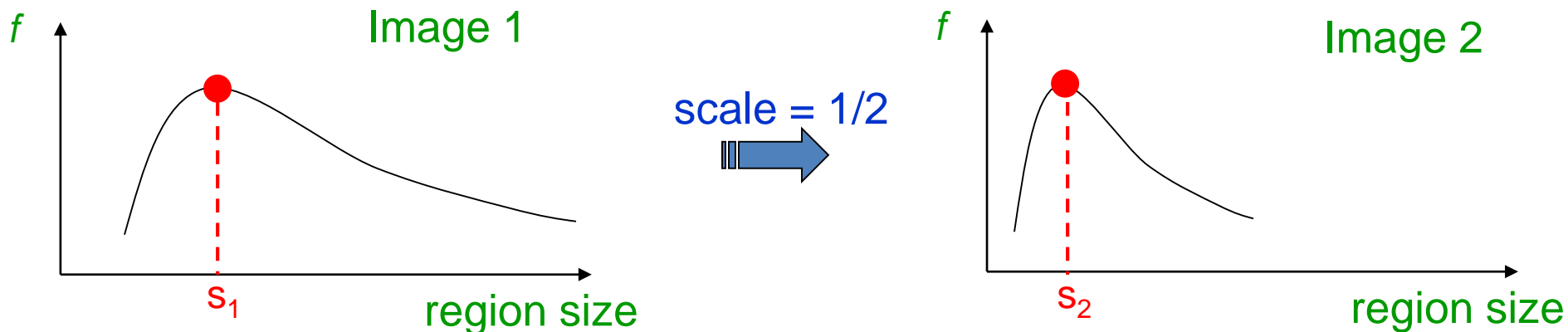
Scale invariant detection

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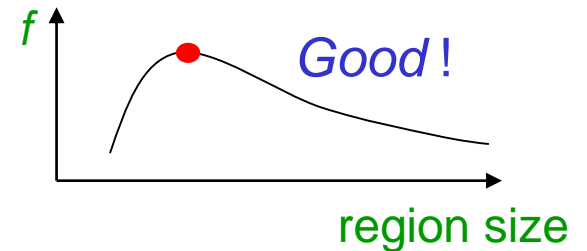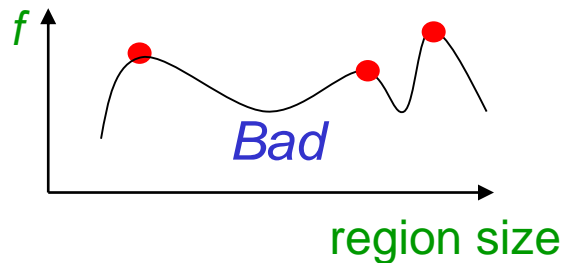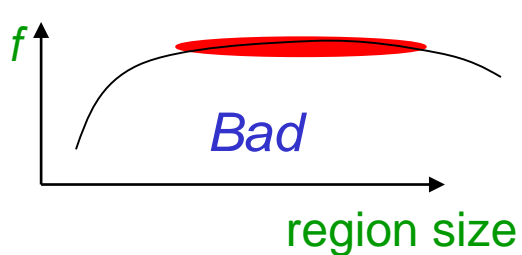
Automatic scale selection

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

How to find corresponding patch sizes?

Automatic scale selection

- Function responses for increasing scale (scale signature)

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

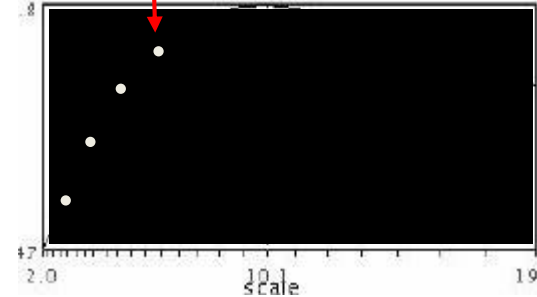$$f(I_{i_1 \ldots i_m}(x', \sigma))$$

K. Grauman, B. Leibe

Automatic scale selection

- Function responses for increasing scale (scale signature)

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

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

K. Grauman, B. Leibe

Automatic scale selection

- Function responses for increasing scale (scale signature)
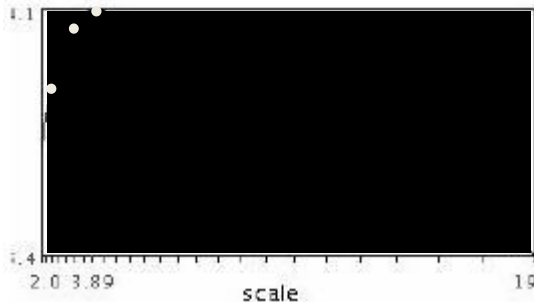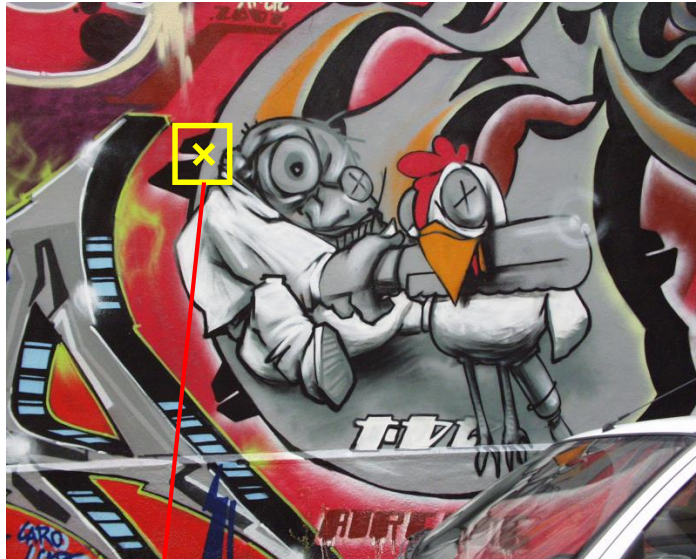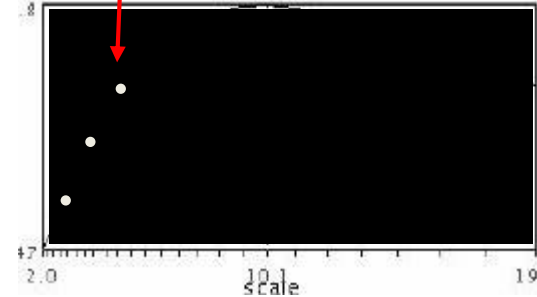
$$f(I_{i_1 \ldots i_m}(x, \sigma))$$

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

Automatic scale selection

- Function responses for increasing scale (scale signature)
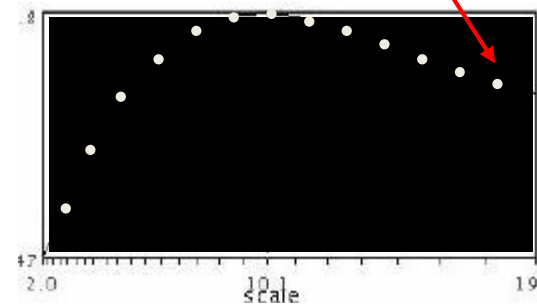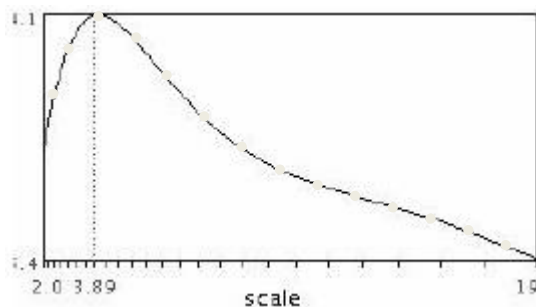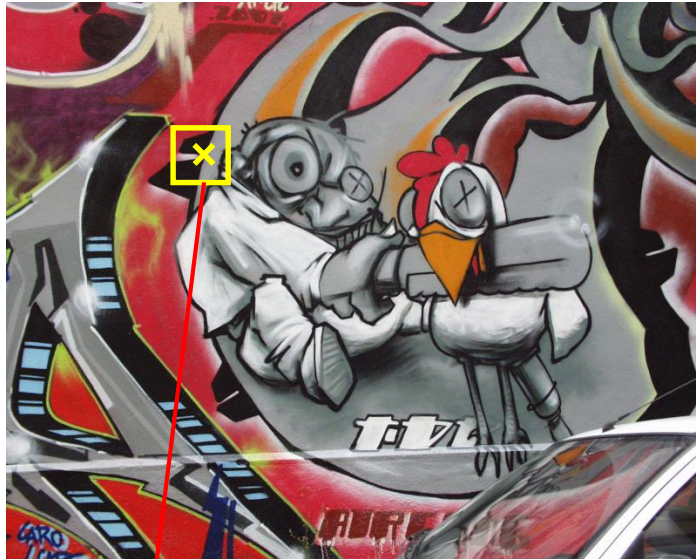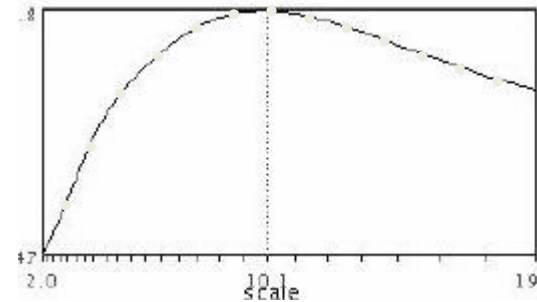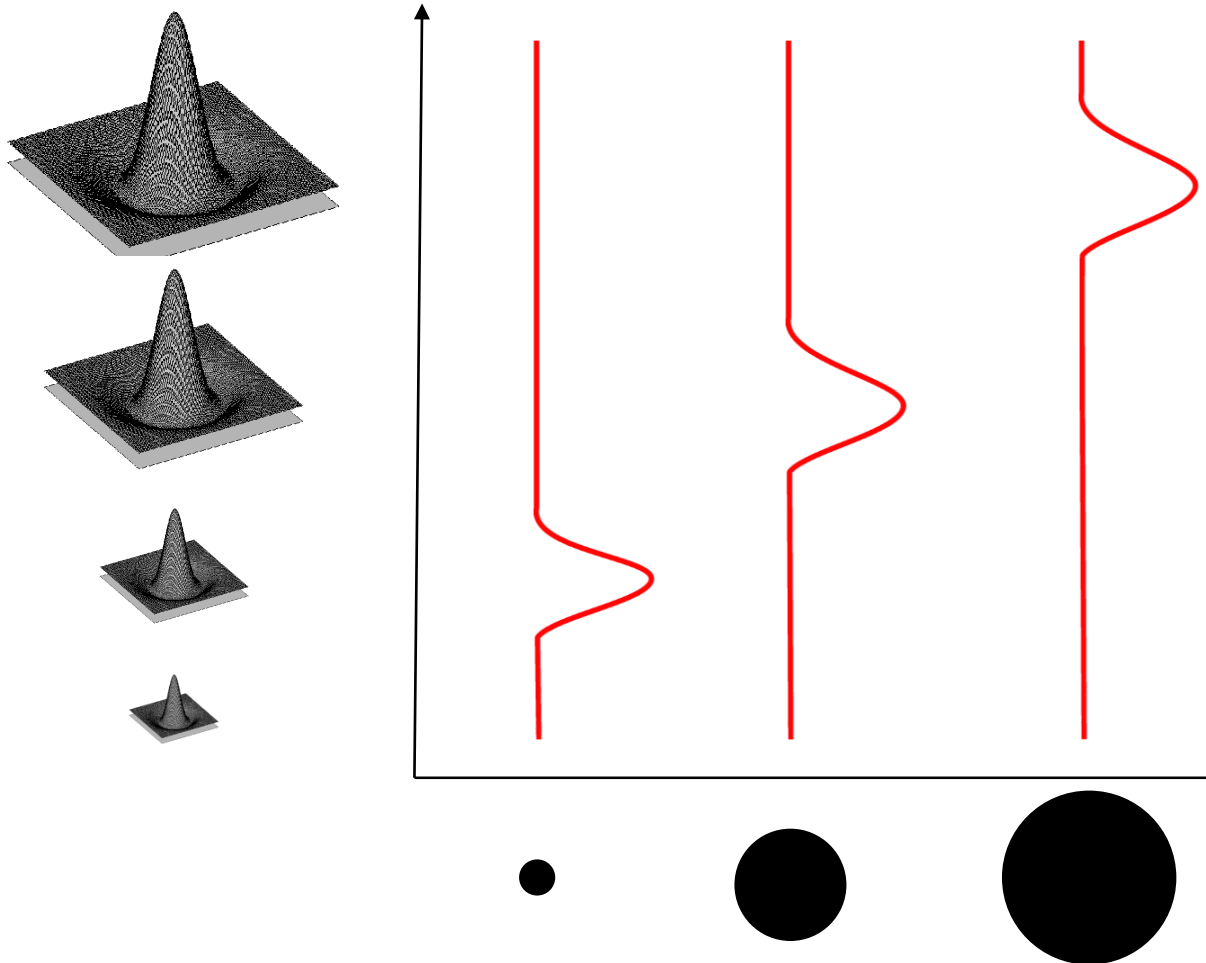

$$f(I_{i_1 \ldots i_m}(x, \sigma))$$

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

Automatic scale selection

- Function responses for increasing scale (scale signature)

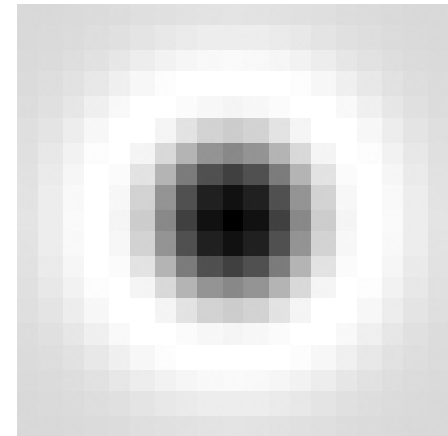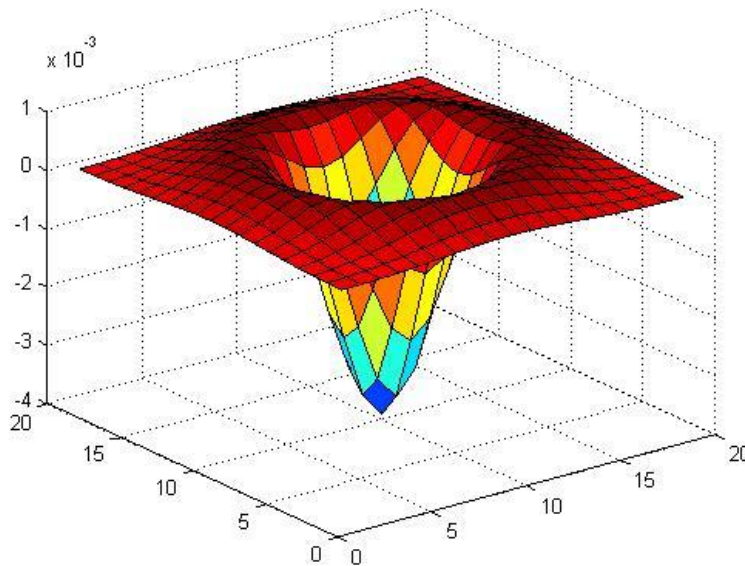$$f(I_{i_1 \dots i_m}(x, \sigma))$$

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

Automatic scale selection

- Function responses for increasing scale (scale signature)

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

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

What is a useful signature function?

- Laplacian of Gaussian = "blob" detector

Blob detection in 2D

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

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

K. Grauman

Difference of Gaussian $\approx$ Laplacian

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

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

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

(Difference of Gaussians)
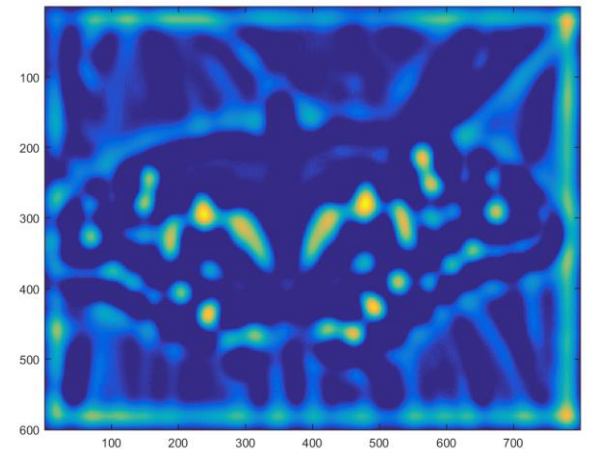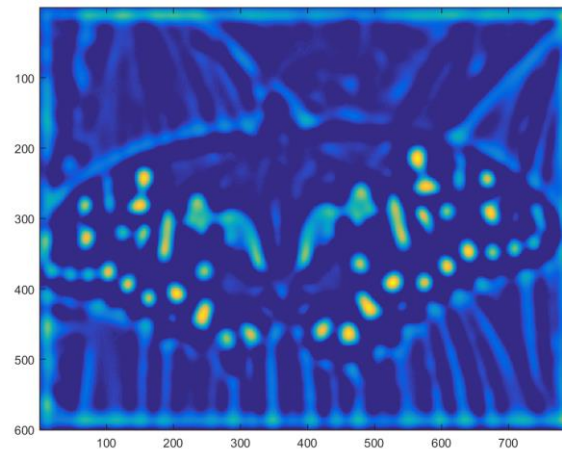

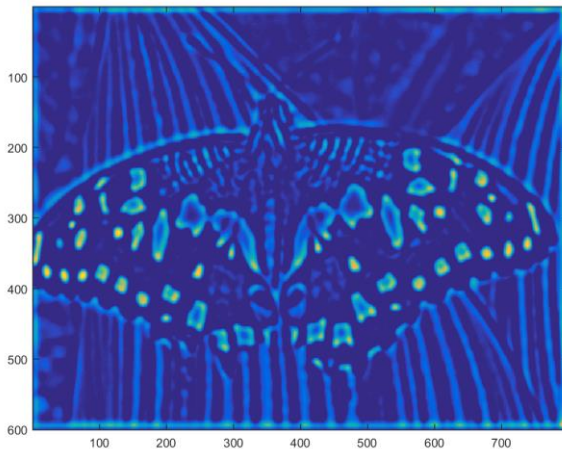$I(k\sigma)$ $I(\sigma)$ $I(k\sigma) - I(\sigma)$
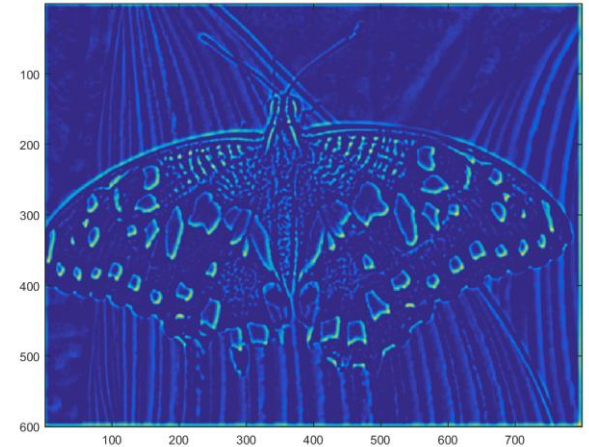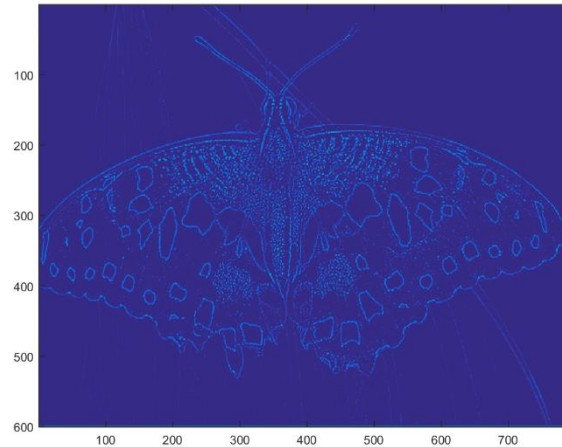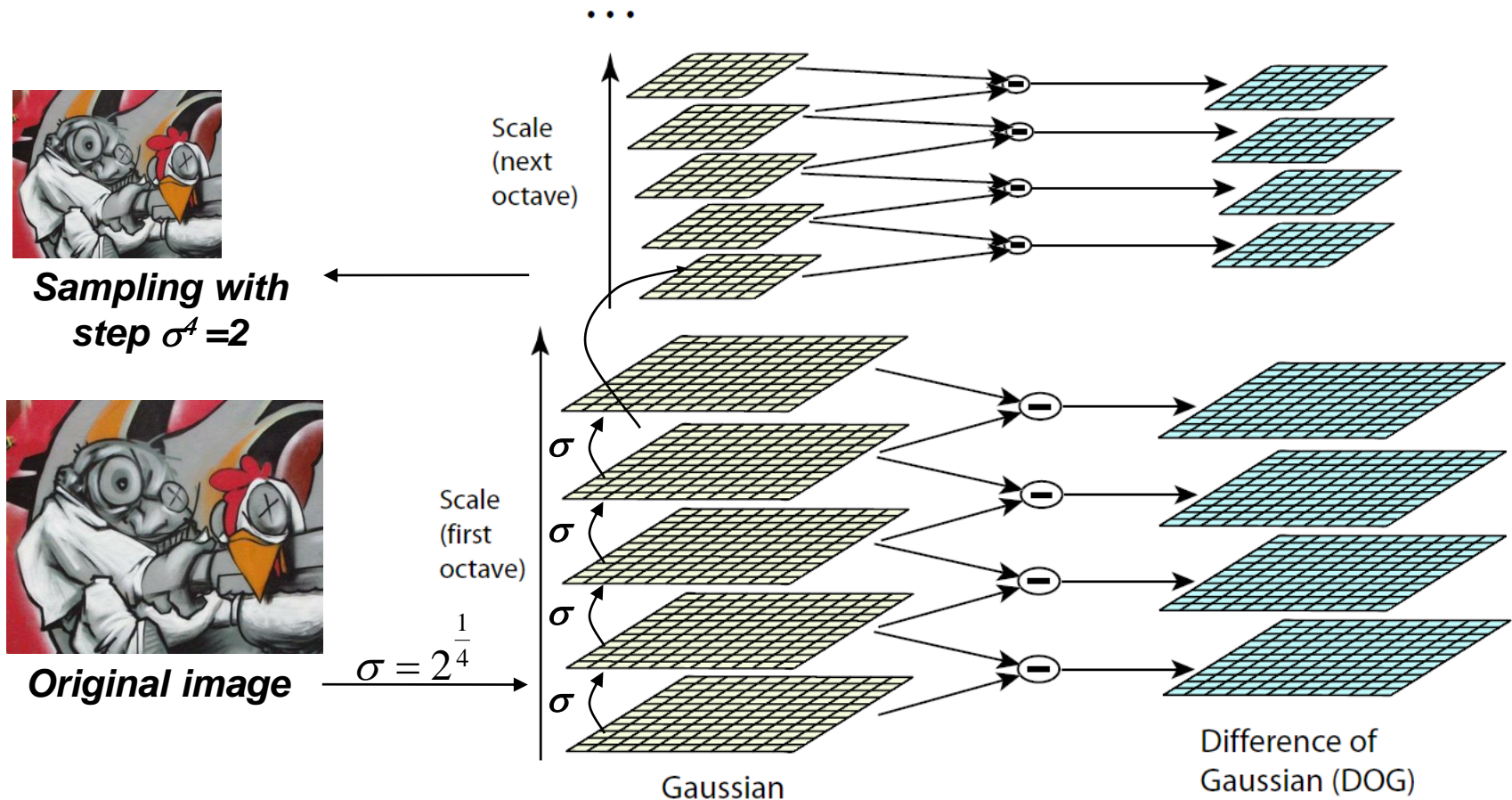

K. Grauman
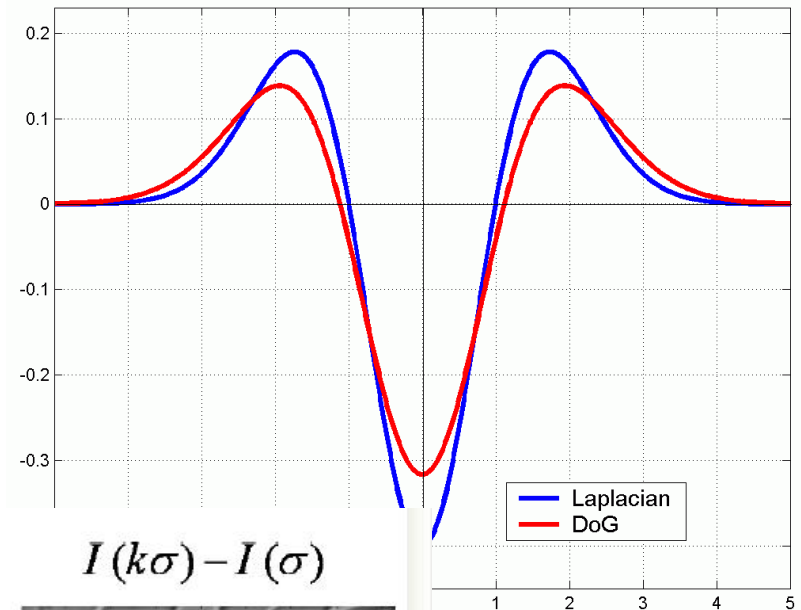
Laplacian pyramid example
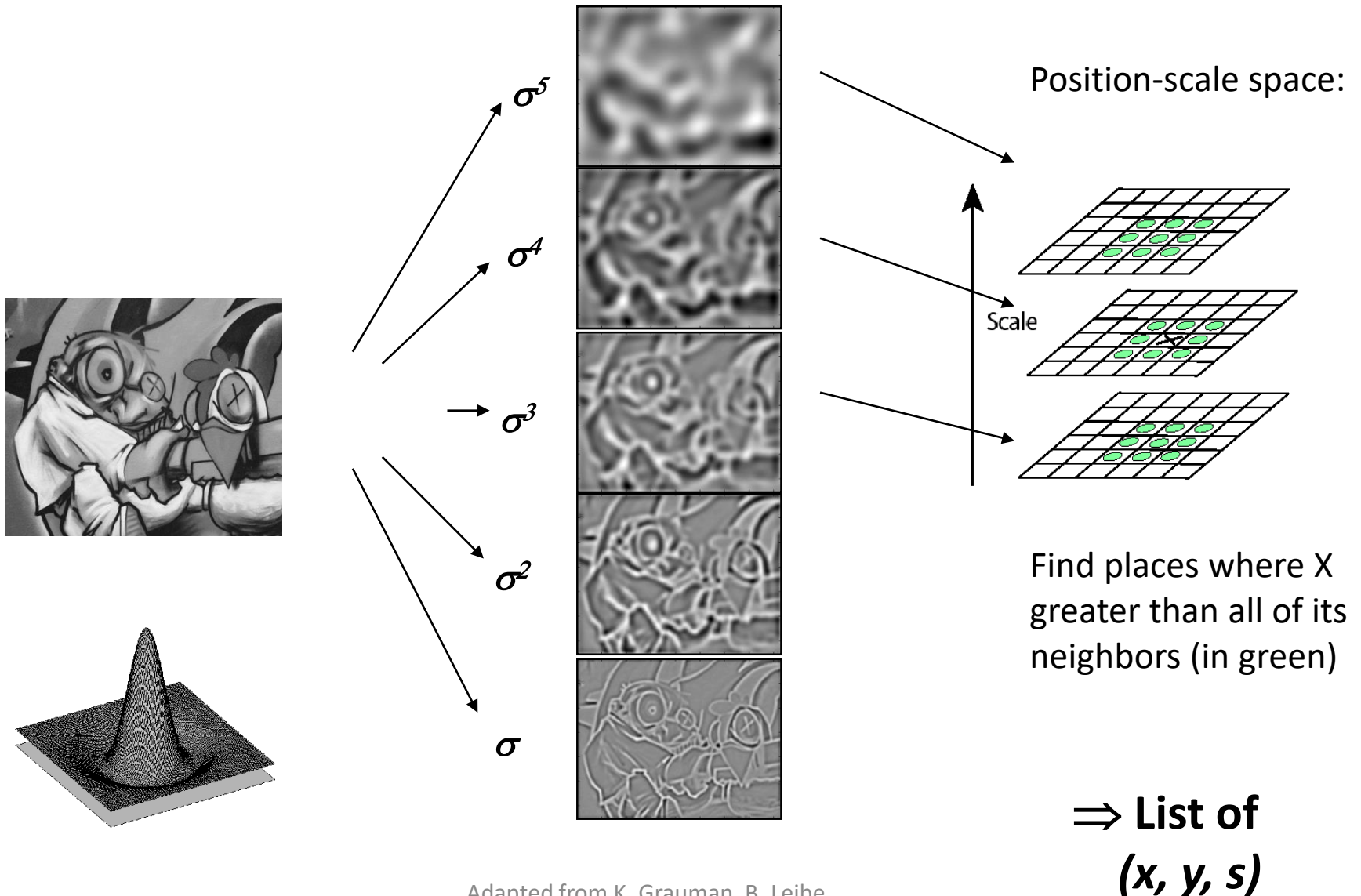
- Allows detection of increasingly coarse detail
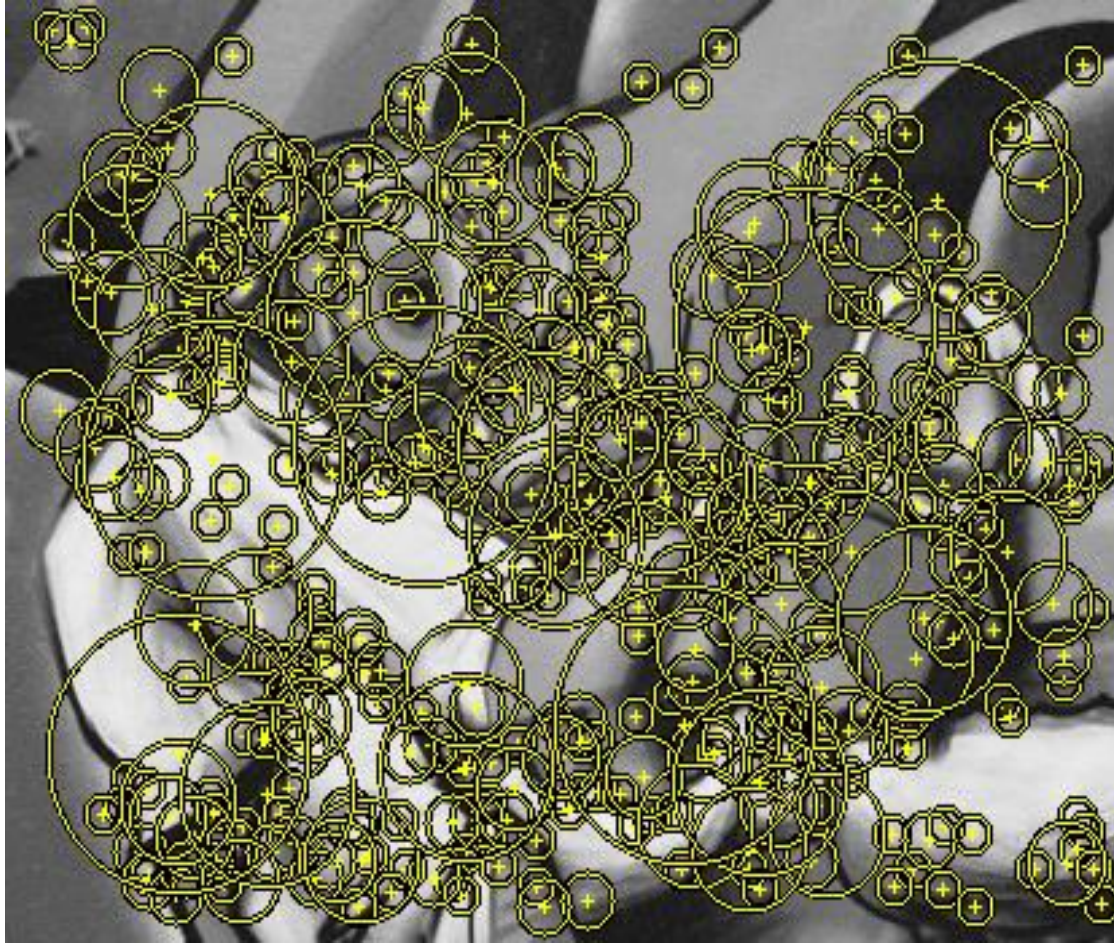
Difference of Gaussian: Efficient computation

- Computation in Gaussian scale pyramid

Sampling with step $\sigma^4 = 2$

Original image $\longrightarrow$ $\sigma = 2^{\frac{1}{4}}$

Scale (next octave)

Scale (first octave)

σ

σ

σ

σ

Gaussian

Difference of Gaussian (DOG)

Find *local maxima* in position-scale space of Difference-of-Gaussian

σ^5

σ^4

σ^3

σ^2

σ

Position-scale space:

Scale

Find places where X greater than all of its neighbors (in green)

$\Rightarrow$ **List of (x, y, s)**

Adapted from K. Grauman, B. Leibe

Results: Difference-of-Gaussian

K. Grauman, B. Leibe
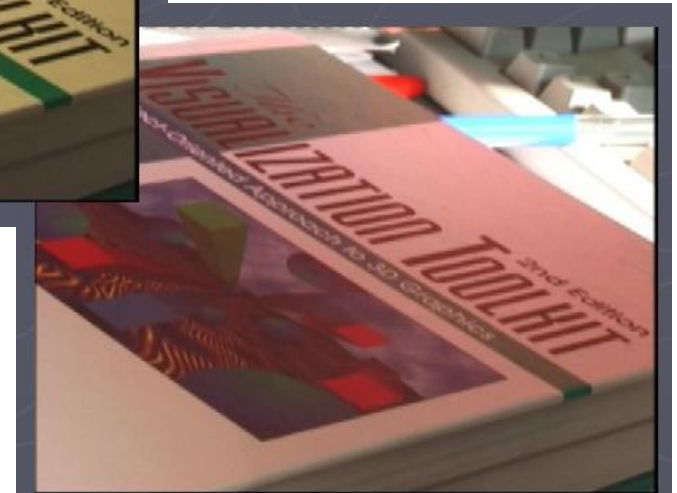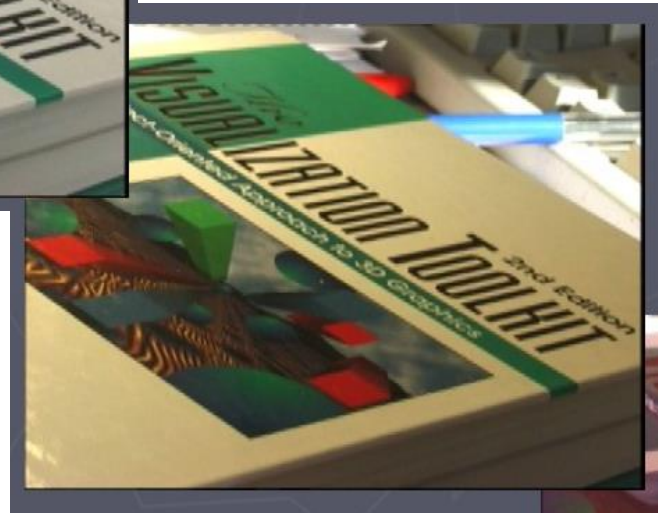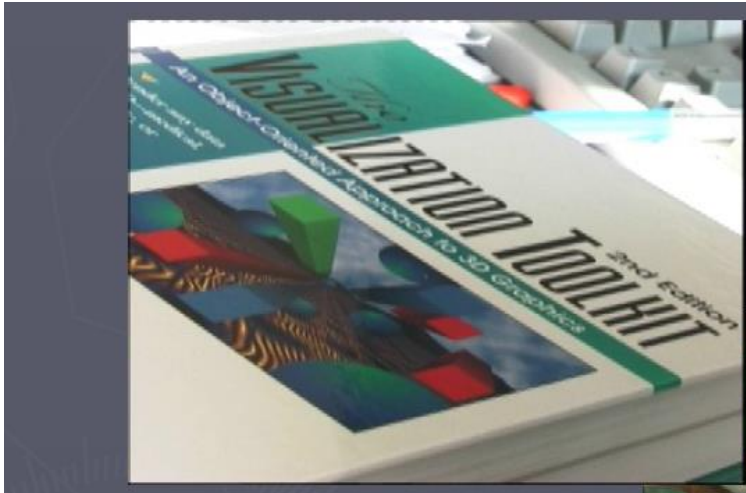
Plan for this lecture

- Feature detection / keypoint extraction
  - Corner detection
  - Properties
  - Blob detection
- Feature description (of detected features)
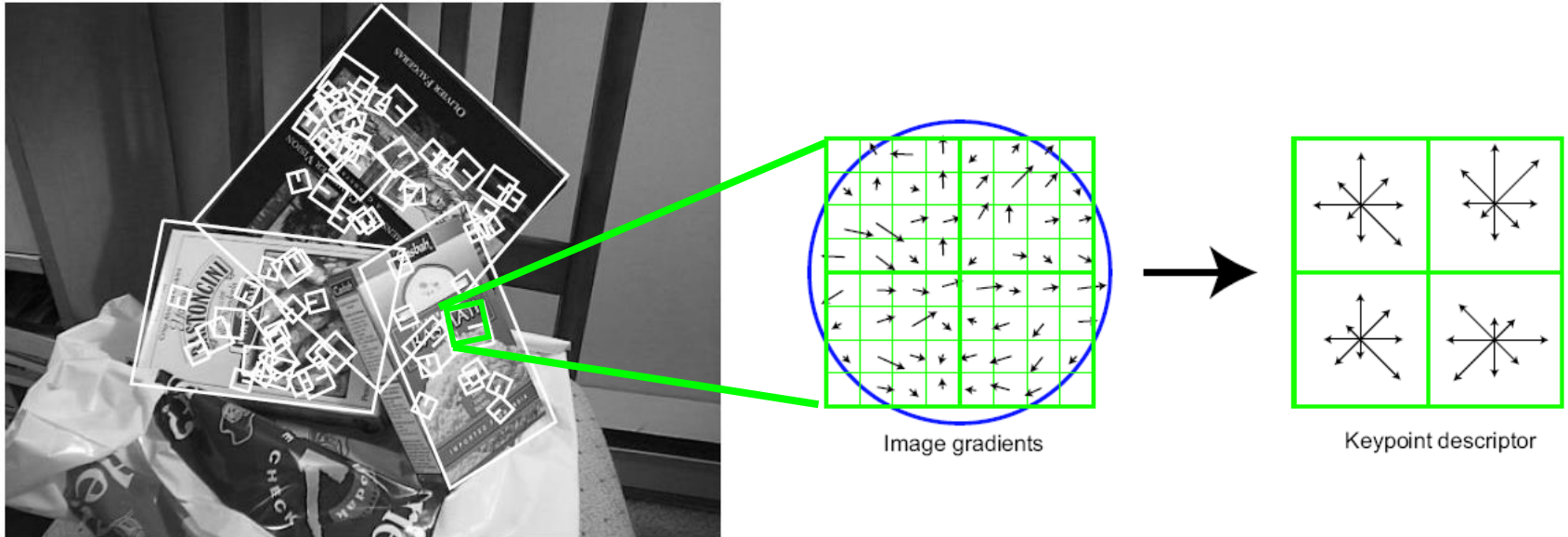- Matching features across images

Geometric transformations
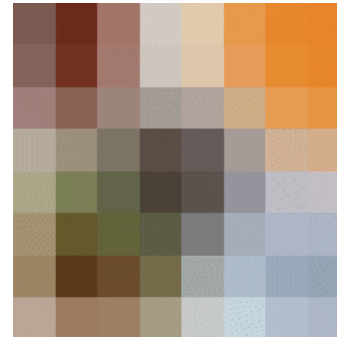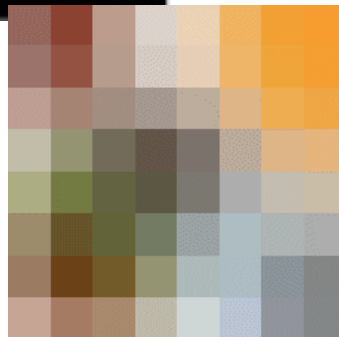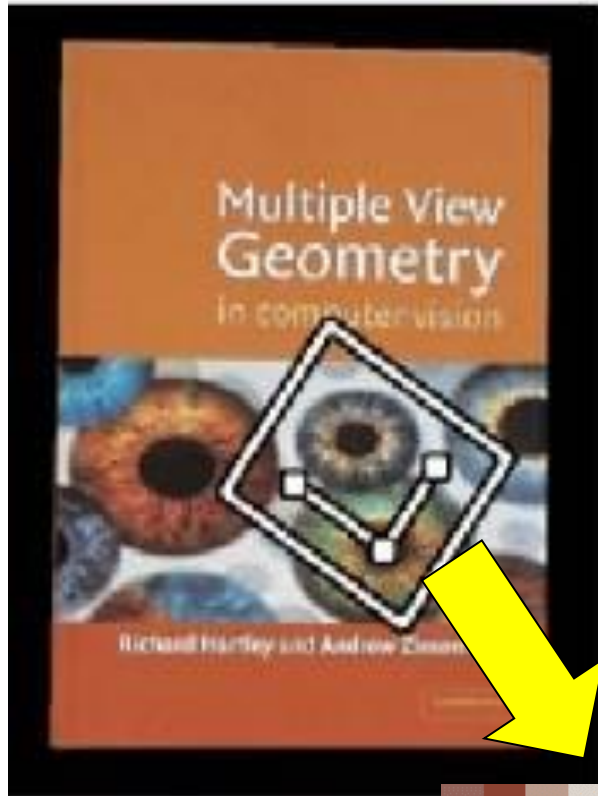
e.g. scale, translation, rotation

Photometric transformations

Scale-Invariant Feature Transform (SIFT) descriptor

Journal + conference versions: 66,498 citations

Image gradients

Keypoint descriptor

Histogram of oriented gradients

- **Captures important texture information**

- **Robust to small translations / affine deformations**
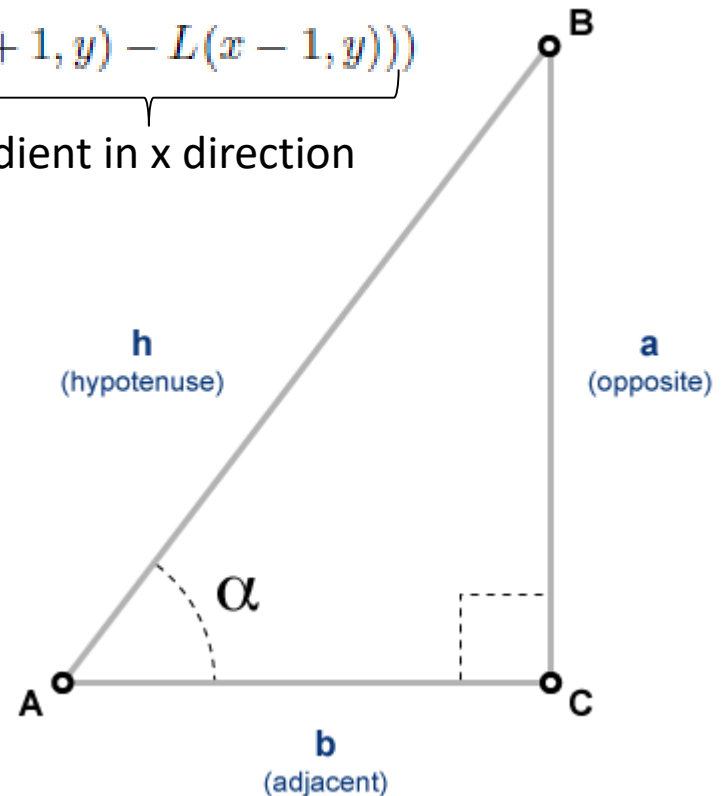
[Lowe, ICCV 1999]

Computing gradients

L = the image intensity

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

gradient in x direction   gradient in y direction

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

gradient in y direction   gradient in x direction
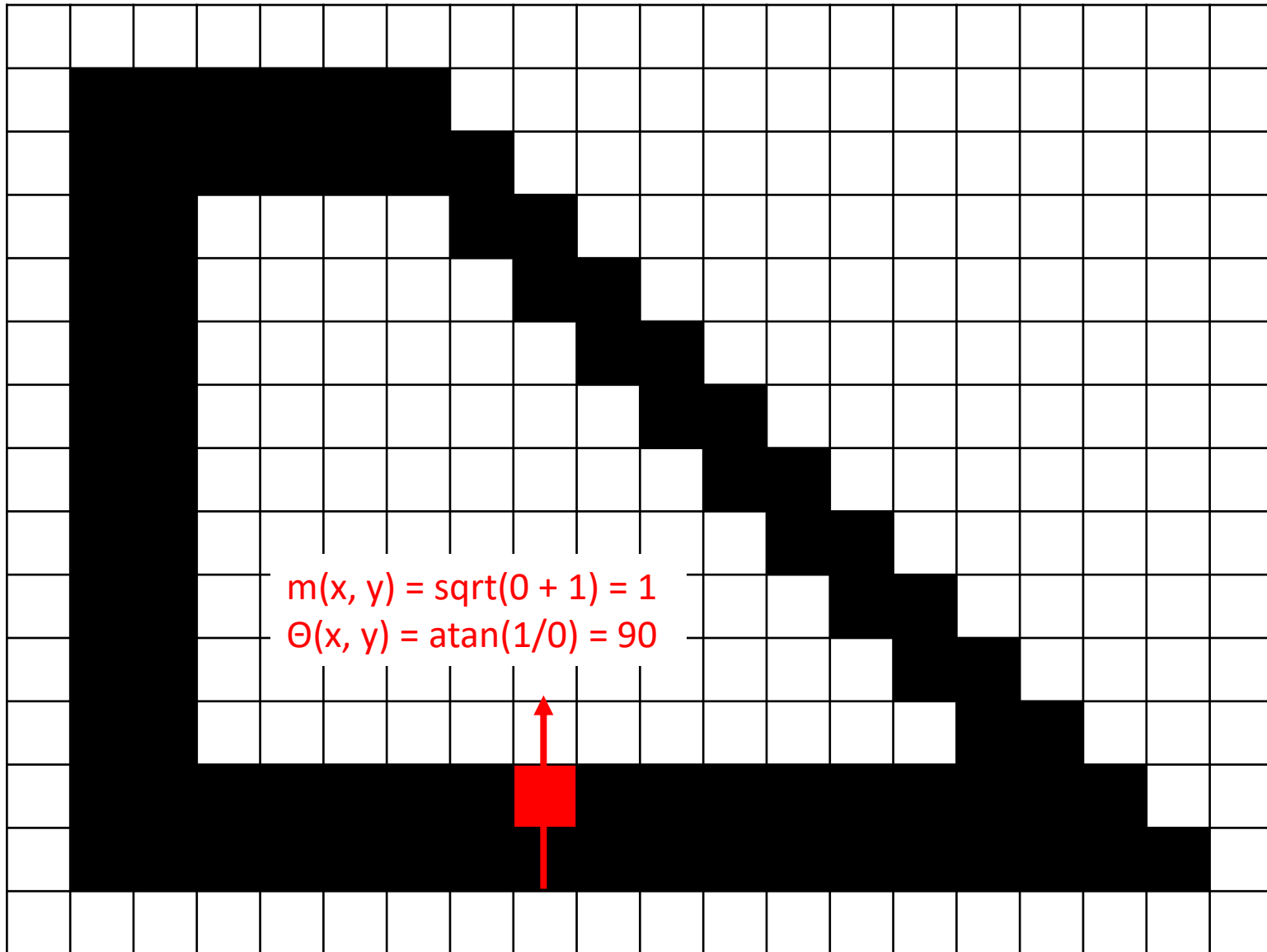
- $\tan(\alpha) = \dfrac{opposite\ side}{adjacent\ side}$

B

h
(hypotenuse)

a
(opposite)

α

A

C

b
(adjacent)

Gradients

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

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

m(x, y) = sqrt(1 + 0) = 1

Θ(x, y) = atan(0/-1) = 0

Gradients

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

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

m(x, y) = sqrt(0 + 1) = 1

Θ(x, y) = atan(1/0) = 90

Gradients

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

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

(note length / magnitude)

m(x, y) = sqrt(1 + 1) = 1.41

Θ(x, y) = atan(1/1) = 45
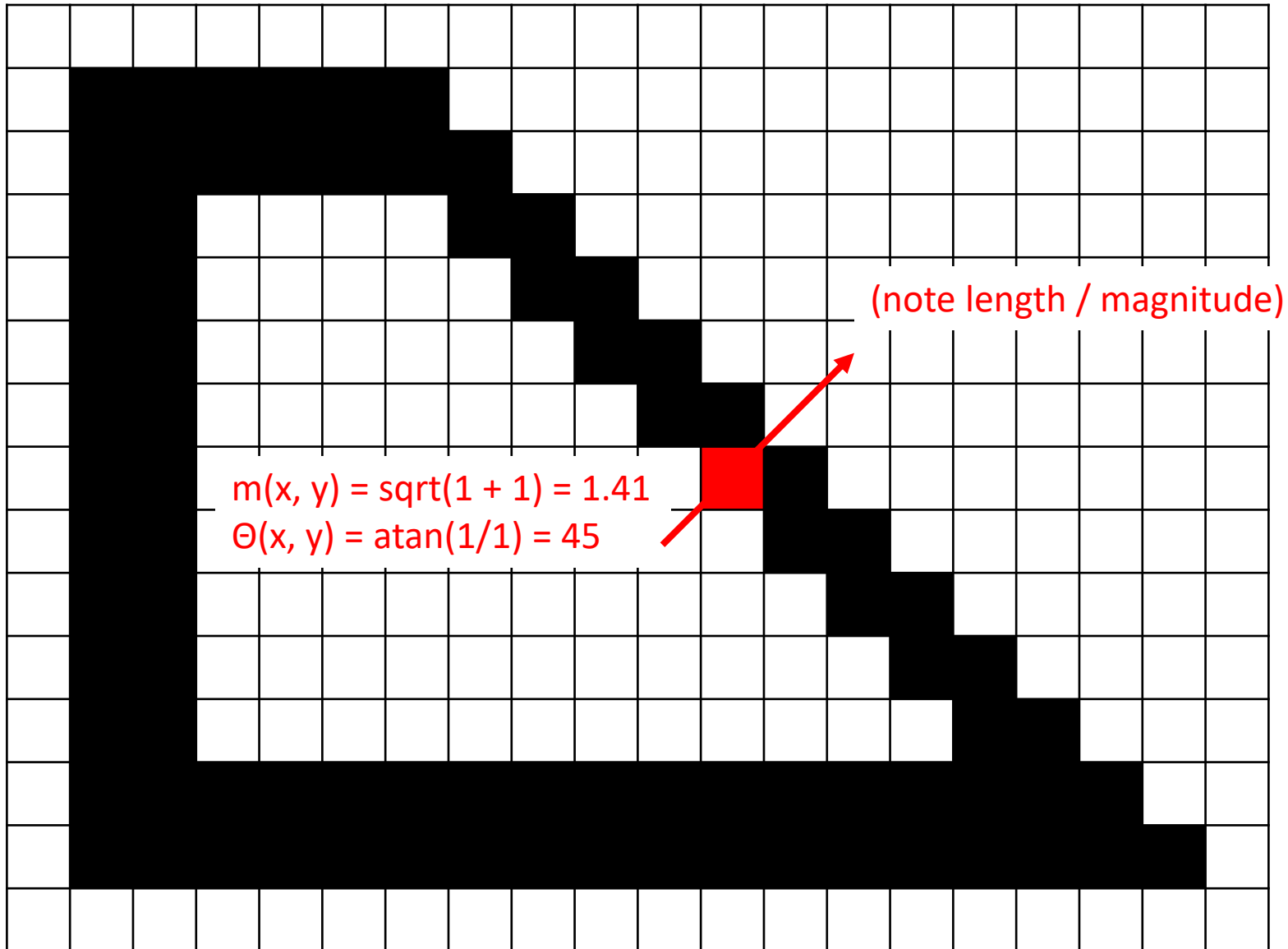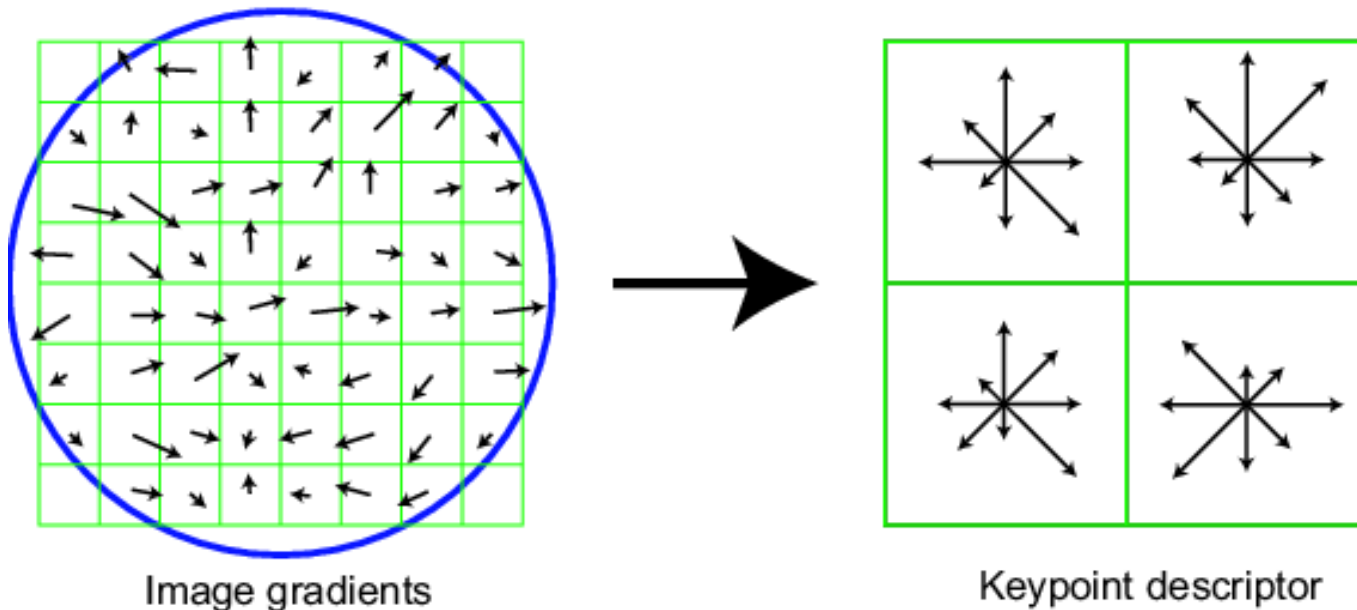
Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute gradient orientation for each pixel
- Create histogram over edge orientations weighted by magnitude
- That's your feature descriptor!

Image gradients

angle histogram

0   2π

Scale Invariant Feature Transform

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Quantize the gradient orientations i.e. snap each gradient to one of 8 angles
- Each gradient contributes not just 1, but magnitude(gradient) to the histogram, i.e. stronger gradients contribute more
- 16 cells * 8 orientations = 128 dimensional descriptor for each detected feature

Image gradients

Keypoint descriptor

Scale Invariant Feature Transform

Uniform weight (ignore magnitude)

Gradients

Histogram of gradients

Scale Invariant Feature Transform

Weight contribution by magnitude (e.g. long = 1, short = 0.5)

Gradients

Histogram of gradients

Scale Invariant Feature Transform

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Quantize the gradient orientations i.e. snap each gradient to one of 8 angles
- Each gradient contributes not just 1, but magnitude(gradient) to the histogram, i.e. stronger gradients contribute more
- 16 cells * 8 orientations = 128 dimensional descriptor for each detected feature
- Normalize + clip (threshold normalize to 0.2) + normalize the descriptor
- After normalizing, we have:

$$\sum_i d_i = 1 \quad \text{such that: } d_i < 0.2$$
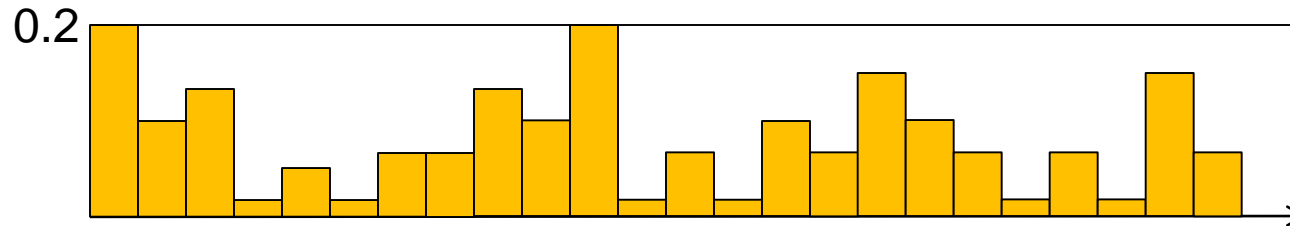
Making descriptor rotation invariant
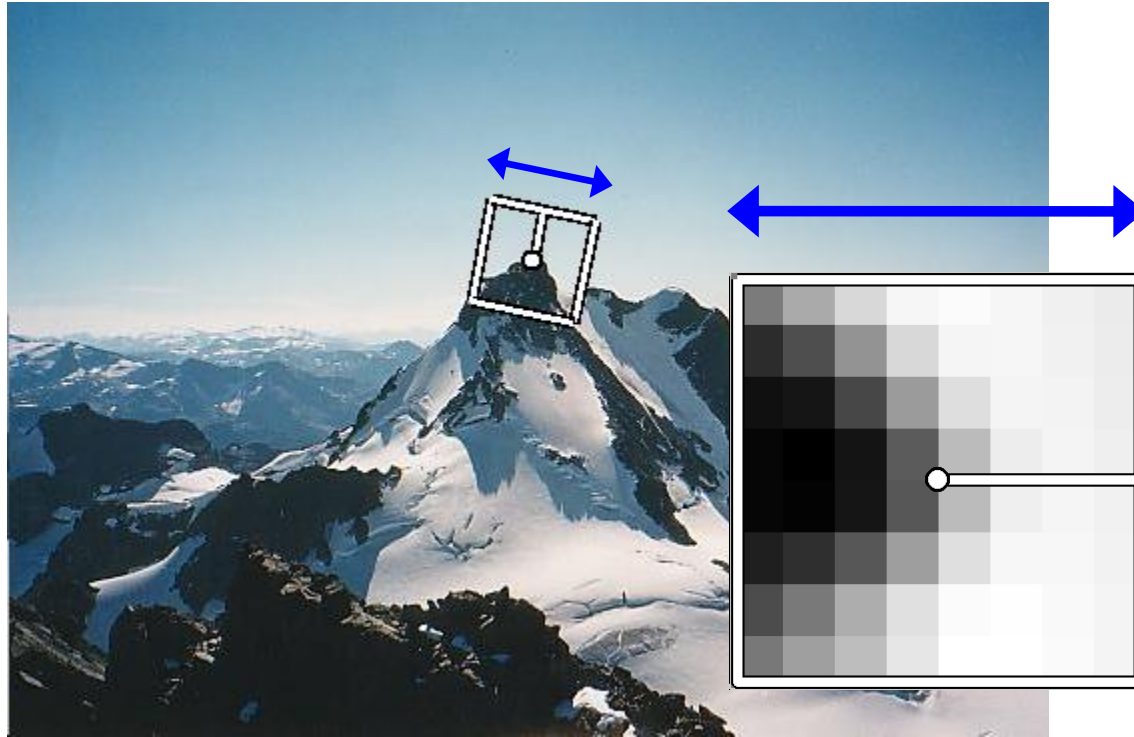
- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation

Image from Matthew Brown

SIFT is robust

- Can handle changes in viewpoint
  - 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

- Can be made to work without feature detection, resulting in "dense SIFT" (more points means robustness to occlusion)

- One commonly used implementation
  - http://www.vlfeat.org/overview/sift.html

Examples of using SIFT
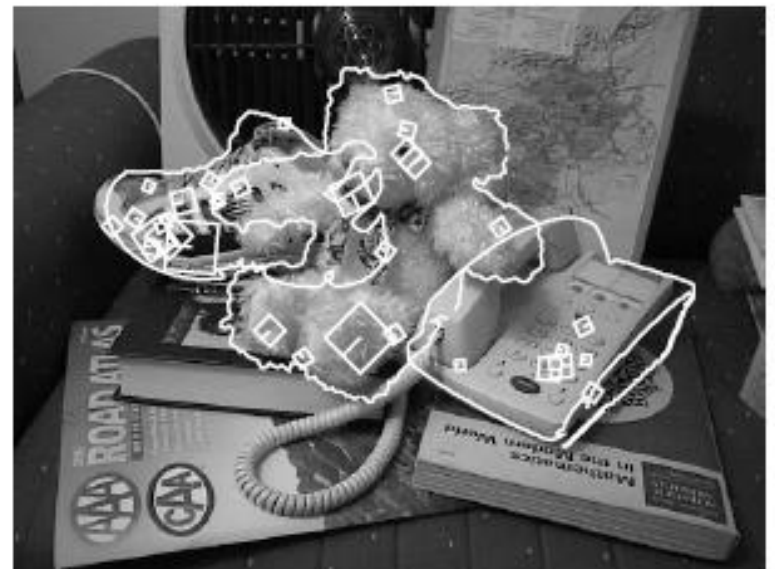
Examples of using SIFT
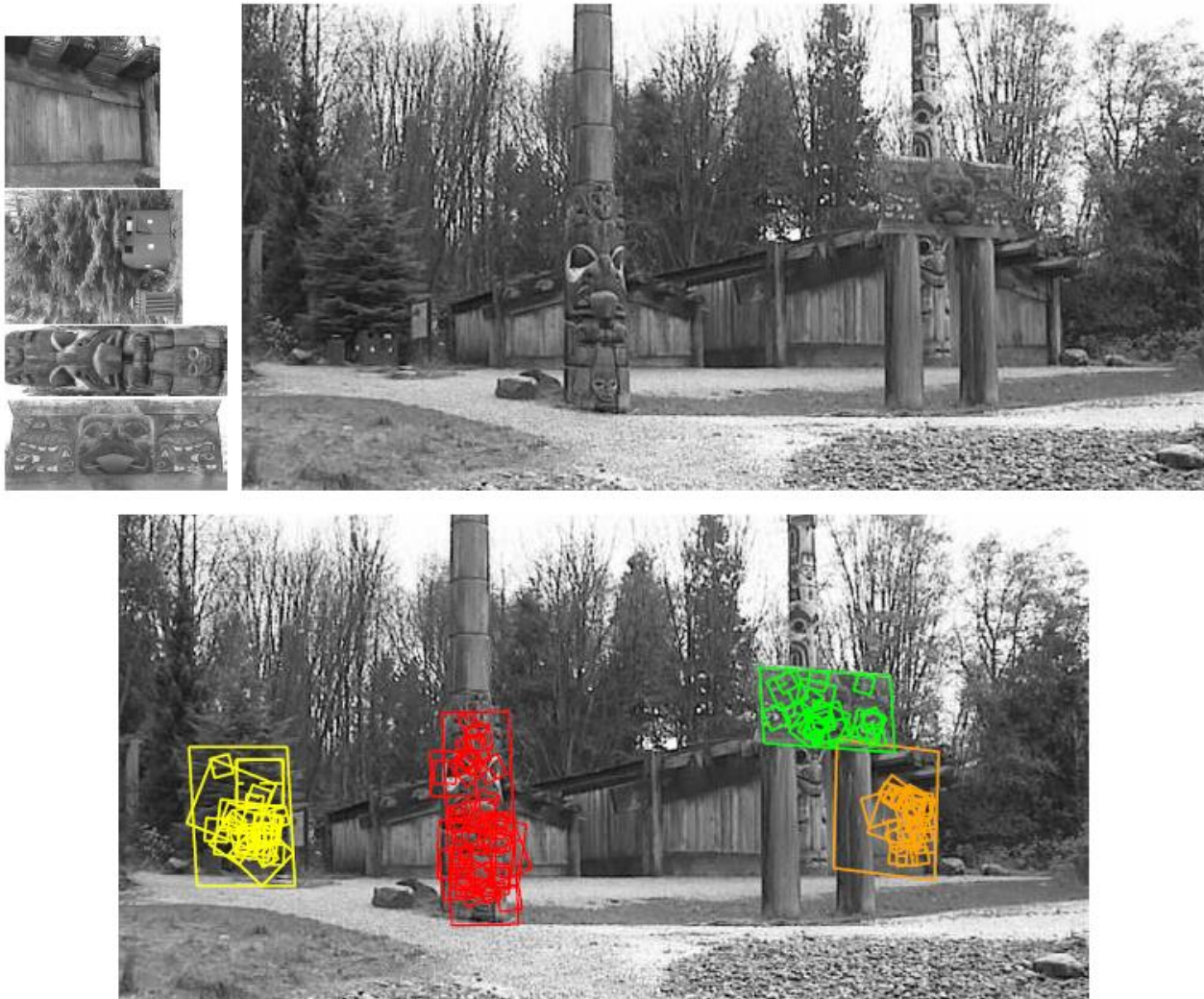
Examples of using SIFT

Applications of local invariant features

- Object recognition
- Indexing and retrieval
- Robot navigation
- 3D reconstruction
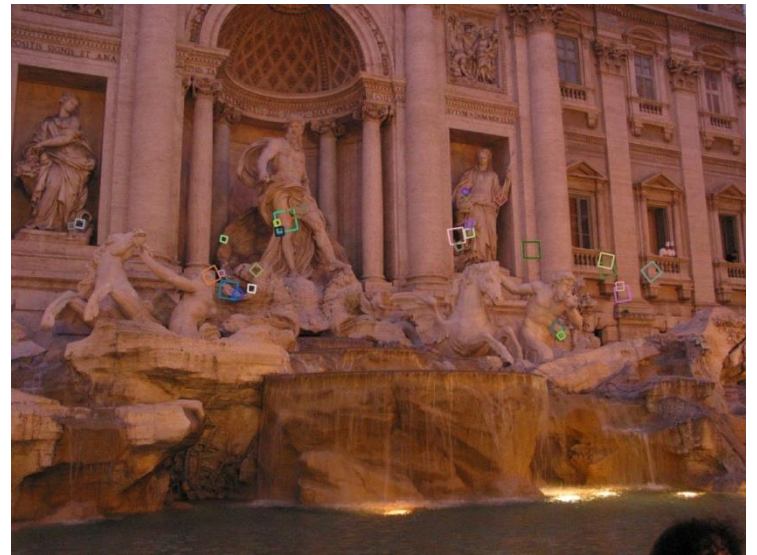- Motion tracking
- Image alignment
- Panoramas and mosaics
- …
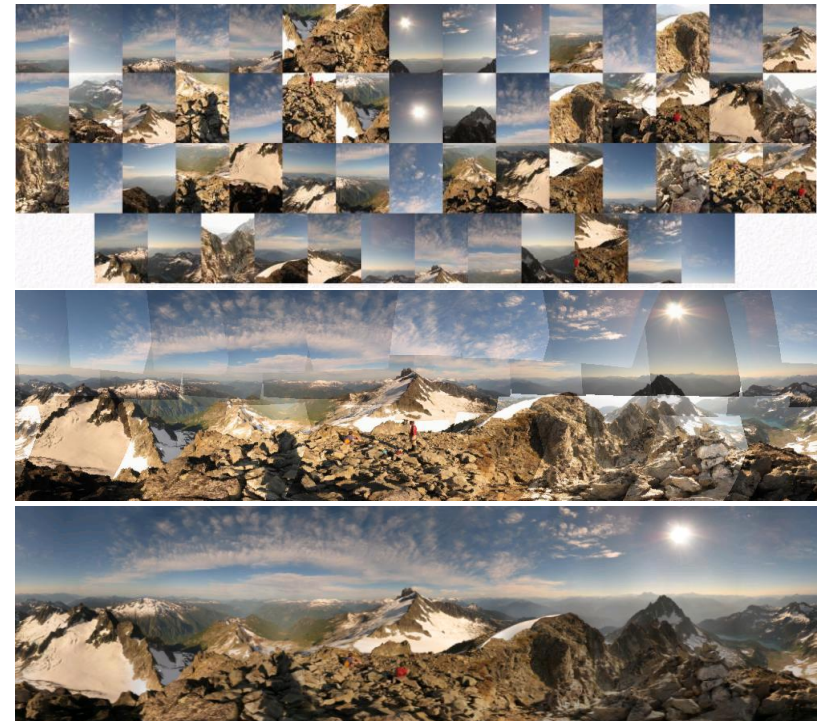
http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Plan for this lecture

- Feature detection / keypoint extraction
  - Corner detection
  - Properties
  - Blob detection
- Feature description (of detected features)
- Matching features across images

Matching local features

Image 1   Image 2
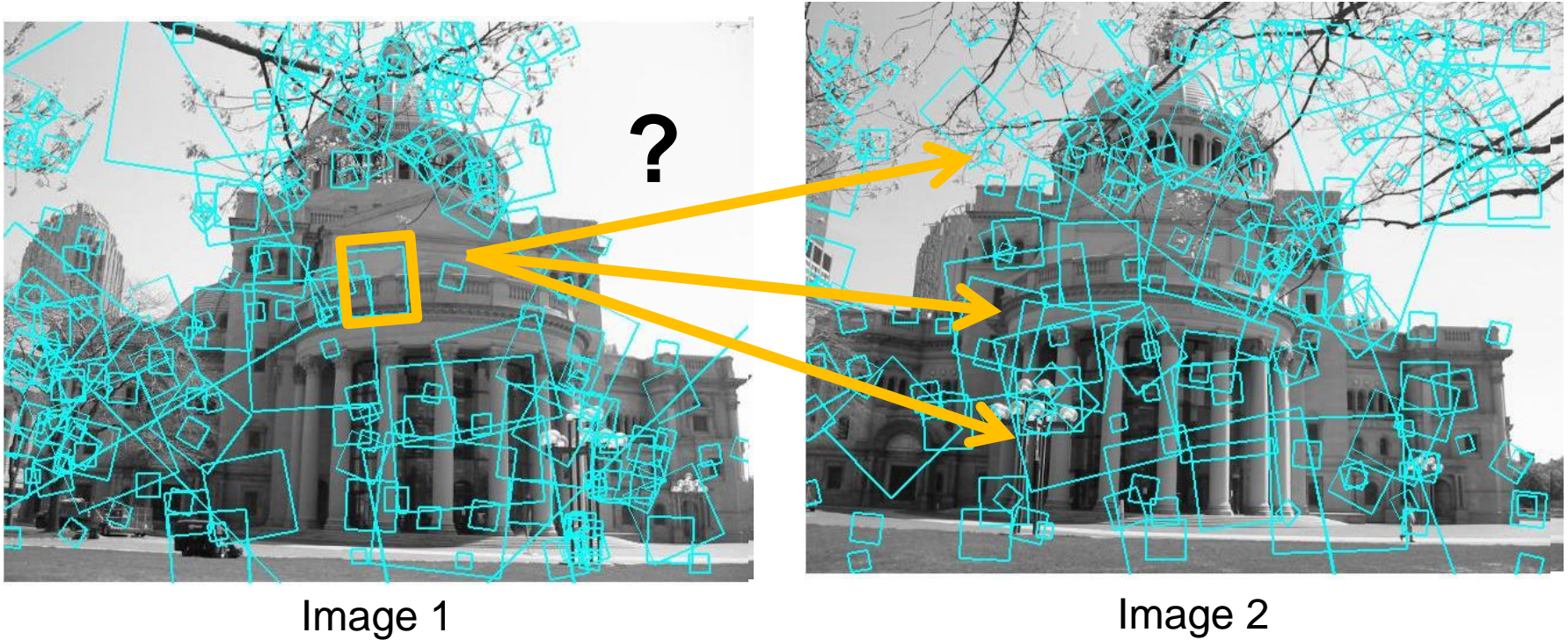
- To generate **candidate matches**, find patches that have the most similar appearance (e.g., lowest feature Euclidean distance)
- Simplest approach: compare query to all other features, take the closest (or closest k, or within a thresholded distance) as matches

Robust matching

Image 1

Image 2

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

Matching SIFT descriptors

- Nearest neighbor (Euclidean distance)
- Threshold ratio of 1st nearest to 2nd nearest descriptor


Lowe IJCV 2004
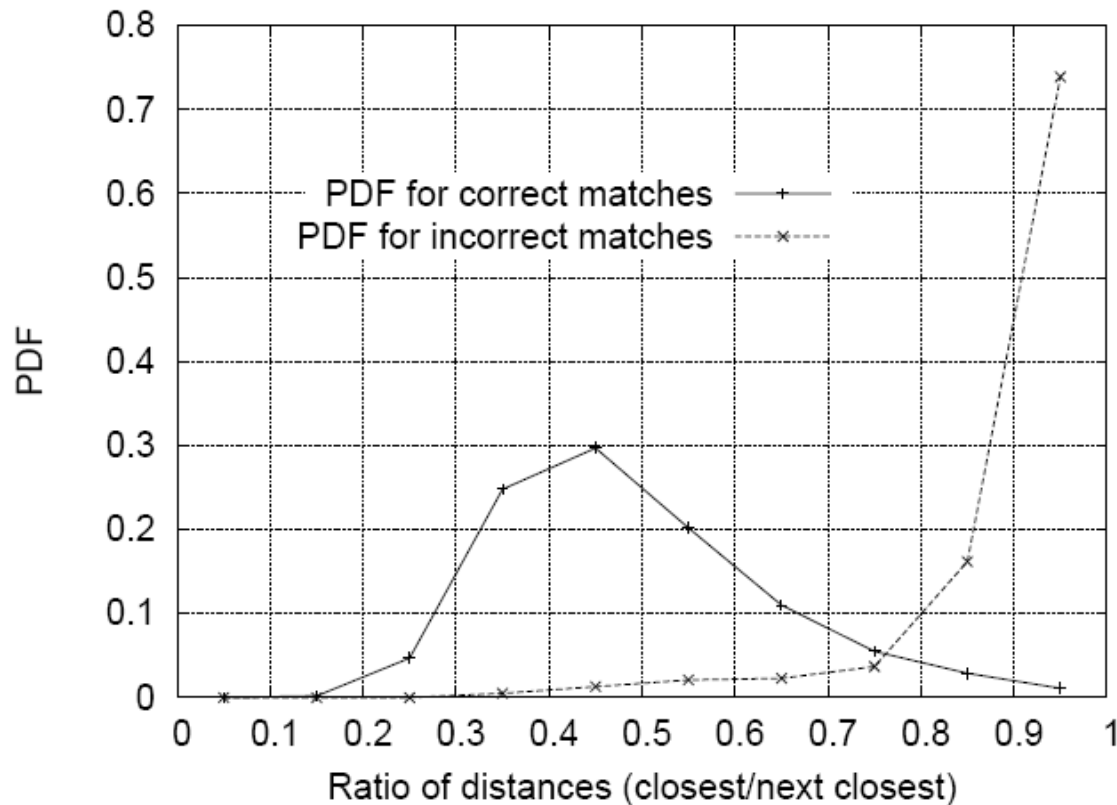
Efficient matching

- So far we discussed matching features across just two images

- What if you wanted to match a query feature from one image, to features from all frames in a video?

- Or an image to other images in a giant database?

- With potentially thousands of features per image, and hundreds to millions of images to search, how to efficiently find those that are relevant to a new image?
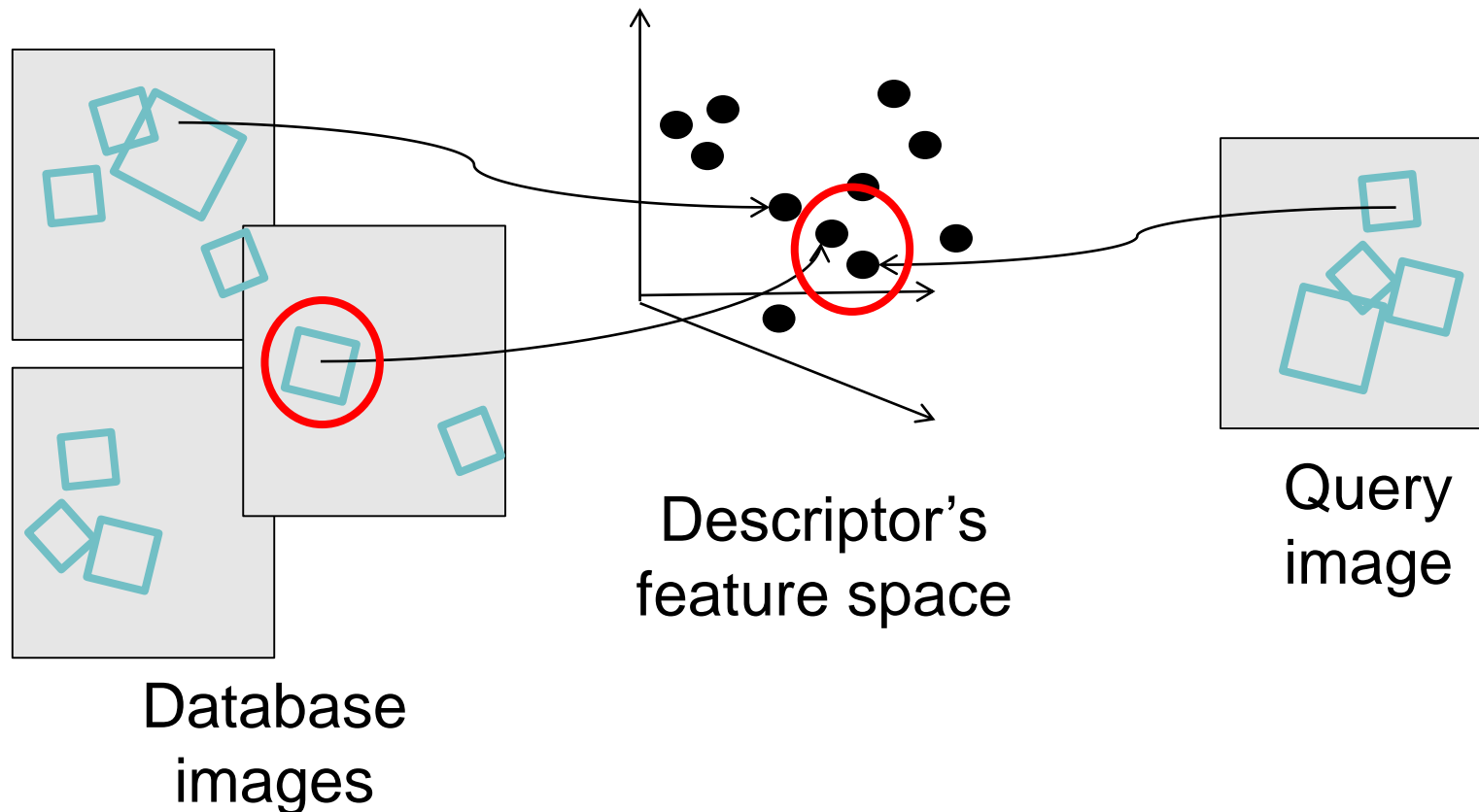
Indexing local features: Setup

- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)

Descriptor's
feature space

Database
images

K. Grauman
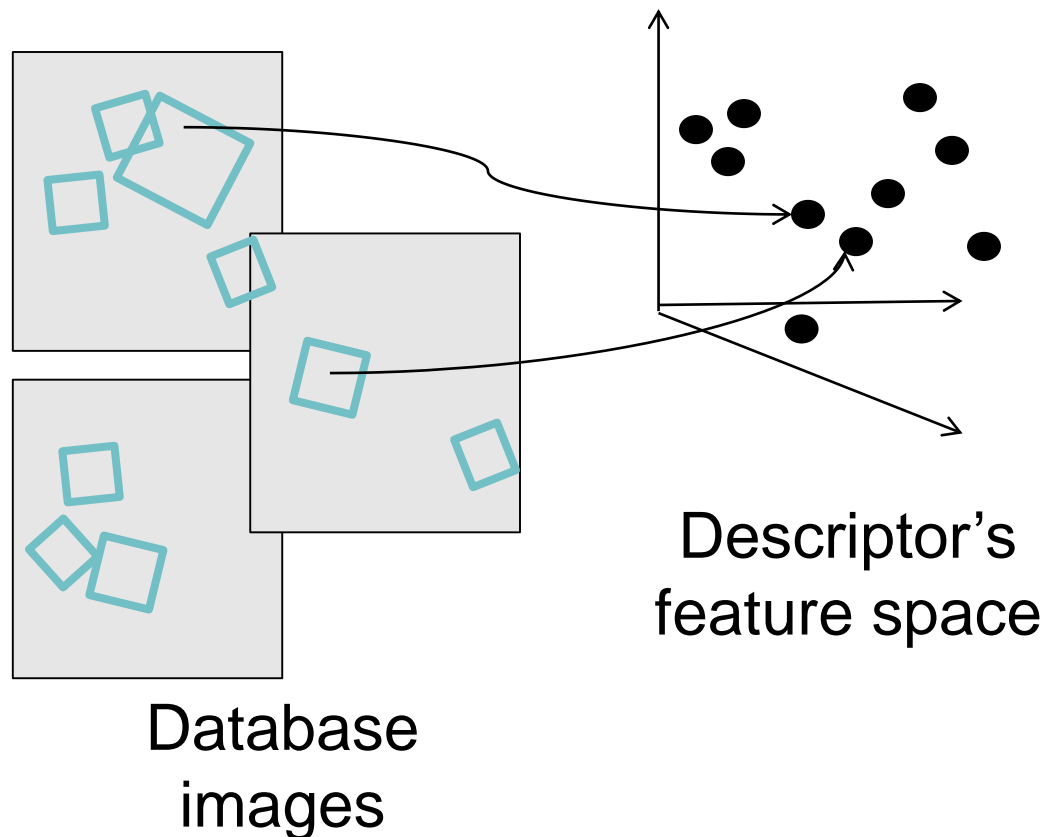
Indexing local features: Setup

- When we see close points in feature space, we have similar descriptors, which indicates similar local content

Database images

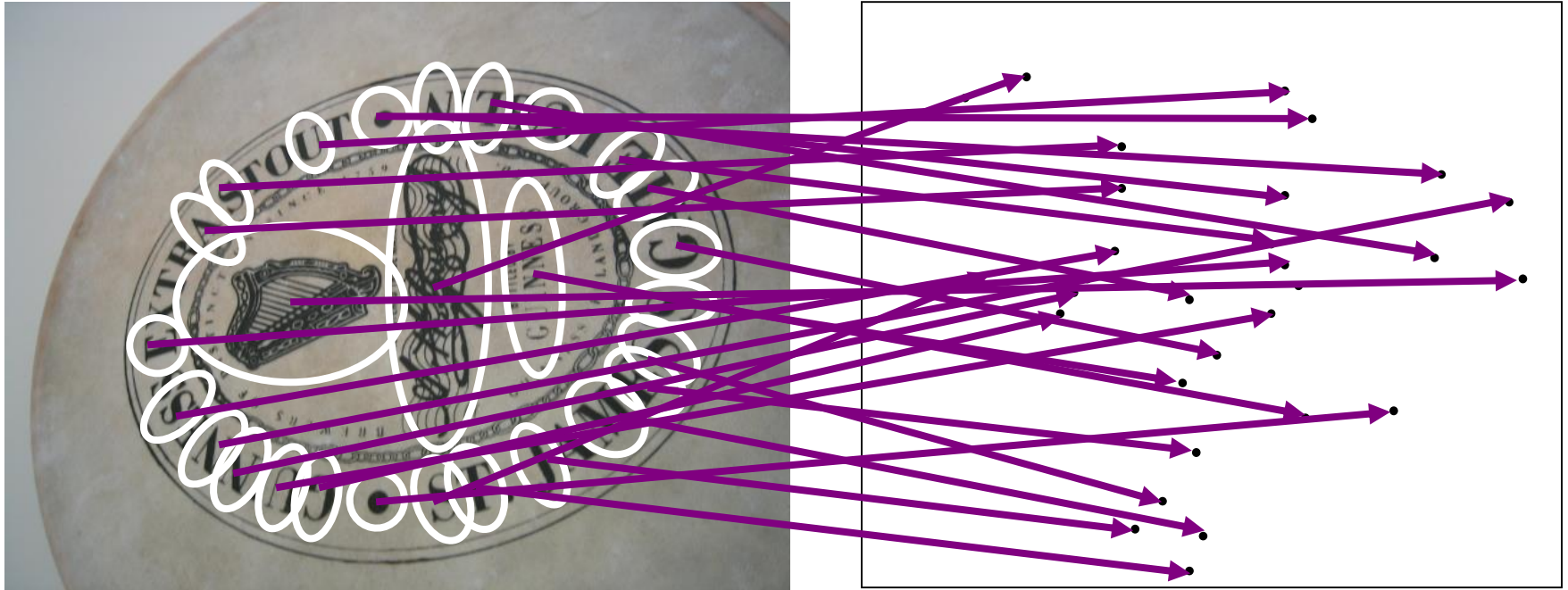Descriptor's feature space

Query image

K. Grauman

Indexing local features: Inverted file index

Index

- For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index…

- We want to find all *images* in which a *feature* occurs.

- To use this idea, we'll need to map our features to "visual words".

K. Grauman

Visual words: Main idea

- Extract some local features from a number of images …

e.g., SIFT descriptor space: each point is 128-dimensional

Visual words: Main idea
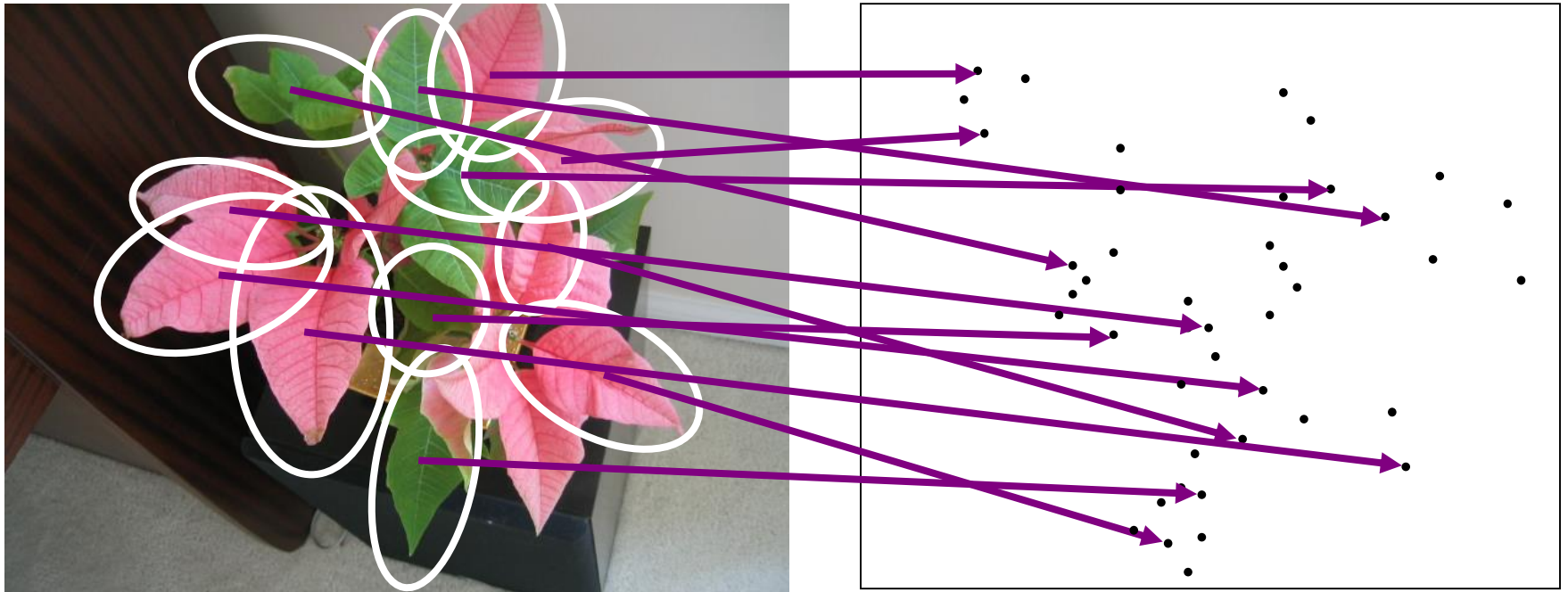
Visual words: Main idea
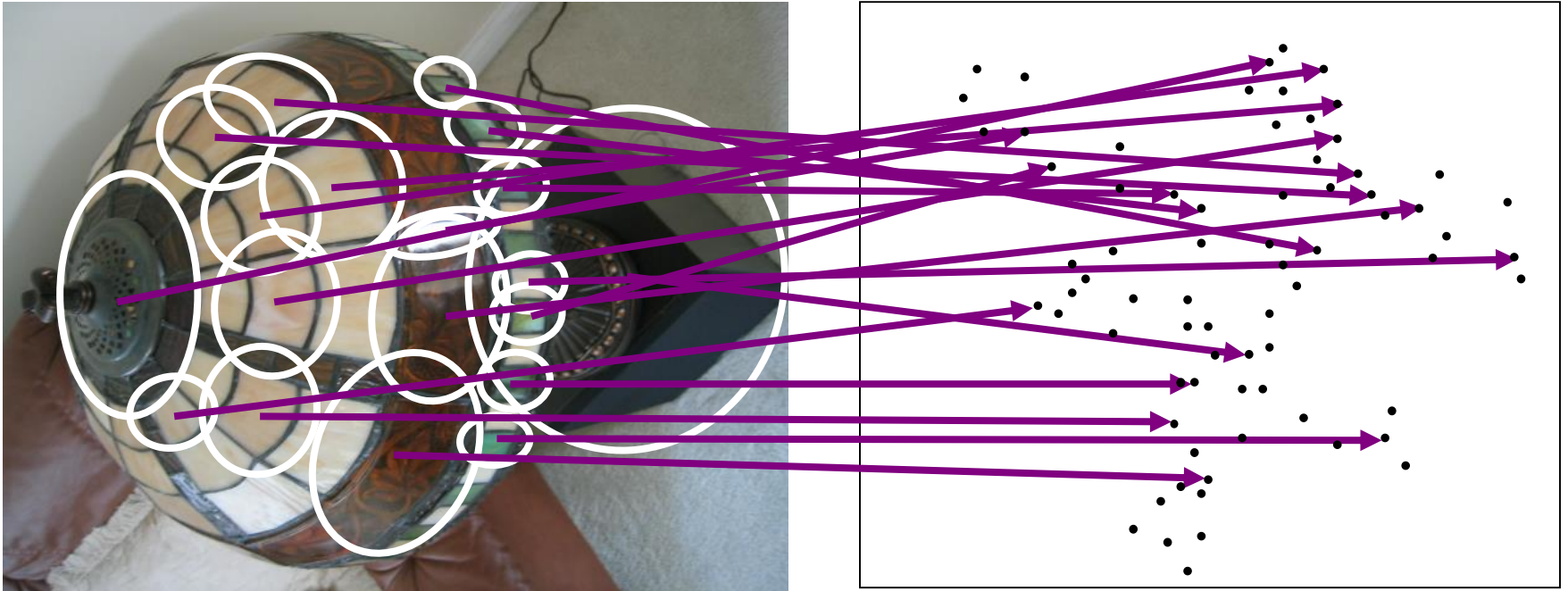
Visual words: Main idea
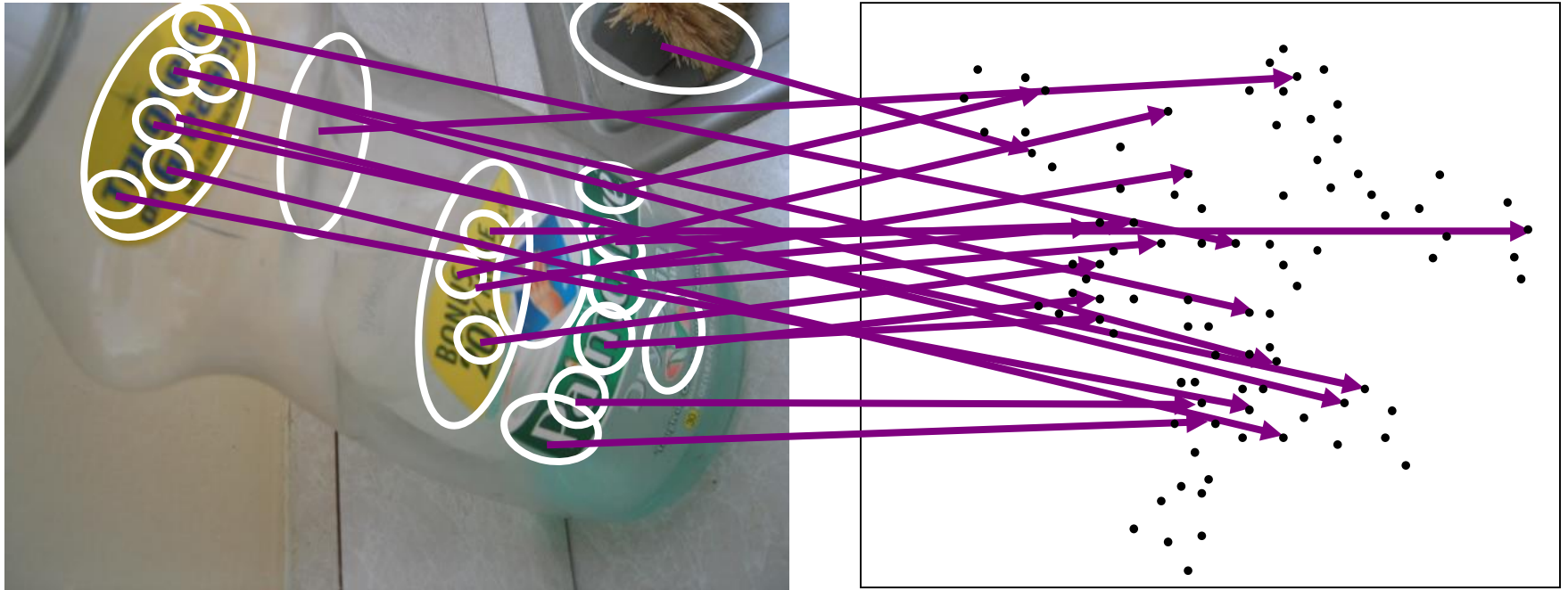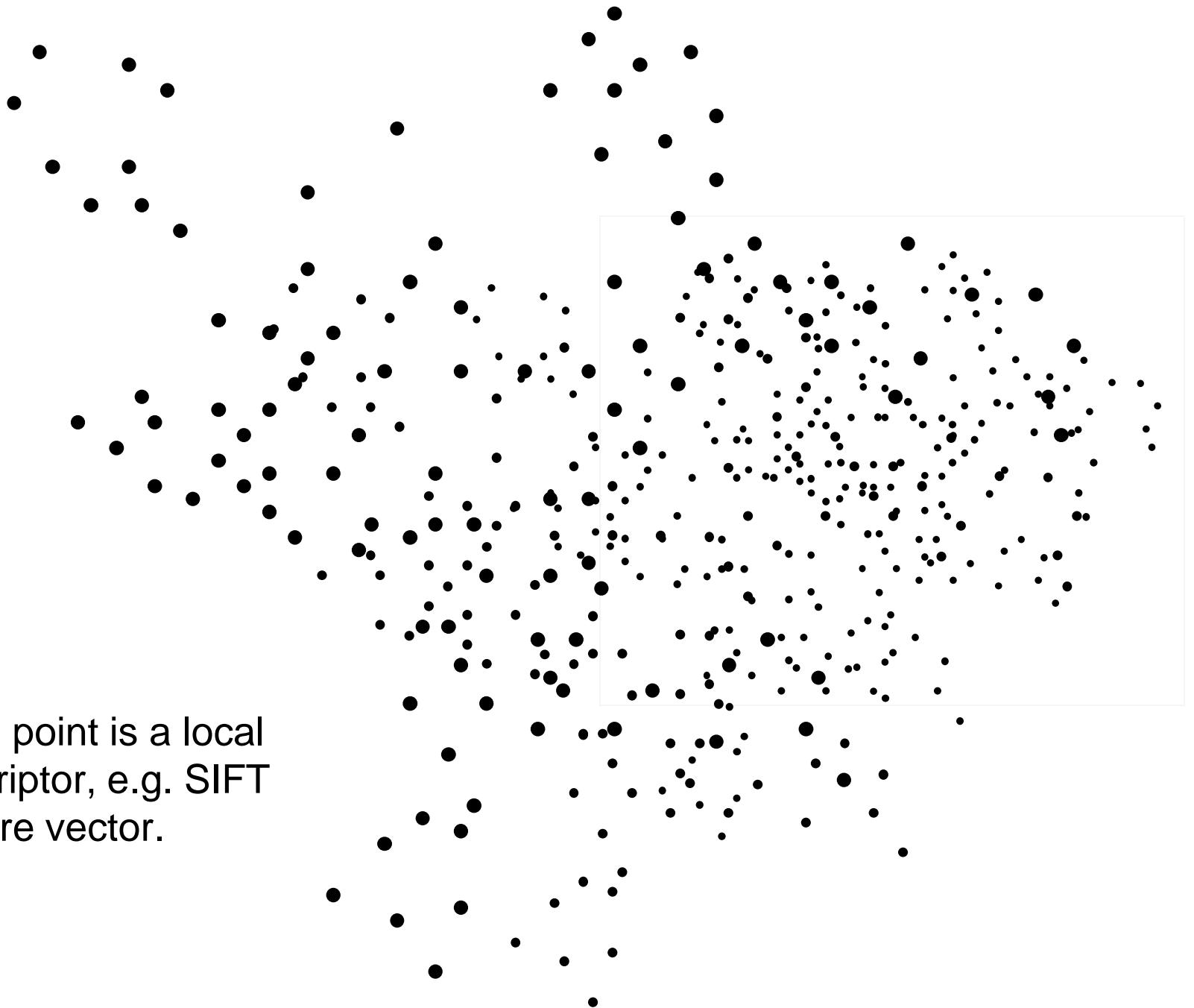
Each point is a local
descriptor, e.g. SIFT
feature vector.
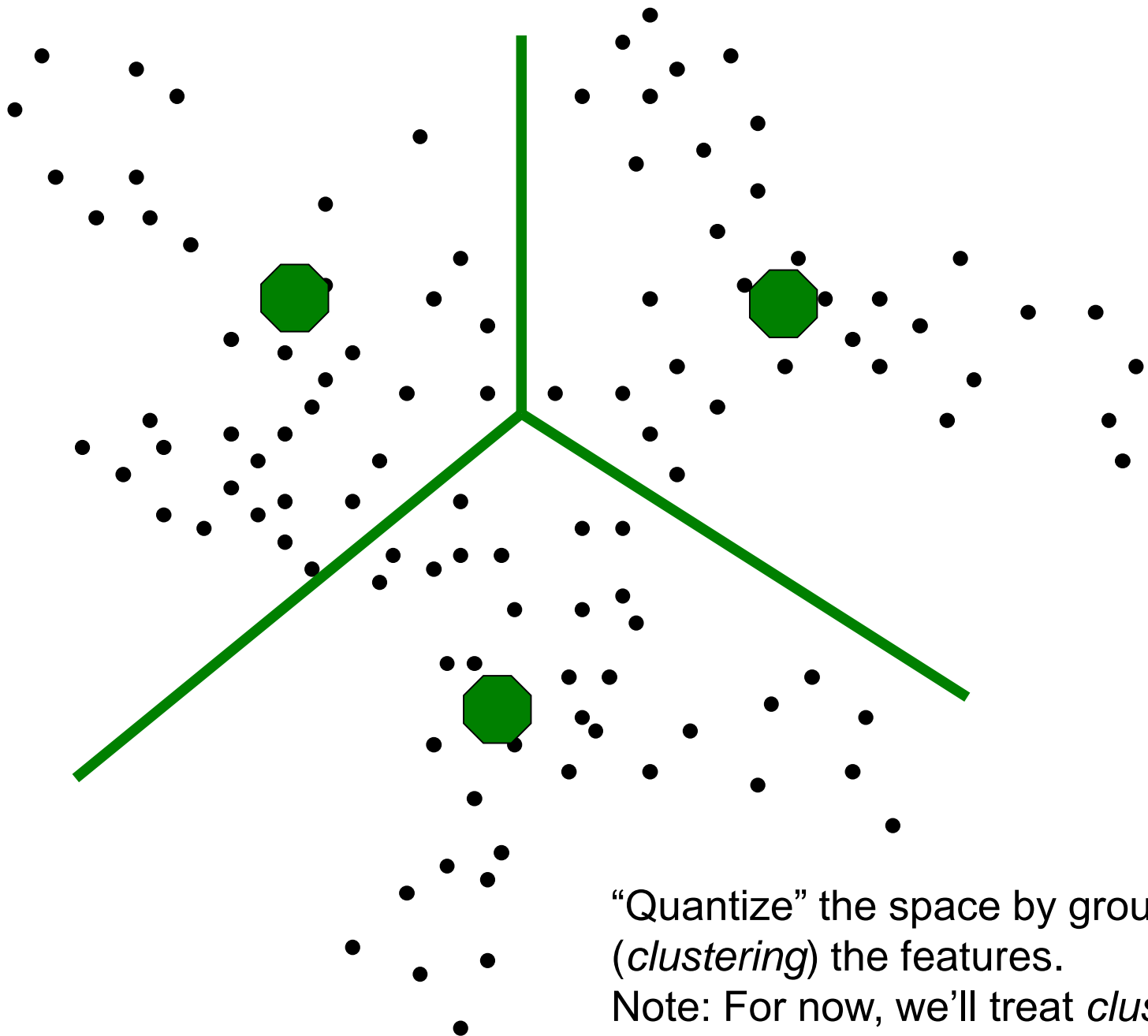
"Quantize" the space by grouping (*clustering*) the features.
Note: For now, we'll treat *clustering* as a black box.

Visual words

- Patches on the right = regions used to compute SIFT
- Each group of patches belongs to the same "visual word"
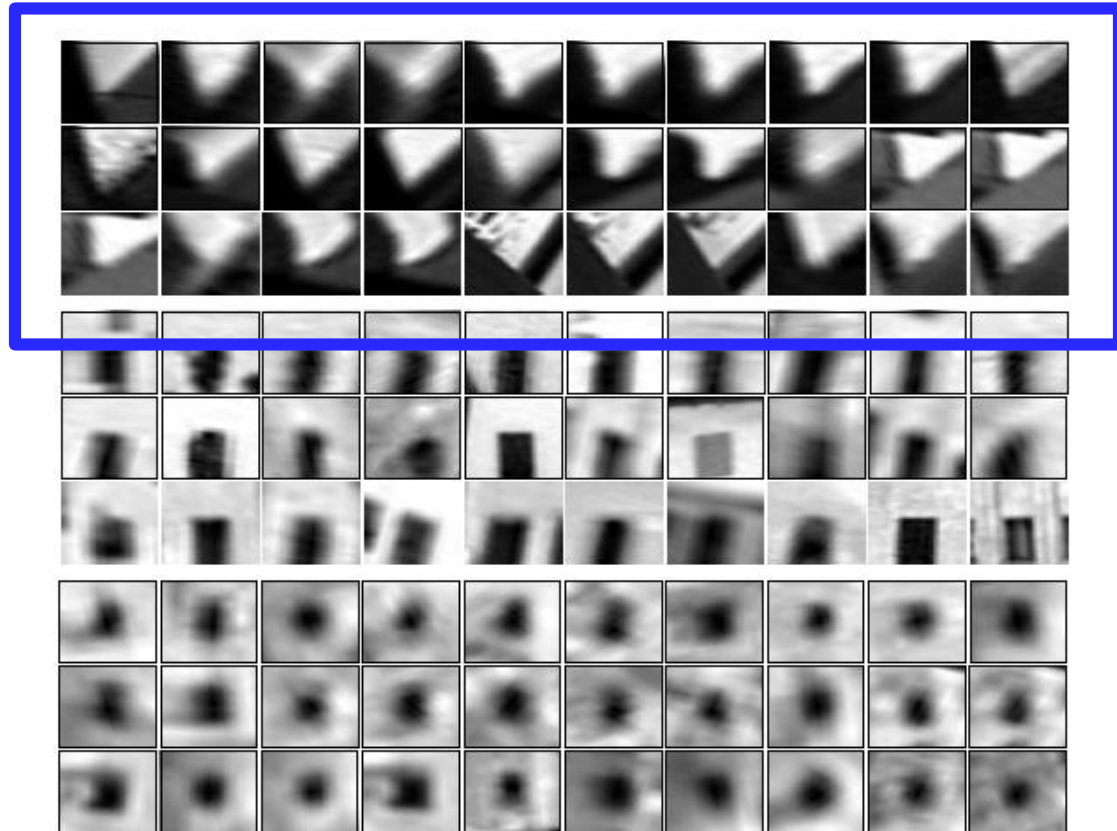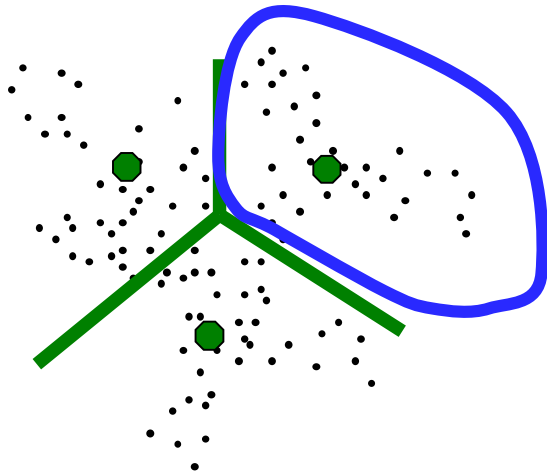
Figure from Sivic & Zisserman, ICCV 2003

Visual words for indexing

- Map high-dimensional descriptors to tokens/words by quantizing the feature space

1 **2**

Word #3

3

Query

Descriptor's feature space
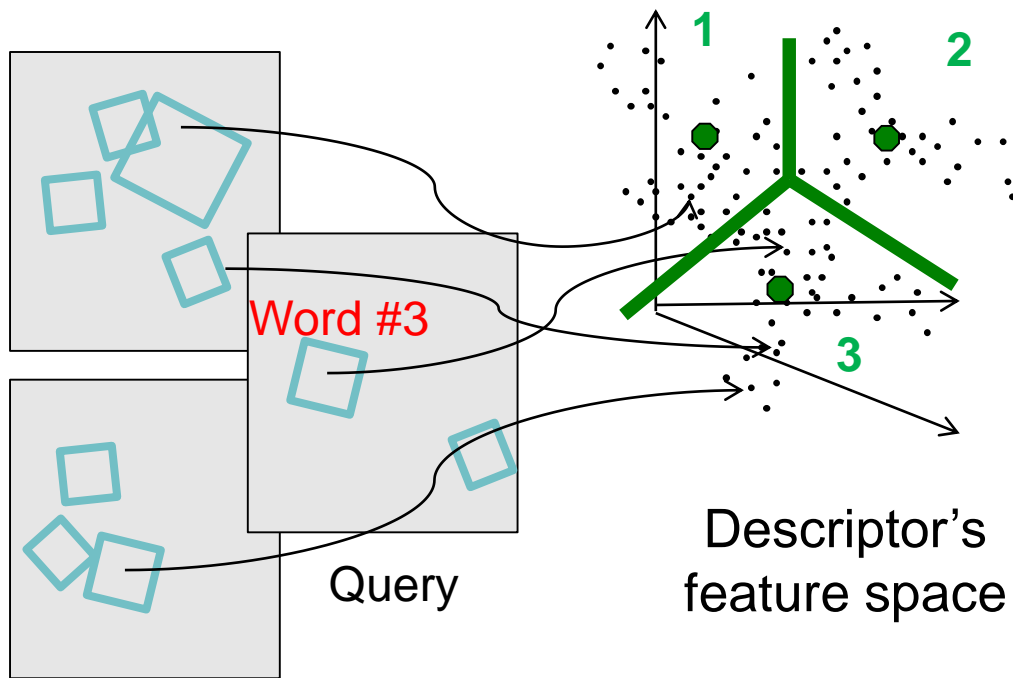
- Each cluster has a center

- To determine which word to assign to new image region (e.q. query), find closest cluster center

- *To compare features:* Only compare query feature to others in same cluster (speed up)

- *To compare images:* see next slide

Inverted file index

Word #	Image #
1	3
2 ...	
7	
8	
9	
10 ...	
91	

Image #1

Image #2

Image #3
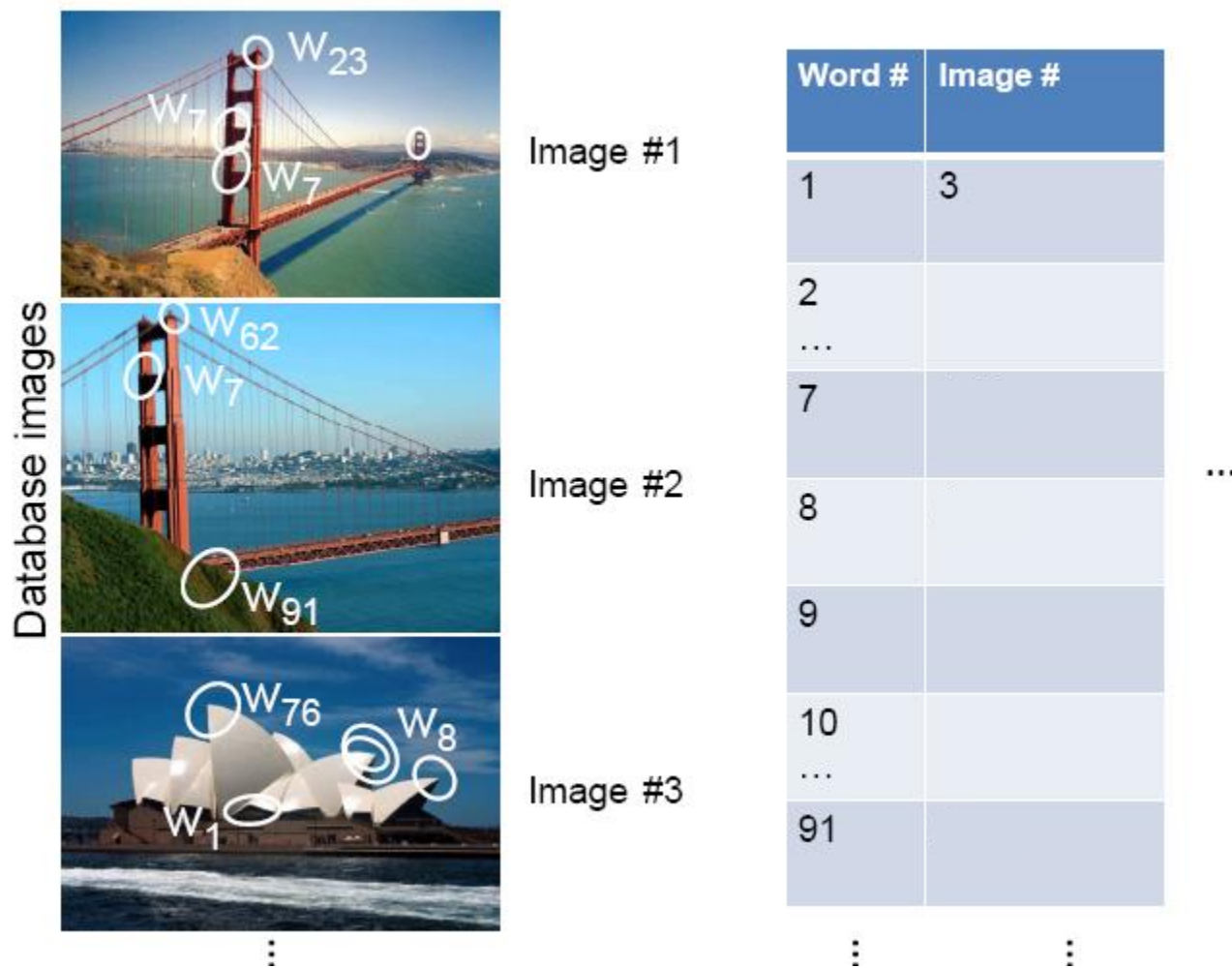
Database images

- Index database images: map each word to image IDs that contain it

K. Grauman

Inverted file index

When will this indexing process give us a gain in efficiency?

New query image

Word #	Image #
1	3
2	
7	1, 2
8	3
9	
10	
91	2

- For a new query image, find which database images share a word with it, and retrieve those images as matches (or inspect only those further)

How to describe documents with words?

Of all the sensory impressions proceeding to the brain, the visual experiences are the dominant ones. Our perception of the world around us is based essentially on the messages that re_____ ___ __m our eyes. For a long ti__ ___ ___ ___ __ retinal image was ___ ___ ___ __sual centers i_ ___ ___ ___ ___ ___ s a movie s___ ___ ___ ___ image ___ discove__ know th__ perceptio__ more com___ following the__ ___ ___ ___ __th to the various c__ ___ __ th__ ___ __ortex, Hubel and Wiesel ha__ ___ __ demonstrate that the *message abo__ image falling on the retina undergoes __ __wise analysis in a system of nerve cell__ stored in columns. In this system each c__ has its specific function and is responsible a specific detail in the pattern of the retinal image.*

sensory, brain, visual, perception, retinal, cerebral cortex, eye, cell, optical nerve, image Hubel, Wiesel

China is forecasting a trade surplus of $90bn (£51bn) to $100bn this year, a threefold increase on 2004's $32bn. The Commerce Ministry said the surplus would be created by a predicted 30% __ ___ __ __ $750bn, compared wi__ ___ ___ ___ $660bn. T__ ___ annoy th__ China's __ deliber__ agrees__ yuan is __ govern__ also need__ demand so __ ___ __ country. China __ ___ th__ ___ __e yuan against the do__ ___ ___ __nd permitted it to trade within a narrow __ __but the US wants the yuan to be allowe__ __de freely. However, Beijing has made it cl__ __t it will take its time and tread carefully be__ allowing the yuan to rise further in value.

China, trade, surplus, commerce, exports, imports, US, yuan, bank, domestic, foreign, increase, trade, value

Describing images w/ visual words

- Summarize entire image based on its distribution (histogram) of word occurrences

- Analogous to bag of words representation commonly used for documents

Feature patches:
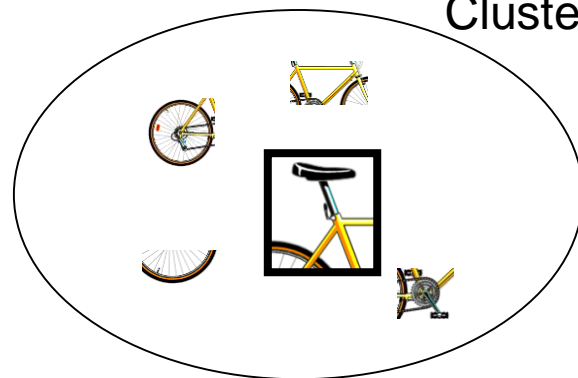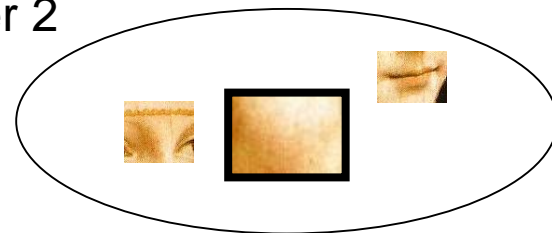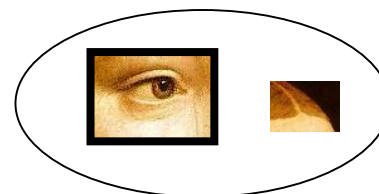

Cluster 1

Cluster 2

Cluster 3

Cluster 4
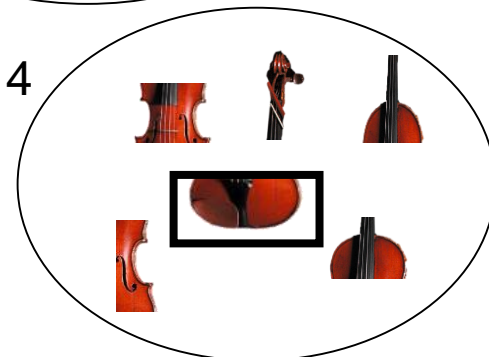
Describing images w/ visual words

- Summarize entire image based on its distribution (histogram) of word occurrences

- Analogous to bag of words representation commonly used for documents

Feature patches:

times appearing

times appearing

times appearing

Visual words

K. Grauman
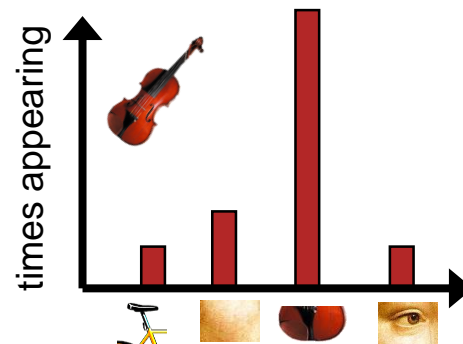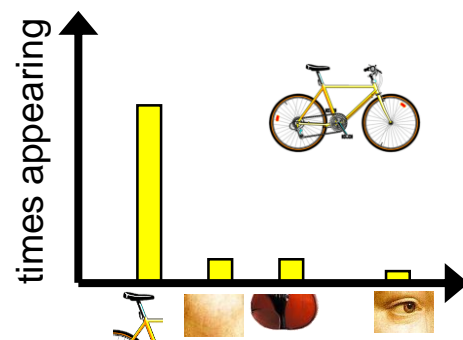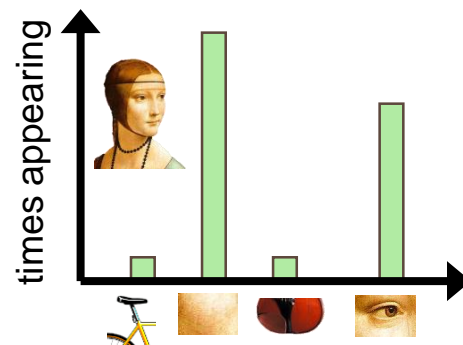
Comparing bags of words

- Rank images by normalized scalar product between their occurrence counts---*nearest neighbor* search for similar images.

[1 8   1   4]

[5 1   1   0]
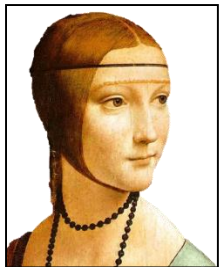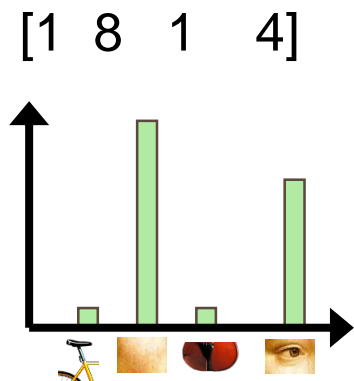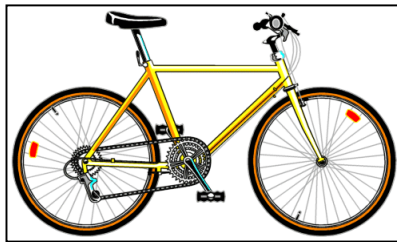
$$sim(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|}$$

$$= \frac{\sum_{i=1}^{V} d_j(i) * q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} * \sqrt{\sum_{i=1}^{V} q(i)^2}}$$

for vocabulary of *V* words

$$\vec{d}_j$$

$$\vec{q}$$

Bags of words: pros and cons

+ flexible to geometry / deformations / viewpoint

+ compact summary of image content

+ good results in practice

- basic model ignores geometry – must verify afterwards, or encode via features

- background and foreground mixed when bag covers whole image

- optimal vocabulary formation remains unclear

Summary: Inverted file index and bags of words similarity

Offline:

• Extract features in database images, cluster them to find words = cluster centers, make index

Online (during search):

1. Extract words in query (extract features and map each to closest cluster center)

2. Use inverted file index to find database images relevant to query

3. Rank database images by comparing word counts of query and database image

Additional references

- <u>Survey paper on local features</u>
  - "Local Invariant Feature Detectors: A Survey" by Tinne Tuytelaars and Krystian Mikolajczyk, in *Foundations and Trends in Computer Graphics and Vision* Vol. 3, No. 3 (2007) 177–280 (mostly Chapters 1, 3.2, 7) http://homes.esat.kuleuven.be/%7Etuytelaa/FT_survey_interestpoints08.pdf

- <u>Making Harris detection scale-invariant</u>
  - "Indexing based on scale invariant interest points" by Krystian Mikolajczyk and Cordelia Schmid, in ICCV 2001 https://hal.archives-ouvertes.fr/file/index/docid/548276/filename/mikolajcICCV2001.pdf

- <u>SIFT paper by David Lowe</u>
  - "Distinctive Image Features from Scale-Invariant Keypoints" by David G. Lowe, in IJCV 2004 http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

Summary

- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Laplacian of Gaussian, automatic scale selection

- Descriptors: robust and selective
  - Histograms for robustness to small shifts and translations (SIFT descriptor)

- Matching: cluster and index
  - Compare images through their feature distribution

Image gradients   Keypoint descriptor