CS 1674/2074: Local features: detection, description and matching

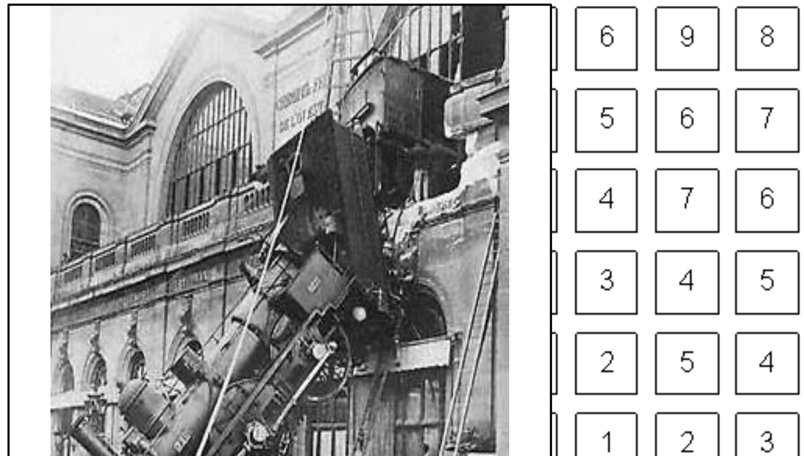
PhD. Nils Murrugarra-Llerena
nem177@pitt.edu

University of Pittsburgh

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

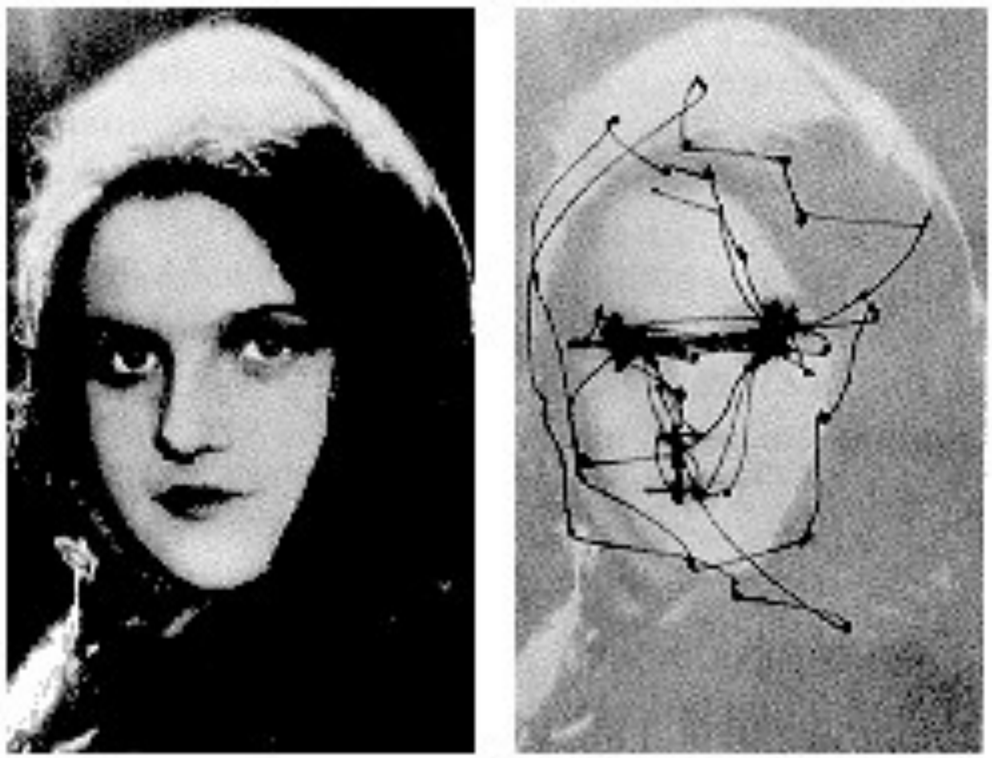


Adapted from S. Narasimhan

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

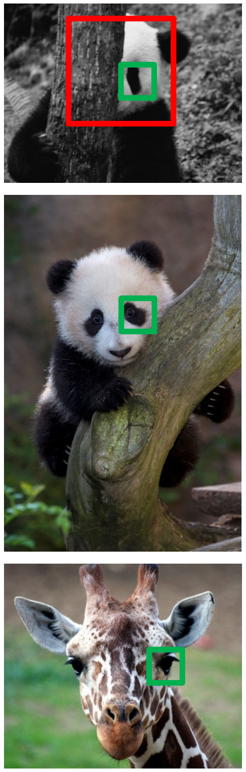
D. Hoiem

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 use those to compute a representation of the whole image
- Local features usually exploit image gradients, ignore color
- Feature ~= *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



Adapted from K. Grauman and D. Hoiem

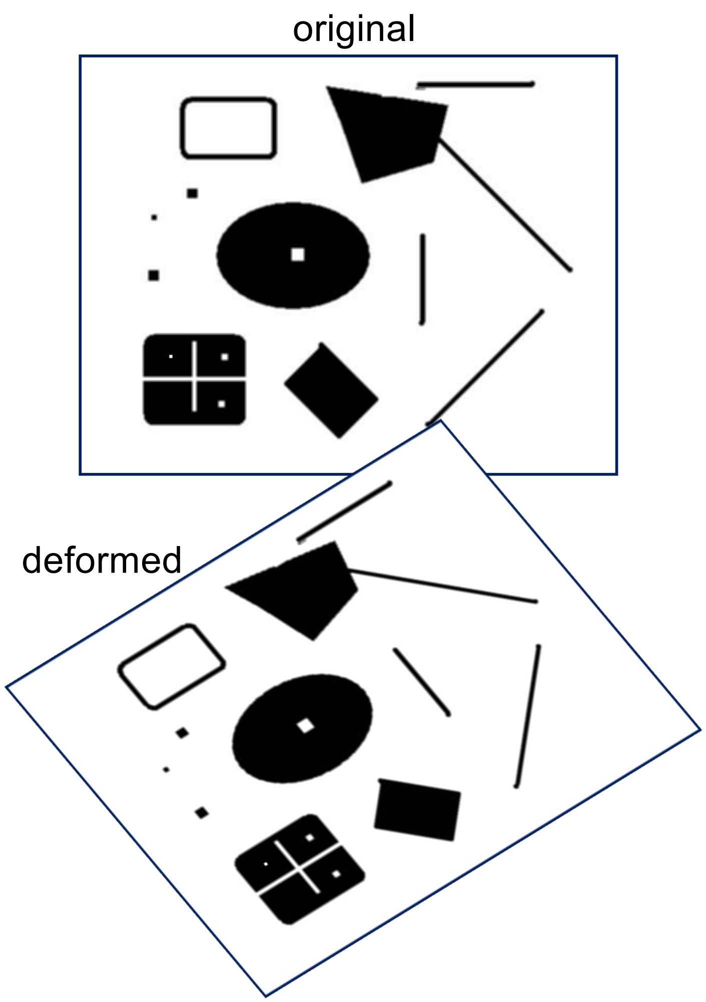
Interest(ing) points

- 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?

Adapted from D. Hoiem

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?



D. Hoiem

Choosing interest points

Where would you tell your friend to meet you?

→ Corner detection

D. Hoiem

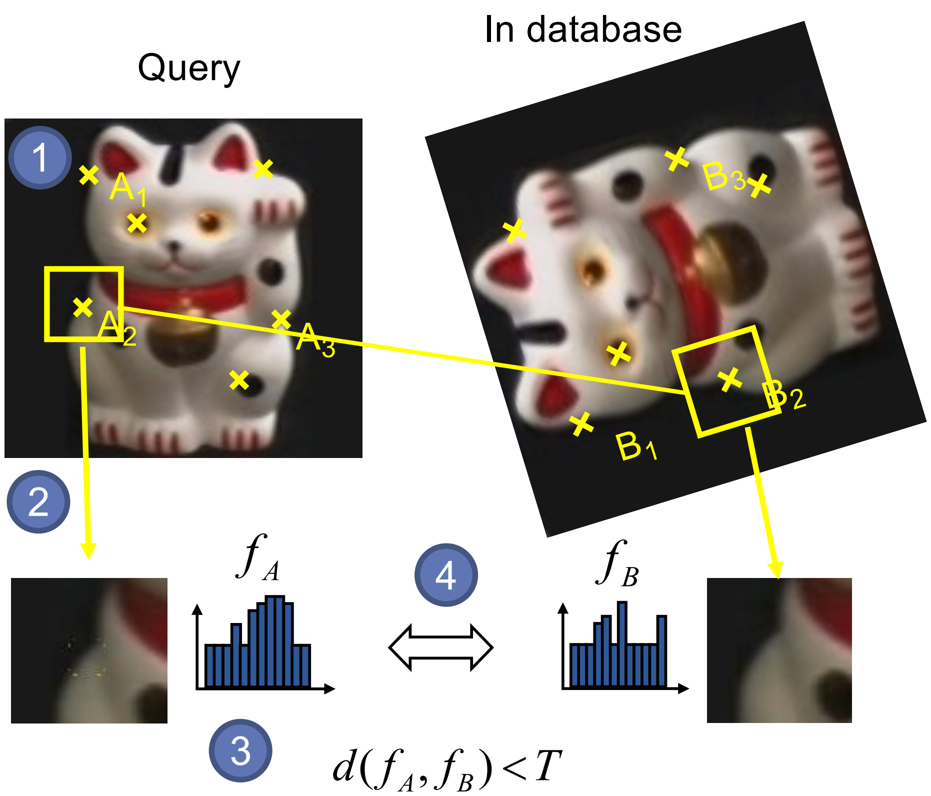
Choosing interest points

Where would you tell your friend to meet you?

→ Blob detection

D. Hoiem

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 content

Adapted from D. Hoiem

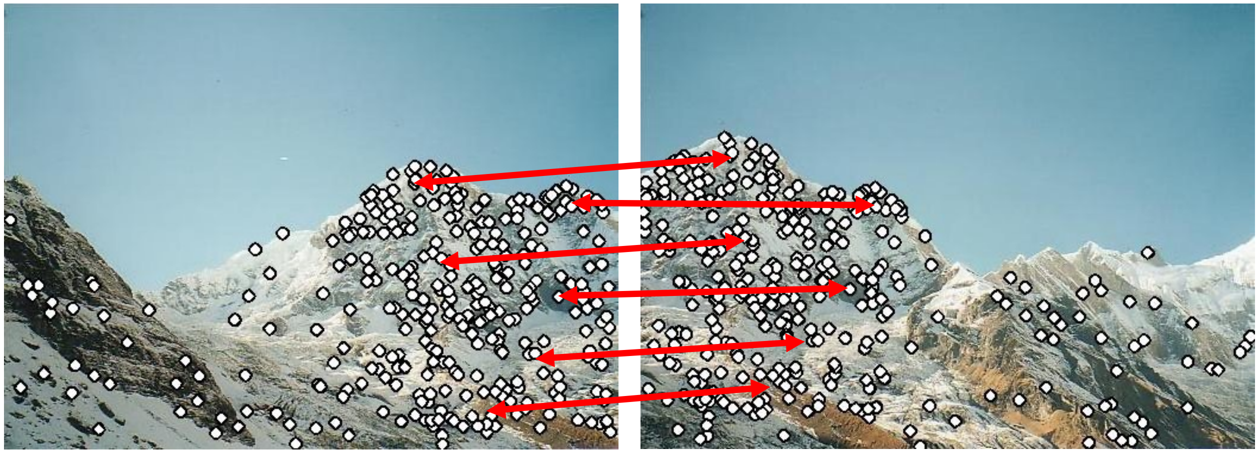
Application 2: Panorama stitching

- We have two images – how do we combine them?


L. Lazebnik

Application 2: Panorama stitching

- We have two images – how do we combine them?



Step 1: extract features

Step 2: match features

L. Lazebnik

Application 2: Panorama stitching

- We have two images – how do we combine them?

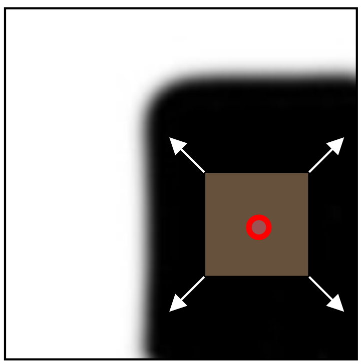
Step 1: extract features
Step 2: match features
Step 3: align images

L. Lazebnik

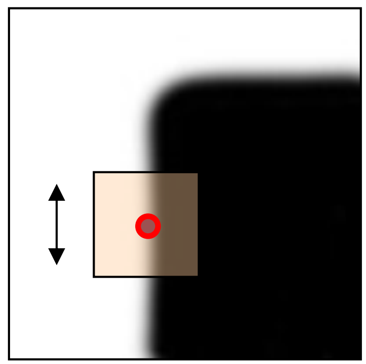
Corners are distinctive interest points

- 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

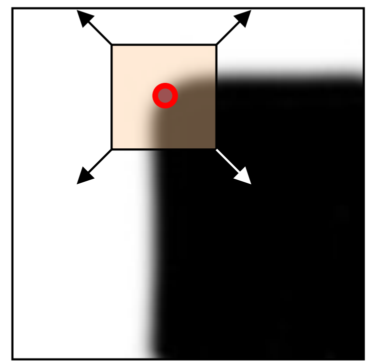
Candidate keypoint



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction

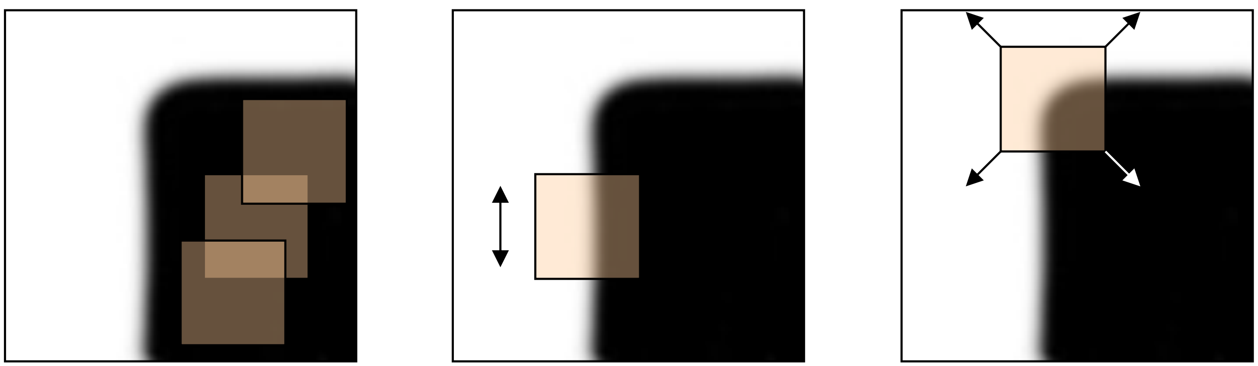


"corner":
significant change
in all directions

Adapted from A. Efros, D. Frolova, D. Simakov

Corners are 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

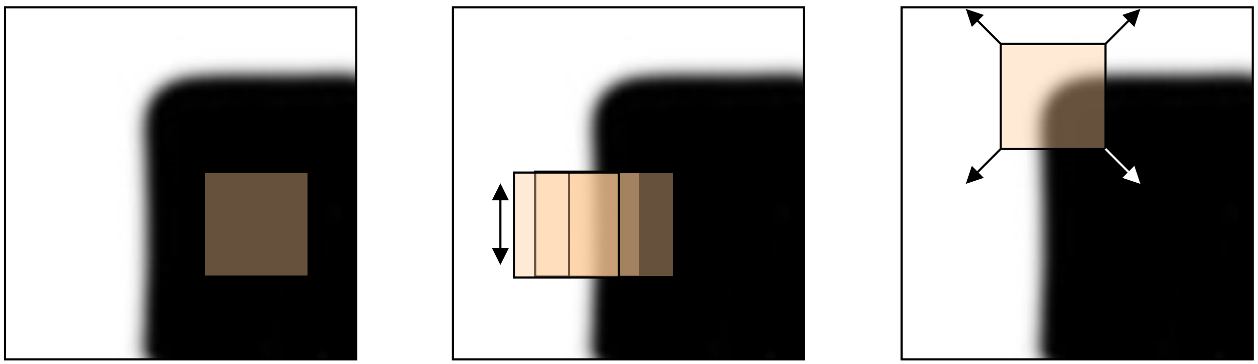
"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Adapted from A. Efros, D. Frolova, D. Simakov

Corners are 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

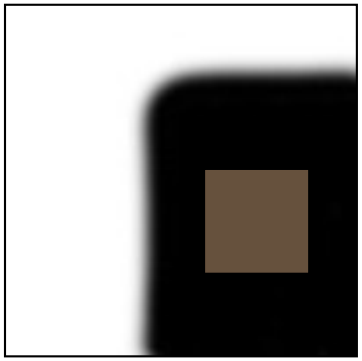
"edge":
no change along
the edge direction

"corner":
significant change
in all directions

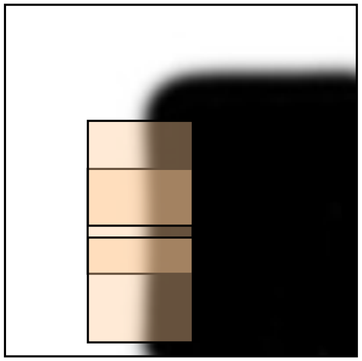
Adapted from A. Efros, D. Frolova, D. Simakov

Corners are distinctive interest points

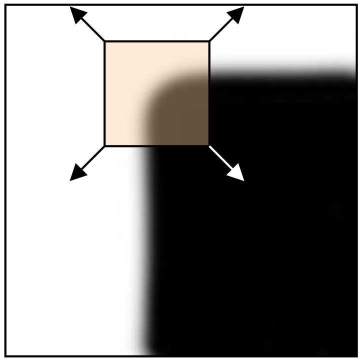
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



“flat” region:
no change in
all directions



“edge”:
no change along
the edge direction

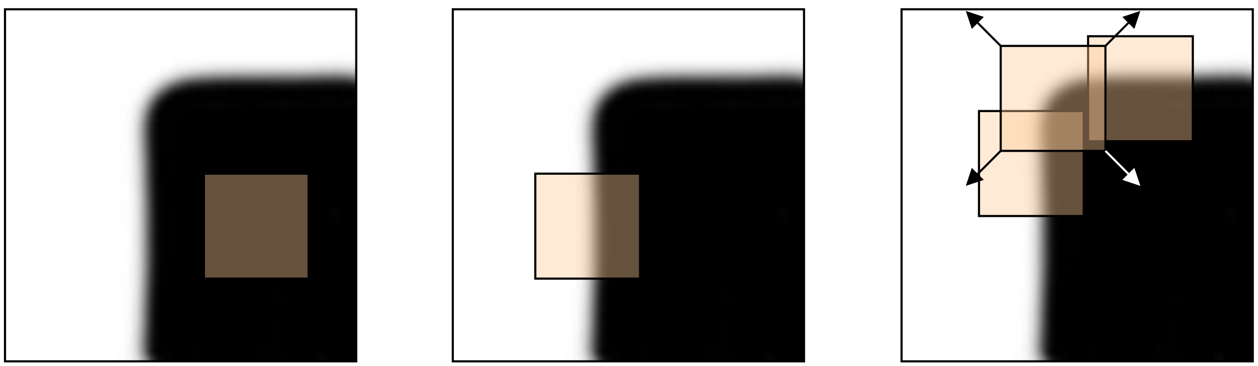


“corner”:
significant change
in all directions

Adapted from A. Efros, D. Frolova, D. Simakov

Corners are 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

Adapted from A. Efros, D. Frolova, D. Simakov

What points would you choose?

K. Grauman

Harris Detector: Mathematics

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

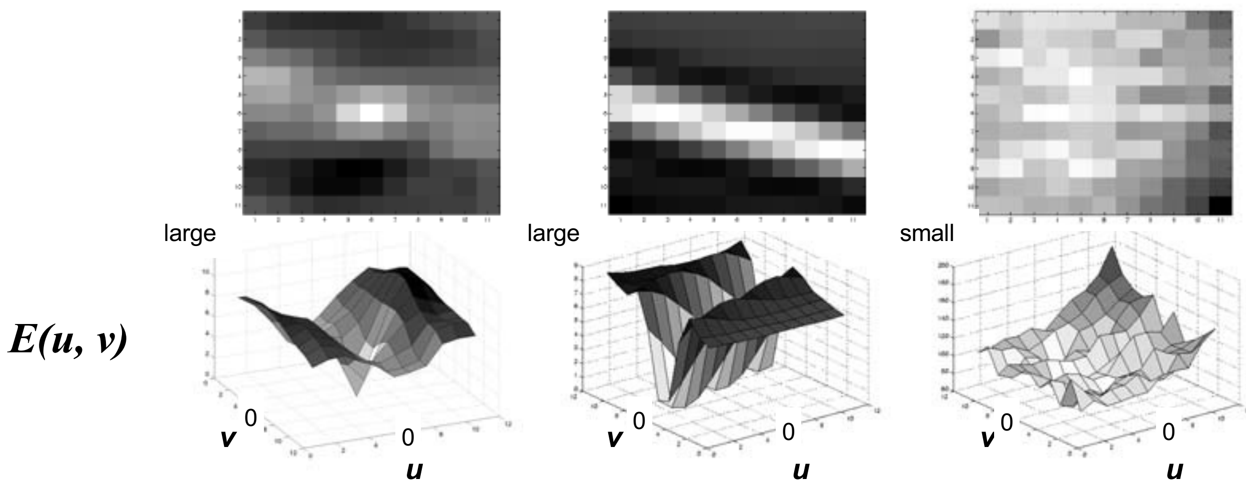
$$E(u,v) = \sum_{x,y} \left[\underbrace{I(x+u, y+v)}_{\text{Shifted intensity}} - \underbrace{I(x,y)}_{\text{Intensity}} \right]^2$$

Adapted from D. Frolova, D. Simakov

Harris Detector: Mathematics

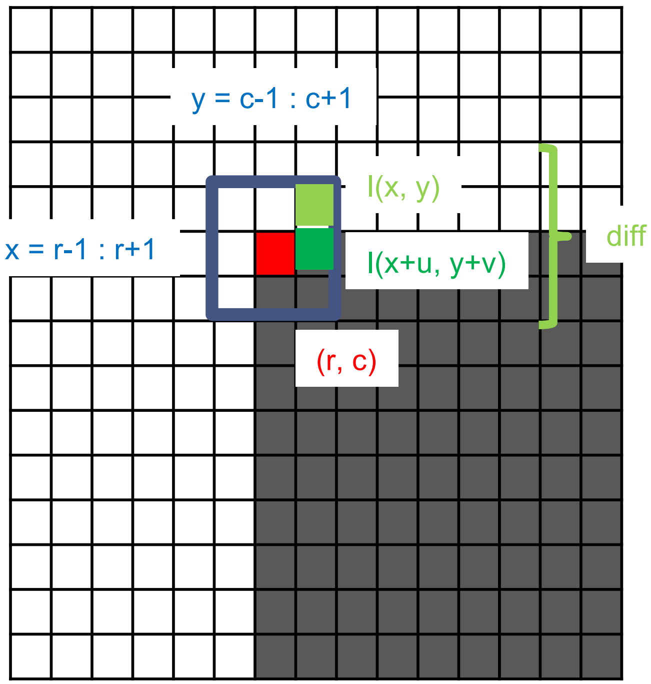
Window-averaged squared change of intensity induced by shifting the patch for a fixed candidate keypoint by [u,v]:

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



Adapted from D. Frolova, D. Simakov

Harris Detector: Mathematics



Here u = 1, v = 0

For every pixel (r, c) as candidate keypoint

Initialize E = zeros(2*max_offset+1, 2*max_offset+1)

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)

[Github repo] – Module 3

Adriana Kovashka

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:

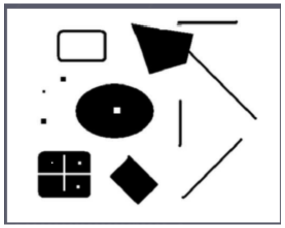
$$M = \sum_{x,y} \begin{bmatrix} I_h^2(x,y) & I_h I_v(x,y) \\ I_h I_v(x,y) & I_v^2(x,y) \end{bmatrix}$$

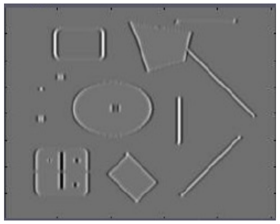
Adapted from D. Frolova, D. Simakov

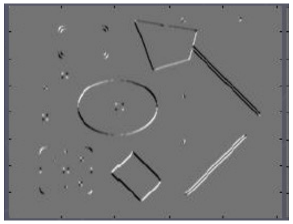
Harris Detector: Mathematics

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

$I_h^2 \rightarrow I_h^2(x, y)$





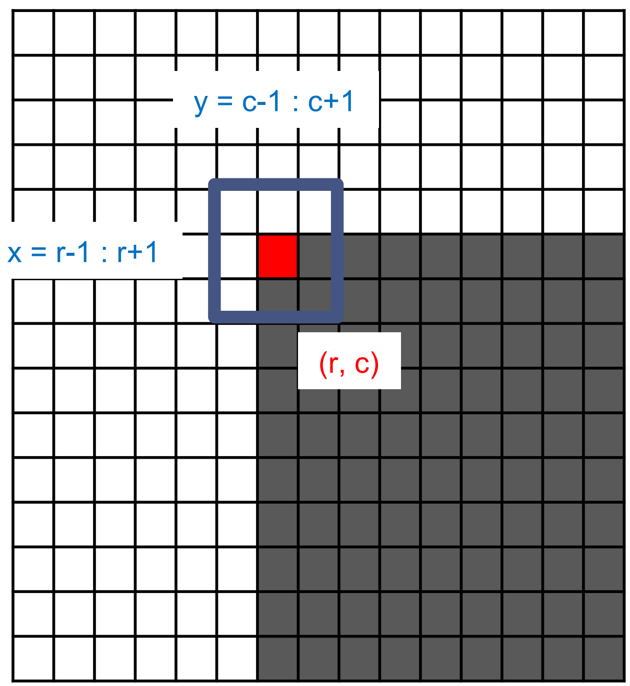


Notation:

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

Adapted from K. Grauman

Harris Detector: Mathematics



Your homework!

Adriana Kovashka

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

$M(0, 0) = ?\ M(0, 0) + I_h(x, y)^2$

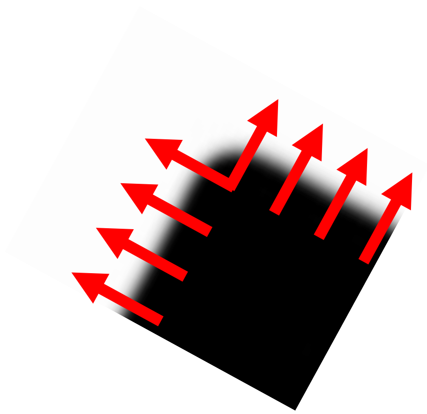
M(0, 1) = ?

M(1, 0) = ?

M(1, 1) = ?

What does the matrix M reveal?

Since M is symmetric, we have $M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$

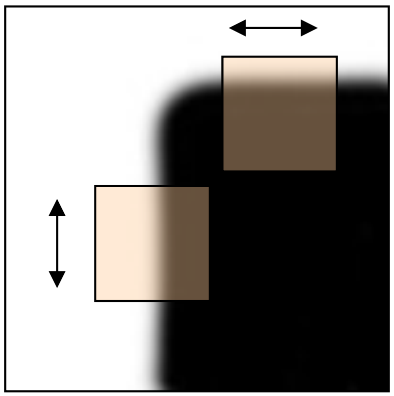


$$Mx_i = \lambda_i x_i$$

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

K. Grauman

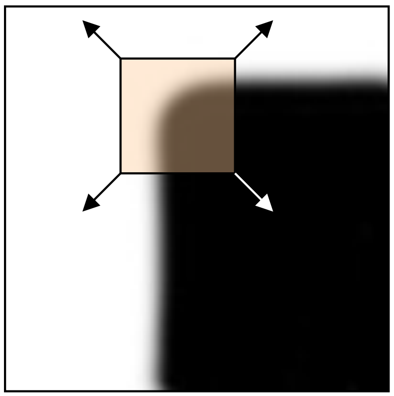
Corner response function



"edge":

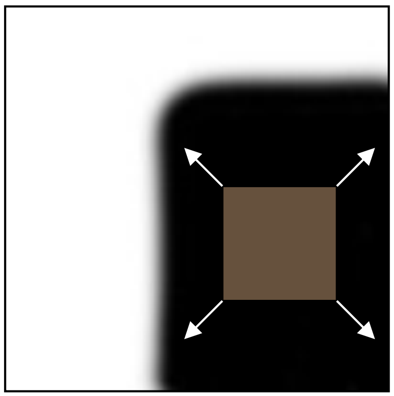
$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$



"corner":

λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$



"flat" region:

λ_1 and λ_2 are small

Adapted from A. Efros, D. Frolova, D. Simakov, K. Grauman

Harris Detector: Mathematics

Measure of corner response:

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

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

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 $$R = \det M - k\left(\operatorname{trace} M\right)^2$$ (k :empirical constant, k = 0.04-0.06)
- Find points with large corner response function R (R > 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

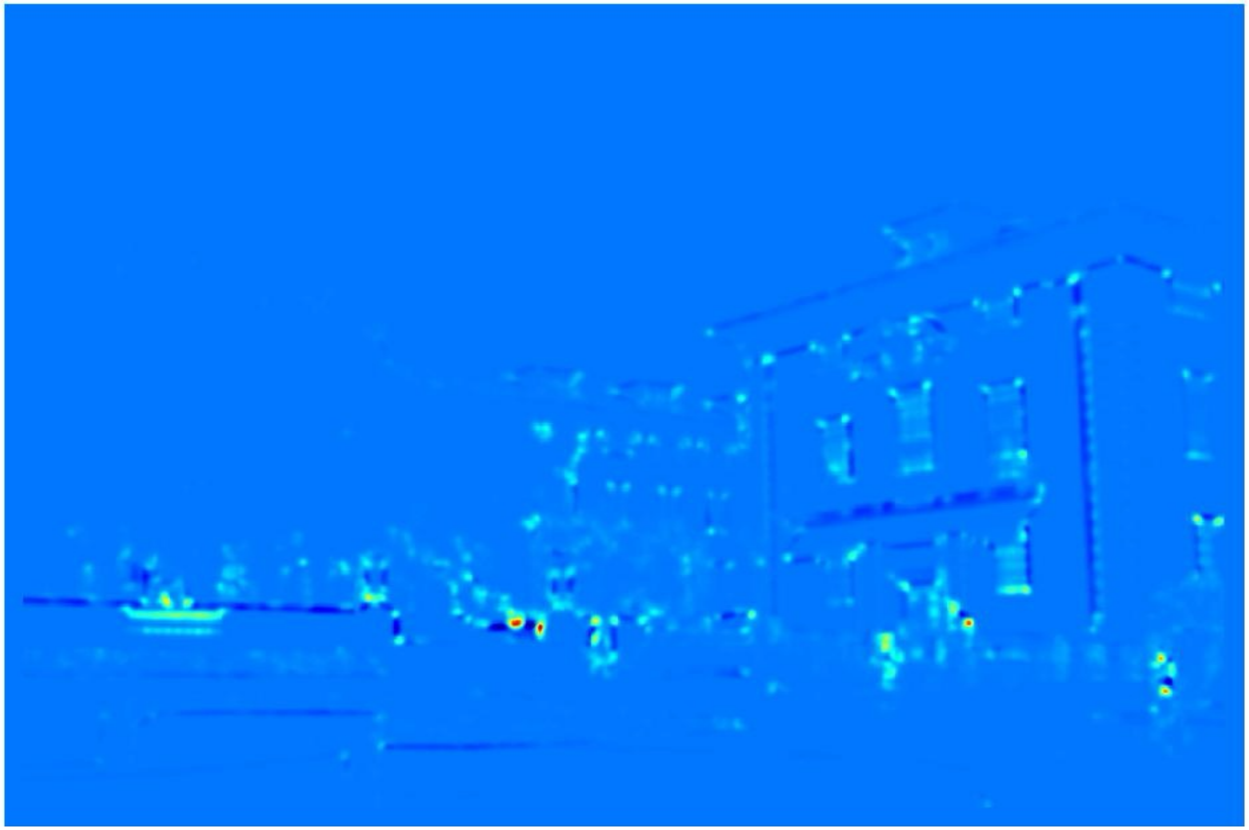
8 neighbors

Example of Harris Application

K. Grauman

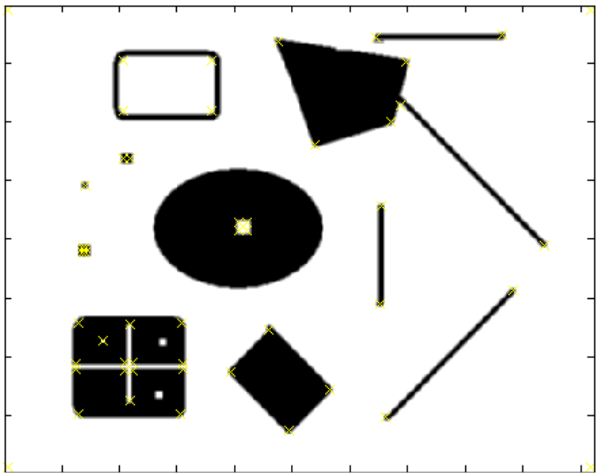
Example of Harris Application

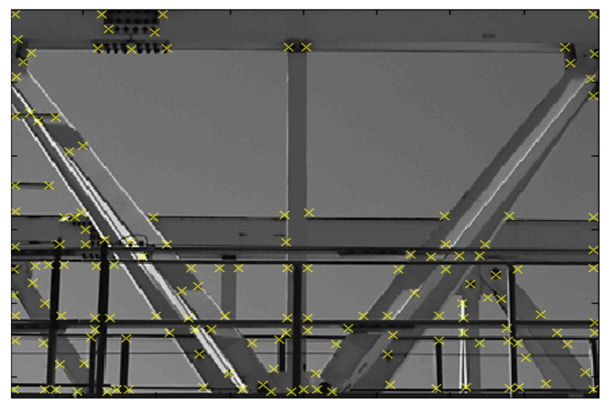
- Corner response at every pixel (red = high, blue = low)



K. Grauman

More Harris Responses





Effect: **A very precise corner detector.**

D. Hoiem

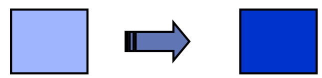
Properties: Invariance vs covariance

- "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.
    - [For 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 $f(ax+b) = a\, f(x) + b$

"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 **Chapter 1, 3.2, 7** http://homes.esat.kuleuven.be/%7Etuytelaa/FT_survey_interestpoints08.pdf

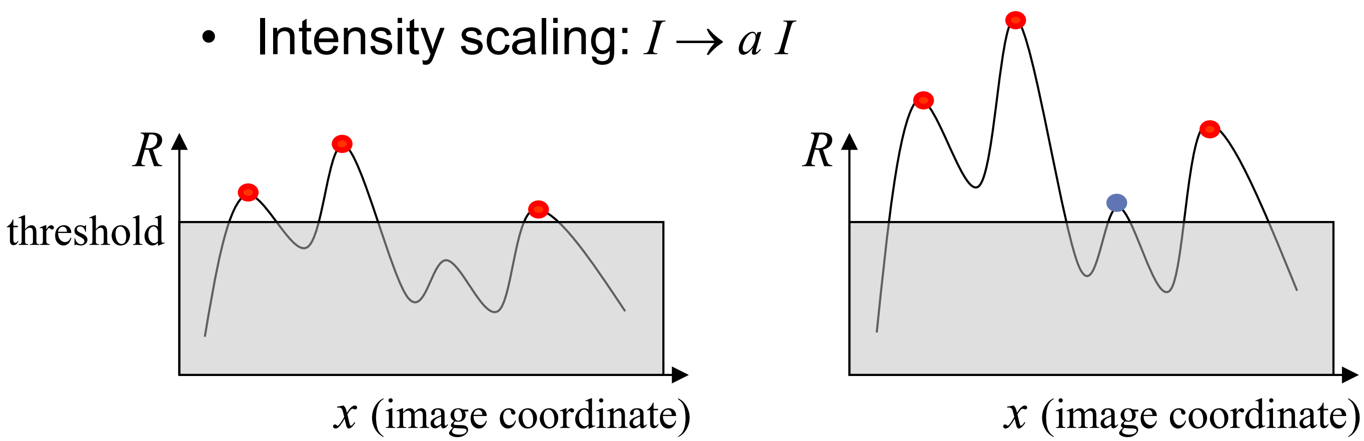
What happens if: Affine intensity change



$I \rightarrow a\,I + b$

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

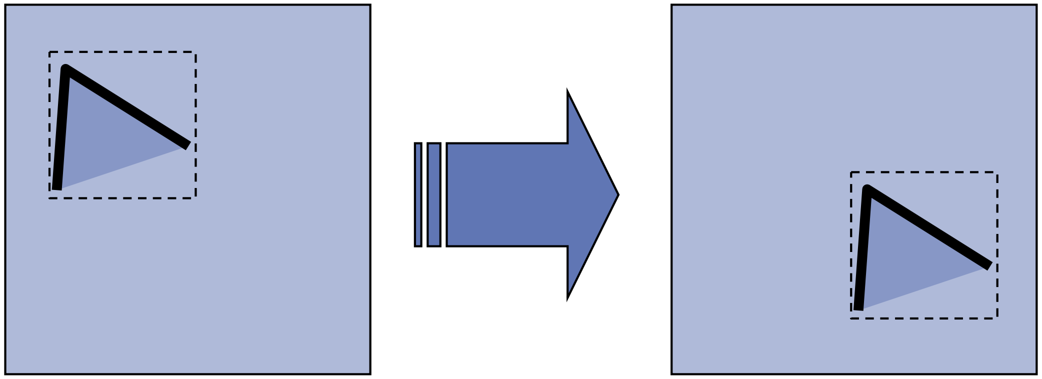
    - Intensity scaling: $I \rightarrow a\,I$



Partially invariant to affine intensity change

L. Lazebnik

What happens if: Image translation

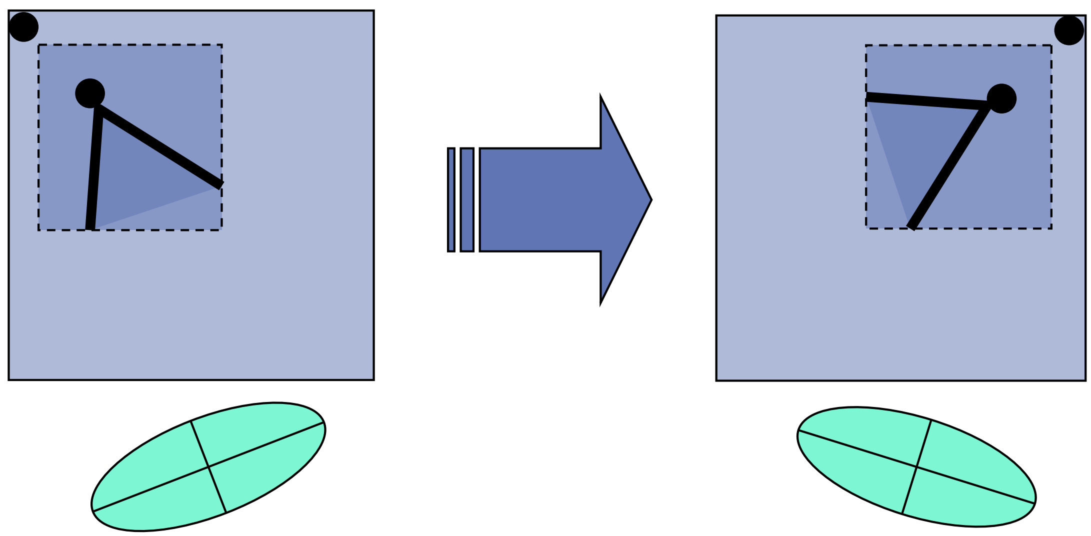


- Derivatives and window function are shift-invariant

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

Adapted from L. Lazebnik

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)

Adapted from L. Lazebnik

What happens if: Scaling

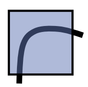
- Invariant to image scale?

image

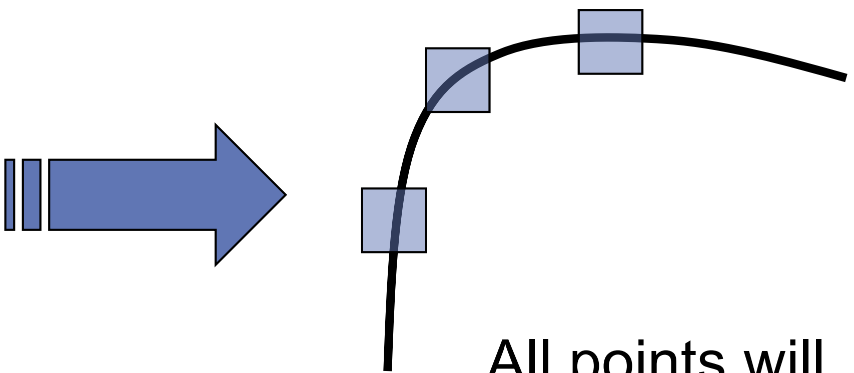
zoomed image

A. Torralba

What happens if: Scaling



Corner



All points will be classified as edges

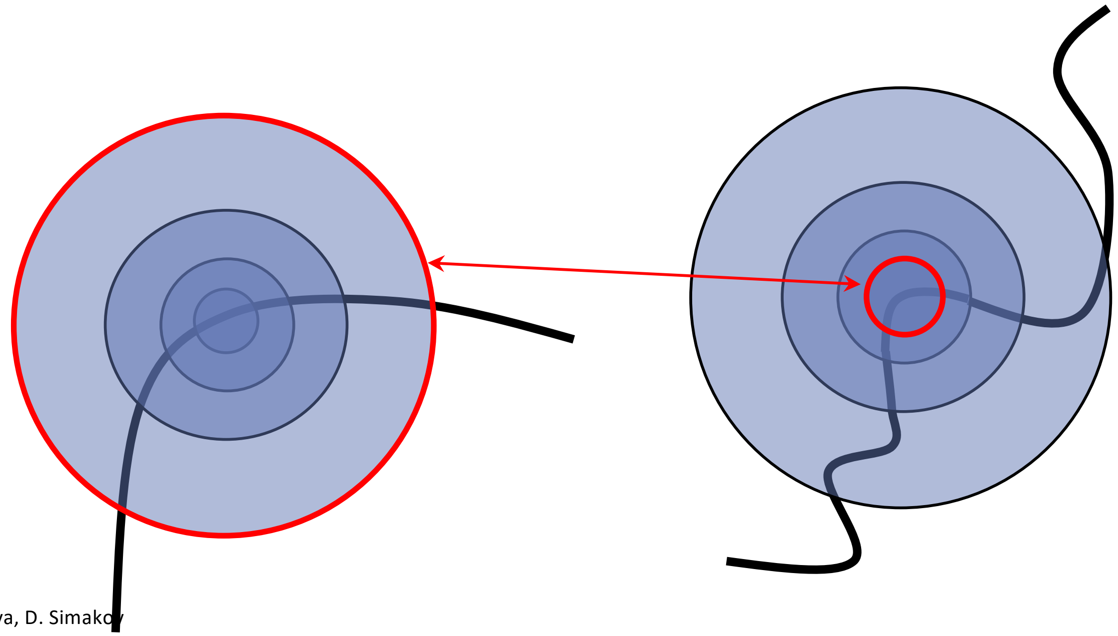
Corner location is **not covariant** to scaling!

(Window size is part of algorithm)

Adapted from L. Lazebnik

Scale invariant detection

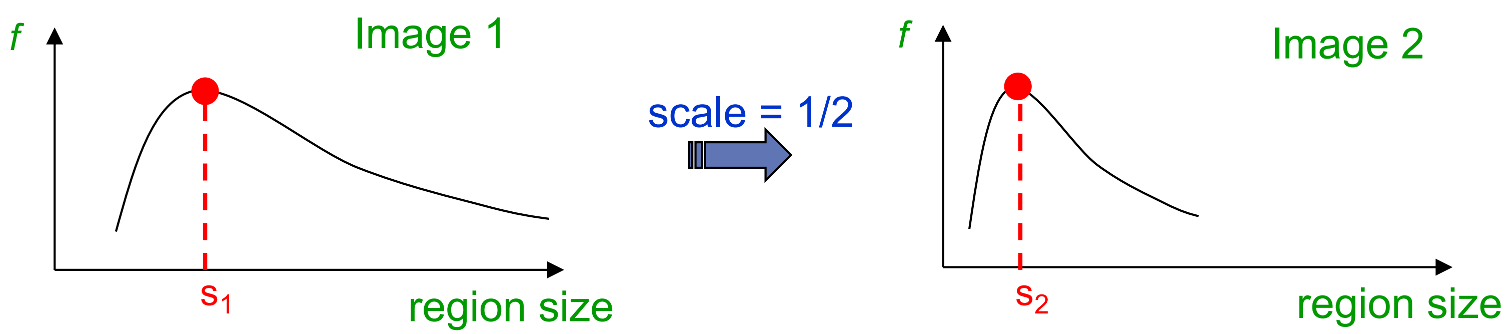
- Problem:
  - How do we choose corresponding windows independently in each image?
  - Do objects have a characteristic scale that we can identify?



Adapted from D. Frolova, D. Simakov

Scale invariant detection

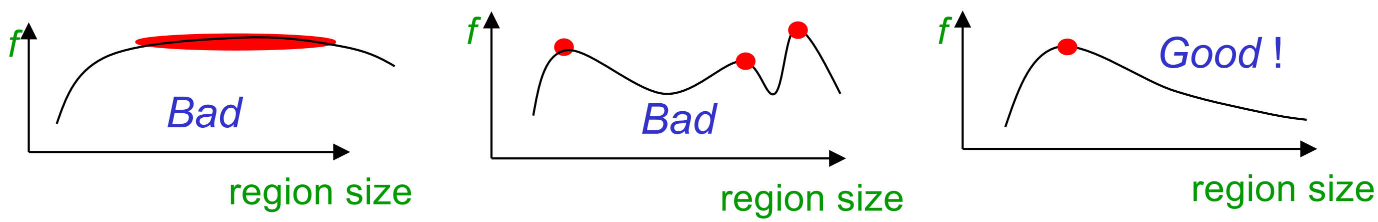
- Solution:
  - Design a function on the region which has the same shape even if the image is resized
  - Take a local maximum of this function



Adapted from A. Torralba

Scale invariant detection

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



Adapted from A. Torralba

Automatic Scale Selection

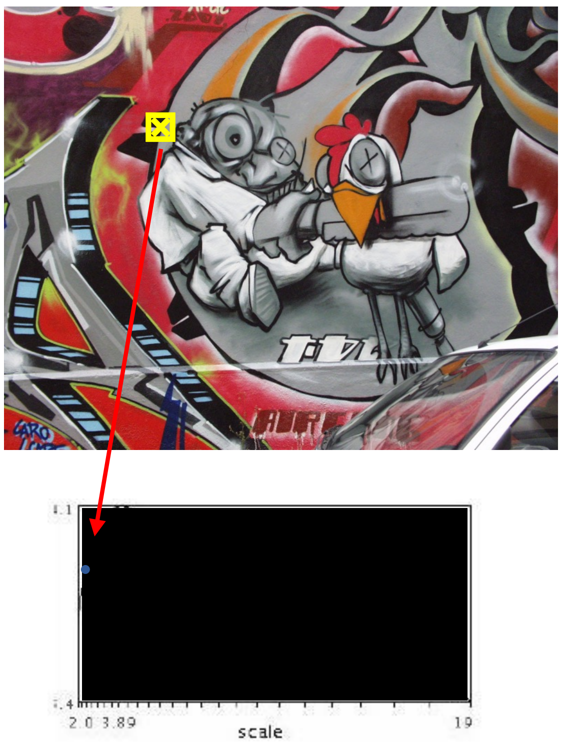
$$f(I_{i_1 \ldots i_m}(x, \sigma)) \;=\; f(I_{i_1 \ldots i_m}(x', \sigma'))$$

How to find corresponding patch sizes?

Adriana Kovashka

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

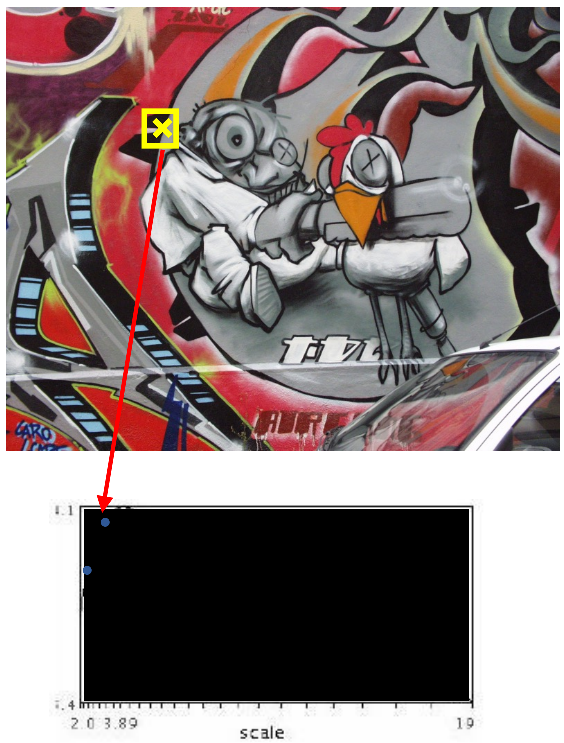


$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

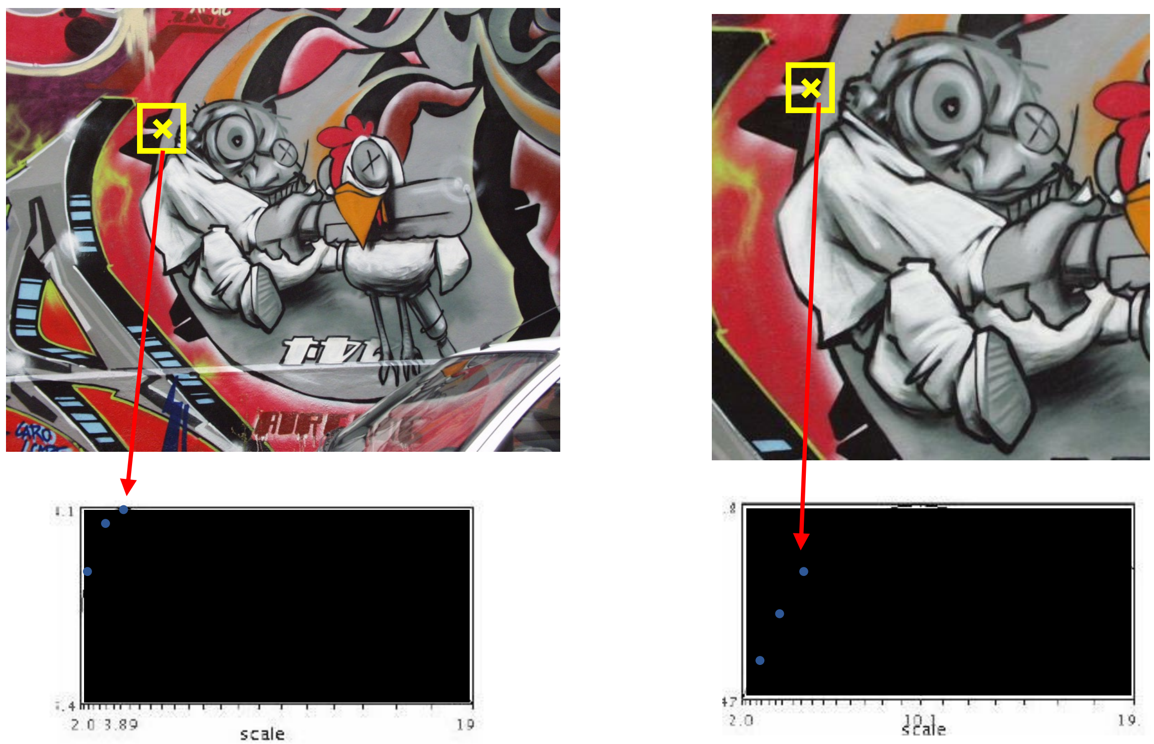


$f(I_{i_1 \ldots i_m}(x,\sigma))$

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

Automatic Scale Selection

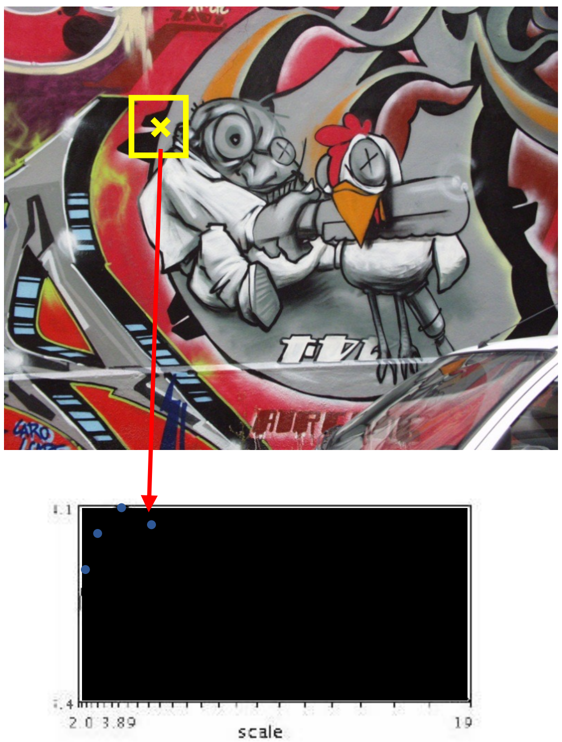
- Function responses for increasing scale (scale signature)



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

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



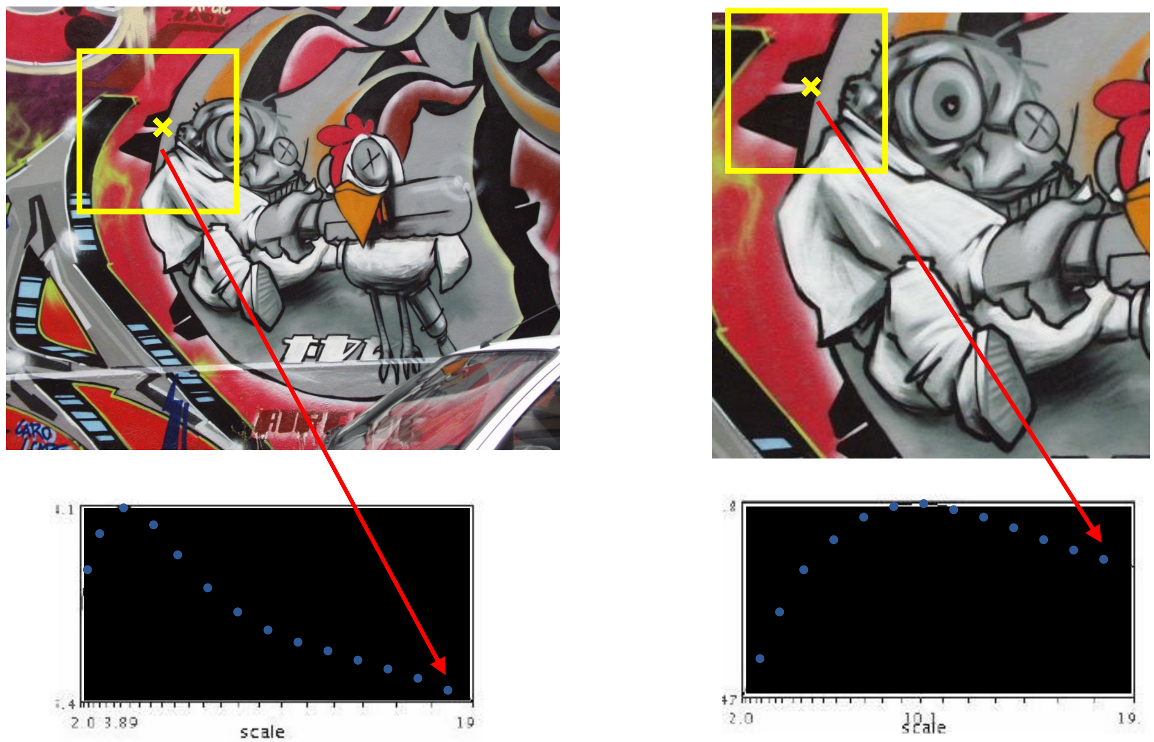
$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

Adriana Kovashka

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

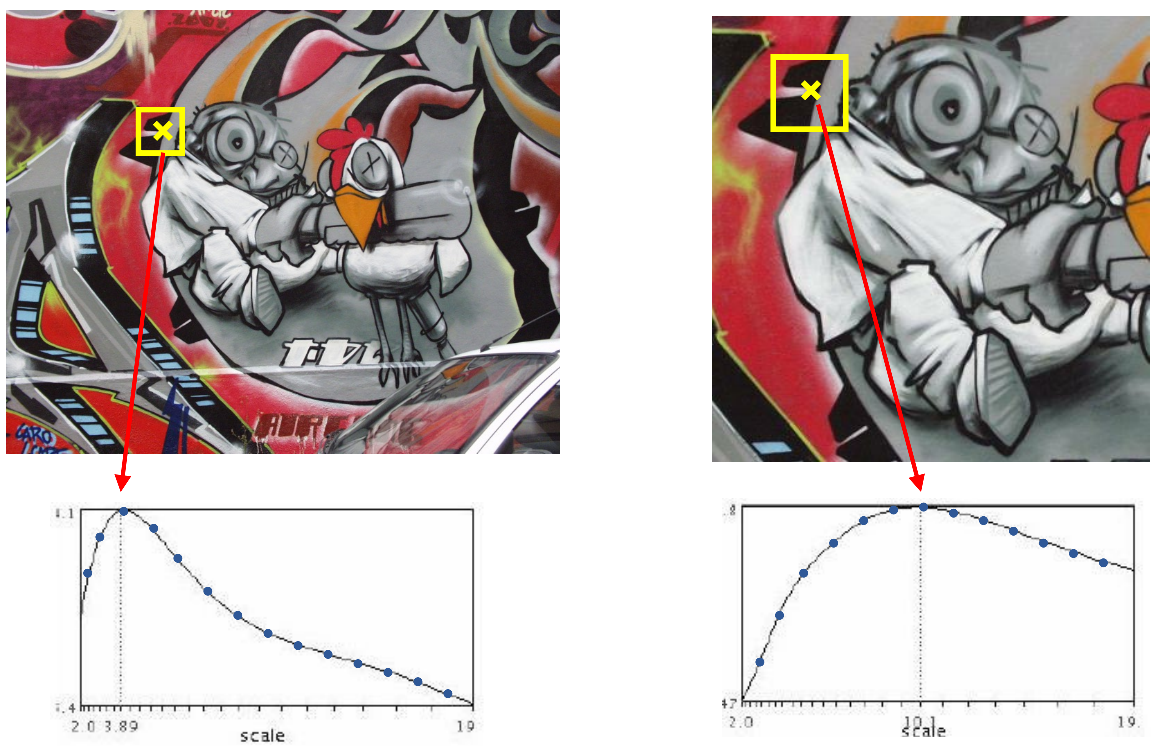


$f(I_{i_1 \ldots i_m}(x, \sigma))$ $\qquad$ $f(I_{i_1 \ldots i_m}(x', \sigma))$

Adriana Kovashka

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

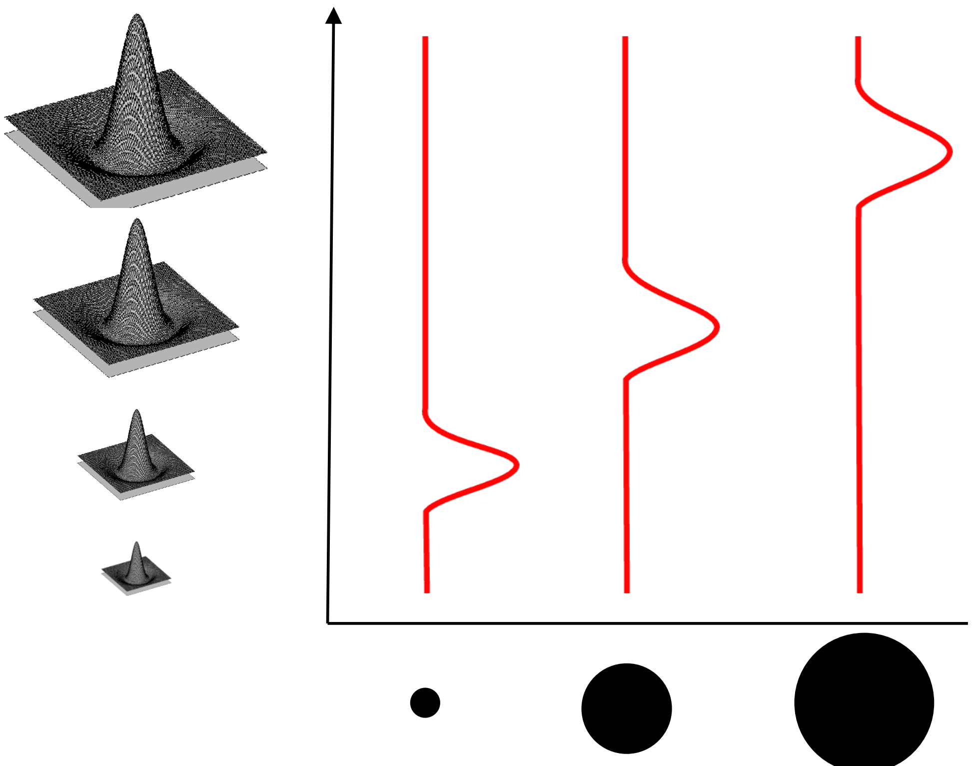


$f(I_{i_1 \ldots i_m}(x, \sigma))$ $\qquad$ $f(I_{i_1 \ldots i_m}(x', \sigma'))$

Adriana Kovashka

What is a useful signature function?

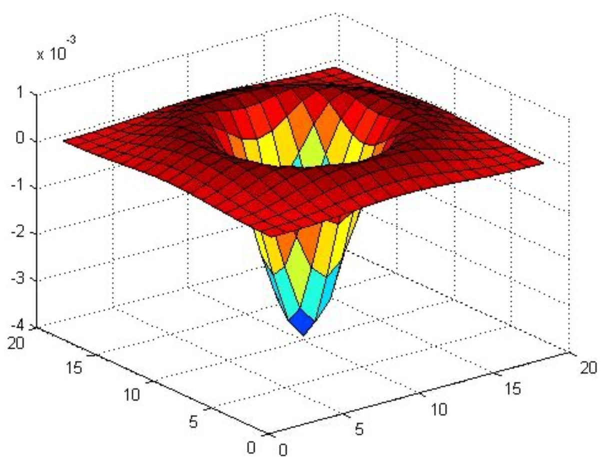
- Laplacian of Gaussian = “blob” detector

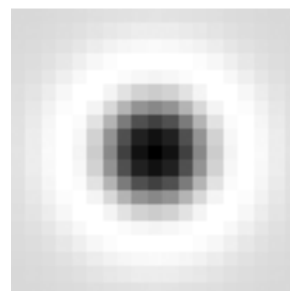


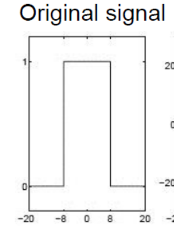
Adriana Kovashka

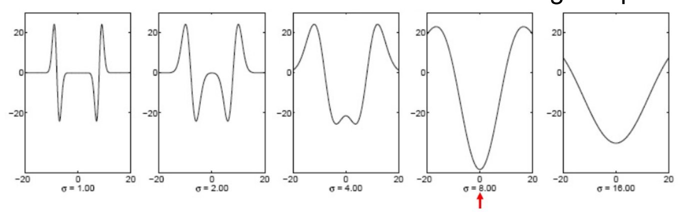
Blob detection in 2D

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









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

Adapted from K. Grauman, L. Lazebnik   Ref: https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm vs http://campar.in.tum.de/Chair/HaukeHeibelGaussianDerivatives

Difference of Gaussian ≈ Laplacian

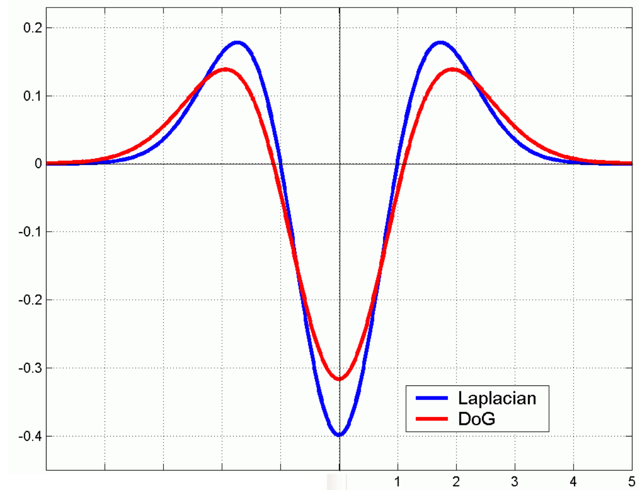
- We can approximate Laplacian with difference of Gaussians; more efficient to implement.

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

(Laplacian)

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

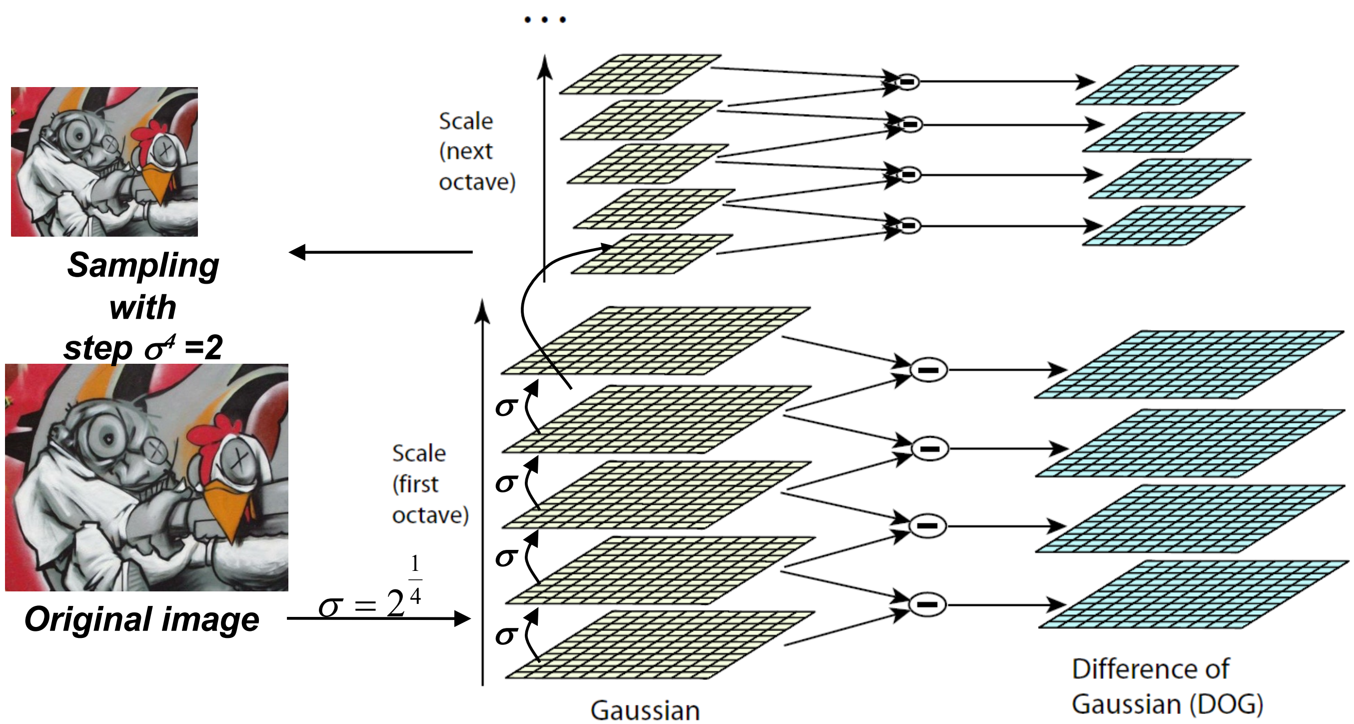
(Difference of Gaussians)



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

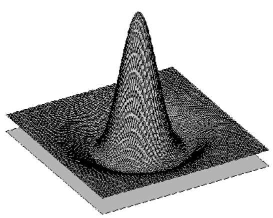
K. Grauman

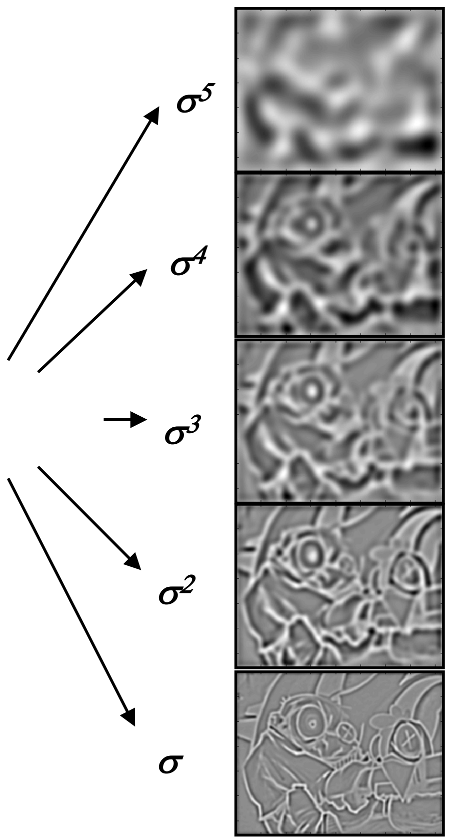
Difference of Gaussian Scale Pyramid



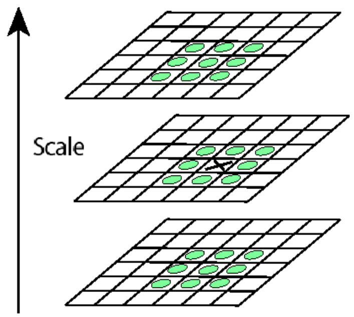
[Github repo] – Module 3

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





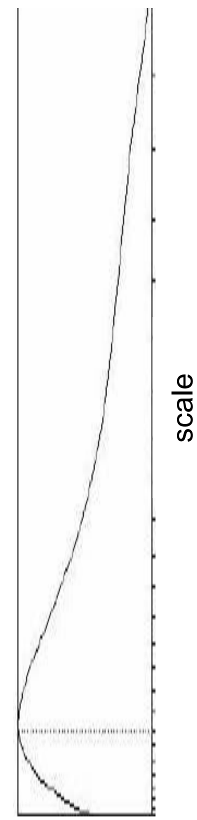
Position-scale space:



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

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

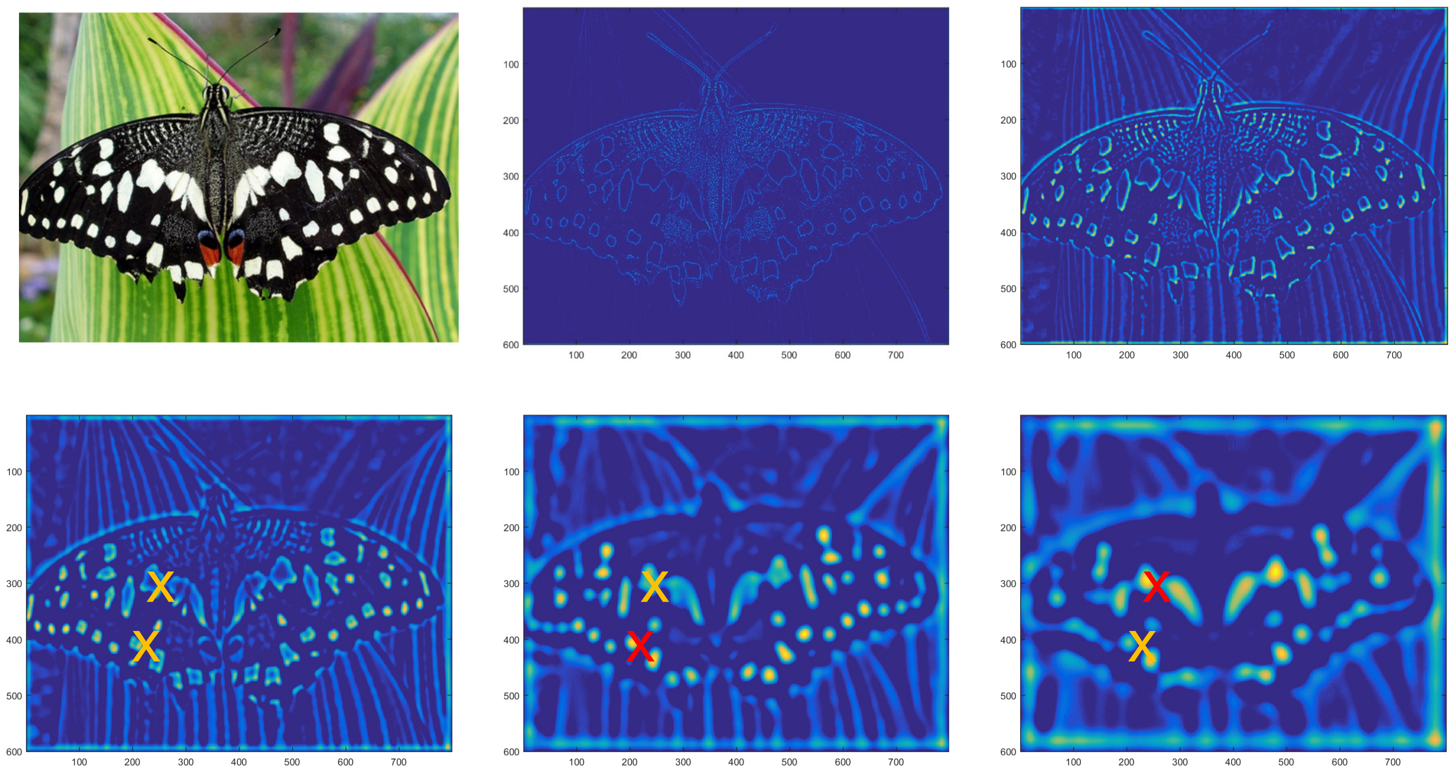
At fixed (x, y)



Adriana Kovashka

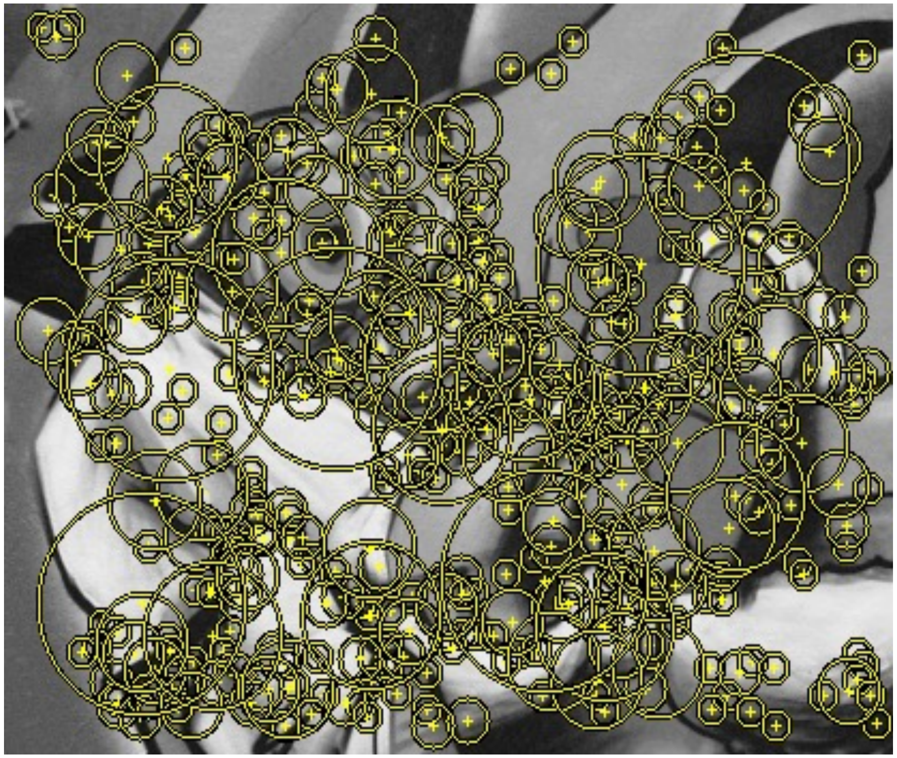
Laplacian pyramid example

- Allows detection of increasingly coarse detail



[Github repo] – Module 3

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

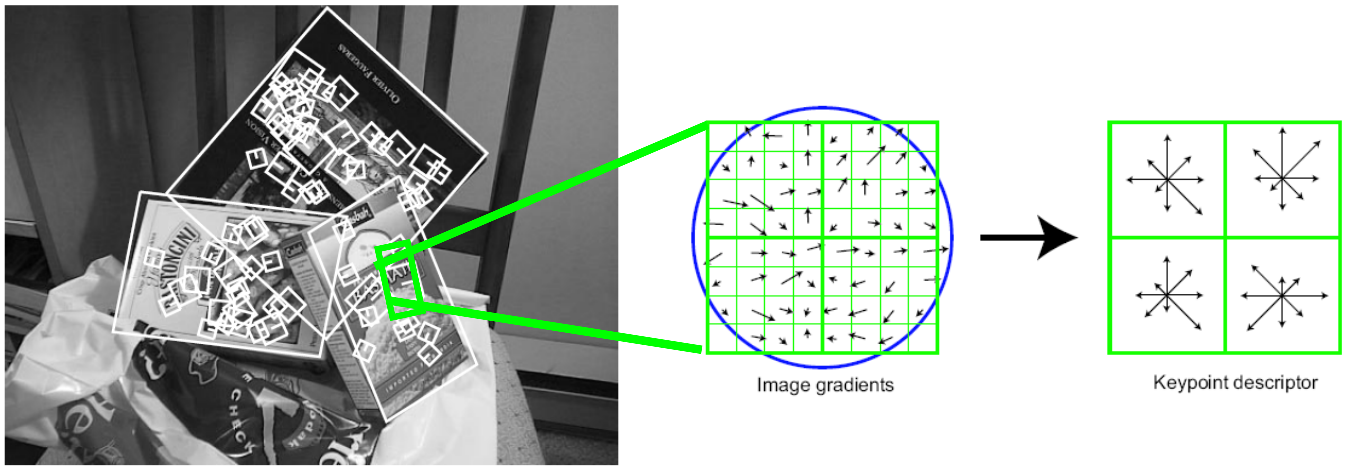
Geometric transformations

e.g. scale, translation , rotation

K. Grauman

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**

[Lowe, ICCV 1999]

K. Grauman, B. Leibe

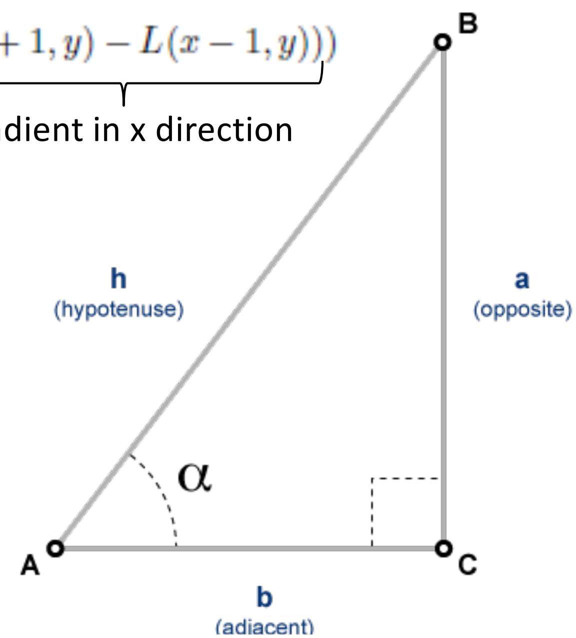
Computing gradients

L = the image intensity

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

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

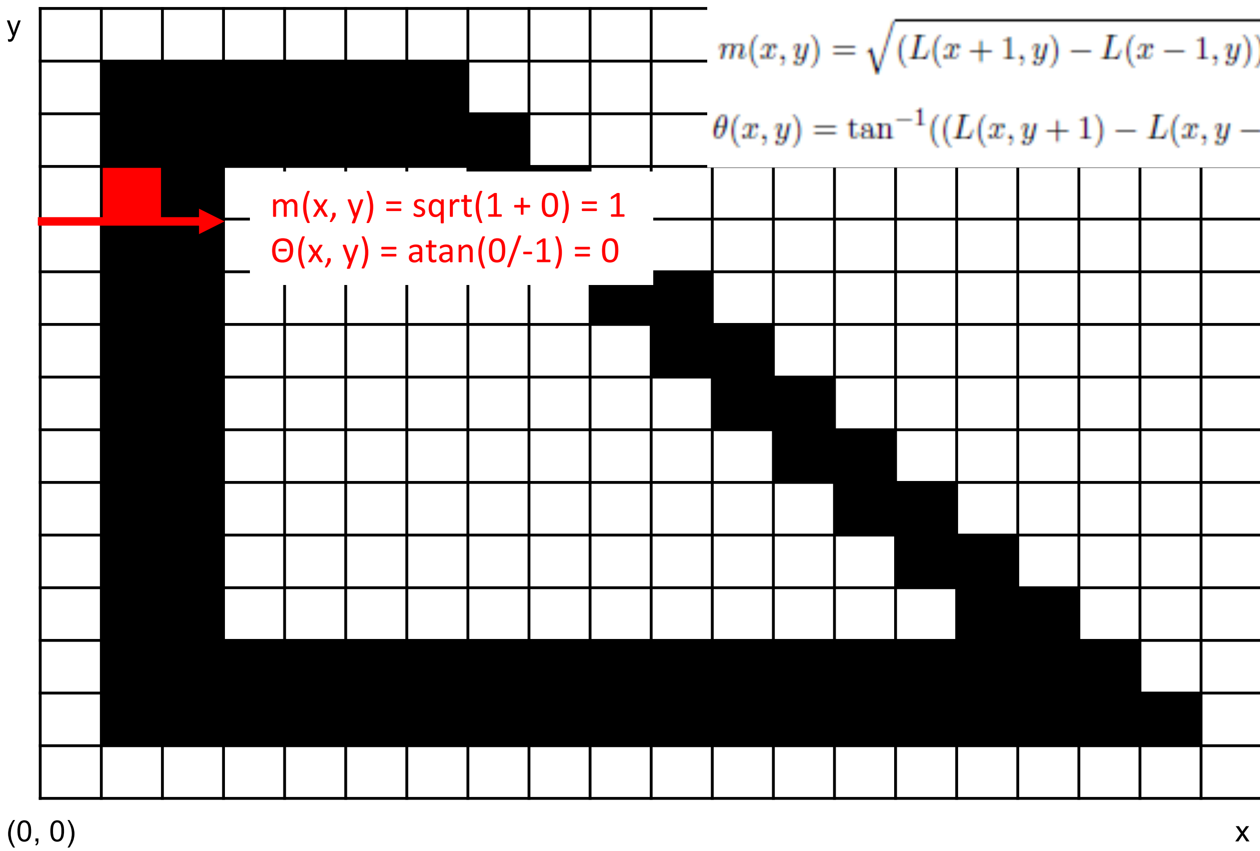
- tan(α)= $\frac{opposite\ side}{adjacent\ side}$



Gradients

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

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



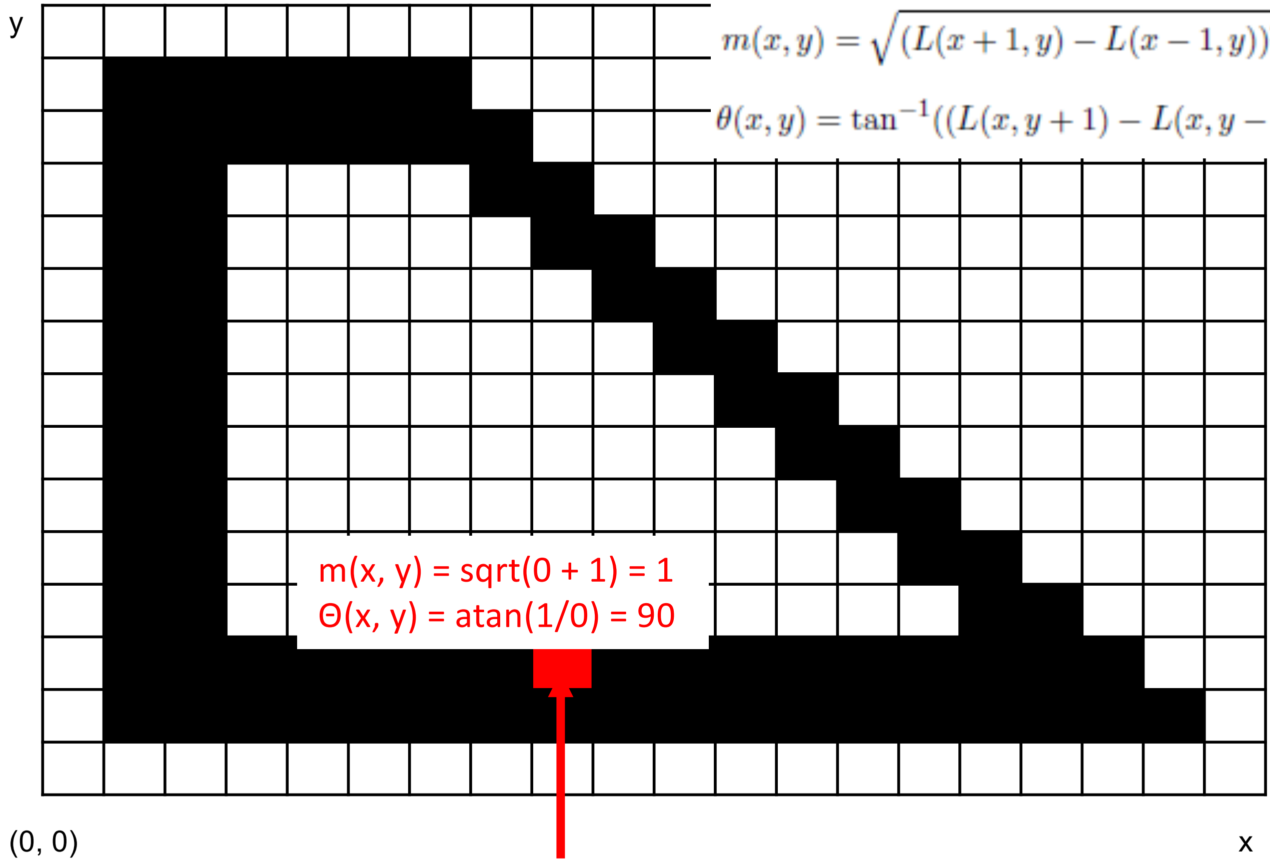
0: Black
1: White

Adriana Kovashka

Gradients

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

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

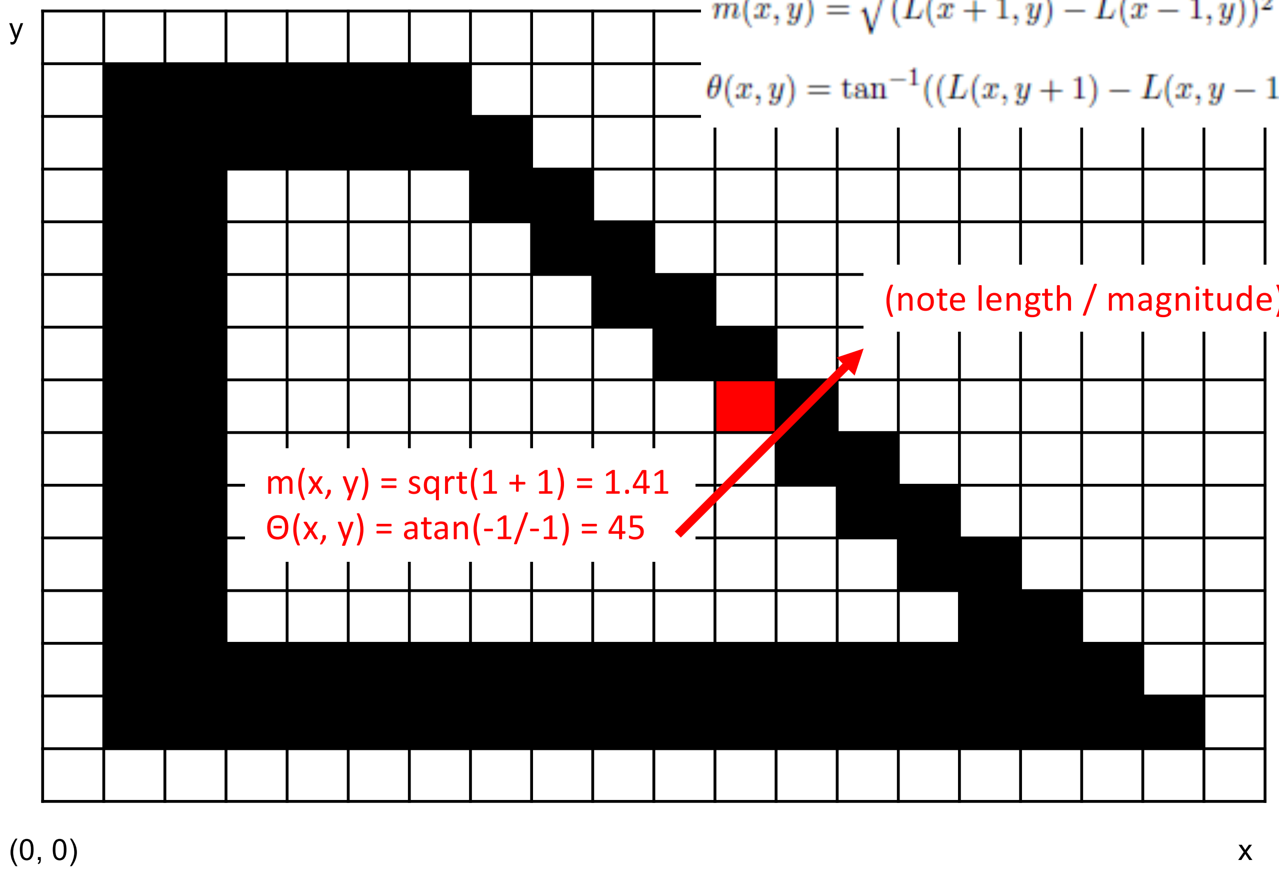


0: Black
1: White

Gradients

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

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

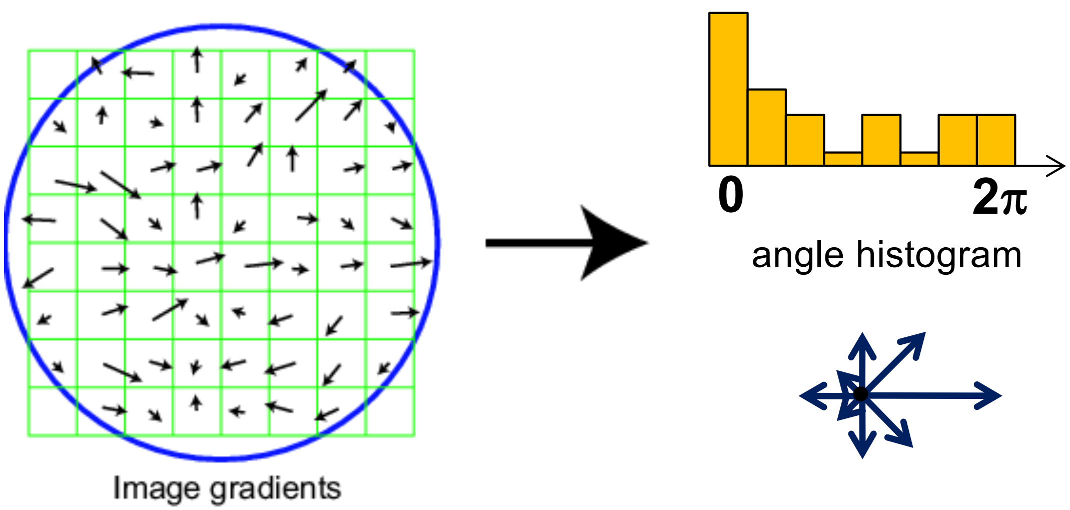


0: Black
1: White

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!

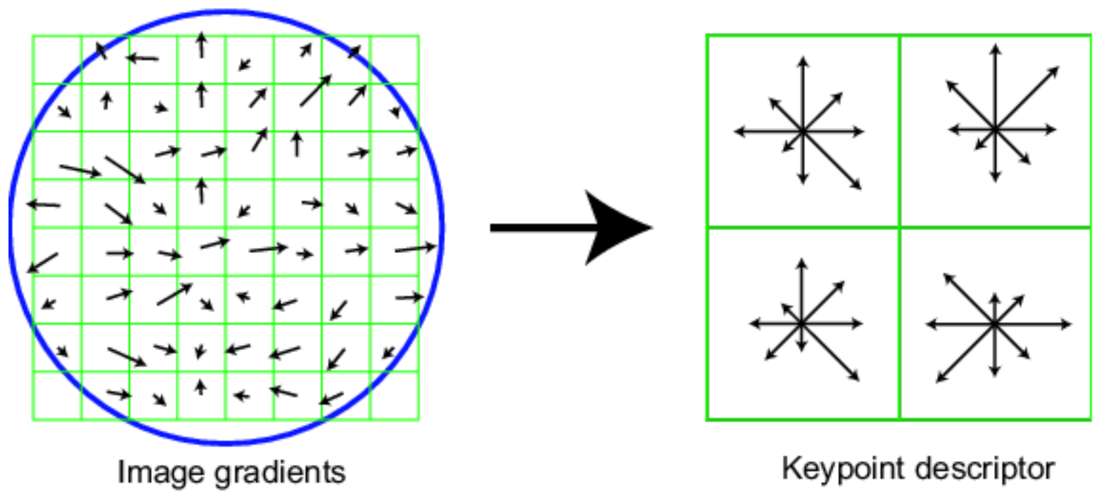


Adapted from L. Zitnick, D. Lowe

Scale Invariant Feature Transform

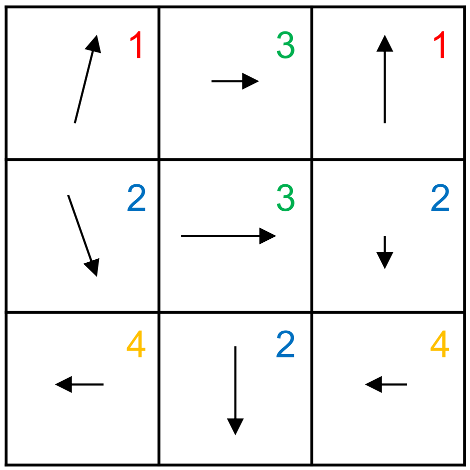
Full version

- 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

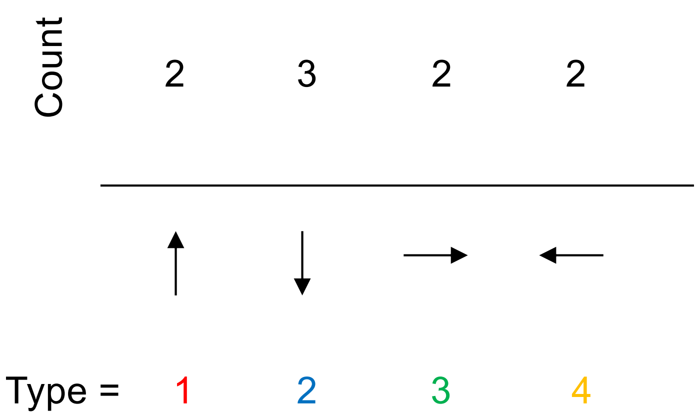


Adapted from L. Zitnick, D. Lowe

Scale Invariant Feature Transform



Uniform weight (ignore magnitude)

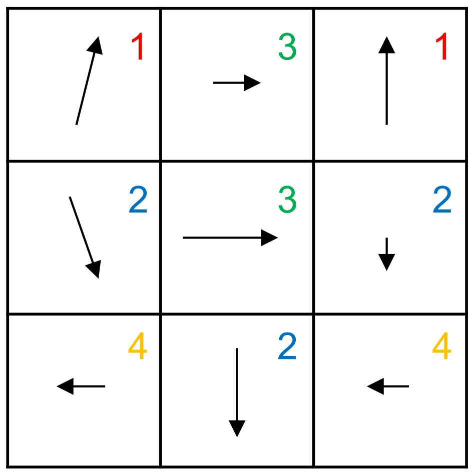


Gradients

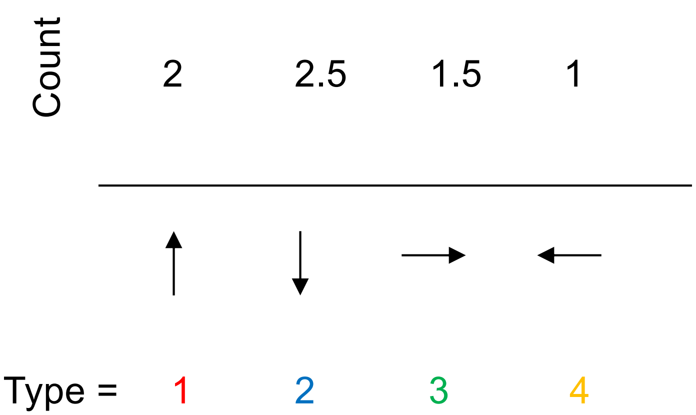
Histogram of gradients

Adriana Kovashka

Scale Invariant Feature Transform



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



Gradients

Histogram of gradients

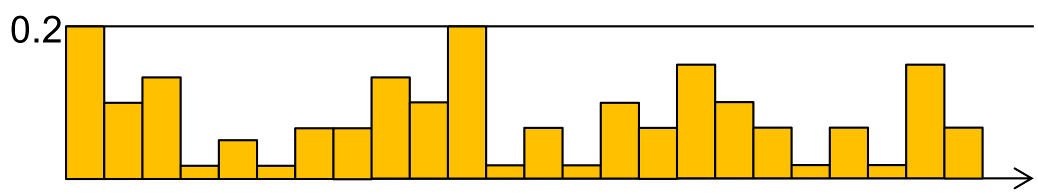
Adriana Kovashka

Scale Invariant Feature Transform

Full version

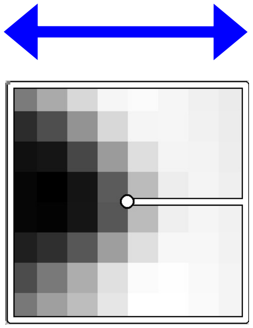
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Quantize the gradient orientations i.e. snap each gradient to one of 8 angles
- Each gradient contributes not just 1, but magnitude(gradient) to the histogram, i.e. stronger gradients contribute more
- 16 cells * 8 orientations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

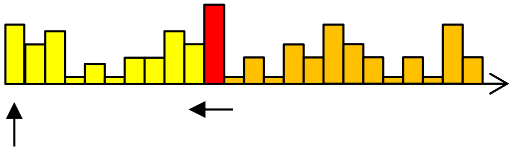
$$\boxed{\sum_i d_i = 1 \quad \text{such that: } d_i < 0.2}$$

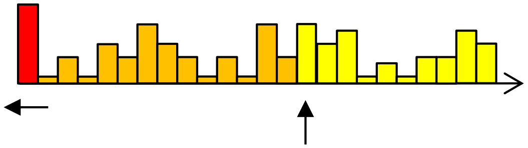


Adapted from L. Zitnick, D. Lowe

Making descriptor rotation invariant







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

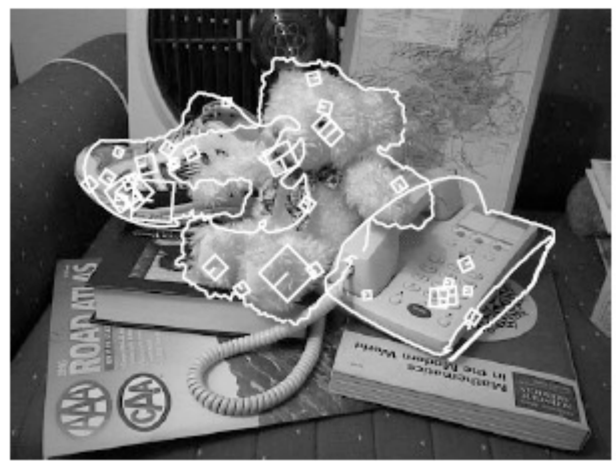
Adapted from K. Grauman, image from Matthew Brown

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
- 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

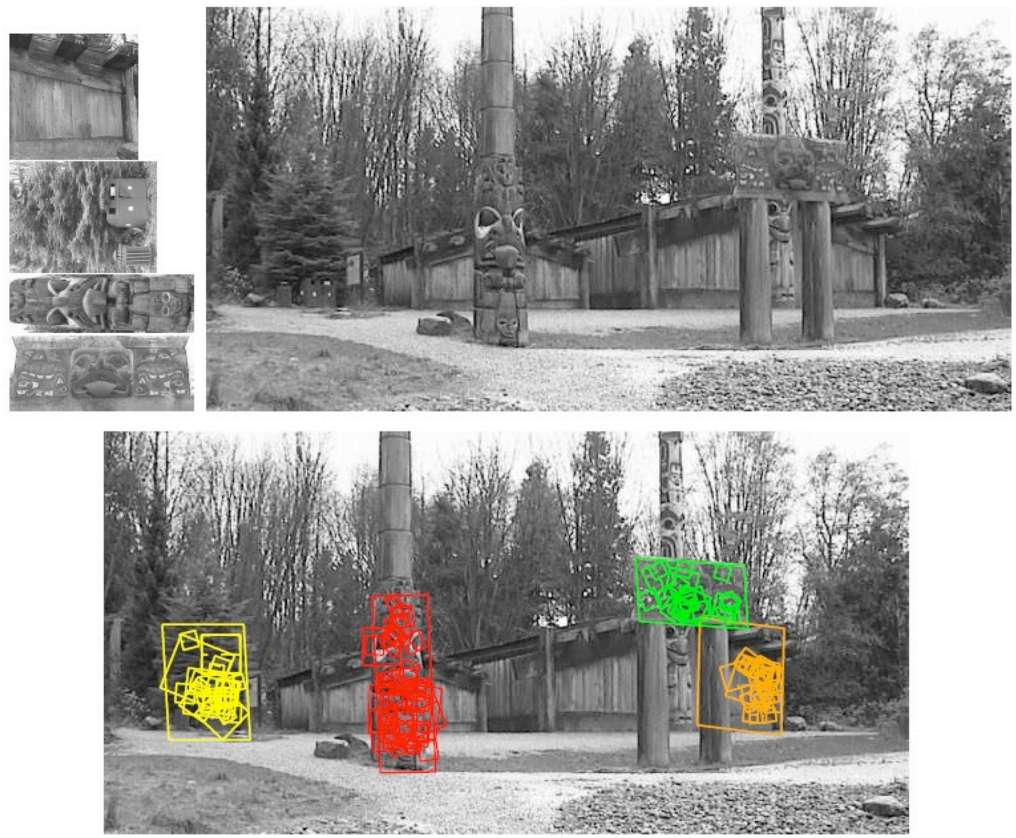
Adapted from S. Seitz

Examples of using SIFT



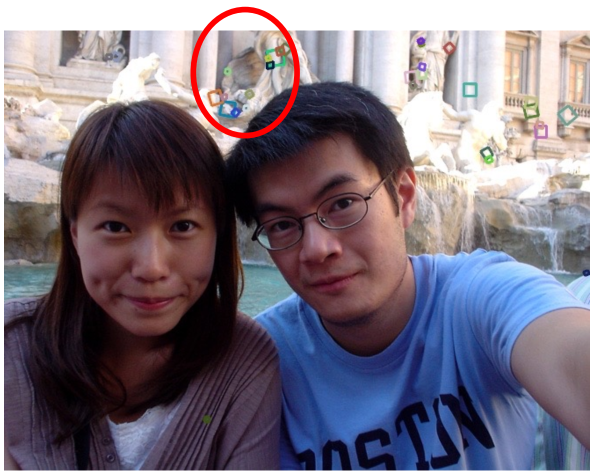
Adriana Kovashka

Examples of using SIFT



Adriana Kovashka

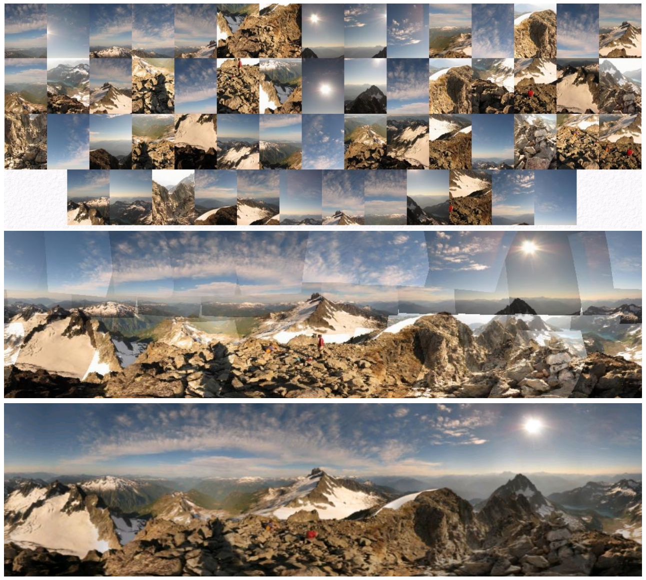
Examples of using SIFT



Images from S. Seitz

Applications of local invariant features

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



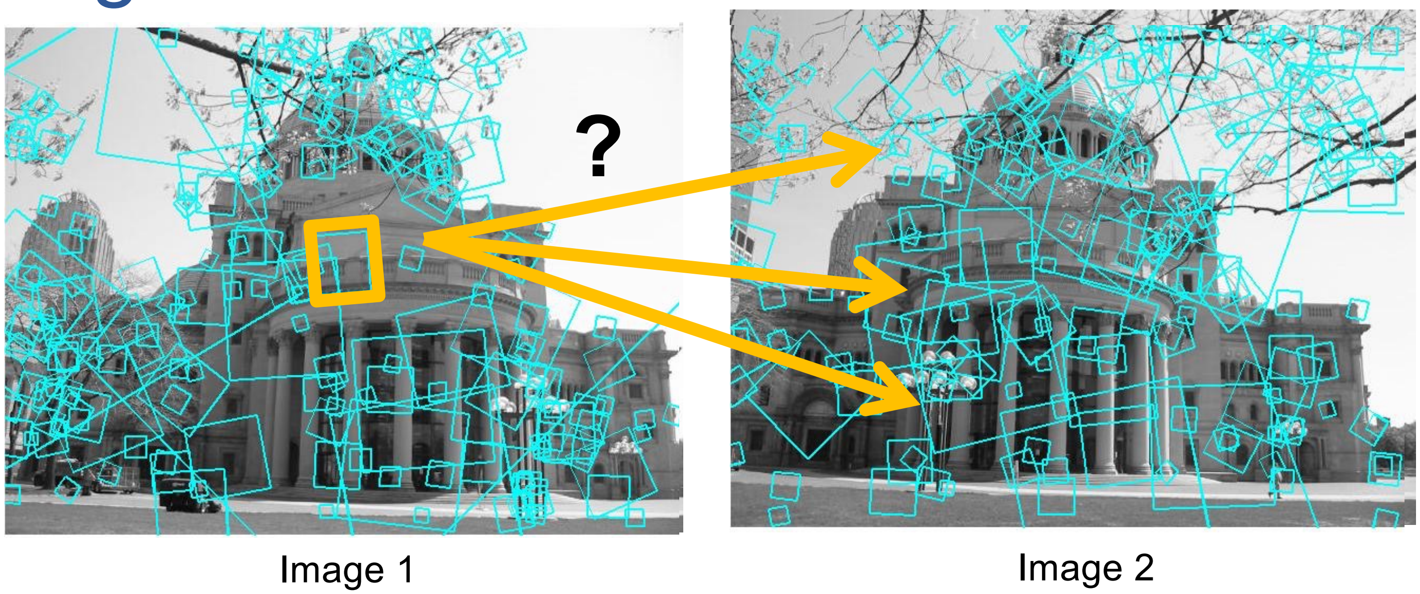
http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Adapted from K. Grauman and L. Lazebnik

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

K. Grauman

Robust Matching

Image 1

Image 2

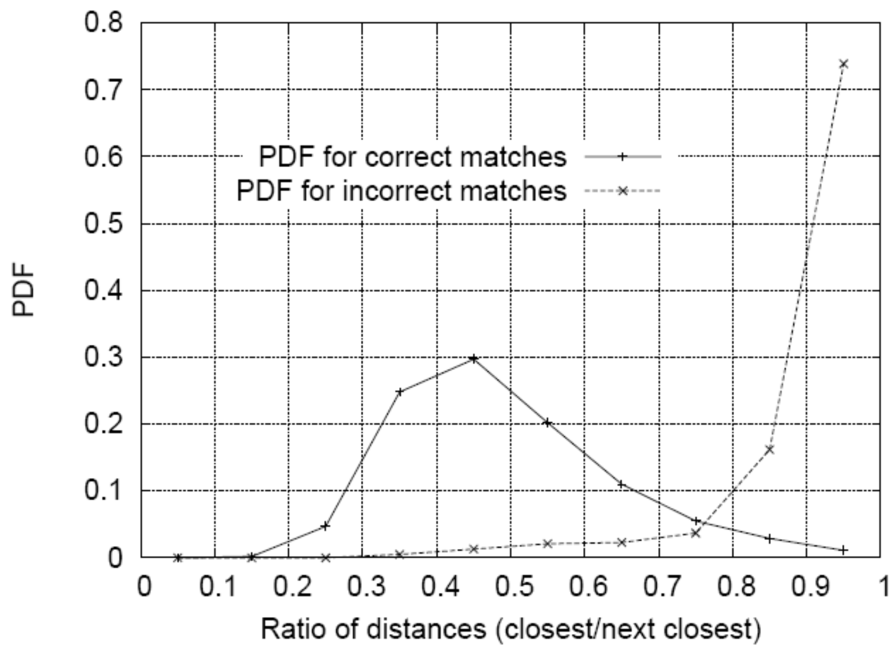
- At what Euclidean distance value do we have a good match?
- To add robustness to matching, can consider **ratio**: distance of query to best match / distance to second best match
  - If low, first match looks good
  - If high, could be ambiguous match

d(q, fv1) / d(q, fv2)

K. Grauman

Matching SIFT descriptors

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



PDF: Probability Density Function

Lowe IJCV 2004

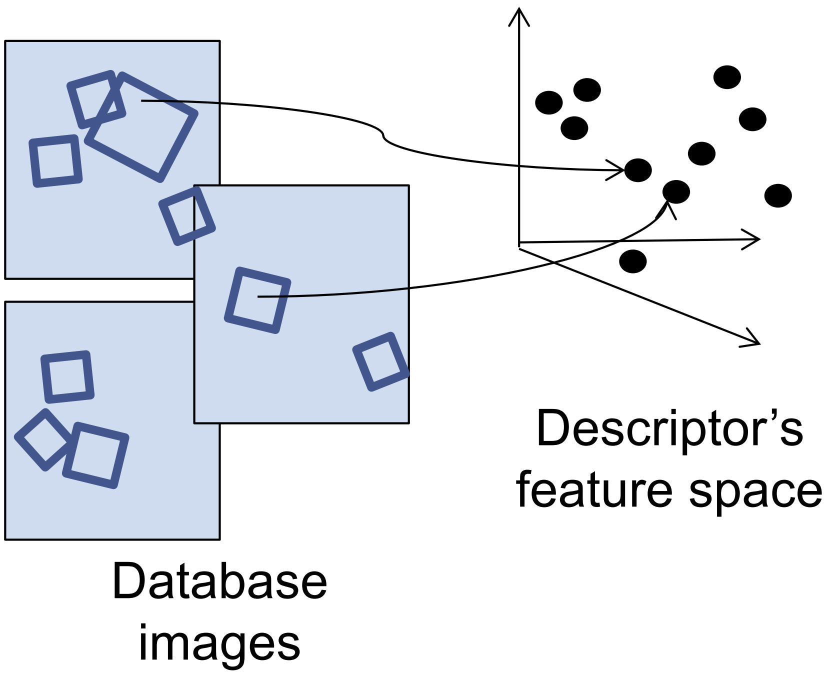
Efficient Matching

- 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?

Adapted from K. Grauman

Matching Local Features Setup

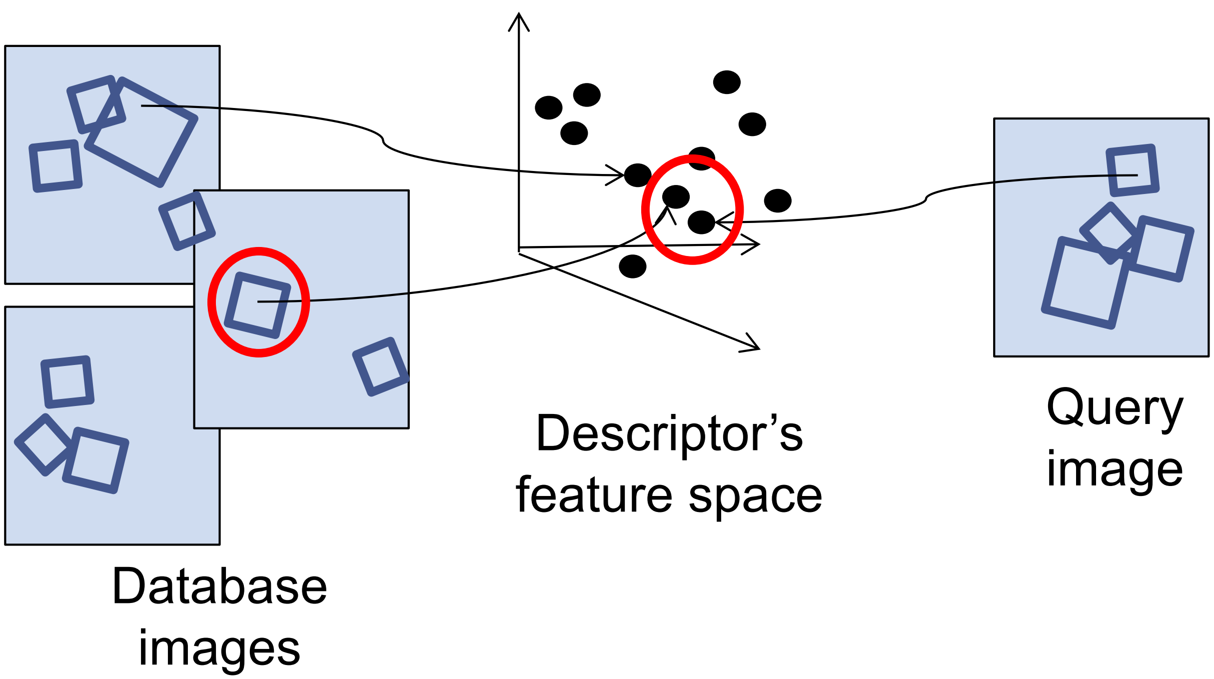
- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)



K. Grauman

Matching Local Features Setup

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



K. Grauman

Indexing local features


Index

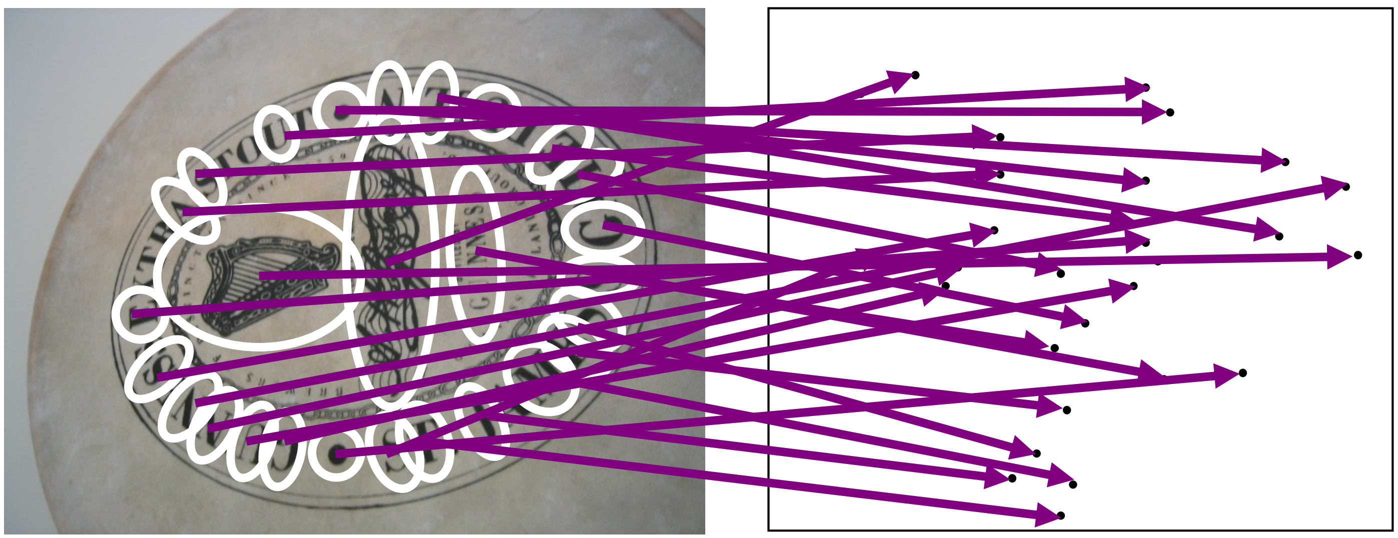
"Along I-75," From Detroit to
Florida; ***inside back cover***
"Drive I-95," From Boston to
Florida; ***inside back cover***
1929 Spanish Trail Roadway; 101-102,104
511 Traffic Information; 83
A1A (Barrier Isl) - I-95 Access; 86
AAA (and CAA); 83
AAA National Office; 88
Abbreviations,
Colored 25 mile Maps; cover
Exit Services; 196
Travelogue; 85
Africa; 177
Agricultural Inspection Stns; 126
Ah-Tah-Thi-Ki Museum; 160
Air Conditioning, First; 112
Alabama; 124
Alachua; 132
County; 131
Alafia River; 143
Alapaha, Name; 126
Alfred B Maclay Gardens; 106
Alligator Alley; 154-155
Alligator Farm, St Augustine; 169
Alligator Hole (definition); 157
Alligator, Buddy; 155
Alligators; 100,135,138,147,156
Anastasia Island; 170
Anhaica; 108-109,146
Apalachicola River; 112
Appleton Mus of Art; 136
Aquifer; 102
Arabian Nights; 94
Art Museum, Ringling; 147
Aruba Beach Cafe; 183
Aucilla River Project; 106
Babcock-Web WMA; 151
Bahia Mar Marina; 184
Baker County; 99
Barefoot Mailmen; 182
Barge Canal; 137
Bee Line Expy; 80
Belz Outlet Mall; 89
Bernard Castro; 136
Big "I"; 165
Big Cypress; 155,158
Big Foot Monster; 105
Billie Swamp Safari; 160
Blackwater River SP; 117
Blue Angels
A4-C Skyhawk; 117
Atrium; 121
Blue Springs SP; 87
Butterfly Center, McGuire; 134
CAA (see AAA)
CCC, The; 111,113,115,135,142
Ca d'Zan; 147
Caloosahatchee River; 152
Name; 150
Canaveral Natnl Seashore; 173
Cannon Creek Airpark; 130
Canopy Road; 106,169
Cape Canaveral; 174
Castillo San Marcos; 169
Cave Diving; 131
Cayo Costa, Name; 150
Celebration; 93
Charlotte County; 149
Charlotte Harbor; 150
Chautauqua; 116
Chipley; 114
Name; 115
Choctawatchee, Name; 115
Circus Museum, Ringling; 147
Citrus; 88,97,130,136,140,180
CityPlace, W Palm Beach; 180
City Maps,
Ft Lauderdale Expwys; 194-195
Jacksonville; 163
Kissimmee Expwys; 192-193
Miami Expressways; 194-195
Orlando Expressways; 192-193
Pensacola; 26
Tallahassee; 191
Tampa-St. Petersburg; 63
St. Augsutine; 191
Civil War; 100,108,127,138,141
Clearwater Marine Aquarium; 187
Collier County; 154
Collier, Barron; 152
Colonial Spanish Quarters; 168
Columbia County; 101,128
Coquina Building Material; 165
Corkscrew Swamp, Name; 154
Cowboys; 95
Crab Trap II; 144
Cracker, Florida; 88,95,132
Crosstown Expy; 11,35,98,143
Cuban Bread; 184
Dade Battlefield; 140
Dade, Maj. Francis; 139-140,161
Dania Beach Hurricane; 184
Daniel Boone, Florida Walk; 117
Daytona Beach; 172-173
De Land; 87
De Soto, Hernando,
Anhaica; 108-109,146
County; 149
Driving Lanes; 85
Duval County; 163
Eau Gallie; 175
Edison, Thomas; 152
Eglin AFB; 116-118
Eight Reale; 176
Ellenton; 144-145
Emanuel Point Wreck; 120
Emergency Callboxes; 83
Epiphytes; 142,148,157,159
Escambia Bay; 119
Bridge (I-10); 119
County; 120
Estero; 153
Everglade,90,95,139-140,154-160
Draining of; 156,181
Wildlife MA; 160
Wonder Gardens; 154
Falling Waters SP; 115
Fantasy of Flight; 95
Fayer Dykes SP; 171
Fires, Forest; 166
Fires, Prescribed ; 148
Fisherman's Village; 151
Flagler County; 171
Flagler, Henry; 97,165,167,171
Florida Aquarium; 186
Florida,
12,000 years ago; 187
Cavern SP; 114
Map of all Expressways; 2-3
Mus of Natural History; 134
National Cemetery ; 141
Part of Africa; 177
Platform; 187
Sheriff's Boys Camp; 126
Sports Hall of Fame; 130
Sun 'n Fun Museum; 97
Supreme Court; 107
Florida's Turnpike (FTP), 178,189
25 mile Strip Maps; 66
Administration; 189
Coin System; 190
Exit Services; 189
HEFT; 76,161,190
History; 189
Names; 189
Service Plazas; 190
Spur SR91; 76
Ticket System; 190
Toll Plazas; 190
Ford, Henry; 152
Fort Barrancas; 122
Buried Alive; 123
Fort Caroline; 164

- 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

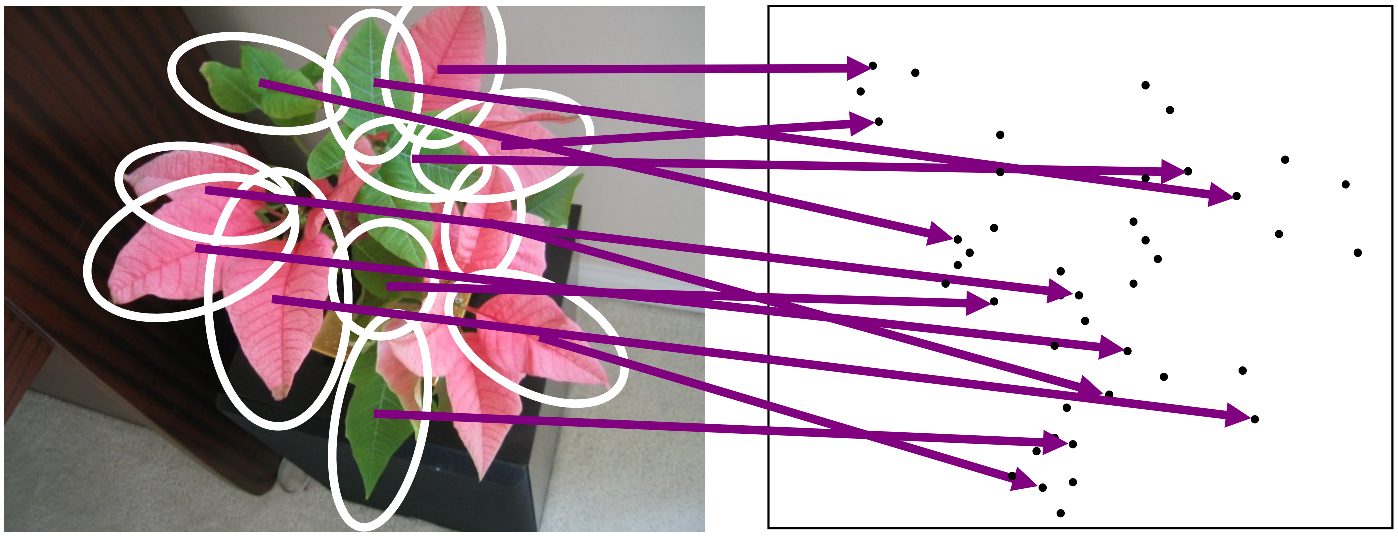
- Extract some local features from a number of images …



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

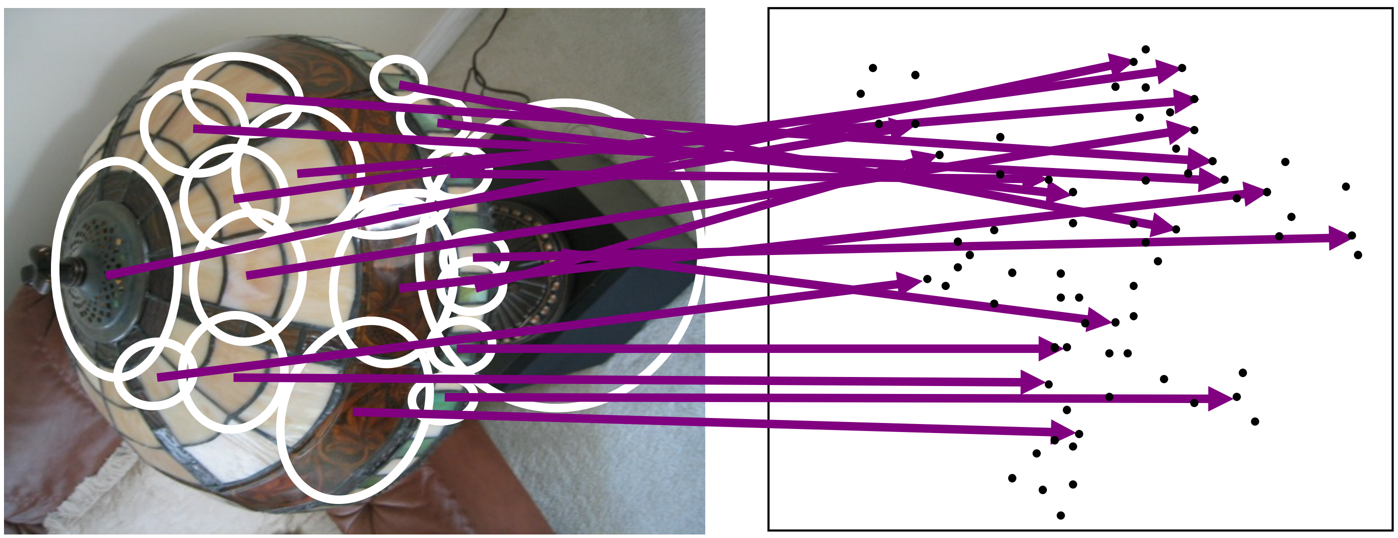
D. Nister, CVPR 2006

Visual Words: main idea



D. Nister, CVPR 2006

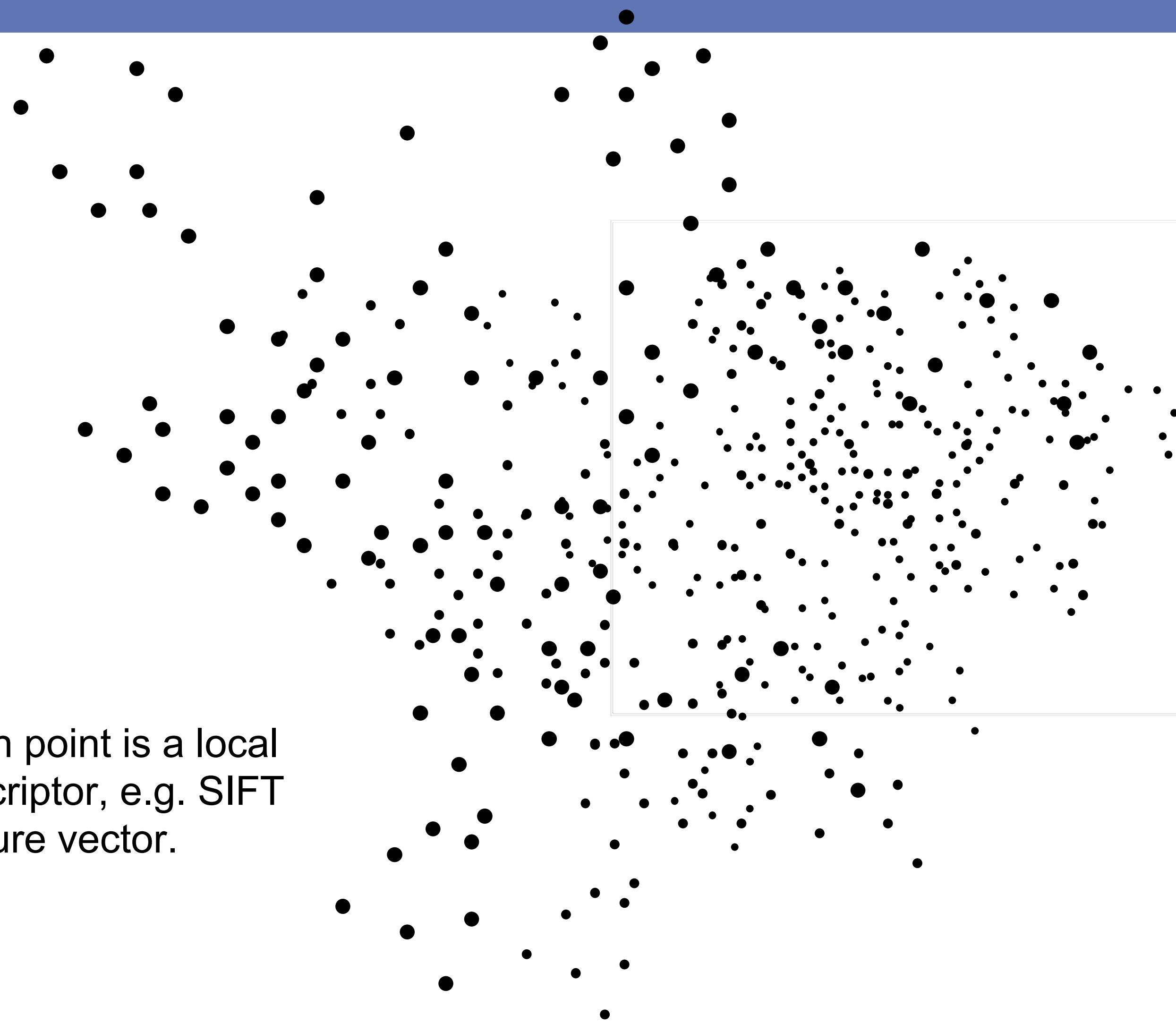
Visual Words: main idea



D. Nister, CVPR 2006

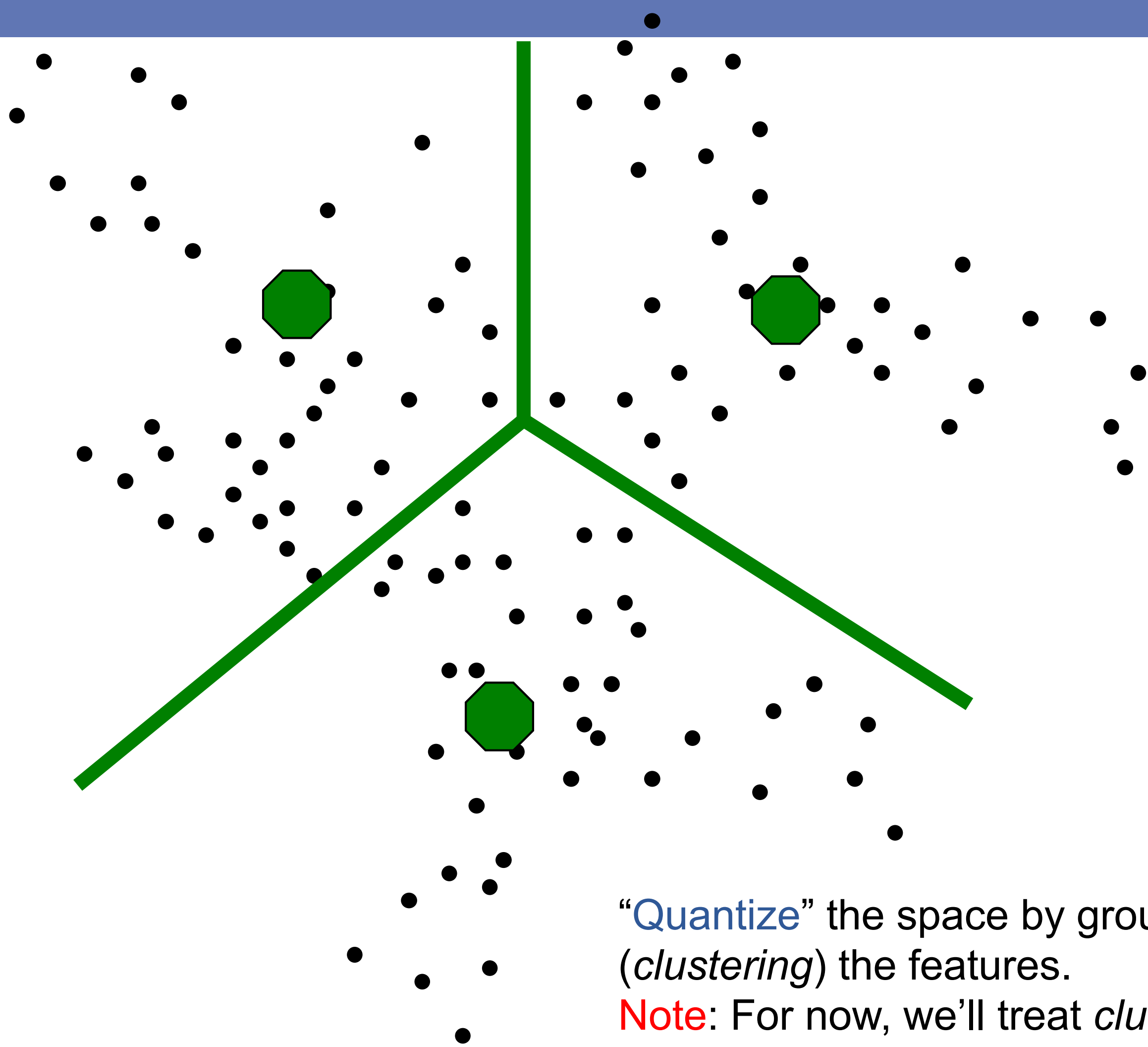
Visual Words: main idea

D. Nister, CVPR 2006



Each point is a local
descriptor, e.g. SIFT
feature vector.

D. Nister, CVPR 2006



"Quantize" the space by grouping (*clustering*) the features.
Note: For now, we'll treat *clustering* as a black box.

D. Nister, CVPR 2006

Visual Words

- Patches on the right = regions used to compute SIFT
- Each group of patches belongs to the same "visual word"

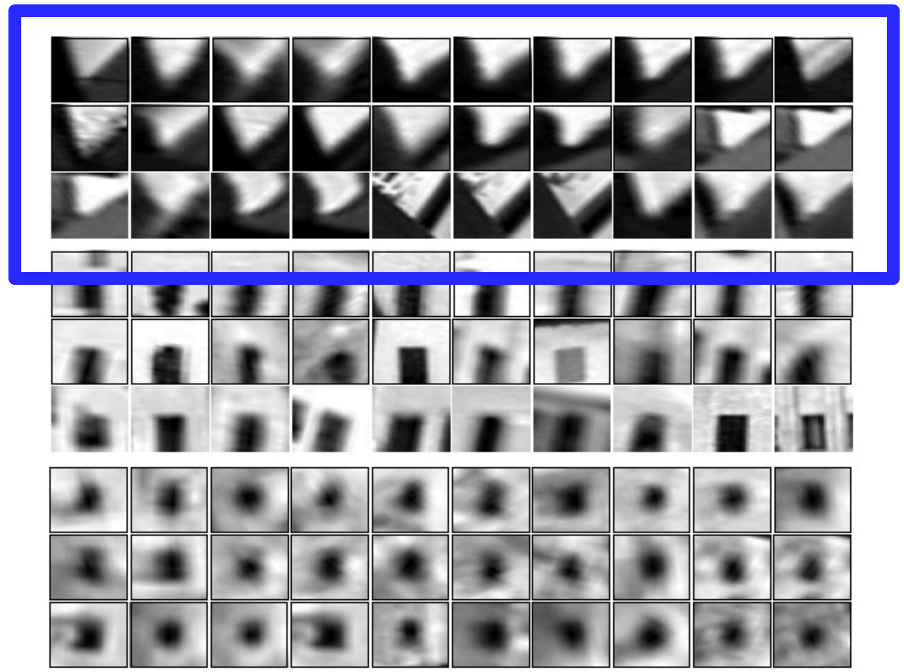
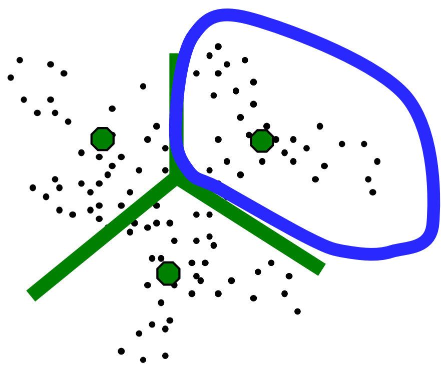


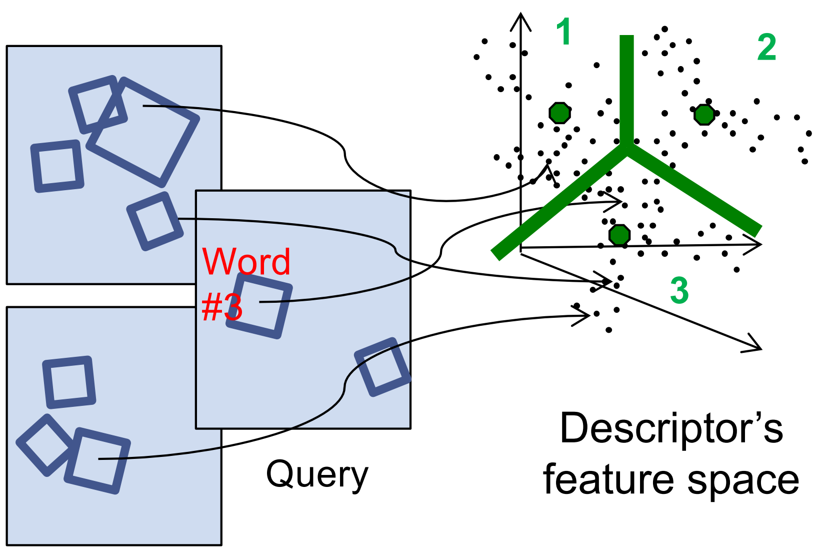
Figure from Sivic & Zisserman, ICCV 2003



Adapted from K. Grauman

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

Adapted from K. Grauman

Inverted File Index

Database images

Image #1

Image #2

Image #3

⋮

Word #	Image #
1	3
2 …	
7	
8	
9	
10 …	
91	
⋮	⋮

…

- 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

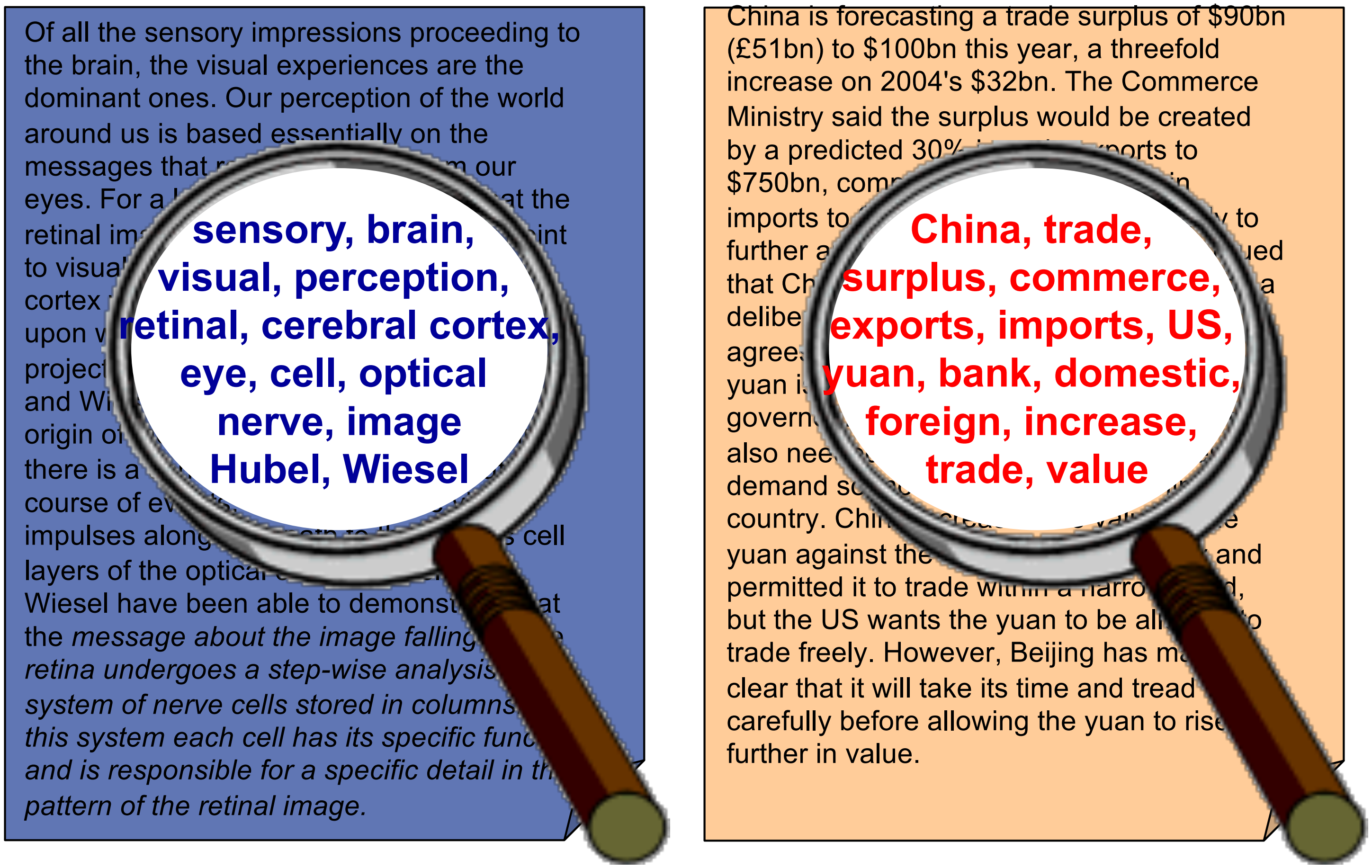
Image #1

Image #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)

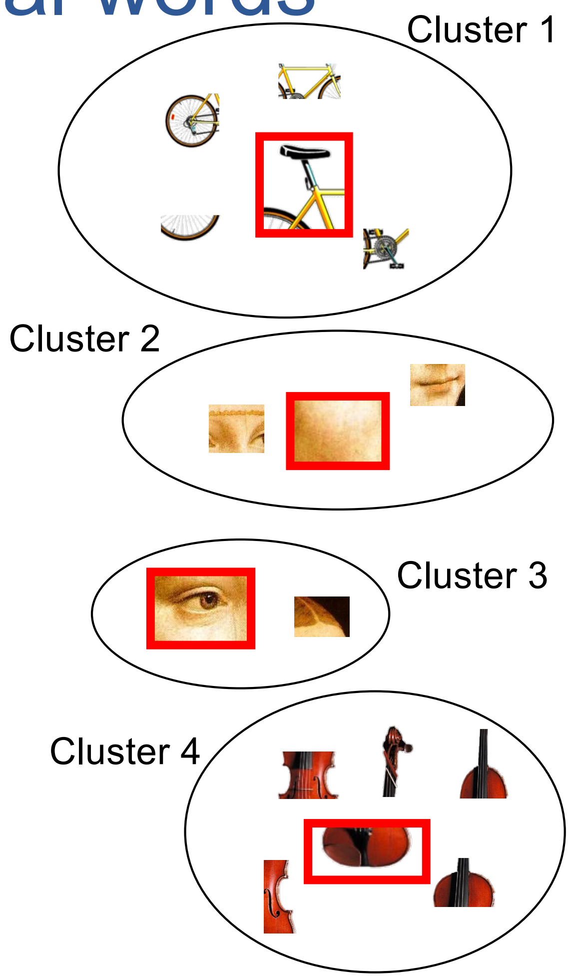
Adapted from K. Grauman

How to describe documents with words



Describing images with visual words

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



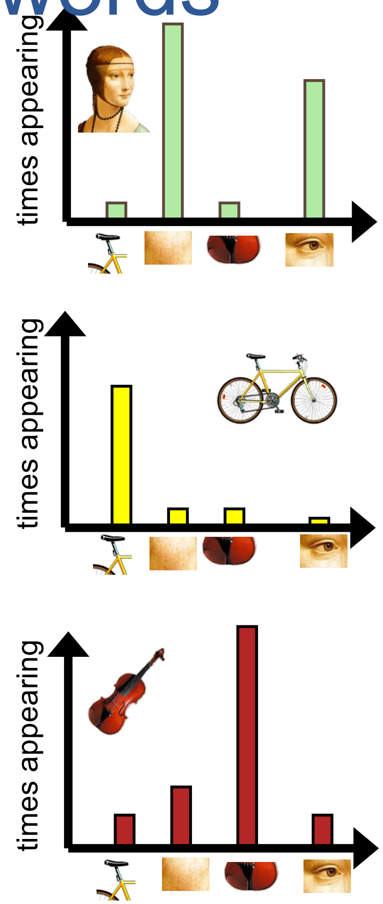
Feature patches:

Adapted from K. Grauman

Describing images with 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:



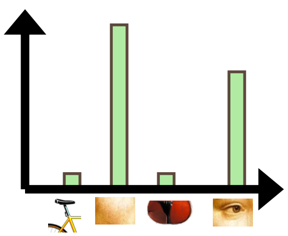
Visual words

K. Grauman

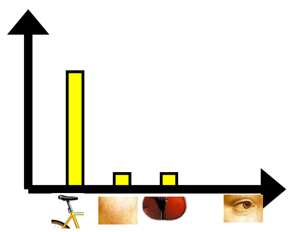
Comparing bags of words

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

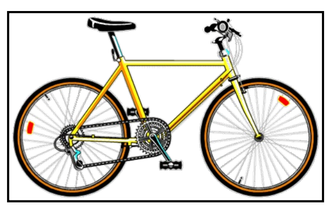
[1 8 1 4]



[5 1 1 0]



$\vec{d}_j$



$\vec{q}$

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

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

for vocabulary of *V* words

Adapted from K. Grauman

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

Adapted from K. Grauman

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

Adapted from K. Grauman

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

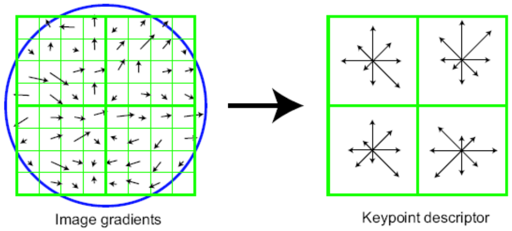
- Making Harris detection scale-invariant
  - "Indexing based on scale invariant interest points" by Krystian Mikolajczyk and Cordelia Schmid, in ICCV 2001 https://hal.archives-ouvertes.fr/file/index/docid/548276/filename/mikolajcICCV2001.pdf

- 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

Adapted from D. Hoiem, K. Grauman