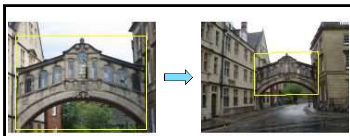
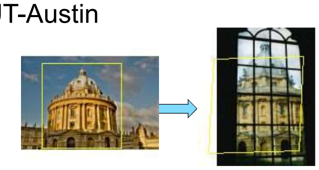


Recognizing object instances

Kristen Grauman
UT-Austin



Announcements

- Assignment 1 is out, due Fri Sept 22
- Presentation assignments - up this week
- Reminder – no laptops, phones, etc. in class please

Plan for today

- 1. Basics in feature extraction: filtering
- 2. Invariant local features
- 3. Recognizing object instances

Basics in feature extraction

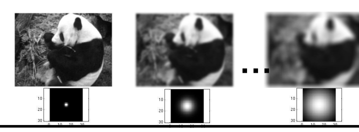
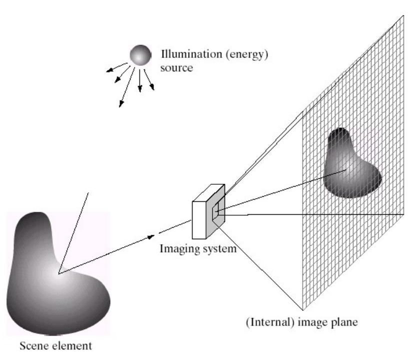
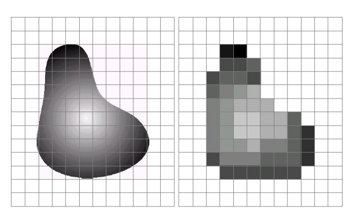


Image Formation



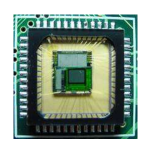
Slide credit: Derek Hoiem

Digital images



a b

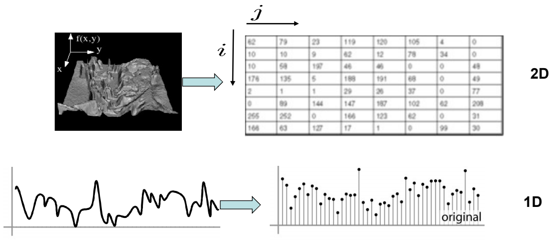
FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.



Slide credit: Derek Hoiem

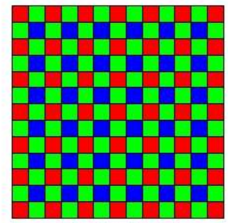
Digital images

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.



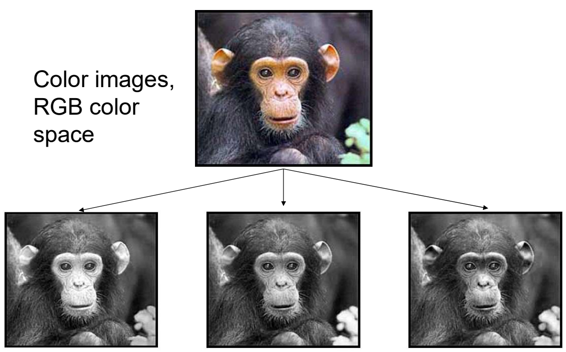
Adapted from S. Seitz

Digital color images



Digital color images

Color images, RGB color space



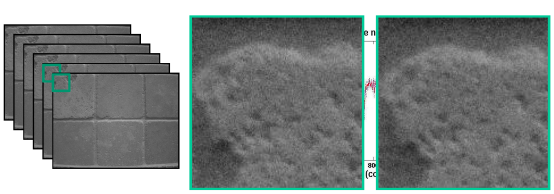
Kristen Grauman

Main idea: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a "filter" or mask saying how to combine values from neighbors.
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Adapted from Derek Hoiem

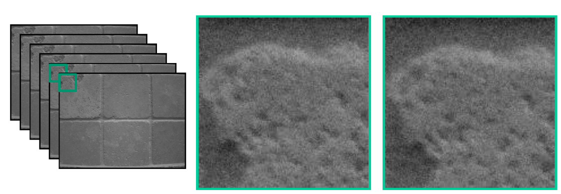
Motivation: noise reduction



- Even multiple images of the **same static scene** will not be identical.

Kristen Grauman

Motivation: noise reduction



- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- **What if there's only one image?**

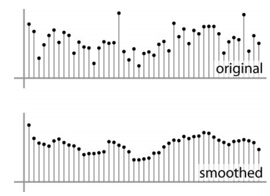
Kristen Grauman

First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

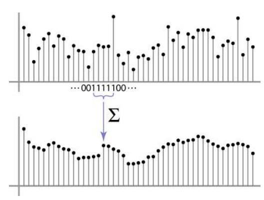


Source: S. Marschner

Weighted Moving Average

Can add weights to our moving average

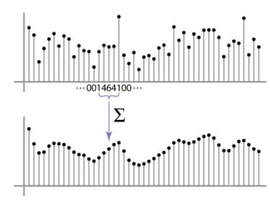
Weights [1, 1, 1, 1, 1] / 5



Source: S. Marschner

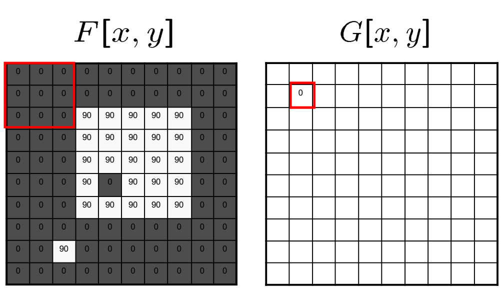
Weighted Moving Average

Non-uniform weights [1, 4, 6, 4, 1] / 16



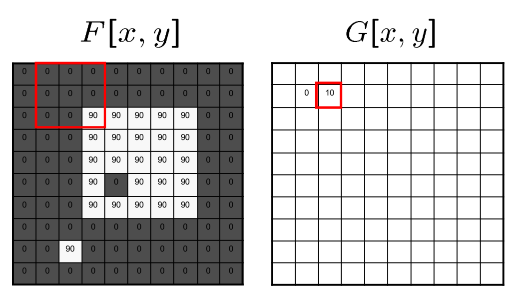
Source: S. Marschner

Moving Average In 2D



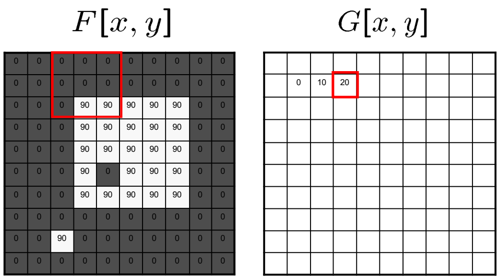
Source: S. Seitz

Moving Average In 2D



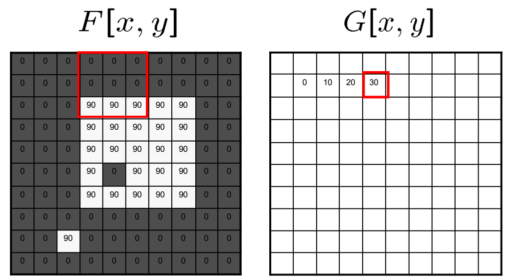
Source: S. Seitz

Moving Average In 2D



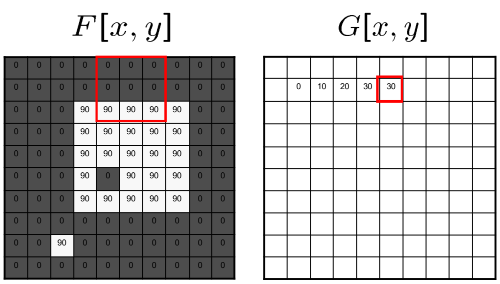
Source: S. Seitz

Moving Average In 2D



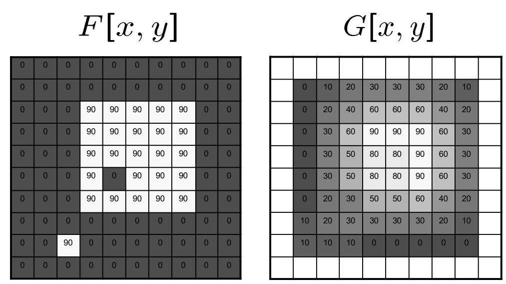
Source: S. Seitz

Moving Average In 2D



Source: S. Seitz

Moving Average In 2D



Source: S. Seitz

Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel } F[i,j]}$$

Now generalize to allow **different weights** depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called **cross-correlation**, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "**kernel**" or "**mask**" *H*[*u*,*v*] is the prescription for the weights in the linear combination.

Averaging filter

- What values belong in the kernel H for the moving average example?

$F[x, y]$ $\otimes$ $H[u, v]$ $G[x, y]$

$F[x, y]$:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$H[u, v]$: $\frac{1}{9}$

1	1	1
1	?	1
1	1	1

"box filter"

$G[x, y]$:

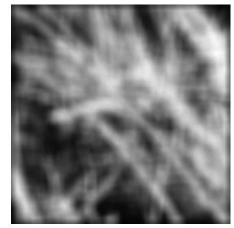
	0	10	20	30	30				

$$G = H \otimes F$$

Smoothing by averaging

depicts box filter:
white = high value, black = low value

original



filtered

What if the filter size was 5 x 5 instead of 3 x 3?

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

$F[x, y]$:

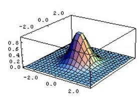
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90		90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$H[u, v]$: $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

This kernel is an approximation of a 2d Gaussian function:

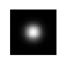
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

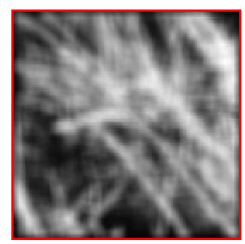


- Removes high-frequency components from the image ("low-pass filter").

Source: S. Seitz

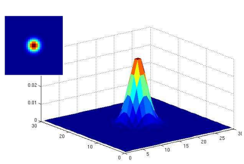
Smoothing with a Gaussian



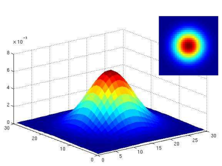


Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing



σ = 2 with 30 x 30 kernel

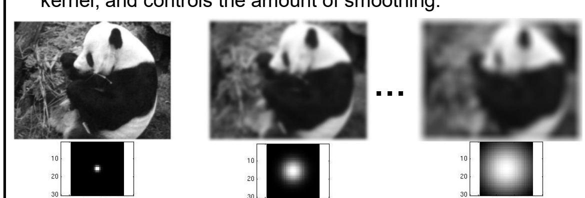


σ = 5 with 30 x 30 kernel

Kristen Grauman

Smoothing with a Gaussian

Parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.



```
for sigma=1:3:10
  h = fspecial('gaussian', fsize, sigma);
  out = imfilter(im, h);
  imshow(out);
  pause;
end
```

Kristen Grauman

Properties of smoothing filters

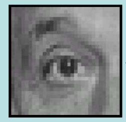
- Smoothing
  - Values positive
  - Sum to 1 → ____________________
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter

Kristen Grauman

Predict the outputs using correlation filtering

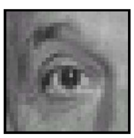
$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$



$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Practice with linear filters



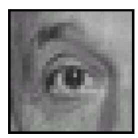
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

?

Source: D. Lowe

Practice with linear filters



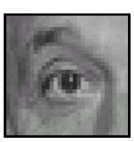
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Filtered (no change)

Source: D. Lowe

Practice with linear filters



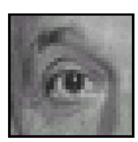
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

?

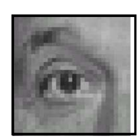
Source: D. Lowe

Practice with linear filters



Original

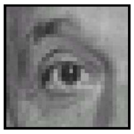
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$



Shifted left by 1 pixel with correlation

Source: D. Lowe

Practice with linear filters



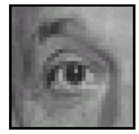
Original

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

?

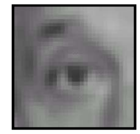
Source: D. Lowe

Practice with linear filters



Original

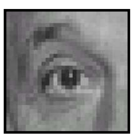
$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters



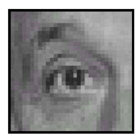
Original

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

?

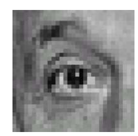
Source: D. Lowe

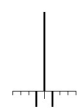
Practice with linear filters



Original

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

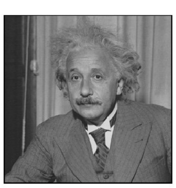




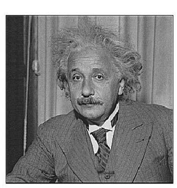
Sharpening filter:
accentuates differences with local average

Source: D. Lowe

Filtering examples: sharpening

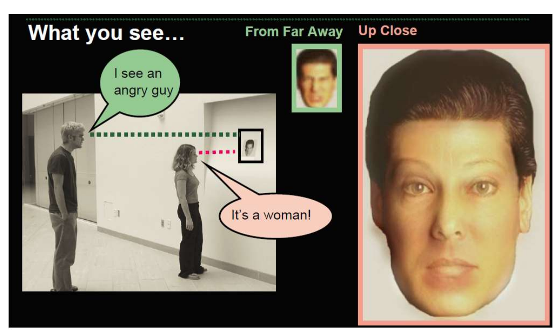


before



after

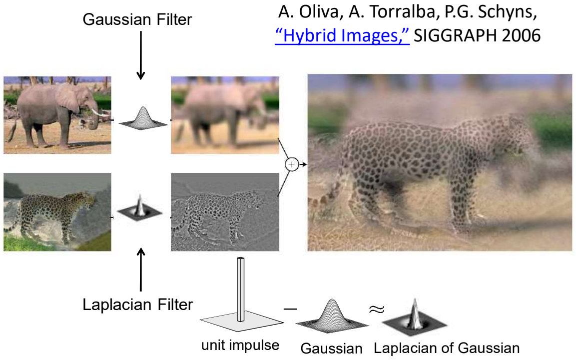
Filtering application: Hybrid Images

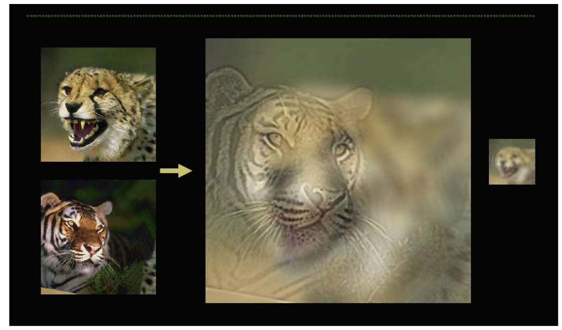


Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

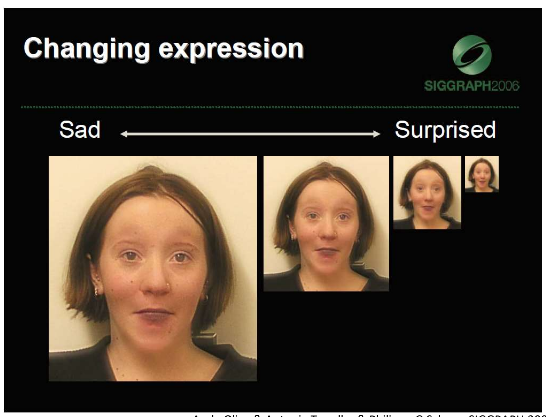
Application: Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006





Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

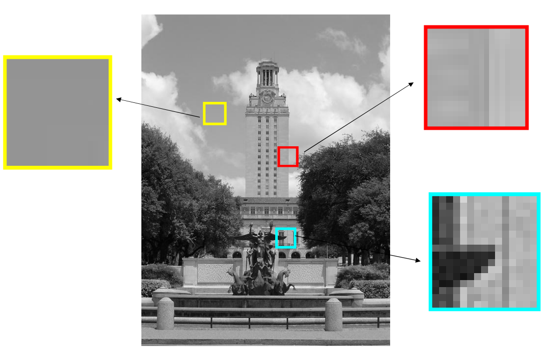


Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Main idea: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a "filter" or mask saying how to combine values from neighbors.
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

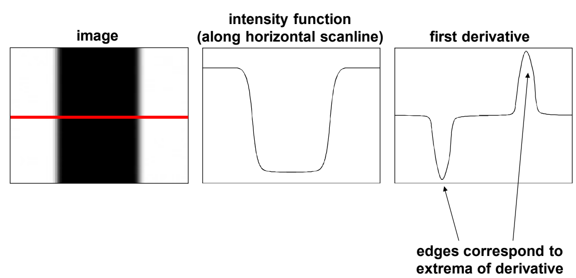
Why are gradients important?



Kristen Grauman

Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Source: L. Lazebnik

Derivatives with convolution

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

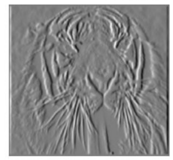
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

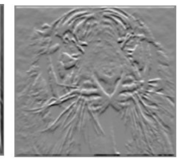
To implement above as convolution, what would be the associated filter?

Kristen Grauman

Partial derivatives of an image

$\frac{\partial f(x,y)}{\partial x}$





$\frac{\partial f(x,y)}{\partial y}$

-1	1

-1
1

? or

1
-1

Which shows changes with respect to x?

Kristen Grauman

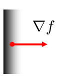
(showing filters for correlation)

Image gradient

The gradient of an image:

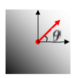
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

The **gradient direction** (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

Slide credit Steve Seitz

Mask properties

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Derivatives
  - ___________ signs used to get high response in regions of high contrast
  - Sum to ___ → no response in constant regions
  - High absolute value at points of high contrast

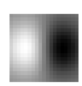
Kristen Grauman

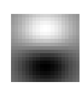
Main idea: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Template matching

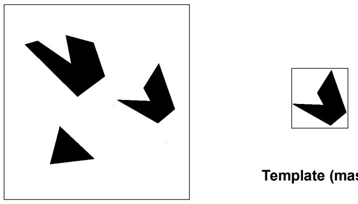
- Filters as **templates**:
  Note that filters look like the effects they are intended to find --- “matched filters”





- Use normalized cross-correlation score to find a given pattern (template) in the image.
- Normalization needed to control for relative brightnesses.

Template matching

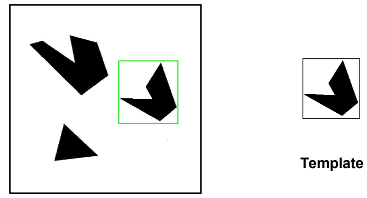


Scene

Template (mask)

A toy example

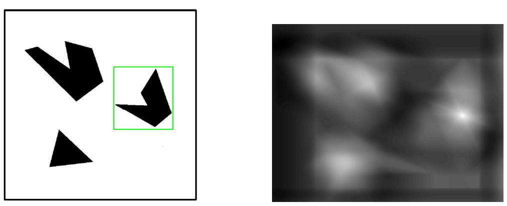
Template matching



Detected template

Template

Template matching



Detected template

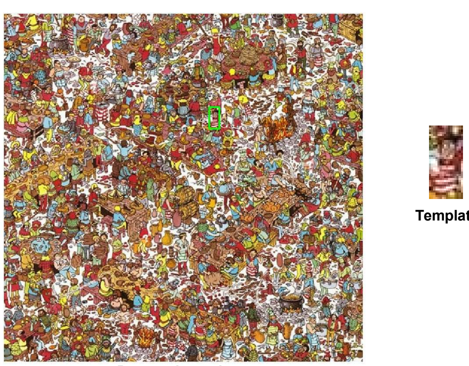
Correlation map

Where's Waldo?

Template

Scene

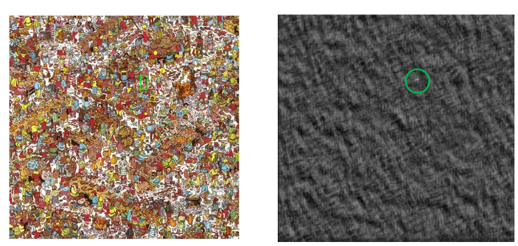
Where's Waldo?



Template

Detected template

Where's Waldo?



Detected template

Correlation map

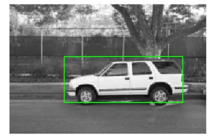
Template matching

Scene

Template

What if the template is not identical to some subimage in the scene?

Template matching



Detected template

Template

Match can be meaningful, if scale, orientation, and general appearance is right.

…but we can do better!...

Summary so far

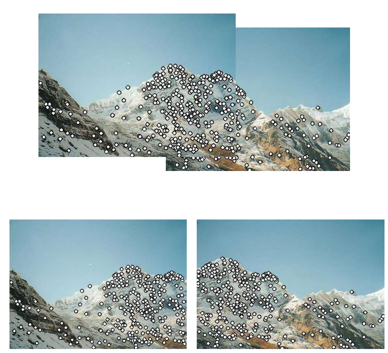
- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Plan for today

- 1. Basics in feature extraction: filtering
- **2. Invariant local features**
- 3. Specific object recognition methods

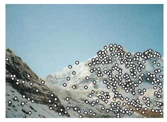
Local features: detection and description

Basic goal



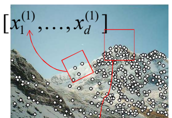
Local features: main components

1) Detection: Identify the interest points



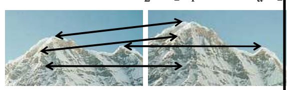
2) Description: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

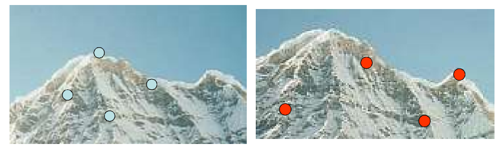
3) Matching: Determine correspondence between descriptors in two views



Kristen Grauman

Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.

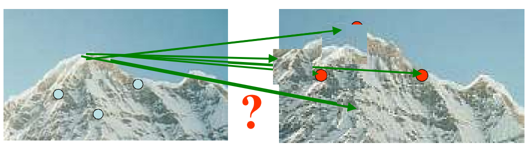


No chance to find true matches!

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

Goal: descriptor distinctiveness

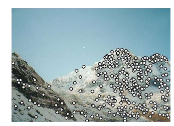
- We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

Local features: main components

1) Detection: Identify the interest points



2) Description: Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views

- What points would you choose?

Detecting corners

Detecting corners

Compute "cornerness" response at every pixel.

Detecting corners

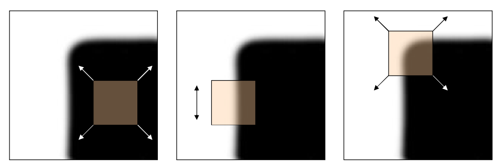
Detecting local invariant features

- Detection of interest points
  - Harris corner detection
  - Scale invariant blob detection: LoG

Corners as distinctive interest points

We should easily recognize the point 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

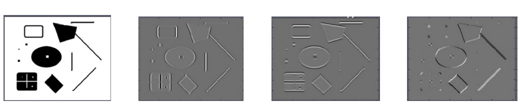
"corner": significant change in all directions

Slide credit: Alyosha Efros, Darya Frolova, Denis Simakov

Corners as distinctive interest points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

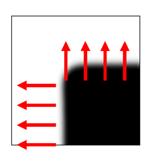
2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



Notation: $I_x \Leftrightarrow \frac{\partial I}{\partial x}$ $I_y \Leftrightarrow \frac{\partial I}{\partial y}$ $I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

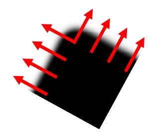
Look for locations where **both** λ's are large.

If either λ is close to 0, then this is **not** corner-like.

What if we have a corner that is not aligned with the image axes?

What does this matrix reveal?

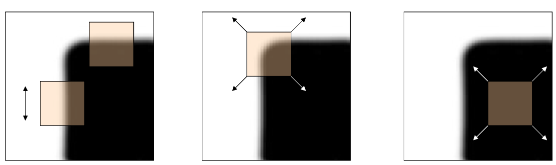
Since *M* is symmetric, we have $M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$



$$Mx_i = \lambda_i x_i$$

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

Corner response function



"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$

"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;

"flat" region
λ_1 and λ_2 are small;

Cornerness score
(other variants possible)

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

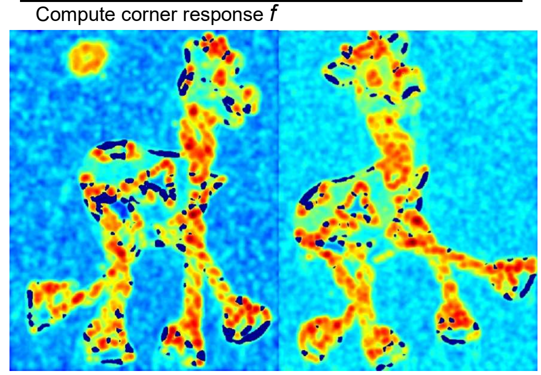
Harris corner detector

1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*f*> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Harris Detector: Steps

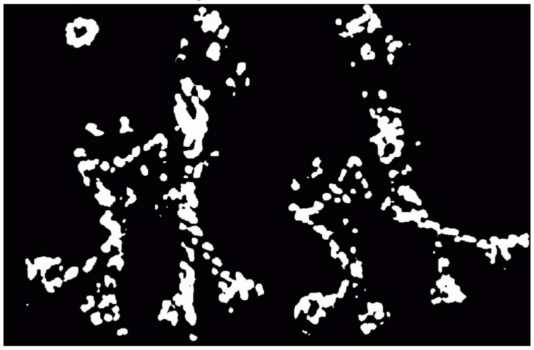
Harris Detector: Steps

Compute corner response ***f***



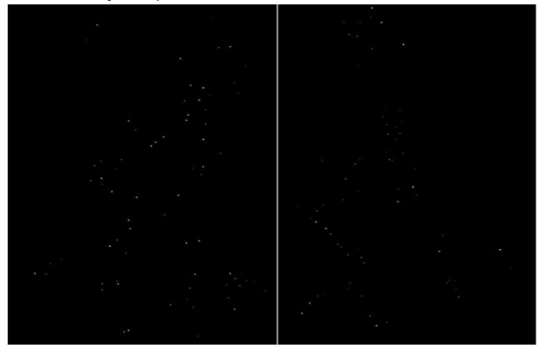
Harris Detector: Steps

Find points with large corner response: *f* > threshold

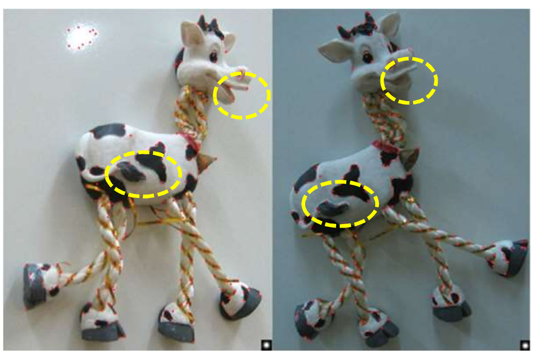


Harris Detector: Steps

Take only the points of local maxima of *f*



Harris Detector: Steps



Properties of the Harris corner detector

Rotation invariant? Yes

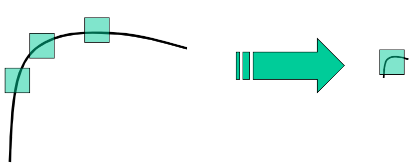
$$M = X\begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}X^T$$

Scale invariant?

Properties of the Harris corner detector

Rotation invariant? Yes

Scale invariant? No



All points will be classified as edges

Corner !

Scale invariant interest points

How can we independently select interest points in each image, such that the detections are repeatable across different scales?

Automatic Scale Selection

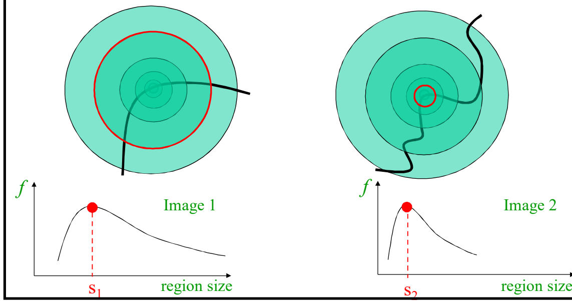
How to find corresponding patch sizes, with only one image in hand?

K. Grauman, B. Leibe

Automatic scale selection

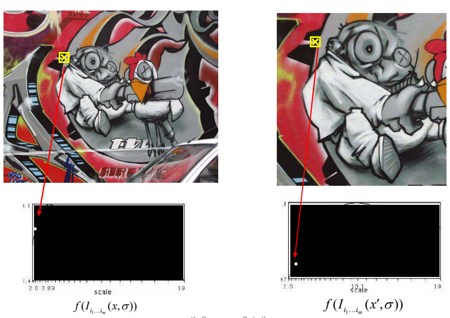
Intuition:

- Find scale that gives local maxima of some function f in both position and scale.



Automatic Scale Selection

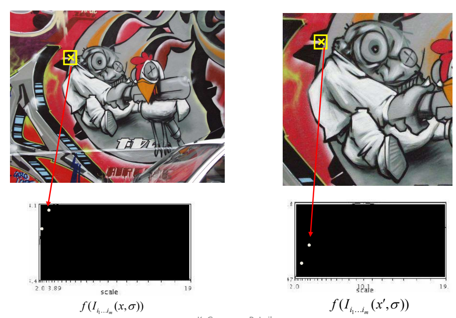
- Function responses for increasing scale (scale signature)



K. Grauman, B. Leibe

Automatic Scale Selection

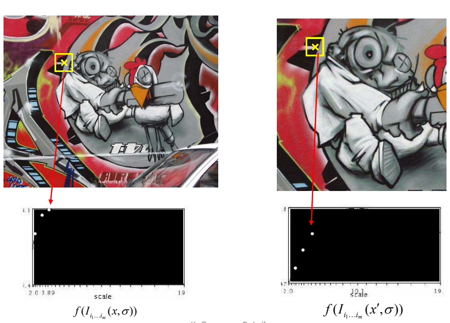
- Function responses for increasing scale (scale signature)



K. Grauman, B. Leibe

Automatic Scale Selection

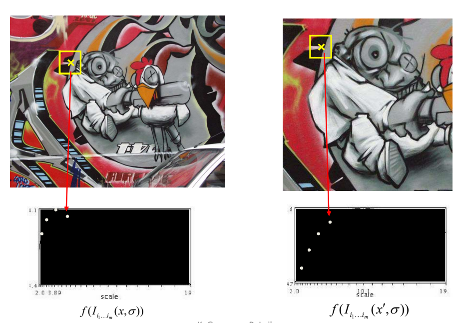
- Function responses for increasing scale (scale signature)



K. Grauman, B. Leibe

Automatic Scale Selection

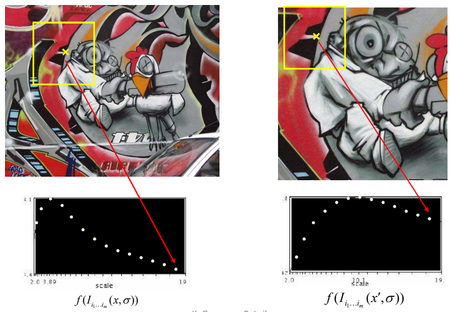
- Function responses for increasing scale (scale signature)



K. Grauman, B. Leibe

Automatic Scale Selection

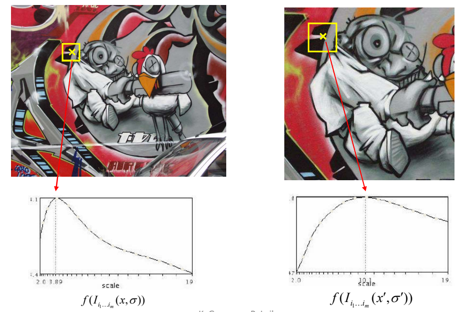
- Function responses for increasing scale (scale signature)



K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

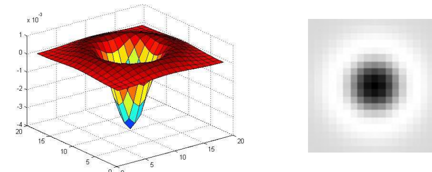


K. Grauman, B. Leibe

What can be the "signature" function?

Blob detection in 2D

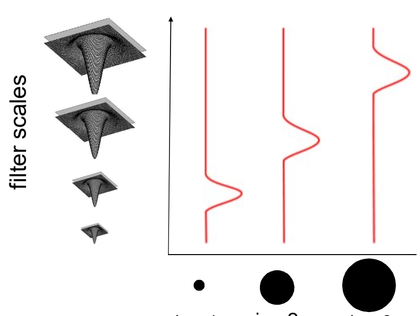
Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D



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

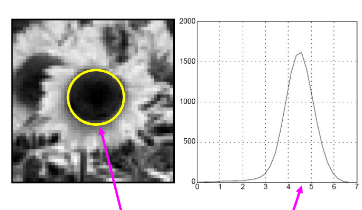
Blob detection in 2D: scale selection

Laplacian-of-Gaussian = "blob" detector $\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$

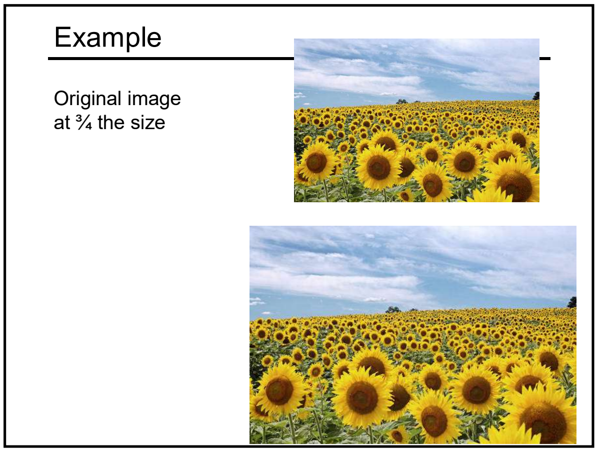


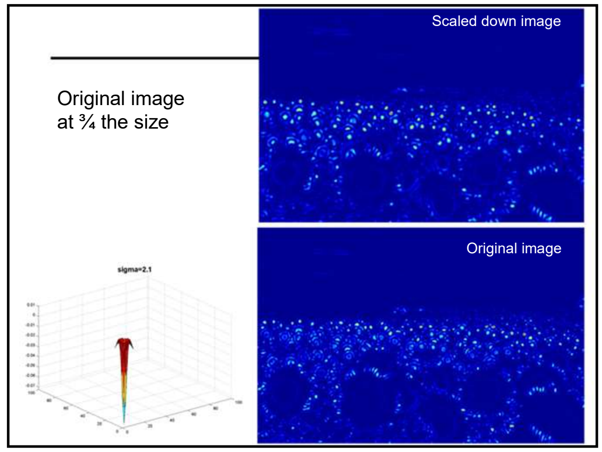
Blob detection in 2D

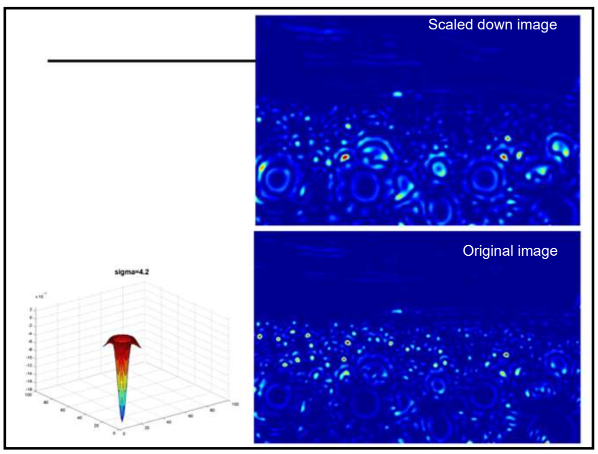
We define the *characteristic scale* as the scale that produces peak of Laplacian response

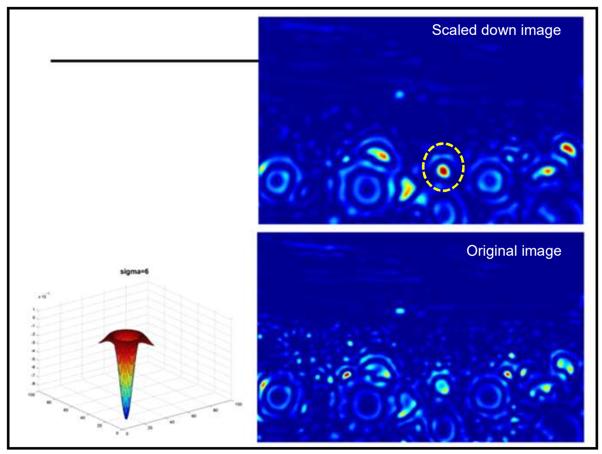


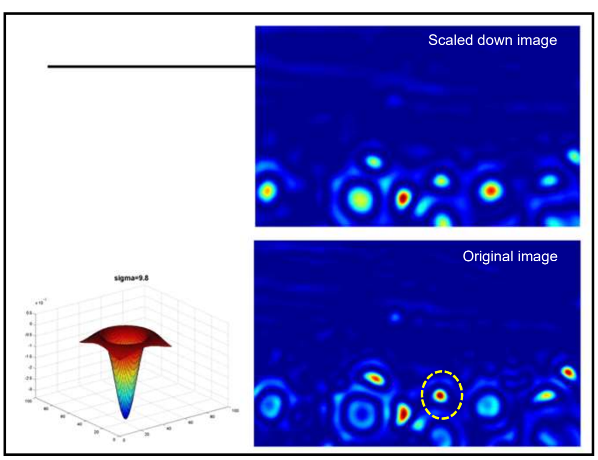
Slide credit: Lana Lazebnik

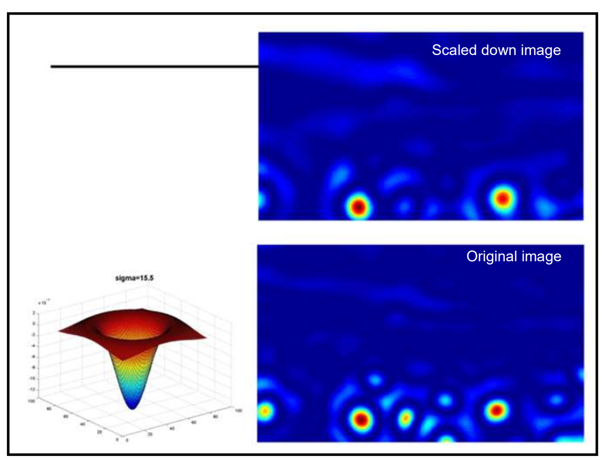

Example
Original image
at ¾ the size

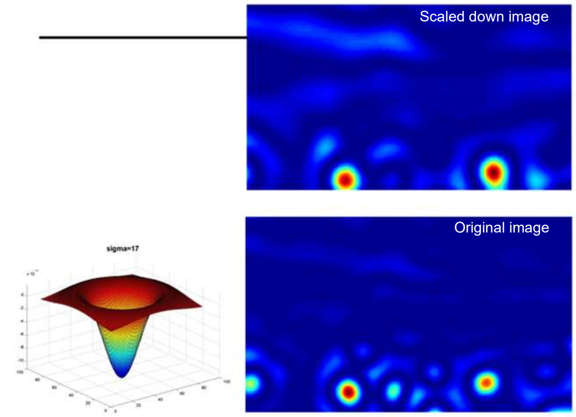

Original image
at ¾ the size
Scaled down image
Original image
sigma=2.1


Scaled down image
Original image
sigma=4.2


Scaled down image
Original image
sigma=6

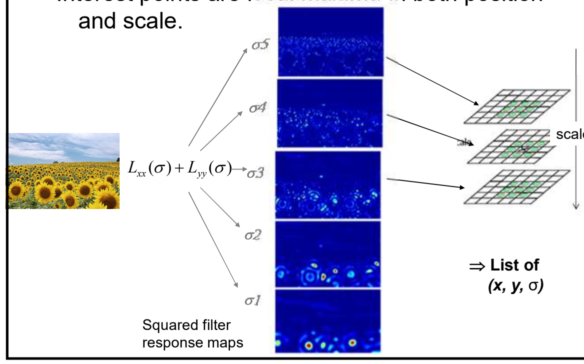

Scaled down image
Original image
sigma=9.8


Scaled down image
Original image
sigma=15.5

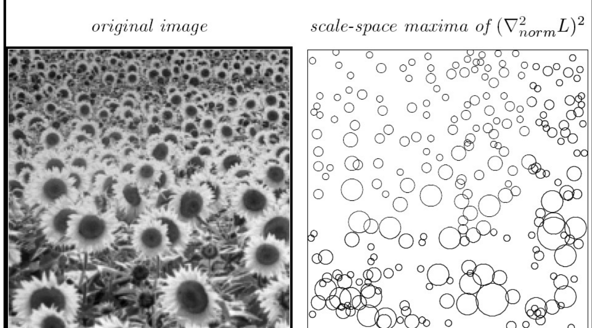


Scale invariant interest points

Interest points are local maxima in both position and scale.



Scale-space blob detector: Example



T. Lindeberg. Feature detection with automatic scale selection. IJCV 1998.

Scale-space blob detector: Example

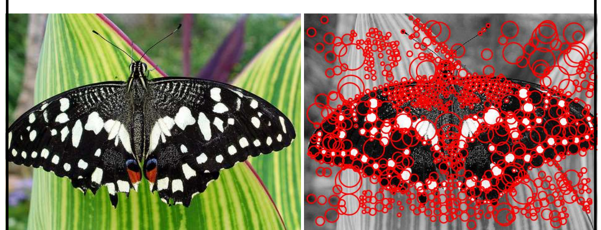


Image credit: Lana Lazebnik

Technical detail

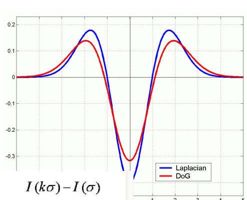
We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

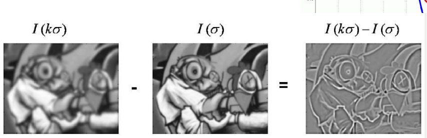
$$L = \sigma^2 \left(G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma) \right)$$

(Laplacian)

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

(Difference of Gaussians)





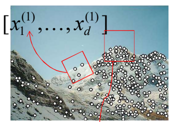
Recap so far: interest points

- Interest point detection
  - Harris corner detector
  - Laplacian of Gaussian, automatic scale selection

Local features: main components

1) Detection: Identify the interest points

2) Description:Extract vector feature descriptor surrounding each interest point.



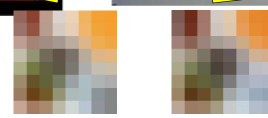
$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

3) Matching: Determine correspondence between descriptors in two views

Kristen Grauman

Geometric transformations

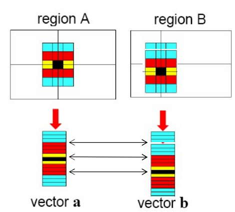


e.g. scale, translation, rotation

Photometric transformations

Figure from T. Tuytelaars ECCV 2006 tutorial

Raw patches as local descriptors

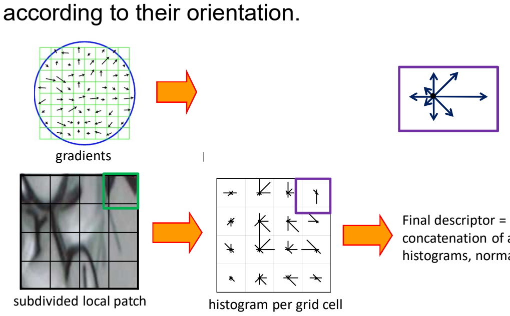


The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector.

But this is very sensitive to even small shifts, rotations.

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

- Use histograms to bin pixels within sub-patches according to their orientation.



Final descriptor = concatenation of all histograms, normalize

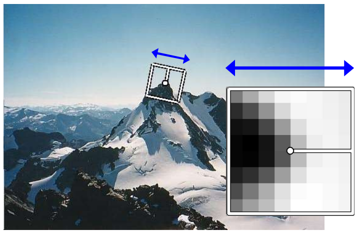
Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

Interest points and their scales and orientations (random subset of 50)

SIFT descriptors

http://www.vlfeat.org/overview/sift.html

Making descriptor rotation invariant

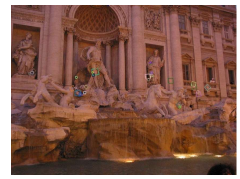


- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation.

Image from Matthew Brown

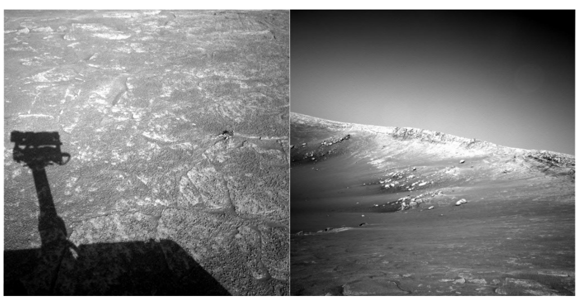
SIFT descriptor [Lowe 2004]

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available, e.g. `http://www.vlfeat.org/overview/sift.html`



Steve Seitz

Example



NASA Mars Rover images

Example

NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

SIFT properties

- Invariant to
  - Scale
  - Rotation
- Partially invariant to
  - Illumination changes
  - Camera viewpoint
  - Occlusion, clutter

Local features: main components

1) Detection: Identify the interest points
2) Description:Extract vector feature descriptor surrounding each interest point.
3) Matching: Determine correspondence between descriptors in two views

Kristen Grauman

Matching local features

Matching local features

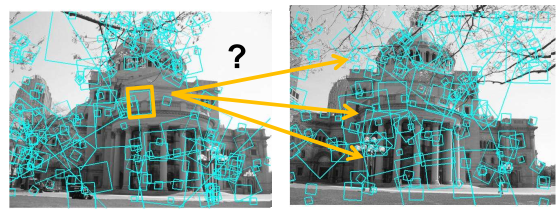


Image 1 Image 2

To generate **candidate matches**, find patches that have the most similar appearance (e.g., lowest SSD)

Simplest approach: compare them all, take the closest (or closest k, or within a thresholded distance)

Ambiguous matches

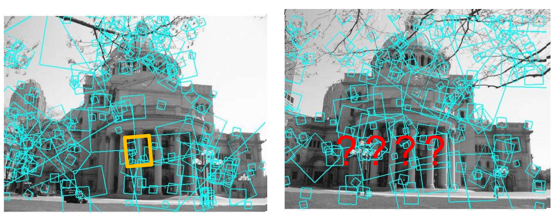


Image 1 Image 2

At what SSD value do we have a good match?

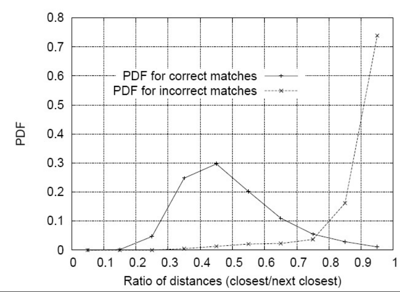
To add robustness to matching, can consider **ratio** : distance to best match / distance to second best match

If low, first match looks good.

If high, could be ambiguous match.

Matching SIFT Descriptors

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



Lowe IJCV 2004

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

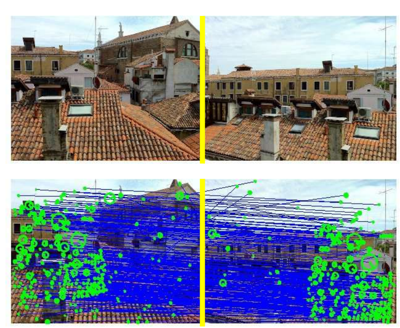
Interest points and their scales and orientations (random subset of 50)

SIFT descriptors

http://www.vlfeat.org/overview/sift.html

SIFT (preliminary) matches

img1 img2



img1 img2

http://www.vlfeat.org/overview/sift.html

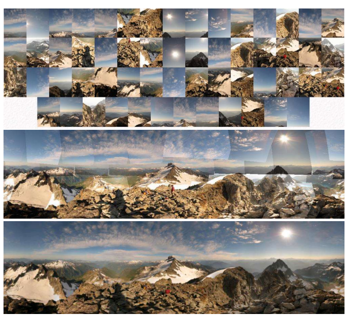
Value of local (invariant) features

- Complexity reduction via selection of distinctive points
- Describe images, objects, parts without requiring segmentation
- Local character means robustness to clutter, occlusion
- Robustness: similar descriptors in spite of noise, blur, etc.

Applications of local invariant features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition
- …

Automatic mosaicing

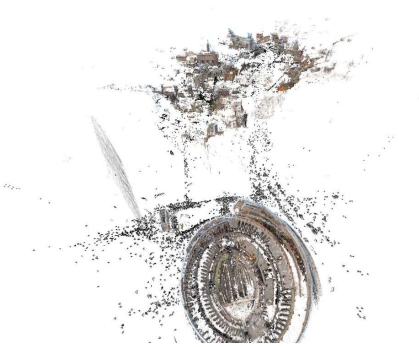


http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Wide baseline stereo

[Image from T. Tuytelaars ECCV 2006 tutorial]

Photo tourism [Snavely et al.]



Recognition of specific objects, scenes

Lowe 1999

Many current applications - 2017

Summary so far

- Interest point detection
  - Harris corner detector
  - Laplacian of Gaussian, automatic scale selection
- Invariant descriptors
  - Rotation according to dominant gradient direction
  - Histograms for robustness to small shifts and translations (SIFT descriptor)

Plan for today

- 1. Basics in feature extraction: filtering
- 2. Invariant local features
- **3. Recognizing object instances**

Recognizing or retrieving specific objects

Example I: Visual search in feature films

Visually defined query

"Groundhog Day" [Rammis, 1993]

"Find this clock"

"Find this place"

Slide credit: J. Sivic

Recognizing or retrieving specific objects

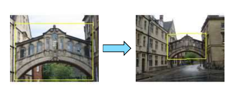
Example II: Search photos on the web for particular places

Find these landmarks ...in these images and 1M more

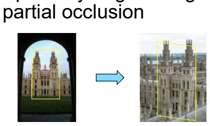
Slide credit: J. Sivic

Why is it difficult?

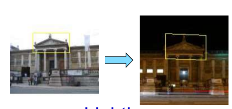
Want to find the object despite possibly large changes in scale, viewpoint, lighting and partial occlusion



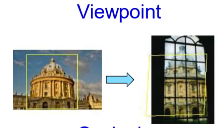
Scale



Viewpoint



Lighting



Occlusion

We can't expect to match such varied instances with a single global template...

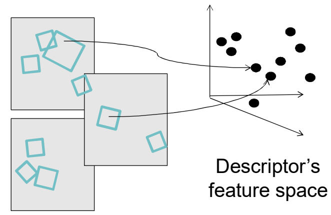
Slide credit: J. Sivic

Instance recognition: key new ideas

- Visual words
  - quantization, index, bags of words
- Spatial verification
  - affine; RANSAC, Hough

Indexing local features

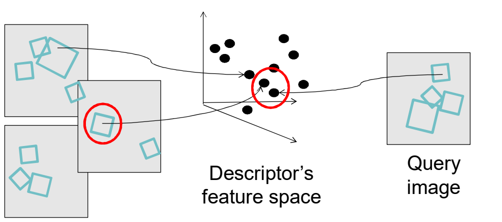
- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)



Kristen Grauman

Indexing local features

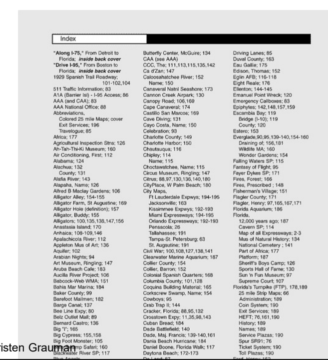
- When we see close points in feature space, we have similar descriptors, which indicates similar local content.



Easily can have millions of features to search!

Kristen Grauman

Indexing local features: inverted file index

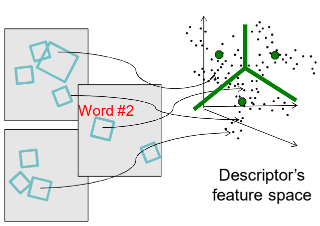


- For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index…
- We want to find all *images* in which a *feature* occurs.
- To use this idea, we'll need to map our features to "visual words".

Kristen Grauman

Visual words

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

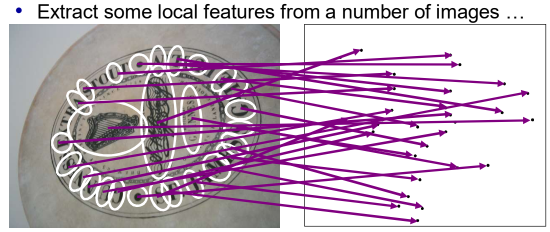


- Quantize via clustering, let cluster centers be the prototype "words"
- Determine which word to assign to each new image region by finding the closest cluster center.

Kristen Grauman

Visual words: main idea

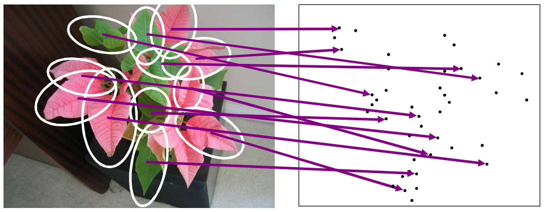
- Extract some local features from a number of images …



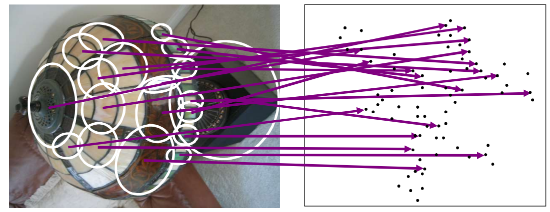
e.g., SIFT descriptor space: each point is 128-dimensional

Slide credit: D. Nister, CVPR 2006

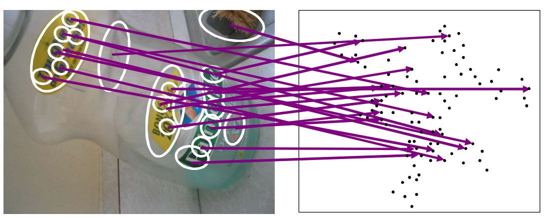
Visual words: main idea

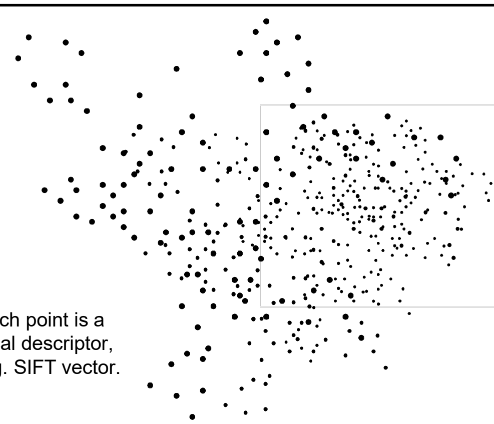


Visual words: main idea

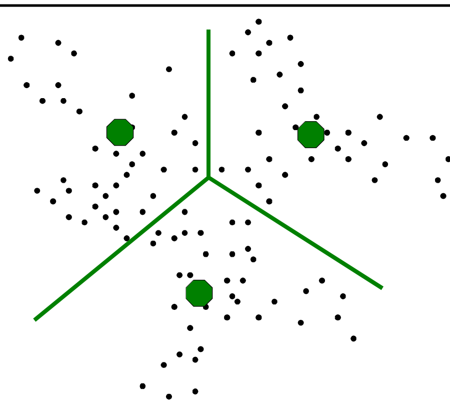


Visual words: main idea



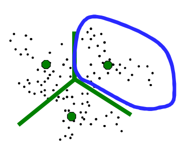


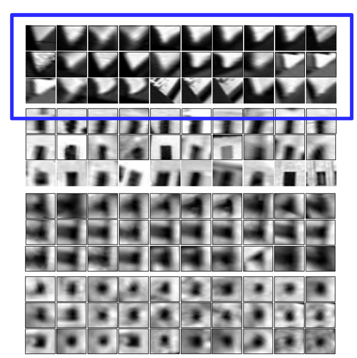
Each point is a local descriptor, e.g. SIFT vector.



Visual words

- Example: each group of patches belongs to the same visual word





Kristen Grauman

Figure from Sivic & Zisserman, ICCV 2003

Inverted file index

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

…

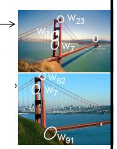
- Database images are loaded into the index mapping words to image numbers

Kristen Grauman

Inverted file index

New query image

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



- New query image is mapped to indices of database images that share a word.

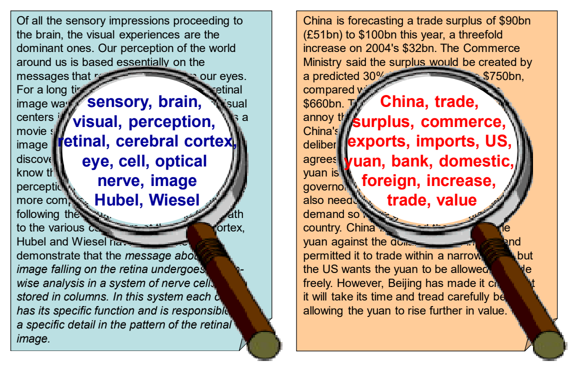
Kristen Grauman

Instance recognition: remaining issues

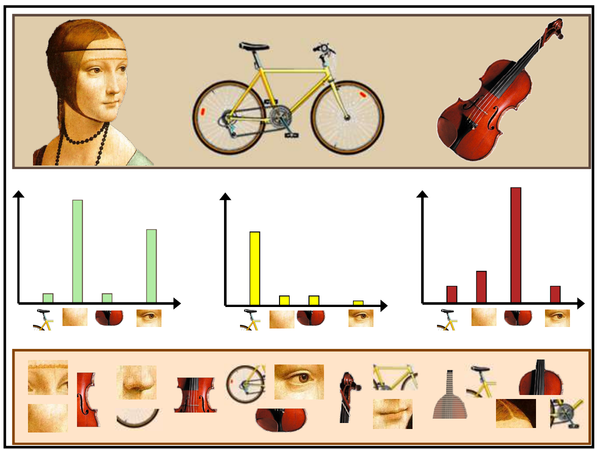
- How to summarize the content of an entire image? And gauge overall similarity?
- How large should the vocabulary be? How to perform quantization efficiently?
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Kristen Grauman

Analogy to documents

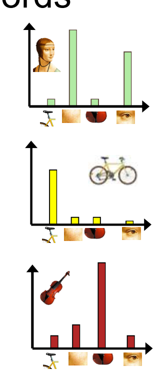


ICCV 2005 short course, L. Fei-Fei



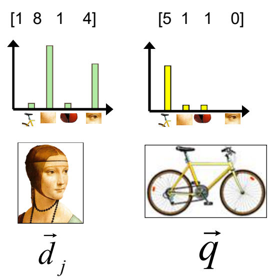
Bags of visual words

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



Comparing bags of words

- Rank frames by normalized scalar product between their (possibly weighted) occurrence counts---*nearest neighbor* search for similar images.



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

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

for vocabulary of *V* words

Inverted file index and bags of words similarity

New query image

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

1. Extract words in query
2. Inverted file index to find relevant frames
3. Compare word counts

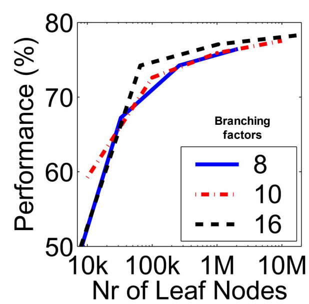
Kristen Grauman

Instance recognition: remaining issues

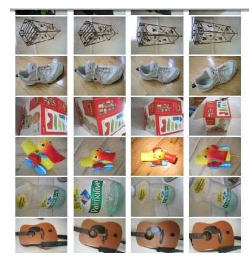
- How to summarize the content of an entire image? And gauge overall similarity?
- How large should the vocabulary be? How to perform quantization efficiently?
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Kristen Grauman

Vocabulary size



Results for recognition task with 6347 images

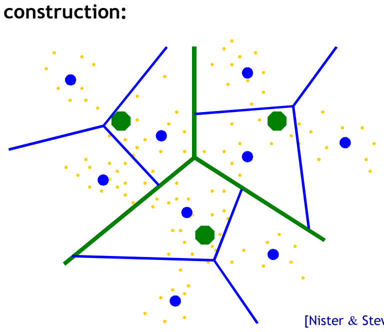


Influence on performance, sparsity?

Nister & Stewenius, CVPR 2006

Vocabulary Trees: hierarchical clustering for large vocabularies

- **Tree construction:**



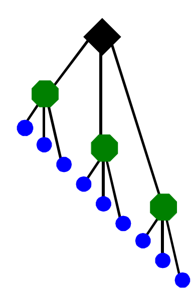
[Nister & Stewenius, CVPR'06]

K. Grauman, B. Leibe

Slide credit: David Nister

Visual Object Recognition Tutorial

Vocabulary Tree



[Nister & Stewenius, CVPR'06]

K. Grauman, B. Leibe

Slide credit: David Nister

Visual Object Recognition Tutorial

Vocabulary trees: complexity

Number of words given tree parameters:
branching factor and number of levels

Word assignment cost vs. flat vocabulary

Visual words/bags of words

+ flexible to geometry / deformations / viewpoint
+ compact summary of image content
+ provides vector representation for sets
+ very good results in practice

- background and foreground mixed when bag covers whole image
- optimal vocabulary formation remains unclear
- basic model ignores geometry – must verify afterwards, or encode via features

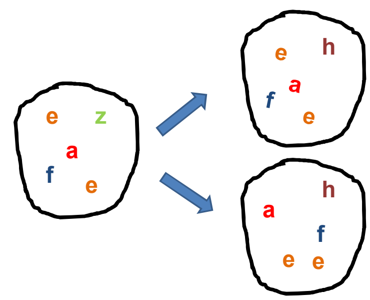
Kristen Grauman

Instance recognition: remaining issues

- How to summarize the content of an entire image? And gauge overall similarity?
- How large should the vocabulary be? How to perform quantization efficiently?
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

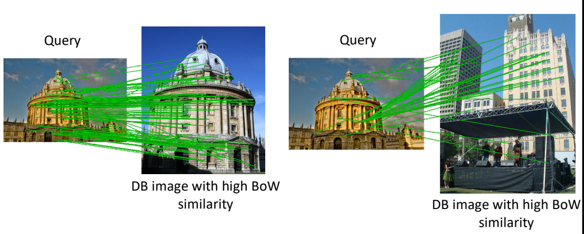
Kristen Grauman

Which matches better?



Derek Hoiem

Spatial Verification



Both image pairs have many visual words in common.

Slide credit: Ondrej Chum

Spatial Verification

Only some of the matches are mutually consistent

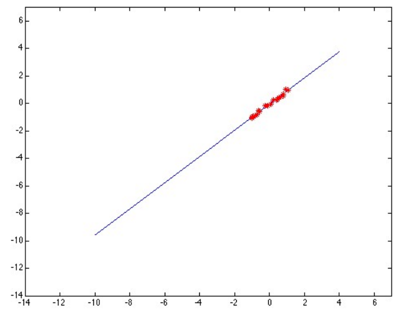
Slide credit: Ondrej Chum

Spatial Verification: two basic strategies

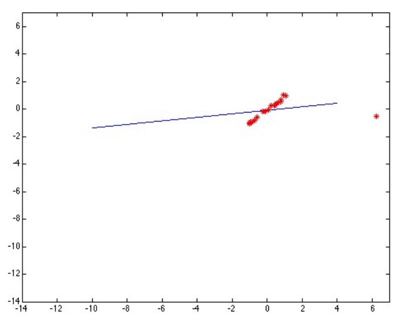
- RANSAC
- Generalized Hough Transform

Kristen Grauman

Outliers affect least squares fit



Outliers affect least squares fit



RANSAC

- RANdom Sample Consensus
- **Approach**: we want to avoid the impact of outliers, so let's look for "inliers", and use those only.
- **Intuition**: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

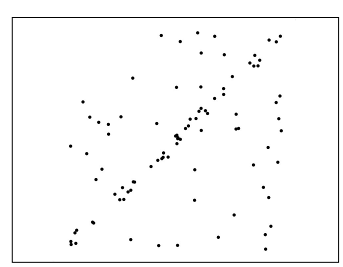
RANSAC for line fitting

Repeat ***N*** times:

- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Lana Lazebnik

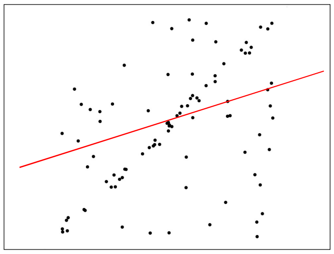
RANSAC for line fitting example



Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

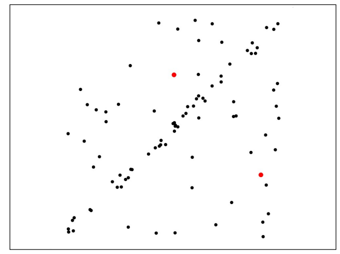


Least-squares fit

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

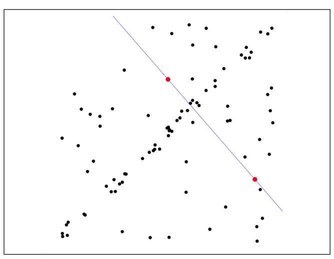


1. Randomly select minimal subset of points

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

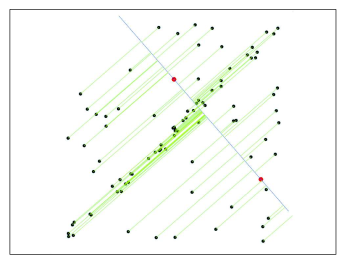


1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

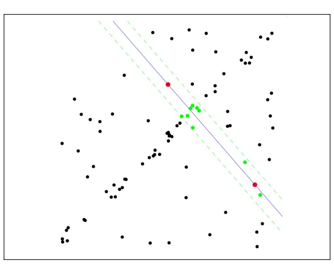


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

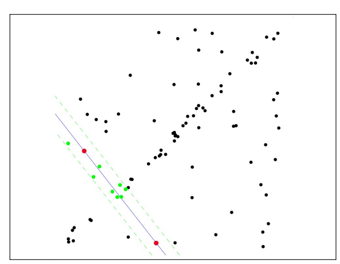


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

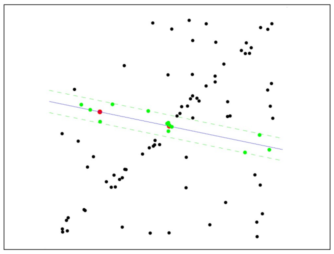


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

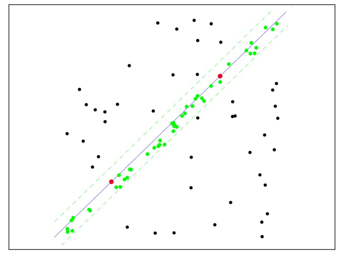
211

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

Uncontaminated sample



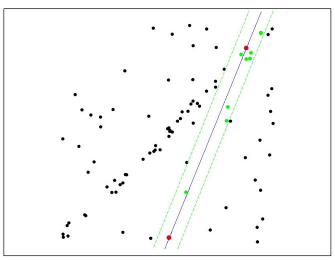
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

212

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

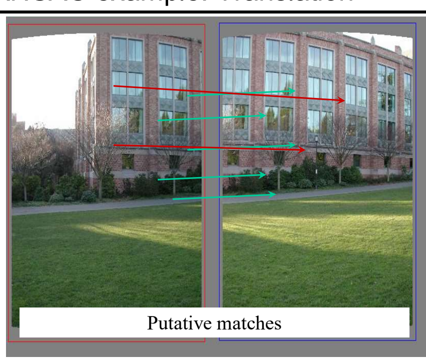
Source: R. Raguram

Lana Lazebnik

That is an example fitting a model (line)…

What about fitting a transformation (translation)?

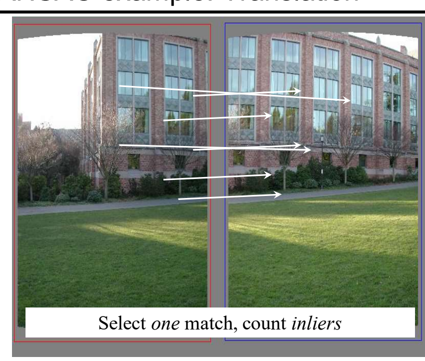
RANSAC example: Translation



Putative matches

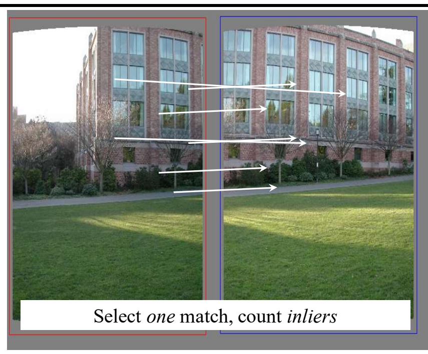
Source: Rick Szeliski

RANSAC example: Translation

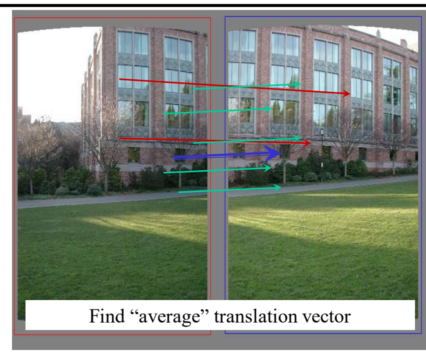


Select *one* match, count *inliers*

RANSAC example: Translation



RANSAC example: Translation



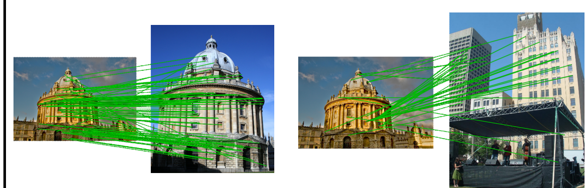
RANSAC: General form

- RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate
2. Compute model from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute estimate of model on all of the inliers

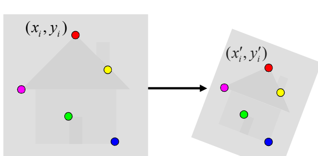
- Keep the model with the largest number of inliers

RANSAC verification



For matching specific scenes/objects, common to use an **affine transformation** for spatial verification

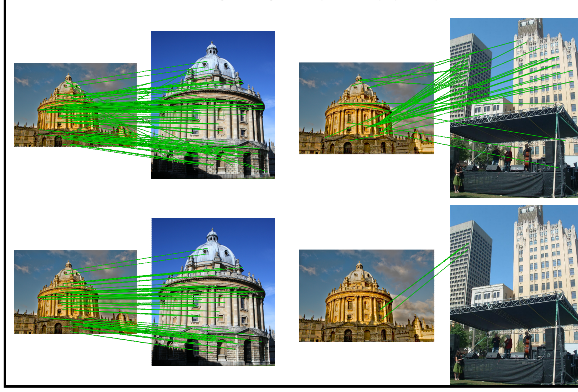
Fitting an affine transformation



Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras.

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

RANSAC verification



Spatial Verification: two basic strategies

- RANSAC
  - Typically sort by BoW similarity as initial filter
  - Verify by checking support (inliers) for possible affine transformations
    - e.g., "success" if find an affine transformation with > N inlier correspondences
- Generalized Hough Transform
  - Let each matched feature cast a vote on location, scale, orientation of the model object
  - Verify parameters with enough votes

Kristen Grauman

Spatial Verification: two basic strategies

- RANSAC
  - Typically sort by BoW similarity as initial filter
  - Verify by checking support (inliers) for possible affine transformations
    - e.g., "success" if find an affine transformation with > N inlier correspondences
- Generalized Hough Transform
  - Let each matched feature cast a vote on location, scale, orientation of the model object
  - Verify parameters with enough votes

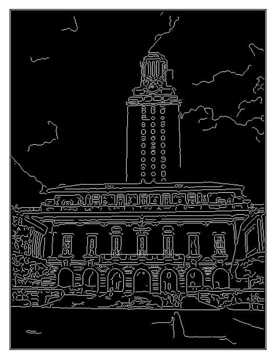
Kristen Grauman

Voting

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features *vote for all models that are compatible with it*.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, *but* typically their votes should be inconsistent with the majority of "good" features.

Kristen Grauman

Difficulty of line fitting

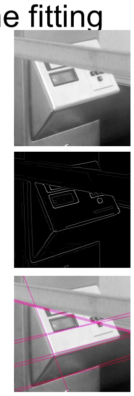


Kristen Grauman

Hough Transform for line fitting

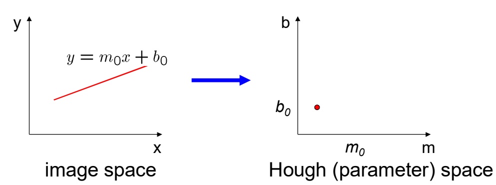
- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?
- **Hough Transform** is a voting technique that can be used to answer all of these questions.

  Main idea:
  1. Record vote for each possible line on which each edge point lies.
  2. Look for lines that get many votes.



Kristen Grauman

Finding lines in an image: Hough space



Connection between image (x,y) and Hough (m,b) spaces
- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

Slide credit: Steve Seitz

Finding lines in an image: Hough space

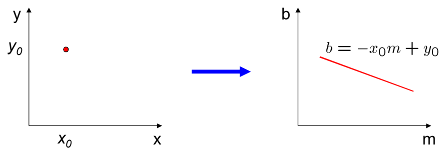


image space

Hough (parameter) space

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - this is a line in Hough space

Slide credit: Steve Seitz

Finding lines in an image: Hough space

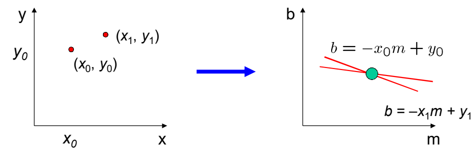


image space

Hough (parameter) space

What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1)?

- It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

Finding lines in an image: Hough algorithm

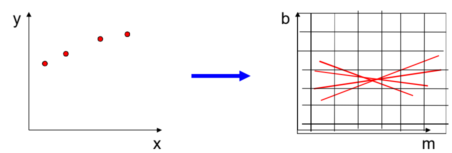


image space

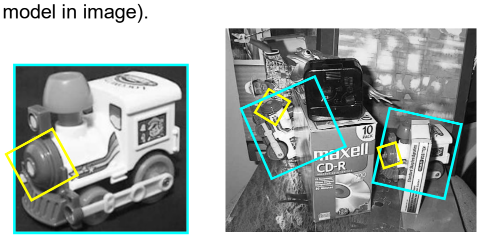
Hough (parameter) space

How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?

- Let each edge point in image space *vote* for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Voting: Generalized Hough Transform

- If we use scale, rotation, and translation invariant local features, then each feature match gives an alignment hypothesis (for scale, translation, and orientation of model in image).



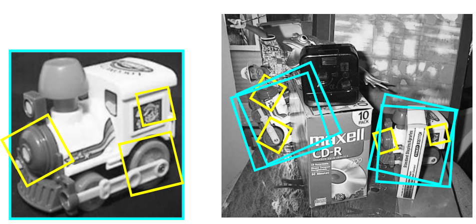
Model

Novel image

Adapted from Lana Lazebnik

Voting: Generalized Hough Transform

- A hypothesis generated by a single match may be unreliable,
- So let each match **vote** for a hypothesis in Hough space



Model

Novel image

Gen Hough Transform details (Lowe's system)

- **Training phase:** For each model feature, record 2D location, scale, and orientation of model (relative to normalized feature frame)
- **Test phase:** Let each match btwn a test SIFT feature and a model feature vote in a 4D Hough space
  - Use broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times image size for location
  - Vote for two closest bins in each dimension
- Find all bins with at least three votes and perform geometric verification
  - Estimate least squares *affine* transformation
  - Search for additional features that agree with the alignment

David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Slide credit: Lana Lazebnik

Example result

Background subtract for model boundaries

Objects recognized,

Recognition in spite of occlusion

[Lowe]

Gen Hough vs RANSAC

GHT
- Single correspondence -> vote for all consistent parameters
- Represents uncertainty in the model parameter space
- Linear complexity in number of correspondences and number of voting cells; beyond 4D vote space impractical
- Can handle high outlier ratio

RANSAC
- Minimal subset of correspondences to estimate model -> count inliers
- Represents uncertainty in image space
- Must search all data points to check for inliers each iteration
- Scales better to high-d parameter spaces

Kristen Grauman

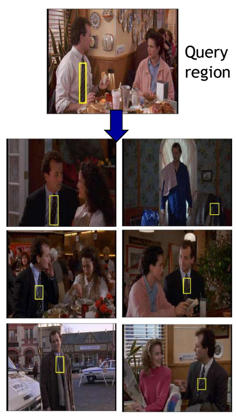
Visual Object Recognition Tutorial

Video Google System

1. Collect all words within query region
2. Inverted file index to find relevant frames
3. Compare word counts
4. Spatial verification

Sivic & Zisserman, ICCV 2003

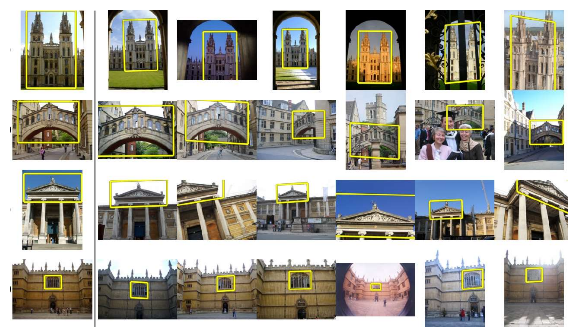
- Demo online at :
- http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html



Query region

Retrieved frames

Object retrieval with large vocabularies and fast spatial matching, Philbin et al., CVPR 2007



Query

Results from 5k Flickr images (demo available for 100k set)

[Philbin CVPR'07]

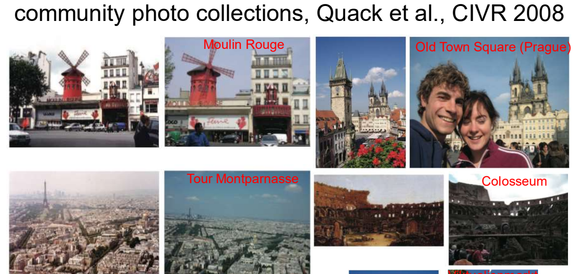
Instance recognition applications

- Snap, pick, pay

- https://www.usatoday.com/videos/tech/2014/10/31/18261641/

Slide credit: Kristen Grauman

World-scale mining of objects and events from community photo collections, Quack et al., CIVR 2008



Auto-annotate by connecting to content on Wikipedia!

Example Applications

Visual Object Recognition Tutorial

Mobile tourist guide
- Self-localization
- Object/building recognition
- Photo/video augmentation

B. Leibe

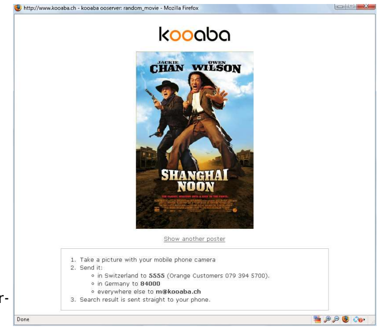
[Quack, Leibe, Van Gool, CIVR'08]

Web Demo: Movie Poster Recognition

Visual Object Recognition Tutorial

50'000 movie posters indexed

Query-by-image from mobile phone available in Switzer-land



http://www.kooaba.com/en/products_engine.html#

Recognition via feature matching+spatial verification

Pros:
- Effective when we are able to find reliable features within clutter
- Great results for matching specific instances

Cons:
- Scaling with number of models
- Spatial verification as post-processing – not seamless, expensive for large-scale problems
- Not suited for category recognition.

Kristen Grauman

Summary (Part 3)

- **Matching local invariant features**
  - To find specific objects and scenes.
- **Bag of words** representation: quantize feature space to make discrete set of visual words
  - Summarize image by distribution of words
  - Index individual words
- **Inverted index**: pre-compute index to enable faster search at query time
- **Recognition of instances via alignment:** matching local features followed by spatial verification
  - Robust fitting : RANSAC, GHT

Kristen Grauman

Coming up

- Today - sign sheet if not registered / on wait list
- Read assigned papers, review 2
  - Don't be afraid of the ImageNet IJCV paper!
- Assignment 1 out now, due Sept 22