

CS 1674: Intro to Computer Vision

Local Features: Detection, Description and Matching

Prof. Adriana Kovashka
University of Pittsburgh
February 1, 2022

Plan for this lecture

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

An image is a set of pixels...



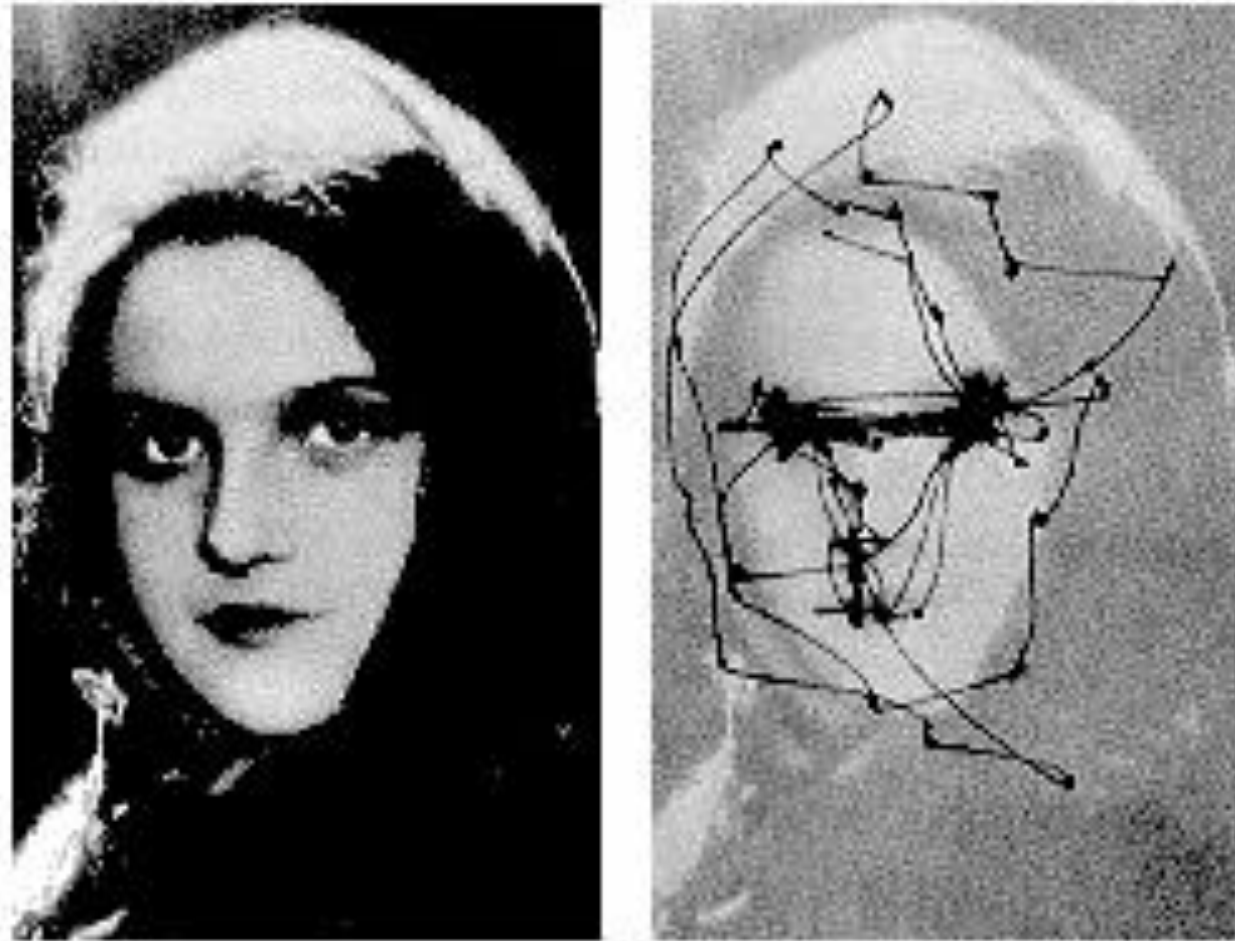
6	9	8
5	6	7
4	7	6
3	4	5
2	5	4
1	2	3



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



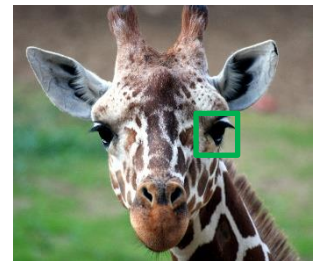
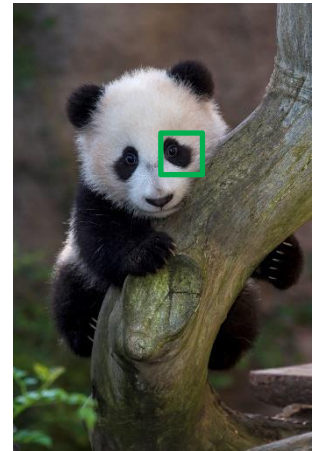
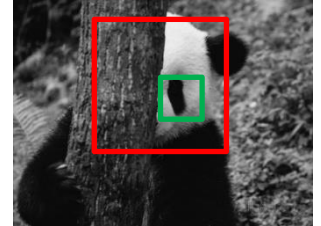
Yarbus eye tracking

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 use those to compute a representation of the whole image
- Local features usually exploit image gradients, ignore color
- Feature \sim *vector* of gradient statistics for a window with *particular location and size*

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 (panda to panda)
- Distinctiveness
 - Each feature has a distinctive description
 - Minimize wrong matches (panda to giraffe)
- Compactness and efficiency
 - Many fewer features than image pixels

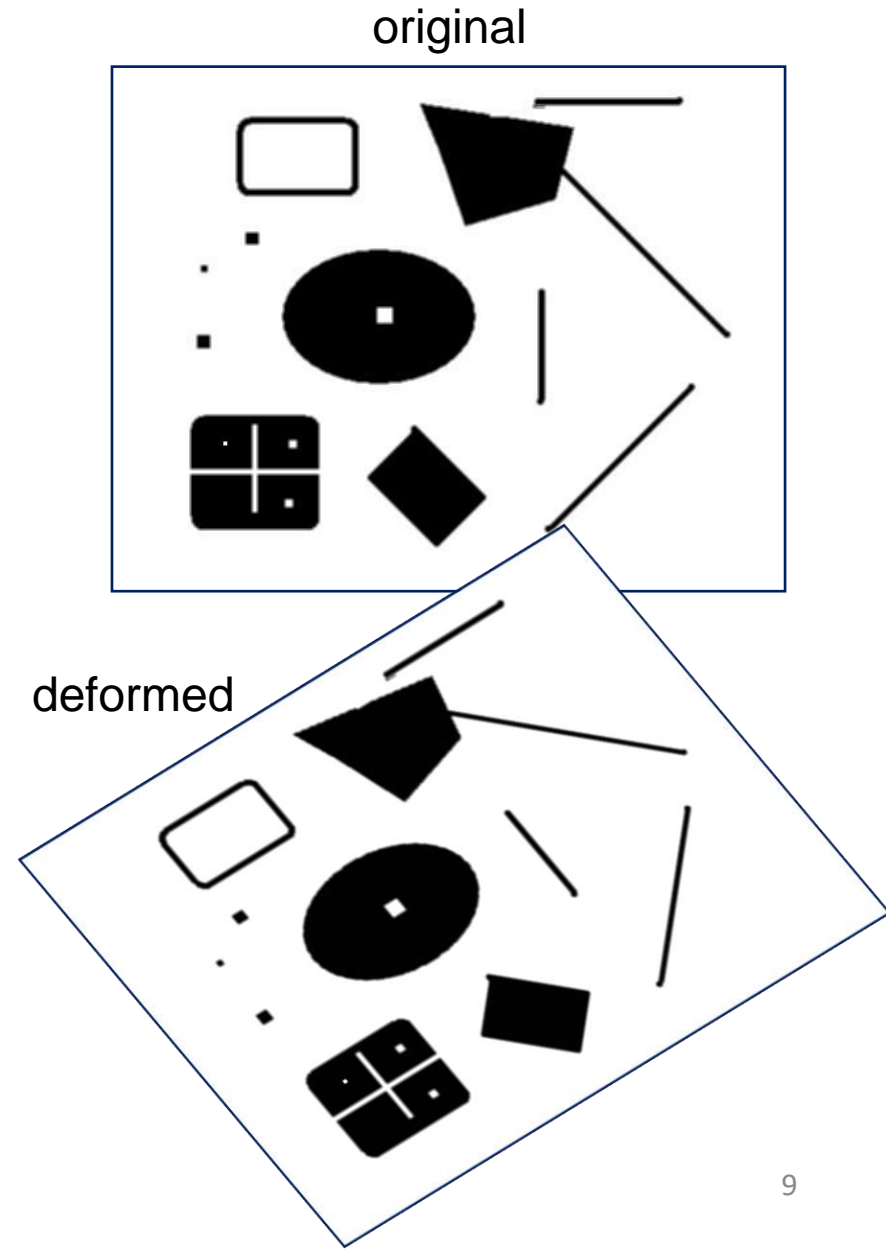


Interest(ing) points

- Note: “interest points” = “keypoints”, also sometimes called “features”
- Many applications
 - Recognition: which patches are likely to tell us something about the object category?
 - Image search: which points would allow us to match images between query and database?
 - 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?



Choosing interest points

Where would you
tell your friend to
meet you?

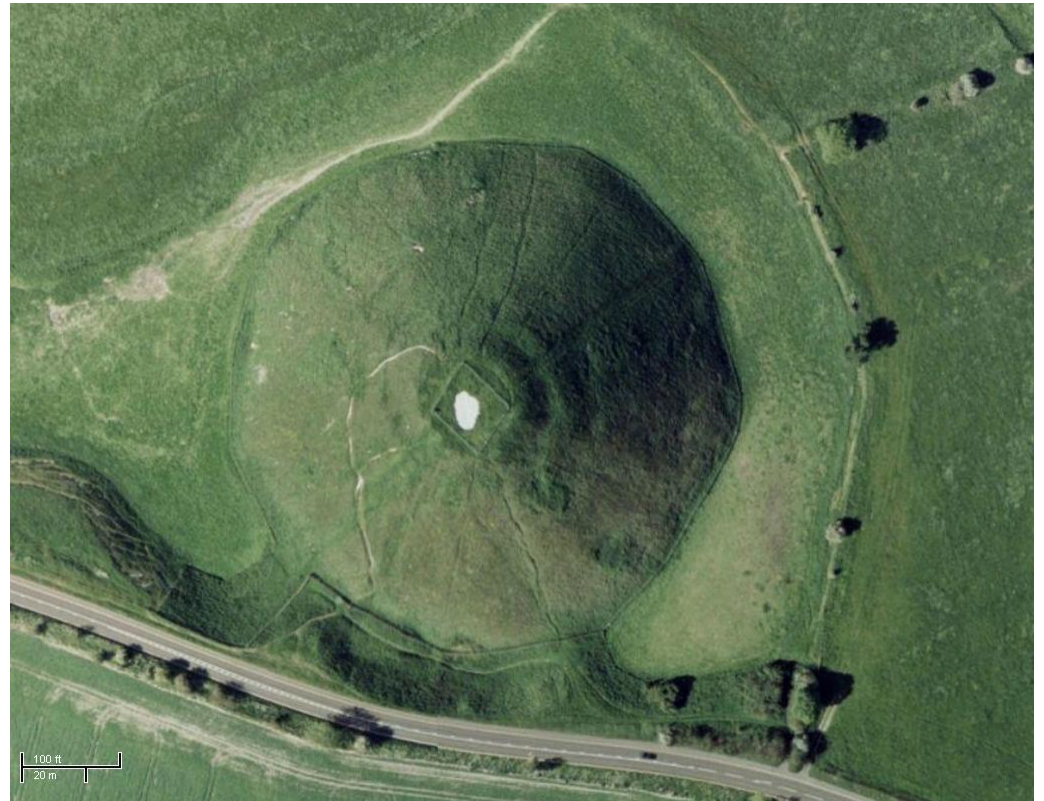
→ Corner detection



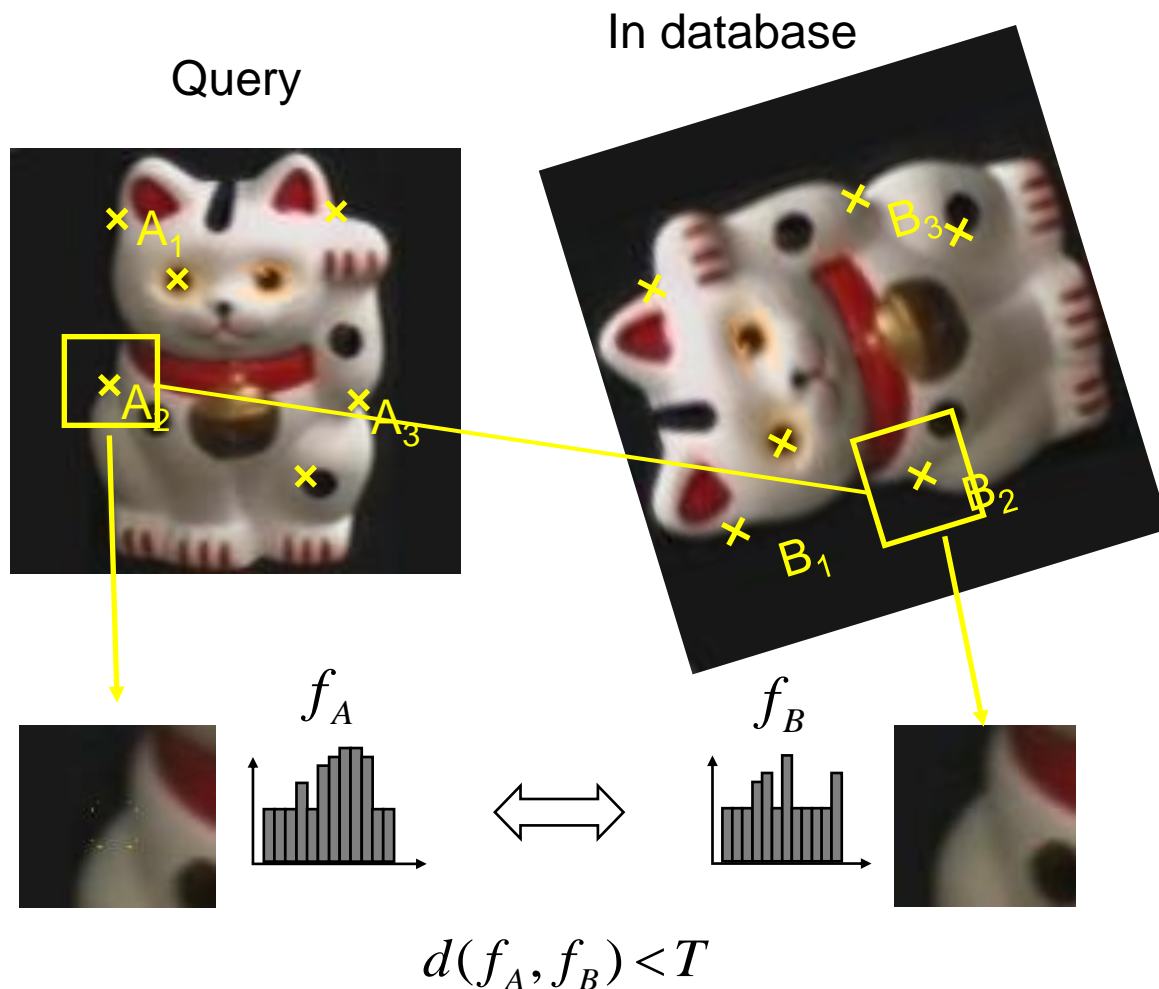
Choosing interest points

Where would you
tell your friend to
meet you?

→ Blob detection



Application 1: Keypoint Matching for Search



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

Application 1: Keypoint Matching For Search

Query



In database



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 content₁₃

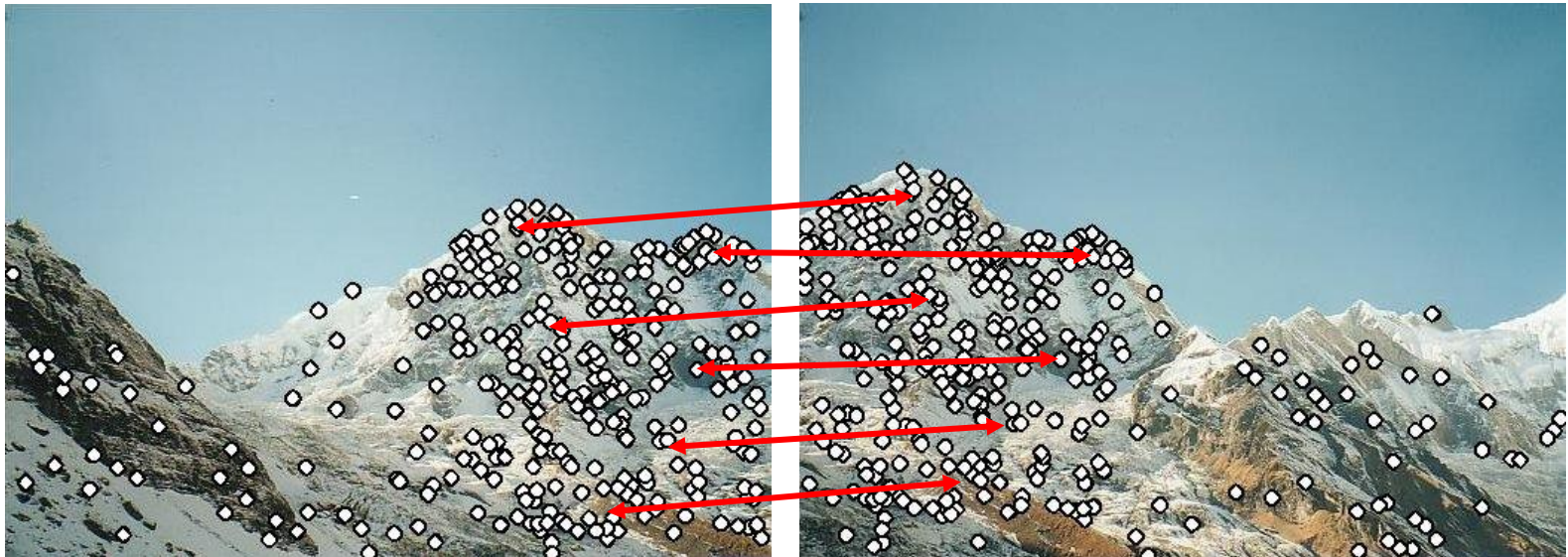
Application 2: Panorama stitching

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

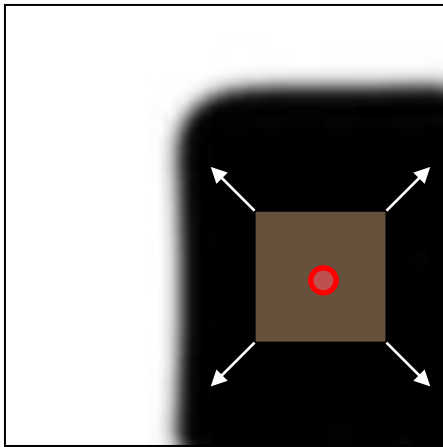
Step 2: match features

Step 3: align images

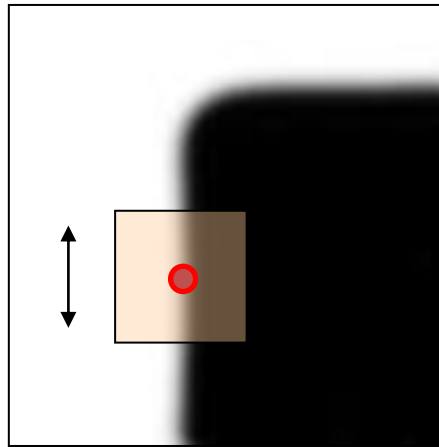
Corners as distinctive interest points

- 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

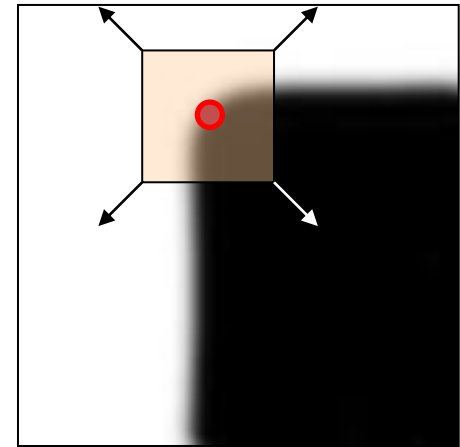
● Candidate keypoint



“flat” region:
no change in
all directions



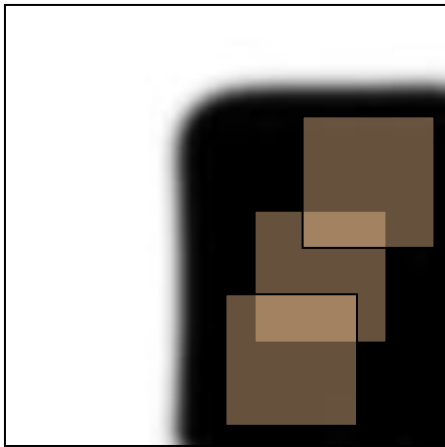
“edge”:
no change along
the edge direction



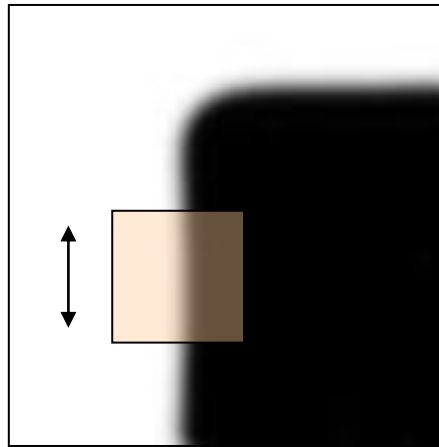
“corner”:
significant change
in all directions

Corners as distinctive interest points

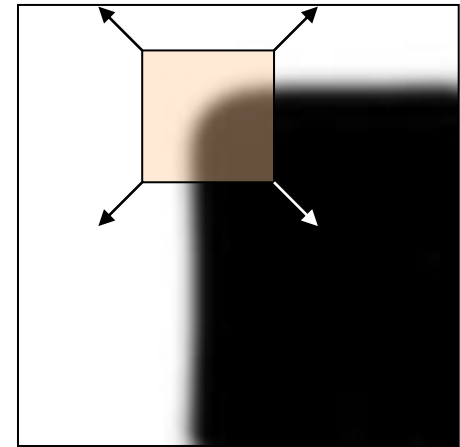
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



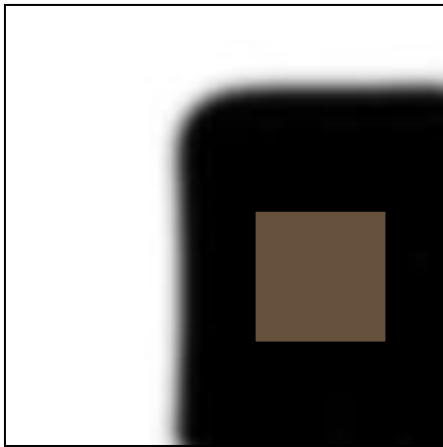
“edge”:
no change along
the edge direction



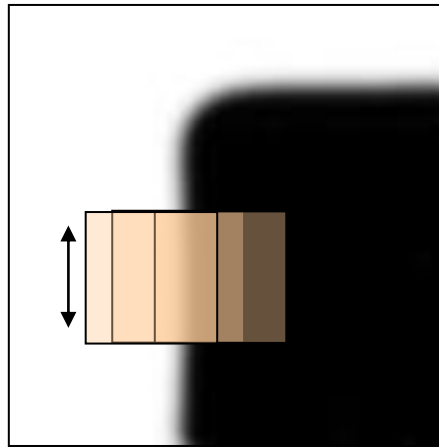
“corner”:
significant change
in all directions

Corners as distinctive interest points

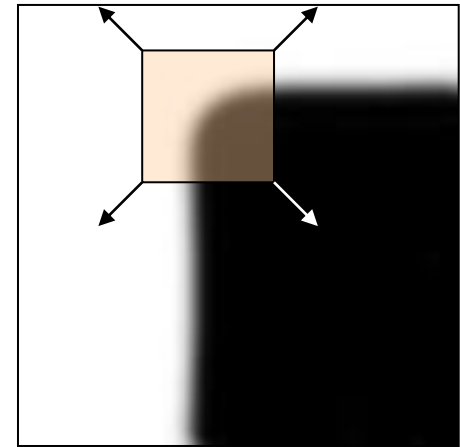
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



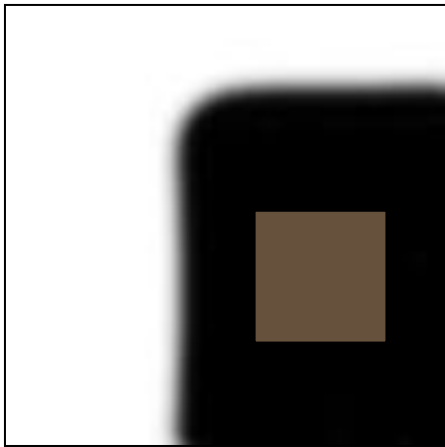
“edge”:
no change along
the edge direction



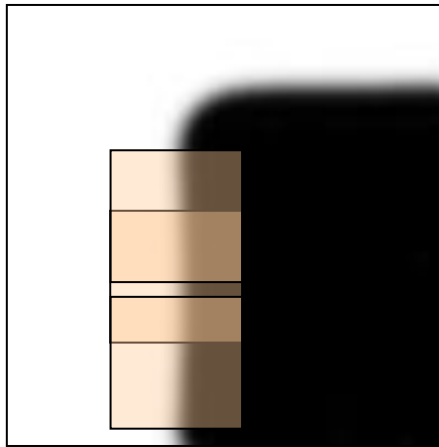
“corner”:
significant change
in all directions

Corners as distinctive interest points

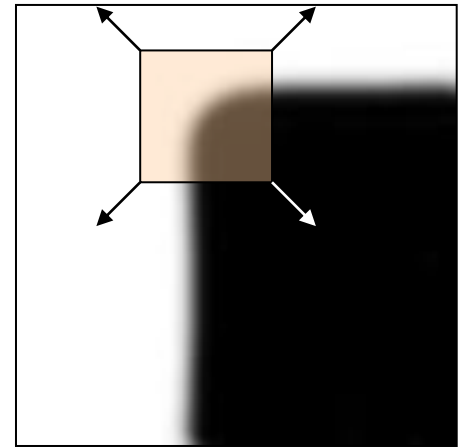
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



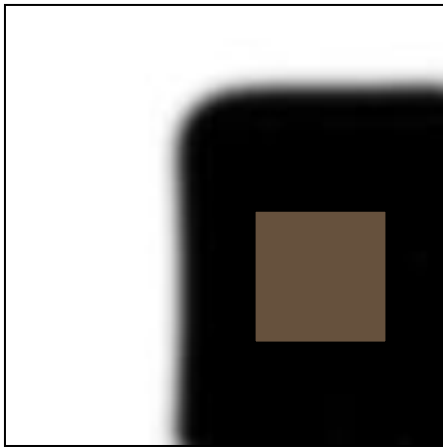
“edge”:
no change along
the edge direction



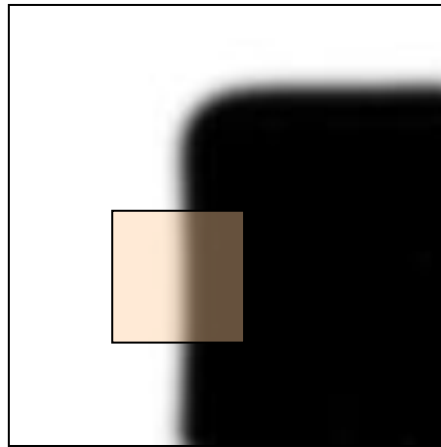
“corner”:
significant change
in all directions

Corners as distinctive interest points

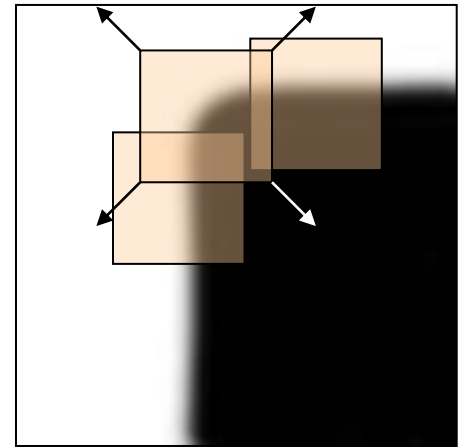
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“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 $[u, v]$:

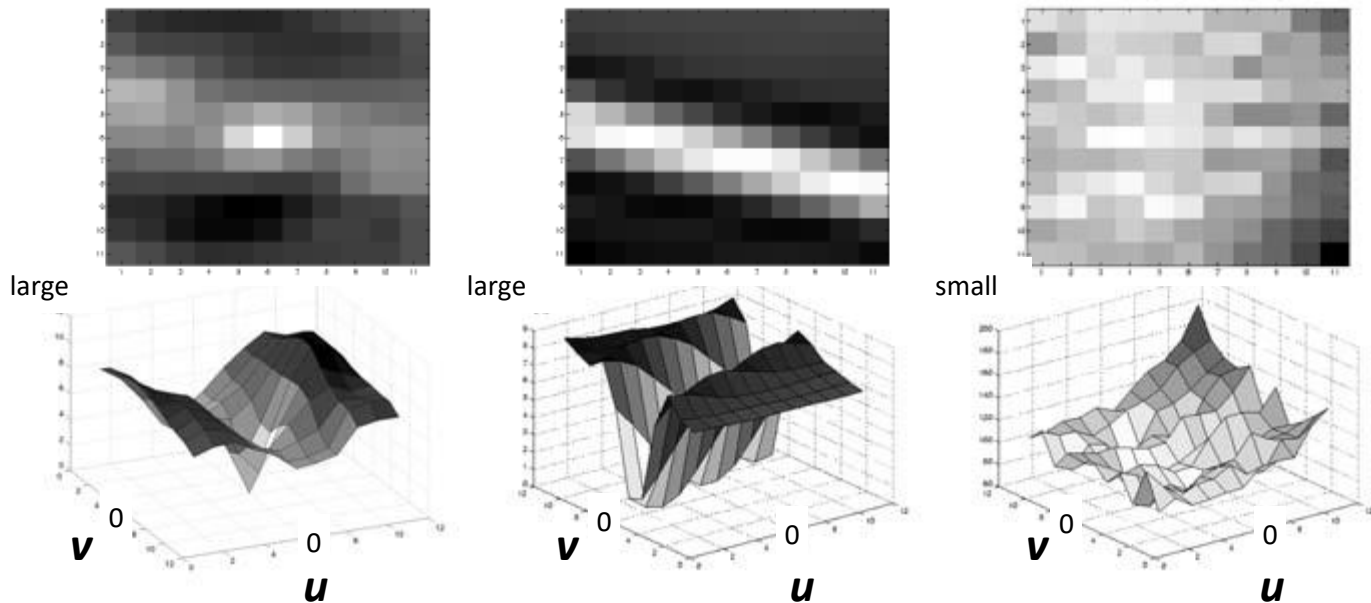
$$E(u, v) = \sum_{x, y} [I(x + u, y + v) - I(x, y)]^2$$


The diagram illustrates the components of the Harris detector equation. A green rectangular box contains the equation $E(u, v) = \sum_{x, y} [I(x + u, y + v) - I(x, y)]^2$. Below the box, there are two light green rounded rectangular labels. The label 'Shifted intensity' has an arrow pointing to the term $I(x + u, y + v)$ in the equation. The label 'Intensity' has an arrow pointing to the term $I(x, y)$ in the equation.

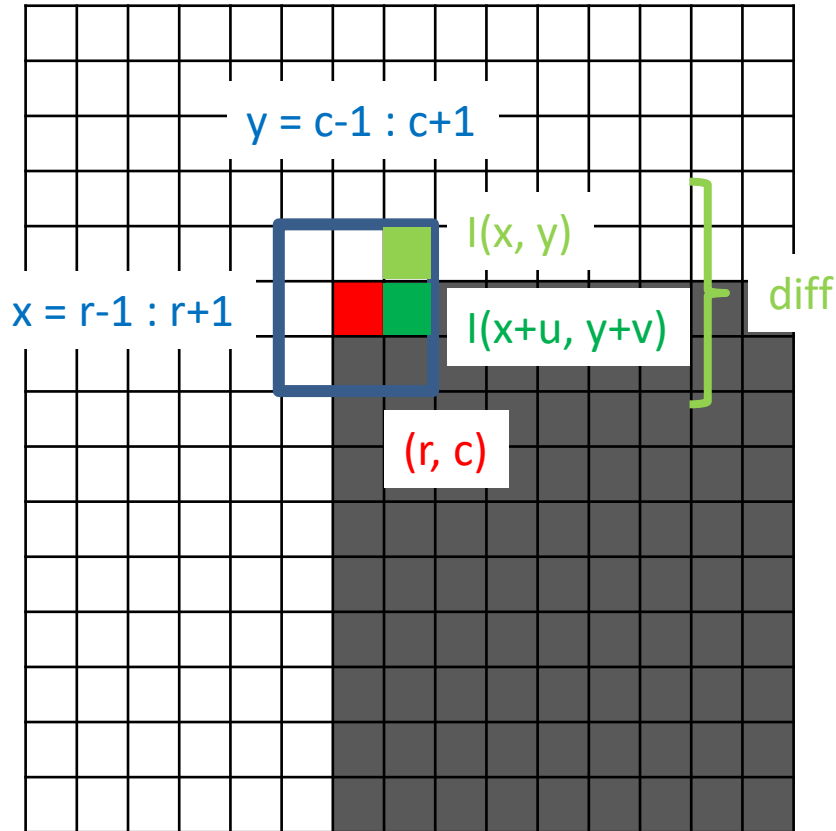
Harris Detector: Mathematics

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

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



Harris Detector: Mathematics



Here $u = 1, v = 0$

For every pixel (r, c) as candidate keypoint
Initialize $E = \text{zeros}(\text{max_offset}, \text{max_offset})$
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 $E(u, v)$

See [autocorr_surface.m](#)

Harris Detector: Mathematics

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

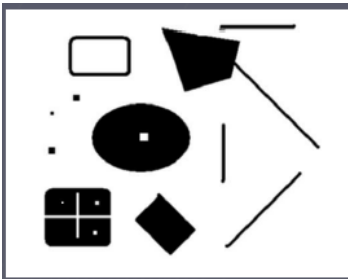
$$E(u, v) \cong [u, v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

Harris Detector: Mathematics

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



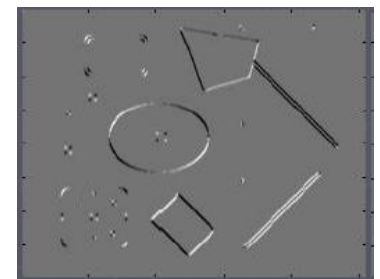
Notation:



$$I_h \Leftrightarrow \frac{\partial I}{\partial x}$$

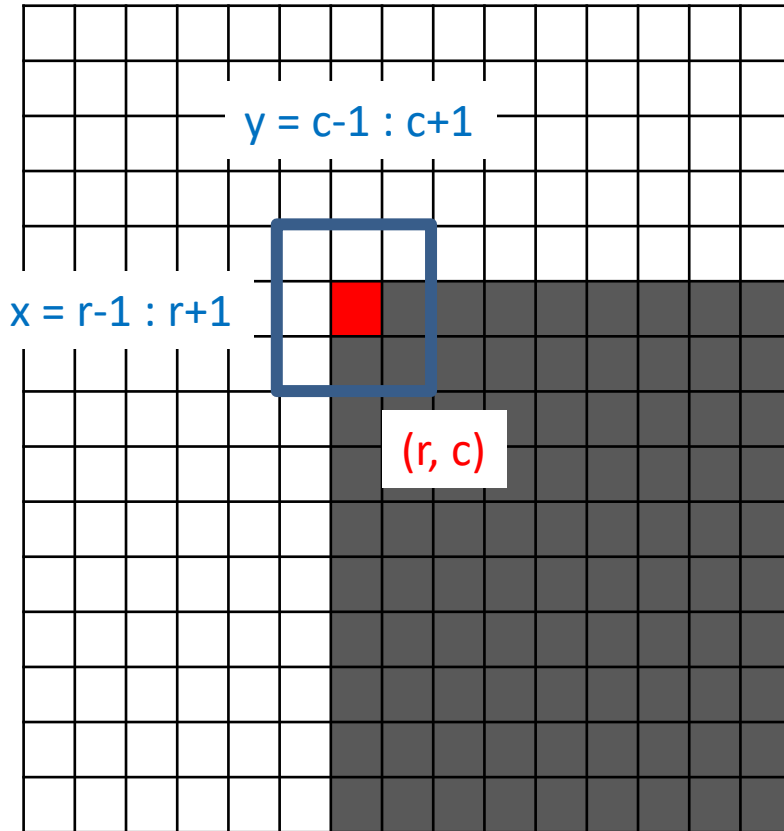


$$I_v \Leftrightarrow \frac{\partial I}{\partial y}$$



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

Harris Detector: Mathematics



Your homework!

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

For every pixel (r, c) as candidate keypoint

Initialize $M = \text{zeros}(2, 2)$

For $x = r-1 : r+1$

For $y = c-1 : c+1$

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

$M(1, 2) = ?$

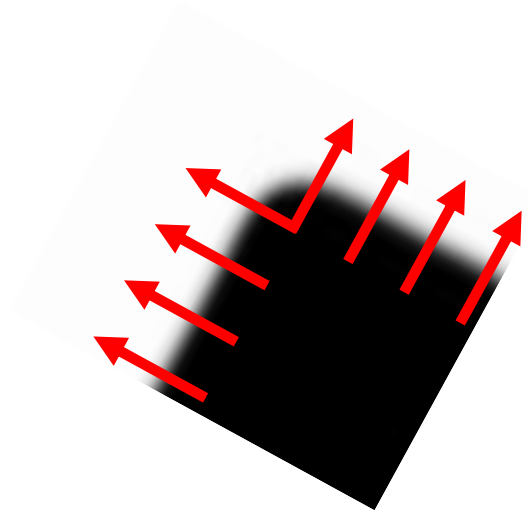
$M(2, 1) = ?$

$M(2, 2) = ?$

What does the matrix M reveal?

Since M is symmetric, we have

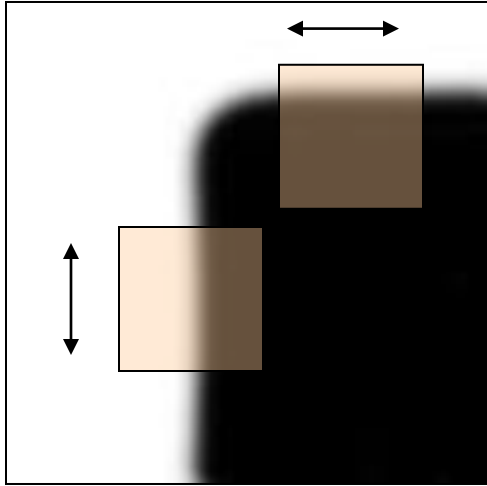
$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$



$$Mx_i = \lambda_i x_i$$

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

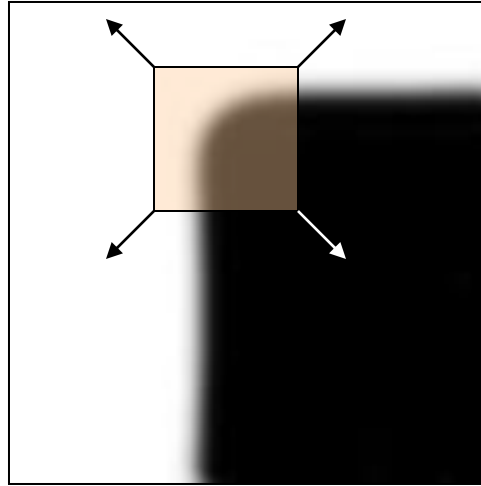
Corner response function



“edge”:

$$\lambda_1 \gg \lambda_2$$

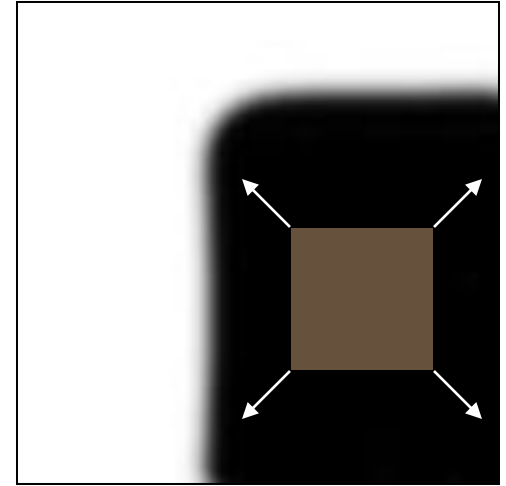
$$\lambda_2 \gg \lambda_1$$



“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:

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

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

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

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

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

by looping over neighbors x, y

- Compute

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

(k : empirical constant, $k = 0.04-0.06$)

- Find points with large corner response function R
($R > \text{threshold}$)

Harris Detector: Algorithm

- Finally, perform 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)

3	12	2
8	15	9
6	14	19

4 neighbors

3	12	2
8	15	9
6	14	19

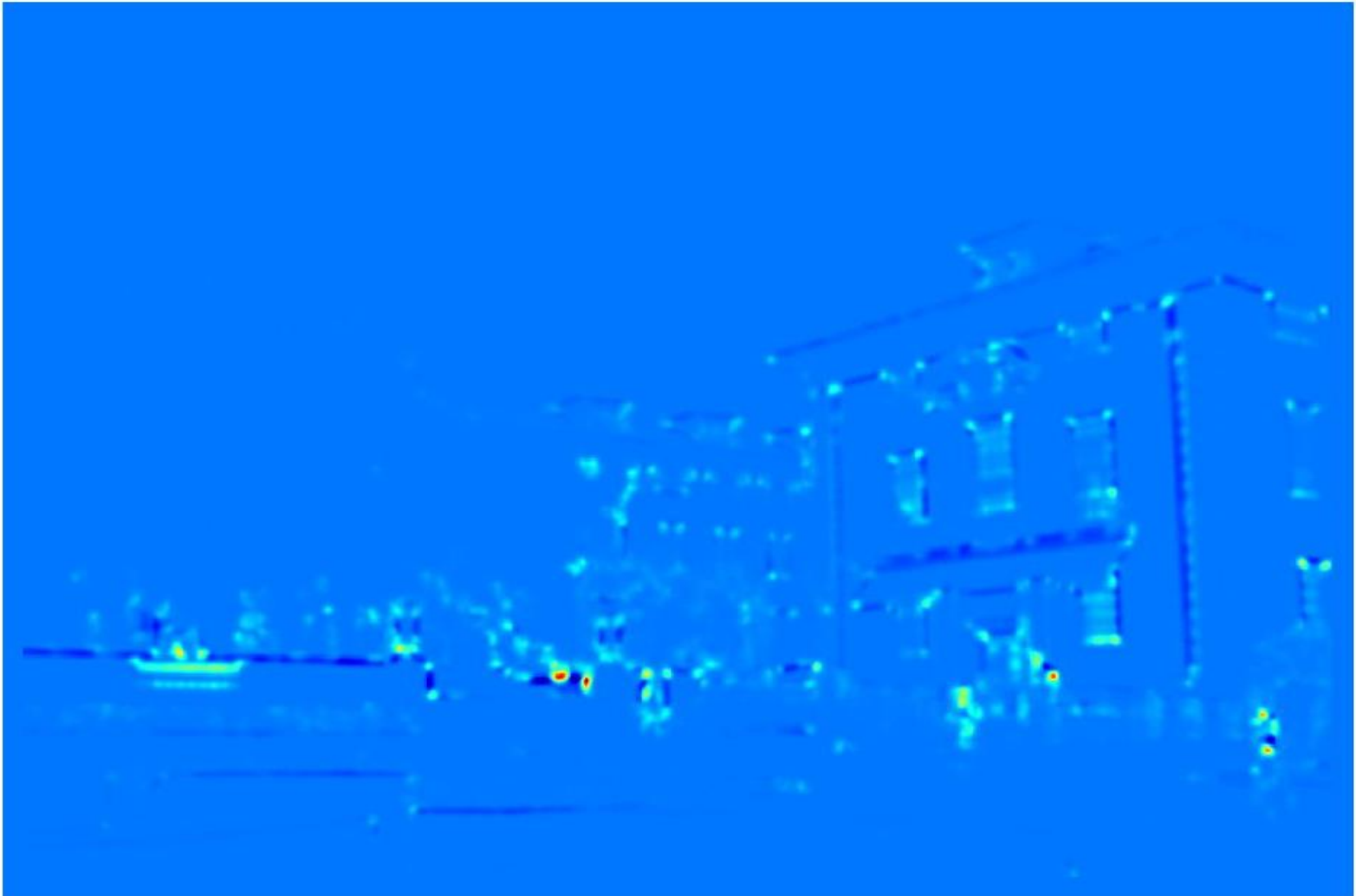
8 neighbors

Example of Harris application

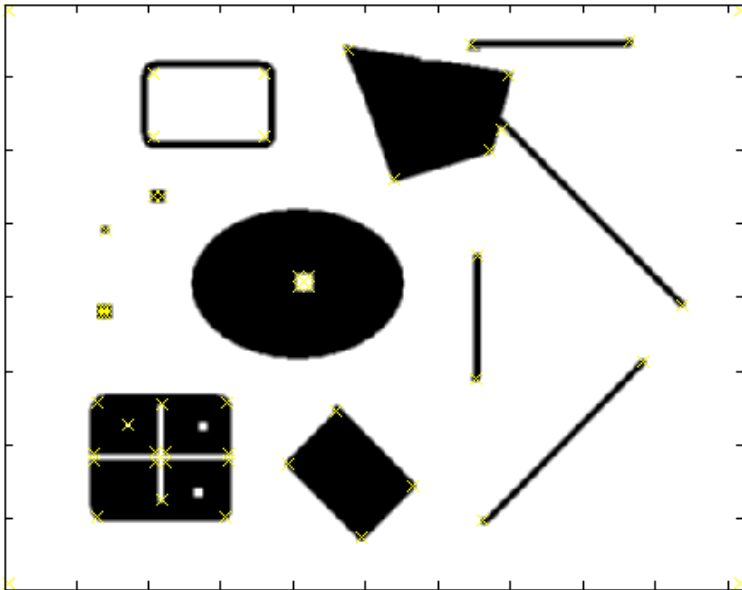


Example of Harris application

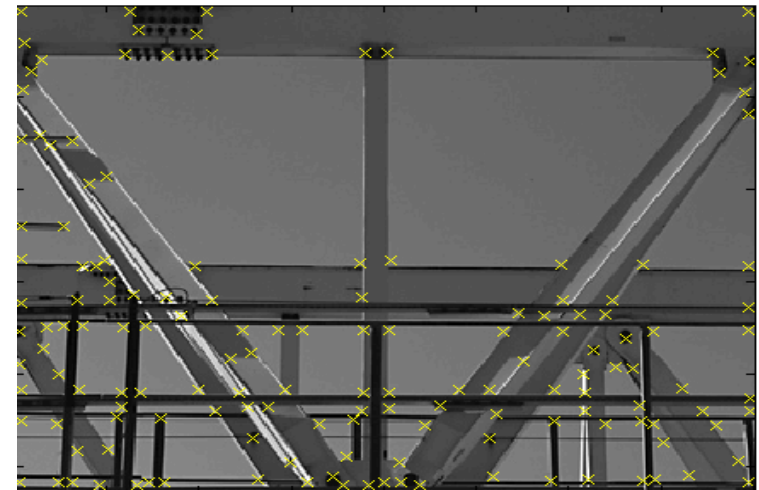
- Corner response at every pixel (red = high, blue = low)



More Harris responses



Effect: A very precise corner detector.



Properties: Invariance vs covariance

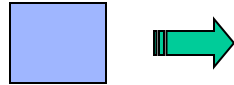
“A function is *invariant* under a certain family of transformations if its value does not change when a transformation from this family is applied to its argument.

- [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.

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.

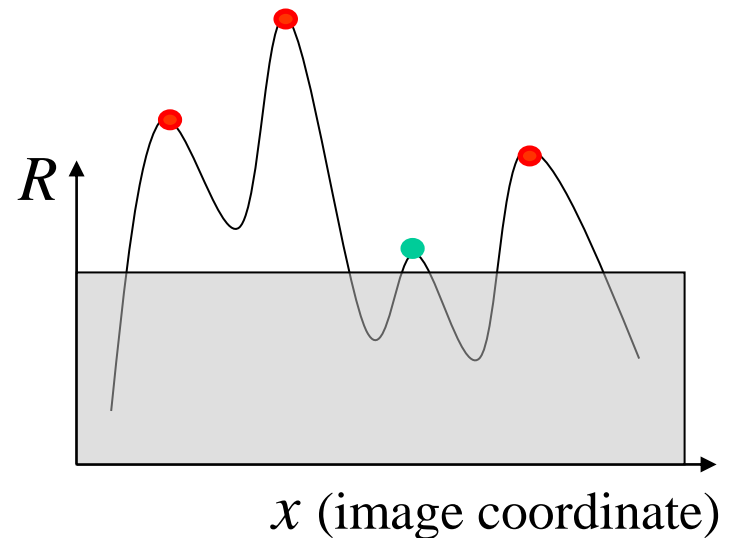
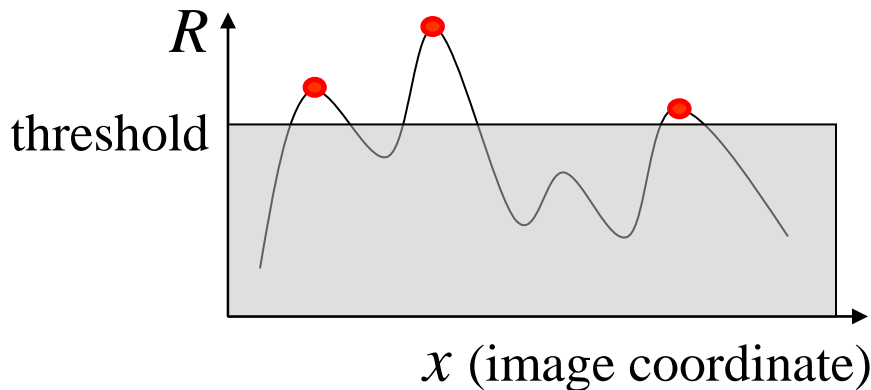
- The orientation of the major axis of inertia of the surface is covariant under the same family of transformations, since rotating a 2D surface will affect the orientation of its major axis in exactly the same way.”
- Another example: If f is *invariant* under linear transformations, then $f(ax+b) = f(x)$, and if it is *covariant* with respect to these transformations, then $f(ax+b) = a f(x) + b$

What happens if: Affine intensity change



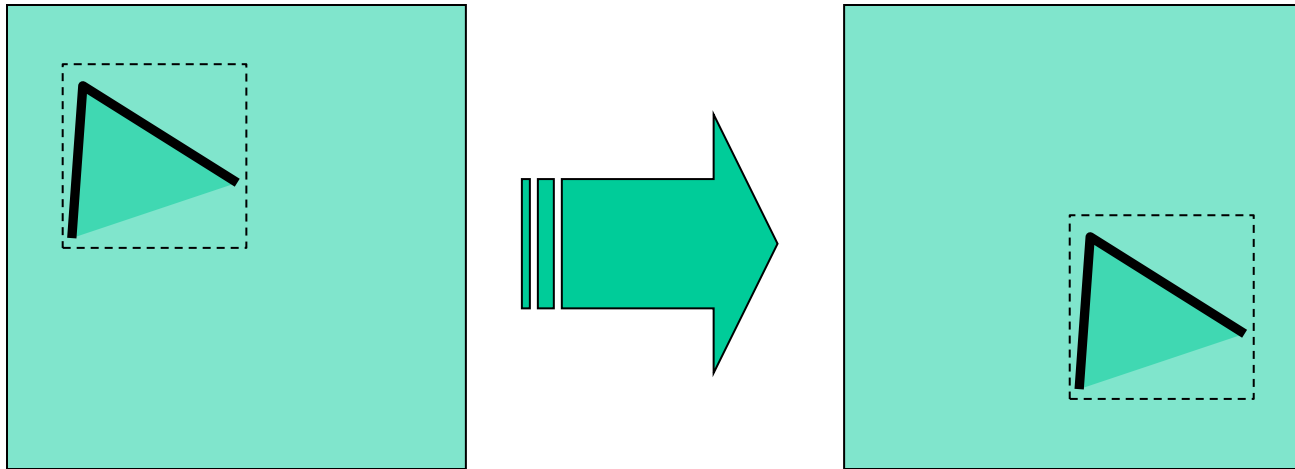
$$I \rightarrow a I + b$$

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



Partially invariant to affine intensity change

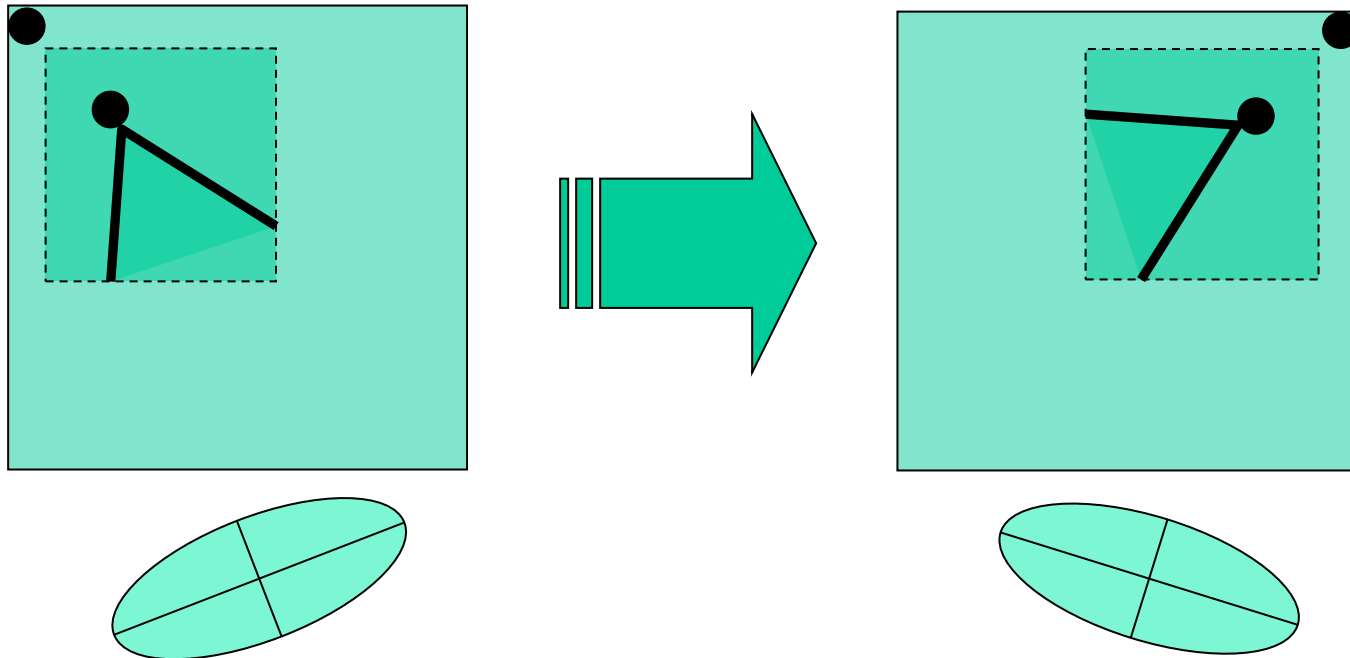
What happens if: Image translation



- 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?

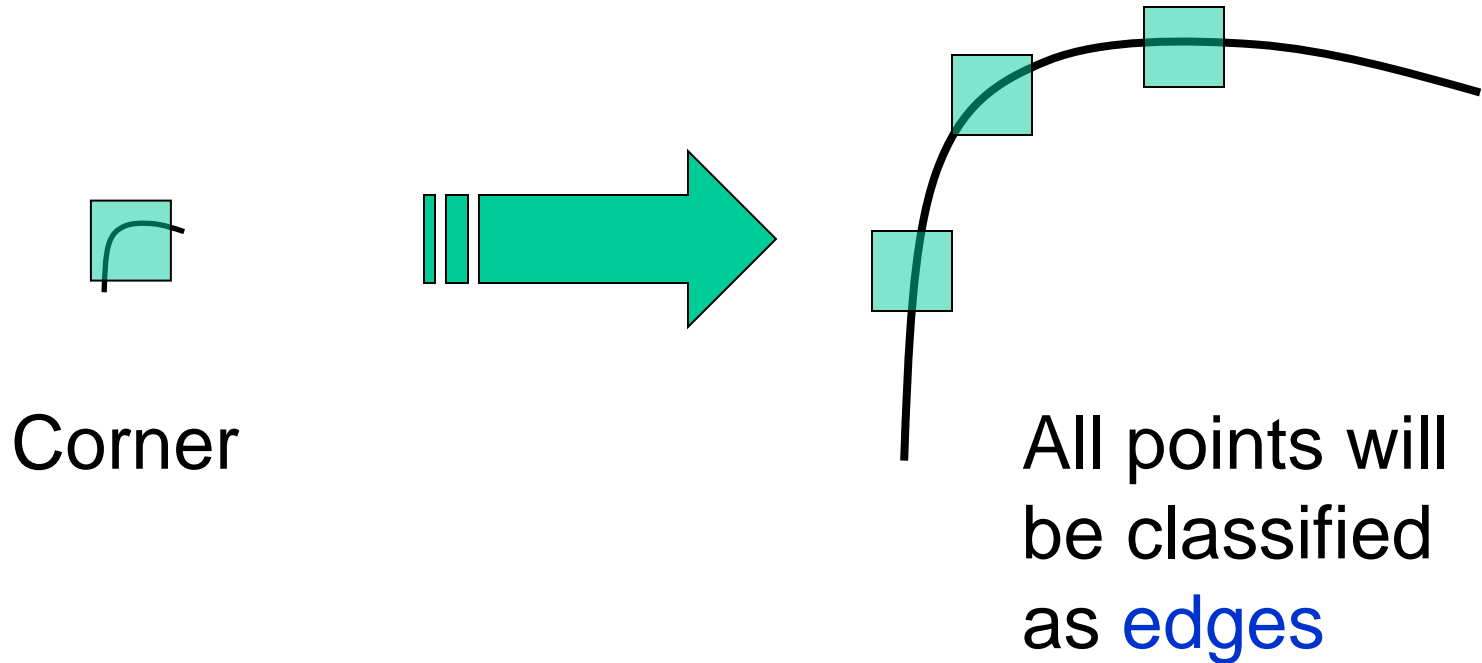


image



zoomed image

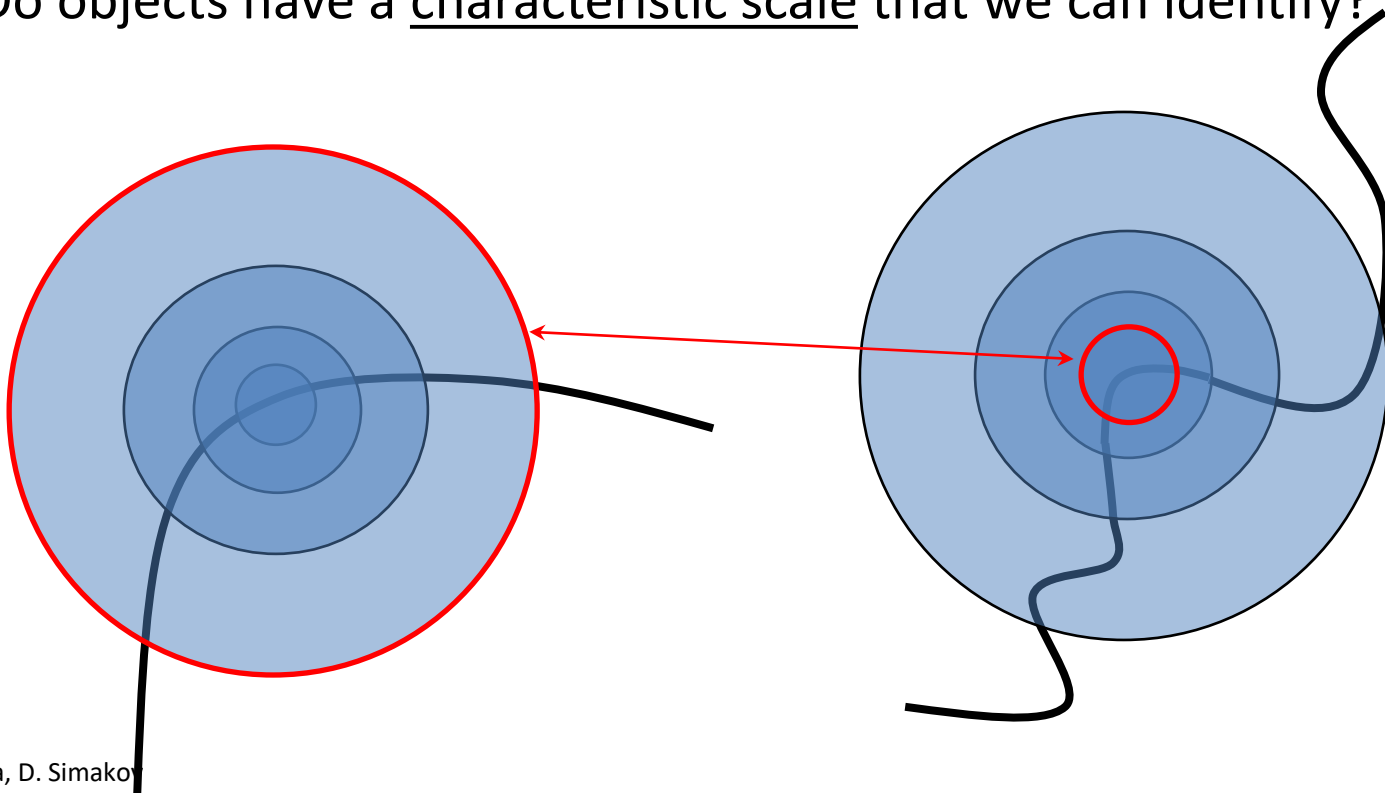
What happens if: Scaling



Corner location is not covariant to scaling!
(Window size is part of algorithm)

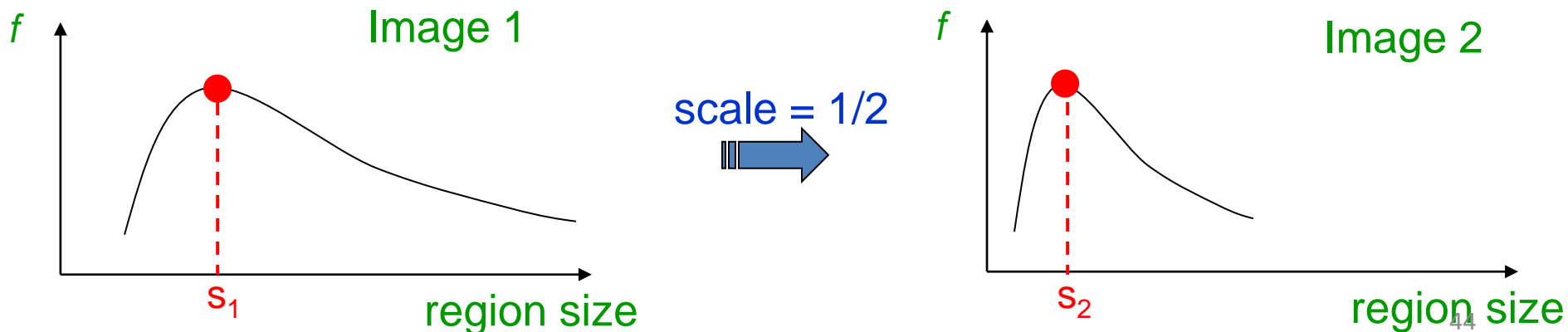
Scale invariant detection

- Problem:
 - How do we choose corresponding windows *independently* in each image?
 - Do objects have a characteristic scale that we can identify?



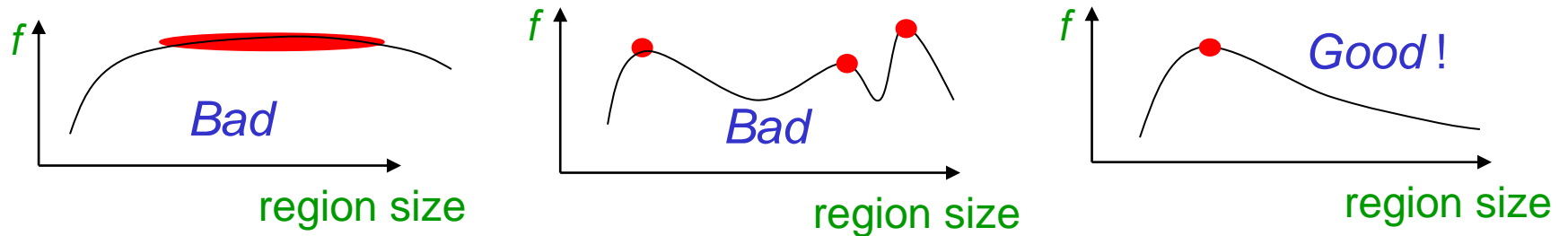
Scale invariant detection

- 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



Scale invariant detection

- A “good” function for scale detection:
has one stable sharp peak



Automatic scale selection

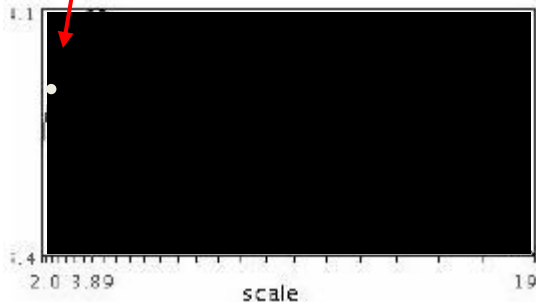


$$f(I_{i_1 \dots i_m}(x, \sigma)) = f(I_{i_1 \dots i_m}(x', \sigma'))$$

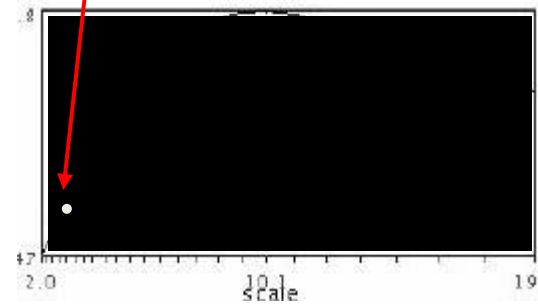
How to find corresponding patch sizes?

Automatic scale selection

- Function responses for increasing scale (scale signature)



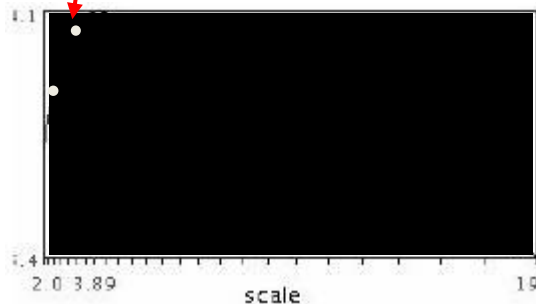
$$f(I_{i_1...i_m}(x, \sigma))$$



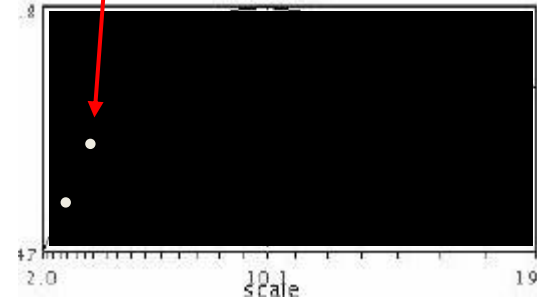
$$f(I_{i_1...i_m}(x', \sigma))$$

Automatic scale selection

- Function responses for increasing scale (scale signature)



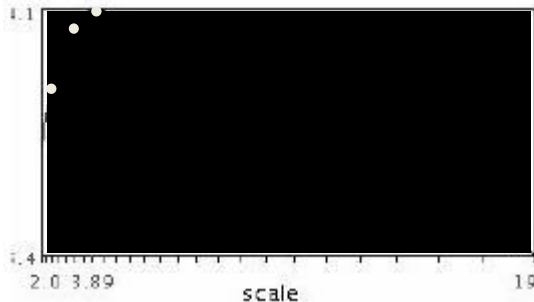
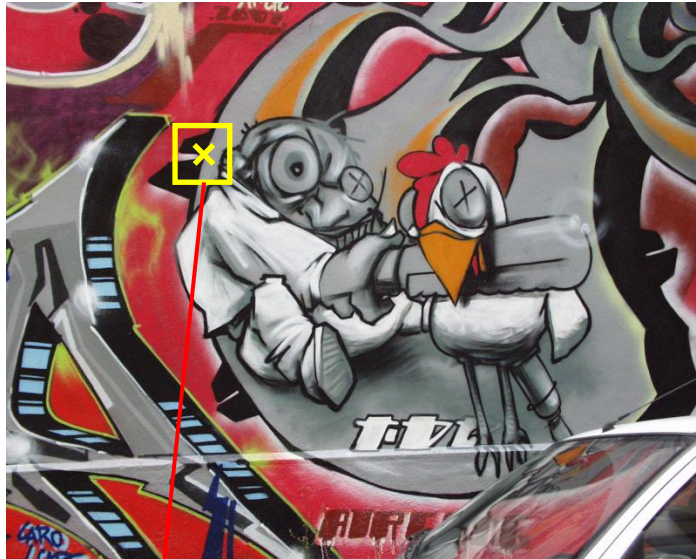
$$f(I_{i_1 \dots i_m}(x, \sigma))$$



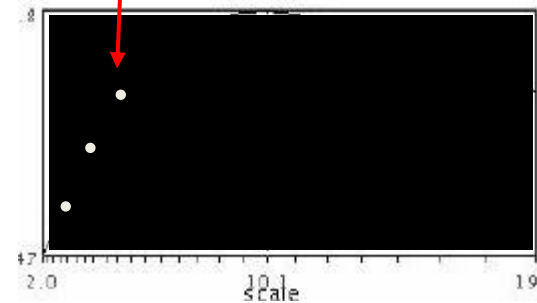
$$f(I_{i_1 \dots i_m}(x', \sigma))$$

Automatic scale selection

- Function responses for increasing scale (scale signature)



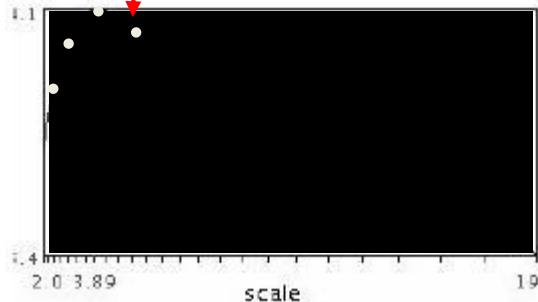
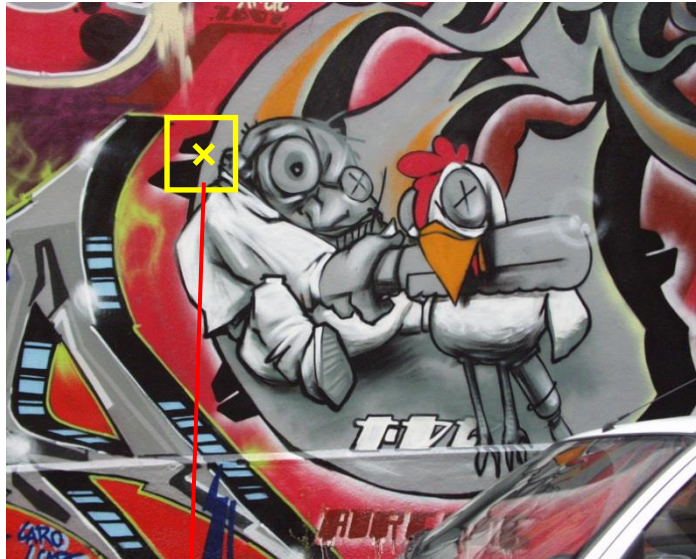
$$f(I_{i_1 \dots i_m}(x, \sigma))$$



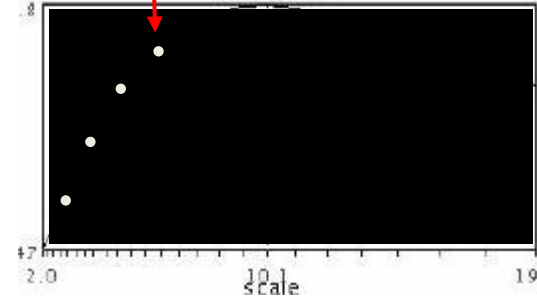
$$f(I_{i_1 \dots i_m}(x', \sigma))$$

Automatic scale selection

- Function responses for increasing scale (scale signature)



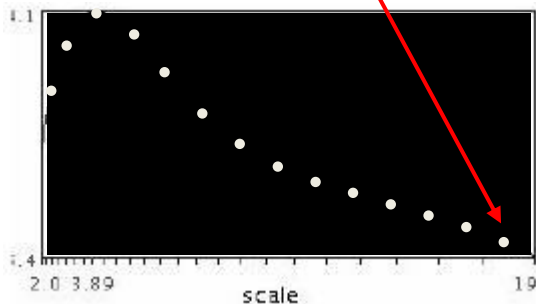
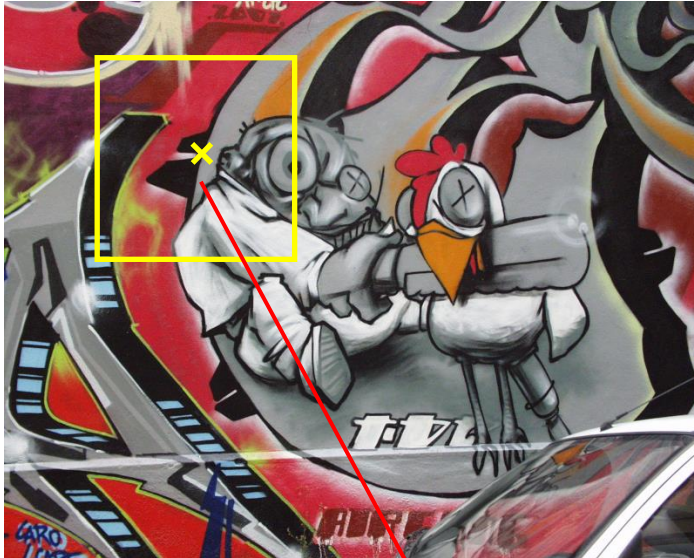
$$f(I_{i_1...i_m}(x, \sigma))$$



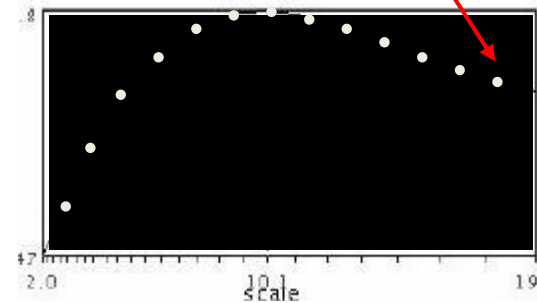
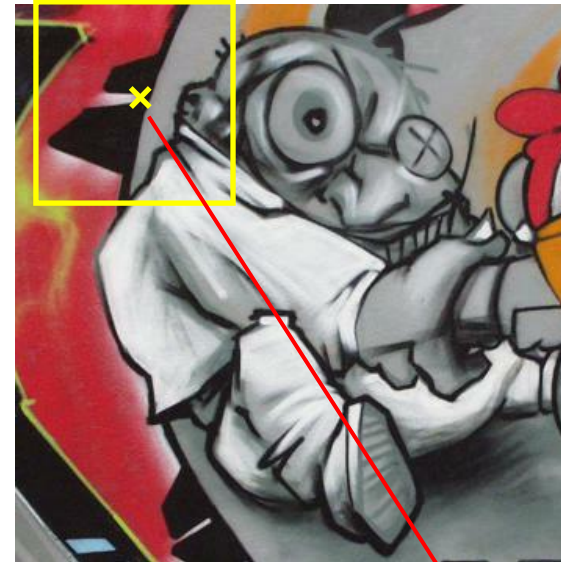
$$f(I_{i_1...i_m}(x', \sigma))$$

Automatic scale selection

- Function responses for increasing scale (scale signature)



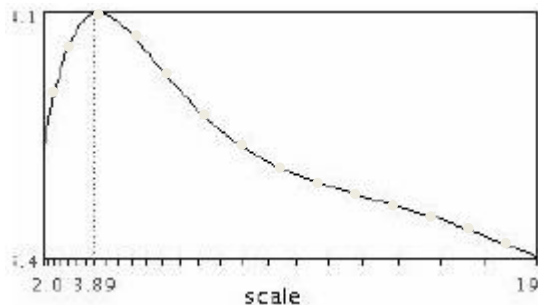
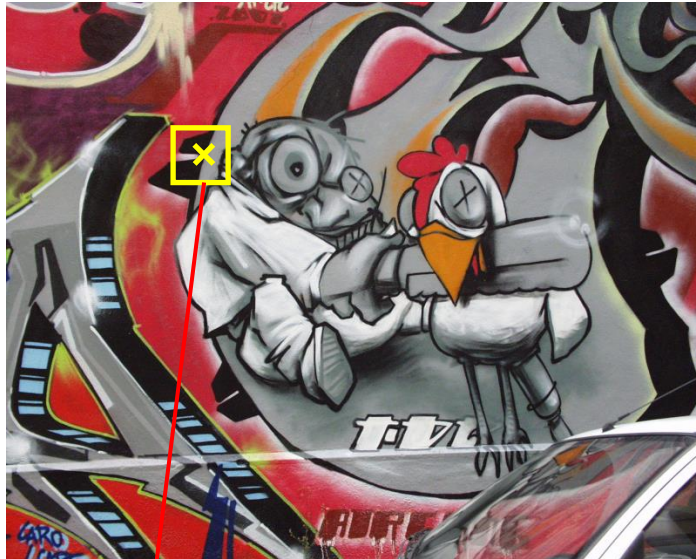
$$f(I_{i_1...i_m}(x, \sigma))$$



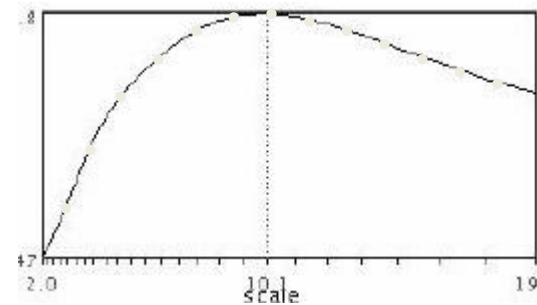
$$f(I_{i_1...i_m}(x', \sigma))$$

Automatic scale selection

- Function responses for increasing scale (scale signature)



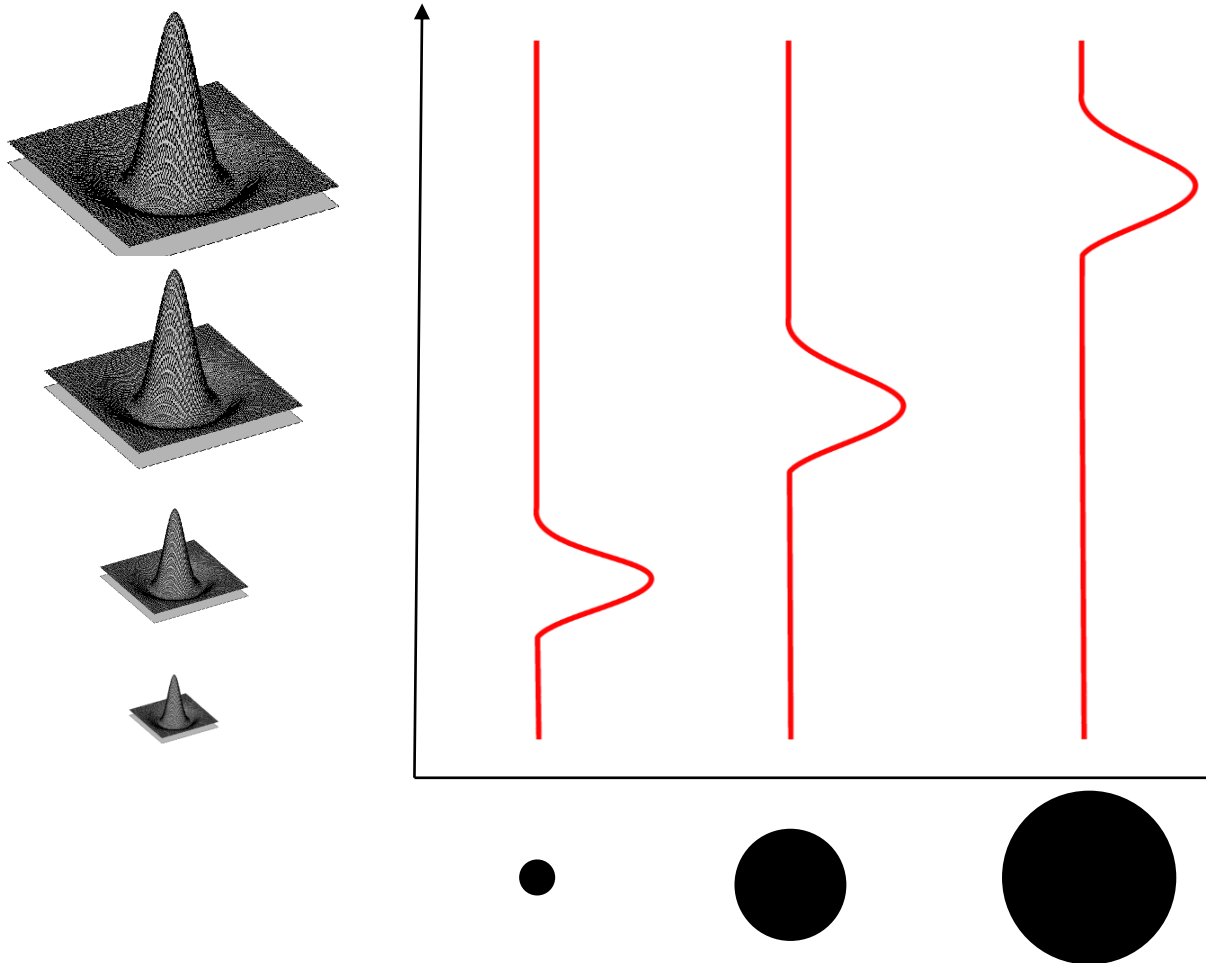
$$f(I_{i_1 \dots i_m}(x, \sigma))$$



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

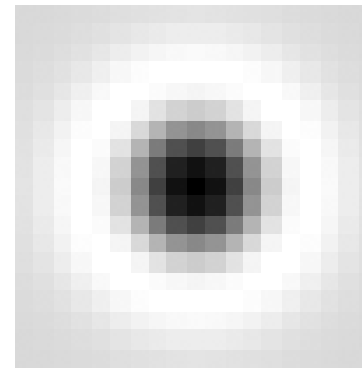
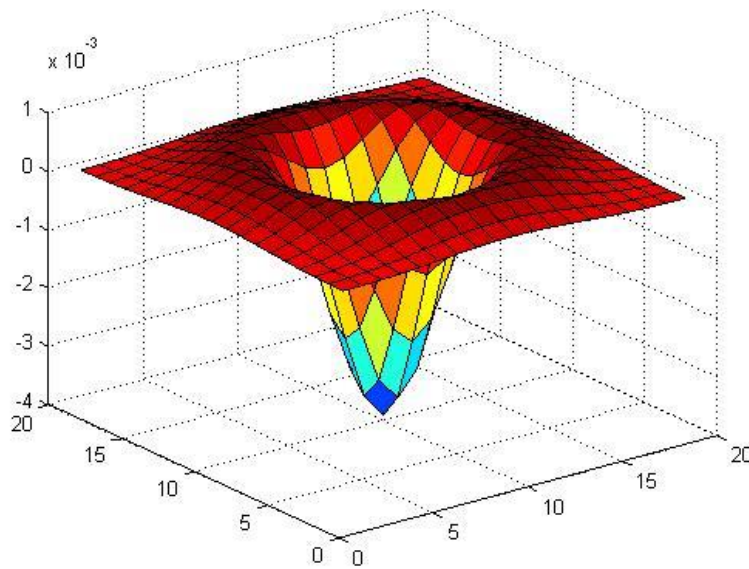
What is a useful signature function?

- Laplacian of Gaussian = “blob” detector

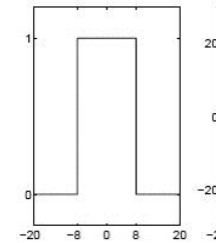


Blob detection in 2D

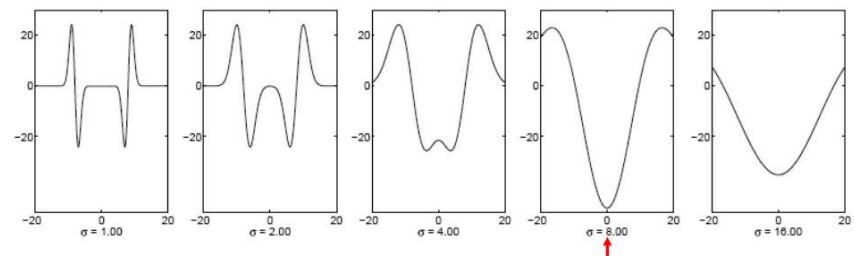
- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D, second derivative of Gaussian



Original signal



Edge response



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

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Difference of Gaussian \approx Laplacian

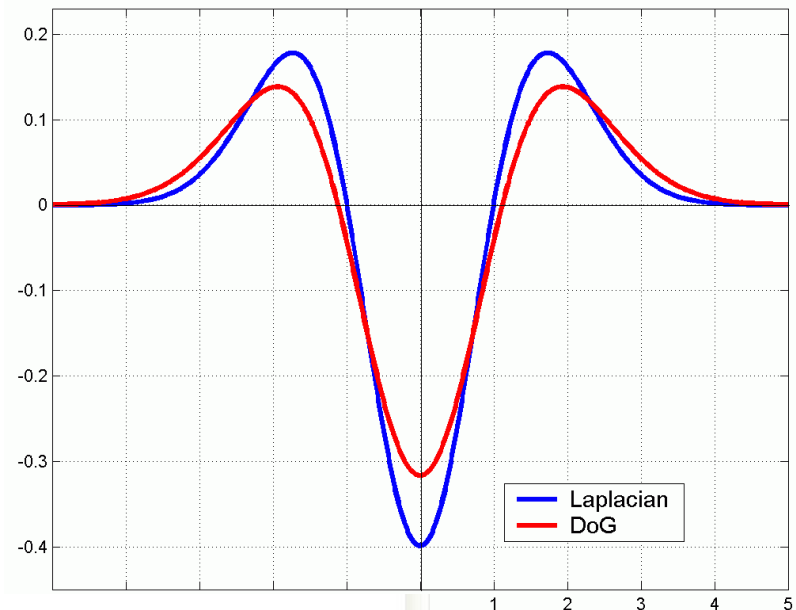
- We can approximate Laplacian with difference of Gaussians; more efficient to implement.

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

(Laplacian)

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

(Difference of Gaussians)



$I(k\sigma)$

$I(\sigma)$

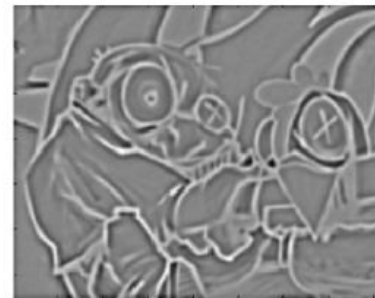
$I(k\sigma) - I(\sigma)$



-

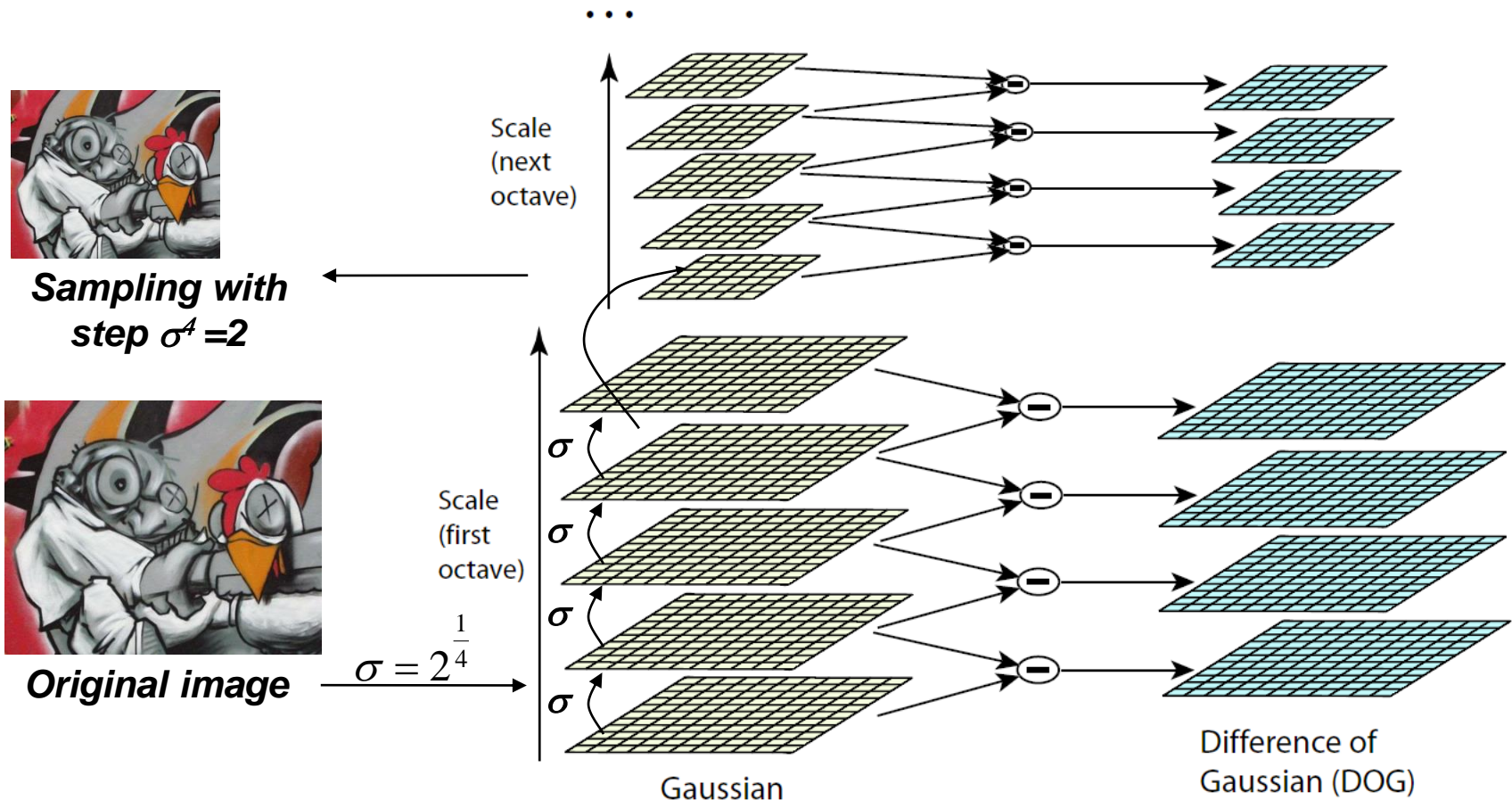


=

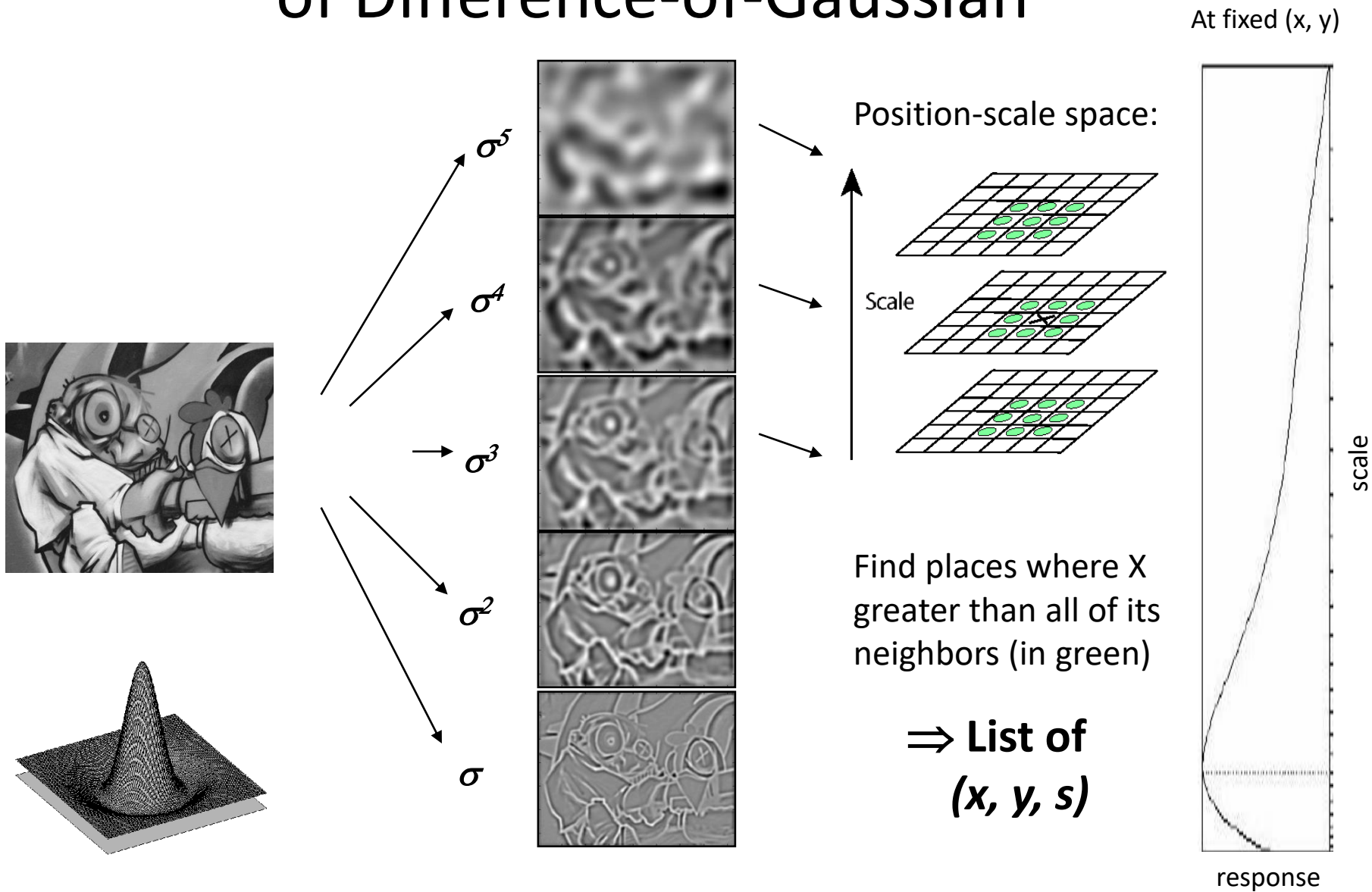


Difference of Gaussian Scale Pyramid

See blobs.m



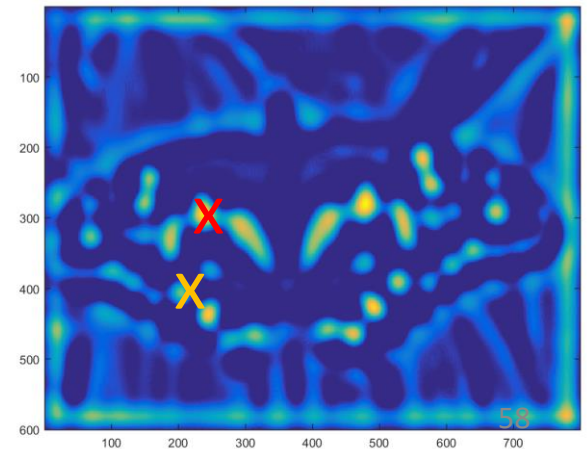
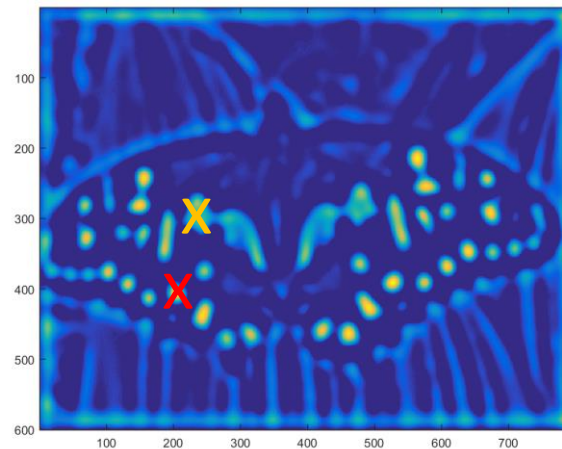
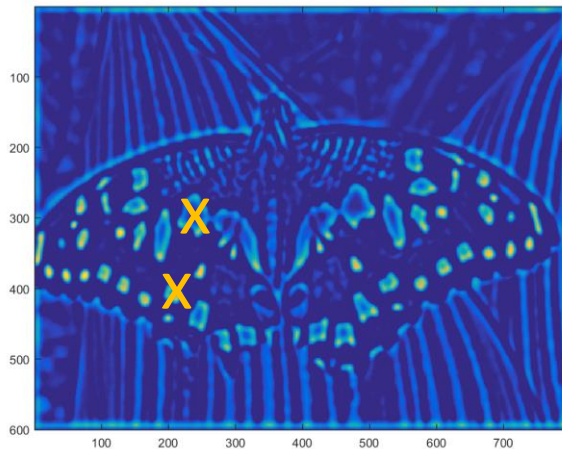
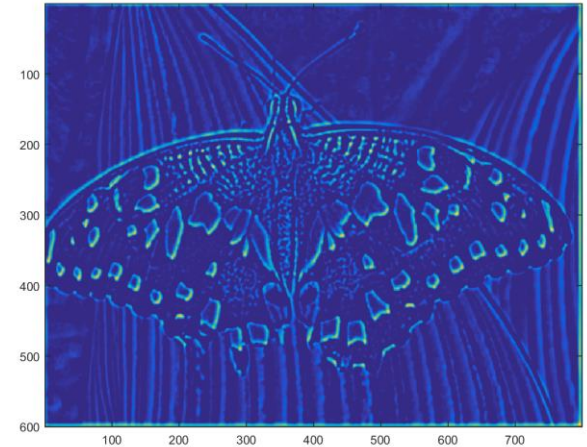
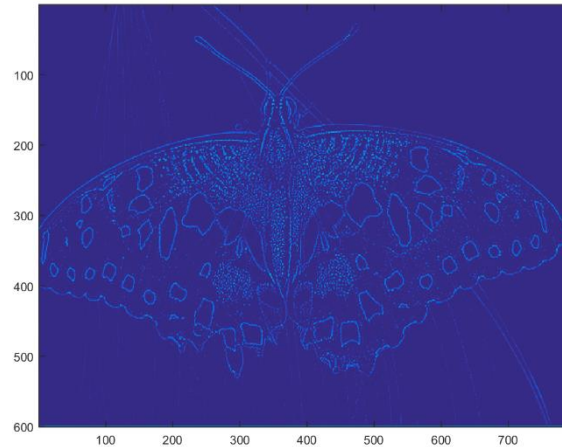
Find *local maxima* in position-scale space of Difference-of-Gaussian



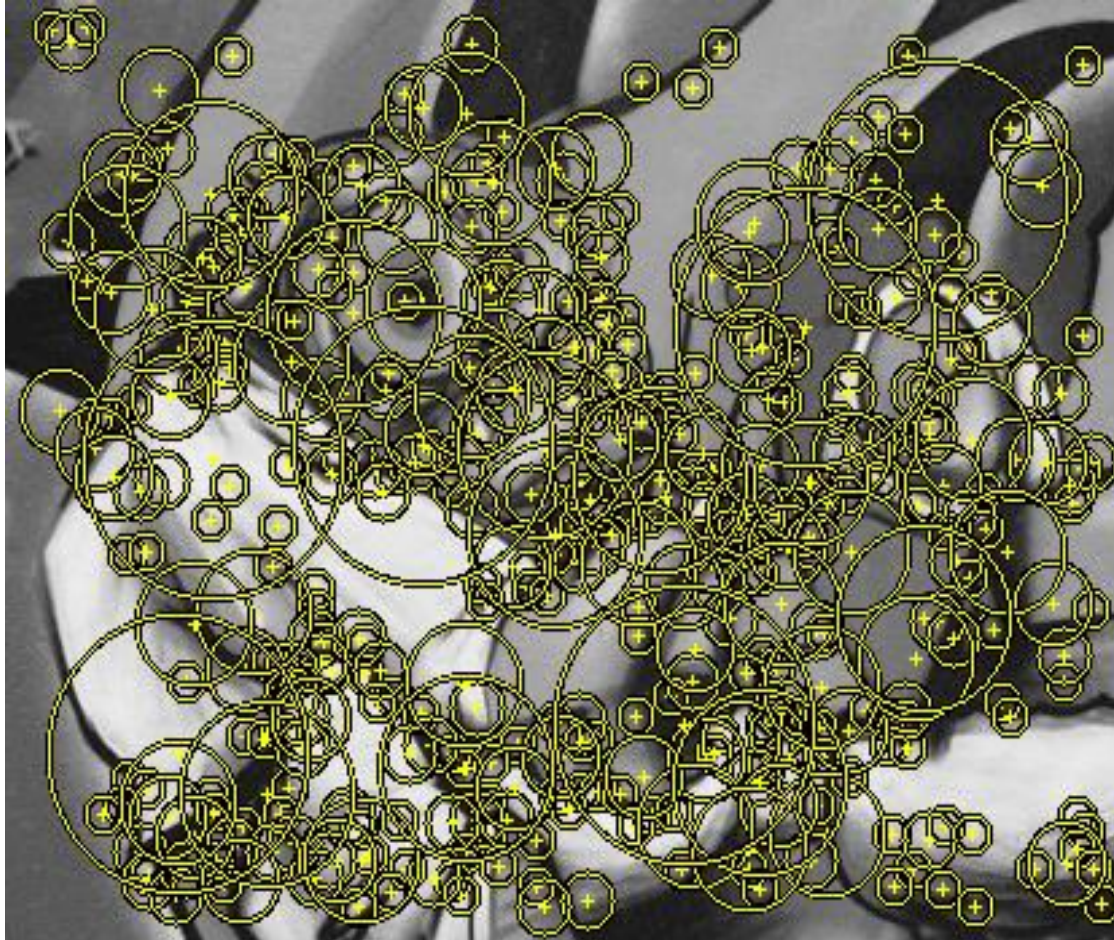
Laplacian pyramid example

- Allows detection of increasingly coarse detail

See `blobs.m`



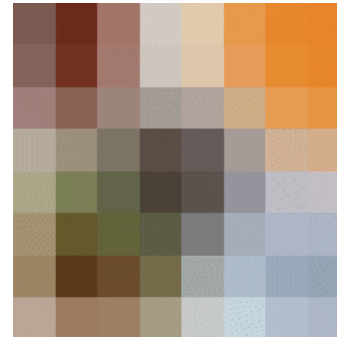
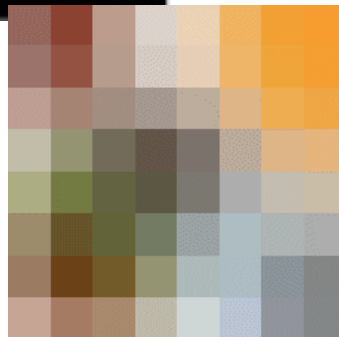
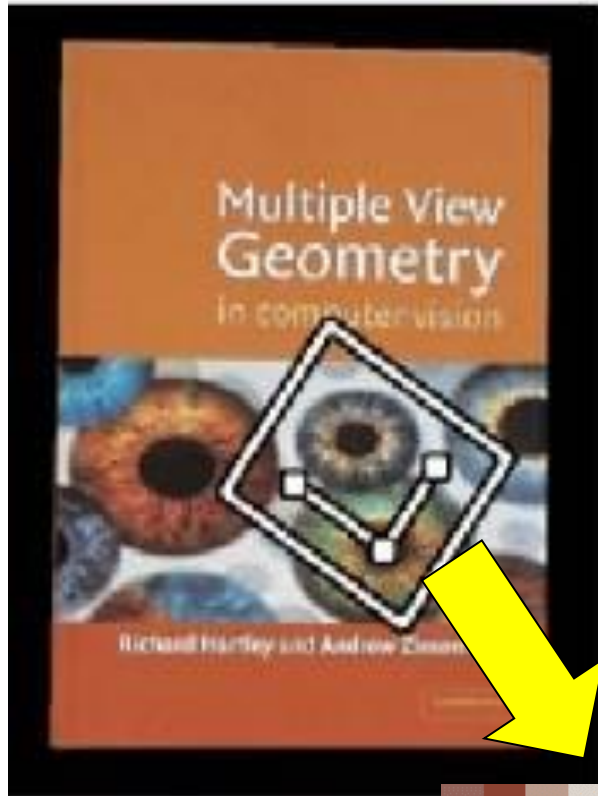
Results: Difference-of-Gaussian



Plan for this lecture

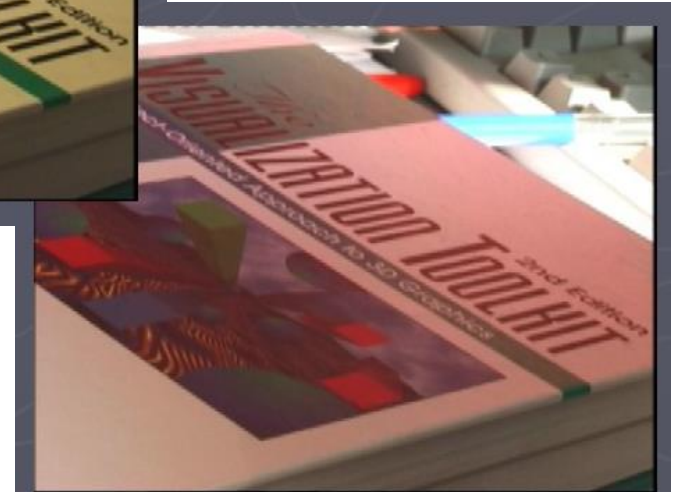
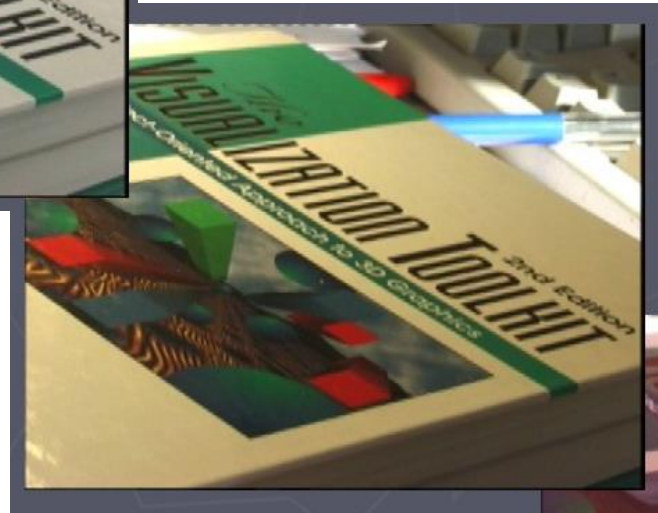
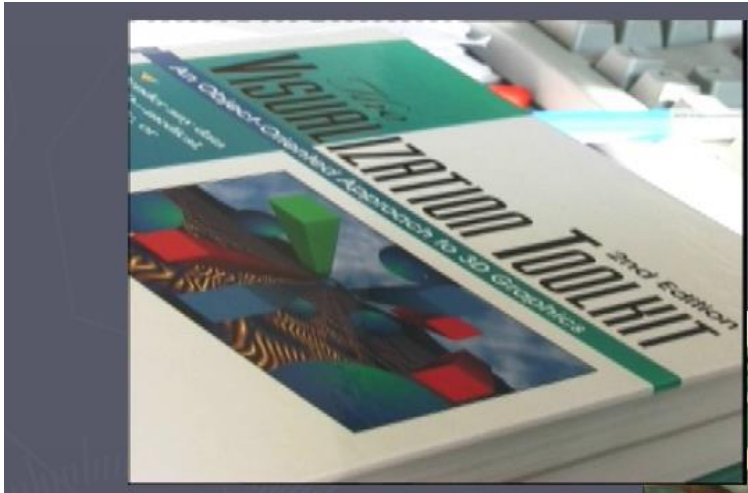
- Feature detection / keypoint extraction
 - Corner detection
 - Blob detection
- Feature description (of detected features)
- Matching features across images

Geometric transformations



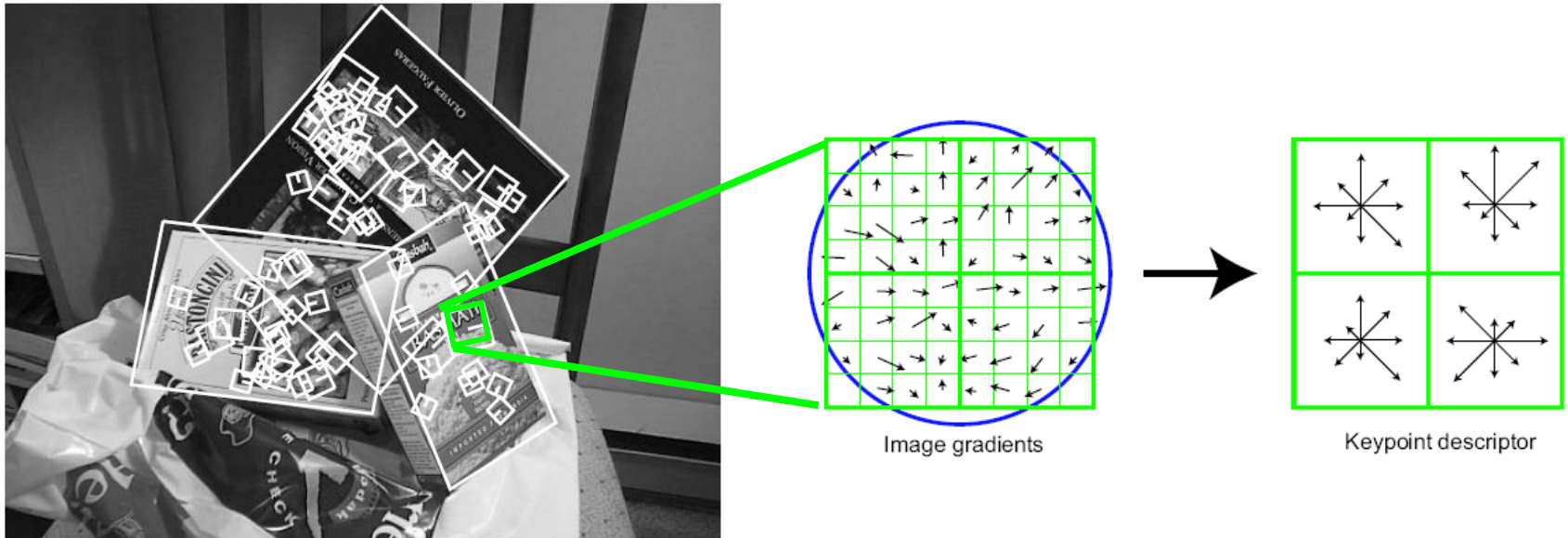
e.g. scale,
translation,
rotation

Photometric transformations



Scale-Invariant Feature Transform (SIFT) descriptor

Journal + conference versions: 87,527 citations (AlexNet paper has 93,821)



Histogram of oriented gradients

- Captures important texture information
- Robust to small translations / affine deformations

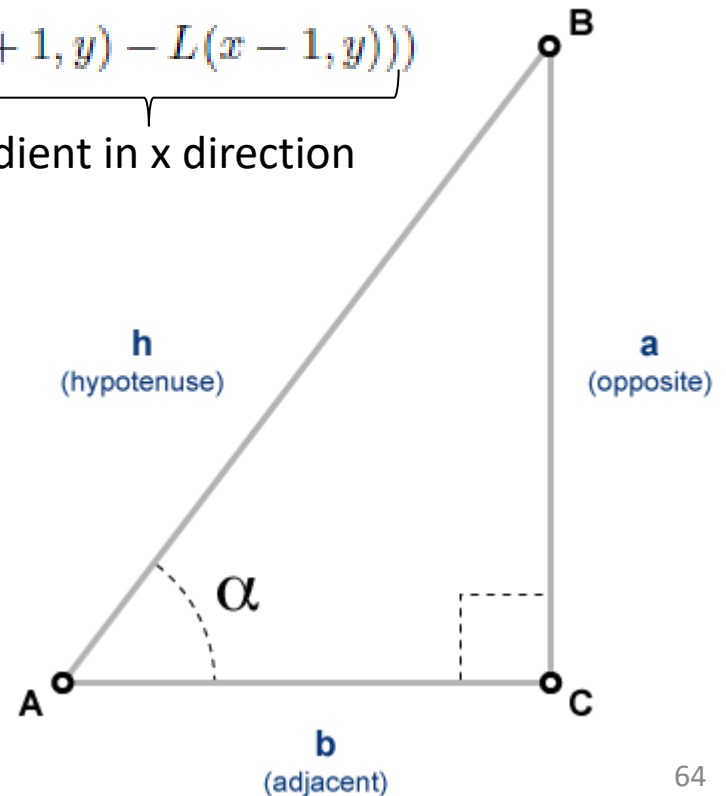
Computing gradients

L = the image intensity

$$m(x, y) = \sqrt{\underbrace{(L(x+1, y) - L(x-1, y))^2}_{\text{gradient in x direction}} + \underbrace{(L(x, y+1) - L(x, y-1))^2}_{\text{gradient in y direction}}}$$

$$\theta(x, y) = \tan^{-1}\left(\underbrace{(L(x, y+1) - L(x, y-1))}_{\text{gradient in y direction}} / \underbrace{(L(x+1, y) - L(x-1, y))}_{\text{gradient in x direction}}\right)$$

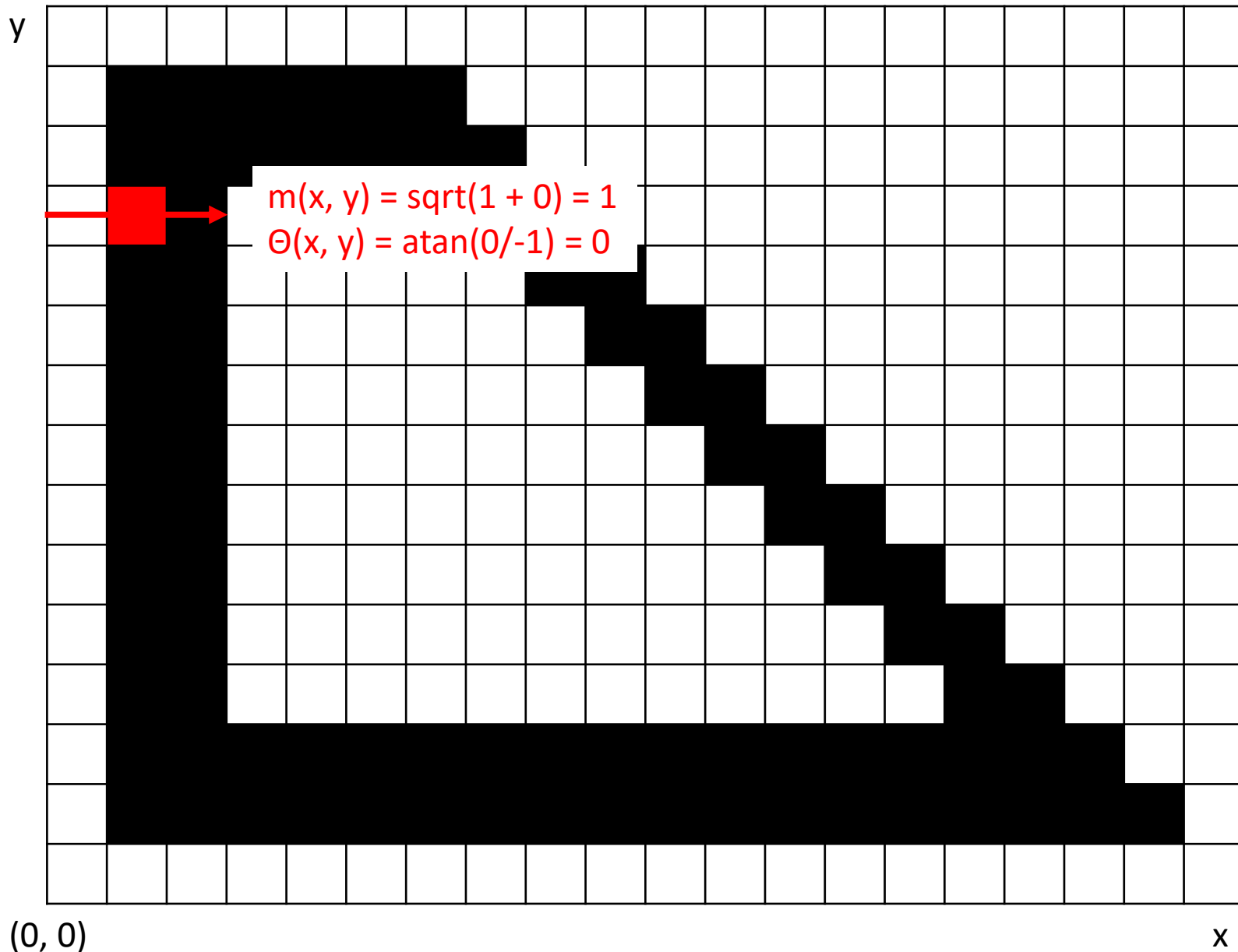
- $\tan(\alpha) = \frac{\text{opposite side}}{\text{adjacent side}}$



Gradients

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

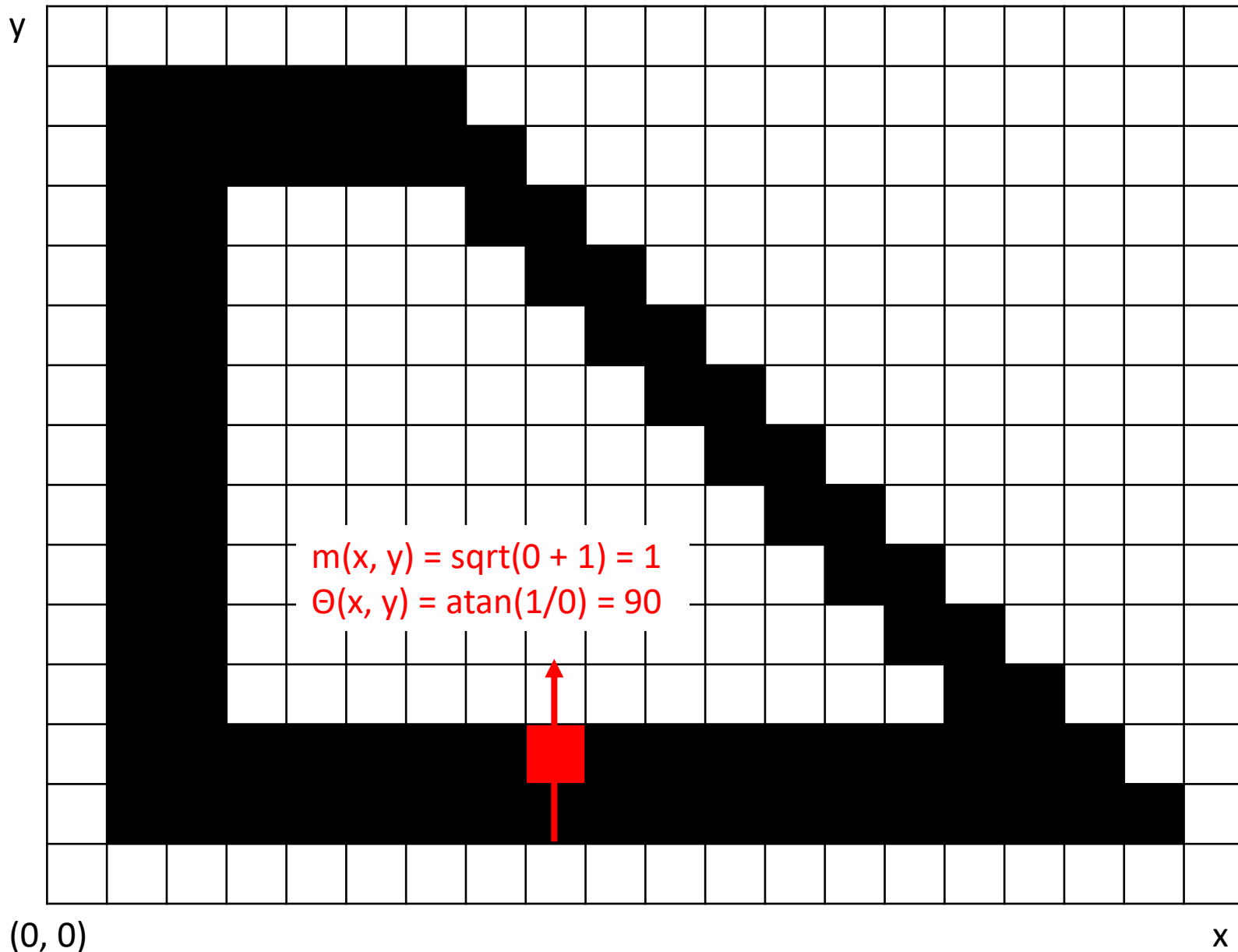
$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$



Gradients

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

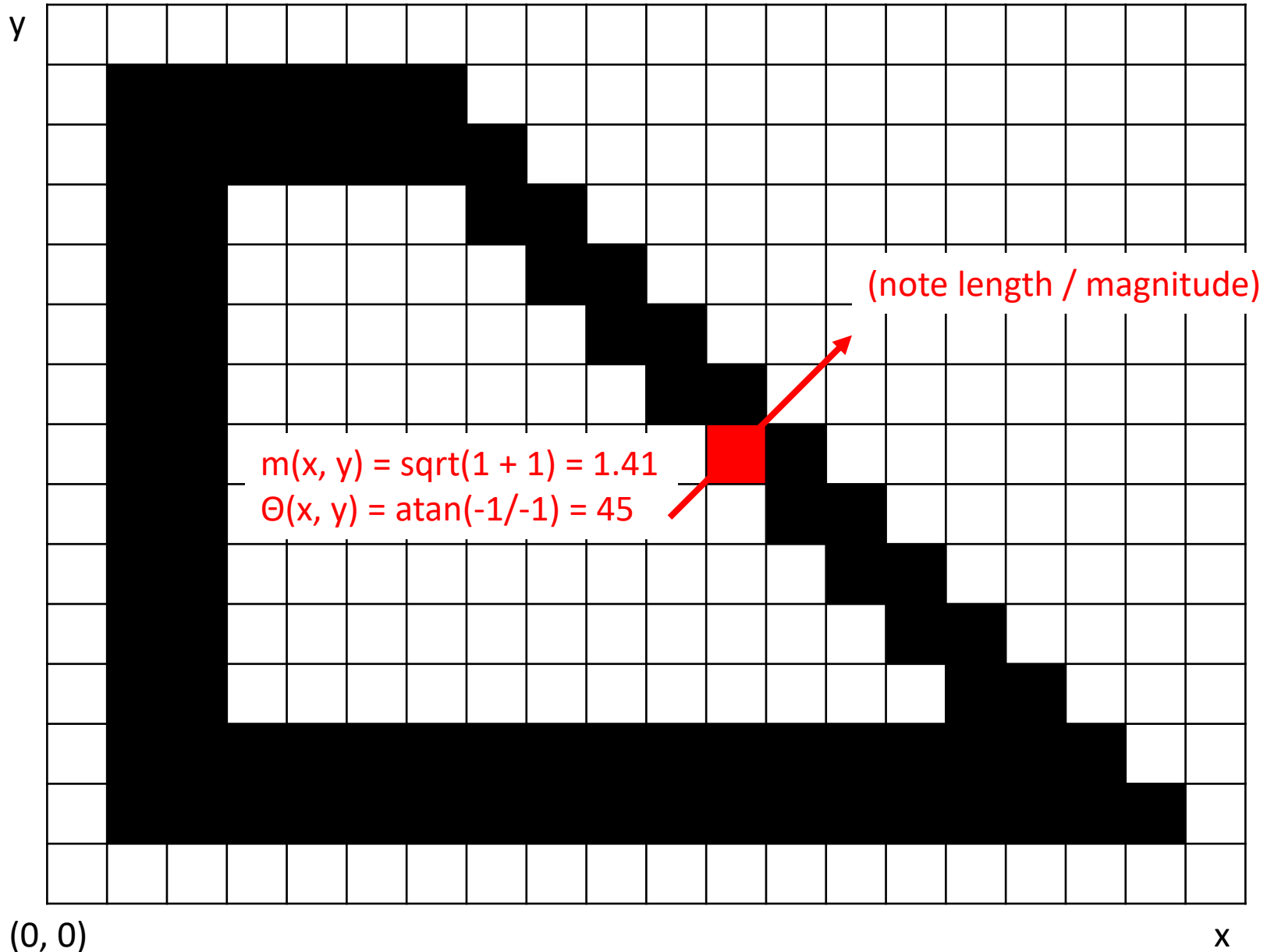
$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$



Gradients

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

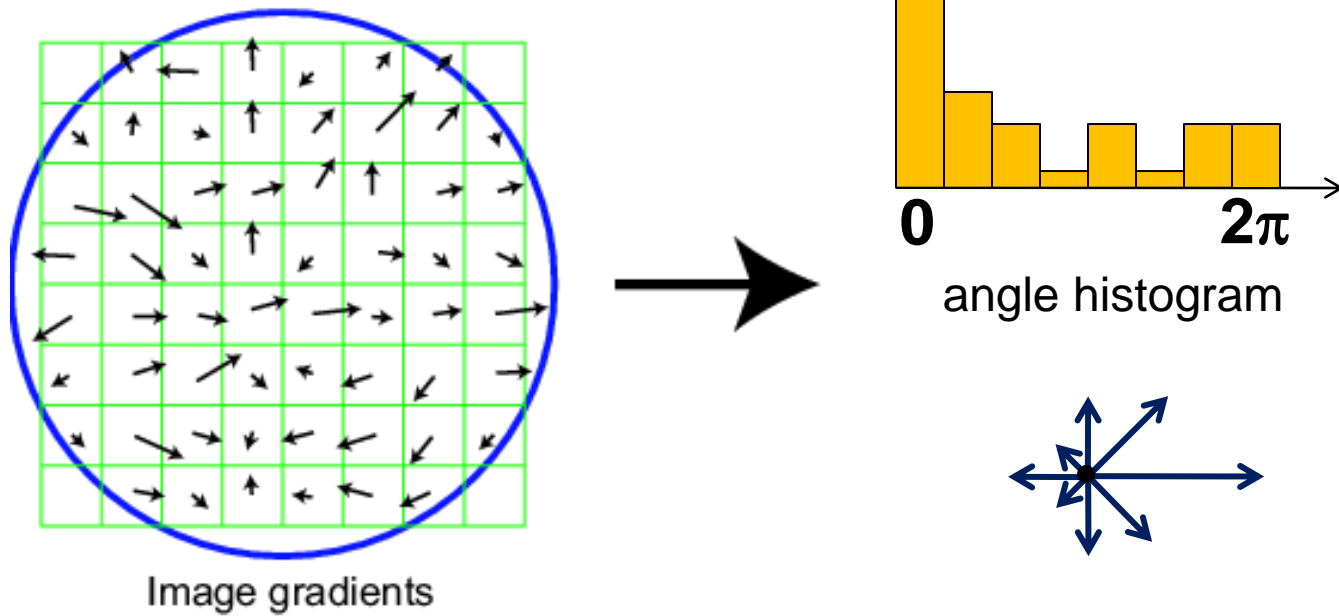
$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$



Scale Invariant Feature Transform

Basic idea:

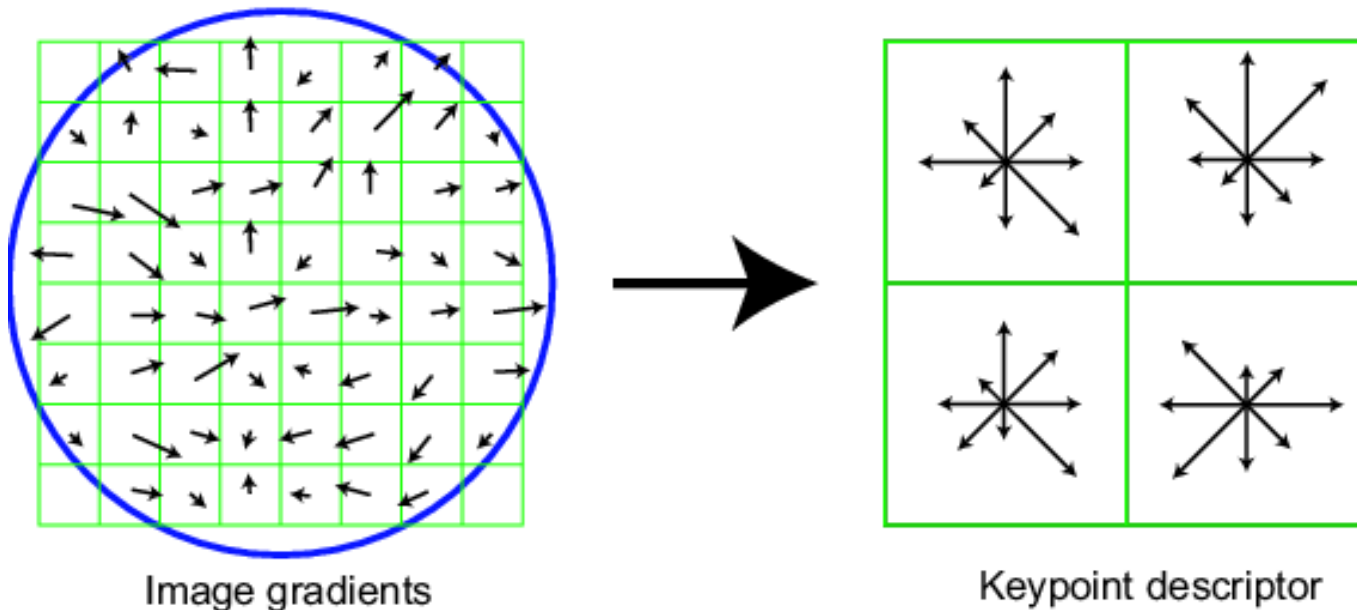
- Take 16x16 square window around detected feature
- Compute gradient orientation for each pixel
- Create histogram over edge orientations weighted by magnitude
- That's your feature descriptor!












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
- $16 \text{ cells} * 8 \text{ orientations} = 128 \text{ dimensional descriptor for each detected feature}$

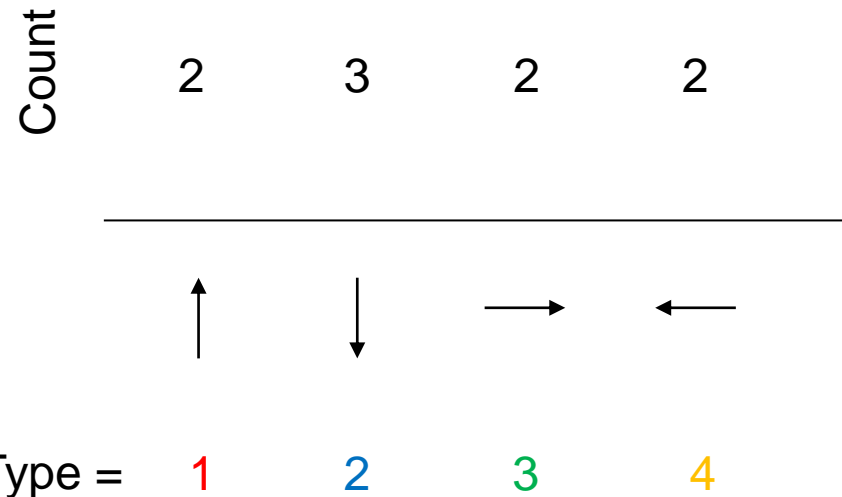


Scale Invariant Feature Transform

 1	 3	 1
 2	 3	 2
 4	 2	 4










Gradients

Uniform weight (ignore magnitude)



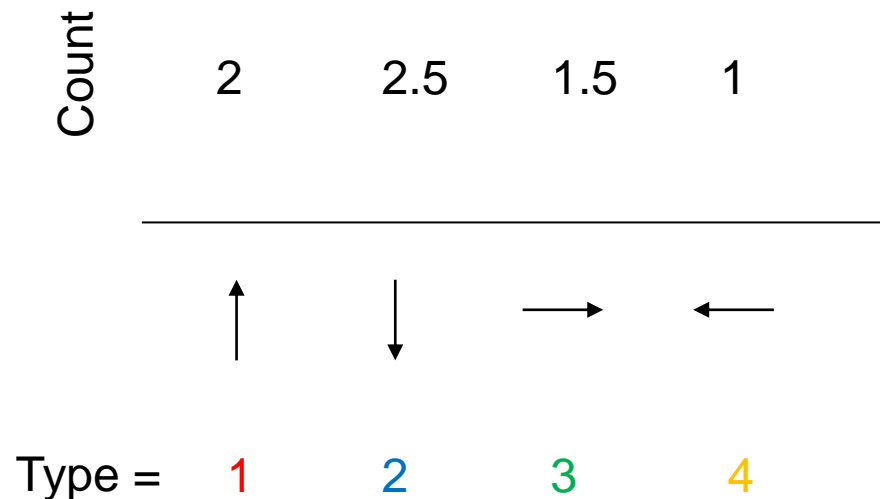
Histogram of gradients

Scale Invariant Feature Transform

 1	 3	 1
 2	 3	 2
 4	 2	 4

Gradients

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



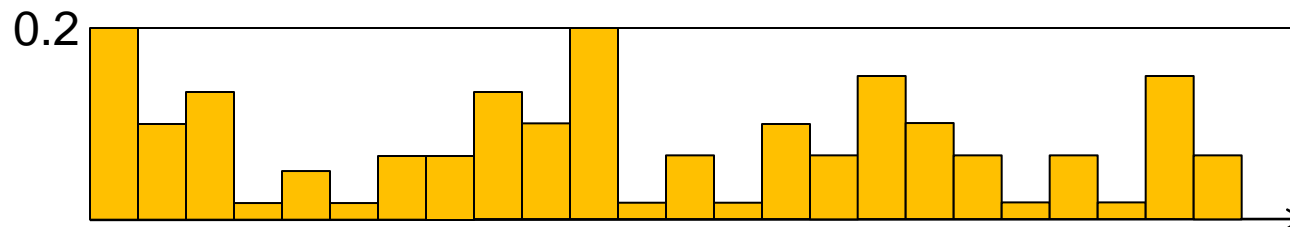
Histogram of gradients

Scale Invariant Feature Transform

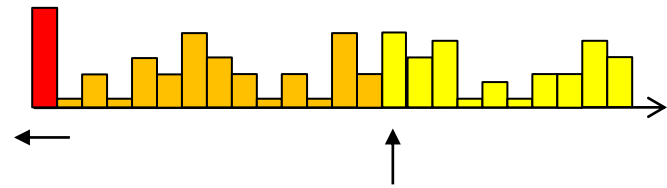
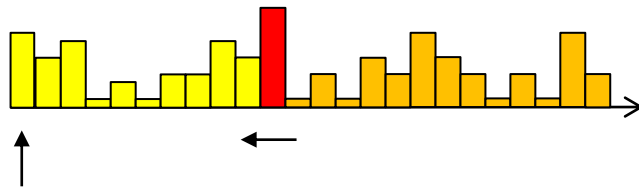
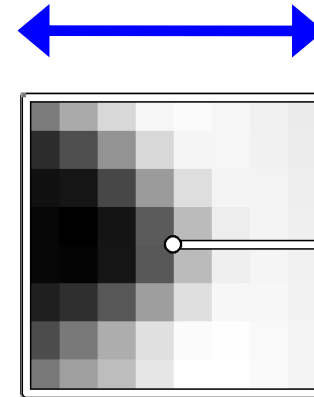
Full version

- 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
- We want:

$$\sum_i d_i = 1 \quad \text{such that: } d_i < 0.2$$



Making descriptor rotation invariant

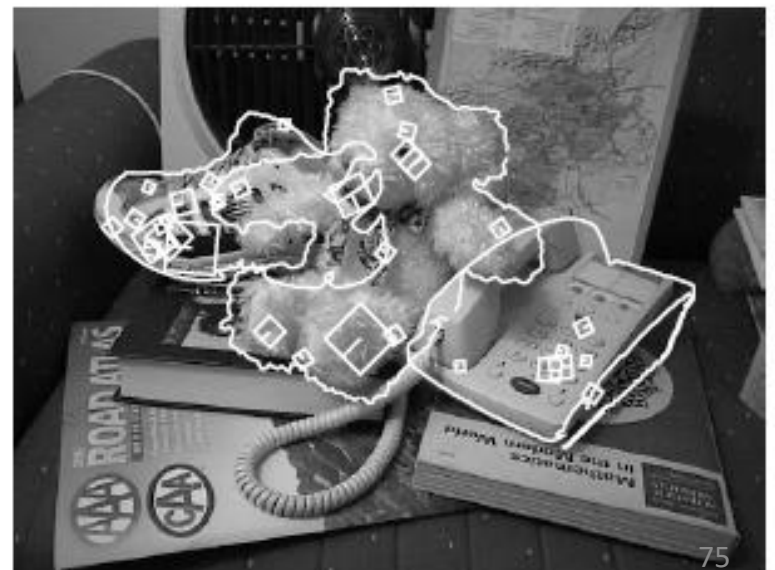


- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation

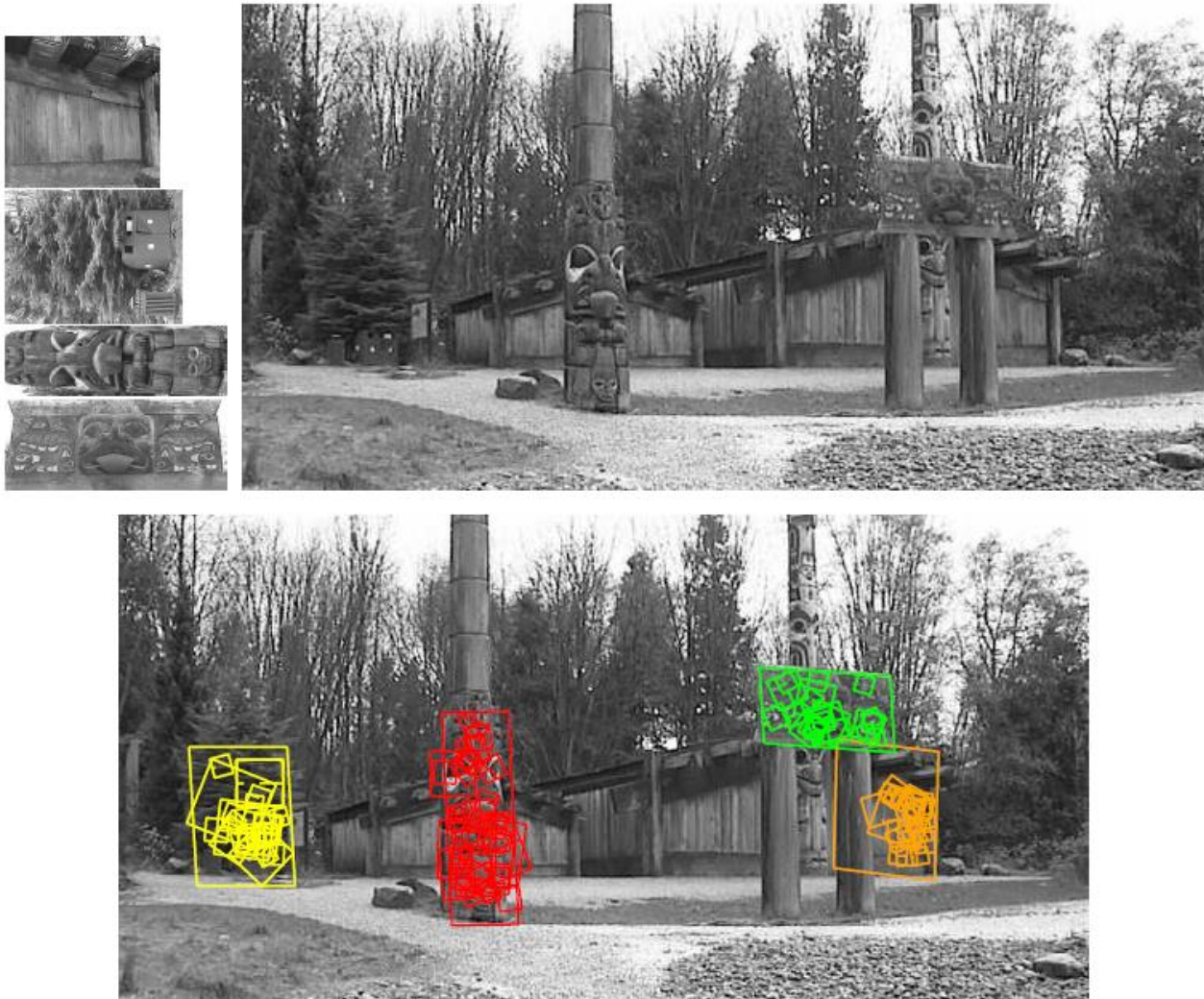
SIFT is robust

- 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>

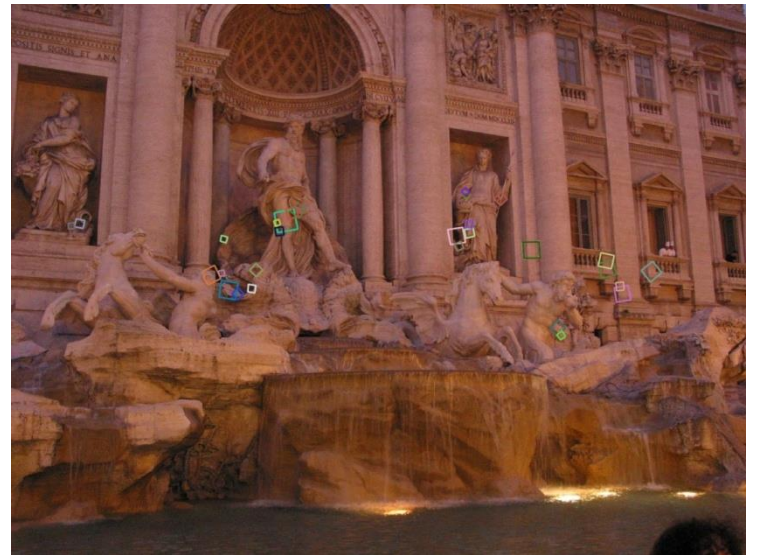
Examples of using SIFT



Examples of using SIFT

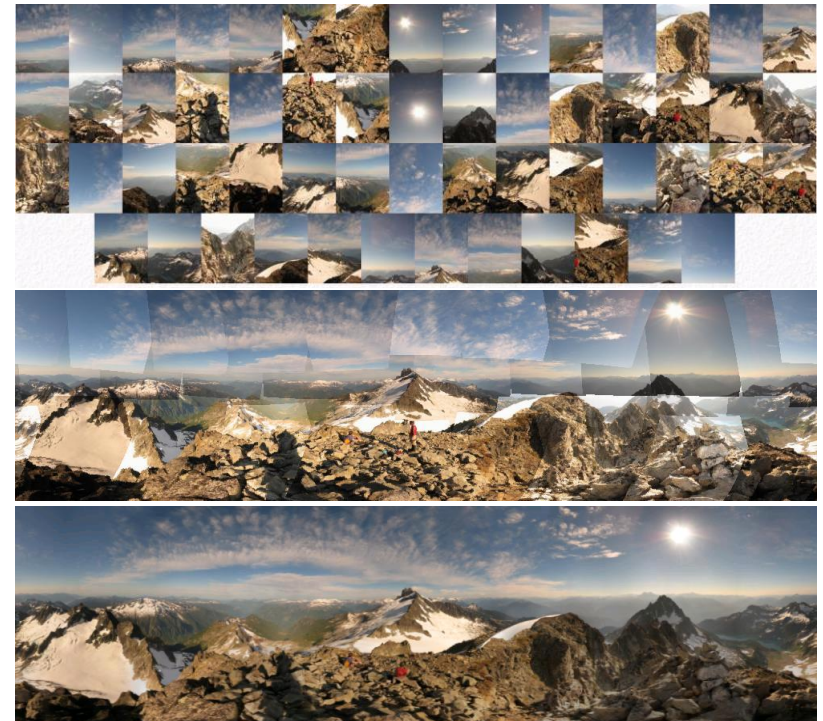


Examples of using SIFT



Applications of local invariant features

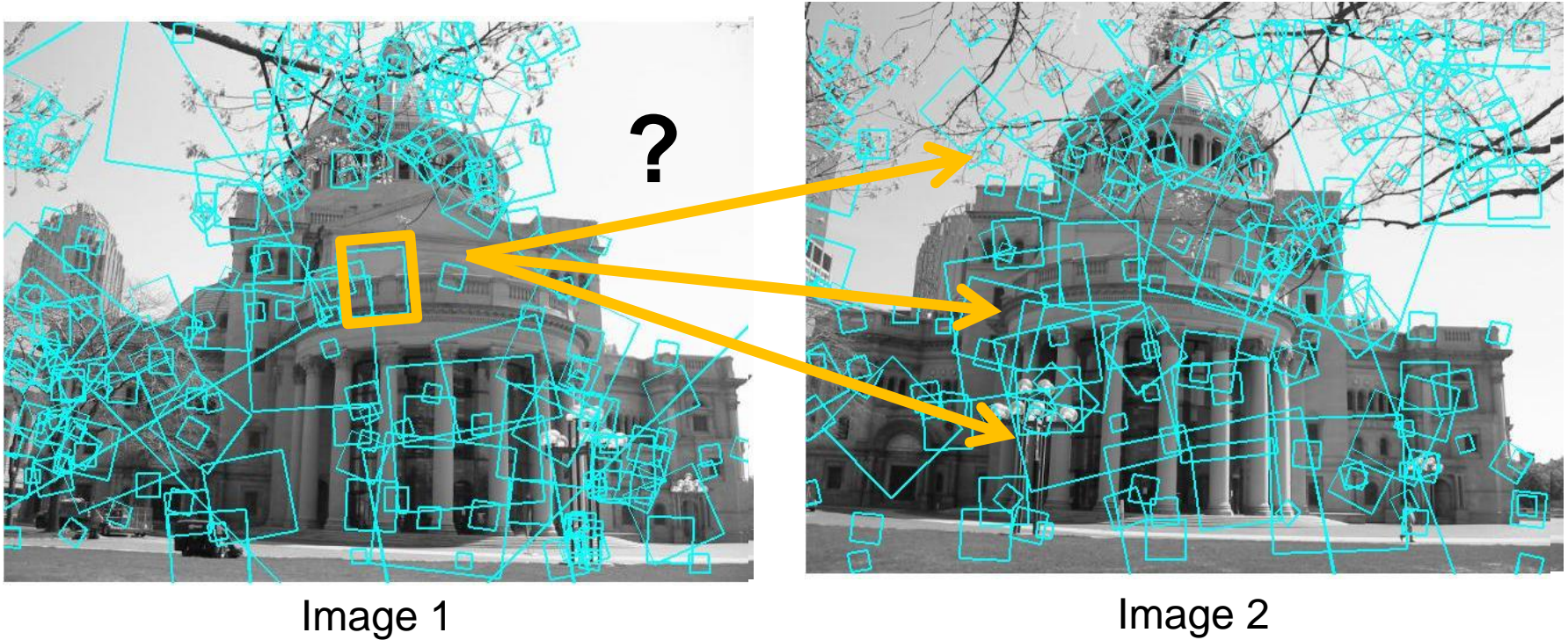
- Object recognition
- Indexing and retrieval
- Robot navigation
- 3D reconstruction
- Motion tracking
- Image alignment
- Panoramas and mosaics
- ...



Plan for this lecture

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

Matching local features



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

Robust matching



Image 1



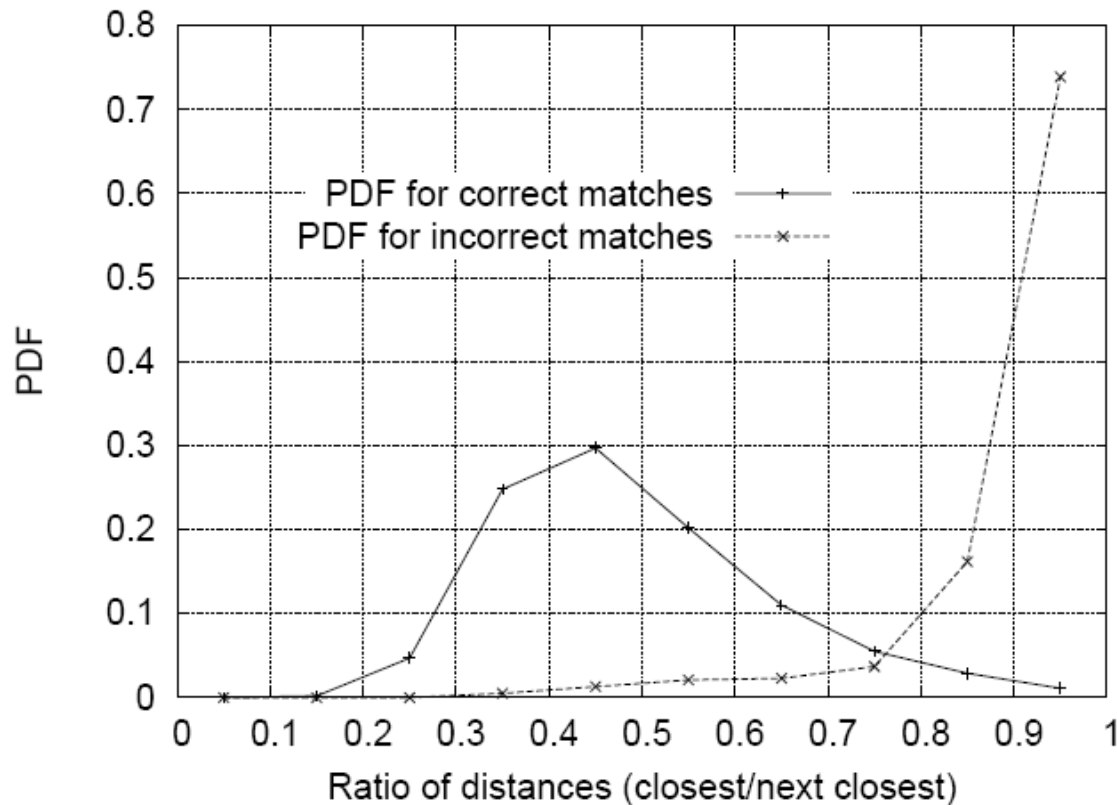
Image 2

- 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

$$d(q, f_1) / d(q, f_2)$$

Matching SIFT descriptors

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

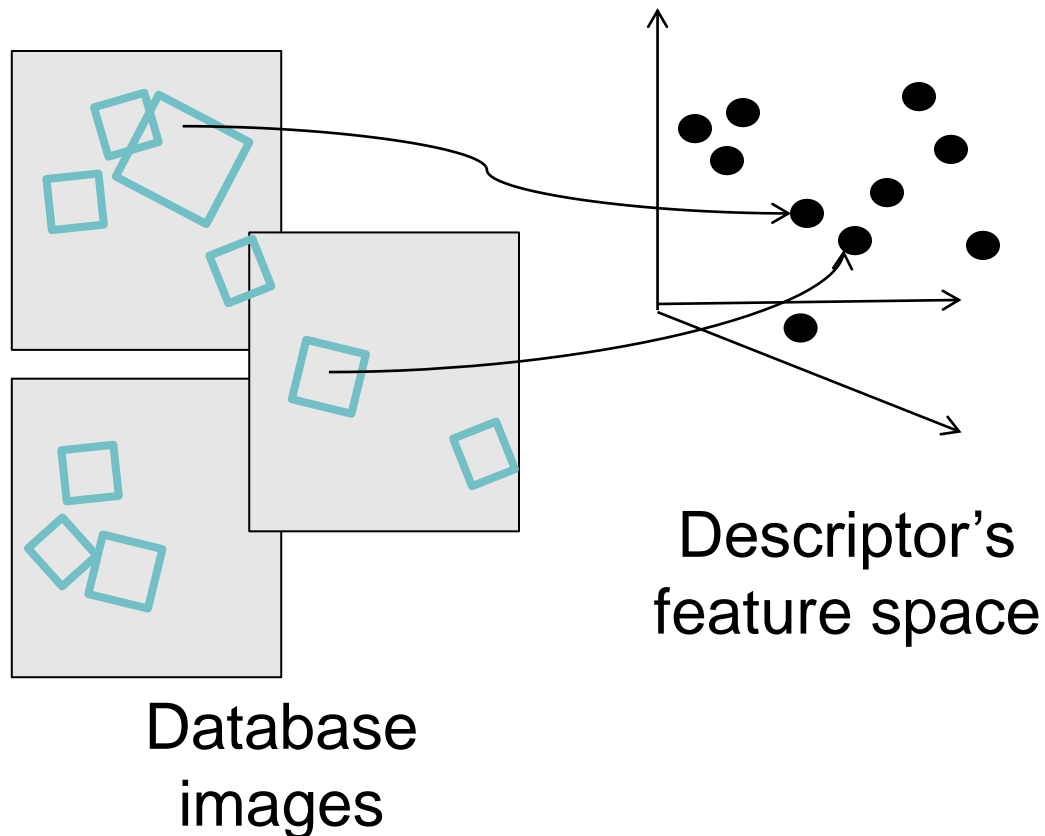


Efficient matching

- 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?

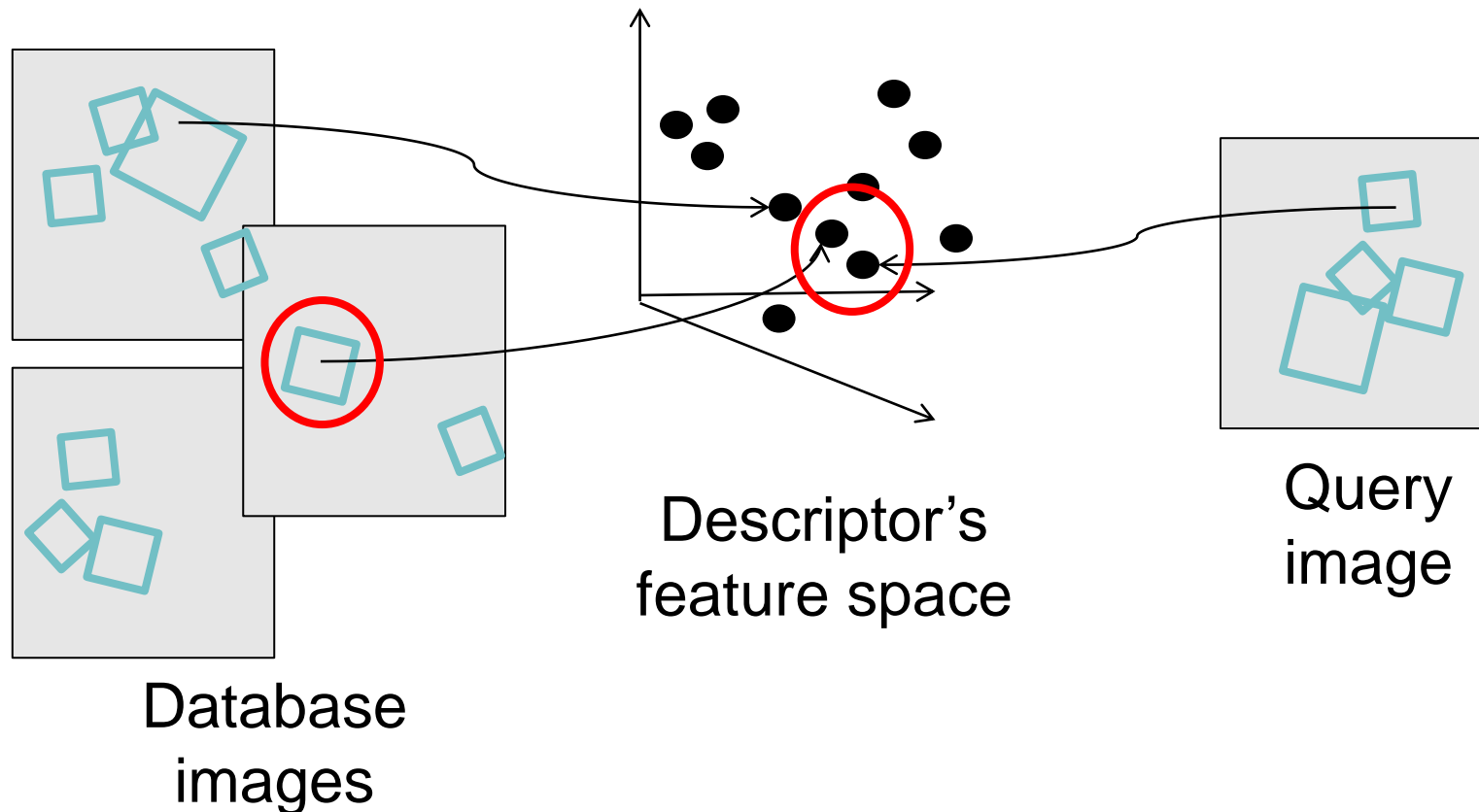
Matching local features: Setup

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



Matching local features: Setup

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



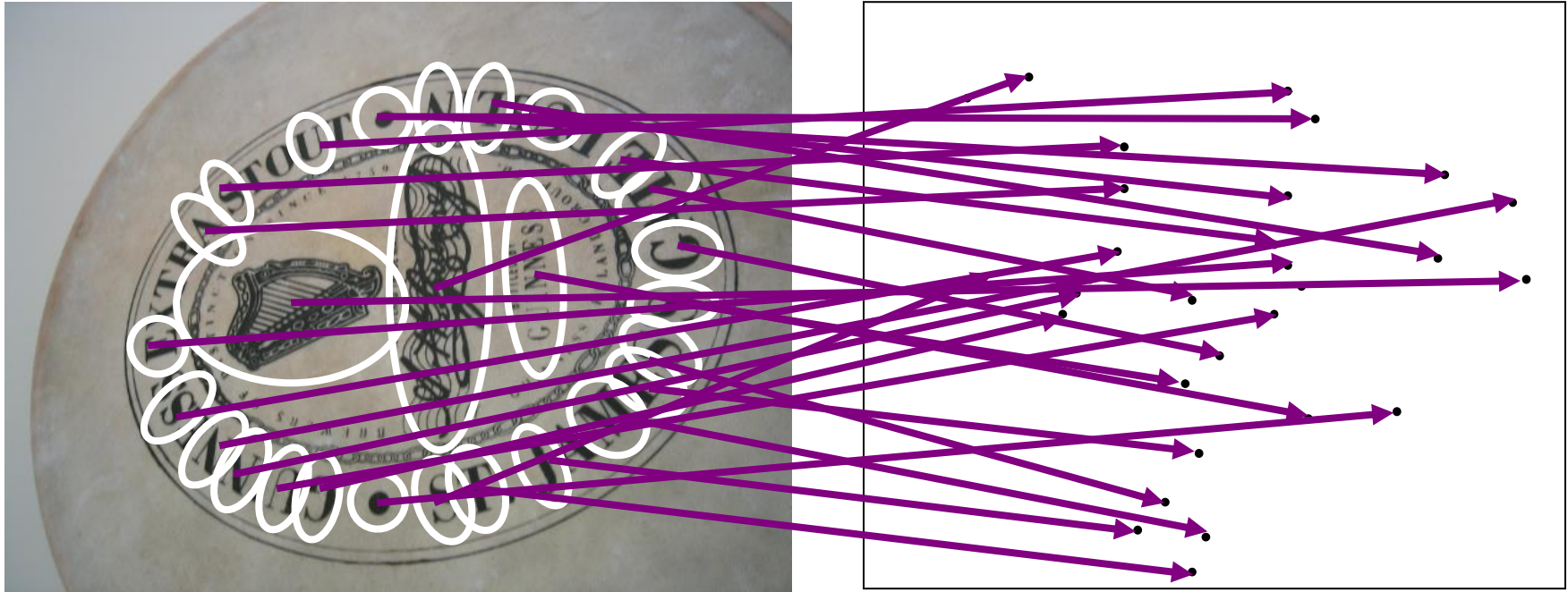
Indexing local features

Index		
"Along I-75," From Detroit to Florida; <i>inside back cover</i>	Butterfly Center, McGuire; 134	Driving Lanes; 85
"Drive I-95," From Boston to Florida; <i>inside back cover</i>	CAA (see AAA)	Duval County; 163
1929 Spanish Trail Roadway; 101-102,104	CCC, The; 111,113,115,135,142	Eau Gallie; 175
511 Traffic Information; 83	Ca d'Zan; 147	Edison, Thomas; 152
A1A (Barrier Isl) - I-95 Access; 86	Caloosahatchee River; 152	Eglin AFB; 116-118
AAA (and CAA); 83	Name; 150	Eight Reale; 176
AAA National Office; 88	Canaveral Natnl Seashore; 173	Ellenton; 144-145
Abbreviations,	Cannon Creek Airpark; 130	Emanuel Point Wreck; 120
Colored 25 mile Maps; cover	Canopy Road; 106,169	Emergency Callboxes; 83
Exit Services; 196	Cape Canaveral; 174	Epiphytes; 142,148,157,159
Travelogue; 85	Castillo San Marcos; 169	Escambia Bay; 119
Africa; 177	Cave Diving; 131	Bridge (I-10); 119
Agricultural Inspection Strns; 126	Cayo Costa, Name; 150	County; 120
Ah-Tah-Thi-Ki Museum; 160	Celebration; 93	Estero; 153
Air Conditioning, First; 112	Charlotte County; 149	Everglade,90,95,139-140,154-160
Alabama; 124	Charlotte Harbor; 150	Draining of; 156,181
Alachua; 132	Chautauqua; 116	Wildlife MA; 160
County; 131	Chipay; 114	Wonder Gardens; 154
Alafia River; 143	Name; 115	Falling Waters SP; 115
Alapaha, Name; 126	Choctawatchee, Name; 115	Fantasy of Flight; 95
Alfred B Maclay Gardens; 106	Circus Museum, Ringling; 147	Fayer Dykes SP; 171
Alligator Alley; 154-155	Citrus; 88,97,130,136,140,180	Fires, Forest; 166
Alligator Farm, St Augustine; 169	CityPlace, W Palm Beach; 180	Fires, Prescribed ; 148
Alligator Hole (definition); 157	City Maps,	Fisherman's Village; 151
Alligator, Buddy; 155	Fl Lauderdale Expwys; 194-195	Flagler County; 171
Alligators; 100,135,138,147,156	Jacksonville; 163	Flagler, Henry; 97,165,167,171
Anastasia Island; 170	Kissimmee Expwys; 192-193	Florida Aquarium; 186
Anhaica; 108-109,146	Miami Expressways; 194-195	Florida,
Apalachicola River; 112	Orlando Expressways; 192-193	12,000 years ago; 187
Appleton Mus of Art; 136	Pensacola; 26	Cavern SP; 114
Aquifer; 102	Tallahassee; 191	Map of all Expressways; 2-3
Arabian Nights; 94	Tampa-St. Petersburg; 63	Mus of Natural History; 134
Art Museum, Ringling; 147	St. Augustine; 191	National Cemetery ; 141
Aruba Beach Cafe; 183	Civil War; 100,108,127,138,141	Part of Africa; 177
Aucilla River Project; 106	Clearwater Marine Aquarium; 187	Platform; 187
Babcock-Web WMA; 151	Collier County; 154	Sheriff's Boys Camp; 126
Bahia Mar Marina; 184	Collier, Barron; 152	Sports Hall of Fame; 130
Baker County; 99	Colonial Spanish Quarters; 168	Sun 'n Fun Museum; 97
Barefoot Mailmen; 182	Columbia County; 101,128	Supreme Court; 107
Barge Canal; 137	Coquina Building Material; 165	Florida's Turnpike (FTP). 178,189
Bee Line Expy; 80	Corkscrew Swamp, Name; 154	25 mile Strip Maps; 66
Belz Outlet Mall; 89	Cowboys; 95	Administration; 189
Bernard Castro; 136	Crab Trap II; 144	Coin System; 190
Big "I"; 165	Cracker, Florida; 88,95,132	Exit Services; 189
Big Cypress; 155,158	Crosstown Expy; 11,35,98,143	HEFT; 76,161,190
Big Foot Monster; 105	Cuban Bread; 184	History; 189
Billie Swamp Safari; 160	Dade Battlefield; 140	Names; 189
Blackwater River SP; 117	Dade, Maj. Francis; 139-140,161	Service Plazas; 190
Blue Angels	Dania Beach Hurricane; 184	Spur SR91; 76
	Daniel Boone, Florida Walk; 117	Ticket System; 190
	Daytona Beach; 172-173	Toll Plazas; 190
	De Land; 87	Ford, Henry; 152

- 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".

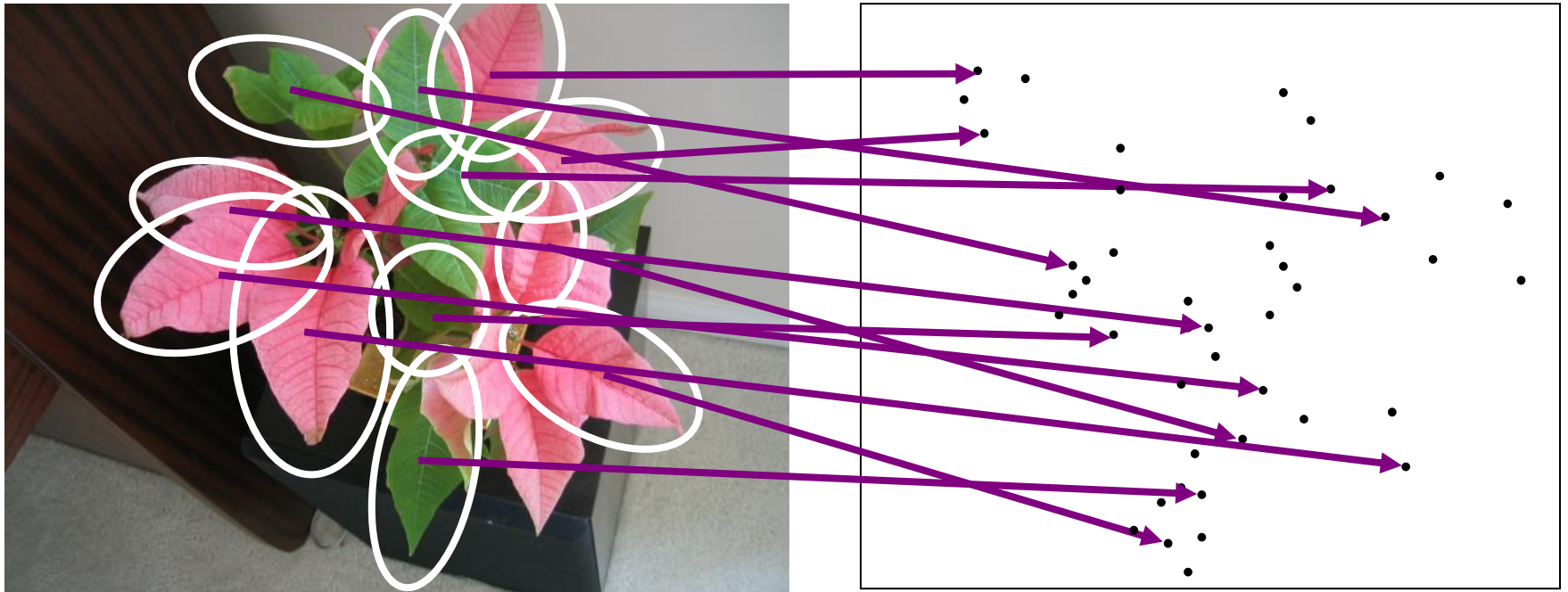
Visual words: Main idea

- Extract some local features from a number of images ...

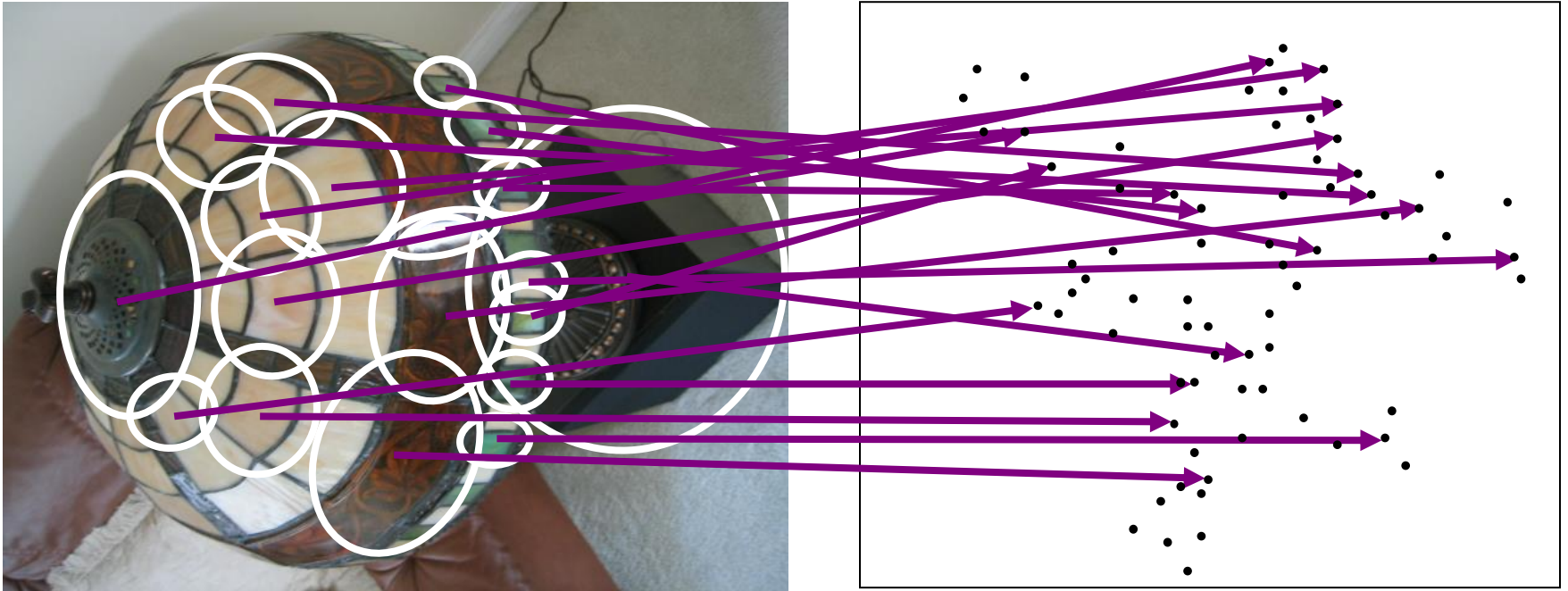


e.g., SIFT descriptor space: each point is 128-dimensional

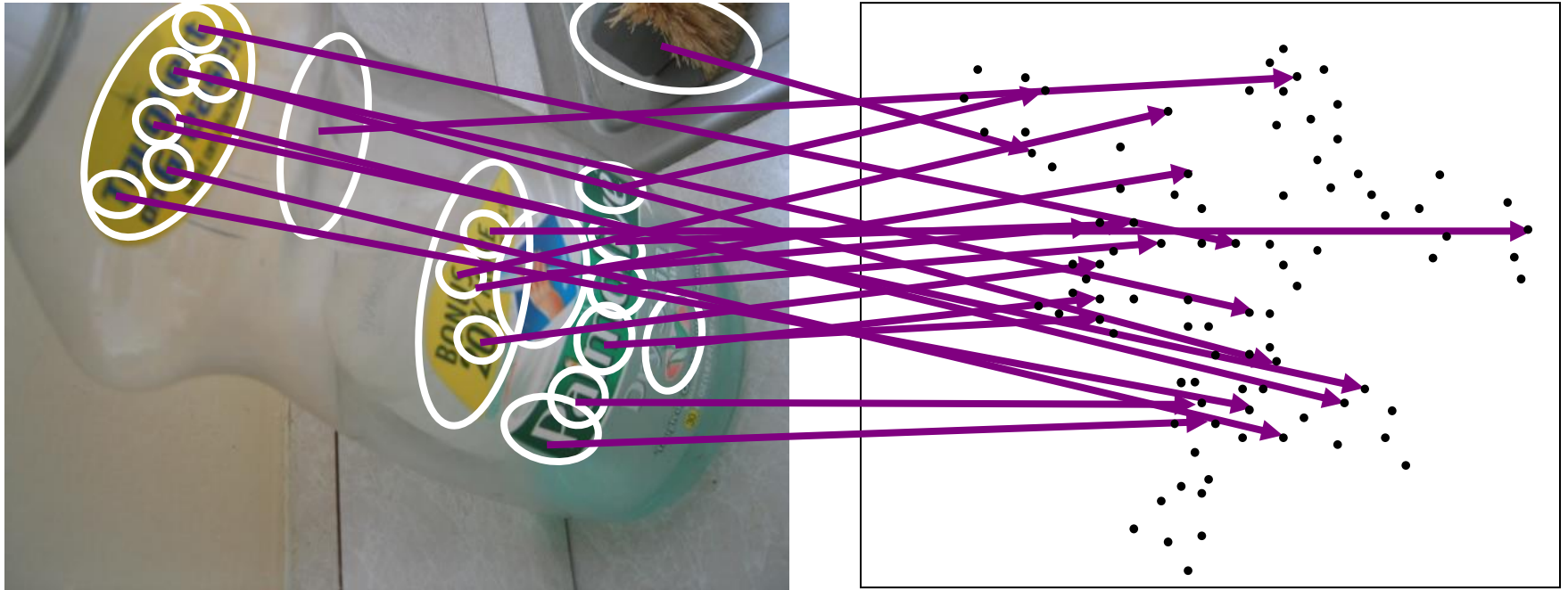
Visual words: Main idea

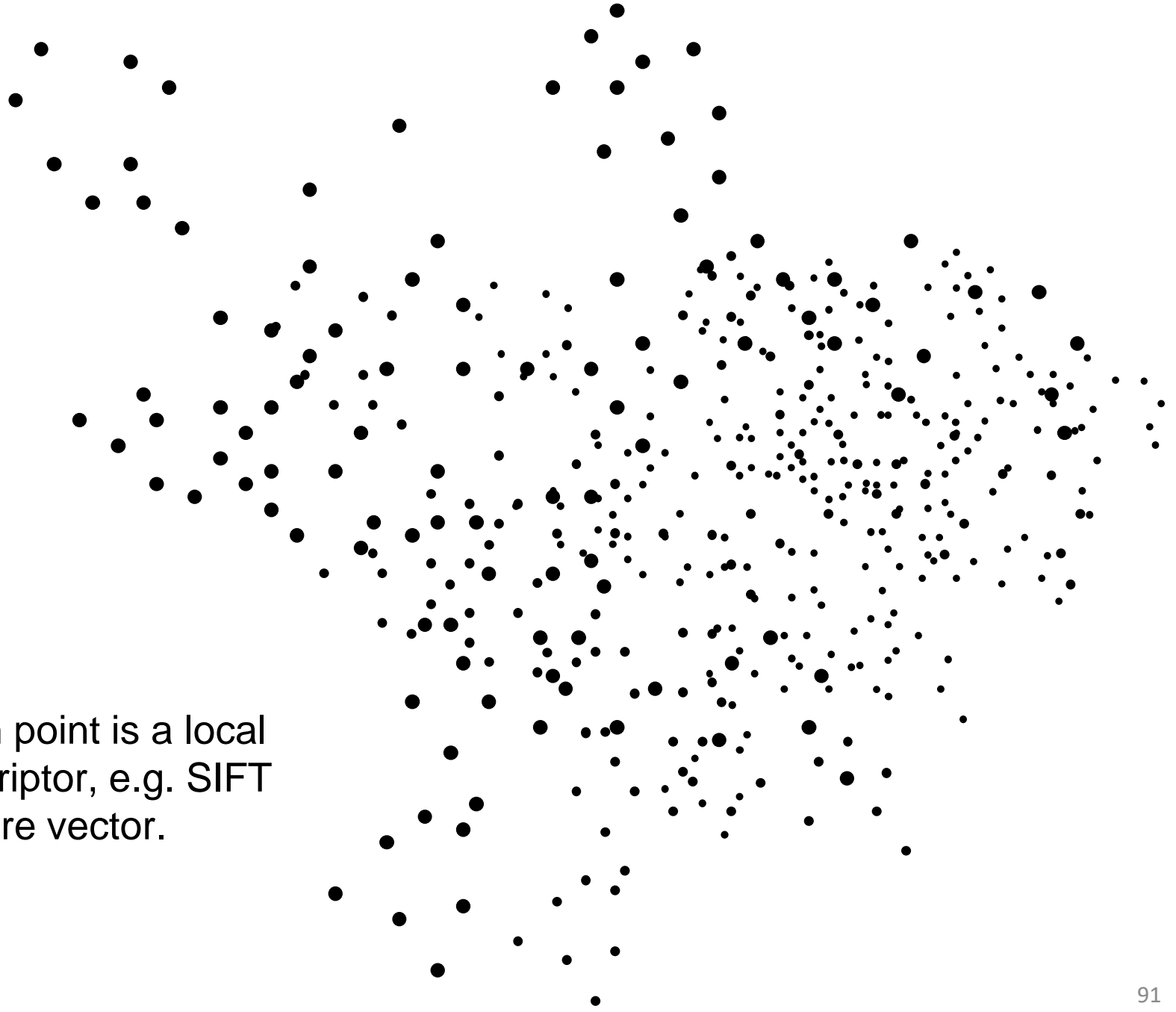


Visual words: Main idea

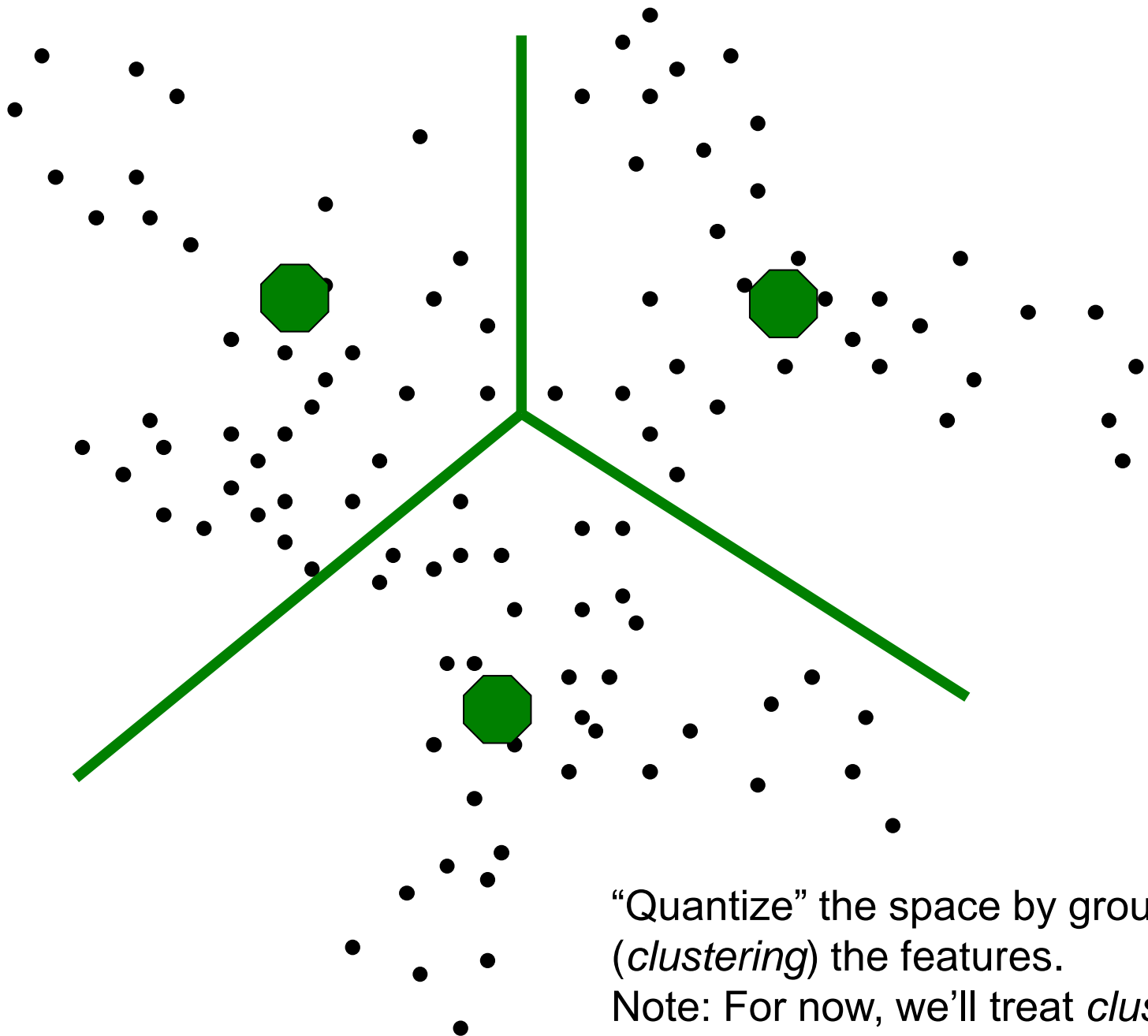


Visual words: Main idea



A scatter plot of black dots of varying sizes representing local descriptors like SIFT feature vectors. The dots are distributed across the upper and middle portions of the slide, with a higher density in the center-right area. The dots vary in size, with some being significantly larger than others, possibly indicating a magnitude or weight associated with each descriptor.

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”

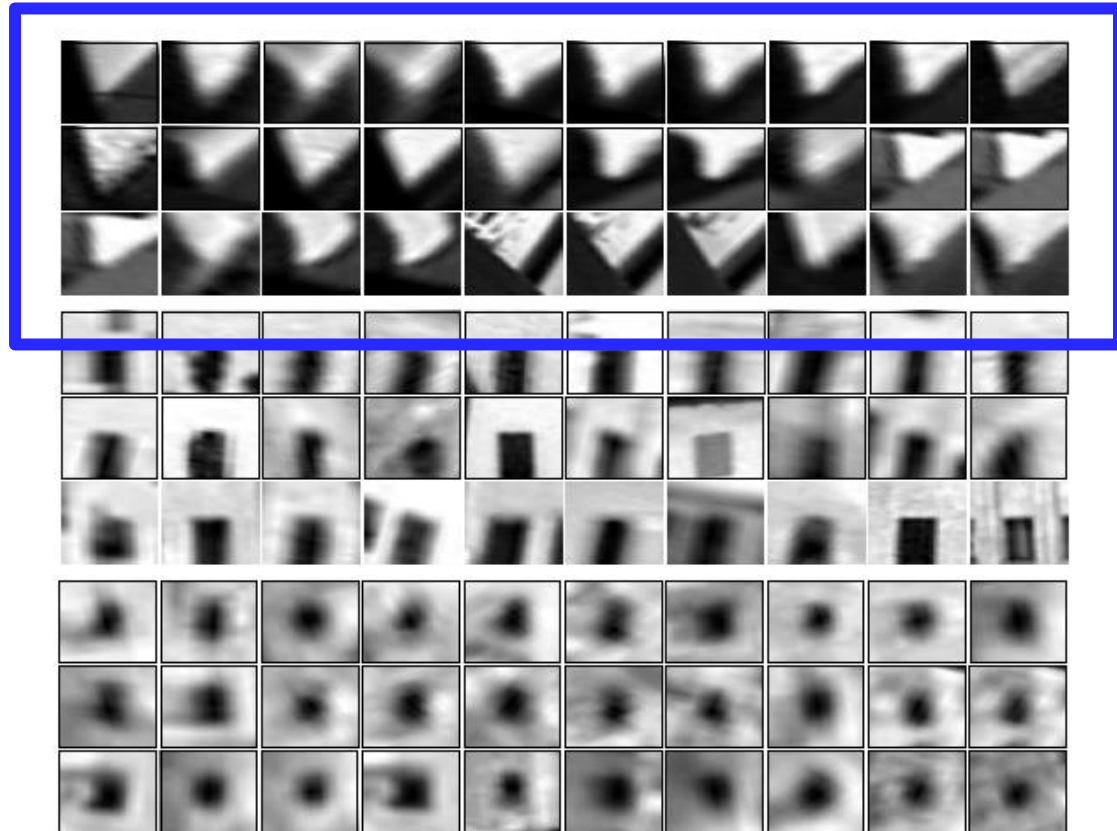
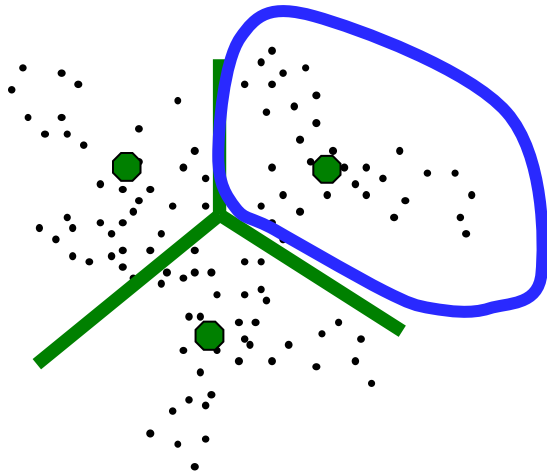
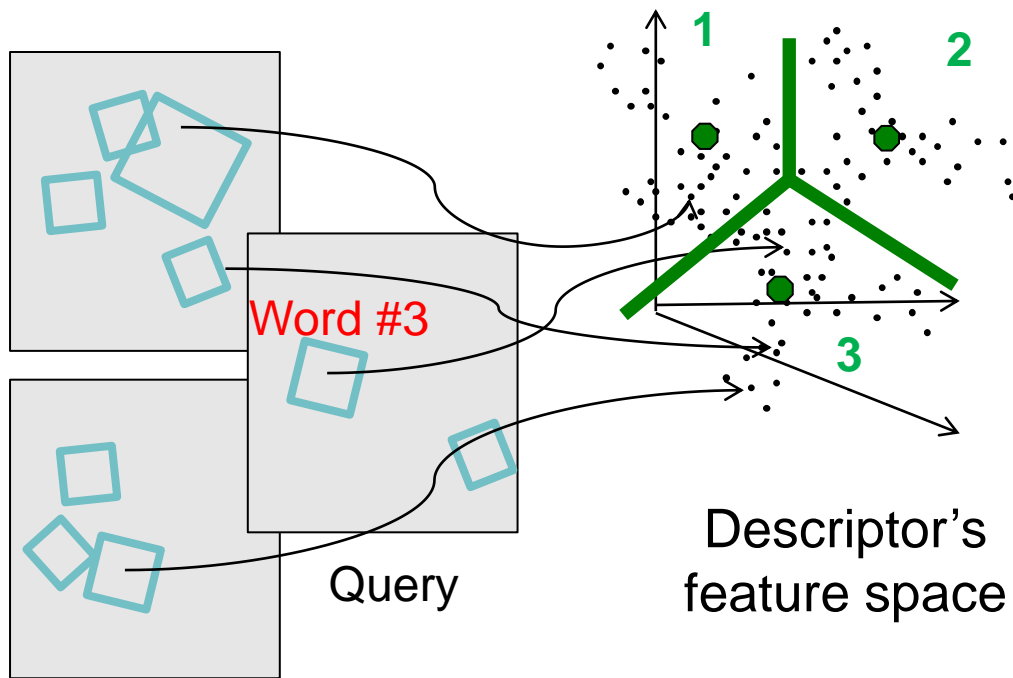


Figure from Sivic & Zisserman, ICCV 2003

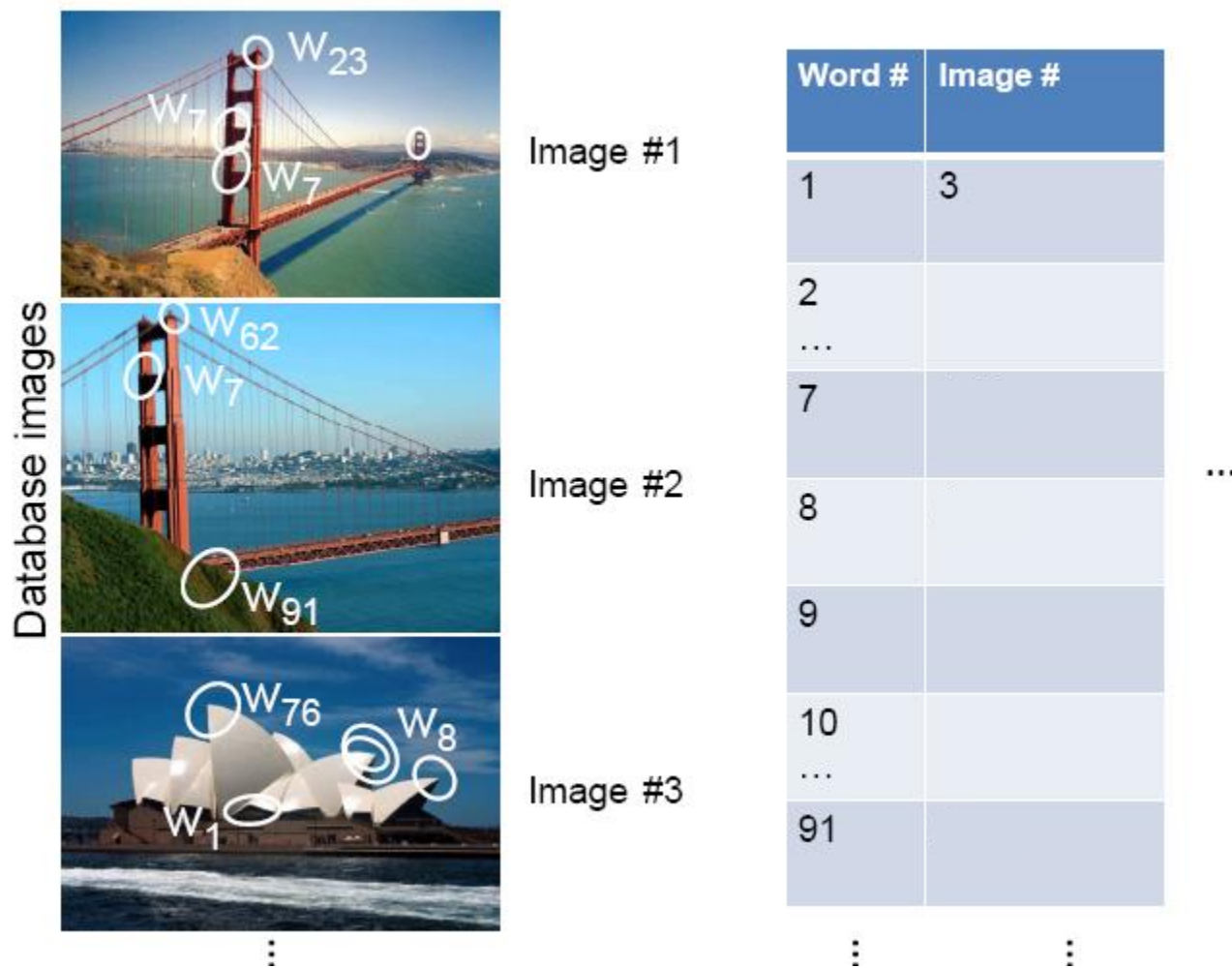
Visual words for indexing

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



- Each cluster has a center
- To determine which word to assign to new image region (e.g. query), find closest cluster center
- *To compare features:*
Only compare query to others in same cluster, or just compare word IDs
- *To compare images:*
see next few slides

Inverted file index



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

Inverted file index

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



New query image

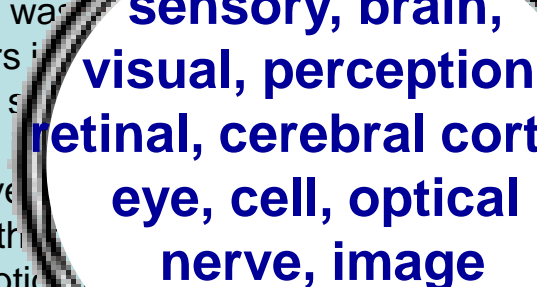
Word #	Image #
1	3
2	
...	
7	1, 2
8	3
9	
10	
...	
91	2



- 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 reach us through our eyes.



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

following the discovery of the visual cortex, Hubel and Wiesel have demonstrated that the message about the image falling on the retina undergoes a point-by-point analysis in a system of nerve cells stored in columns. In this system each cell has its specific function and is responsible for a specific detail in the pattern of the retinal image.

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% increase in exports to \$750bn, compared with \$575bn in 2004.

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

demand so much that it has caused inflation in the country. China has been reluctant to let the yuan against the dollar rise further, and has permitted it to trade within a narrow band, but the US wants the yuan to be allowed to trade freely. However, Beijing has made it clear that it will take its time and tread carefully before allowing the yuan to rise further in value.

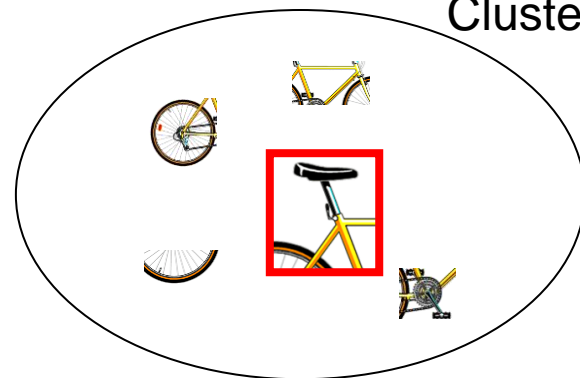
Describing images w/ visual words

- Summarize entire image based on its distribution (histogram) of word occurrences
- Analogous to bag of words representation commonly used for documents

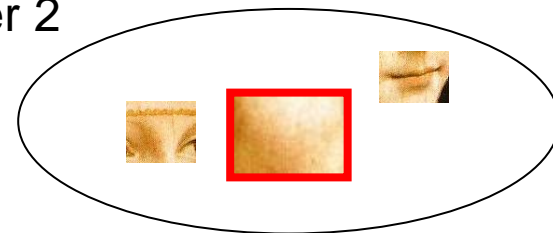
Feature patches:



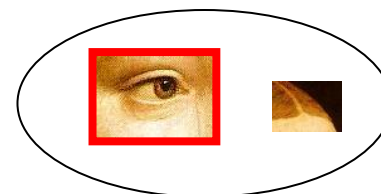
Cluster 1



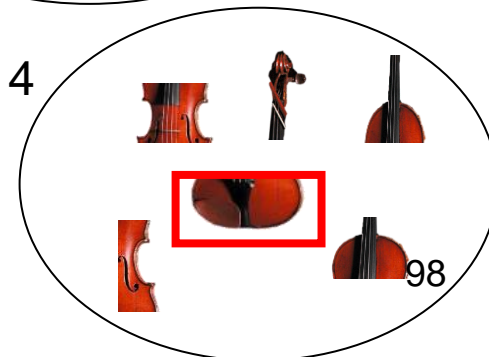
Cluster 2



Cluster 3



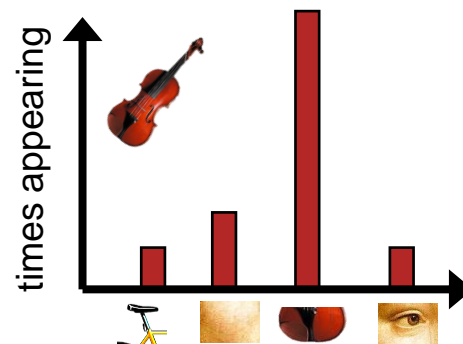
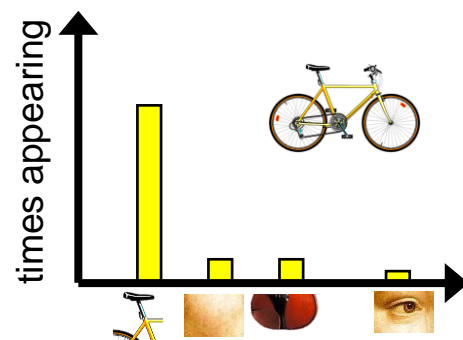
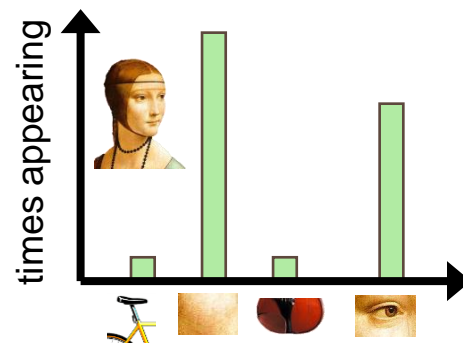
Cluster 4



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

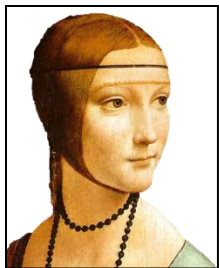
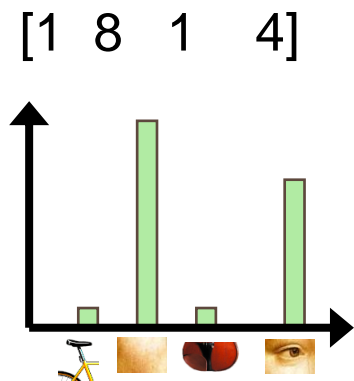
Feature patches:



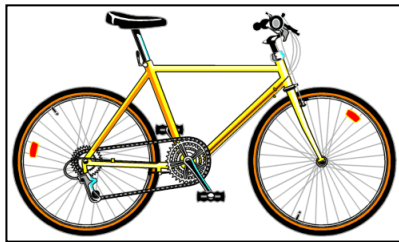
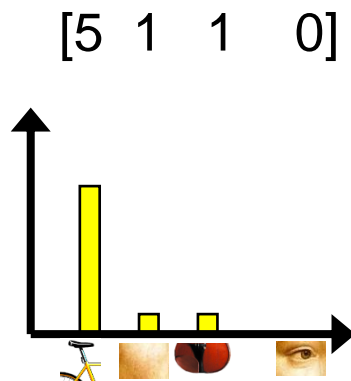
Visual words

Comparing bags of words

- Similarity of images measured as normalized scalar product between their word occurrence counts
- Can be used to rank results (nearest neighbors of query)



\vec{d}_j



\vec{q}

$$\text{sim}(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|}$$

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

for vocabulary of V words

Bags of words: pros and cons

- + flexible to geometry / deformations / viewpoint
- + compact summary of image content
- basic model ignores geometry – verify afterwards
- what is the optimal vocabulary size?
- background and foreground mixed when bag covers whole image

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

- Survey paper on local features
 - “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
- Making Harris detection scale-invariant
 - “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>
- SIFT paper by David Lowe
 - “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

