CS 2770: Local features: detection, description and matching

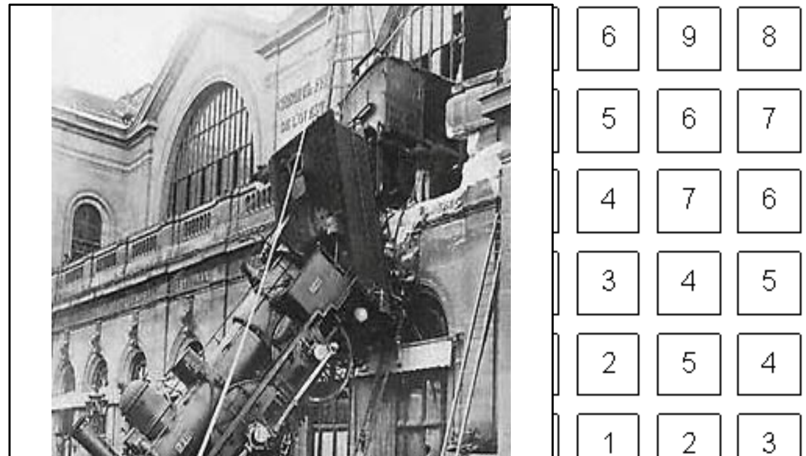
PhD. Nils Murrugarra-Llerena
nem177@pitt.edu

University of Pittsburgh

Plan for this lecture

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

An image is a set of pixels

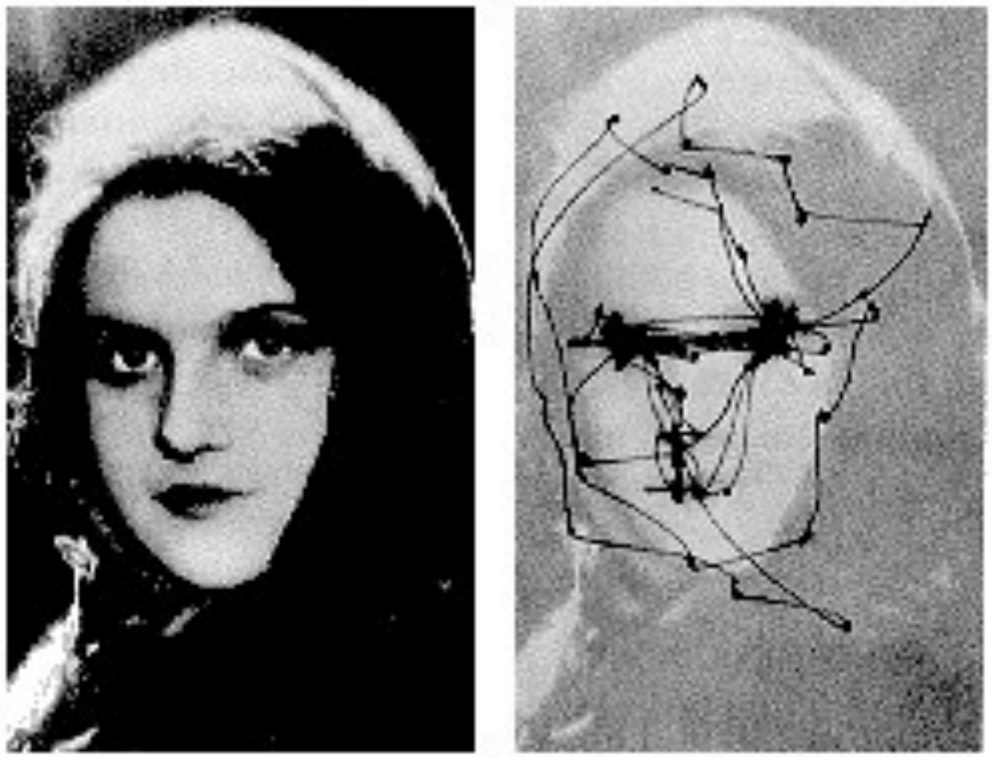


Adapted from S. Narasimhan

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

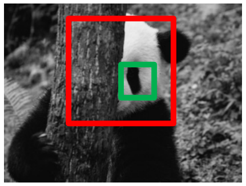
D. Hoiem

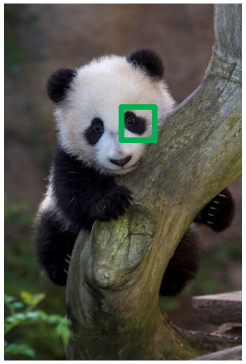
Local features

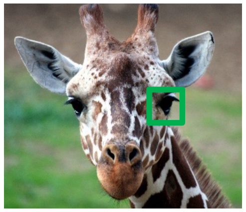
- *Local* means that they only cover a small part of the image
- There will be many local features detected in an image; later we'll use those to compute a representation of the whole image
- Local features usually exploit image gradients, ignore color
- Feature ~= *vector* of gradient statistics for 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

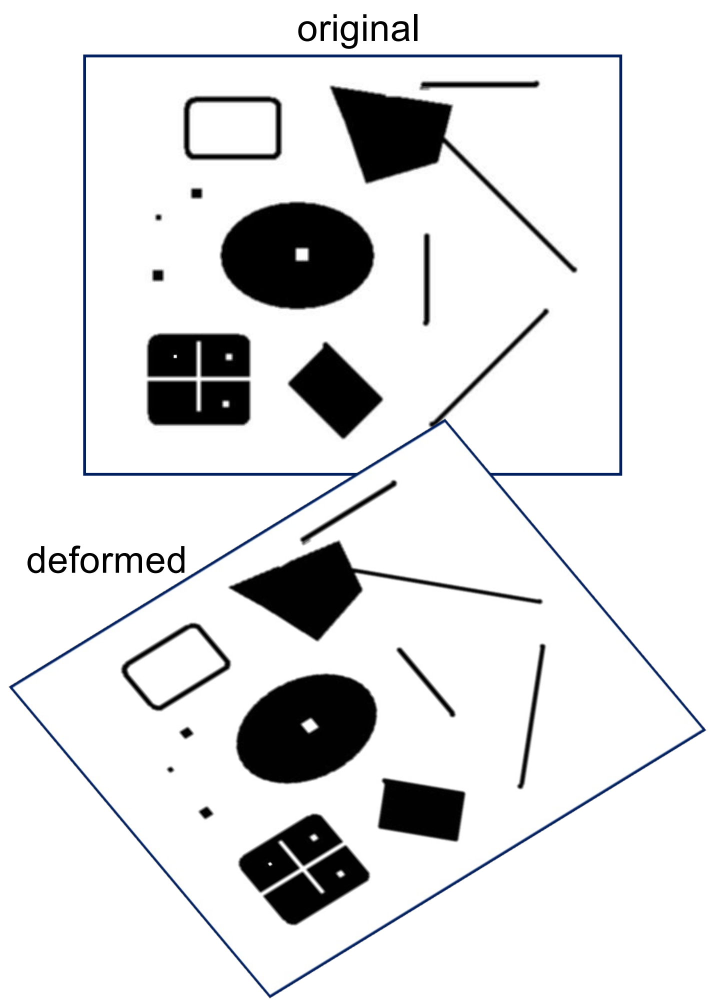
- Note: “interest points” = “keypoints”, also sometimes called “features”

- 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

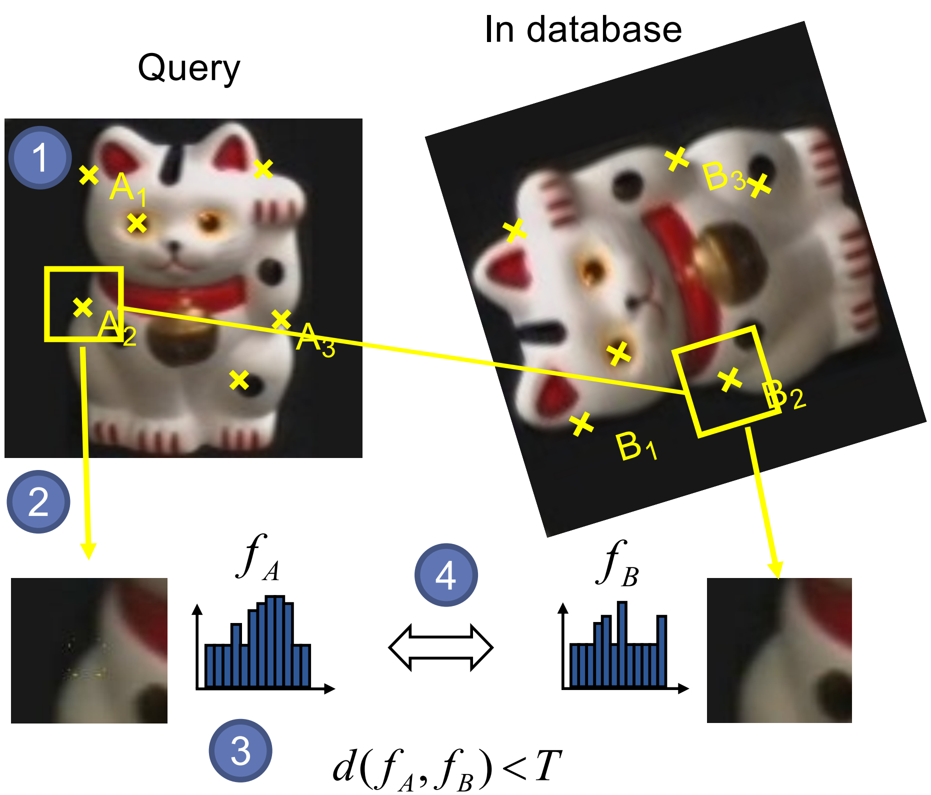
Choosing interest points

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:

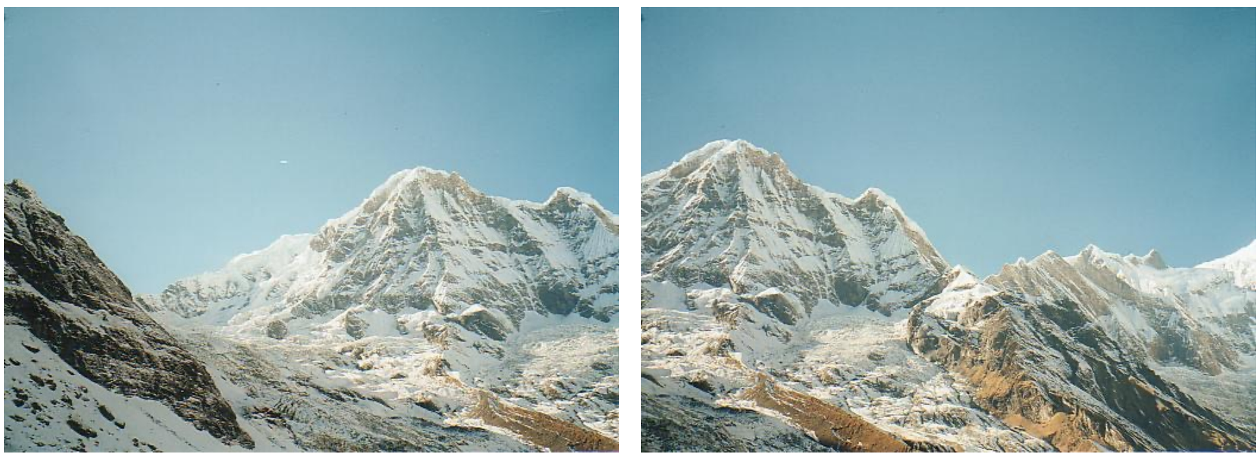
We want to detect *repeatable* and *distinctive* points

- *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

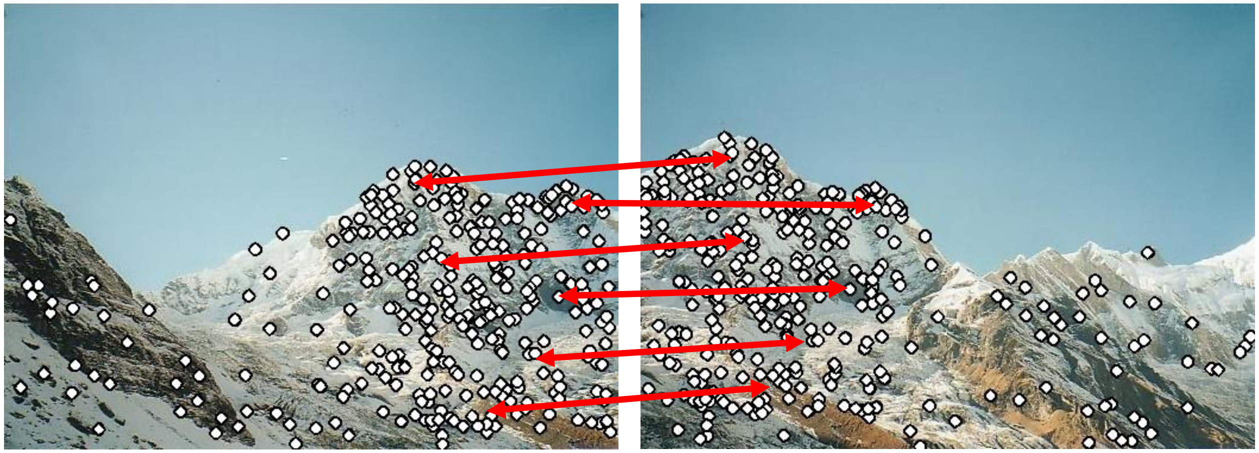
- We have two images – how do we combine them?



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

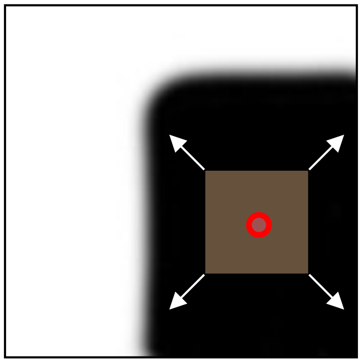
Step 3: align images

L. Lazebnik

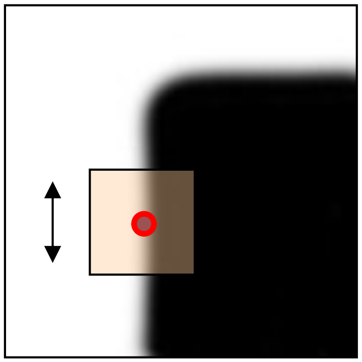
Corners are distinctive interest points

- 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

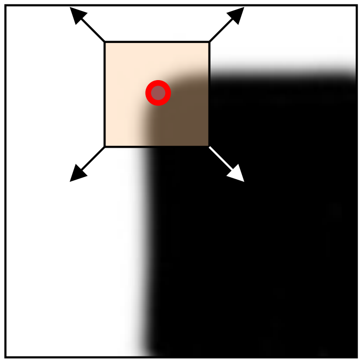
Candidate keypoint



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction

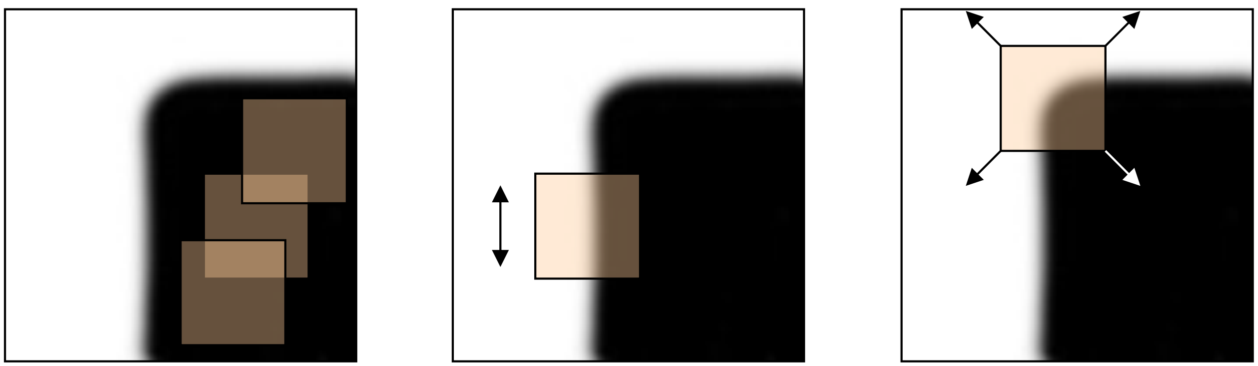


"corner":
significant change
in all directions

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

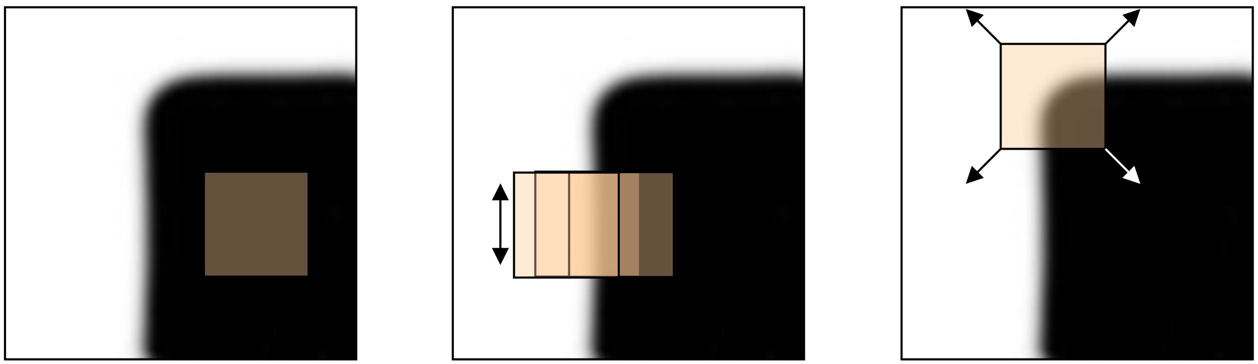
"edge":
no change along
the edge direction

"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

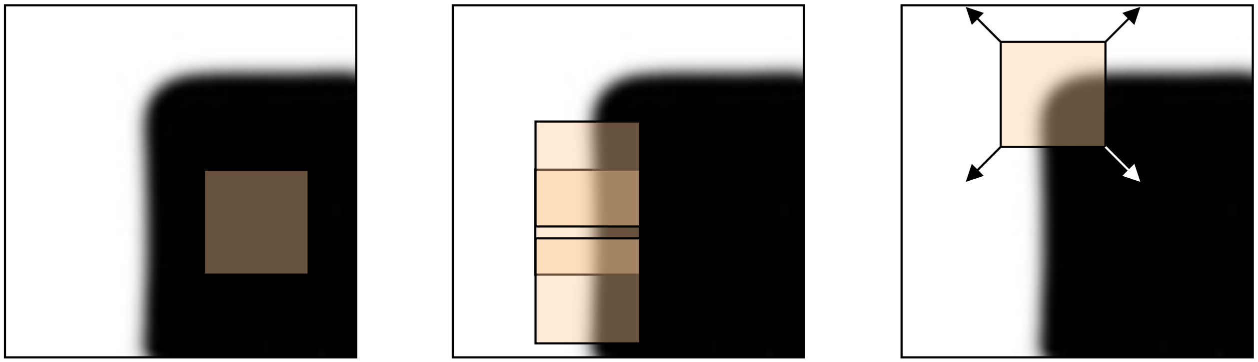
"edge":
no change along
the edge direction

"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

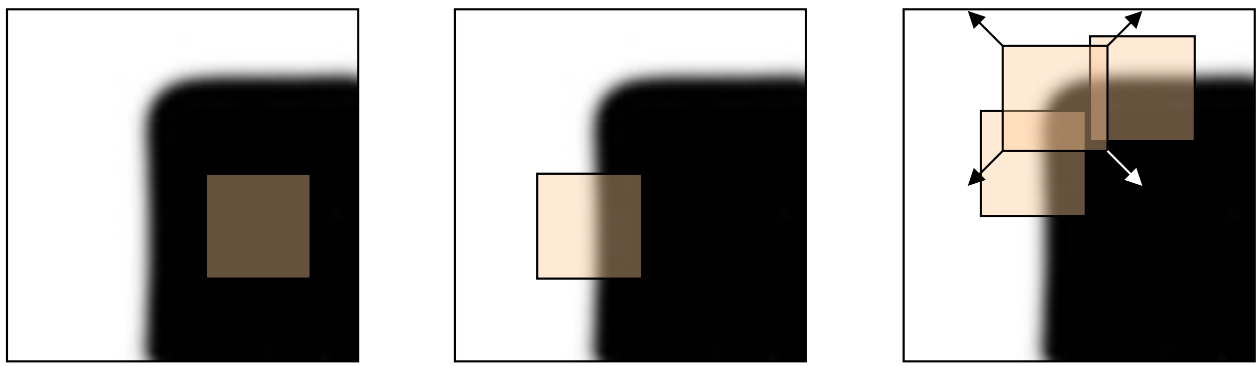
"edge":
no change along
the edge direction

"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]:

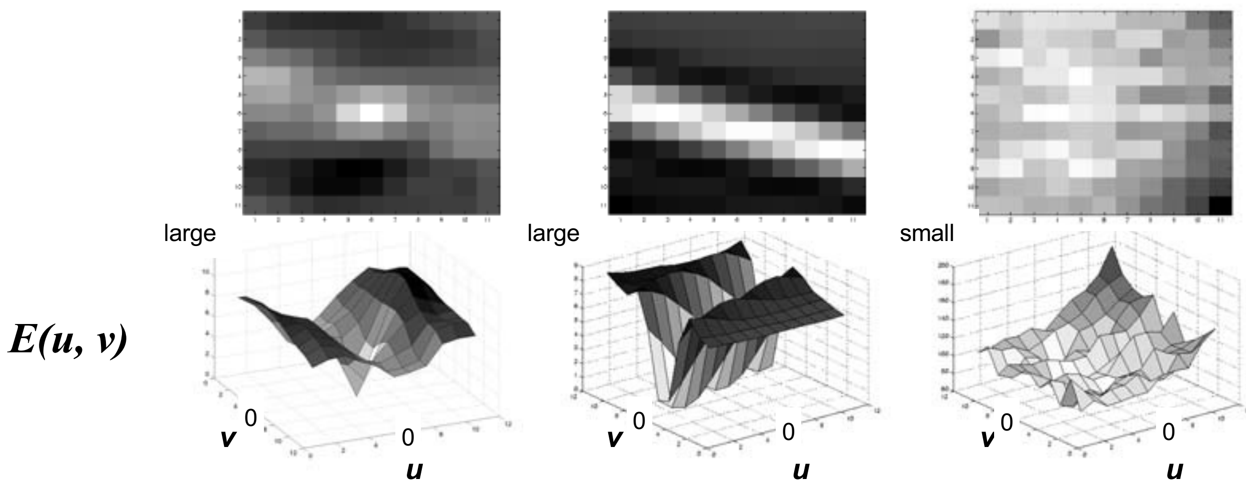
$$E(u,v) = \sum_{x,y} \Big[\underbrace{I(x+u, y+v)}_{\text{Shifted intensity}} - \underbrace{I(x,y)}_{\text{Intensity}} \Big]^2$$

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]:

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

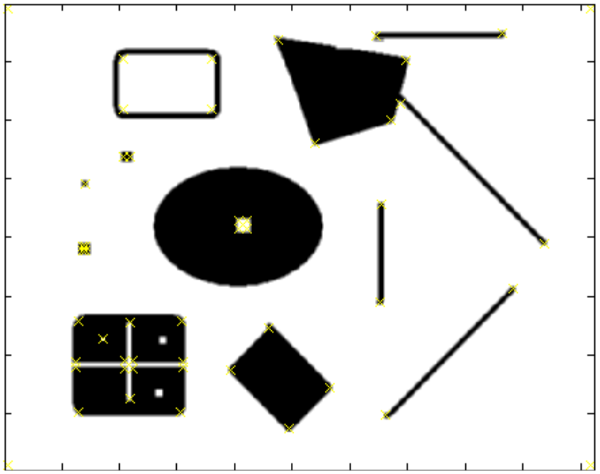


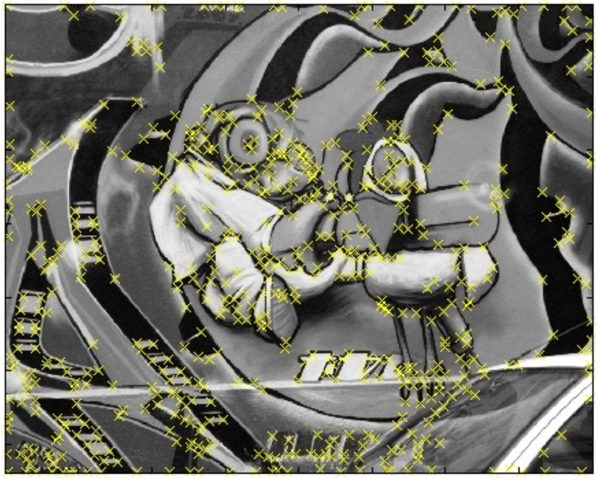
Adapted from D. Frolova, D. Simakov

Example of Harris Application

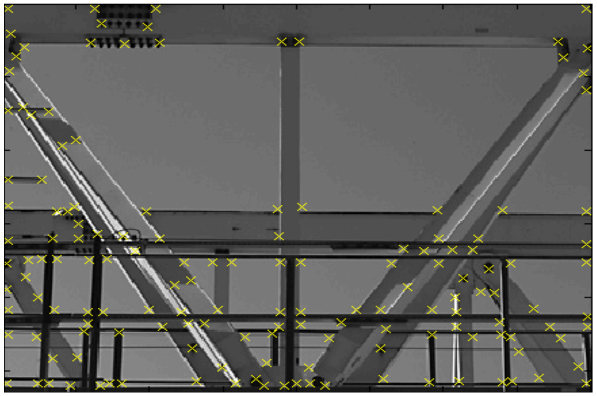
K. Grauman

More Harris Responses





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

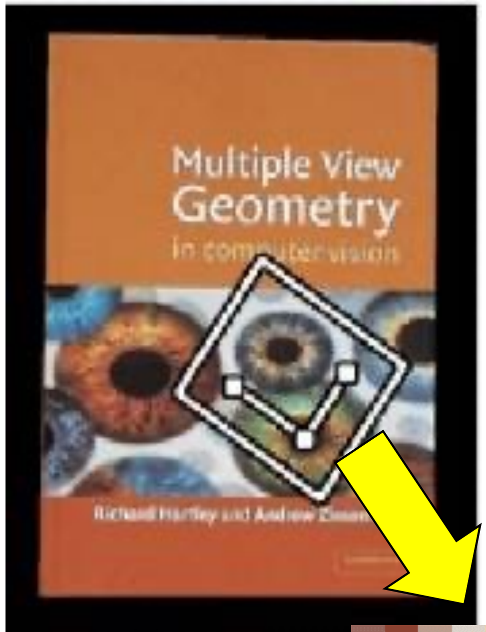


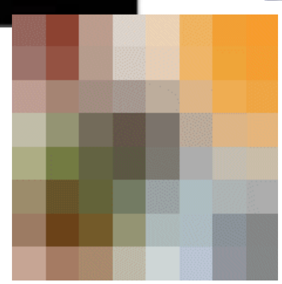
D. Hoiem

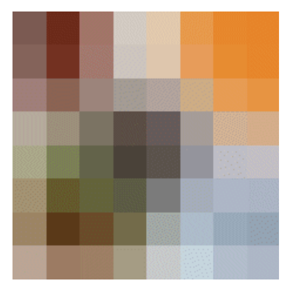
Plan for this lecture

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

Geometric transformations





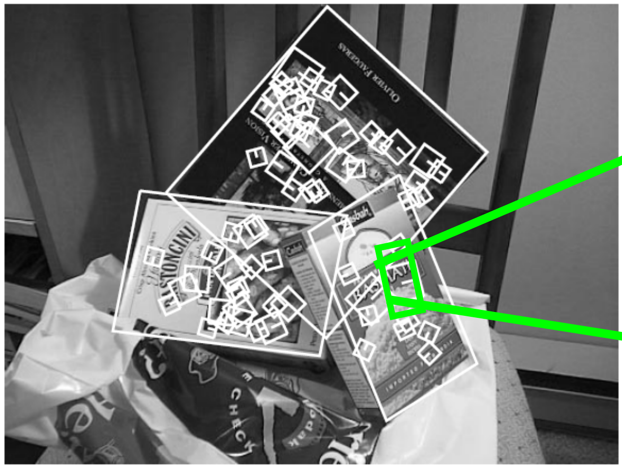


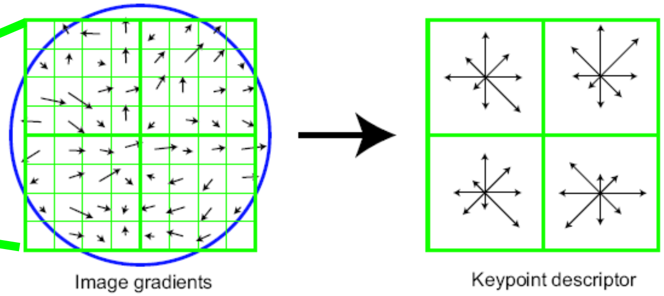
e.g. scale,
translation
, rotation

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

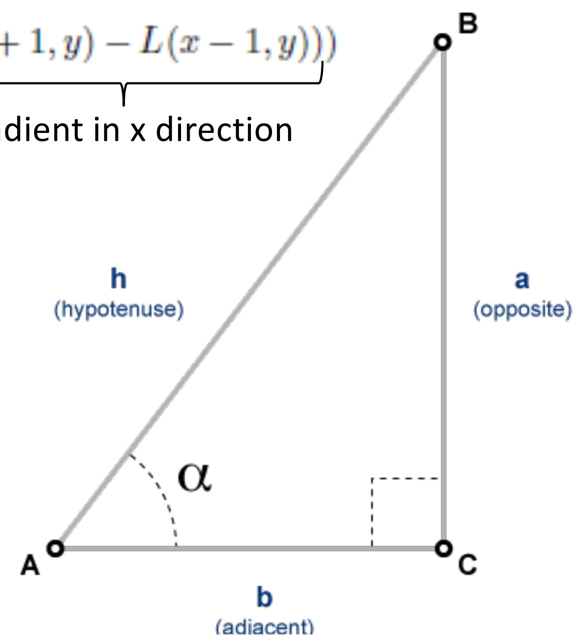
Computing gradients

L = the image intensity

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

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

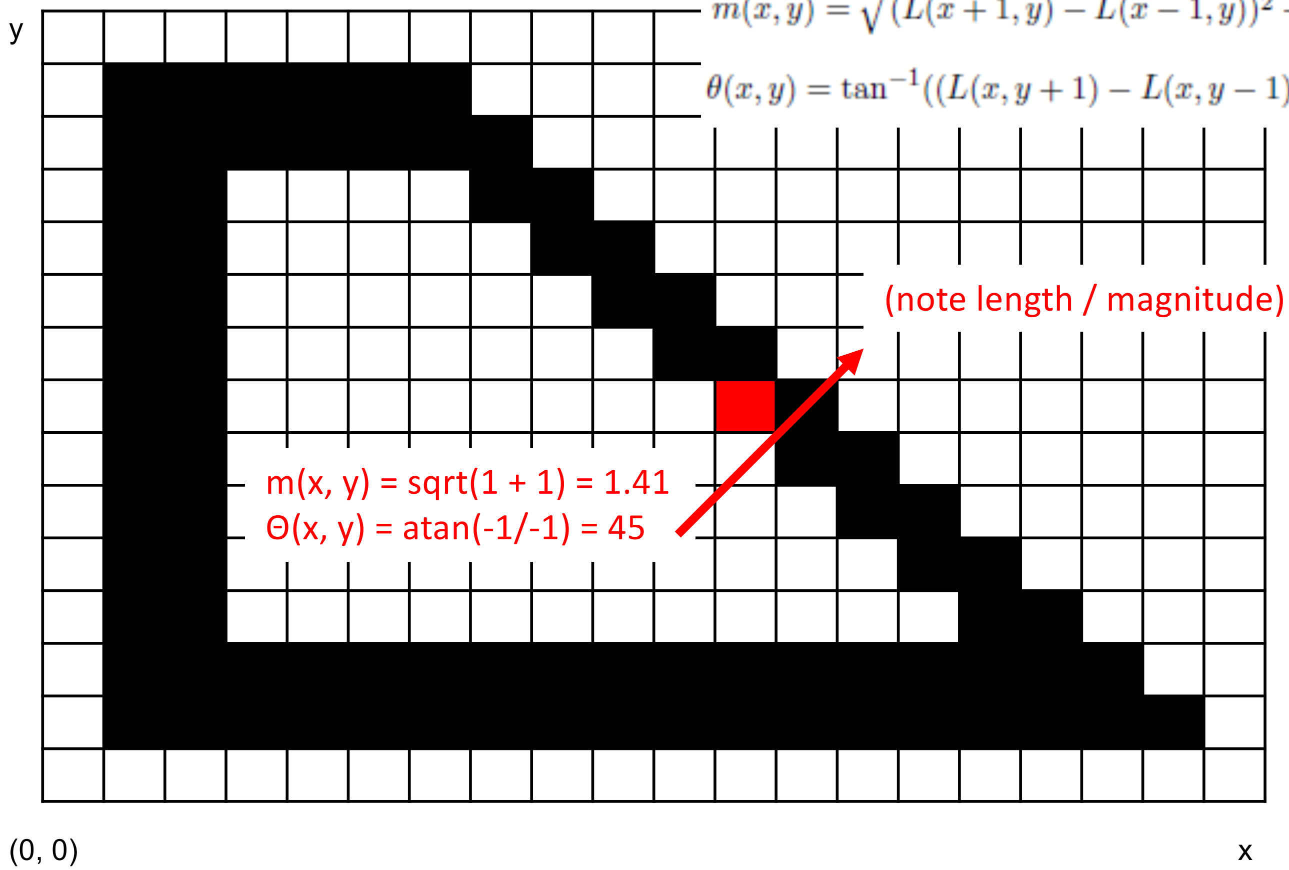
- $\tan(\alpha) = \dfrac{\textit{opposite side}}{\textit{adjacent side}}$



Gradients

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

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

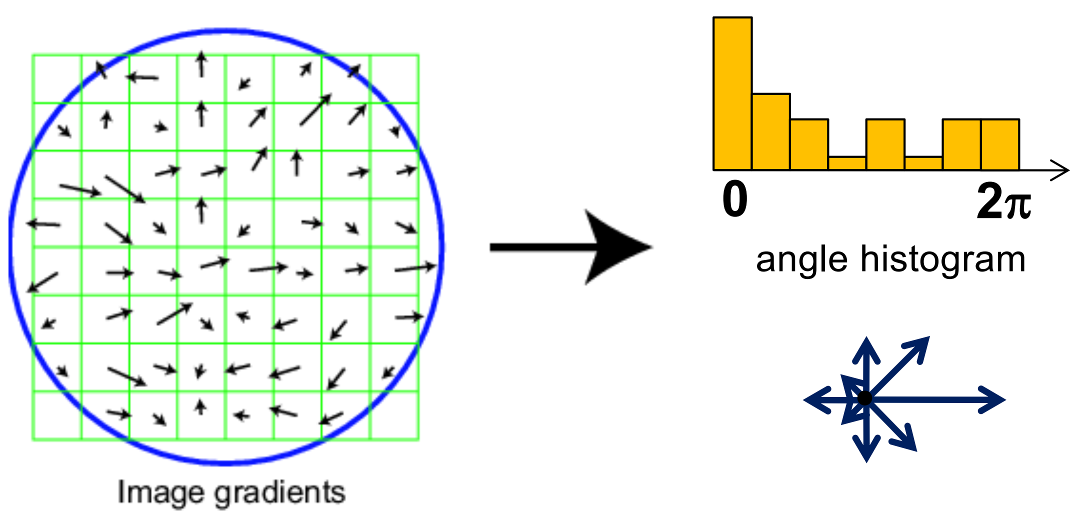


0: Black
1: White

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!

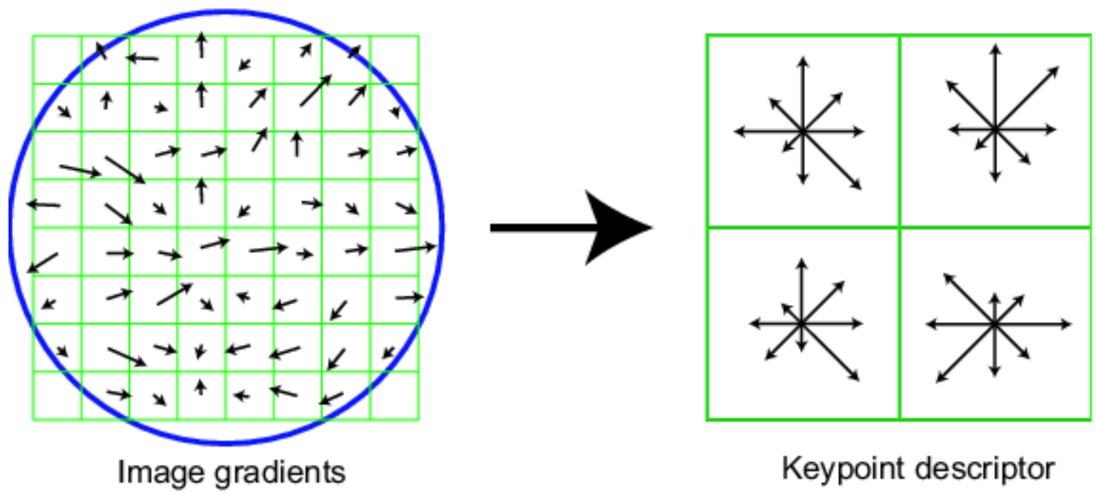


Adapted from L. Zitnick, D. Lowe

Scale Invariant Feature Transform

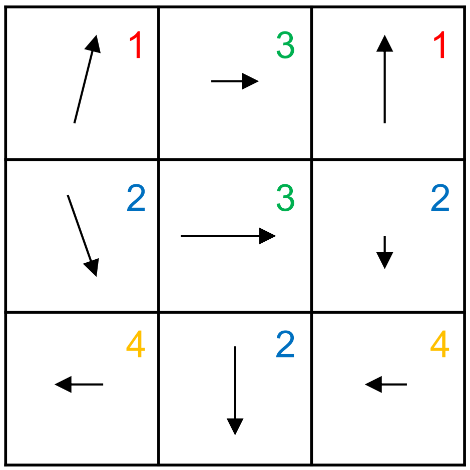
Full version

- 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

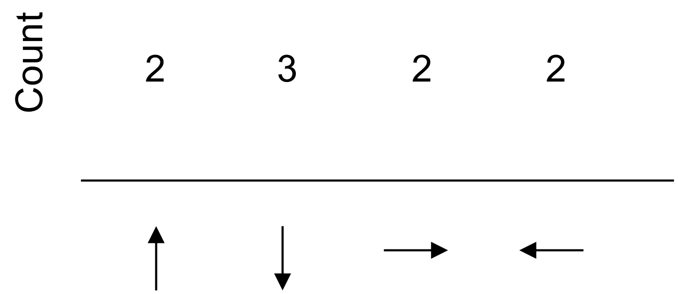


Adapted from L. Zitnick, D. Lowe

Scale Invariant Feature Transform



Uniform weight (ignore magnitude)

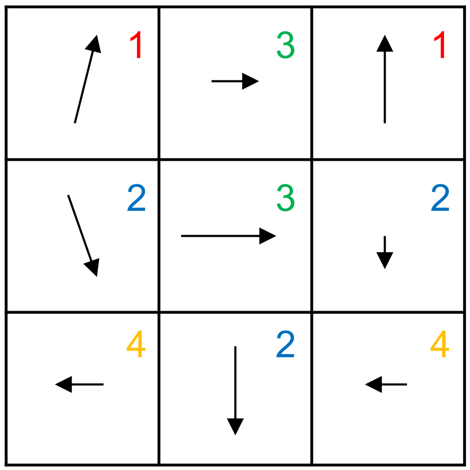


Type = 1 2 3 4

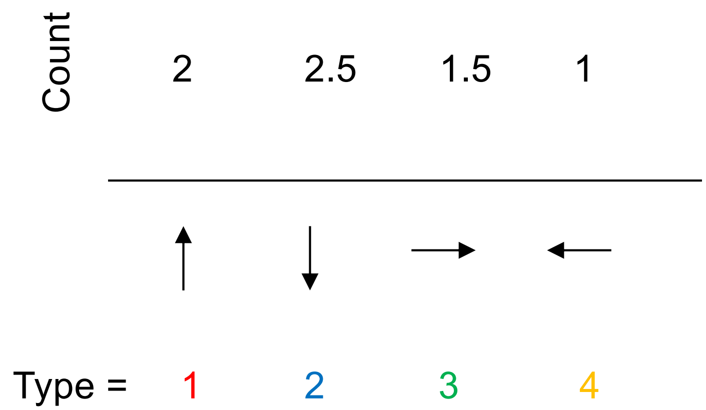
Gradients

Histogram of gradients

Scale Invariant Feature Transform



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



Gradients

Histogram of gradients

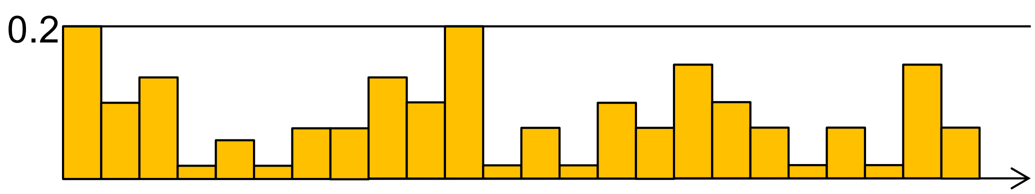
Adriana Kovashka

Scale Invariant Feature Transform

Full version

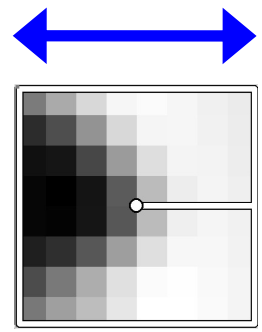
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Quantize the gradient orientations i.e. snap each gradient to one of 8 angles
- Each gradient contributes not just 1, but magnitude(gradient) to the histogram, i.e. stronger gradients contribute more
- 16 cells * 8 orientations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

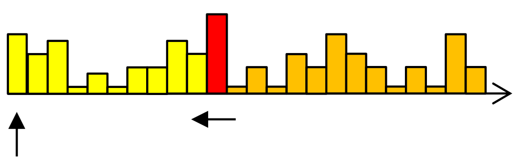
$$\boxed{\sum_i d_i = 1 \quad \text{such that: } d_i < 0.2}$$

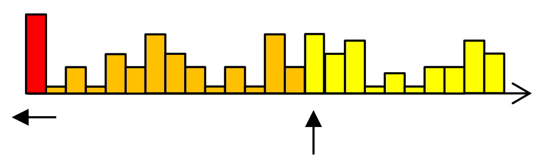


Adapted from L. Zitnick, D. Lowe

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

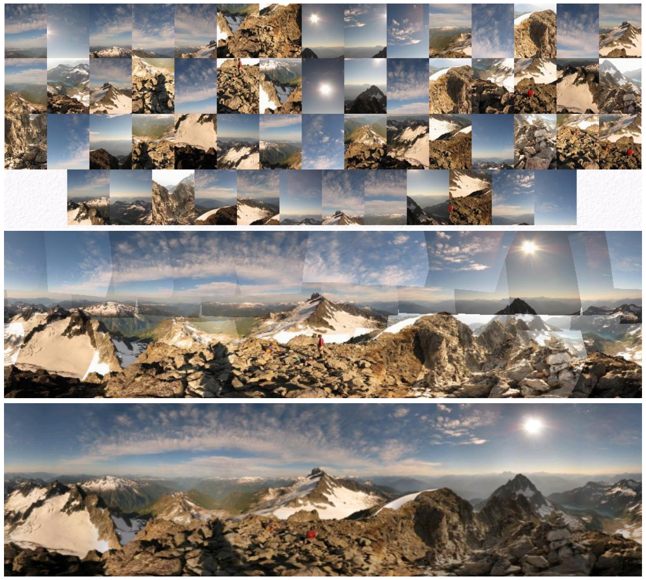
Adapted from S. Seitz

Examples of using SIFT

Adriana Kovashka

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

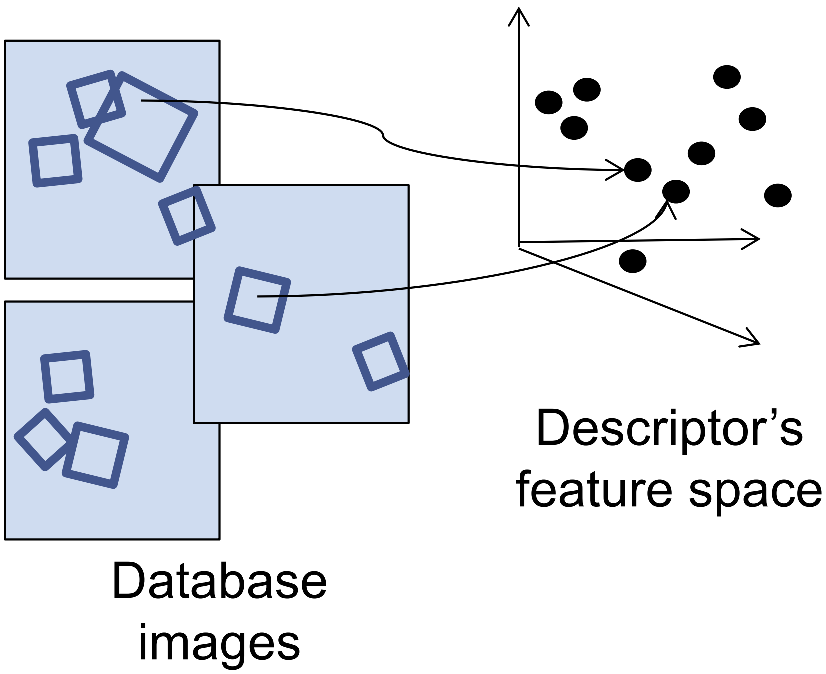
Adapted from K. Grauman and L. Lazebnik

Plan for this lecture

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

Matching Local Features Setup

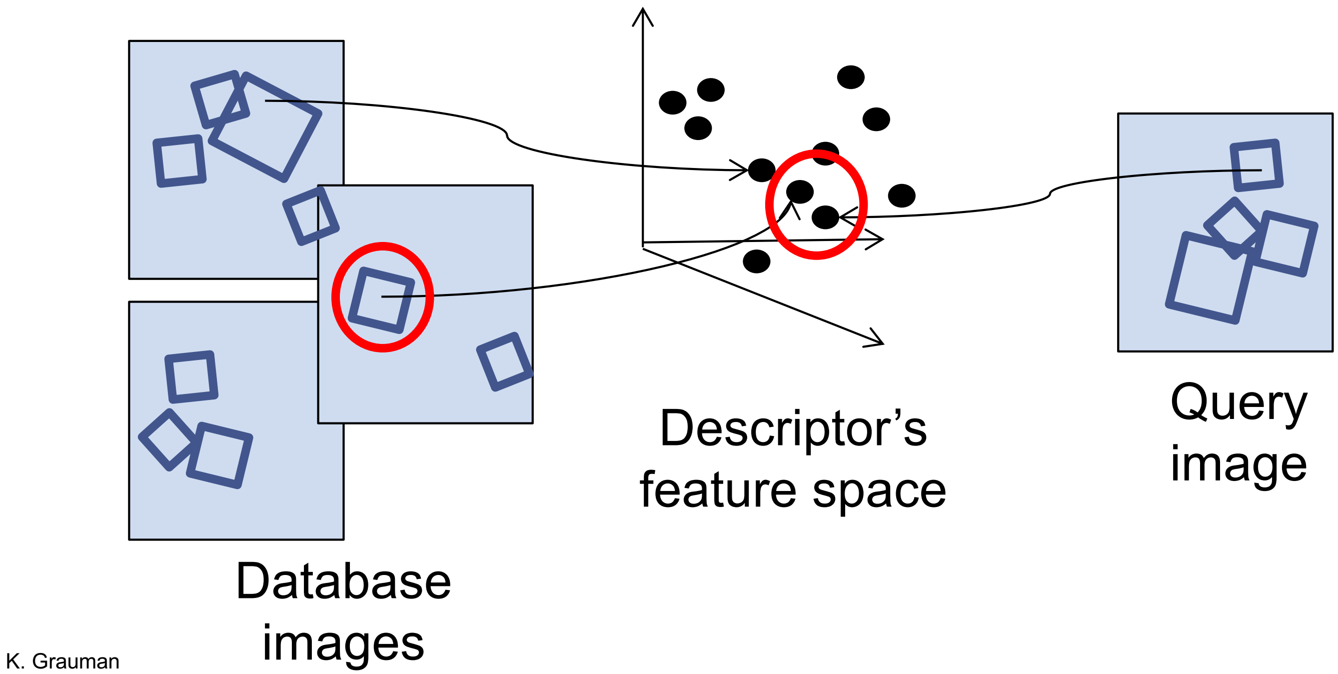
- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)



K. Grauman

Matching Local Features Setup

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



Indexing local features

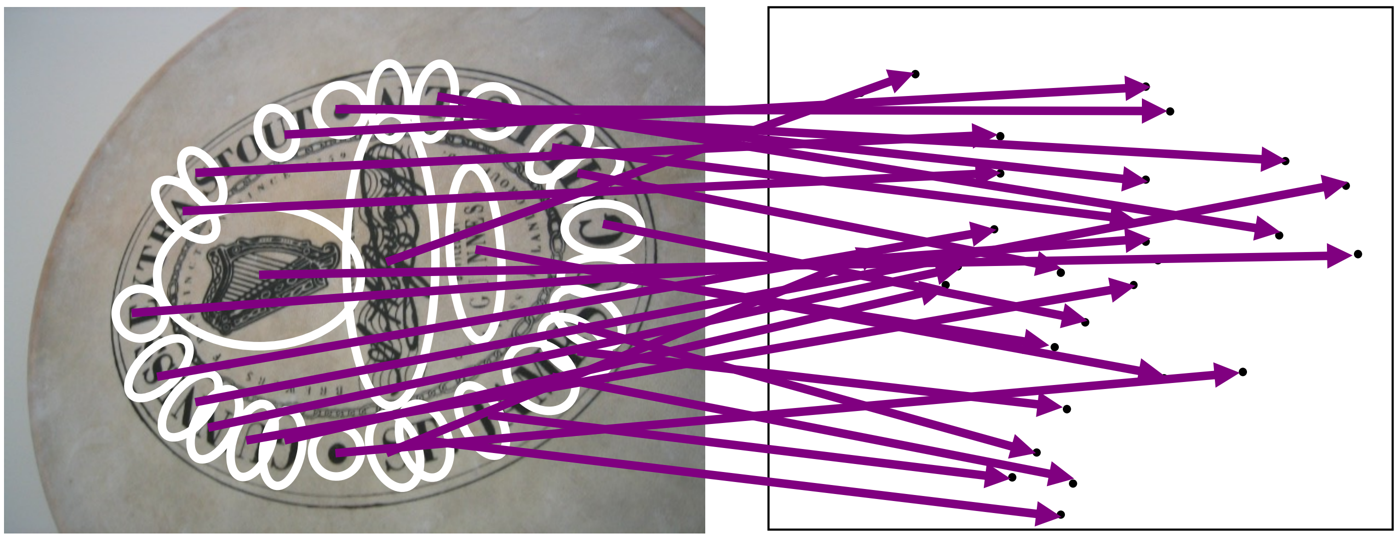

Index

- 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

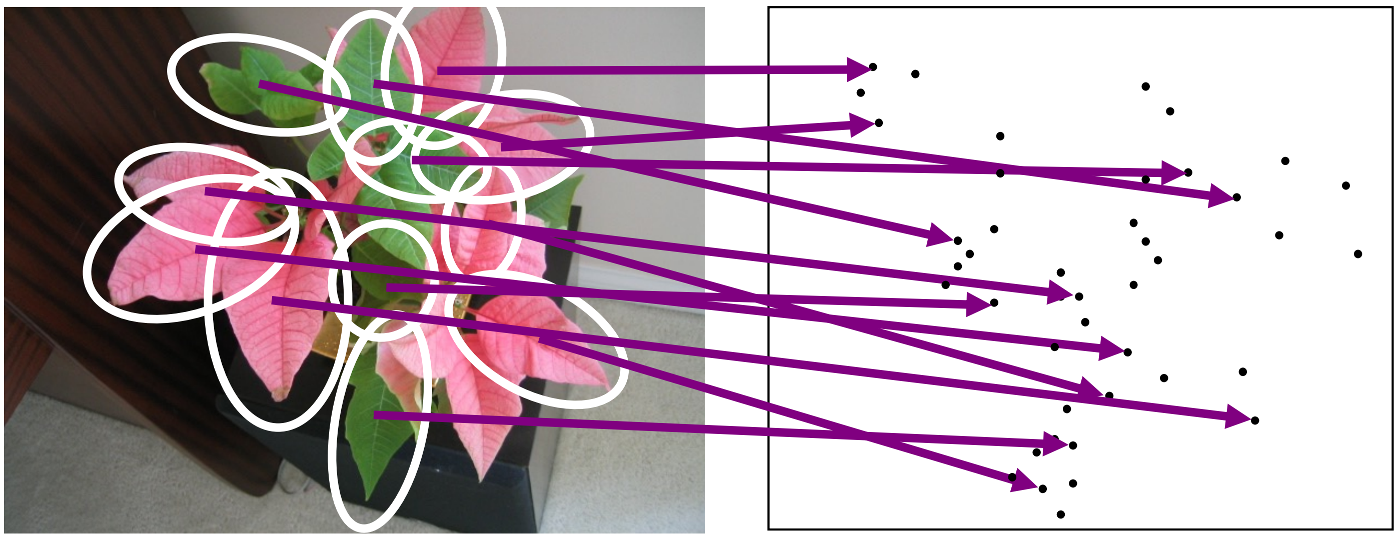
- Extract some local features from a number of images …



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

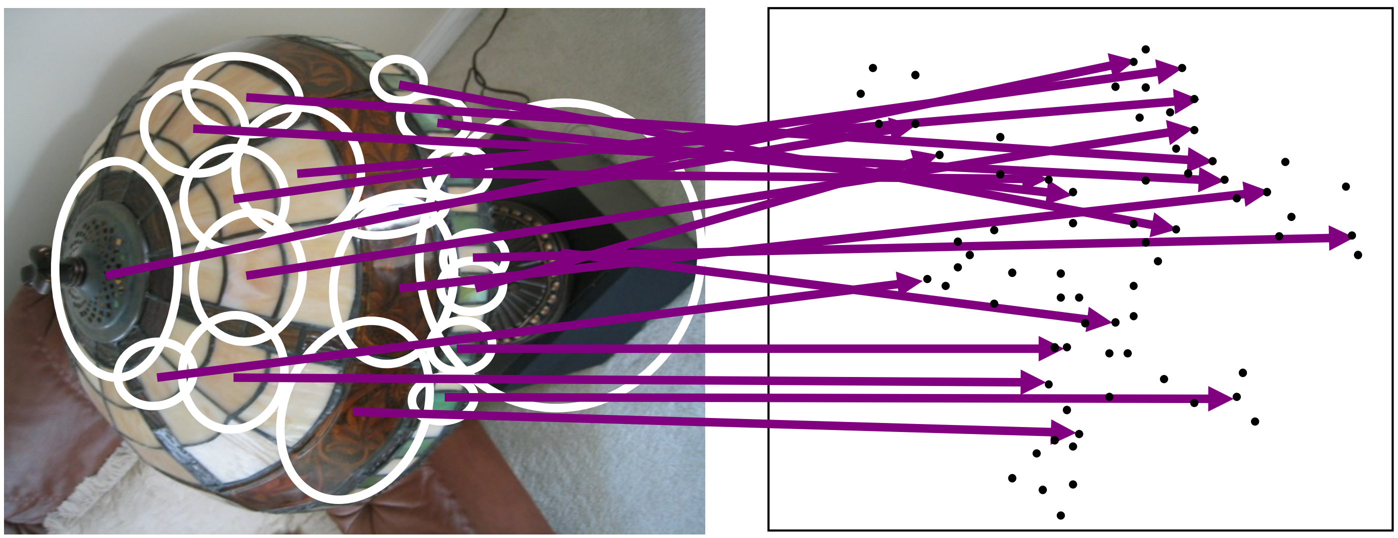
D. Nister, CVPR 2006

Visual Words: main idea



D. Nister, CVPR 2006

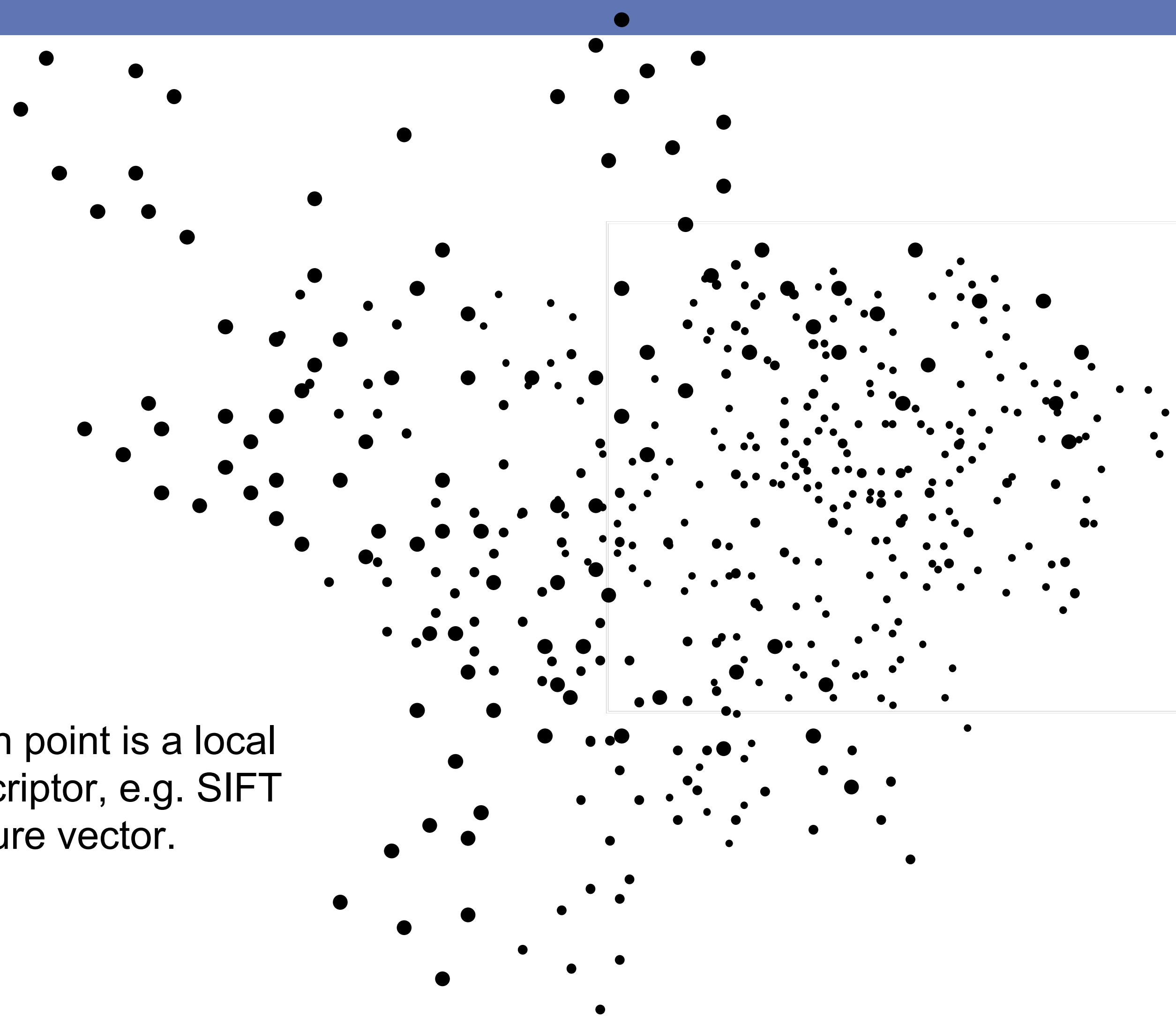
Visual Words: main idea



D. Nister, CVPR 2006

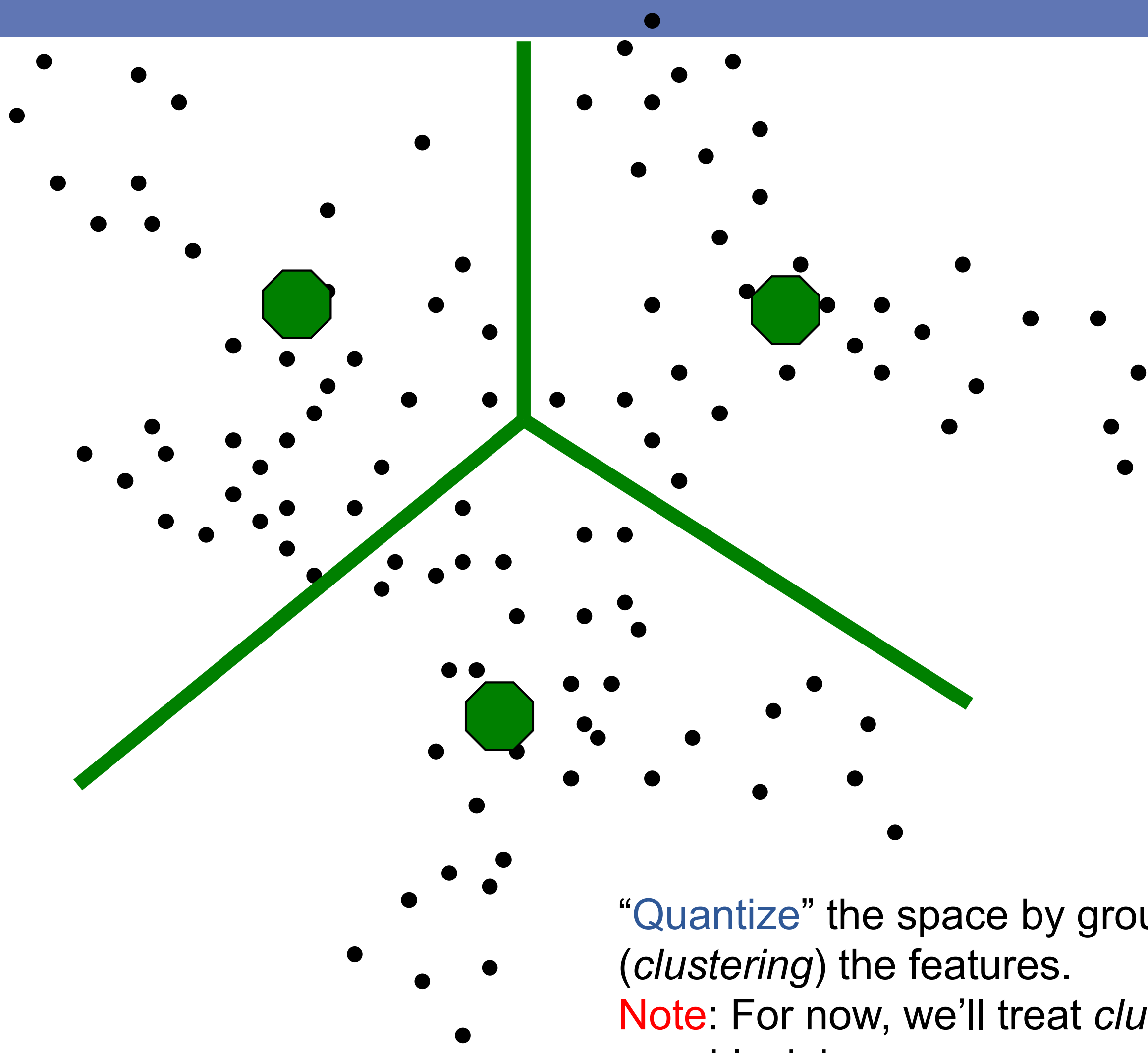
Visual Words: main idea

D. Nister, CVPR 2006



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

D. Nister, CVPR 2006

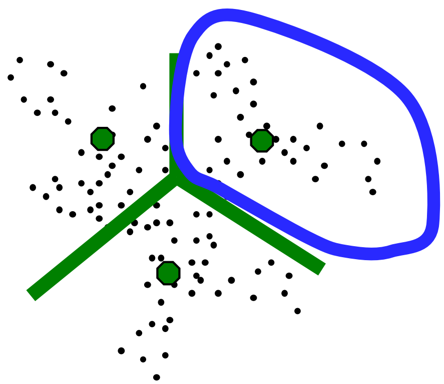


"Quantize" the space by grouping (*clustering*) the features.
Note: For now, we'll treat *clustering* as a black box.

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”



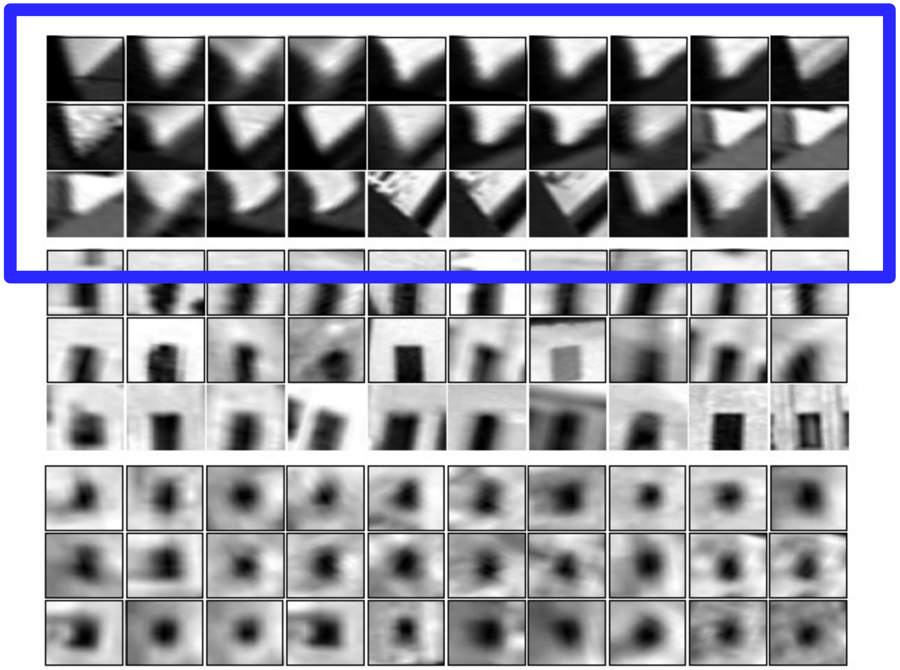
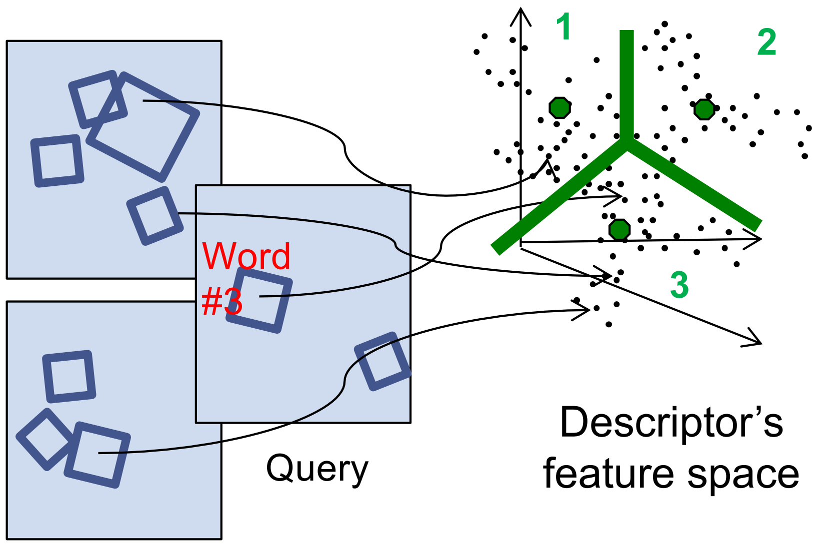


Figure from Sivic & Zisserman, ICCV 2003

Adapted from K. Grauman

Visual Words for Indexing

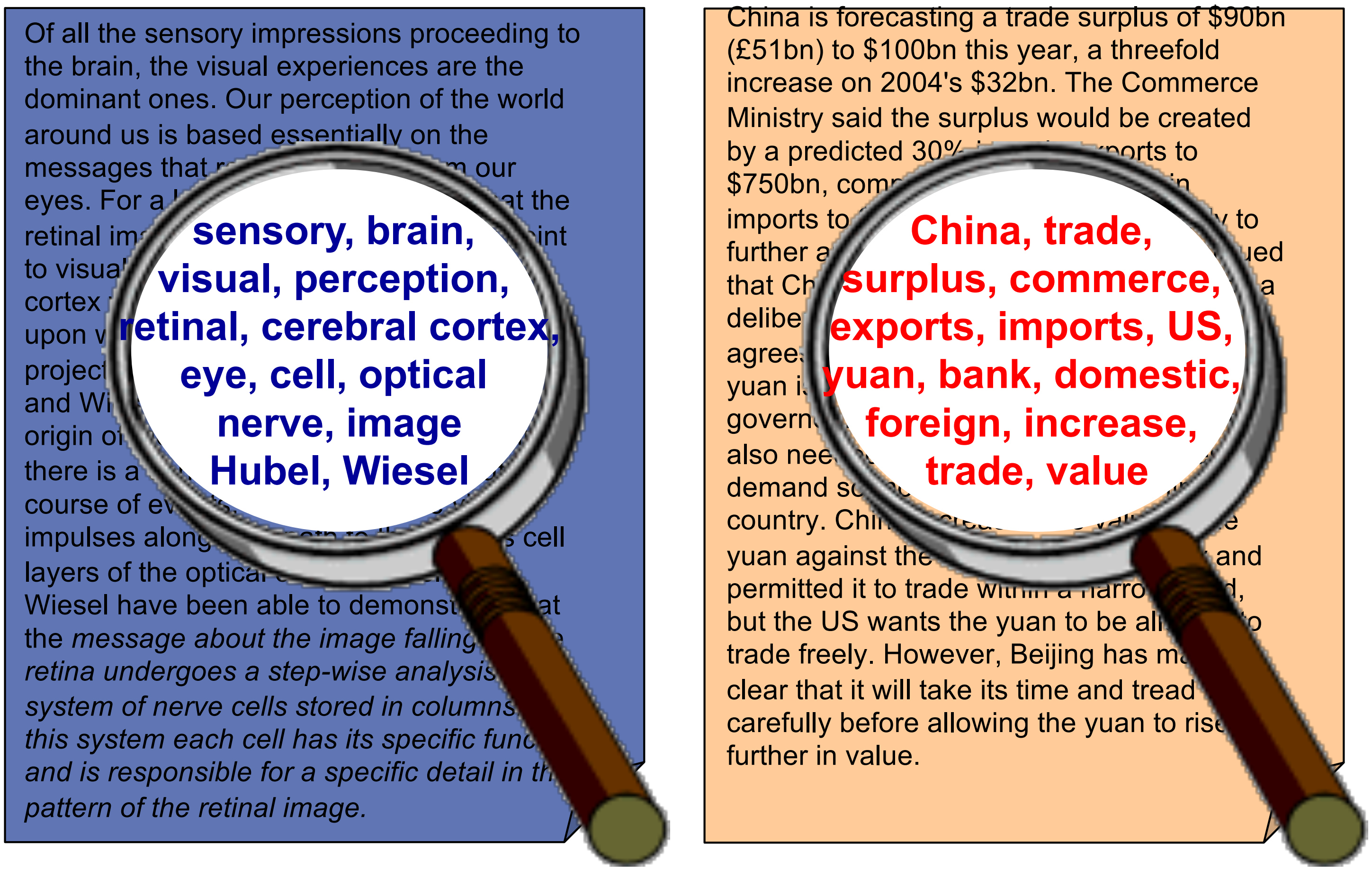
- Map high-dimensional descriptors to tokens/words by quantizing the feature space



- 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

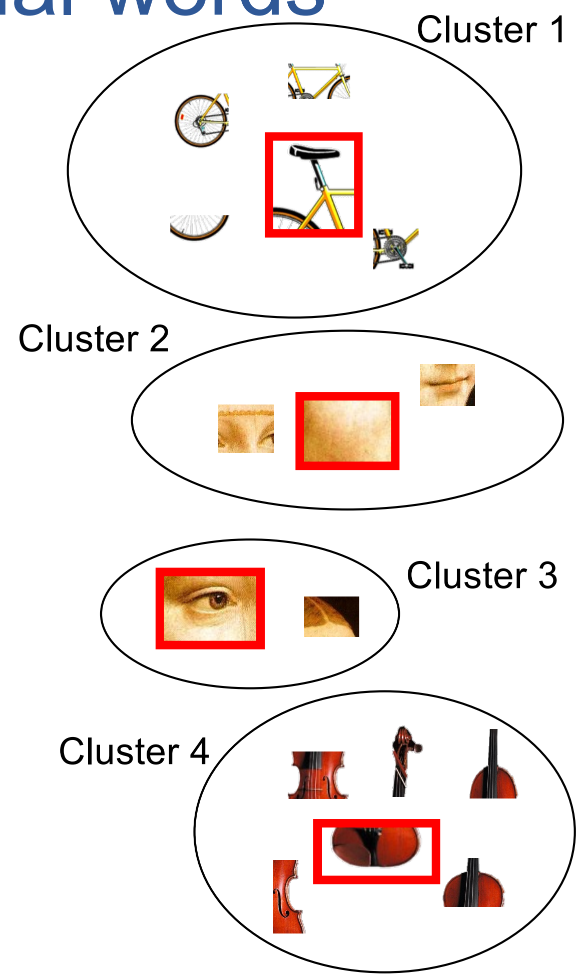
Adapted from K. Grauman

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:

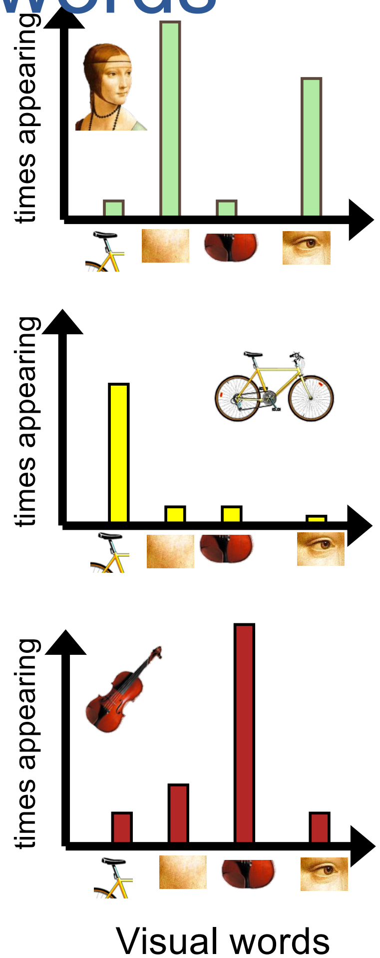
Adapted from K. Grauman

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

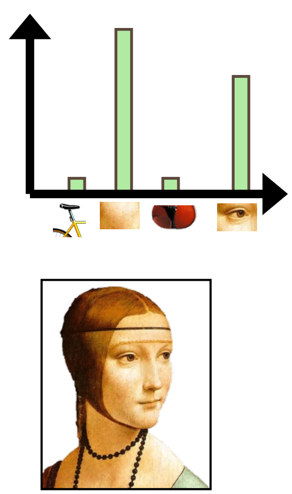
Feature patches:

K. Grauman

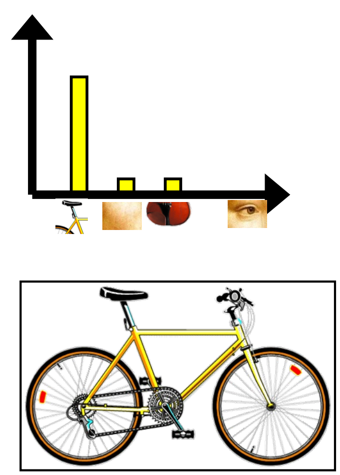


Comparing bags of words

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



$\vec{d}_j$



$\vec{q}$

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

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

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):

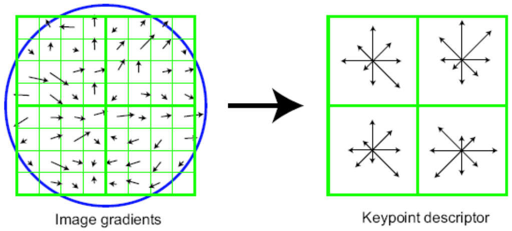
1. Extract words in query (extract features and map each to closest cluster center)
2. Use inverted file index to find database images relevant to query
3. Rank database images by comparing word counts of query and database image

Adapted from K. Grauman

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



Adapted from D. Hoiem, K. Grauman