

CS 2770: Local features: detection, description and matching

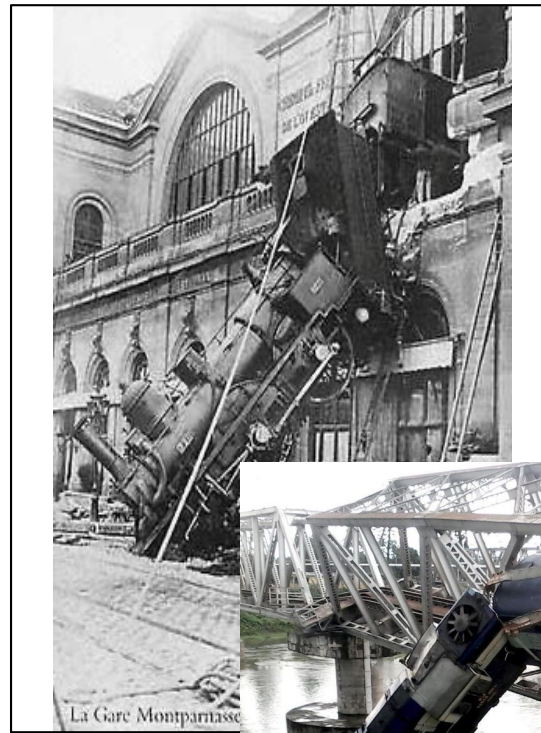
PhD. Nils Murrugarra-Llerena
nem177@pitt.edu



Plan for this lecture

- Feature detection / keypoint extraction
 - Corner 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

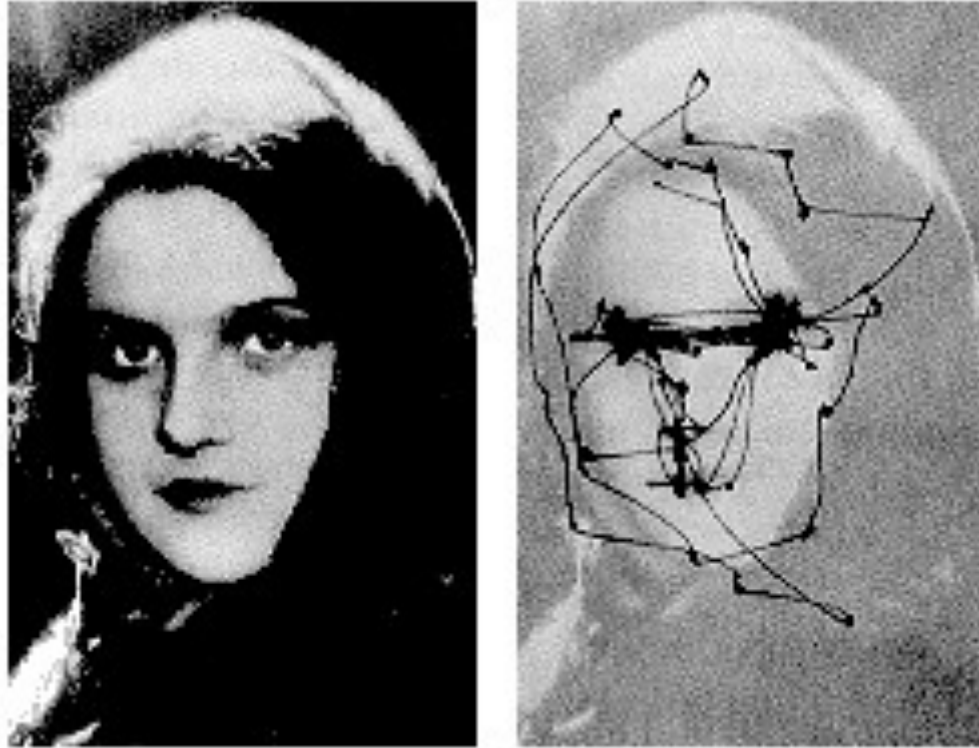


Adapted from S. Narasimhan

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



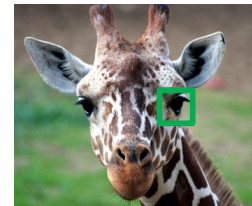
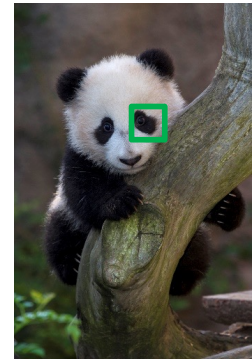
Yarbus eye tracking

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

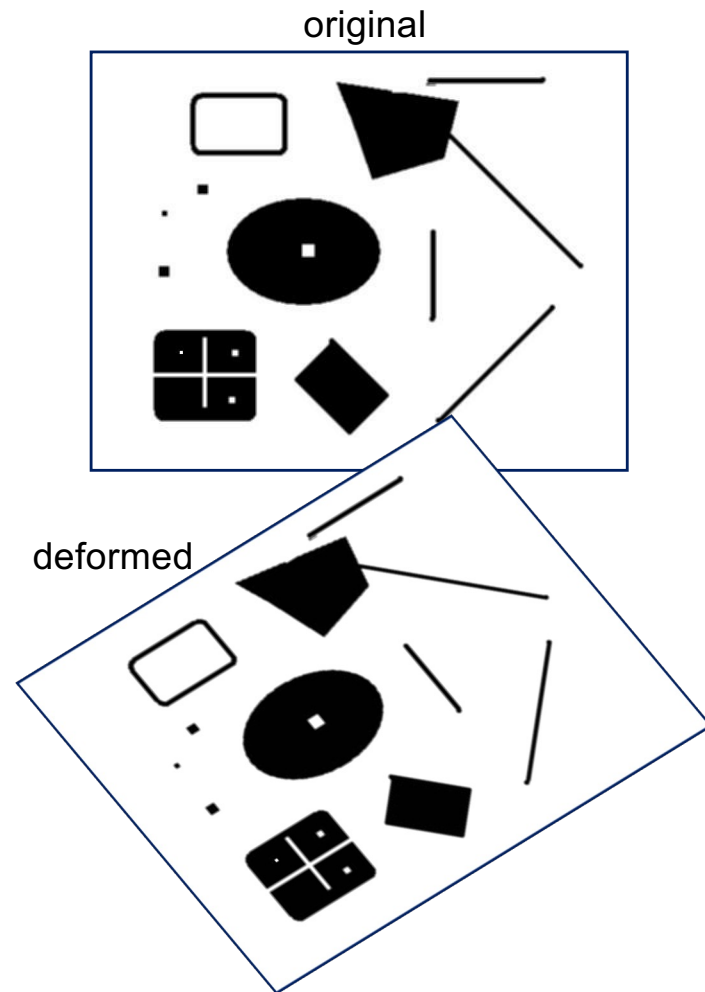


Interest(ing) points

- 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?

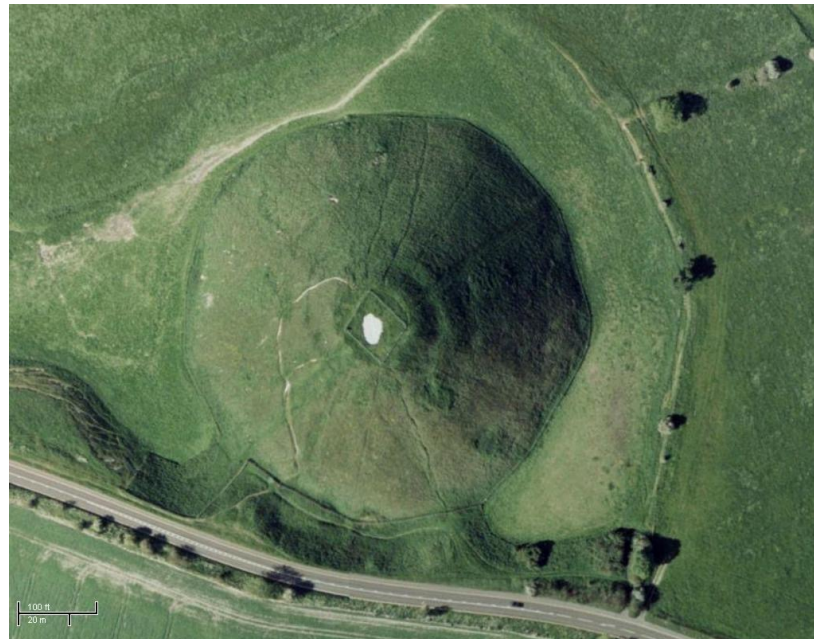
→ Corner detection



Choosing interest points

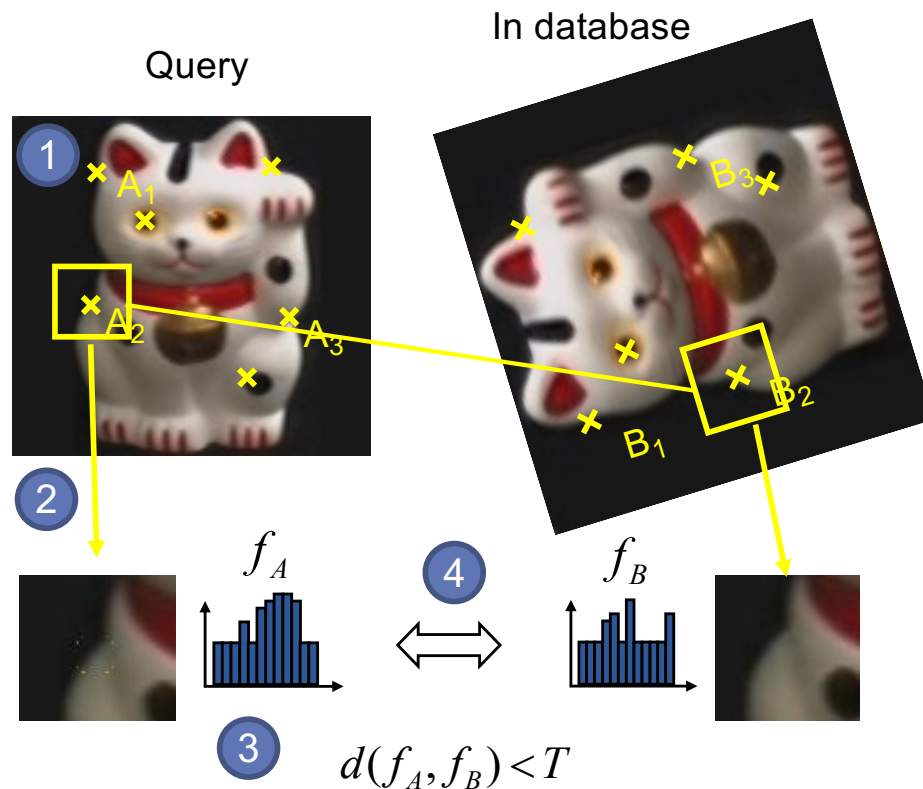
Where would you tell
your friend to meet you?

→ Blob detection



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



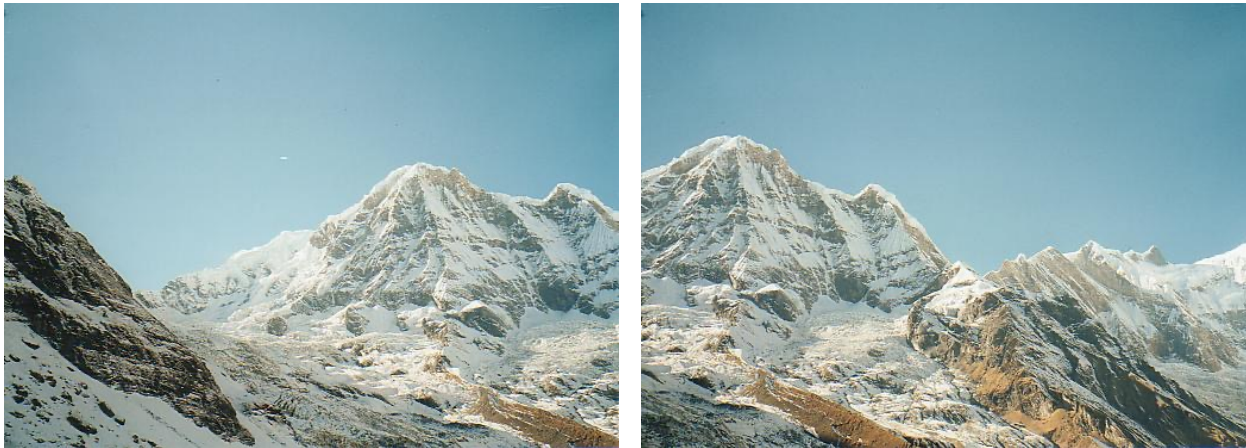
Goal:

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

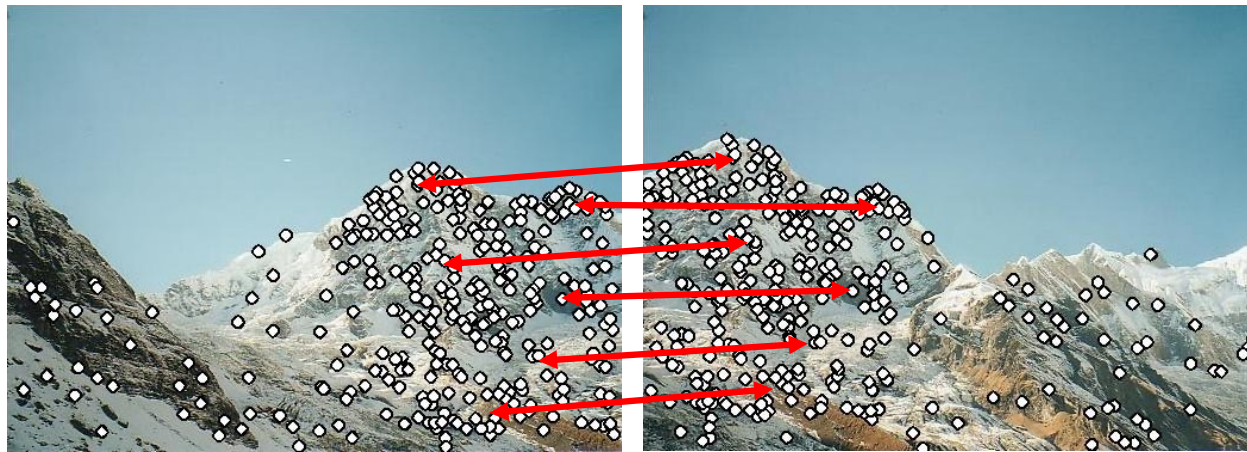
Application 2: Panorama stitching

- 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

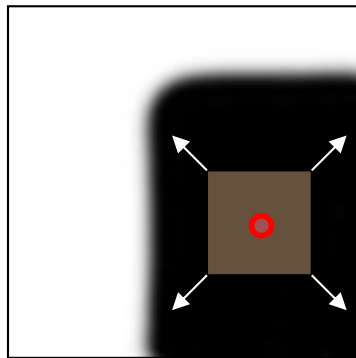
Step 2: match features

Step 3: align images

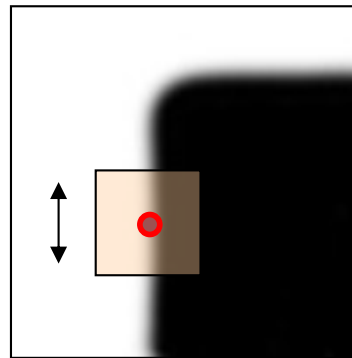
Corners are distinctive interest points

- 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

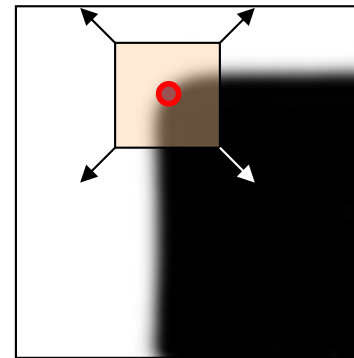
● Candidate keypoint



“flat” region:
no change in
all directions



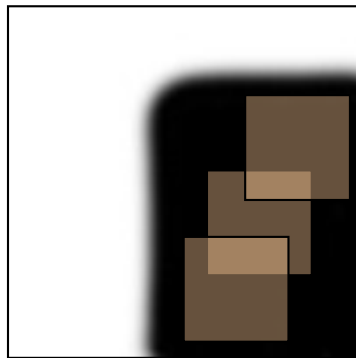
“edge”:
no change along
the edge direction



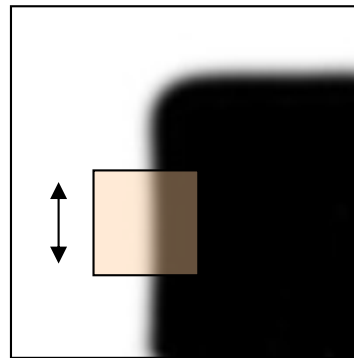
“corner”:
significant change
in all directions

Corners are distinctive interest points

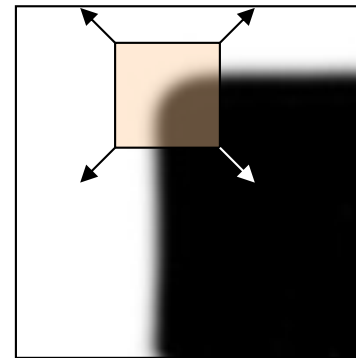
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



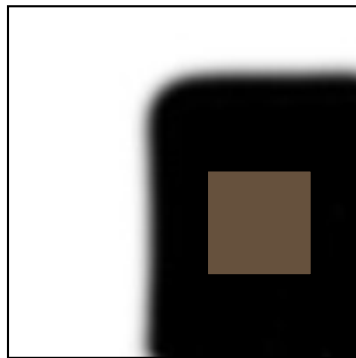
“edge”:
no change along
the edge direction



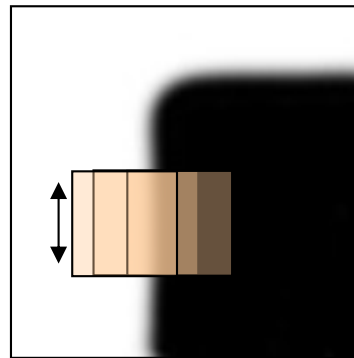
“corner”:
significant change
in all directions

Corners are distinctive interest points

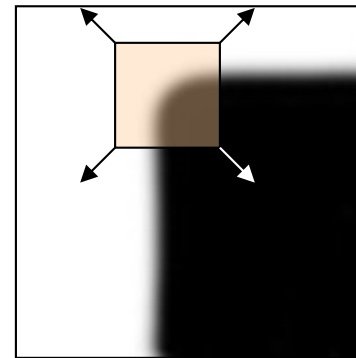
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



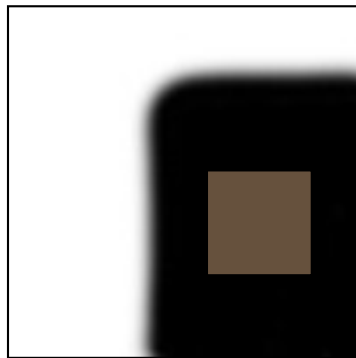
“edge”:
no change along
the edge direction



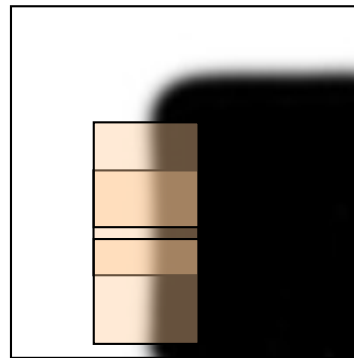
“corner”:
significant change
in all directions

Corners are distinctive interest points

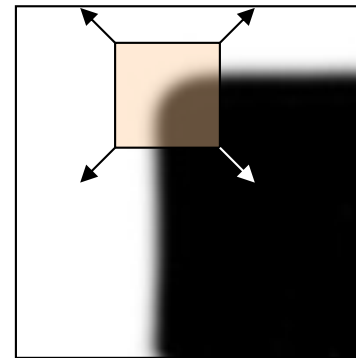
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



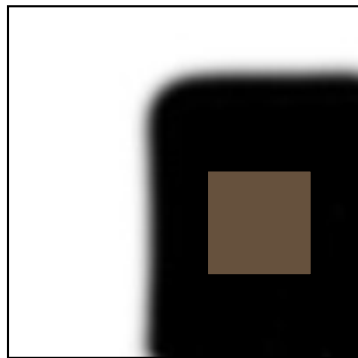
“edge”:
no change along
the edge direction



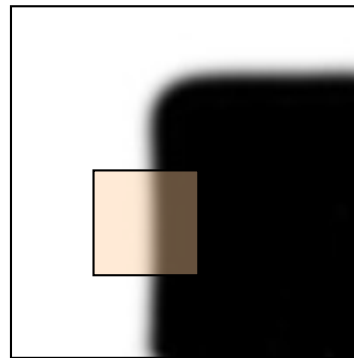
“corner”:
significant change
in all directions

Corners are distinctive interest points

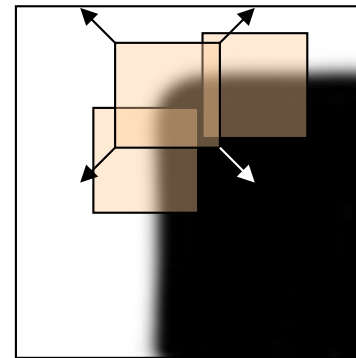
- We should easily recognize the keypoint by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



“edge”:
no change along
the edge direction



“corner”:
significant change
in all directions

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 $[u,v]$:

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

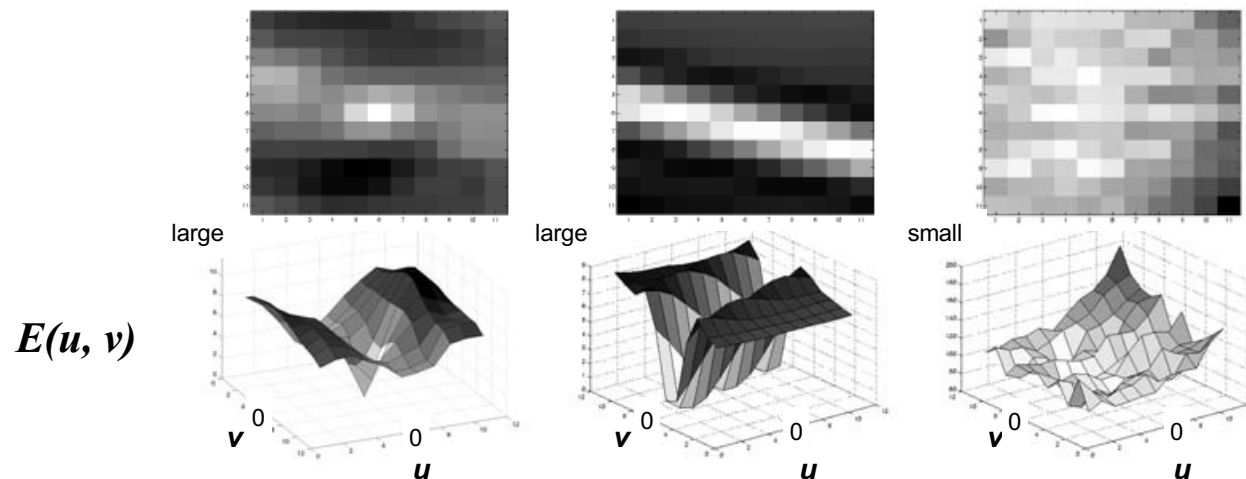
Shifted intensity

Intensity

Harris Detector: Mathematics

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

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



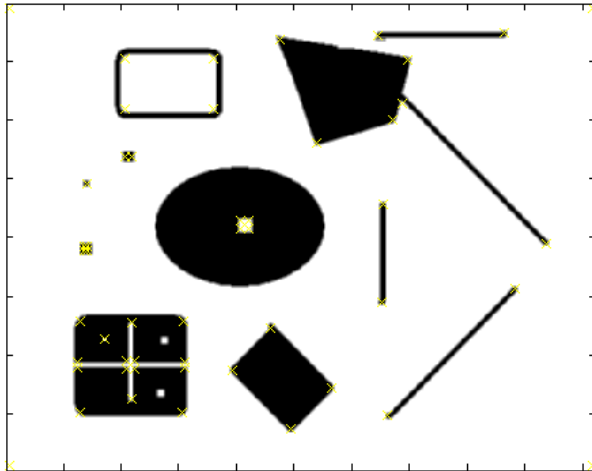
Adapted from D. Frolova, D. Simakov

Example of Harris Application

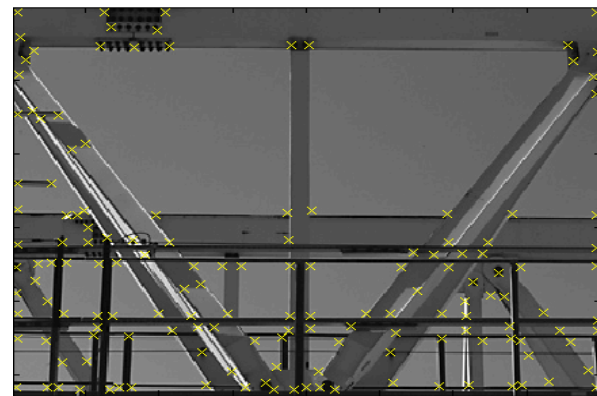


K. Grauman

More Harris Responses



Effect: A very precise corner detector.



Plan for this lecture

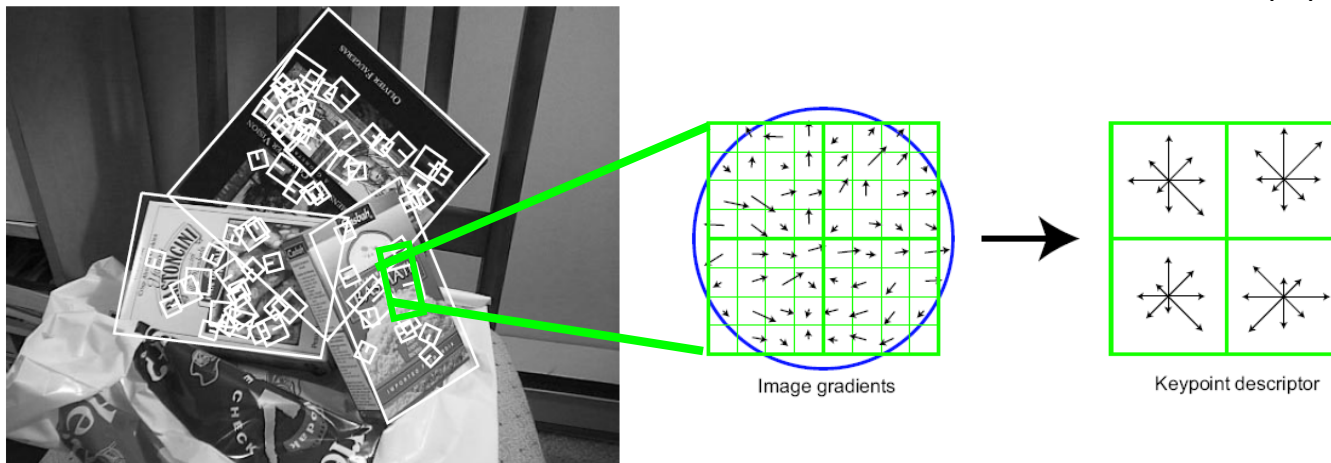
- Feature detection / keypoint extraction
 - Corner detection
- Feature description (of detected features)
- Matching features across images

Geometric 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

[Lowe, ICCV 1999]

K. Grauman, B. Leibe

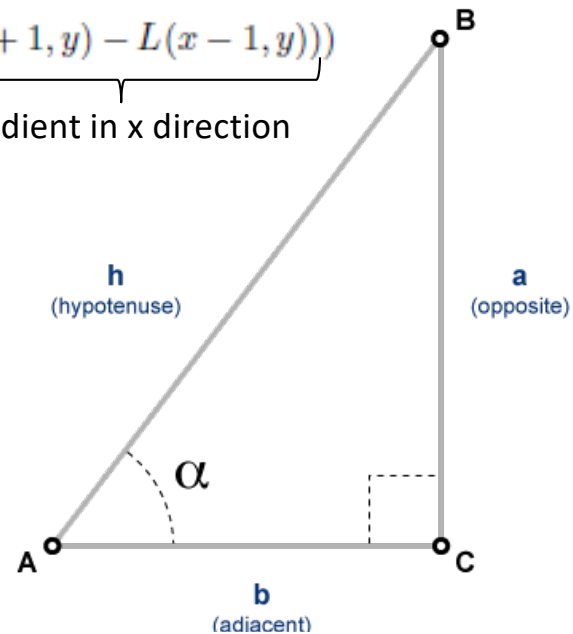
Computing gradients

L = the image intensity

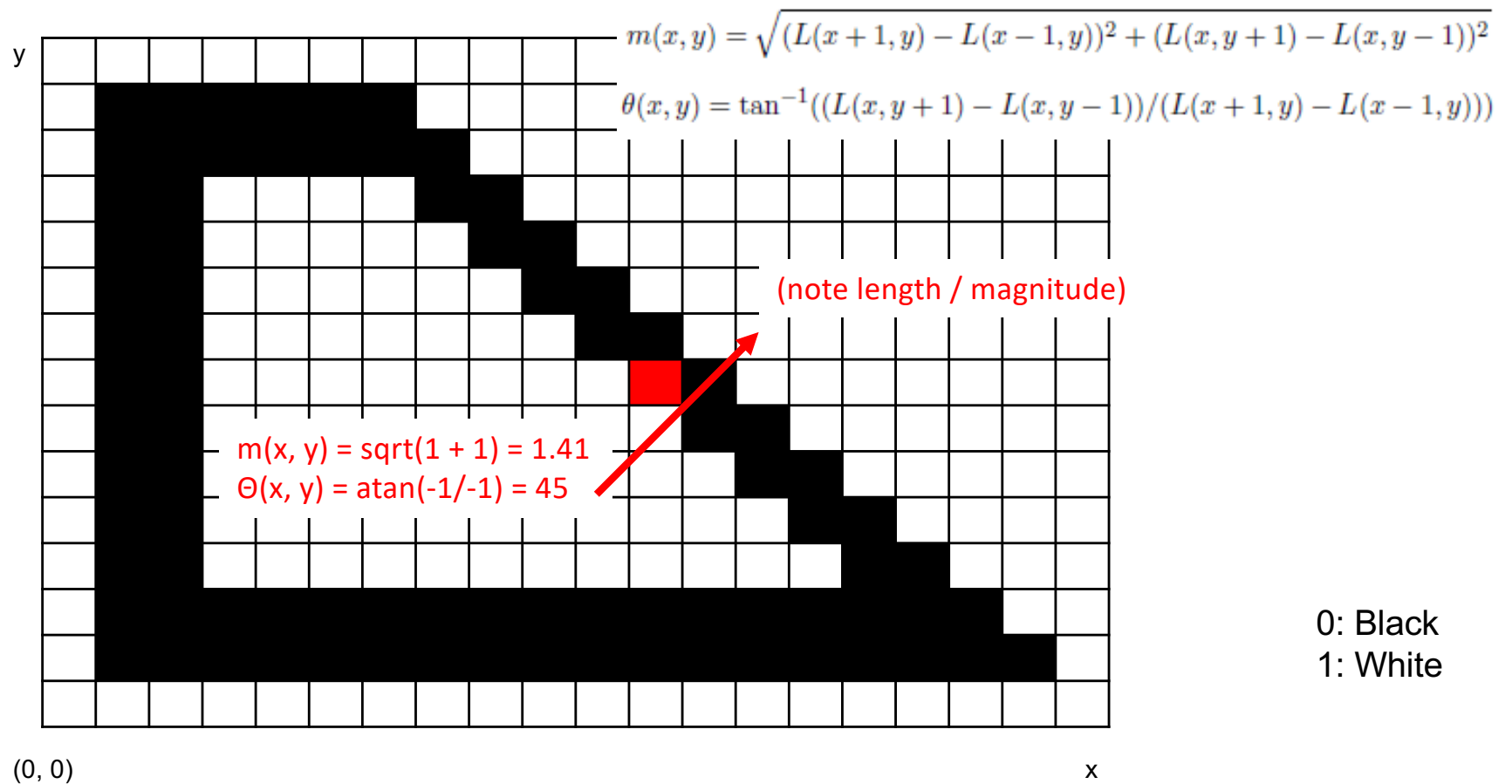
$$m(x, y) = \sqrt{\underbrace{(L(x+1, y) - L(x-1, y))^2}_{\text{gradient in x direction}} + \underbrace{(L(x, y+1) - L(x, y-1))^2}_{\text{gradient in y direction}}}$$

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

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



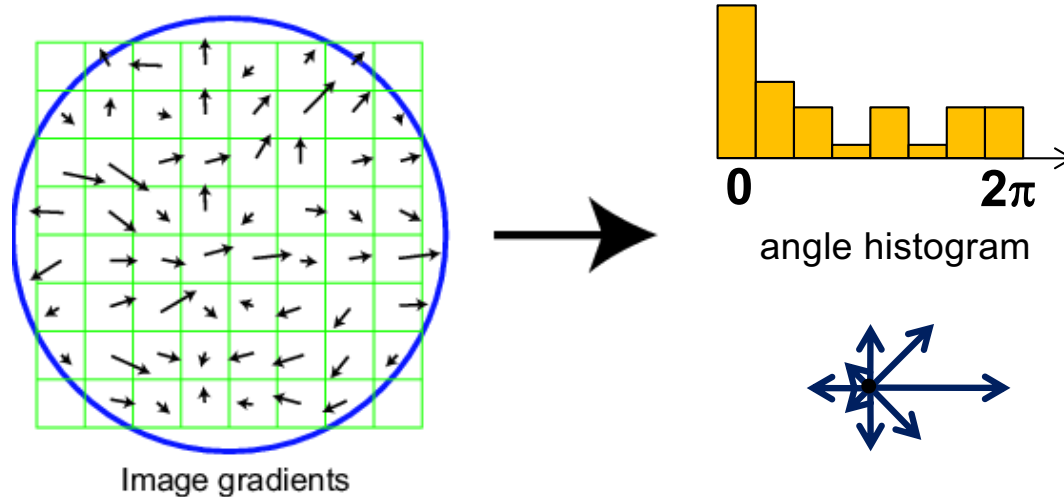
Gradients



Scale Invariant Feature Transform

Basic idea:

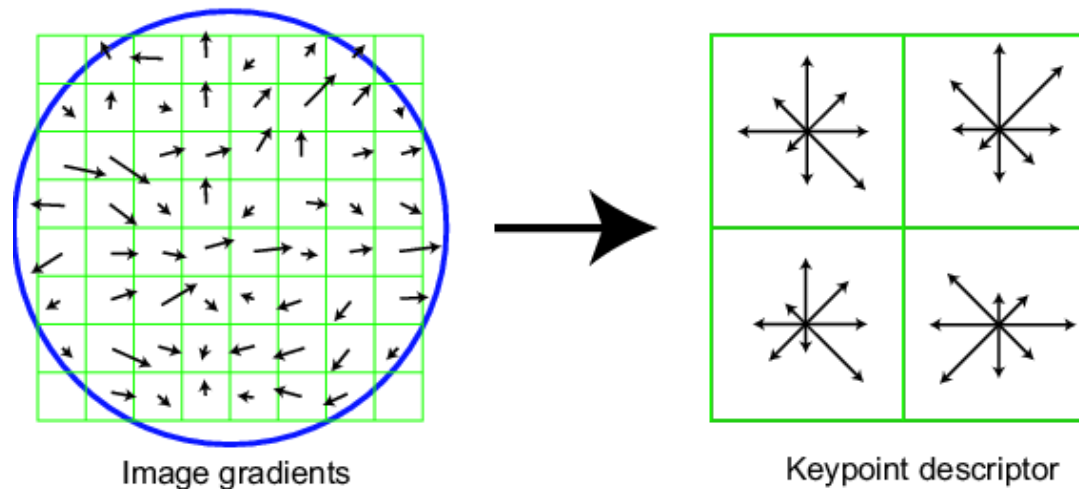
- Take 16x16 square window around detected feature
- Compute gradient orientation for each pixel
- Create histogram over edge orientations weighted by magnitude
- That's your feature descriptor!







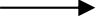




Scale Invariant Feature Transform

Full version

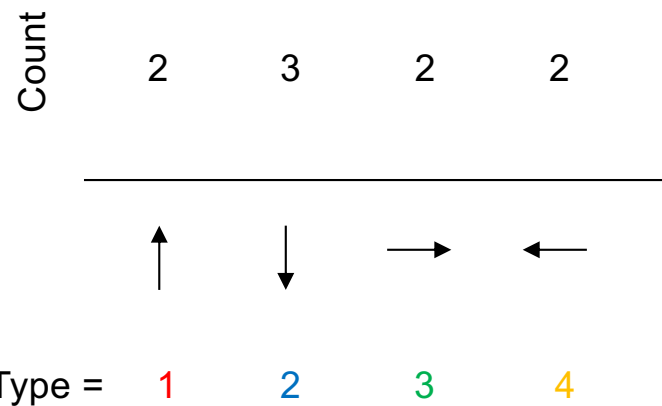
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Quantize the gradient orientations i.e. snap each gradient to one of 8 angles
- Each gradient contributes not just 1, but magnitude(gradient) to the histogram, i.e. stronger gradients contribute more
- 16 cells * 8 orientations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



Scale Invariant Feature Transform

 1	 3	 1
 2	 3	 2
 4	 2	 4










Uniform weight (ignore magnitude)



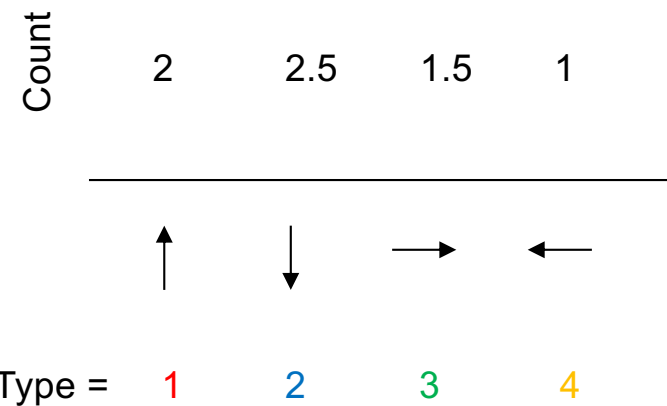
Gradients

Histogram of gradients

Scale Invariant Feature Transform

 1	 3	 1
 2	 3	 2
 4	 2	 4

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



Gradients

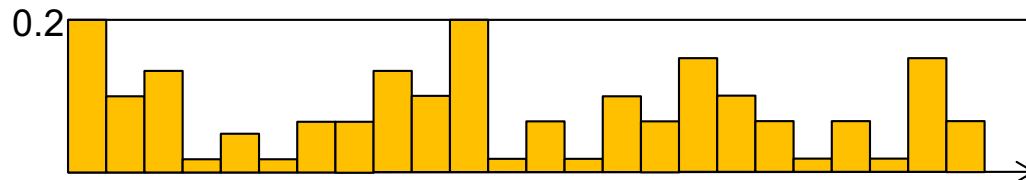
Histogram of gradients

Scale Invariant Feature Transform

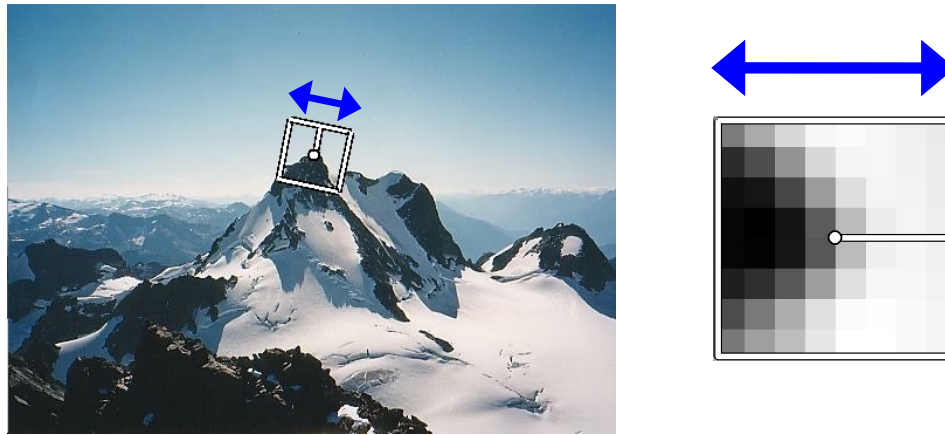
Full version

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

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



Making descriptor rotation invariant

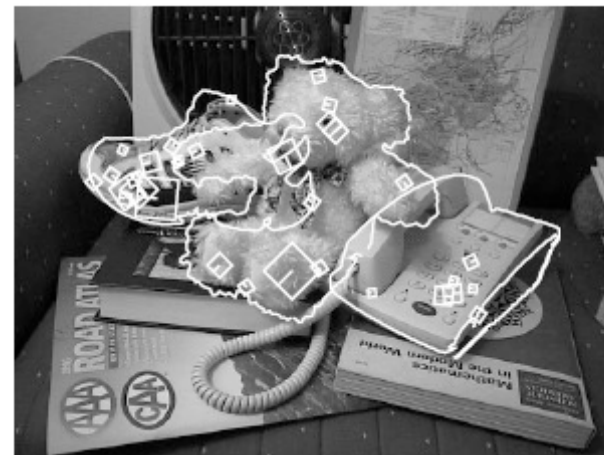


- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation

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
- **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



Applications of local invariant features

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



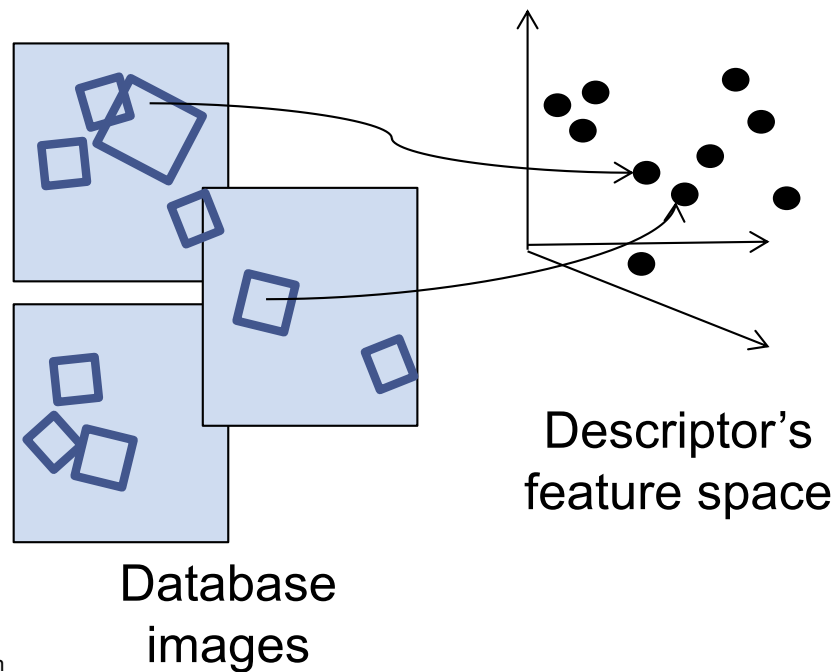
<http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html>

Plan for this lecture

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

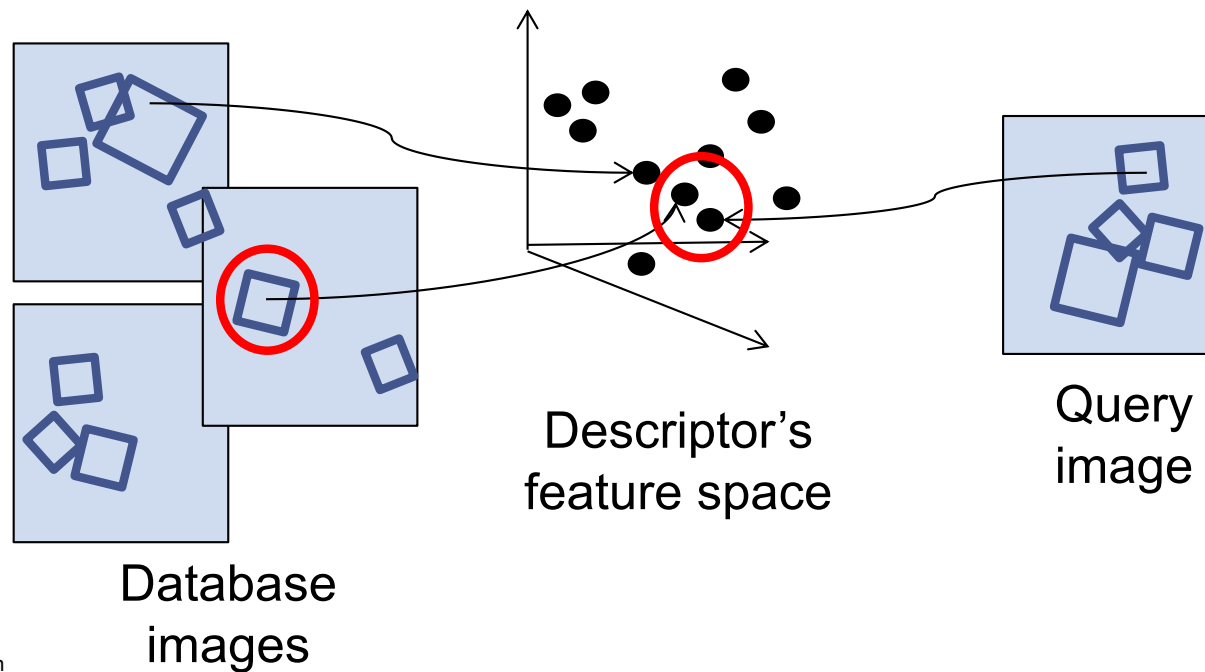
Matching Local Features Setup

- 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



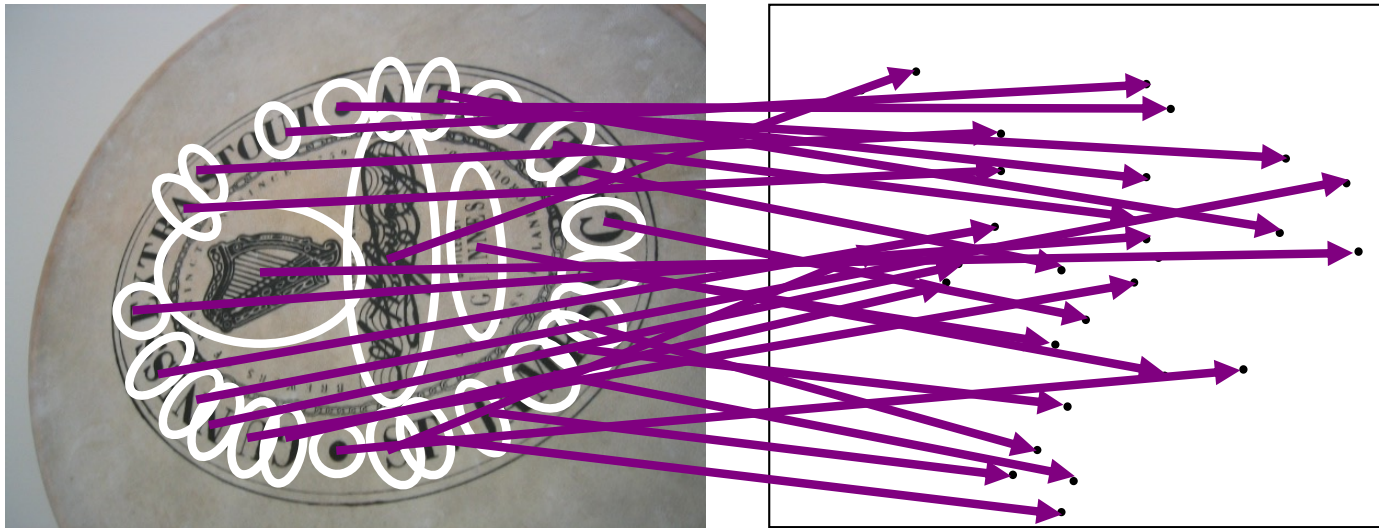
Indexing local features

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	COC, 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	Cansveral 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; 63
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 Stns; 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	ChIPLEY; 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 MacIay 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	Ft 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
Appletor, 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	Cleanwater 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 Mallmen; 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
A4-C Skyhawk; 117	Daniel Boone, Florida Walk; 117	Ticket System; 190
Atrium; 121	Daytona Beach; 172-173	Toll Plazas; 190
Blue Springs SP; 87	De Land; 87	Ford, Henry; 152
	De Soto, Hernando,	Fort Barrancas; 122
	Anhaica; 108-109,146	Buried Alive; 123
	County; 149	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".

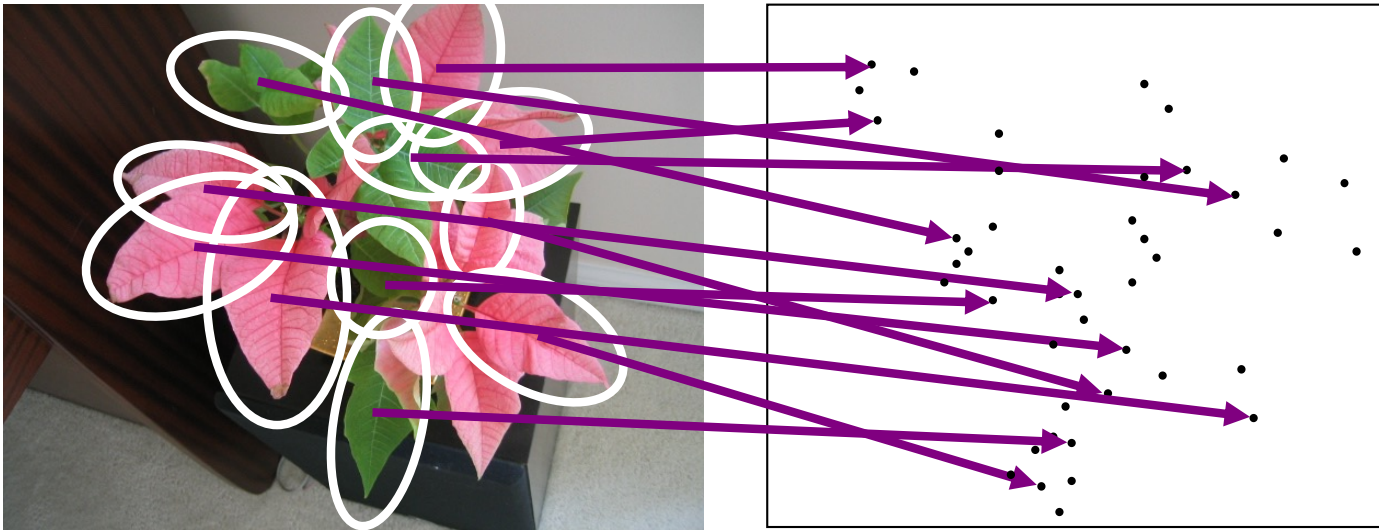
Visual Words: main idea

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

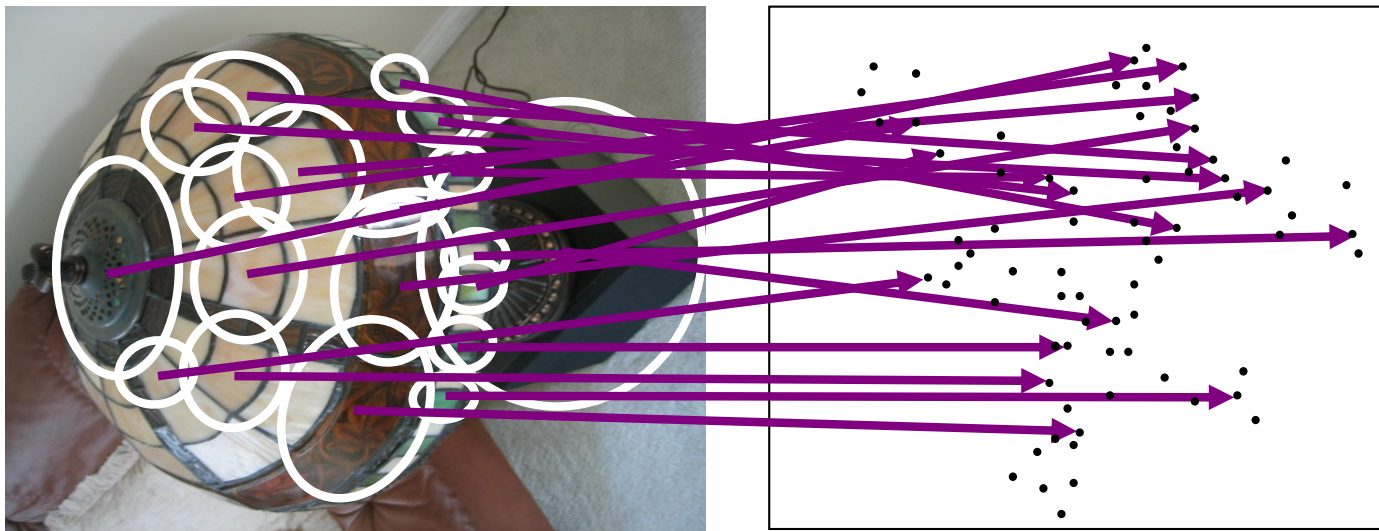


e.g., SIFT descriptor space: each point is 128-dimensional

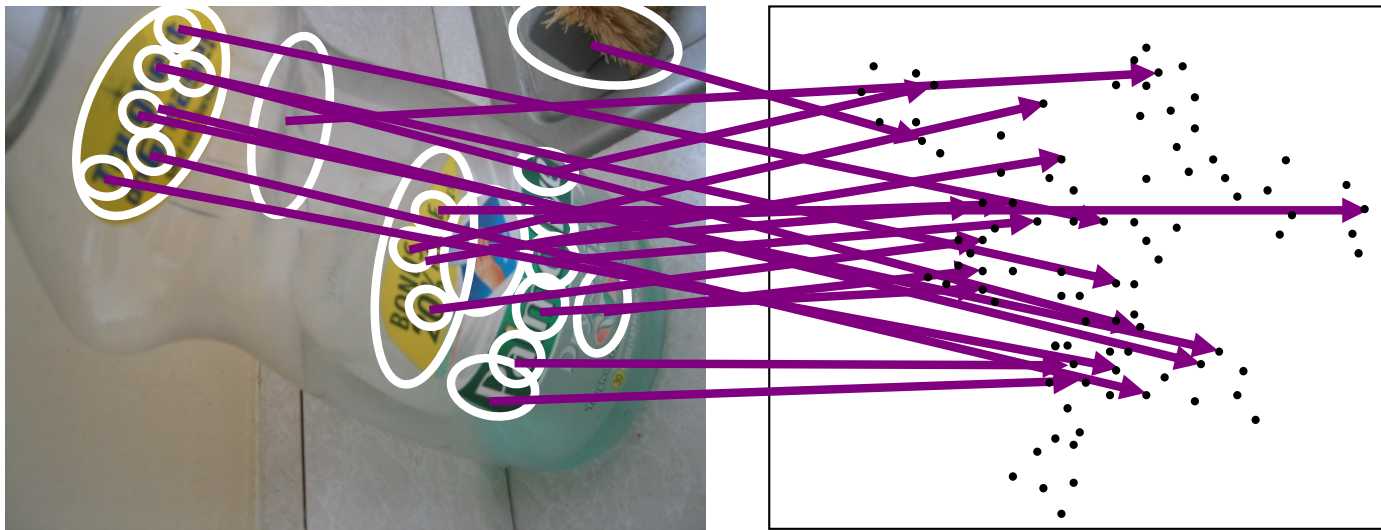
Visual Words: main idea

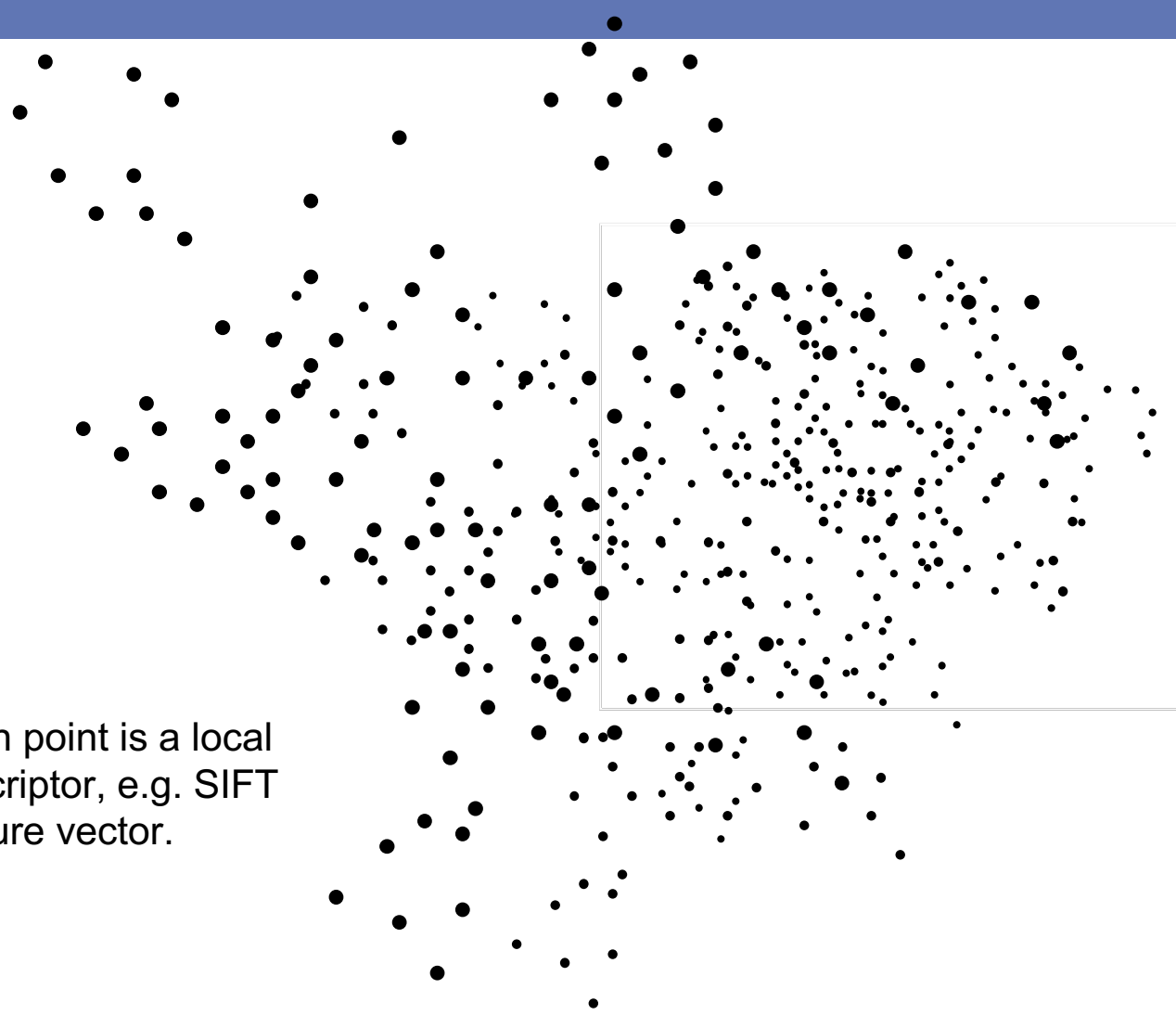


Visual Words: main idea

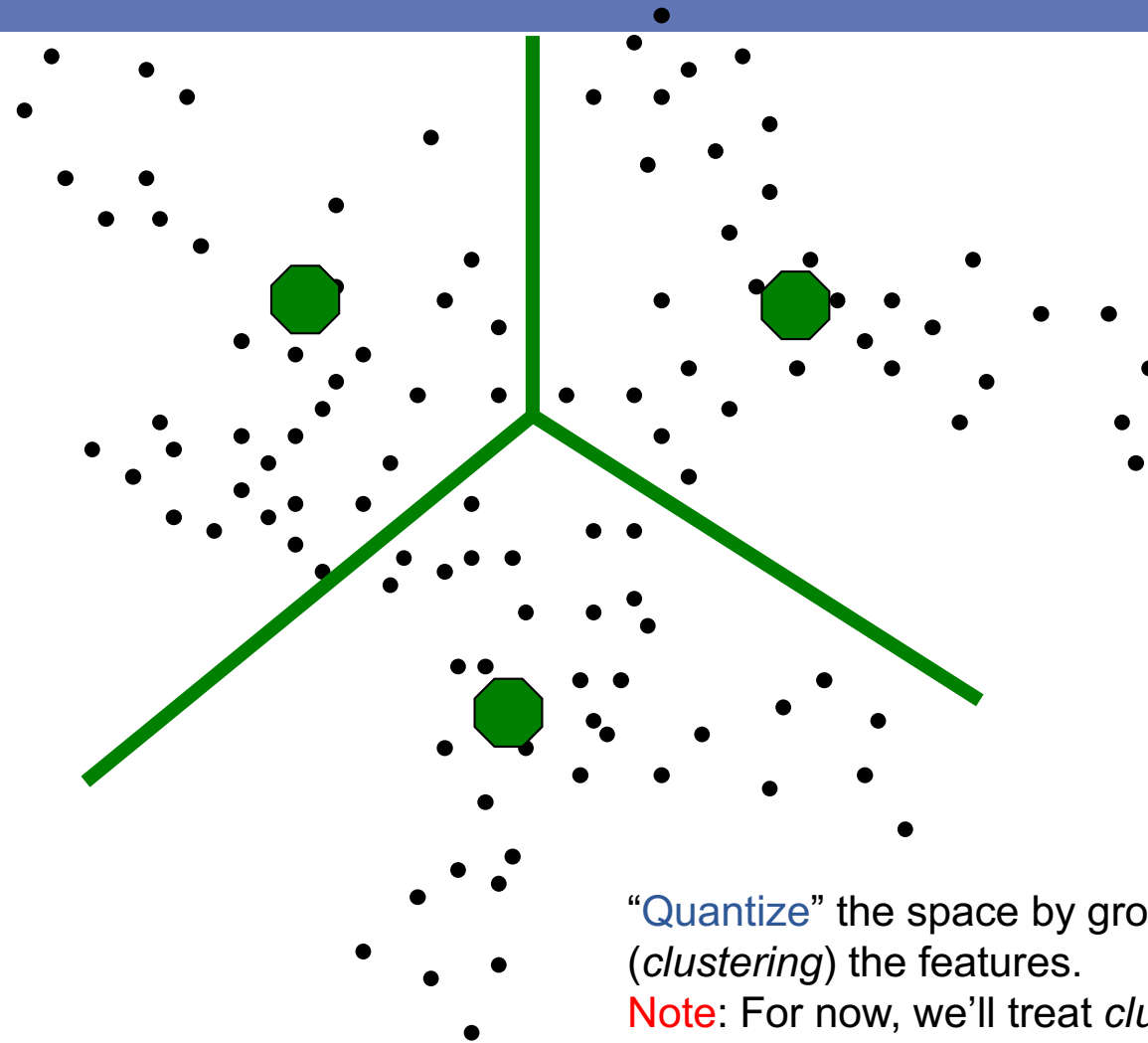


Visual Words: main idea





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



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

Visual Words

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

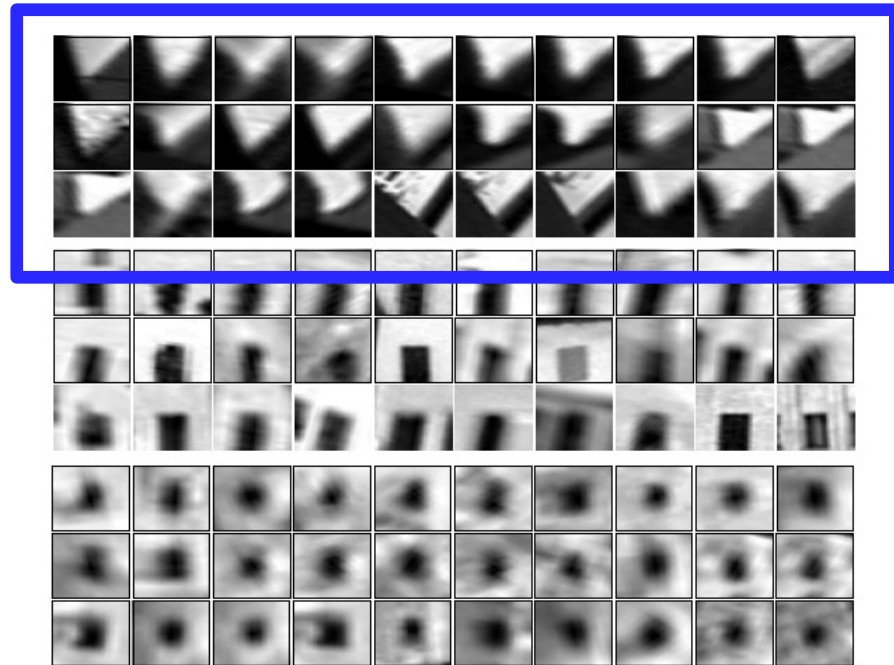
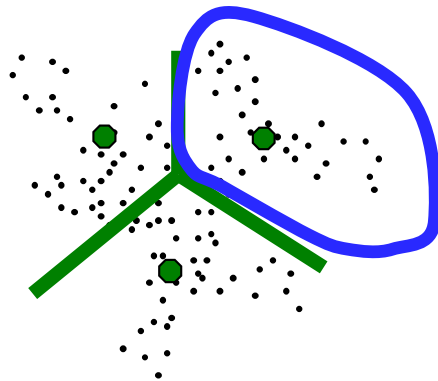
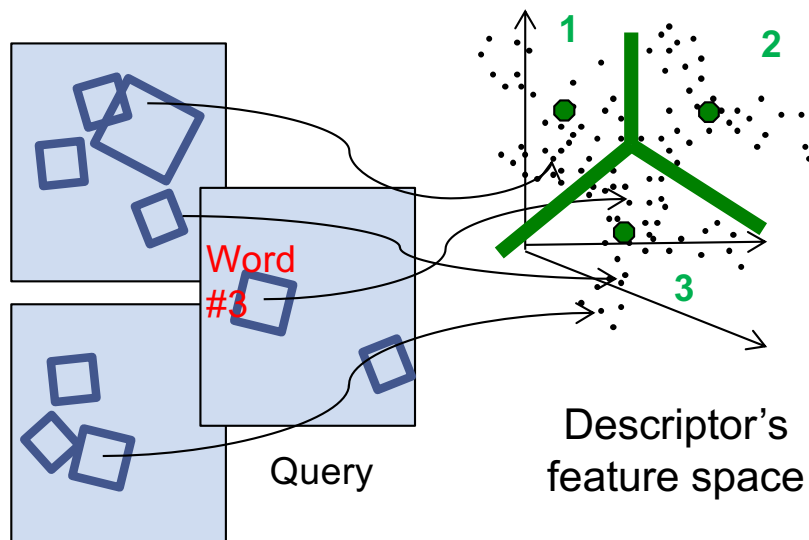


Figure from Sivic & Zisserman, ICCV 2003

Visual Words for Indexing

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

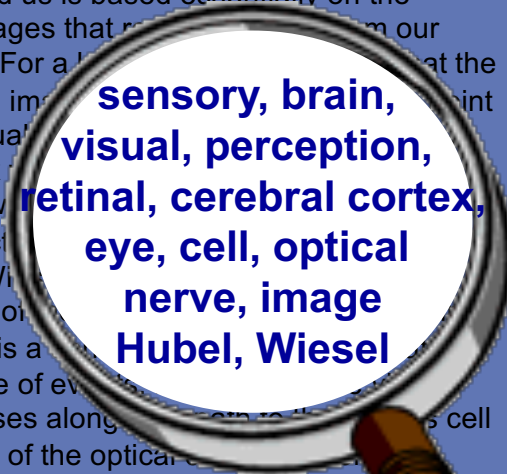


Adapted from K. Grauman

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

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 our eyes. For a long time, it was thought that the retinal image is projected directly to visual cortex. However, Hubel and Wiesel, upon viewing the projection of the retina upon the visual cortex, discovered that the projected image is first processed in the layers of the optical tectum. Hubel and Wiesel have been able to demonstrate that the message about the image falling on the retina undergoes a step-wise 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.



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

China is forecasting a trade surplus of \$90bn (£51bn) to \$100bn this year, a threefold increase on 2004's \$32bn. The Commerce Ministry said the surplus would be created by a predicted 30% increase in exports to \$750bn, compared with \$570bn in 2004. Imports to the US are expected to increase by 10% to \$750bn, compared with \$680bn in 2004. The Commerce Ministry said the surplus would be created by a predicted 30% increase in exports to \$750bn, compared with \$570bn in 2004. Imports to the US are expected to increase by 10% to \$750bn, compared with \$680bn in 2004. The Commerce Ministry said the surplus would be created by a predicted 30% increase in exports to \$750bn, compared with \$570bn in 2004. Imports to the US are expected to increase by 10% to \$750bn, compared with \$680bn in 2004.



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

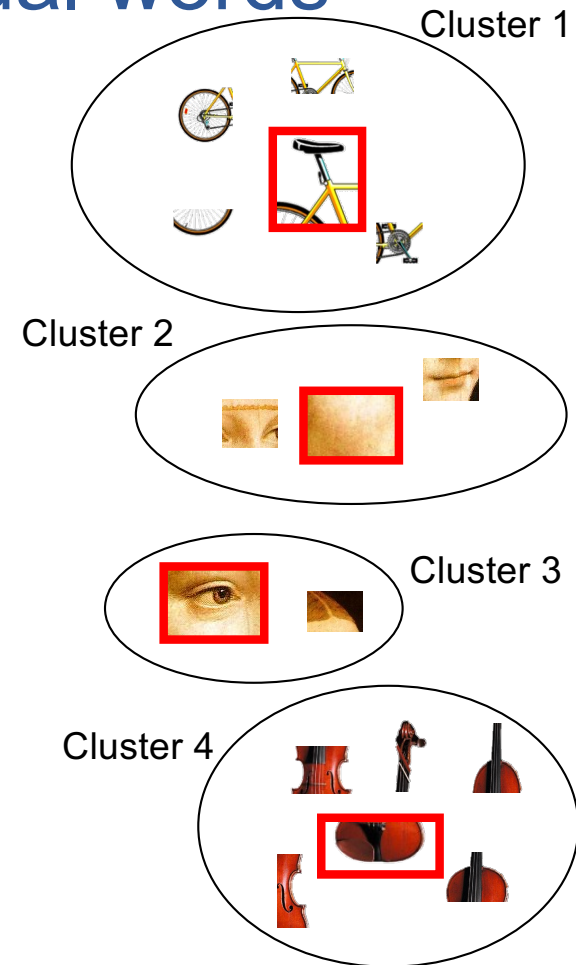
Describing images with visual words

- 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



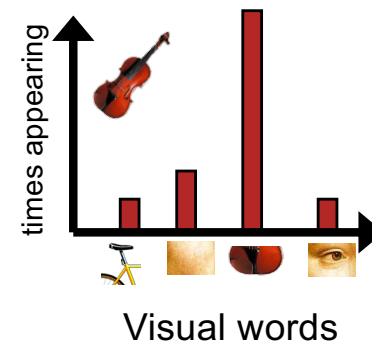
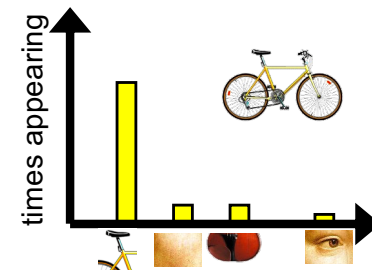
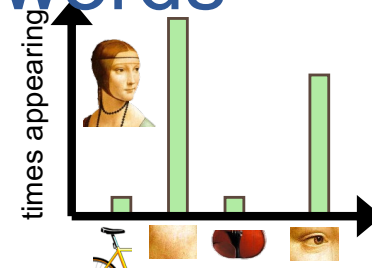
Describing images with visual words

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

Feature patches:

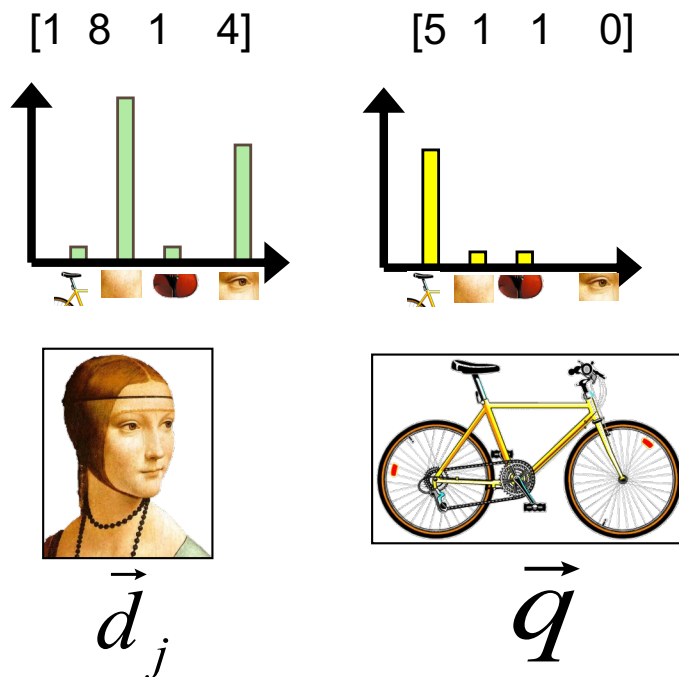


K. Grauman



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)



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

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

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

Summary

- Keypoint detection: repeatable and distinctive
- Descriptors: robust and selective
 - Histograms for robustness to small shifts and translations (SIFT descriptor)
- Matching: cluster and index
 - Compare images through their feature distribution

