

Local Features

CMP719– Computer Vision

Pinar Duygulu

Hacettepe University

Image matching



by [Diva Sian](#)



by [swashford](#)

Harder case

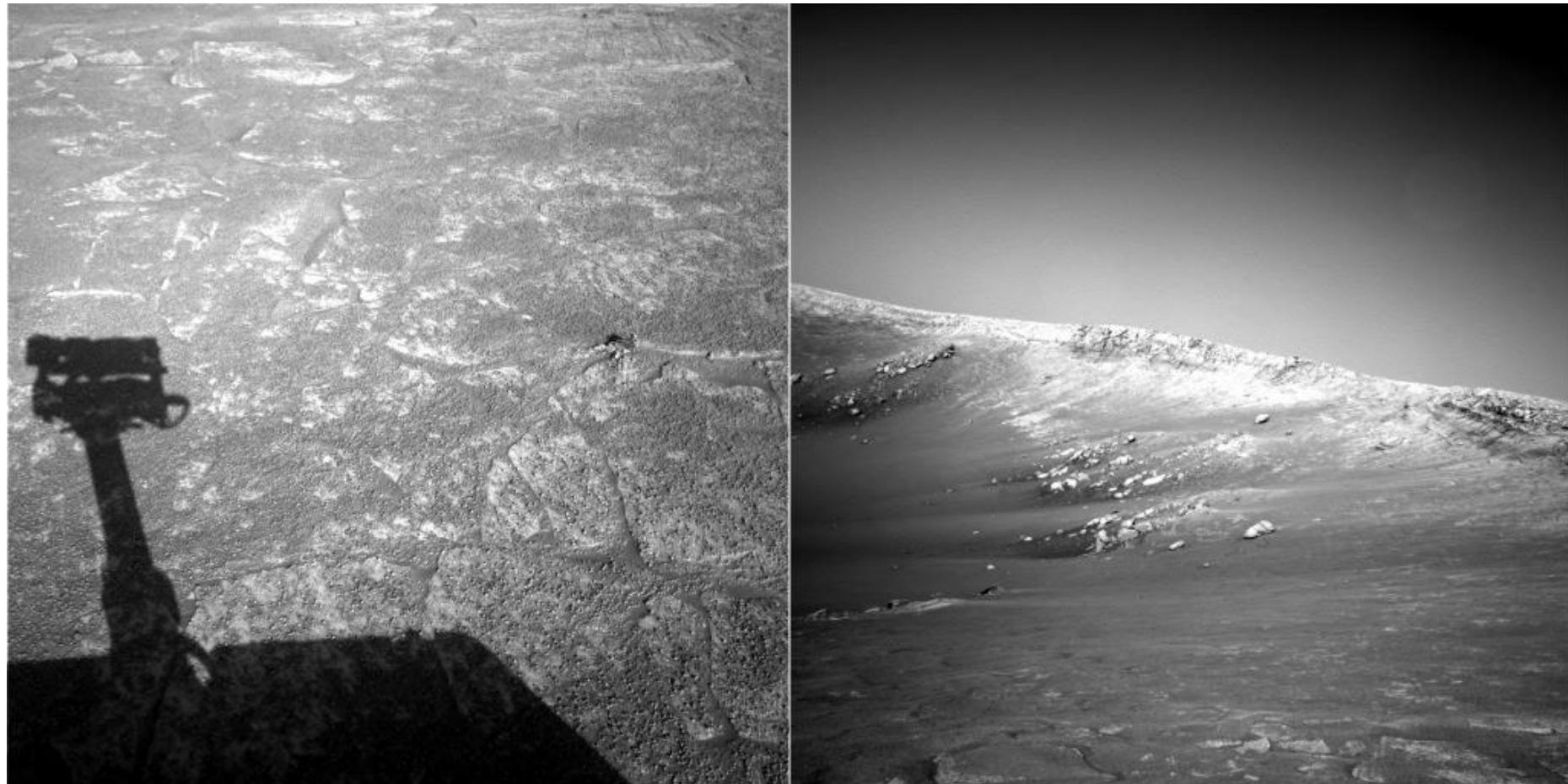


by [Diva Sian](#)



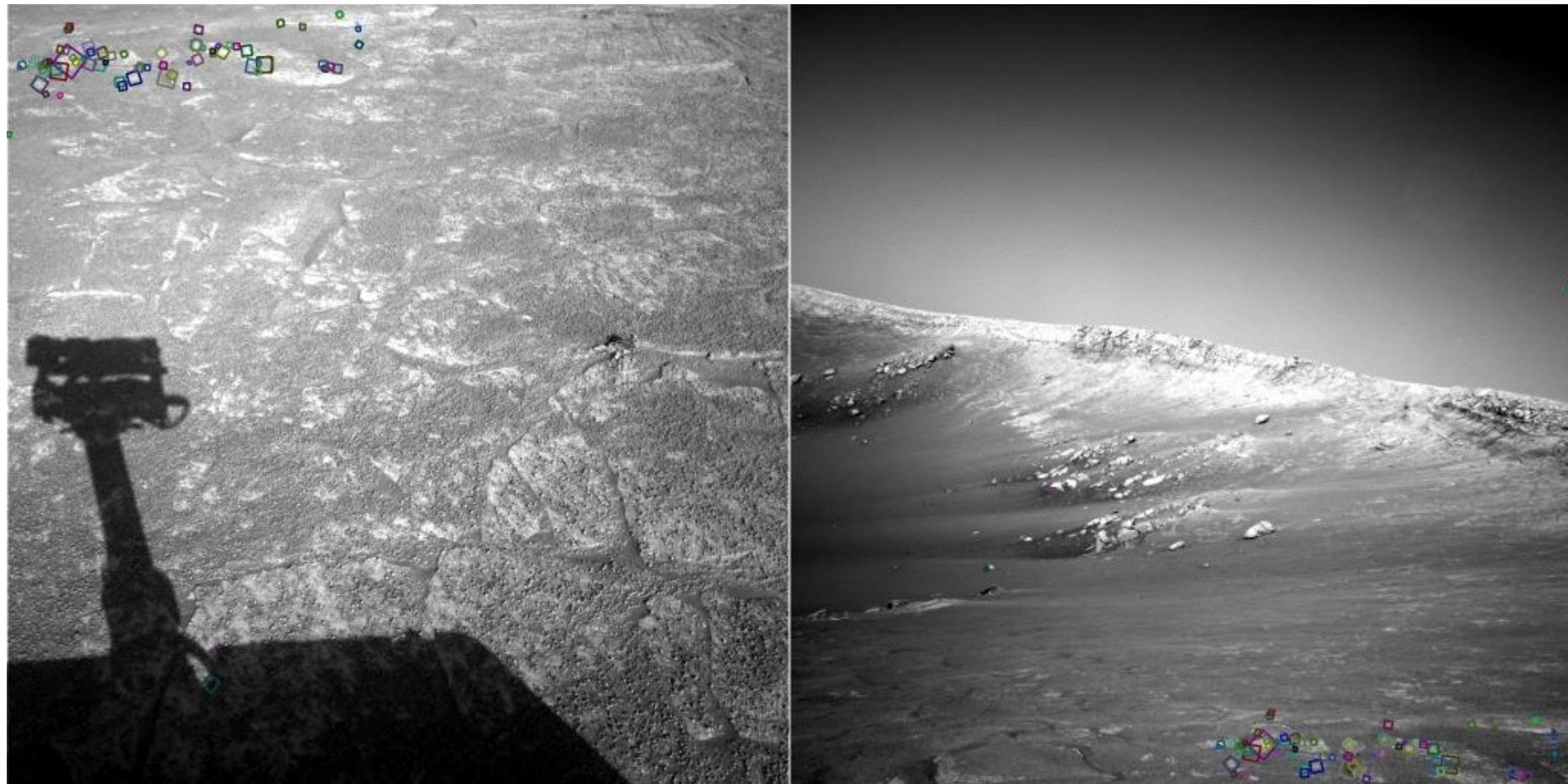
by [scqbt](#)

Harder still?



NASA Mars Rover images

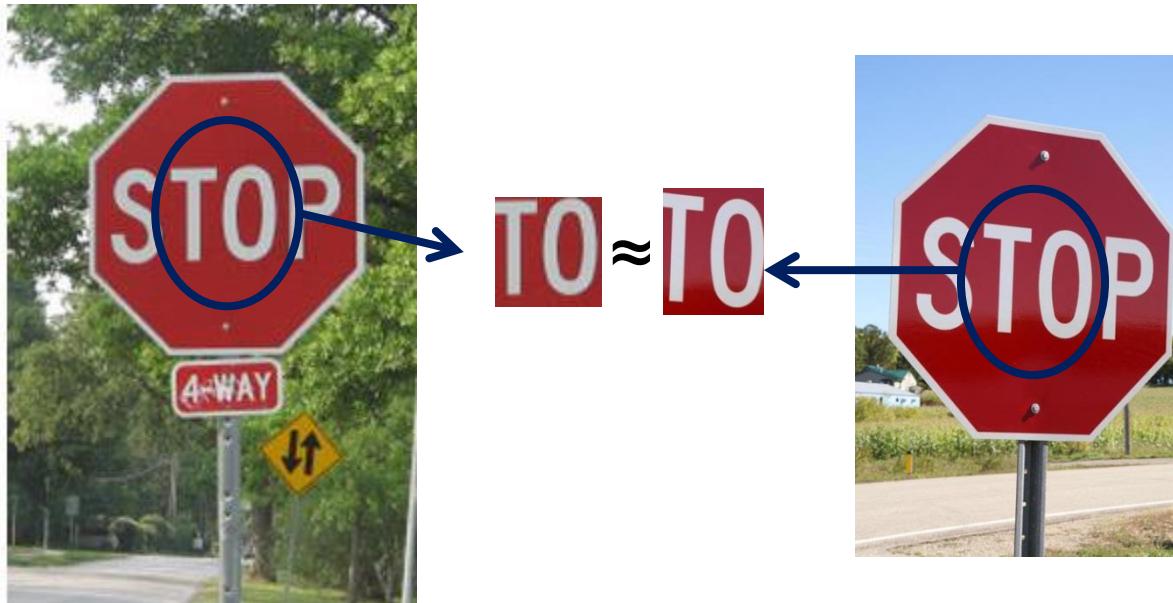
Answer below (look for tiny colored squares...)



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

This section: correspondence and alignment

- Correspondence: matching points, patches, edges, or regions across images

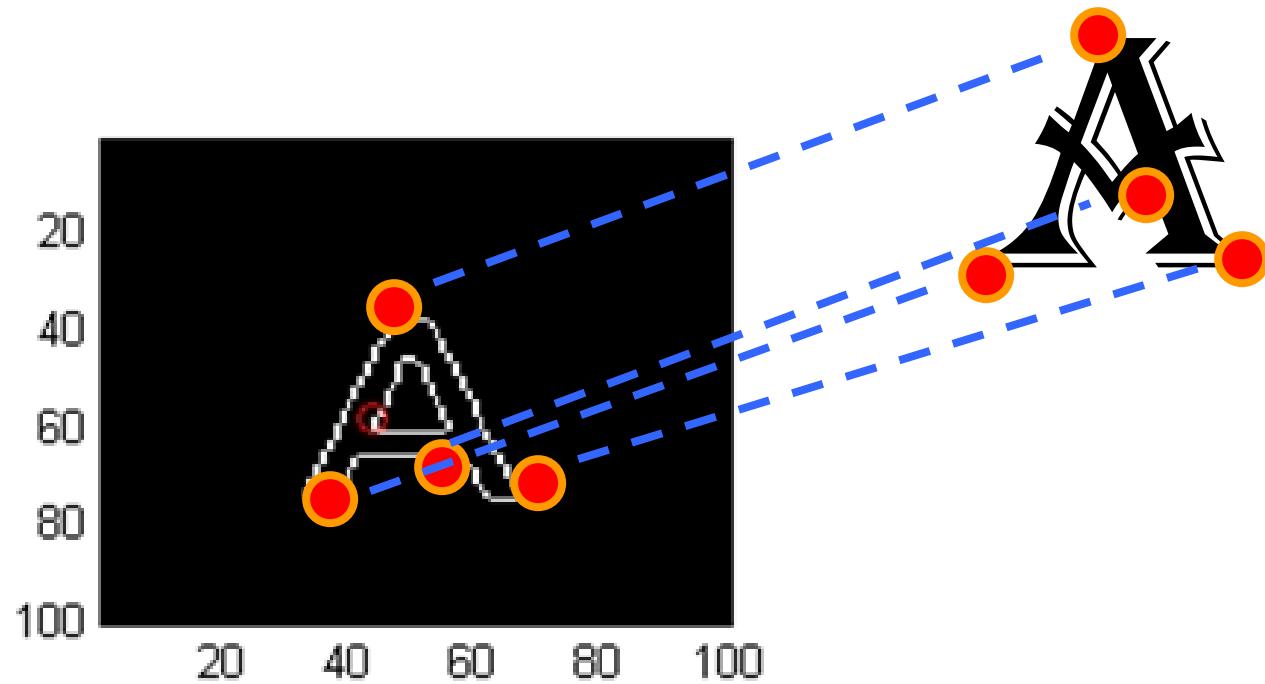


This section: correspondence and alignment

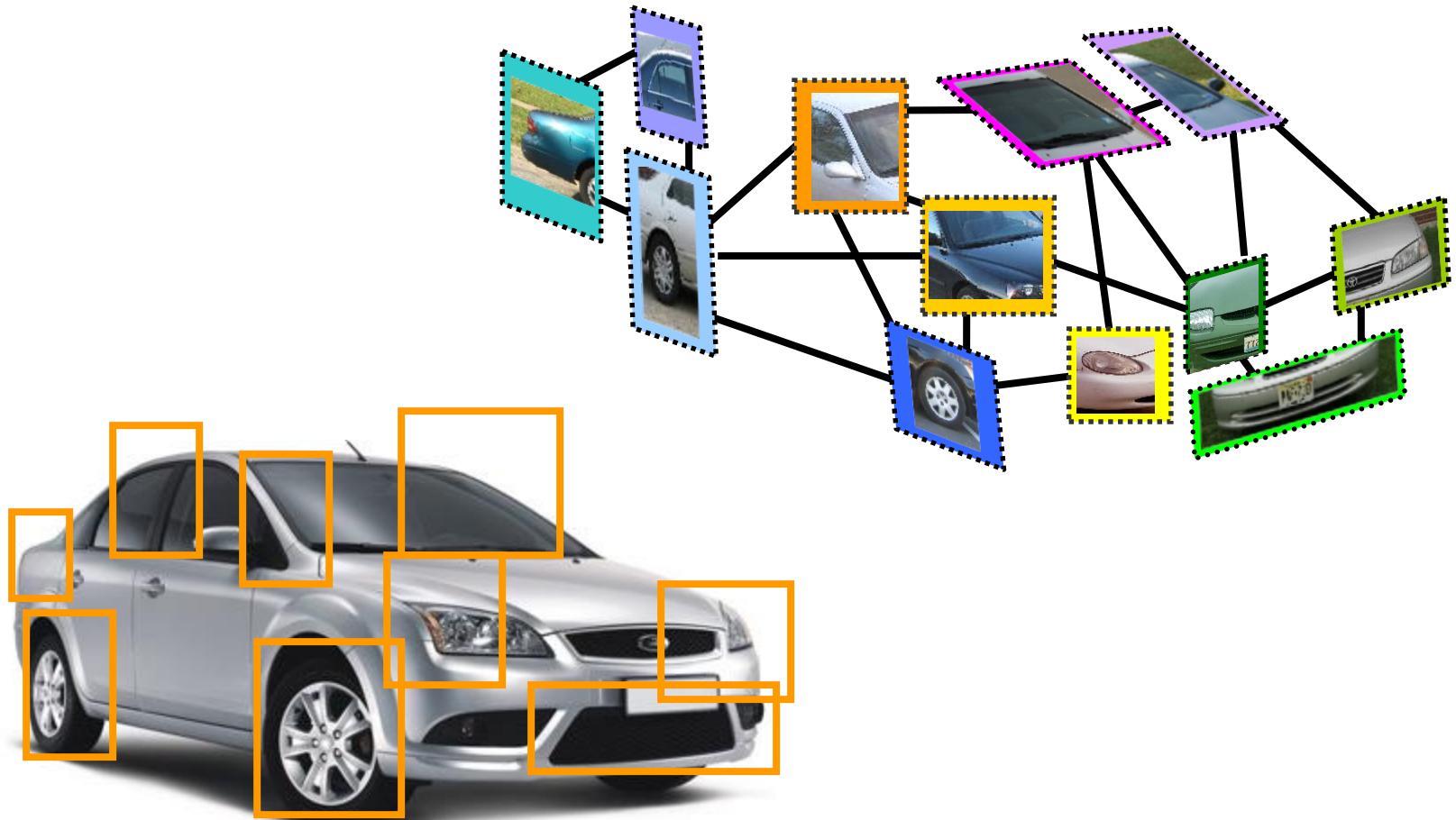
- Alignment: solving the transformation that makes two things match better



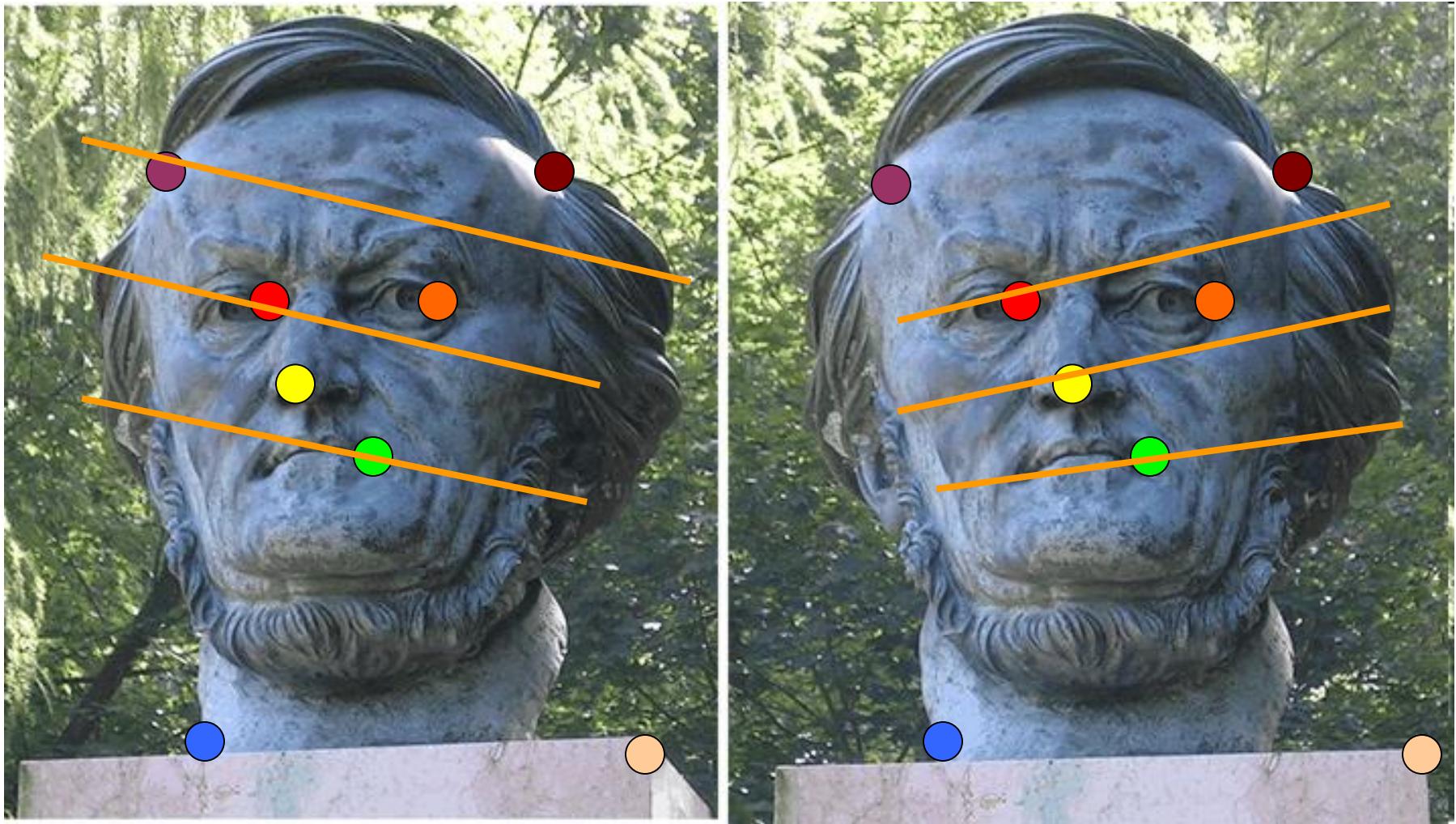
Example: fitting an 2D shape template



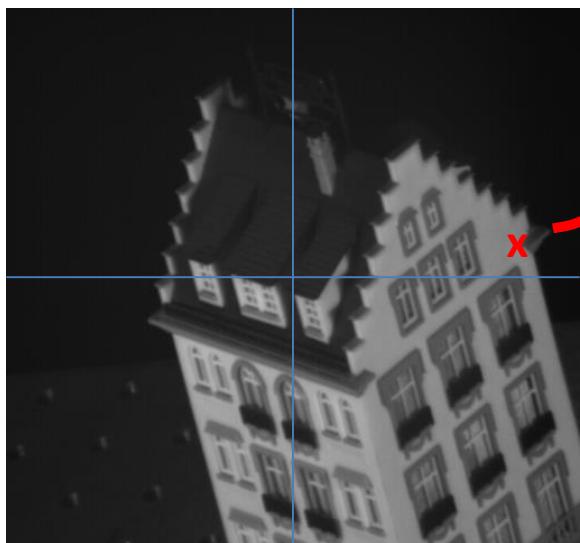
Example: fitting a 3D object model



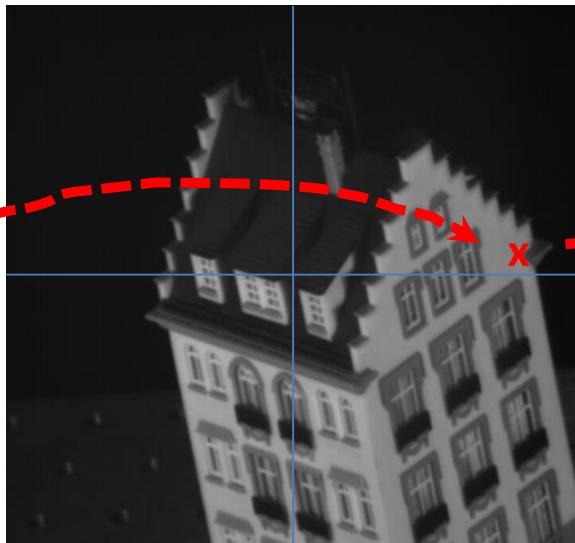
Example: estimating “fundamental matrix” that corresponds two views



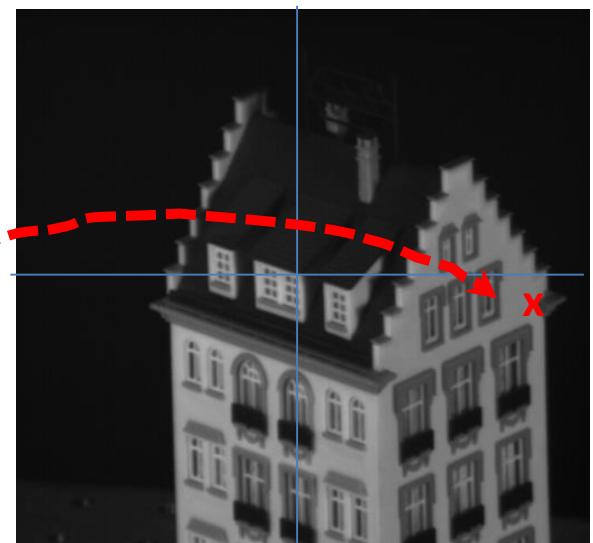
Example: tracking points



frame 0



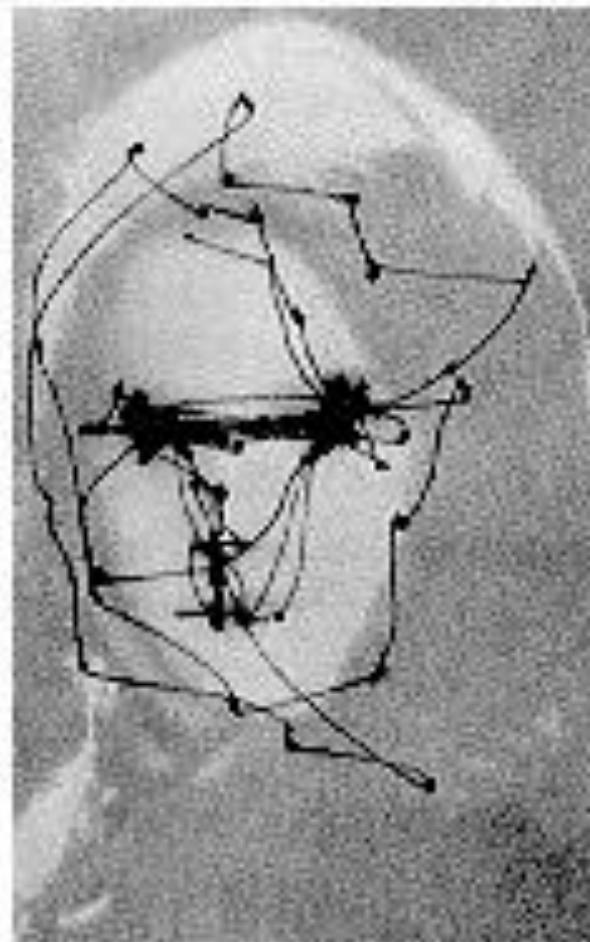
frame 22



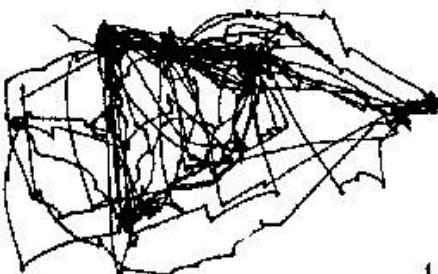
frame 49

Your problem 1 for HW 2!

Human eye movements

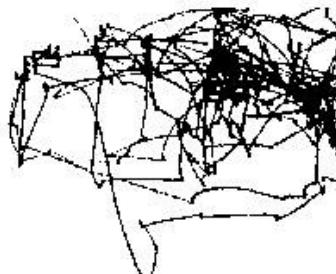


Yarbus eye tracking



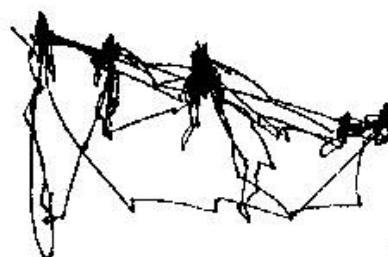
1

Free examination.



2

Estimate material circumstances
of the family



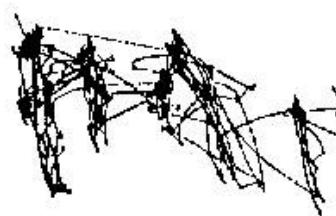
3

Give the ages of the people.



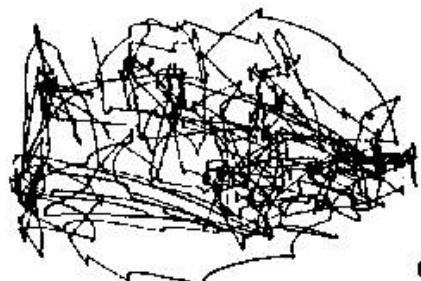
4

Surmise what the family had
been doing before the arrival
of the unexpected visitor.



5

Remember the clothes
worn by the people.



6

Remember positions of people and
objects in the room.



7

Estimate how long the visitor had
been away from the family.

3 min. recordings
of the same
subject

Study by Yarbus

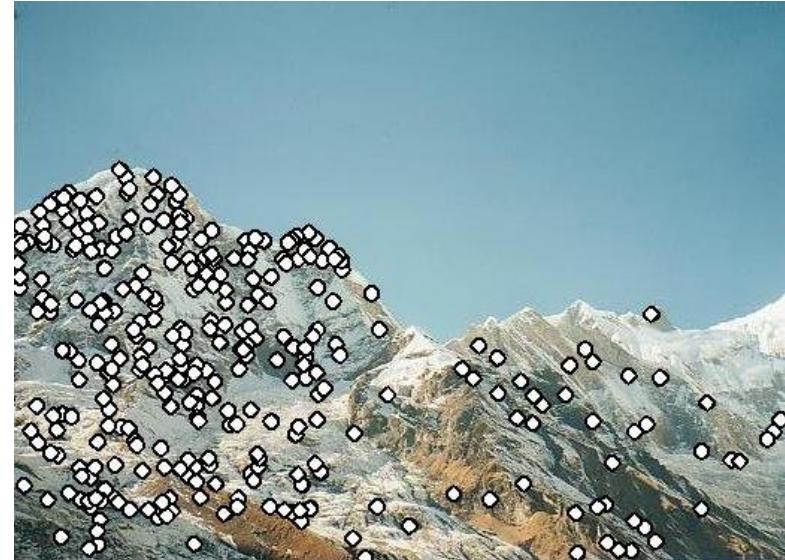
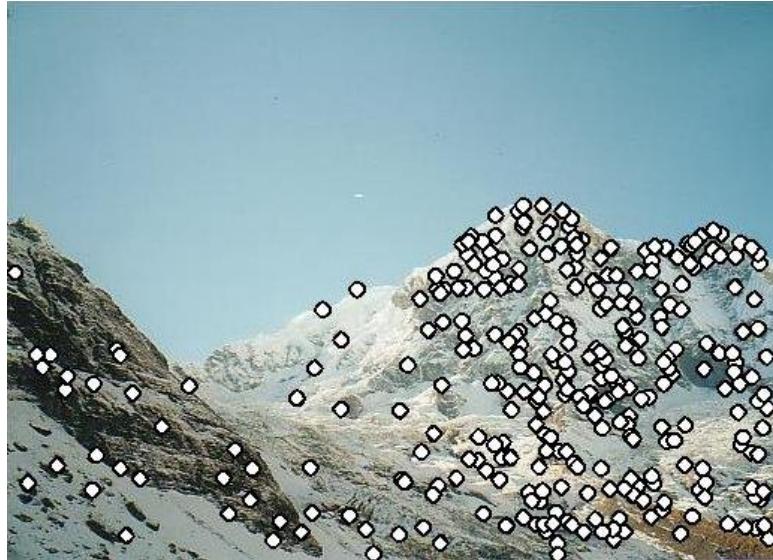
Local features and alignment



- We need to match (align) images
- Global methods sensitive to occlusion, lighting, parallax effects. So look for local features that match well.
- How would you do it by eye?

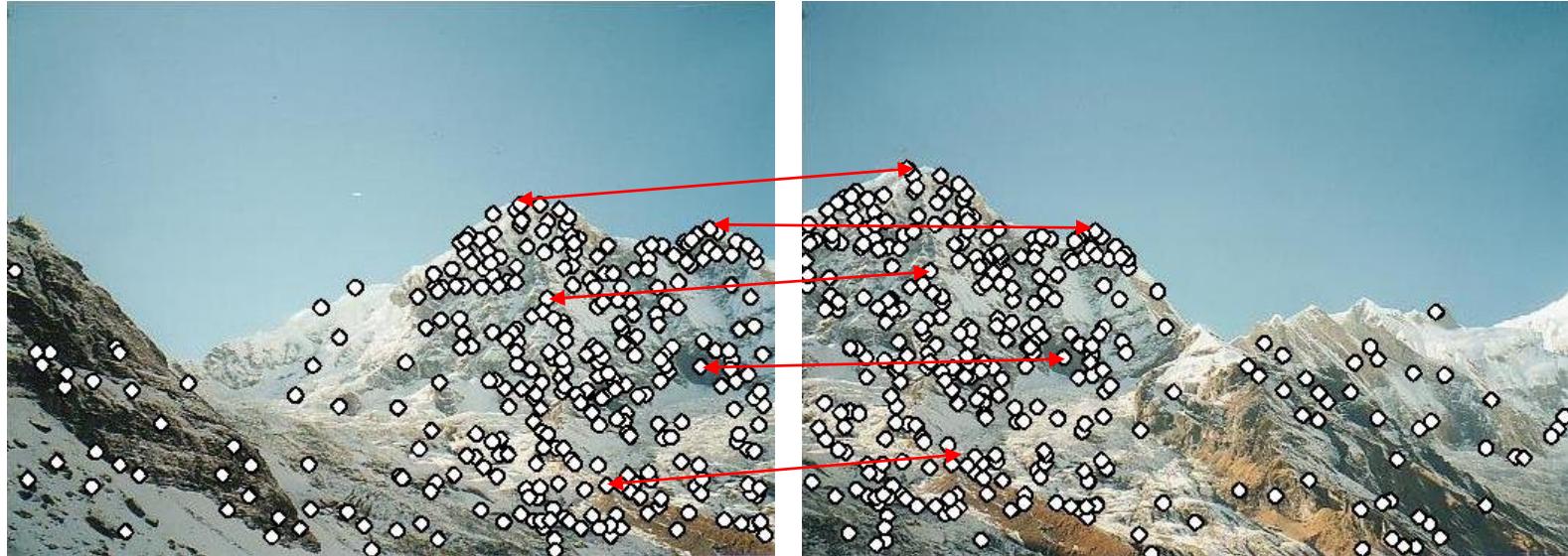
Local features and alignment

- Detect feature points in both images



Local features and alignment

- Detect feature points in both images
- Find corresponding pairs



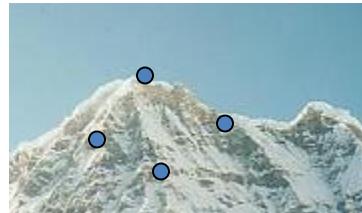
Local features and alignment

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images



Local features and alignment

- Problem 1:
 - Detect the *same* point *independently* in both images

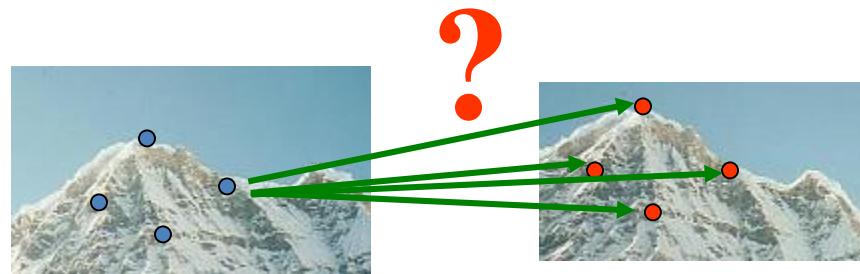


no chance to match!

We need a repeatable detector

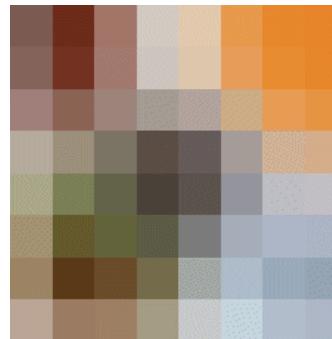
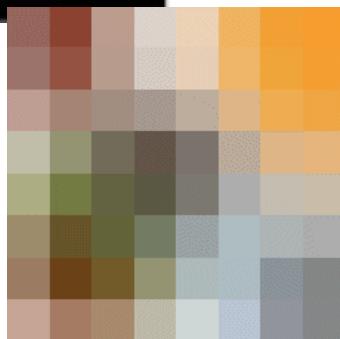
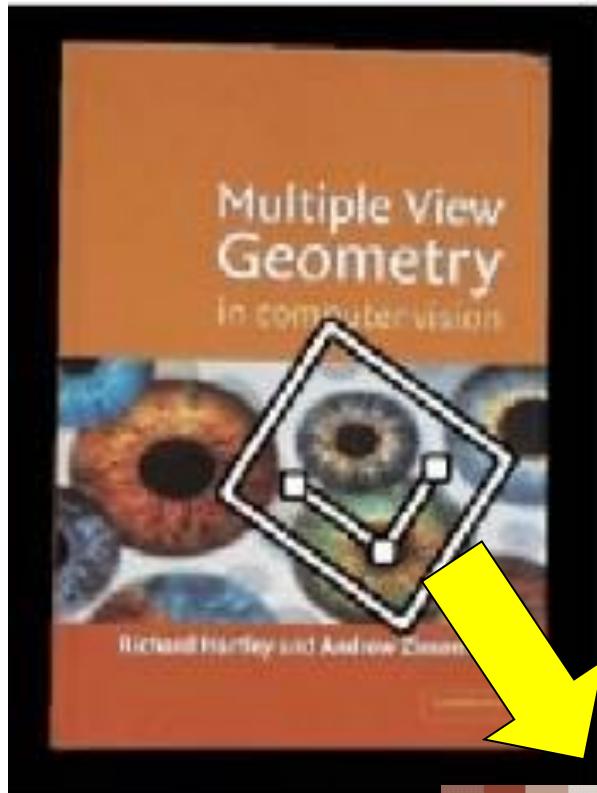
Local features and alignment

- Problem 2:
 - For each point correctly recognize the corresponding one



We need a reliable and distinctive **descriptor**

Geometric transformations



Photometric transformations



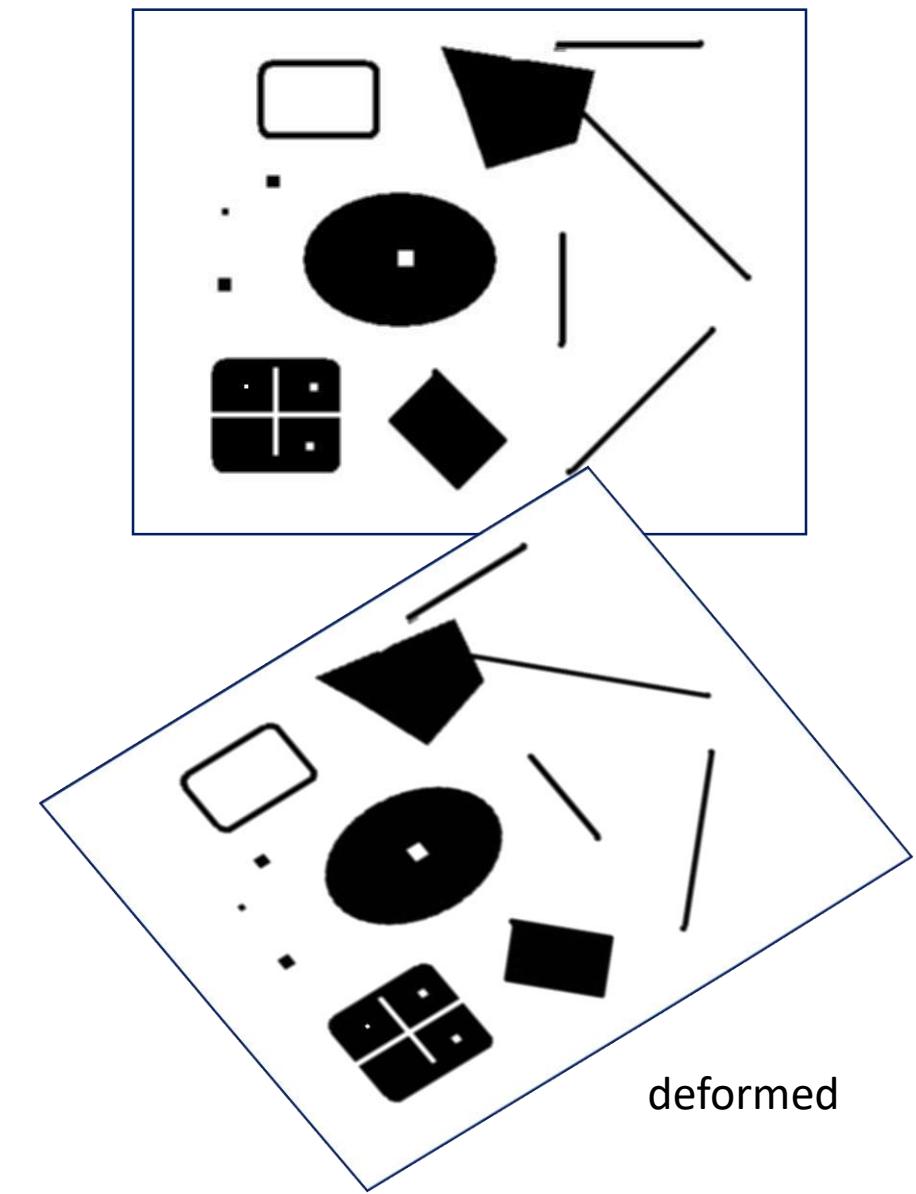
Figure from T. Tuytelaars ECCV 2006 tutorial

And other nuisances...

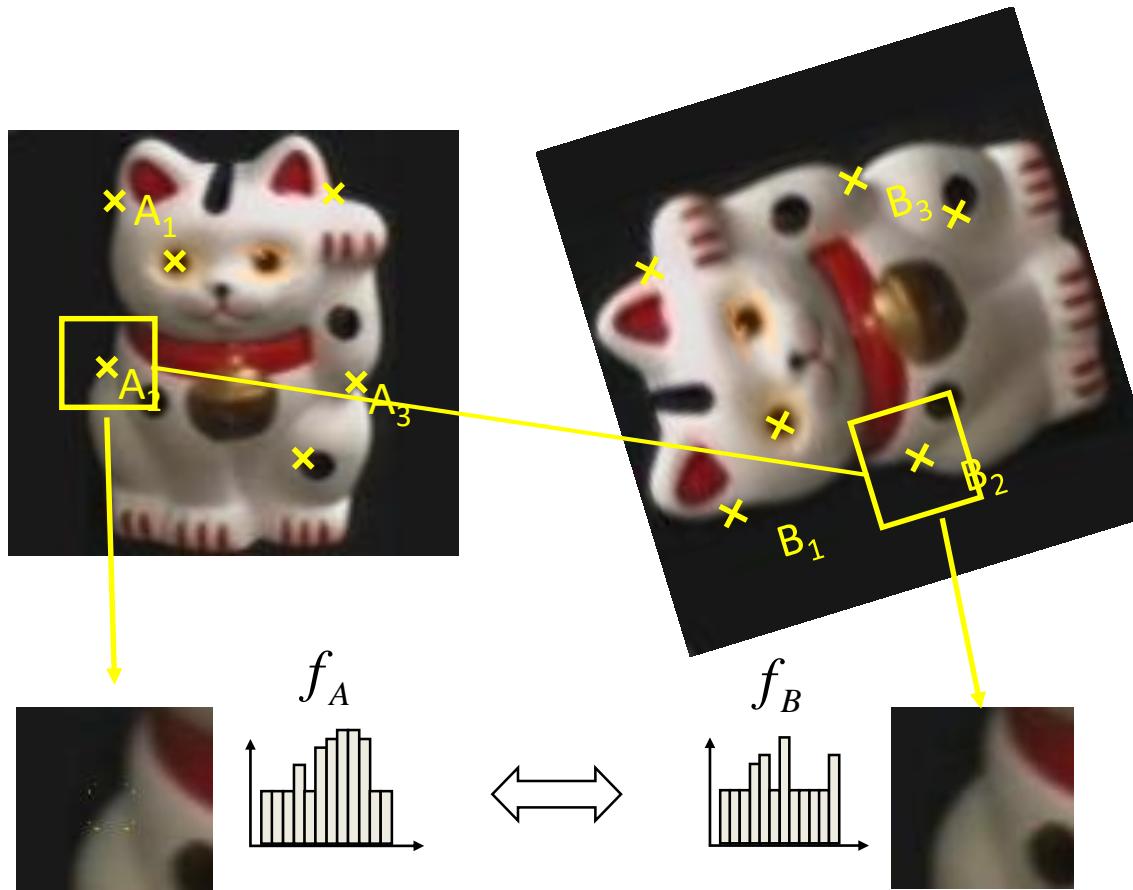
- Noise
- Blur
- Compression artifacts
- ...

This class: 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?



Overview of Keypoint Matching



$$d(f_A, f_B) < T$$

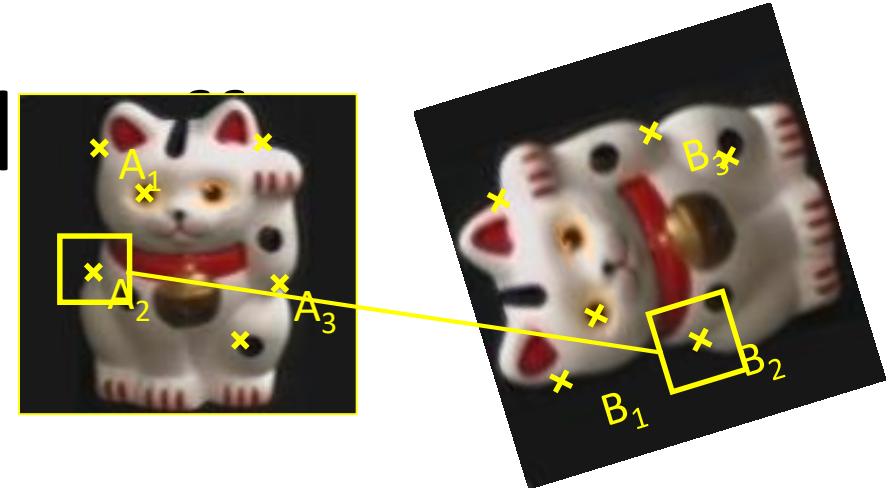
- Find a set of distinctive keypoints**
- Define a region around each keypoint**
- Extract and normalize the region content**
- Compute a local descriptor from the normalized region**
- Match local descriptors**

Goals for Keypoints



Detect points that are *repeatable* and *distinctive*

Key trade-offs



Detection



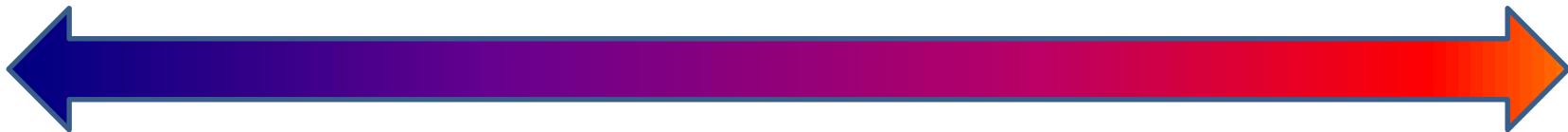
More Repeatable

Robust detection
Precise localization

More Points

Robust to occlusion
Works with less texture

Description



More Distinctive

Minimize wrong matches

More Flexible

Robust to expected variations
Maximize correct matches

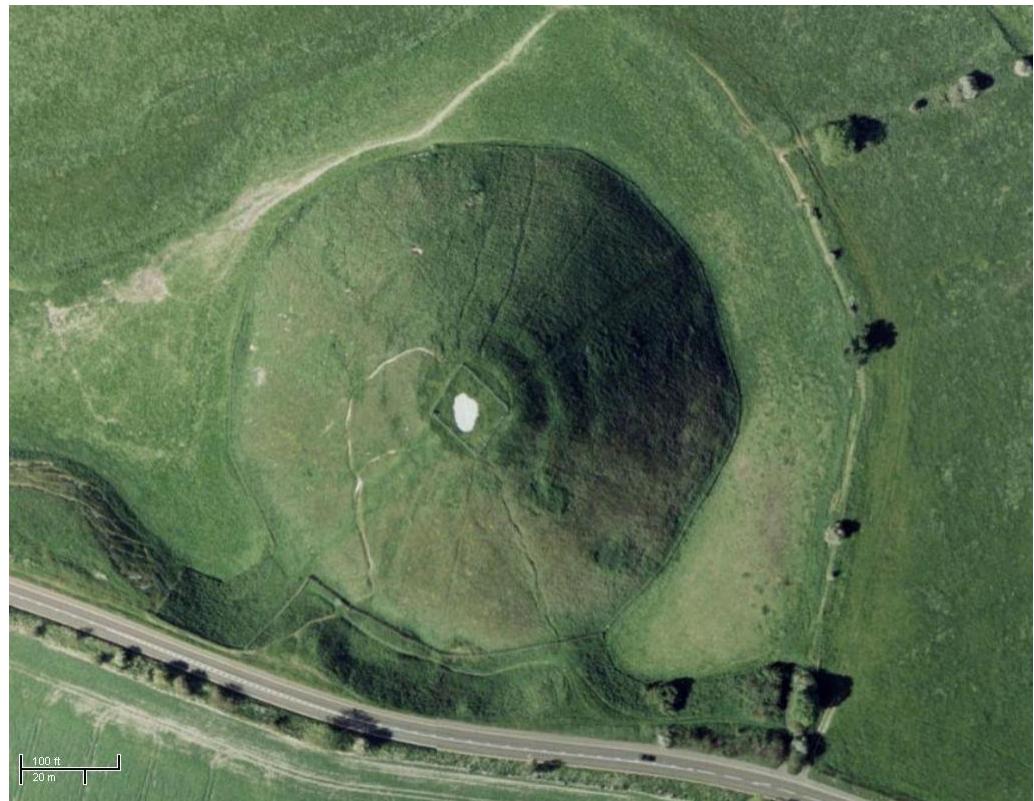
Choosing interest points

Where would you
tell your friend to
meet you?



Choosing interest points

Where would you
tell your friend to
meet you?



Many Existing Detectors Available

Hessian & Harris

[Beaudet '78], [Harris '88]

Laplacian, DoG

[Lindeberg '98], [Lowe 1999]

Harris-/Hessian-Laplace

[Mikolajczyk & Schmid '01]

Harris-/Hessian-Affine

[Mikolajczyk & Schmid '04]

EBR and IBR

[Tuytelaars & Van Gool '04]

MSER

[Matas '02]

Salient Regions

[Kadir & Brady '01]

Others...

Invariant local features

Subset of local feature types designed to be invariant to common geometric and photometric transformations.

Basic steps:

- 1) Detect distinctive interest points
- 2) Extract invariant descriptors

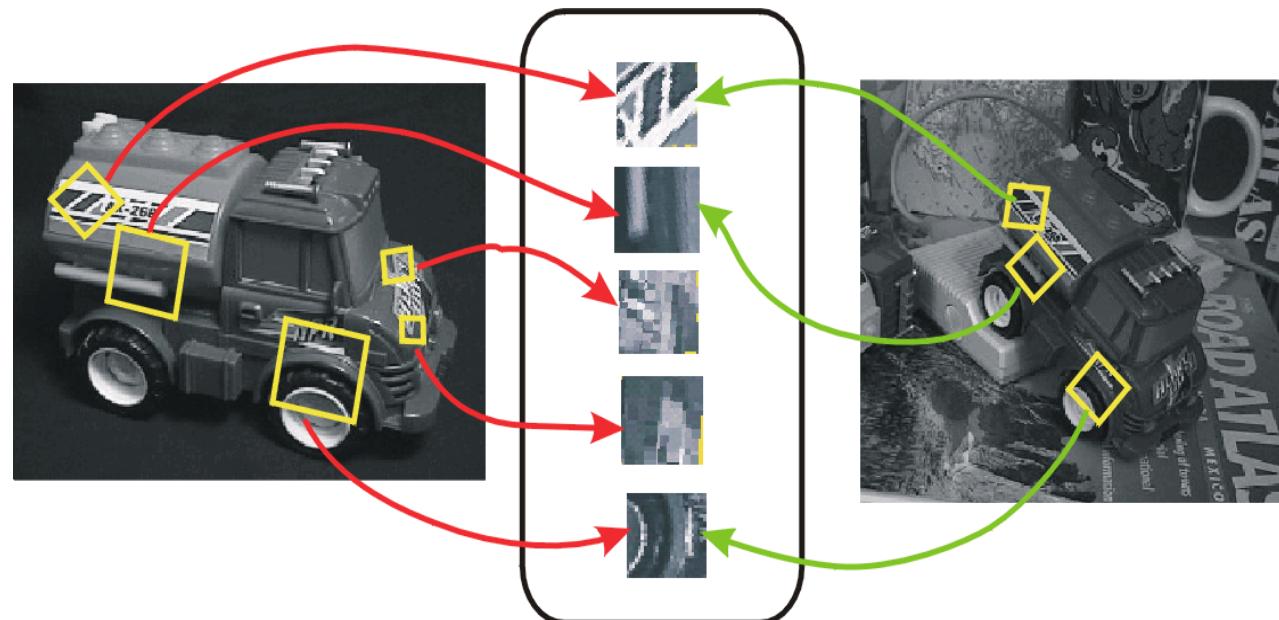


Figure: David Lowe

Main questions

- Where will the interest points come from?
 - What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

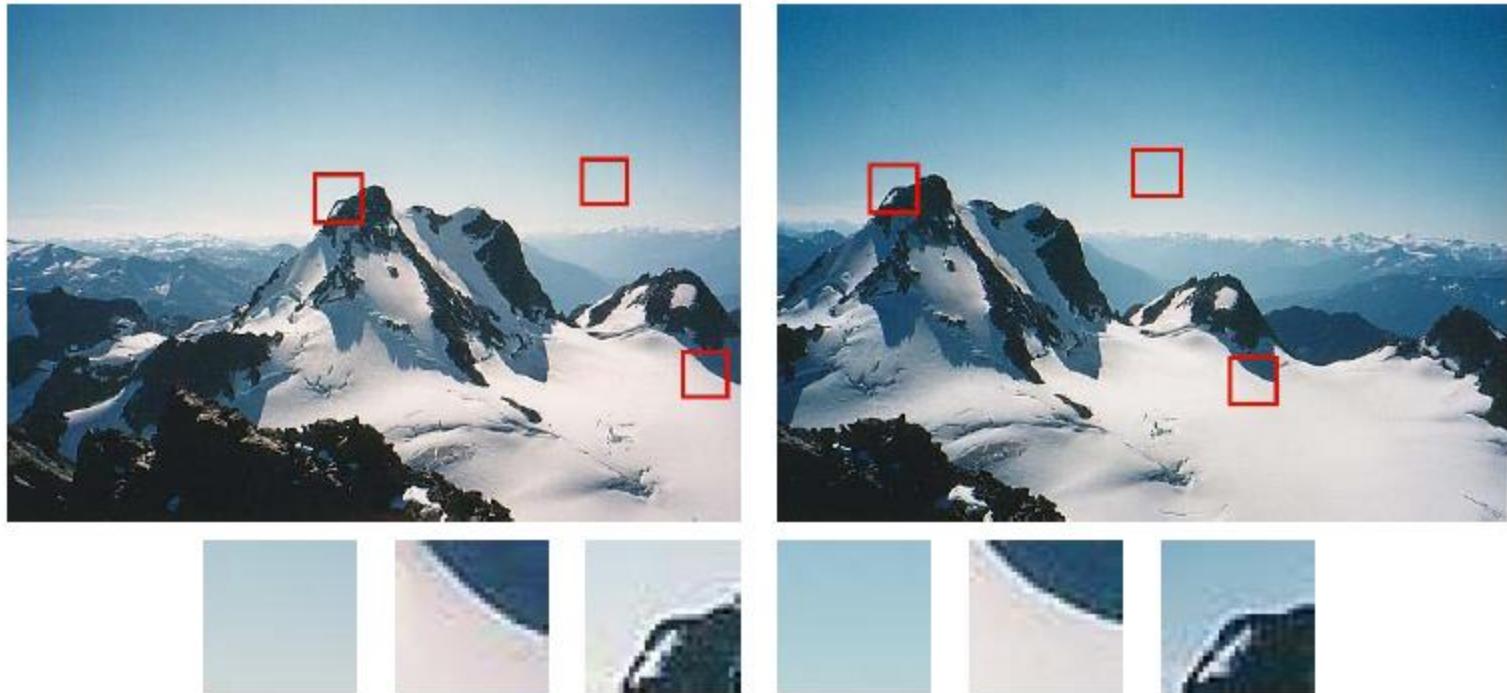
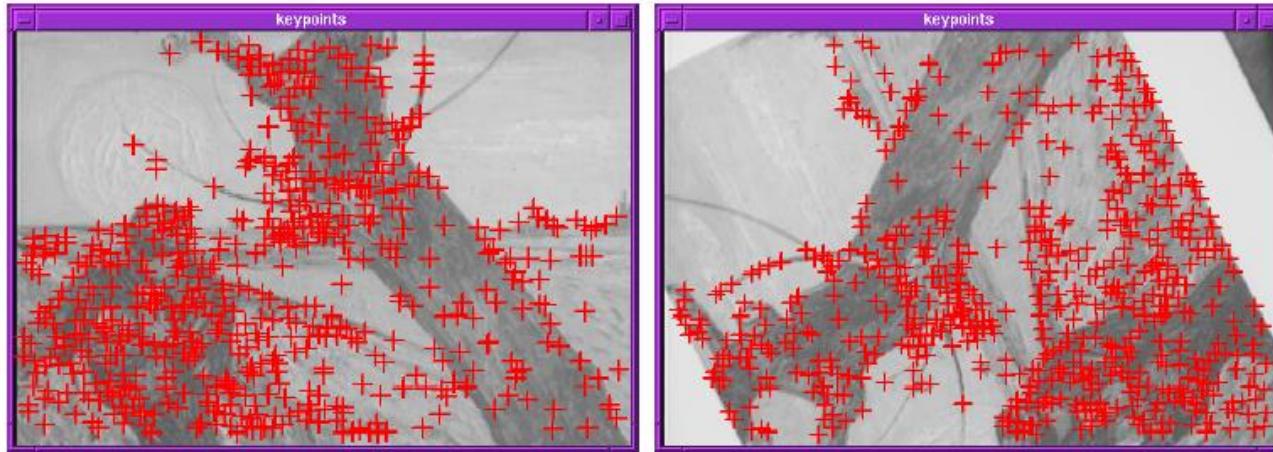


Figure 4.3: *Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.*

Finding Corners

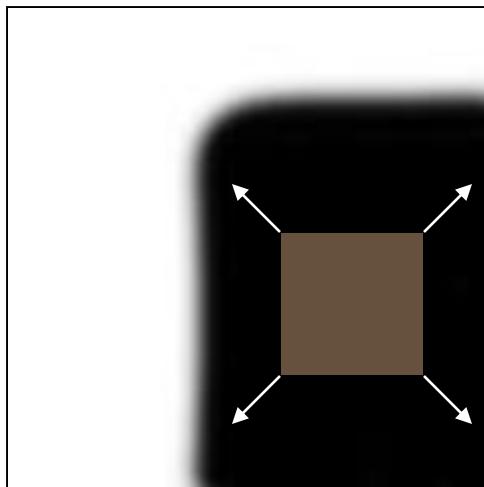


- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and **distinctive**

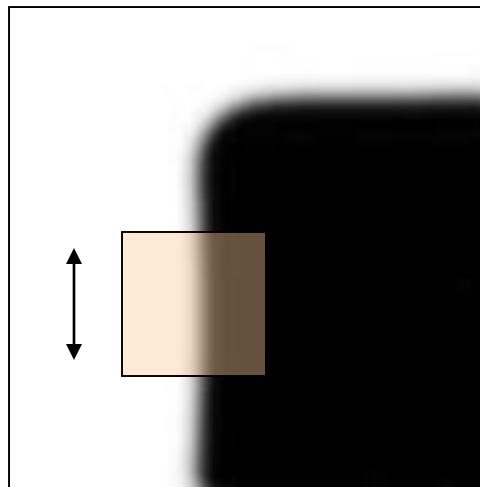
C.Harris and M.Stephens. "[A Combined Corner and Edge Detector.](#)"
Proceedings of the 4th Alvey Vision Conference: pages 147--151.

Corners as distinctive interest points

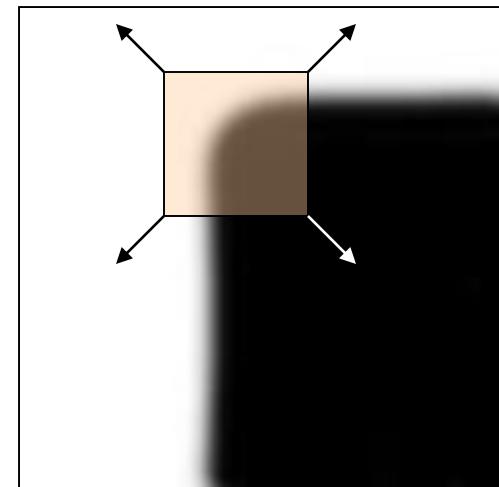
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



“flat” region:
no change in
all directions



“edge”:
no change
along the edge
direction



“corner”:
significant
change in all
directions

Harris Detector formulation

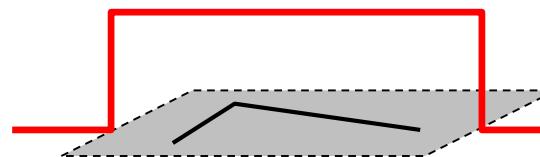
Change of intensity for the shift $[u, v]$:

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

Diagram illustrating the components of the Harris Detector formulation:

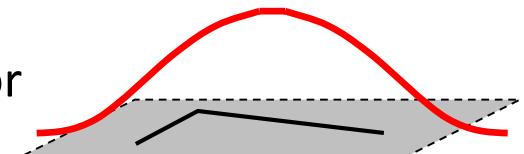
- Window function: A green oval pointing to the term $w(x, y)$.
- Shifted intensity: A green oval pointing to the term $I(x + u, y + v)$.
- Intensity: A green oval pointing to the term $I(x, y)$.

Window function $w(x, y) =$



1 in window, 0 outside

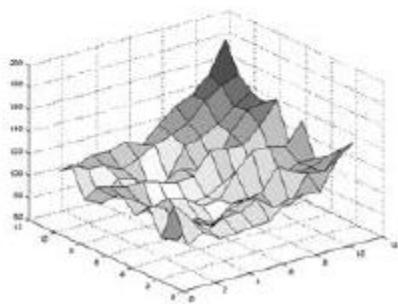
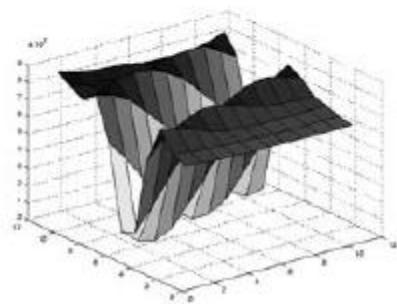
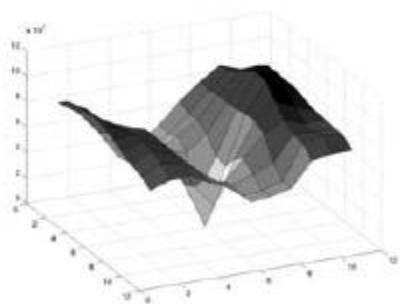
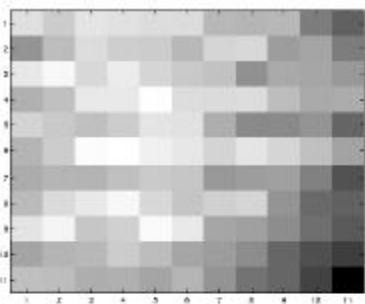
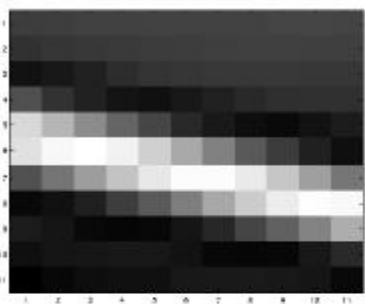
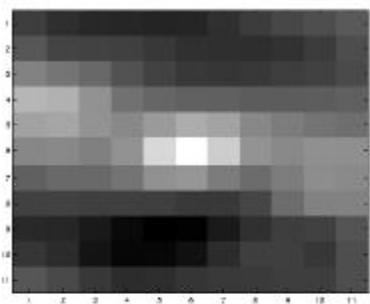
or



Gaussian



(a)



Harris Detector formulation

This measure of change can be approximated by:

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

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

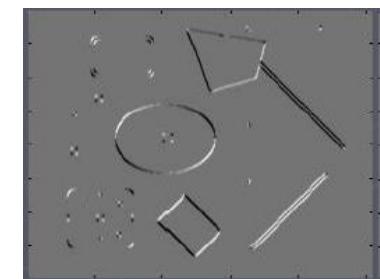
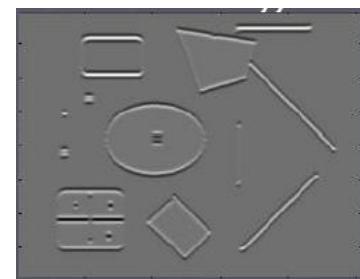
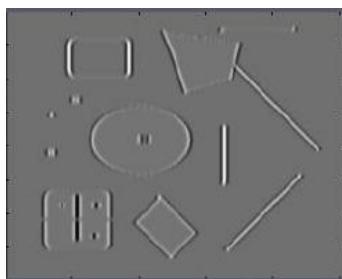
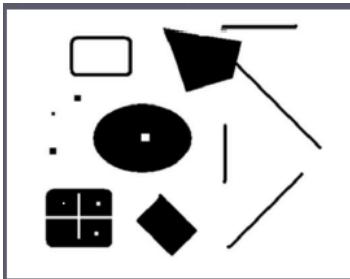
$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Sum over image region – area
we are checking for corner

Gradient with
respect to x, times
gradient with
respect to y

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y]$$

Harris Detector formulation



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

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

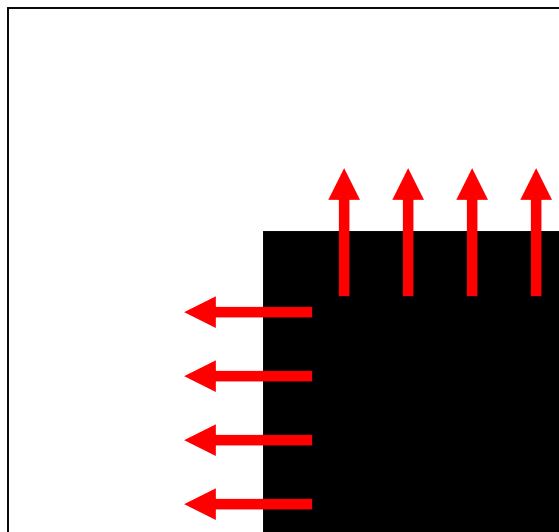
Sum over image region – area
we are checking for corner

Gradient with
respect to x, times
gradient with
respect to y

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y]$$

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

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

This means dominant gradient directions align with x or y axis

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

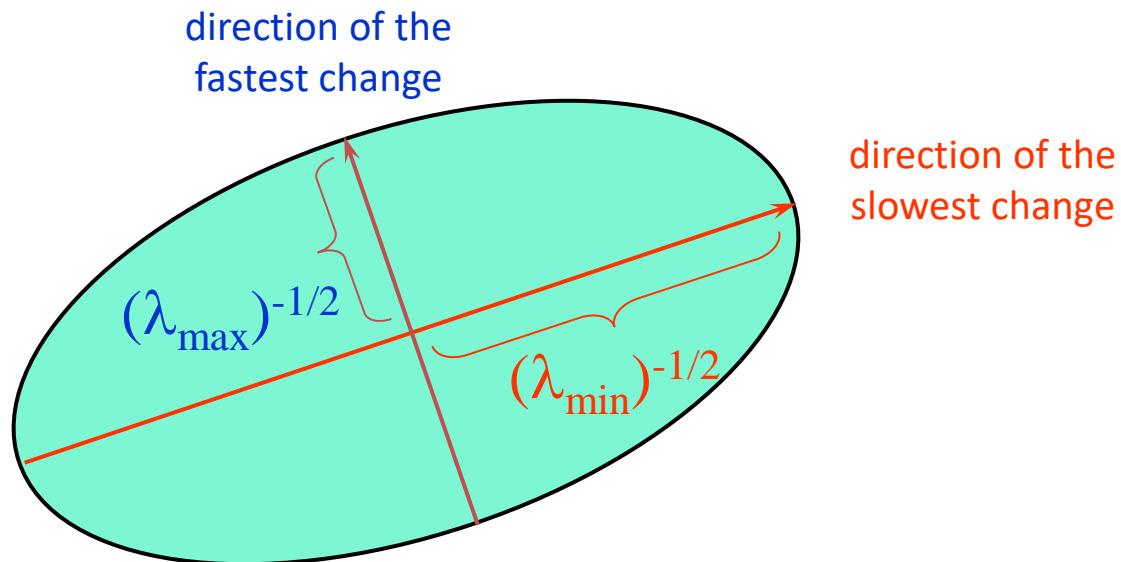
What if we have a corner that is not aligned with the image axes?

General Case

Since M is symmetric, we have

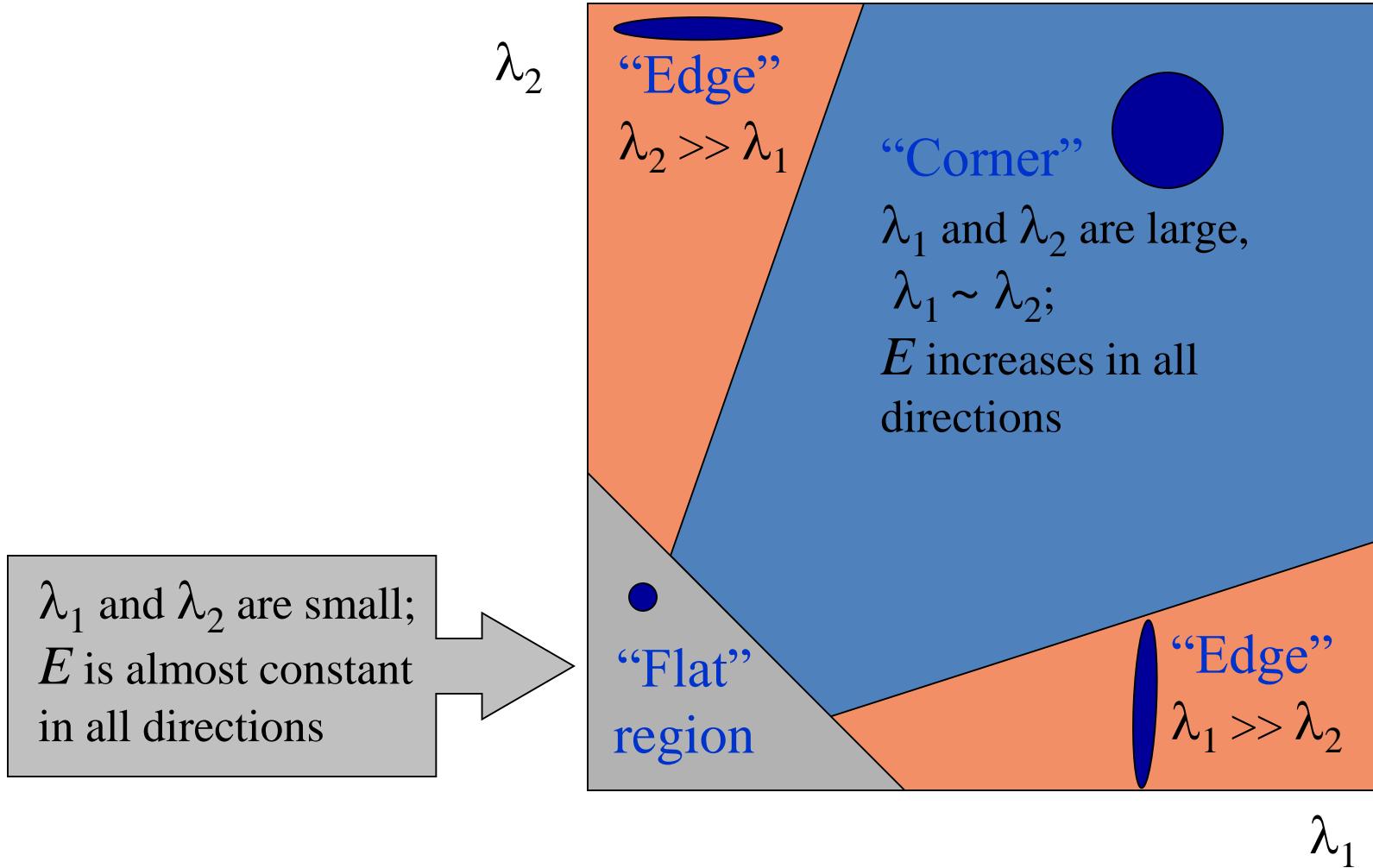
$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

We can visualize M as an ellipse with axis lengths determined by the eigenvalues and orientation determined by R



Interpreting the eigenvalues

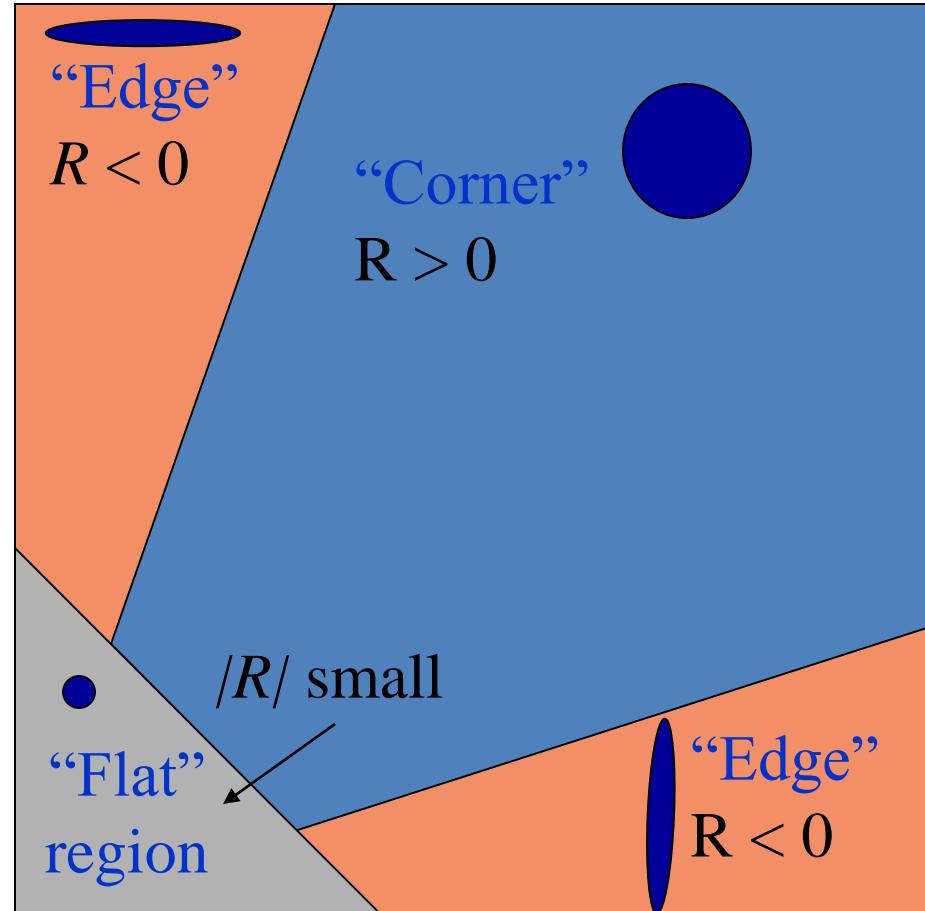
Classification of image points using eigenvalues of M :



Corner response function

$$R = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α : constant (0.04 to 0.06)



Harris Corner Detector

- Algorithm steps:
 - Compute M matrix within all image windows to get their R scores
 - Find points with large corner response ($R > \text{threshold}$)
 - Take the points of local maxima of R

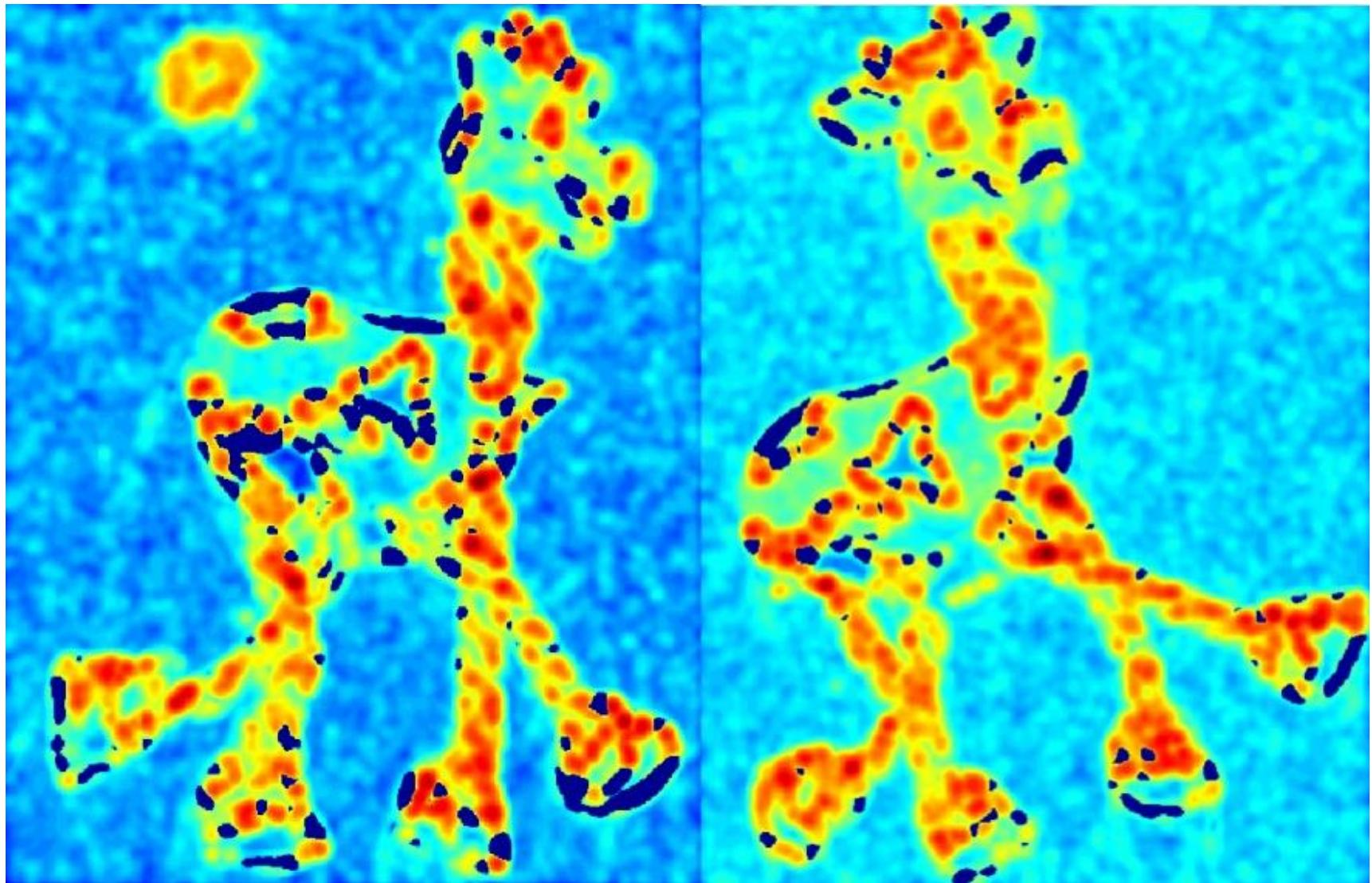
Harris Detector: Workflow



Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

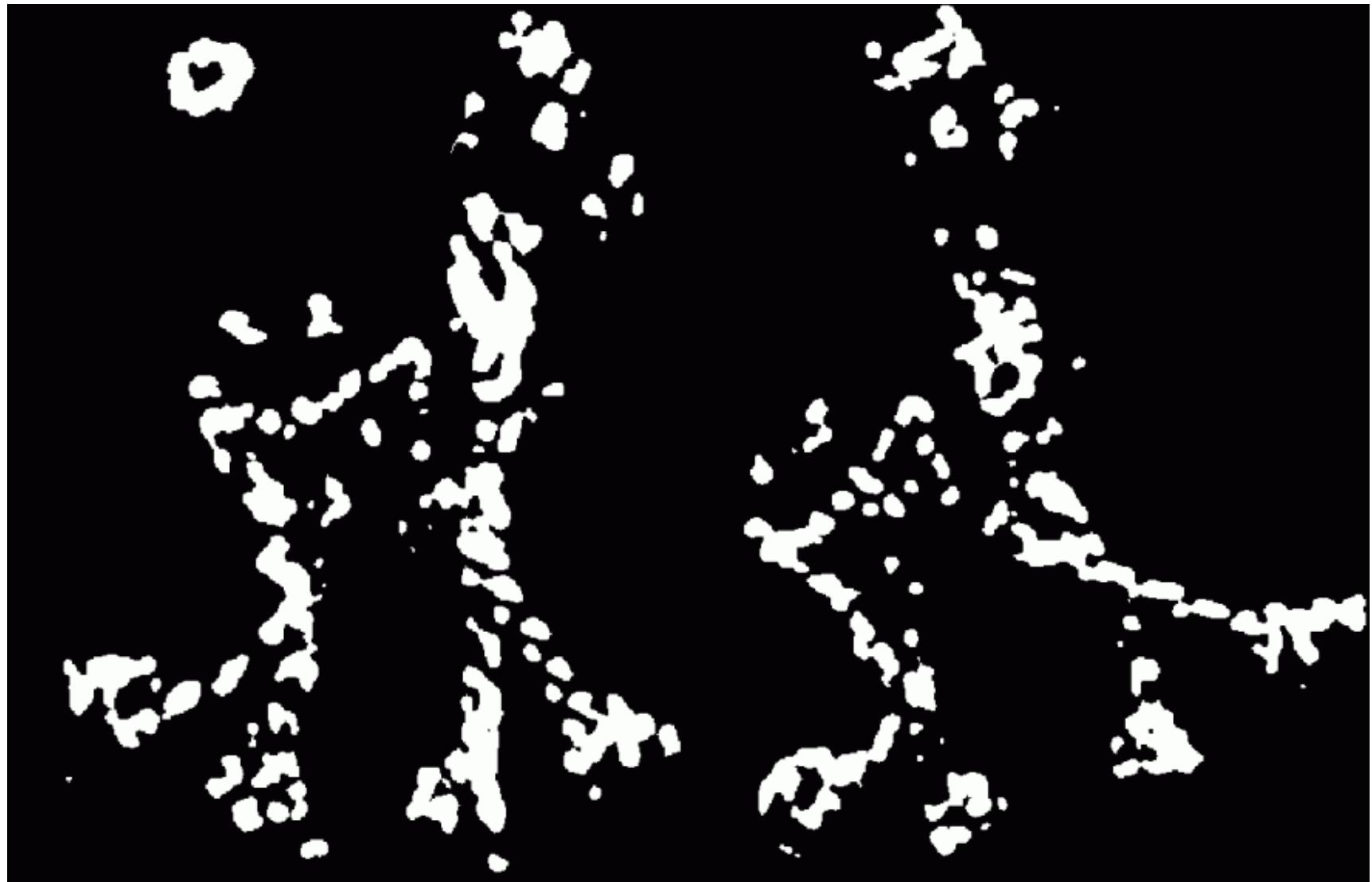
Harris Detector: Workflow

Compute corner response R



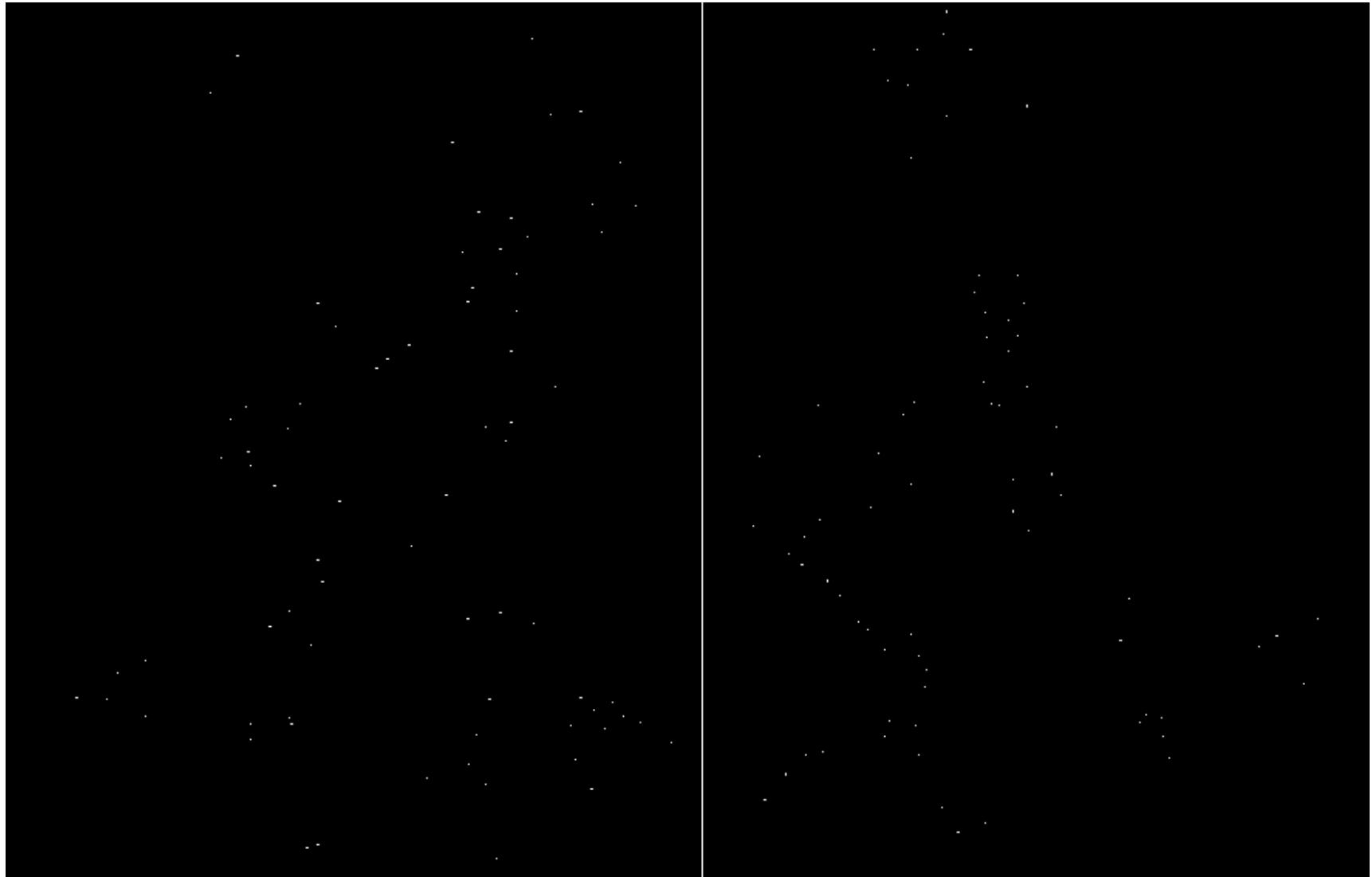
Harris Detector: Workflow

Find points with large corner response: $R > \text{threshold}$



Harris Detector: Workflow

Take only the points of local maxima of R

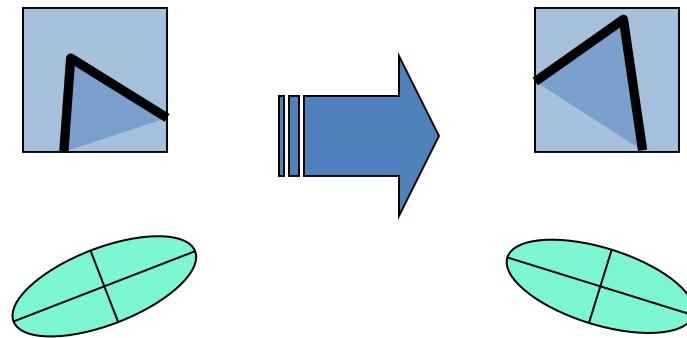


Harris Detector: Workflow



Harris Detector: Properties

- Rotation invariance

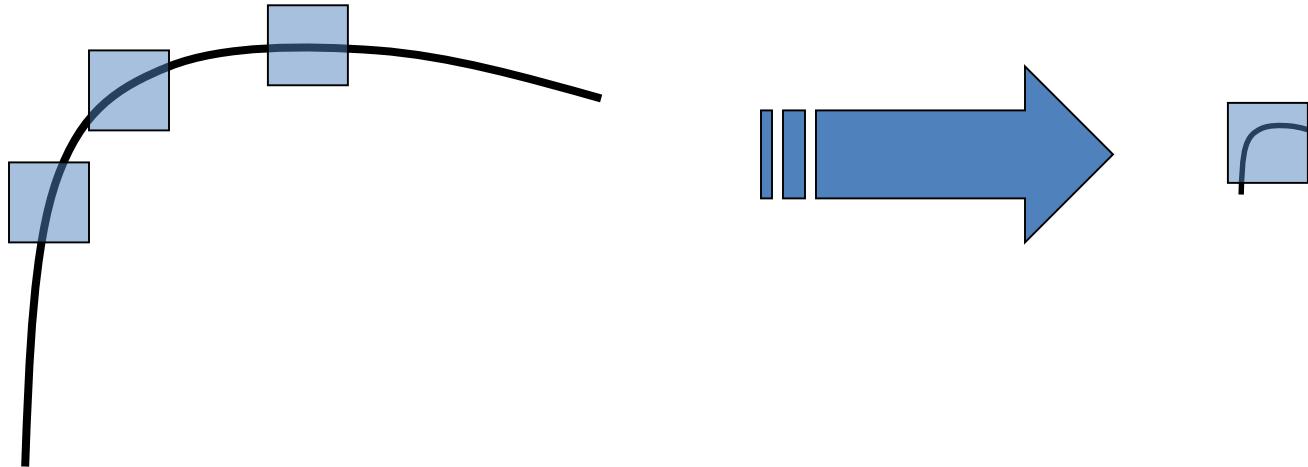


Ellipse rotates but its shape (i.e. eigenvalues)
remains the same

Corner response R is invariant to image rotation

Harris Detector: Properties

- Not invariant to image scale



All points will be
classified as **edges**

Corner !

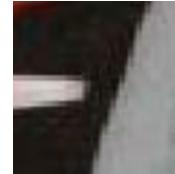
- How can we detect **scale invariant** interest points?

How to cope with transformations?

- Exhaustive search
- Invariance
- Robustness

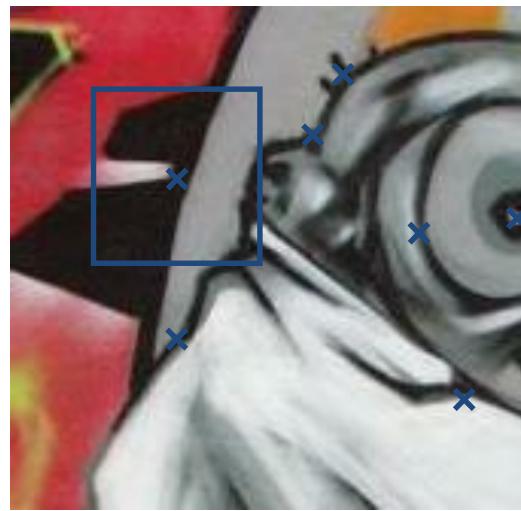
Exhaustive search

- Multi-scale approach



Exhaustive search

- Multi-scale approach



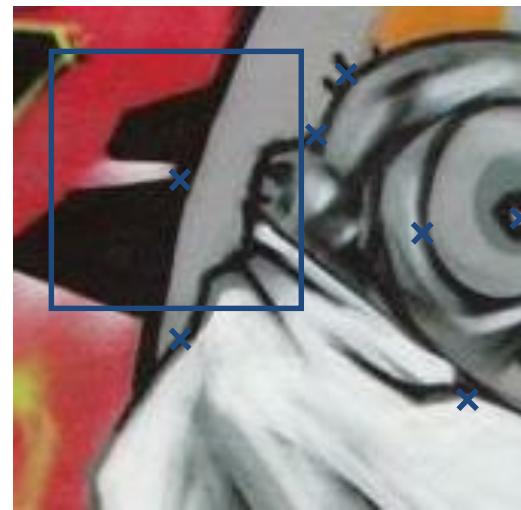
Exhaustive search

- Multi-scale approach



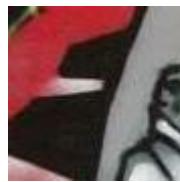
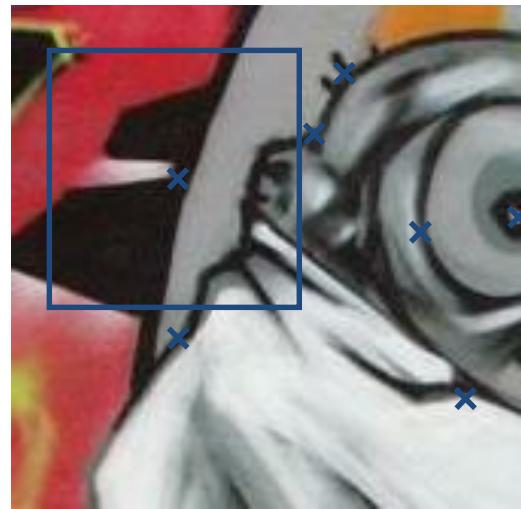
Exhaustive search

- Multi-scale approach



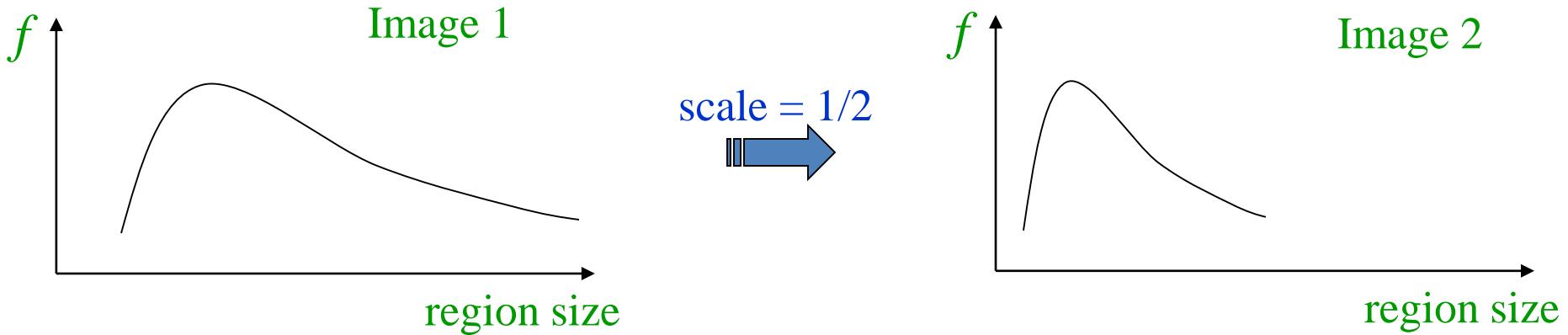
Invariance

- Extract patch from each image individually



Automatic scale selection

- Solution:
 - Design a function on the region, which is “scale invariant” (*the same for corresponding regions, even if they are at different scales*)
Example: average intensity. For corresponding regions (even of different sizes) it will be the same.
 - For a point in one image, we can consider it as a function of region size (patch width)



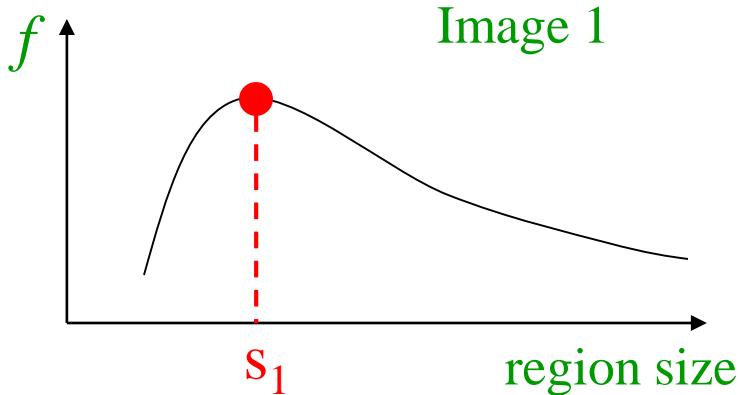
Automatic scale selection

- Common approach:

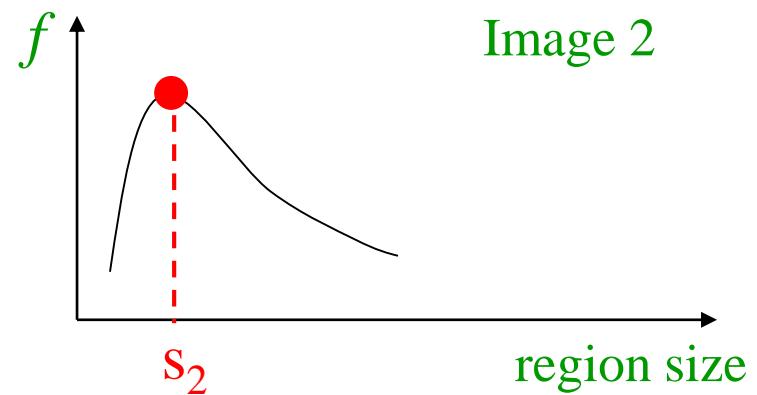
Take a local maximum of this function

Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

Important: this scale invariant region size is found in each image independently!



scale = 1/2
→



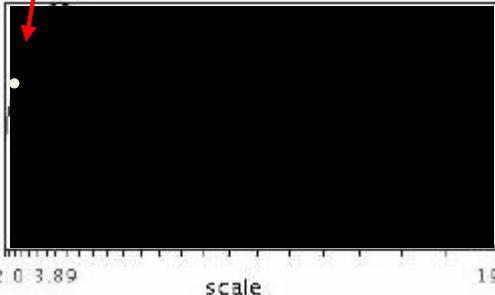
Automatic Scale Selection

- Find the best scale for increasing contrast

S

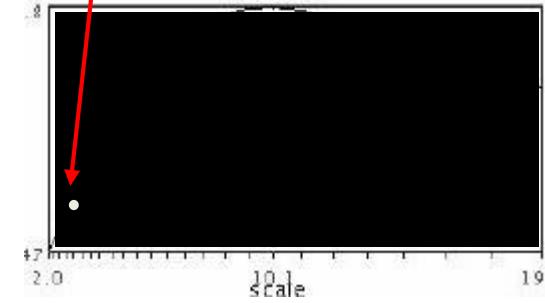


1.1



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

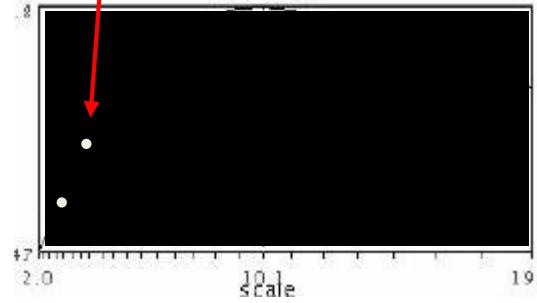
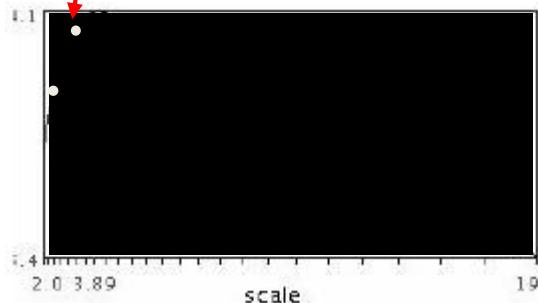
8



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

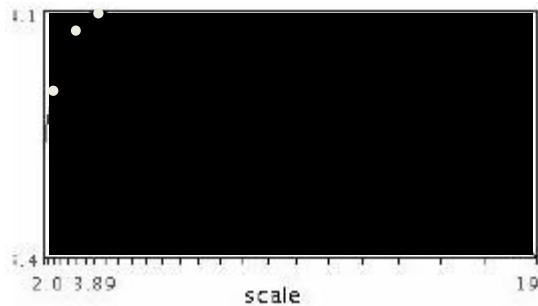
Automatic Scale Selection

- For color images

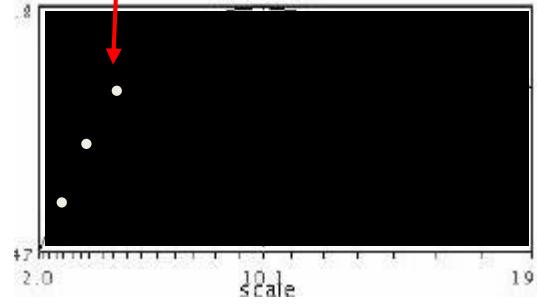


Automatic Scale Selection

- For color images



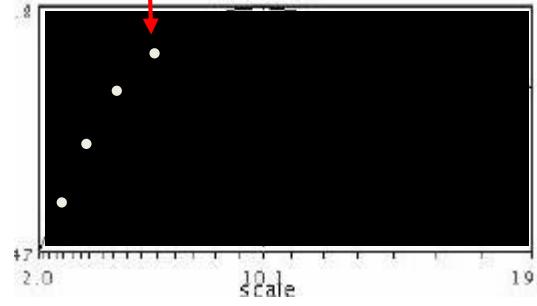
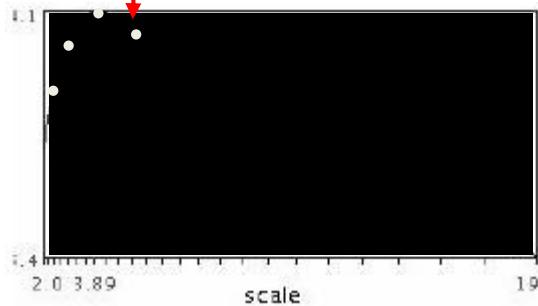
$$f(I_{i_1\dots i_m}(x, \sigma))$$



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

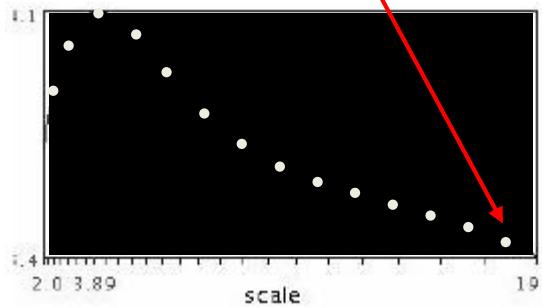
Automatic Scale Selection

- For each color in the image, select a scale

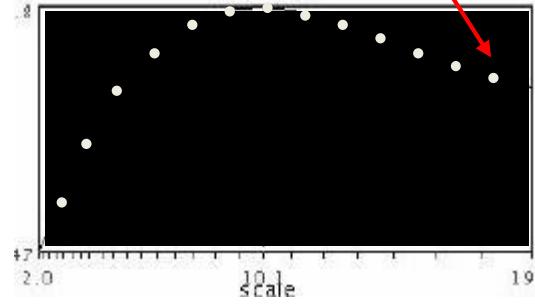


Automatic Scale Selection

- For color images, automatic scale selection



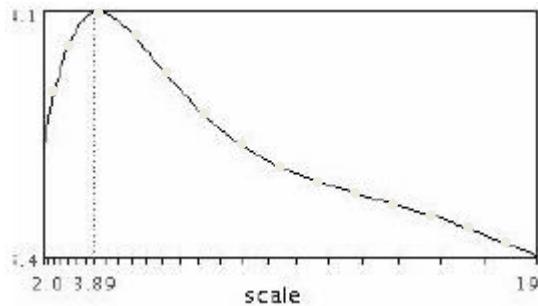
$$f(I_{i_1 \dots i_m}(x, \sigma))$$



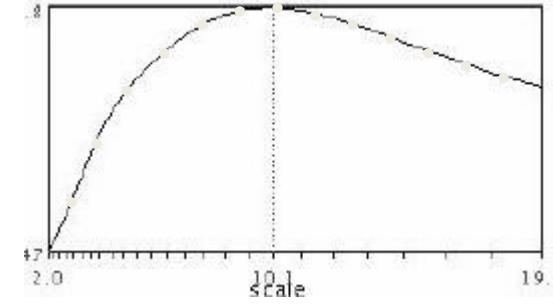
$$f(I_{i_1 \dots i_m}(x', \sigma))$$

Automatic Scale Selection

- For color images, automatic scale selection



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

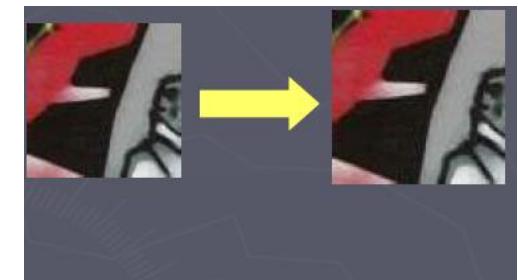
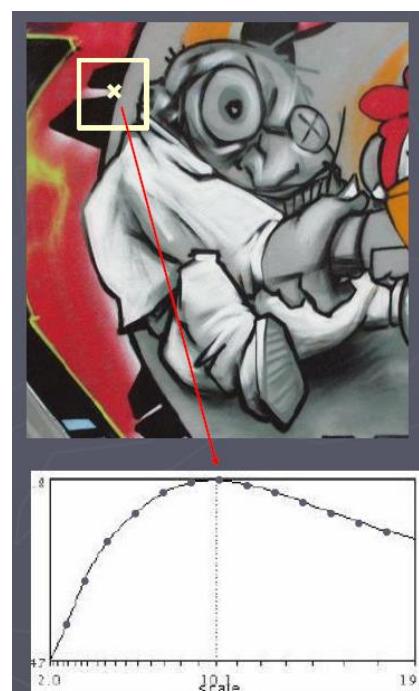
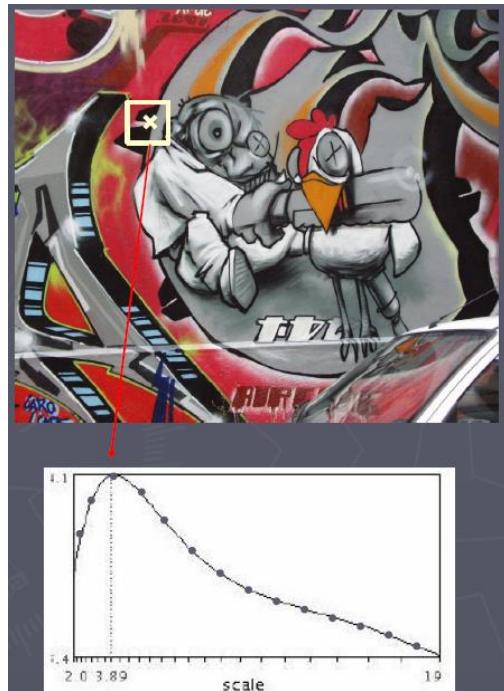


$$f(I_{i_1 \dots i_m}(x', \sigma'))$$

K. Grauman, B. Leibe
K. Grauman, B. Leibe

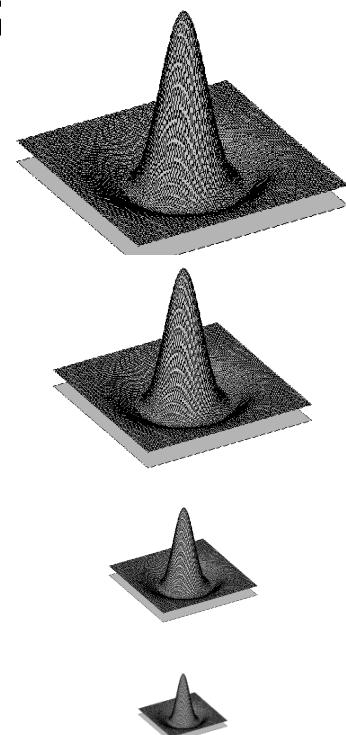
Scale selection

- Use the scale determined by detector to compute descriptor in a normalized frame

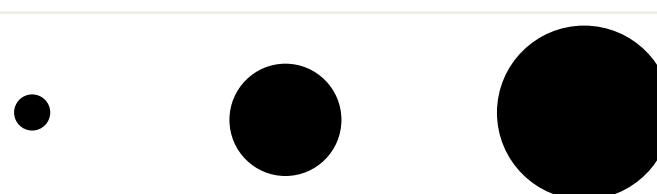
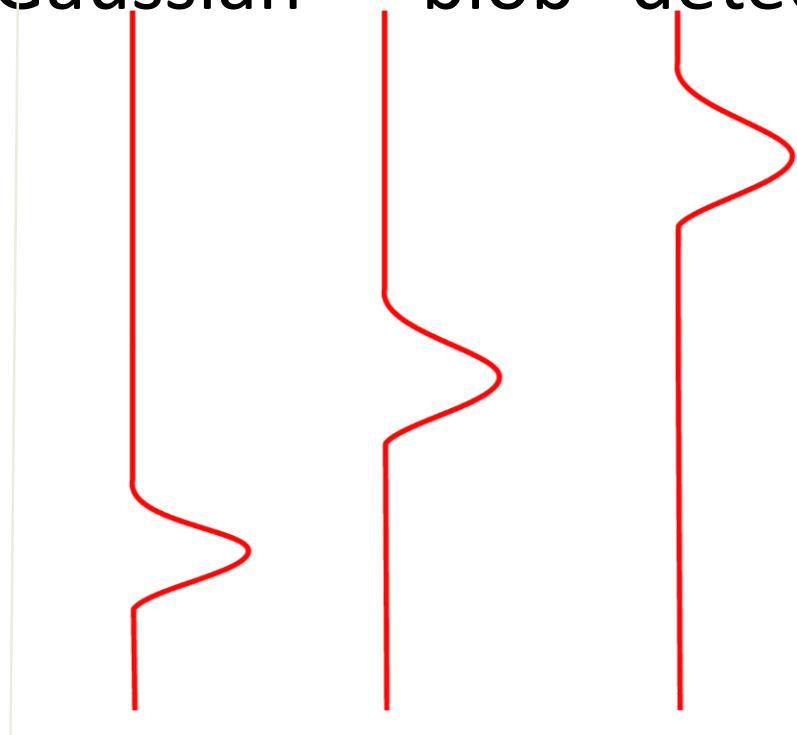


What Is A Useful Signature Function?

- Let's look at the f-Gaussian = “blob” detector

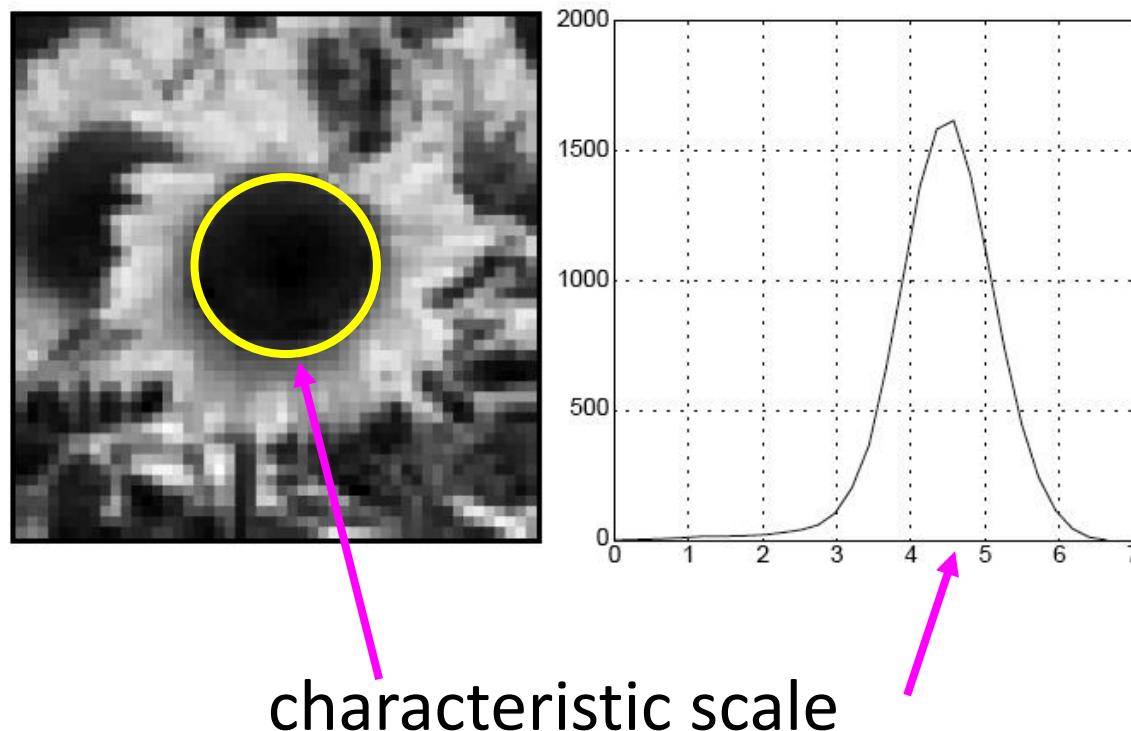


f-Gaussian = “blob” detector



Characteristic scale

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



T. Lindeberg (1998). ["Feature detection with automatic scale selection."](#)

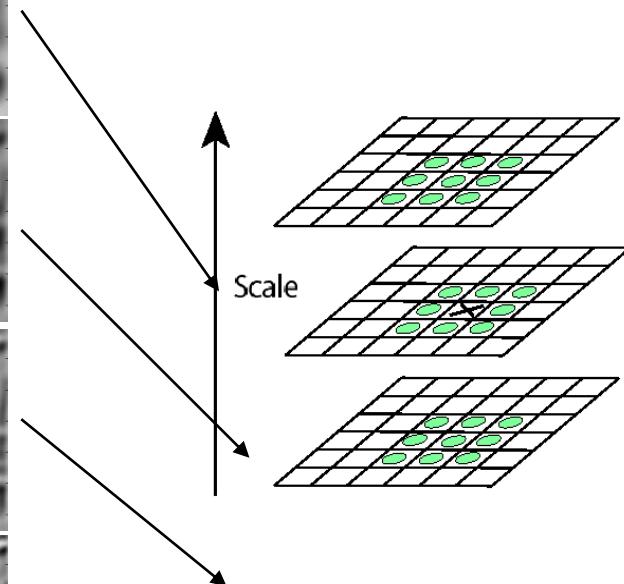
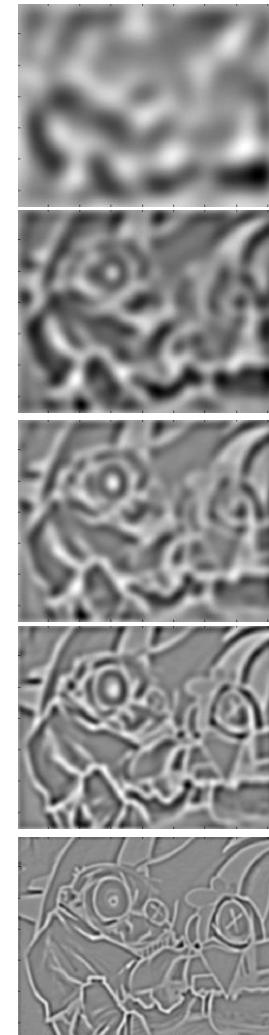
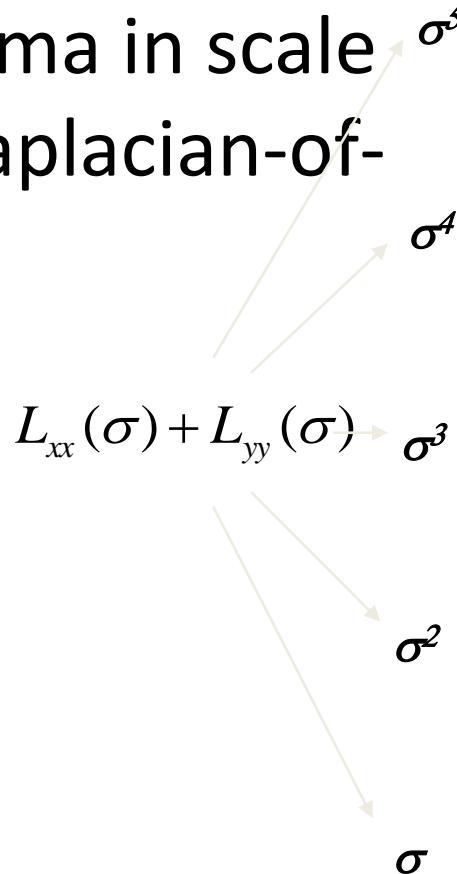
International Journal of Computer Vision **30** (2): pp 77--116.

Source: Lana Lazebnik

Laplacian-of-Gaussian (LoG)

- Interest points:

Local maxima in scale
space of Laplacian-of-
Gaussian



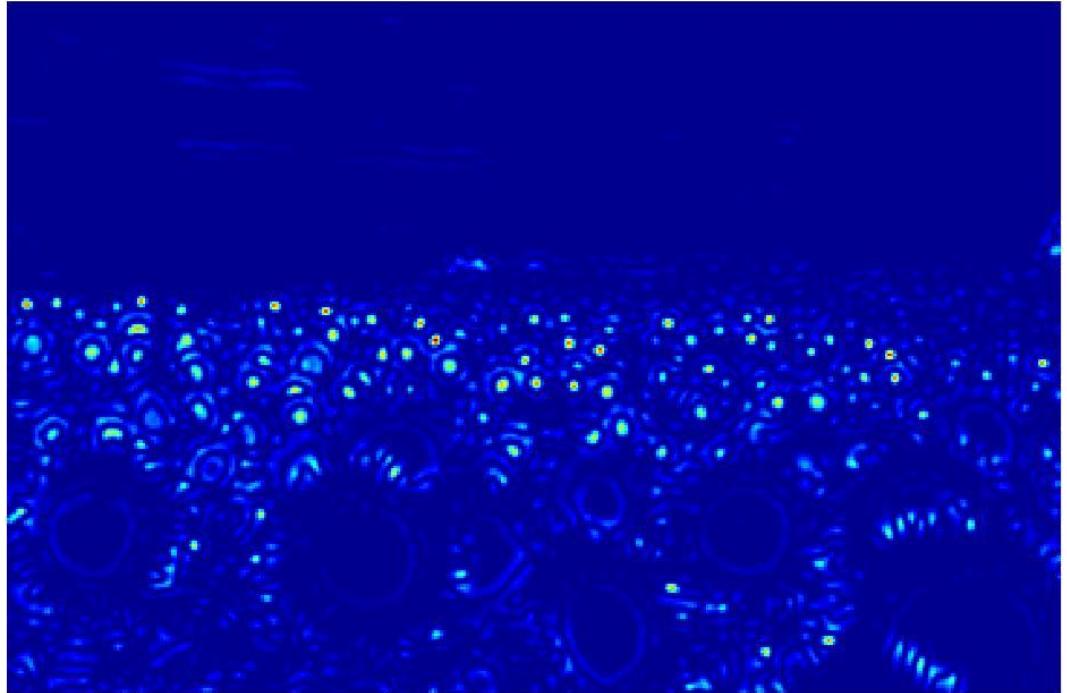
\Rightarrow List of
(x, y, σ)

Example

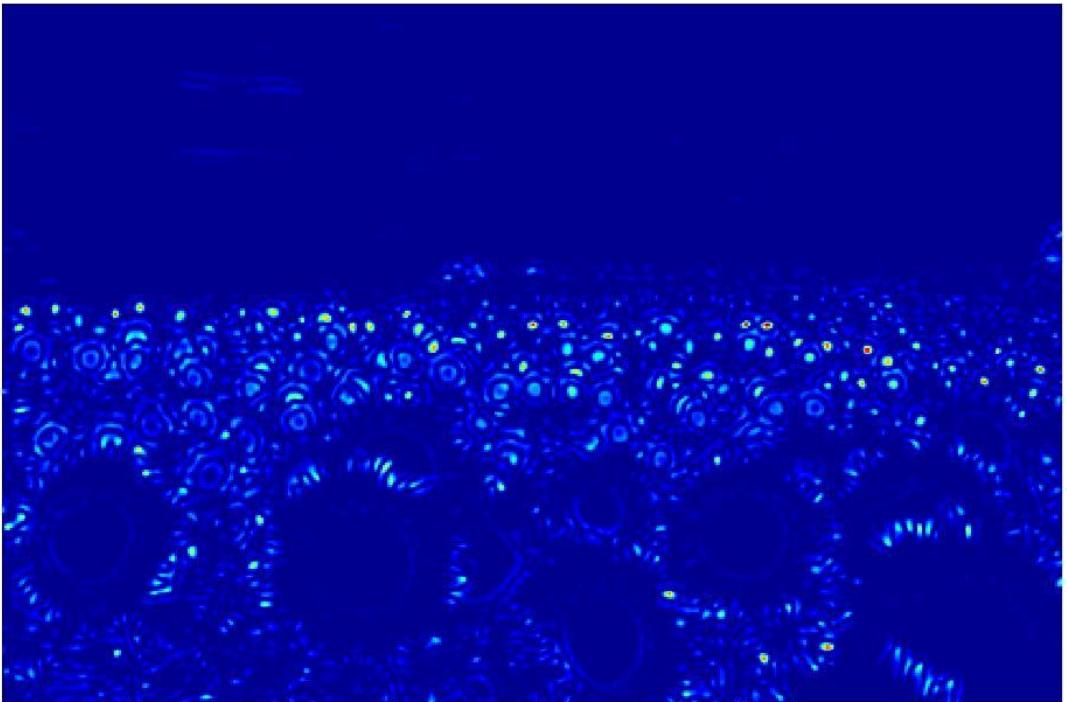
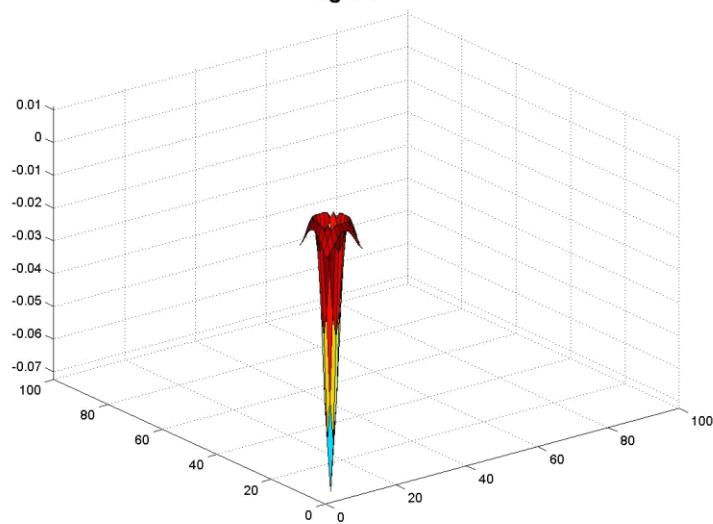
Original image at
 $\frac{3}{4}$ the size

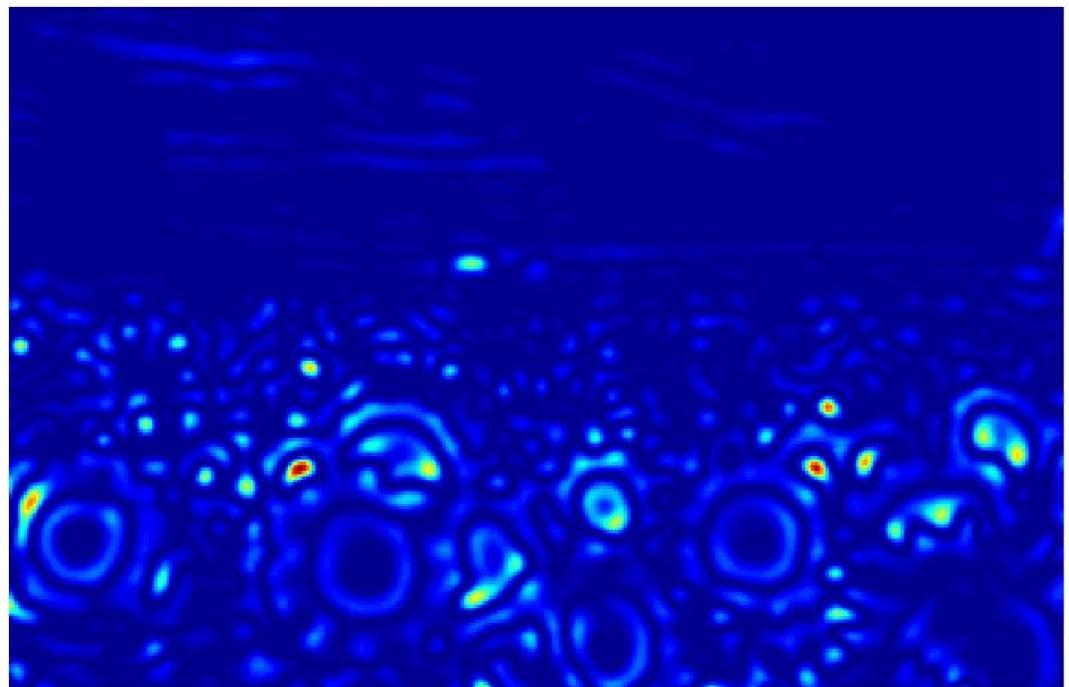


Original image at
¾ the size

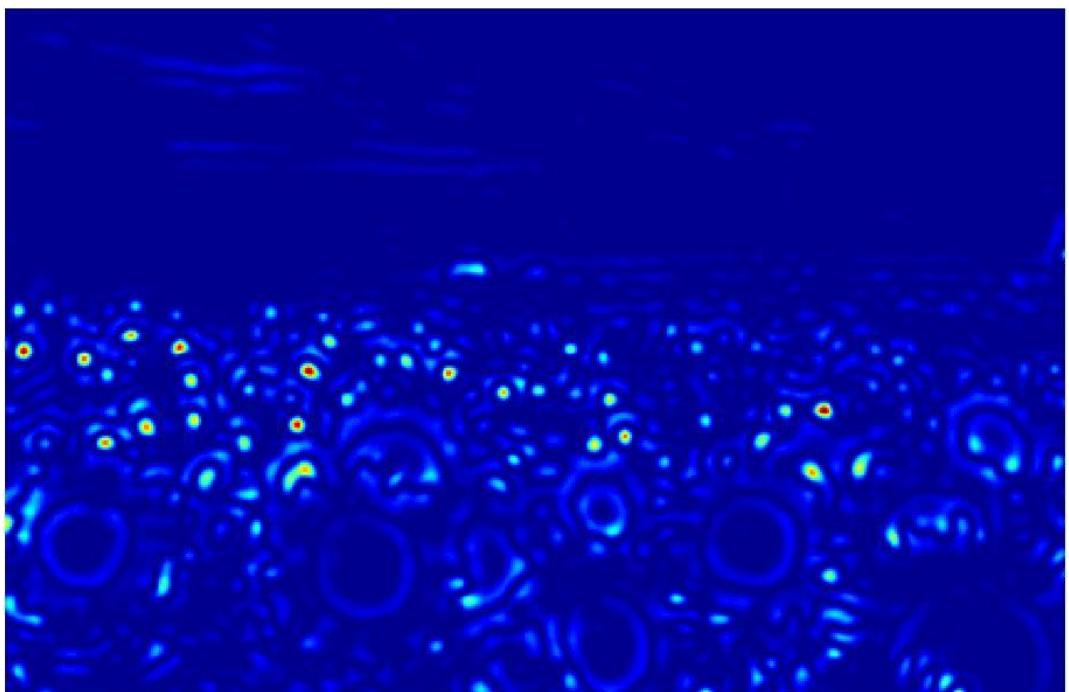
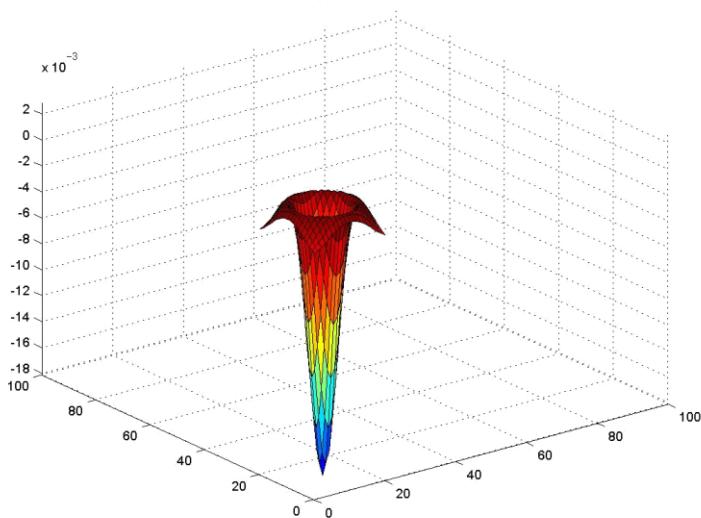


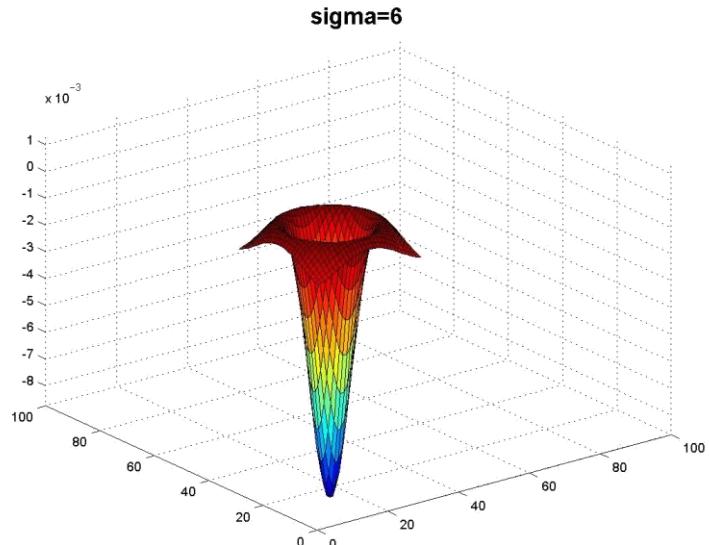
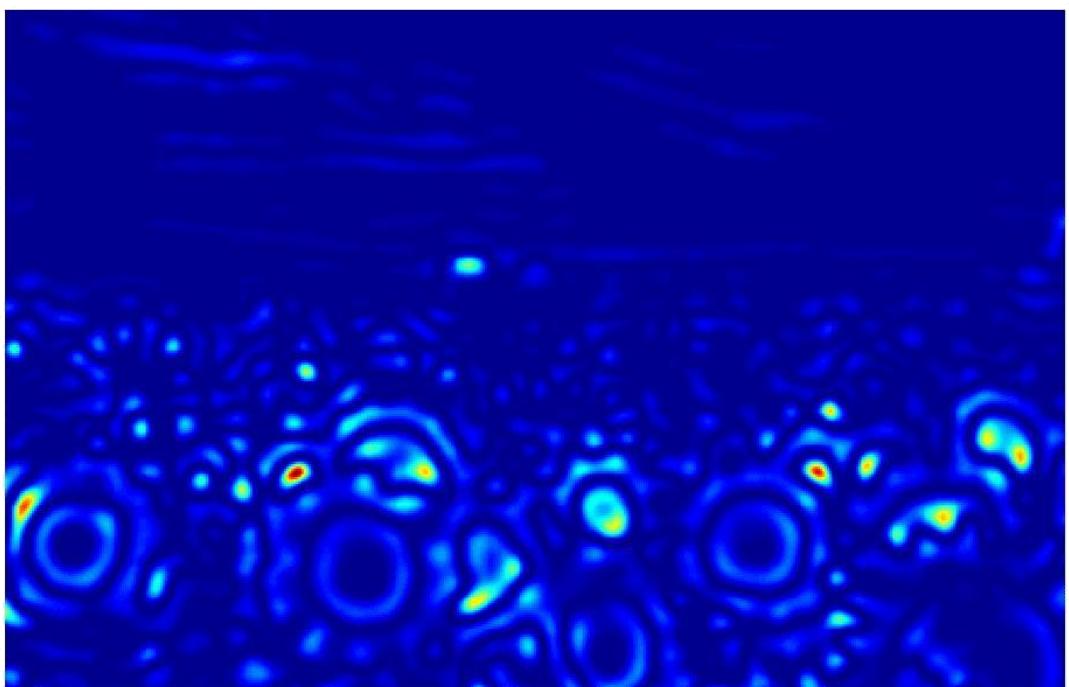
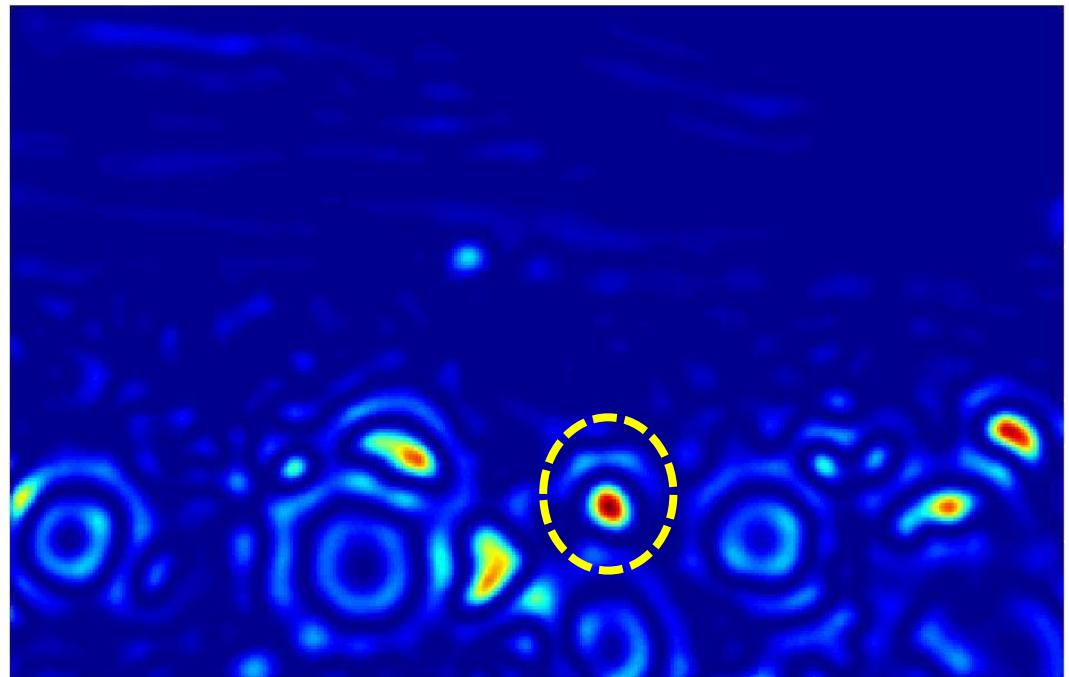
sigma=2.1

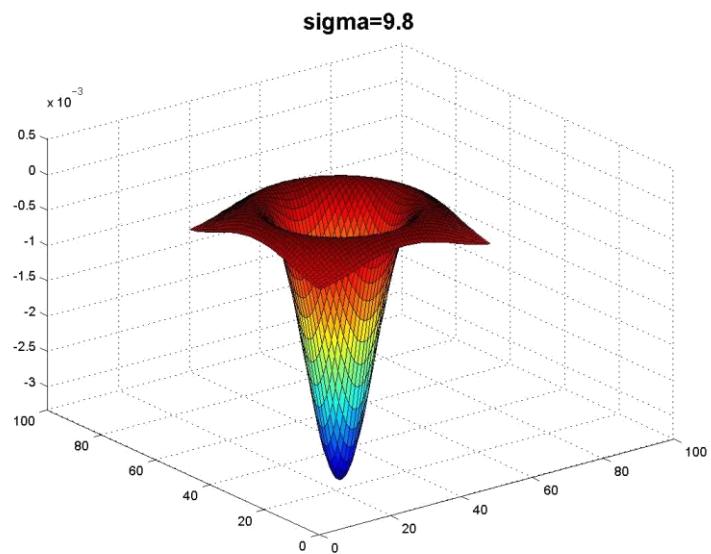
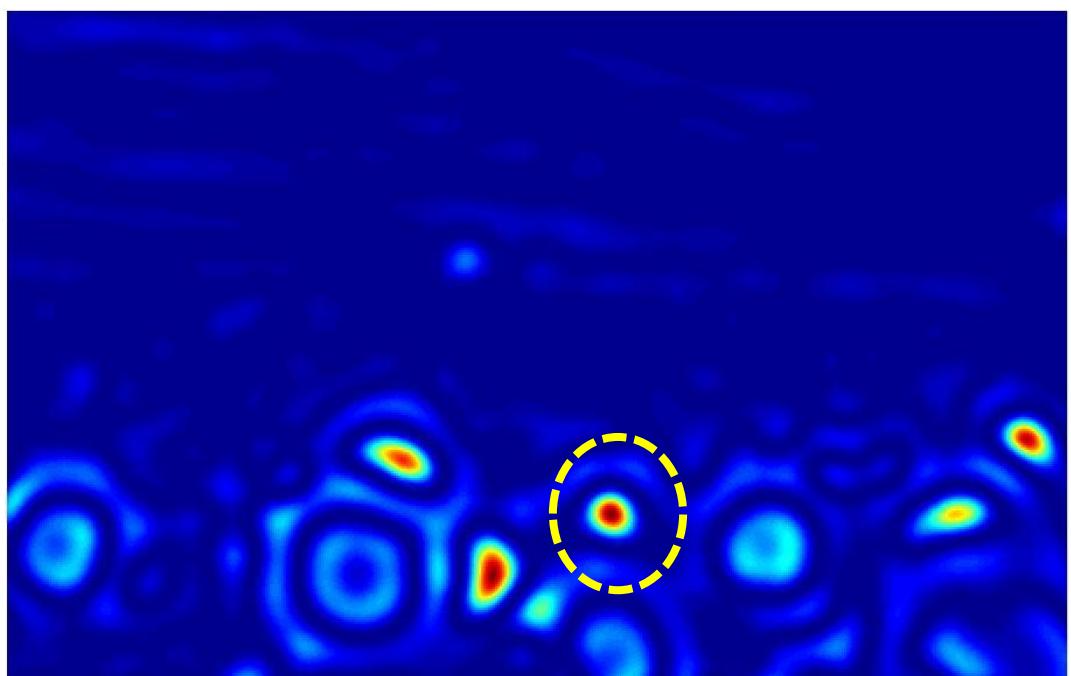
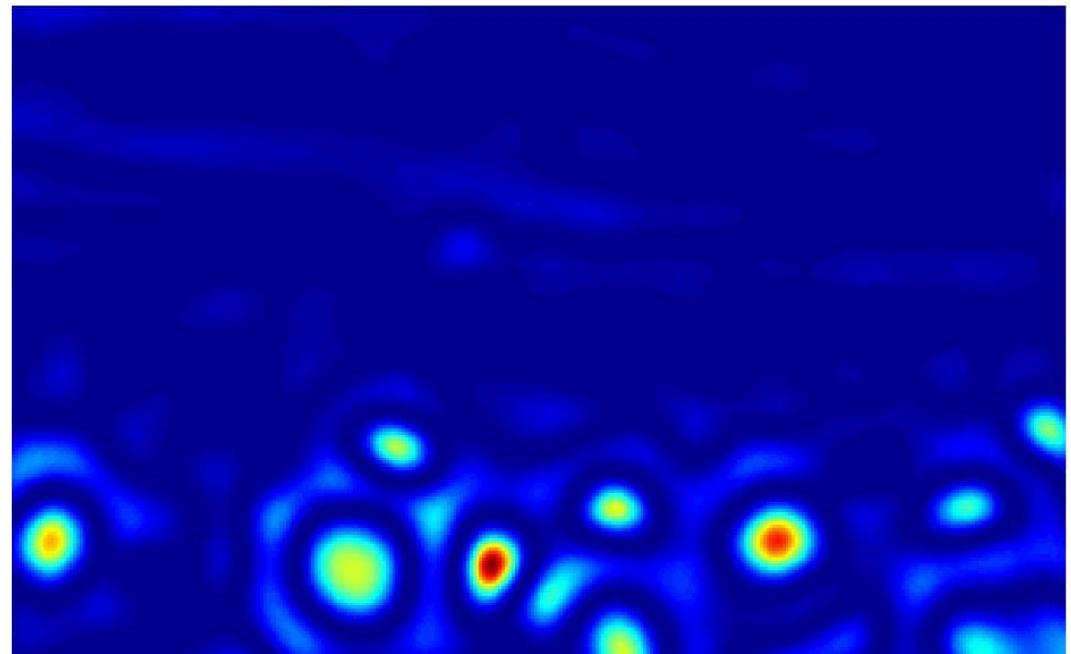


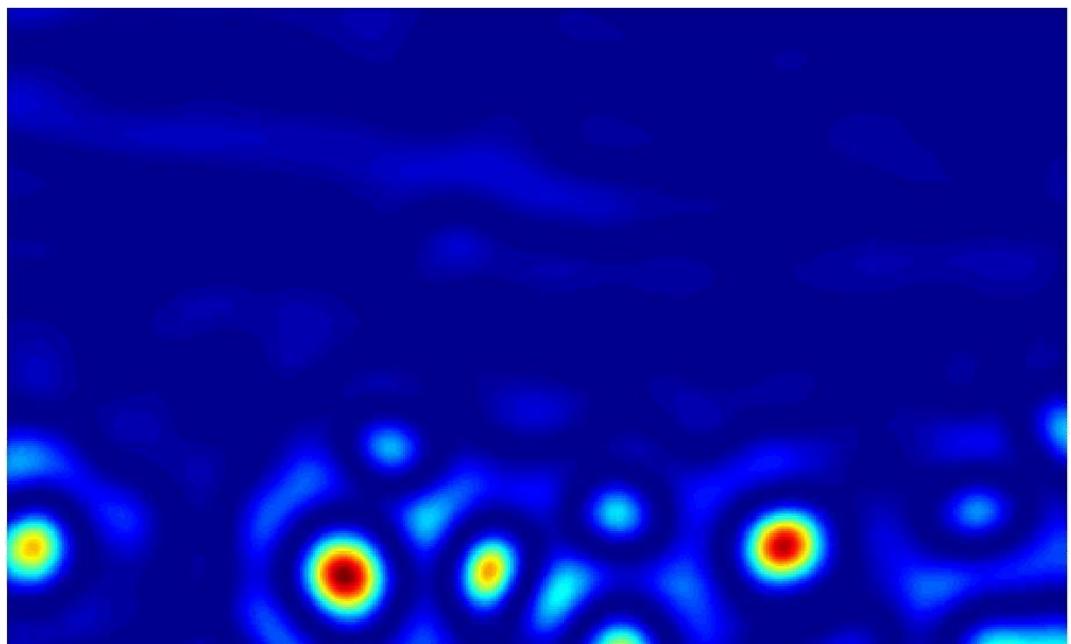
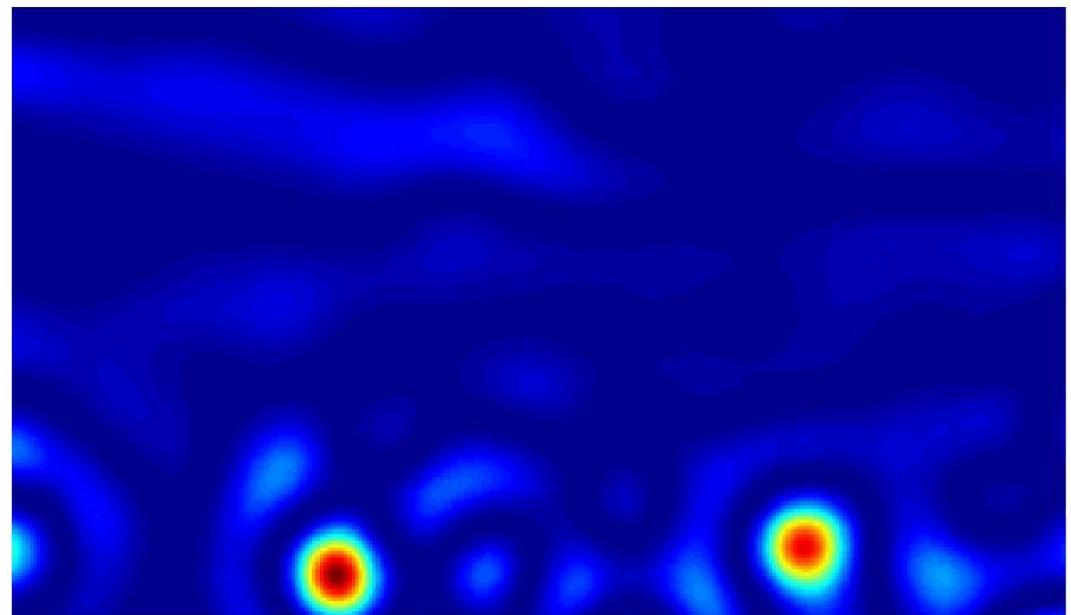


sigma=4.2

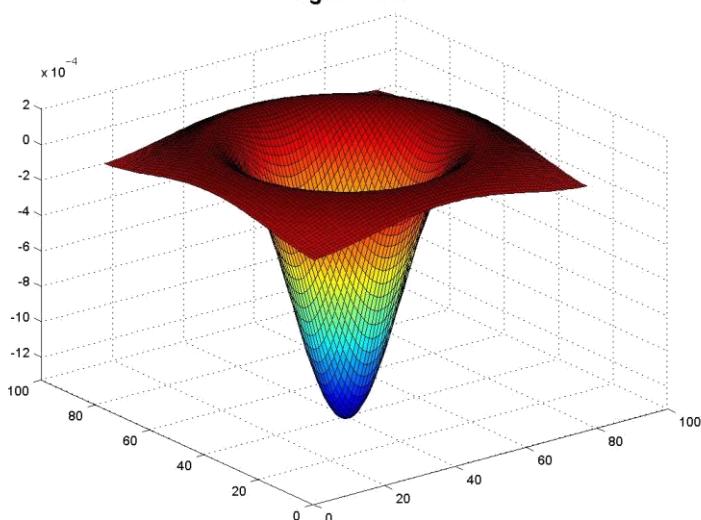








sigma=15.5



Scale invariant interest points

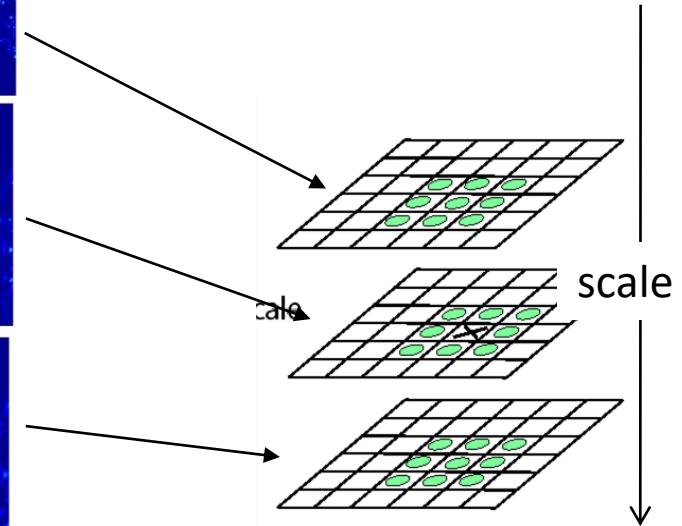
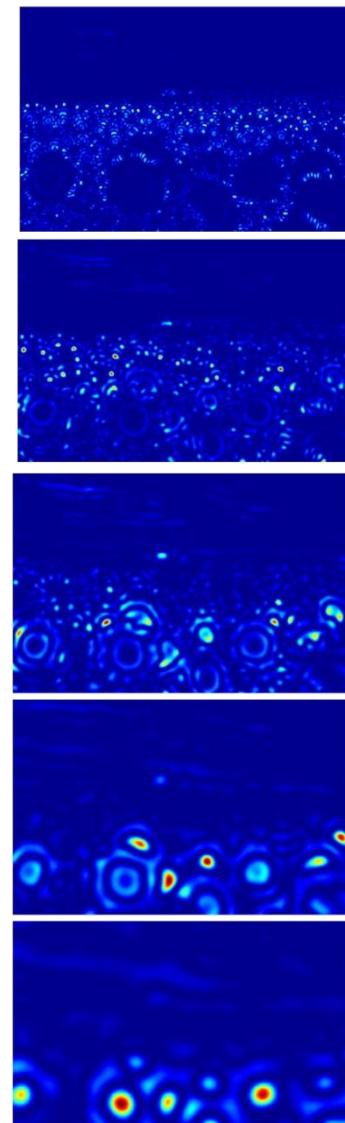
Interest points are local maxima in both position and scale.



$$L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow \sigma_3$$

Squared filter response maps

σ_1
 σ_2
 σ_3
 σ_4
 σ_5



⇒ List of
 (x, y, σ)

Scale-space blob detector: Example

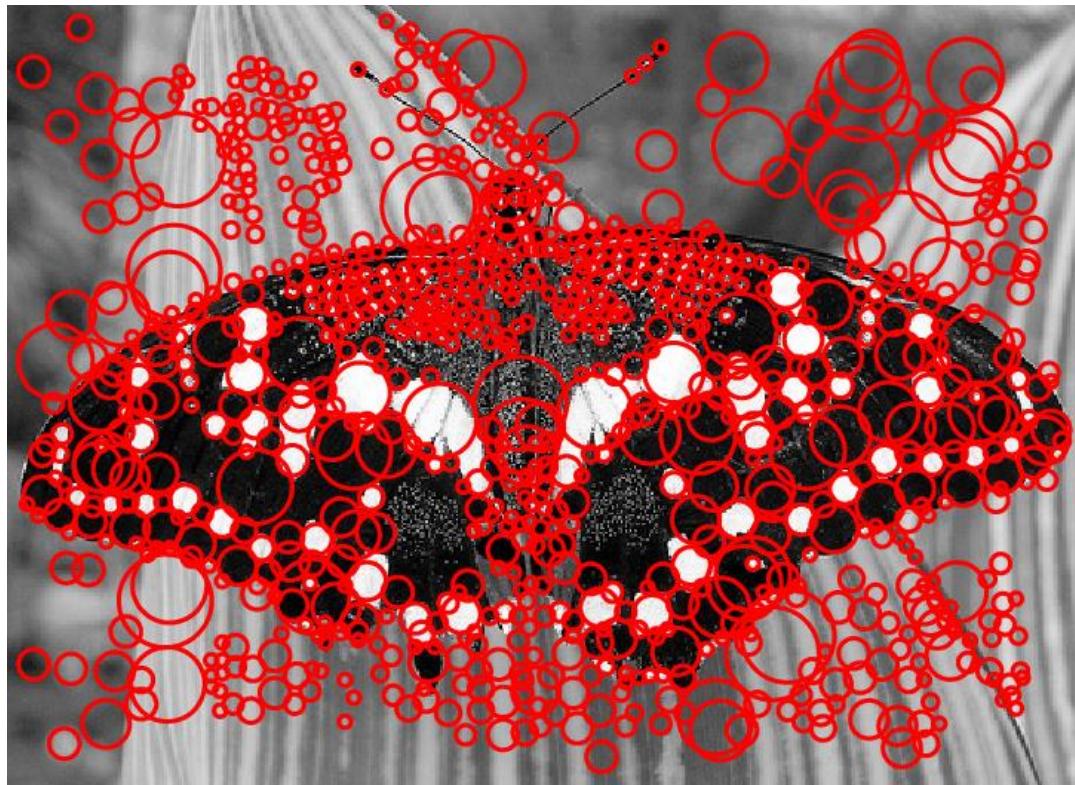


Scale-space blob detector: Example

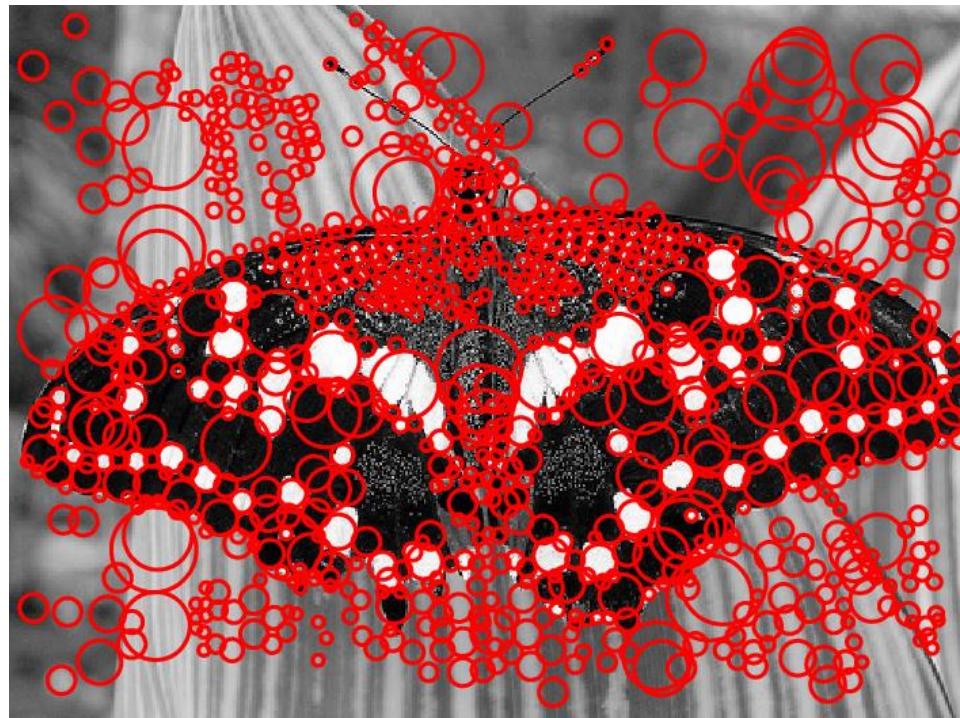


$\sigma = 11.9912$

Scale-space blob detector: Example



Scale-space blob detector: Example



Technical detail

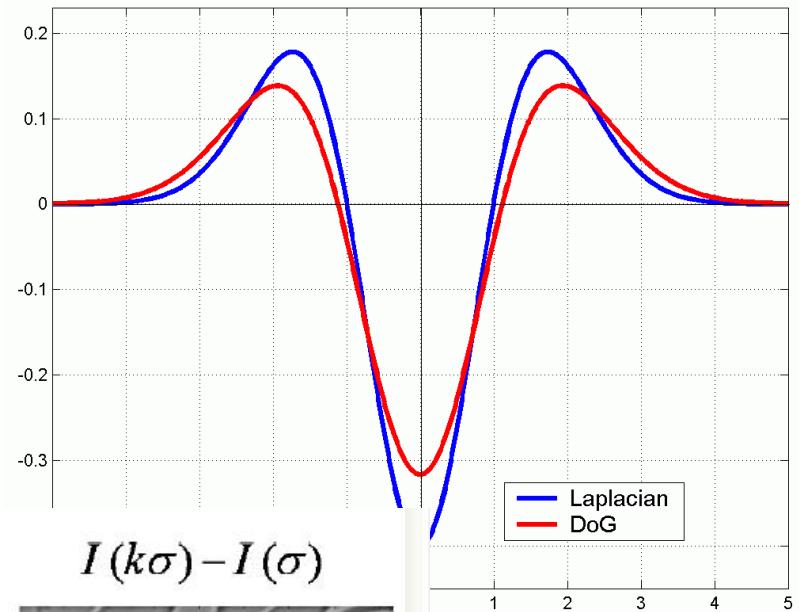
- We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

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

(Laplacian)

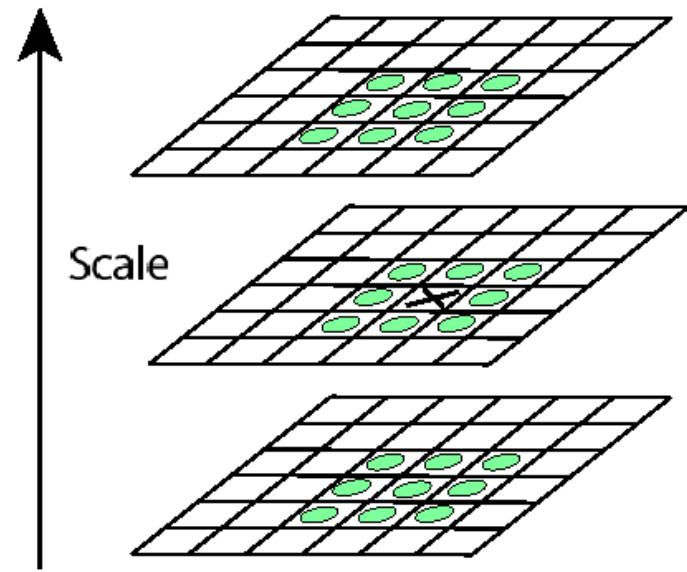
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



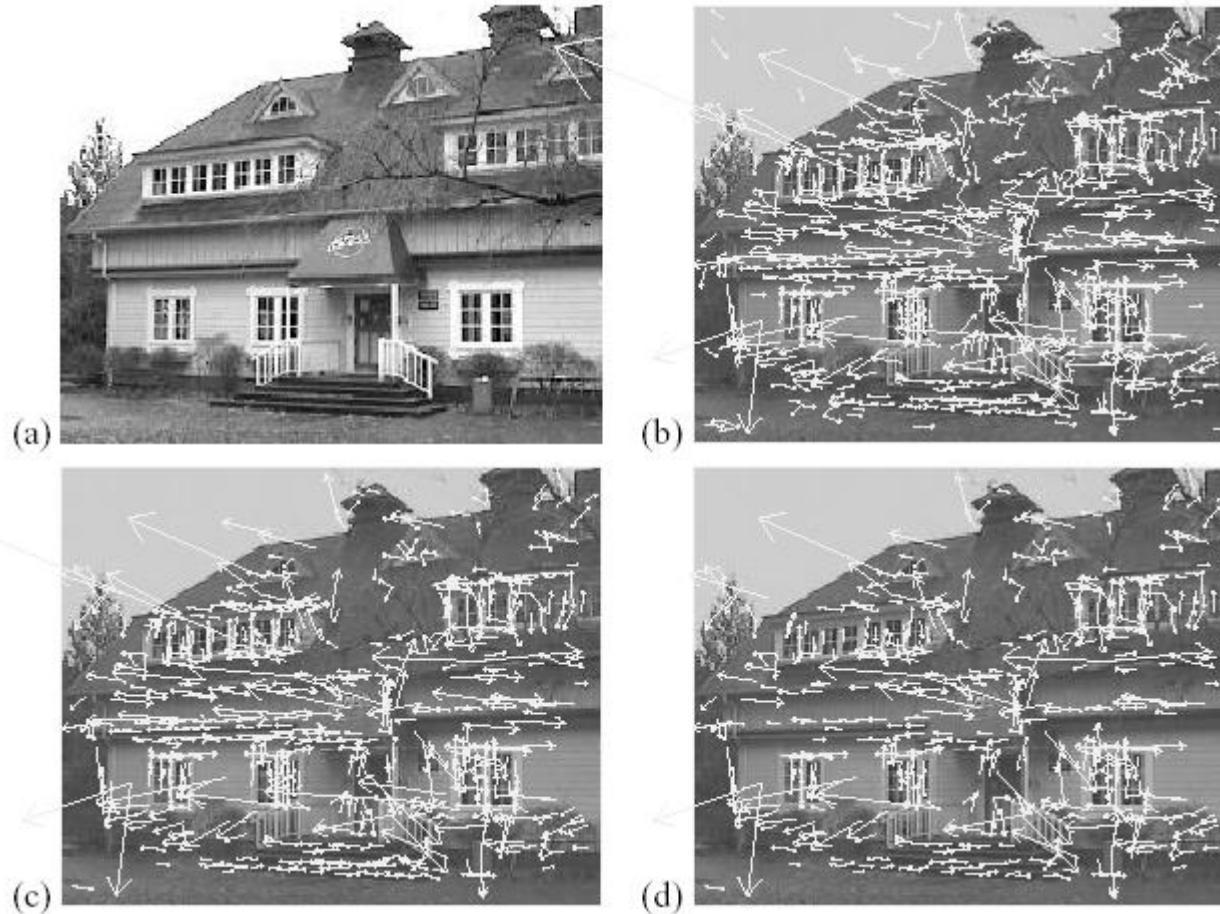
Key point localization with DoG

- Detect maxima of difference-of-Gaussian (DoG) in scale space
- Then reject points with low contrast (threshold)
- Eliminate edge responses



Candidate keypoints:
list of (x, y, σ)

Example of keypoint detection



- (a) 233x189 image
- (b) 832 DOG extrema
- (c) 729 left after peak value threshold
- (d) 536 left after testing ratio of principle curvatures (removing edge responses)

Scale Invariant Detection: Summary

- **Given:** two images of the same scene with a large *scale difference* between them
- **Goal:** find *the same* interest points *independently* in each image
- **Solution:** search for *maxima* of suitable functions in *scale* and in *space* (over the image)

Maximally Stable Extremal Regions

[Matas '02]

- Based on Watershed segmentation algorithm
- Select regions that stay stable over a large parameter range

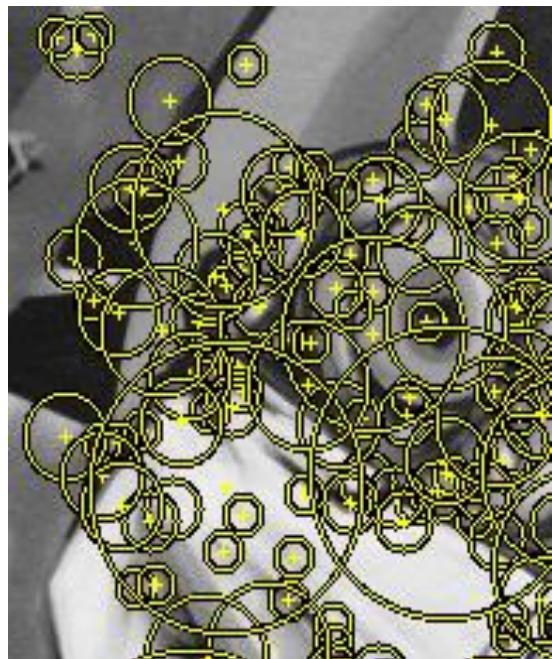
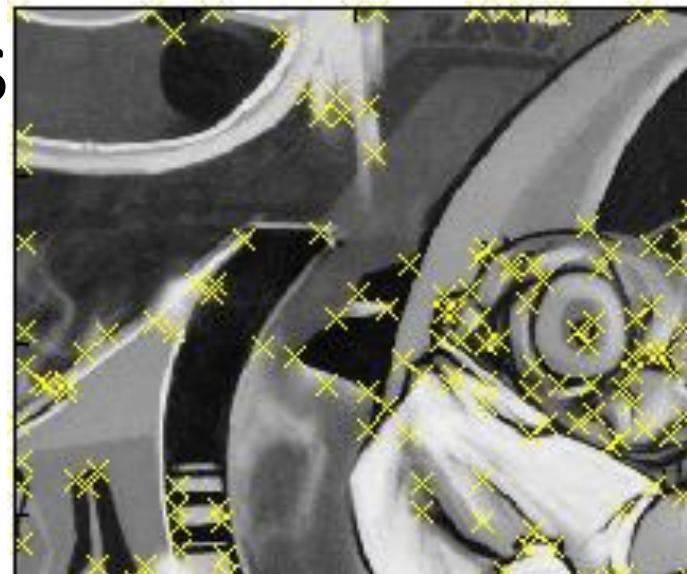


Example Results: MSER



Comparis

Harris

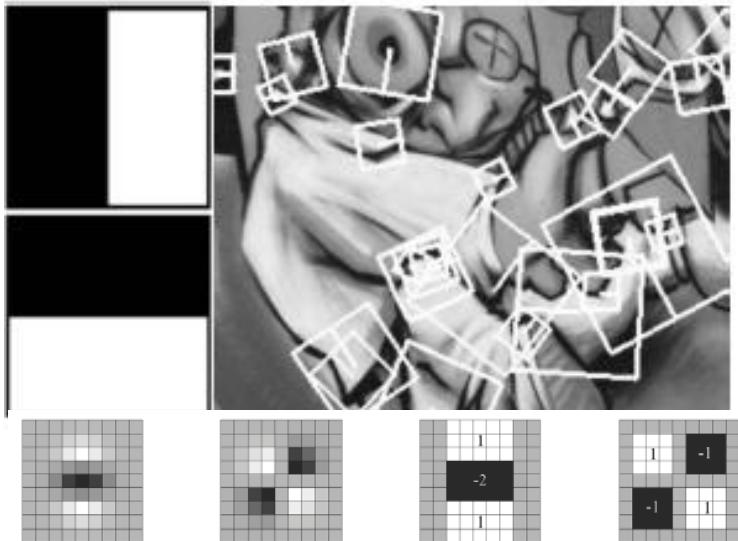


LoG

MSER



Local Descriptors: SURF



Fast approximation of SIFT idea

Efficient computation by 2D box filters & integral images

⇒ 6 times faster than SIFT

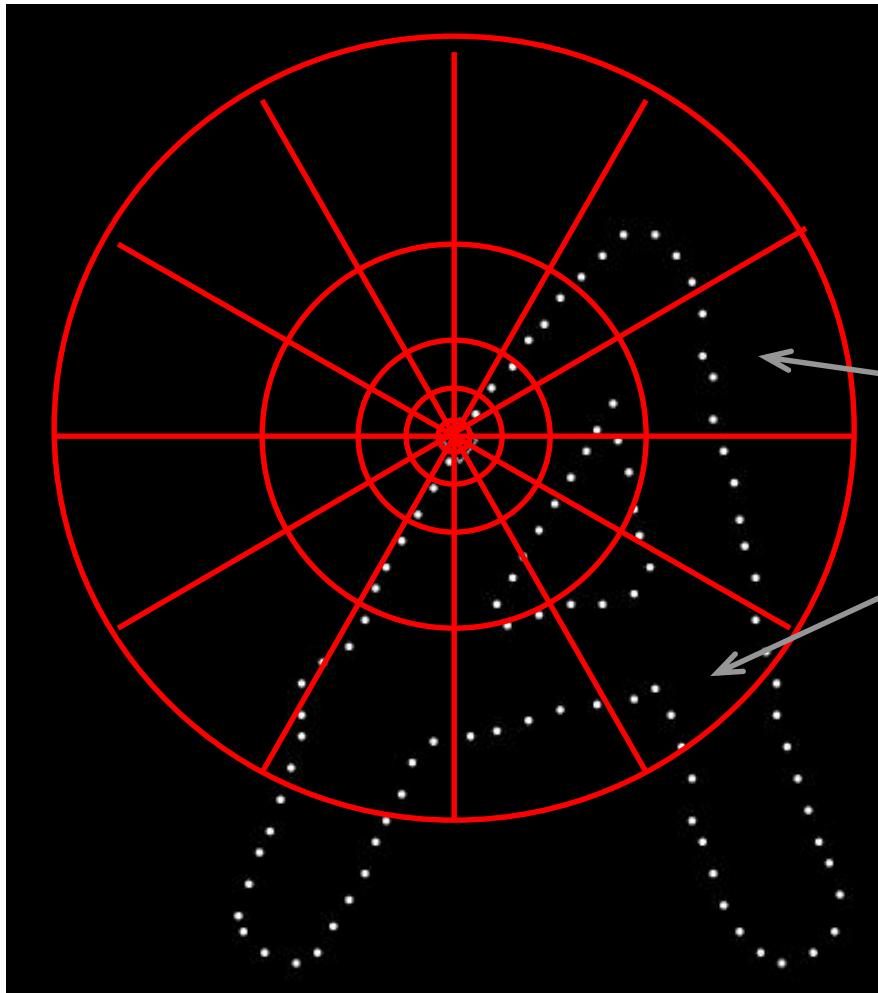
Equivalent quality for object identification

GPU implementation available

**Feature extraction @ 200Hz
(detector + descriptor, 640×480 img)**
<http://www.vision.ee.ethz.ch/~surf>

Many other efficient descriptors
are also available

Local Descriptors: Shape Context



Count the number of points inside each bin, e.g.:

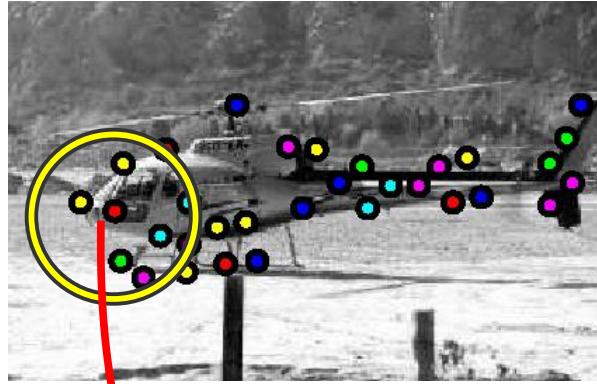
Count = 4

:

Count = 10

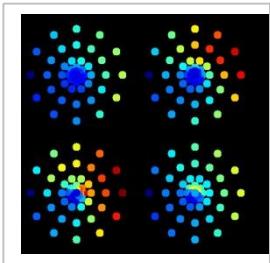
Log-polar binning: more precision for nearby points, more flexibility for farther points.

Local Descriptors: Geometric Blur

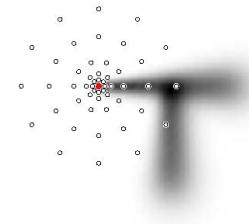


Compute edges
at four
orientations

Extract a patch
in each channel



Example descriptor



(Idealized signal)



**Apply spatially varying
blur and sub-sample**

Comparison of Keypoint Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	✓			✓			+++	+++	+++	++
Hessian		✓		✓			++	++	++	+
SUSAN	✓			✓			++	++	++	+++
Harris-Laplace	✓	(✓)		✓	✓		+++	+++	++	+
Hessian-Laplace	(✓)	✓		✓	✓		+++	+++	+++	+
DoG	(✓)	✓		✓	✓		++	++	++	++
SURF	(✓)	✓		✓	✓		++	++	++	+++
Harris-Affine	✓	(✓)		✓	✓	✓	+++	+++	++	++
Hessian-Affine	(✓)	✓		✓	✓	✓	+++	+++	+++	++
Salient Regions	(✓)	✓		✓	✓	(✓)	+	+	++	+
Edge-based	✓			✓	✓	✓	+++	+++	+	+
MSER		✓		✓	✓	✓	+++	+++	++	+++
Intensity-based		✓		✓	✓	✓	++	++	++	++
Superpixels		✓		✓	(✓)	(✓)	+	+	+	+

Choosing a detector

- What do you want it for?
 - Precise localization in x-y: Harris
 - Good localization in scale: Difference of Gaussian
 - Flexible region shape: MSER
- Best choice often application dependent
 - Harris-/Hessian-Laplace/DoG work well for many natural categories
 - MSER works well for buildings and printed things
- Why choose?
 - Get more points with more detectors
- There have been extensive evaluations/comparisons
 - [Mikolajczyk et al., IJCV'05, PAMI'05]
 - All detectors/descriptors shown here work well

- For most local feature detectors, executables are available online:
 - <http://robots.ox.ac.uk/~vgg/research/affine>
 - <http://www.cs.ubc.ca/~lowe/keypoints/>
 - <http://www.vision.ee.ethz.ch/~surf>

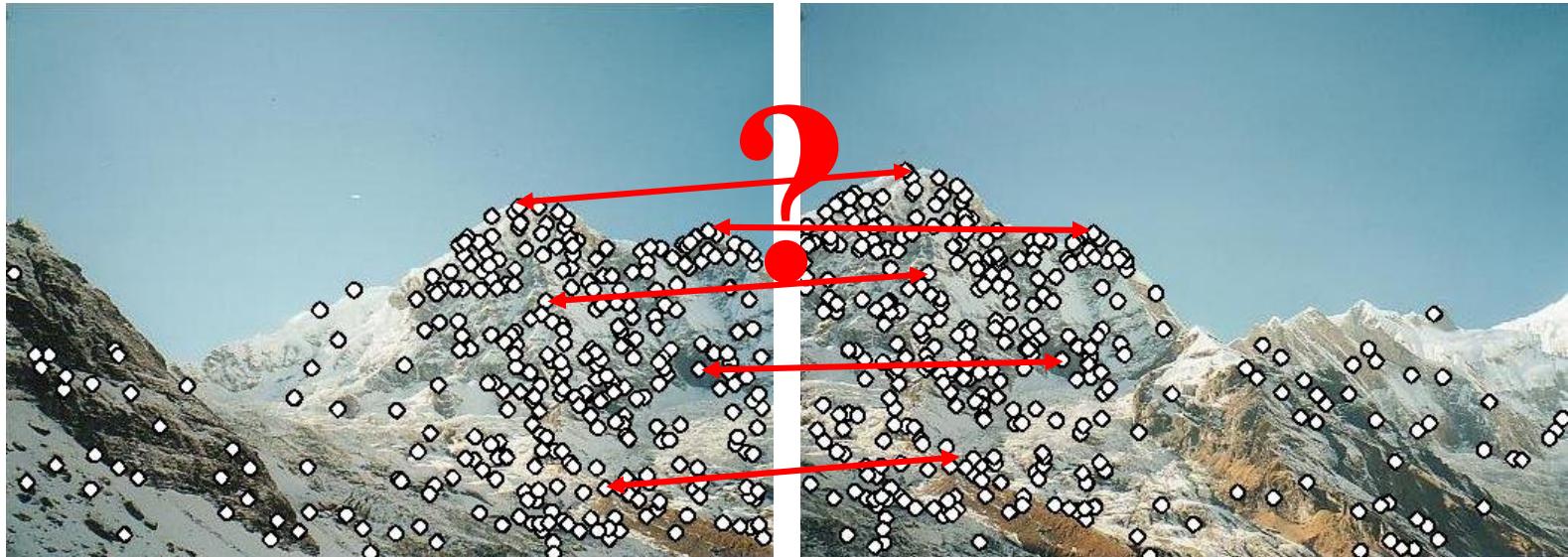
Main questions

- Where will the interest points come from?
 - What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

Local descriptors

- We know how to detect points
- Next question:

How to *describe* them for matching?



Point descriptor should be:

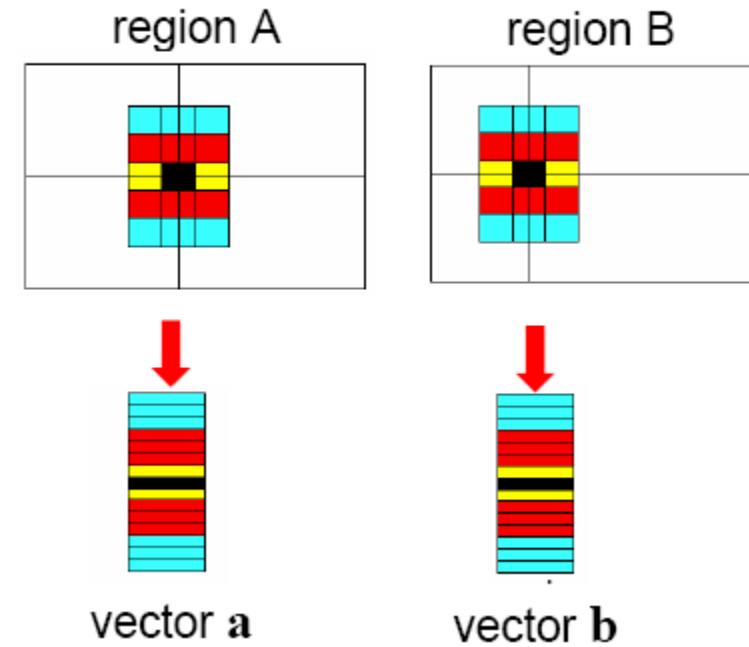
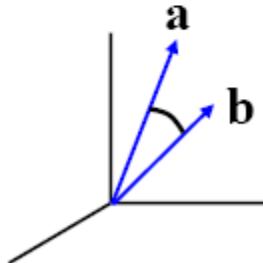
1. Invariant
2. Distinctive

Local descriptors

- Simplest descriptor: list of intensities within a patch.
- What is this going to be invariant to?

Write regions as vectors

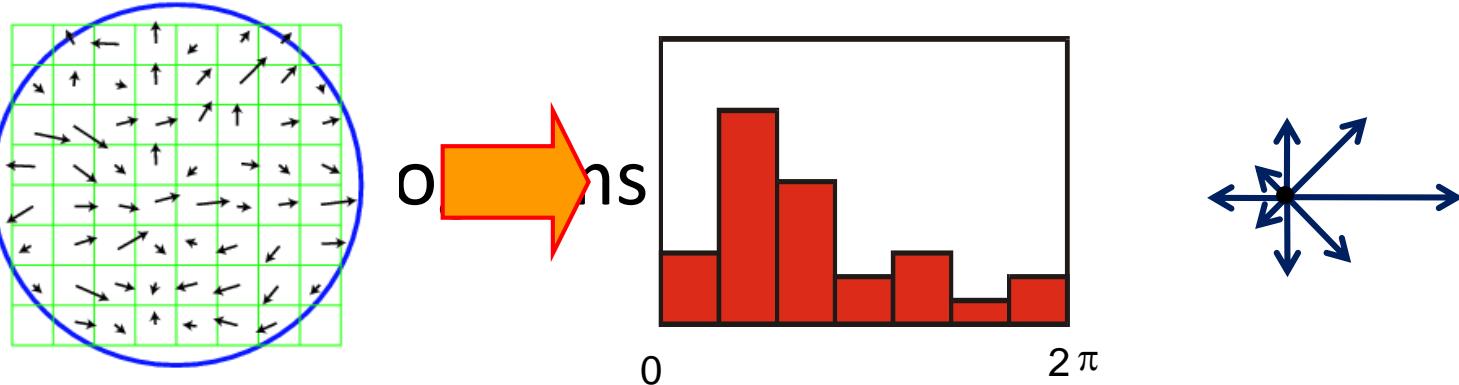
$$A \rightarrow \mathbf{a}, \quad B \rightarrow \mathbf{b}$$



Feature descriptors

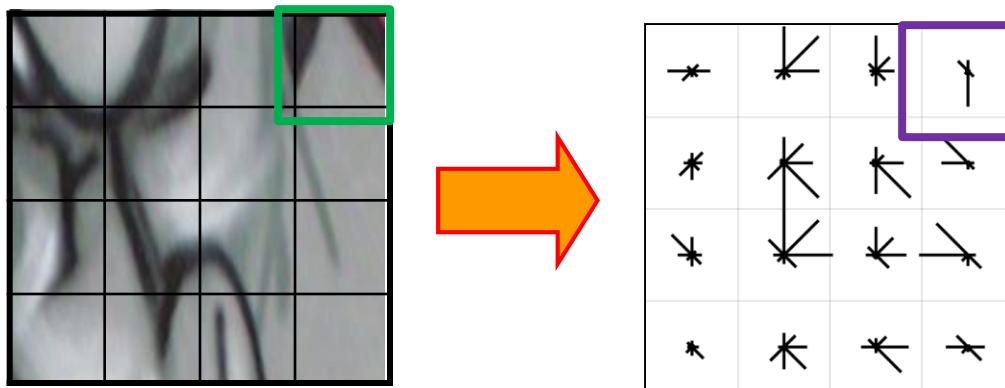
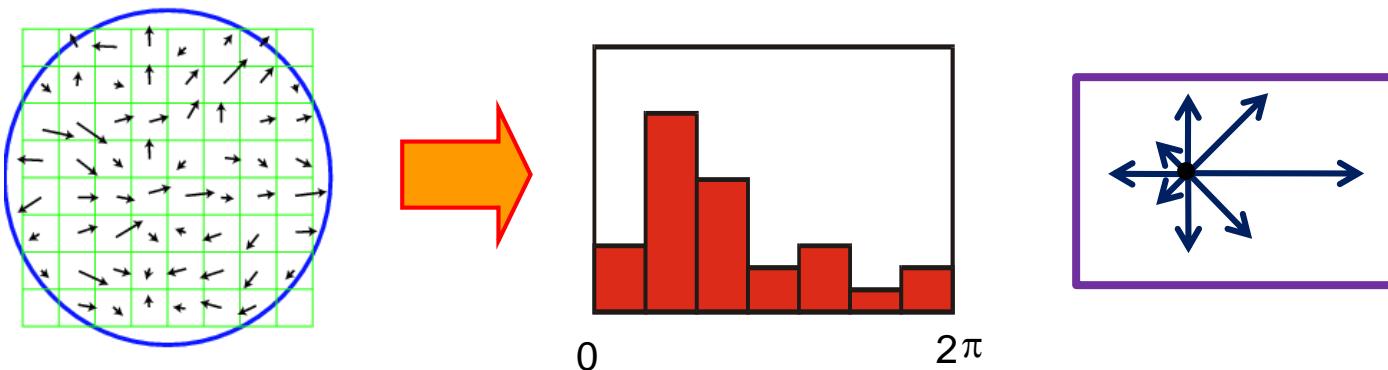
- Disadvantage of patches as descriptors:
 - Small shifts can score a lot



- So instead
- 
- | Angle Bin (0 to 2π) | Frequency |
|--------------------------|-----------|
| [0, $\pi/4$) | 10 |
| [$\pi/4$, $\pi/2$) | 25 |
| [$\pi/2$, $3\pi/4$) | 15 |
| [$3\pi/4$, π) | 5 |
| [π , $5\pi/4$) | 10 |
| [$5\pi/4$, $7\pi/4$) | 5 |
| [$7\pi/4$, 2π) | 5 |

SIFT descriptor [Lowe 2004]

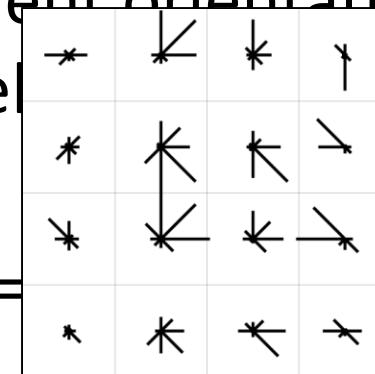
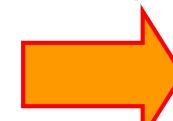
- Use histograms to bin pixels within sub-patches according to their orientation.



*Why subpatches?
Why does SIFT have
some illumination
invariance?*

Feature descriptors: SIFT

- Scale Invariant Feature Transform
- Descriptor computation:
 - Divide patch into 4×4 sub-patches: 16 cells
 - Compute histogram of gradient orientations (8 reference angles) for all pixels in sub-patch
 - Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions

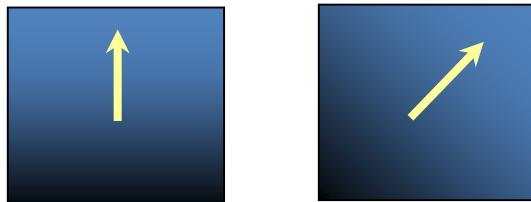


David G. Lowe. "[Distinctive image features from scale-invariant keypoints.](#)" IJCV 60 (2), pp. 91-110, 2004.

Rotation Invariant Descriptors

- Find local orientation

Dominant direction of gradient for the image patch



- Rotate patch according to this angle

This puts the patches into a canonical orientation.

Rotation Invariant Descriptors

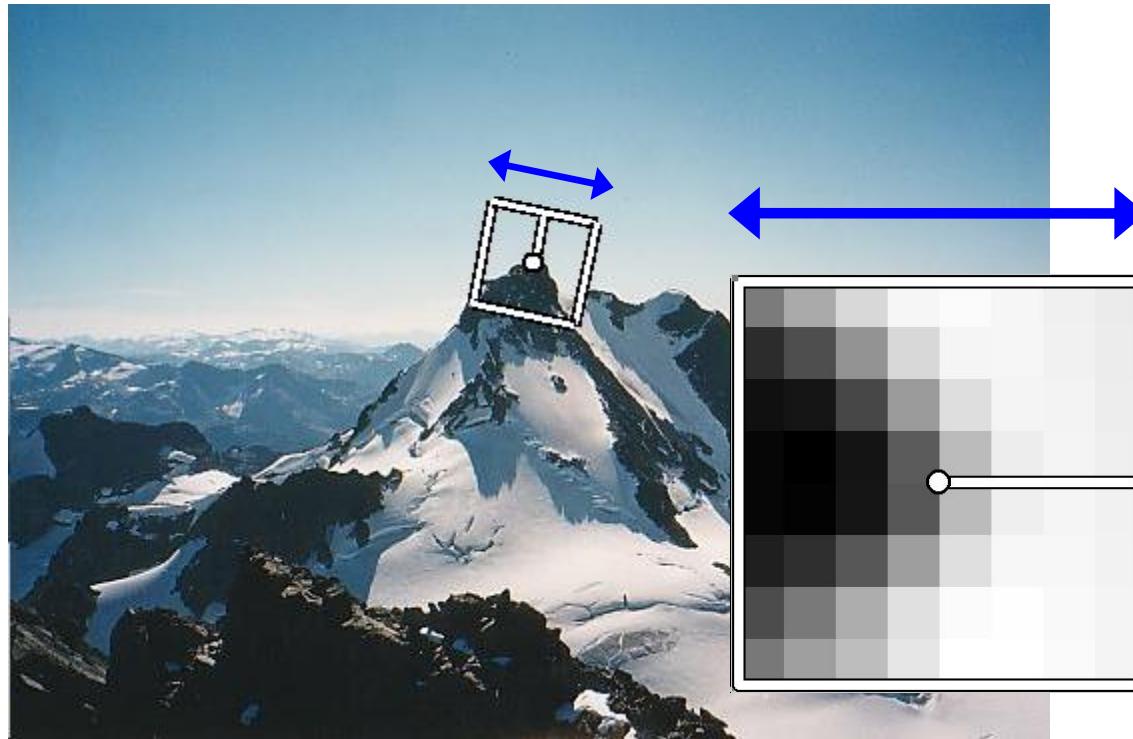
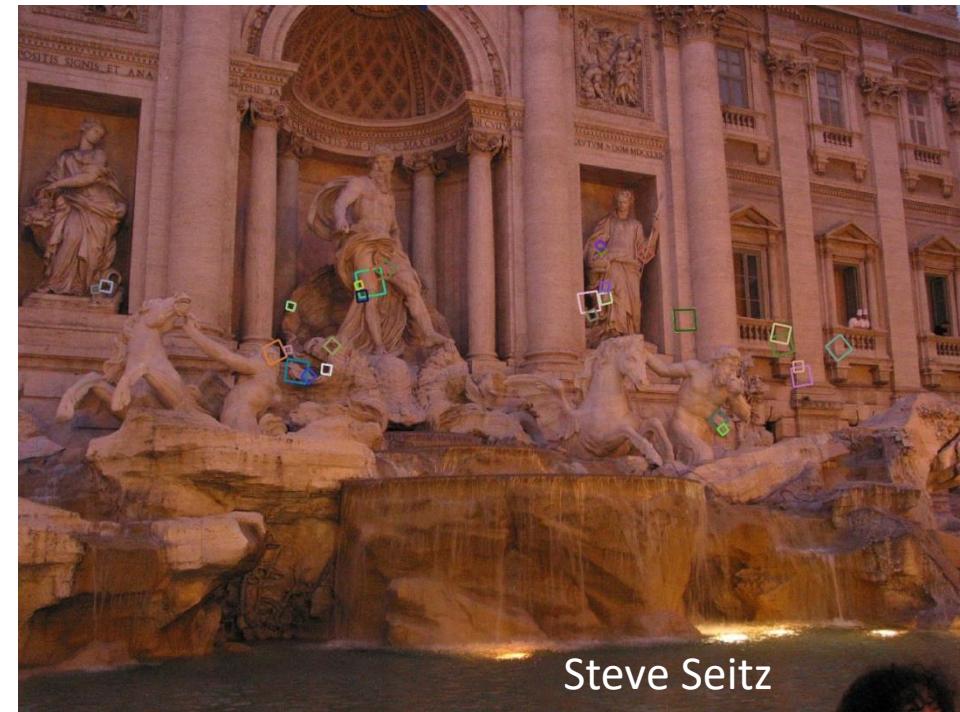


Image from Matthew Brown

Feature descriptors: SIFT

- 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
 - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



Steve Seitz

Working with SIFT descriptors

- One image yields:
 - n 128-dimensional descriptors: each one is a histogram of the gradient orientations within a patch
 - [$n \times 128$ matrix]
 - n scale parameters specifying the size of each patch
 - [$n \times 1$ vector]
 - n orientation parameters specifying the angle of the patch
 - [$n \times 1$ vector]
 - n 2d points giving positions of the patches
 - [$n \times 2$ matrix]



More on feature detection/description

The screenshot shows a web browser window. The address bar contains the URL <http://www.robots.ox.ac.uk/~vgg/research/affine/>. Below the address bar is a search bar with the query "mikolajczyk". To the right of the search bar are buttons for "Search Web" and "Google". The main content area displays the title "Affine Covariant Regions" in blue text. Below the title, there is a section titled "Publications" which lists various research papers under categories like "Region detectors" and "Region descriptors".

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: [K. Mikolajczyk](#) and [C. Schmid](#), Scale and Affine invariant interest point detectors. In IJCV 1(60):63-86, 2004. [PDF](#)
- *MSER*: [J. Matas](#), [O. Chum](#), [M. Urban](#), and [T. Pajdla](#), Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. [PDF](#)
- *IBR & EBR*: [T. Tuytelaars](#) and [L. Van Gool](#), Matching widely separated views based on affine invariant regions. In IJCV 1(59):61-85, 2004. [PDF](#)
- *Salient regions*: [T. Kadir](#), [A. Zisserman](#), and [M. Brady](#), An affine invariant salient region detector. In ECCV p. 404-416, 2004. [PDF](#)

Region descriptors

- *SIFT*: [D. Lowe](#), Distinctive image features from scale invariant keypoints. In IJCV 2(60):91-110, 2004. [PDF](#)

Performance evaluation

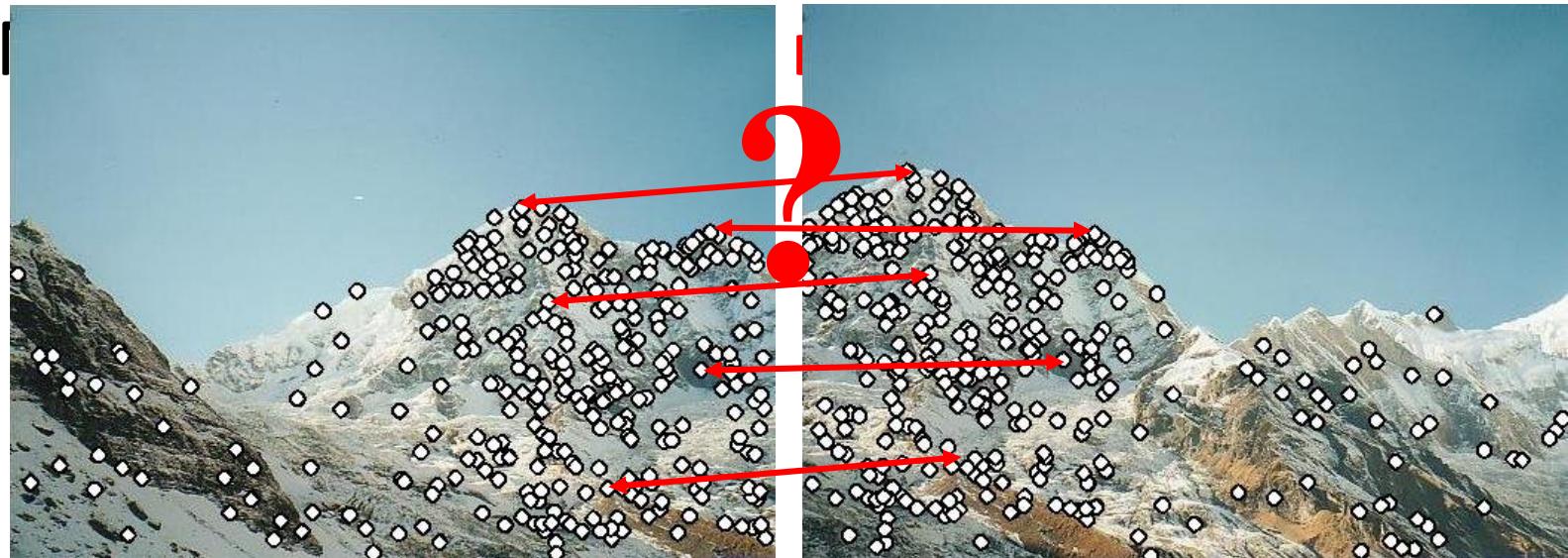
- [K. Mikolajczyk](#), [T. Tuytelaars](#), [C. Schmid](#), [A. Zisserman](#), [J. Matas](#), [F. Schaffalitzky](#), [T. Kadir](#) and [L. Van Gool](#), A comparison of affine region detectors. Technical Report, accepted to IJCV. [PDF](#)
- [K. Mikolajczyk](#), [C. Schmid](#), A performance evaluation of local descriptors. Technical Report, accepted to PAMI. [PDF](#)

Main questions

- Where will the interest points come from?
 - What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

Feature descriptors

We know how to detect and describe good points



Feature matching

Given a feature in I_1 , how to find the best match in I_2 ?

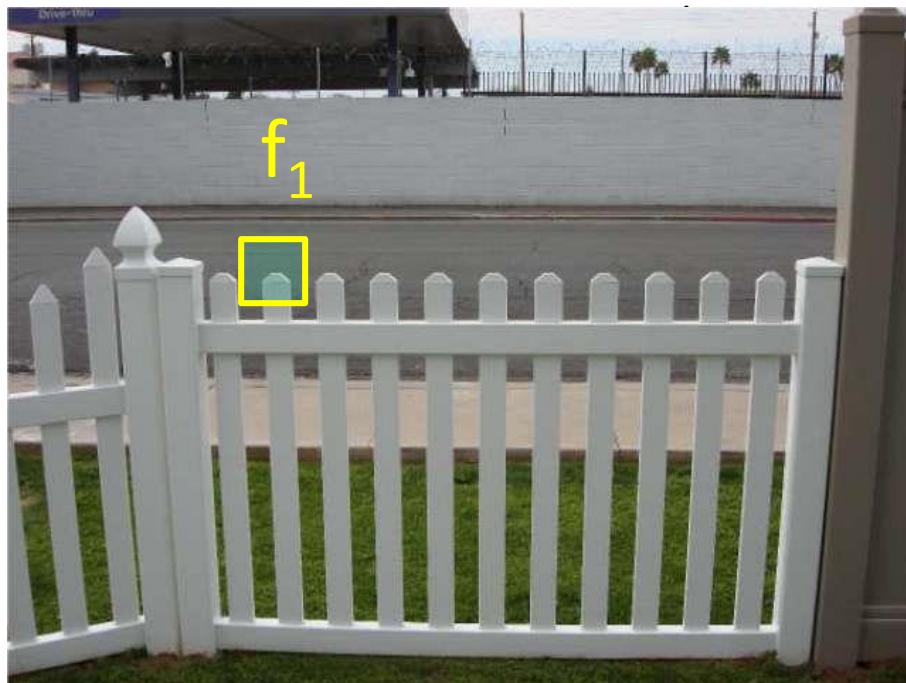
1. Define distance function that compares two descriptors
2. Test all the features in I_2 , find the one with min distance

Feature distance

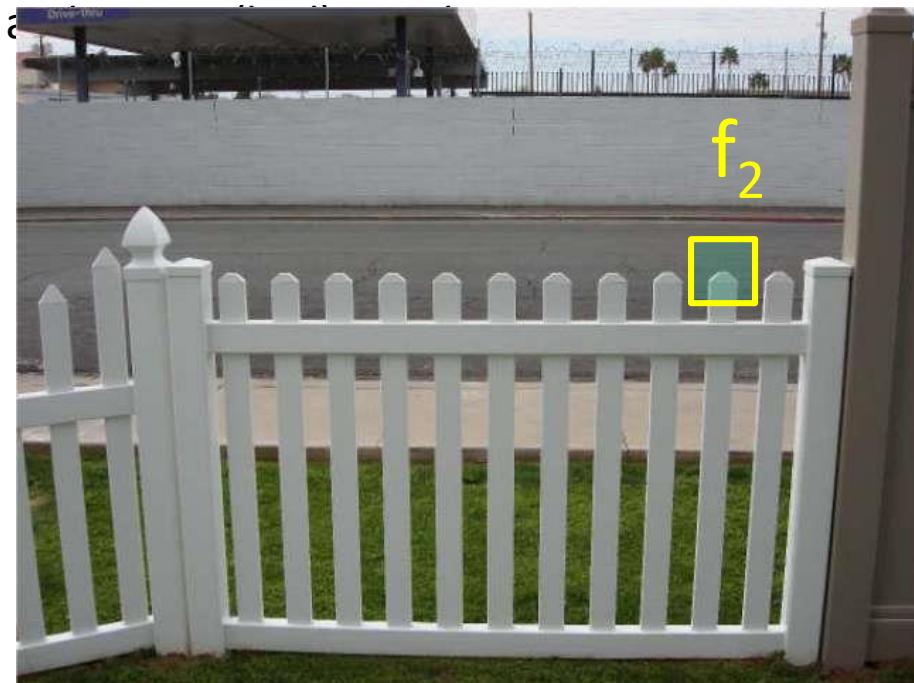
How to define the difference between two features f_1, f_2 ?

- Simple approach is $\text{SSD}(f_1, f_2)$

- sum of square differences between entries of the two descriptors



I_1

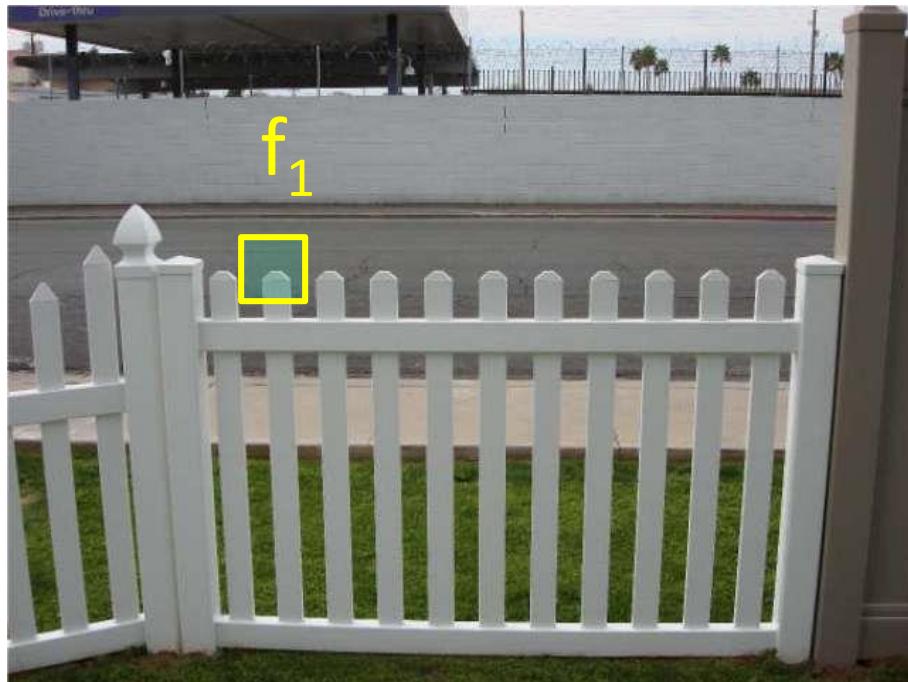


I_2

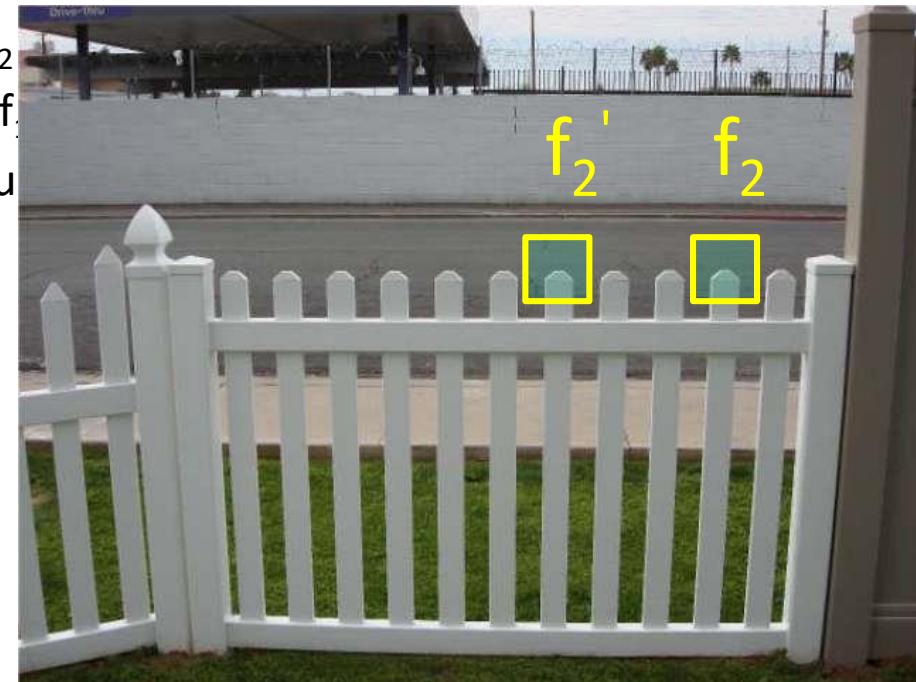
Feature distance

How to define the difference between two features f_1, f_2 ?

- Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$



I_1

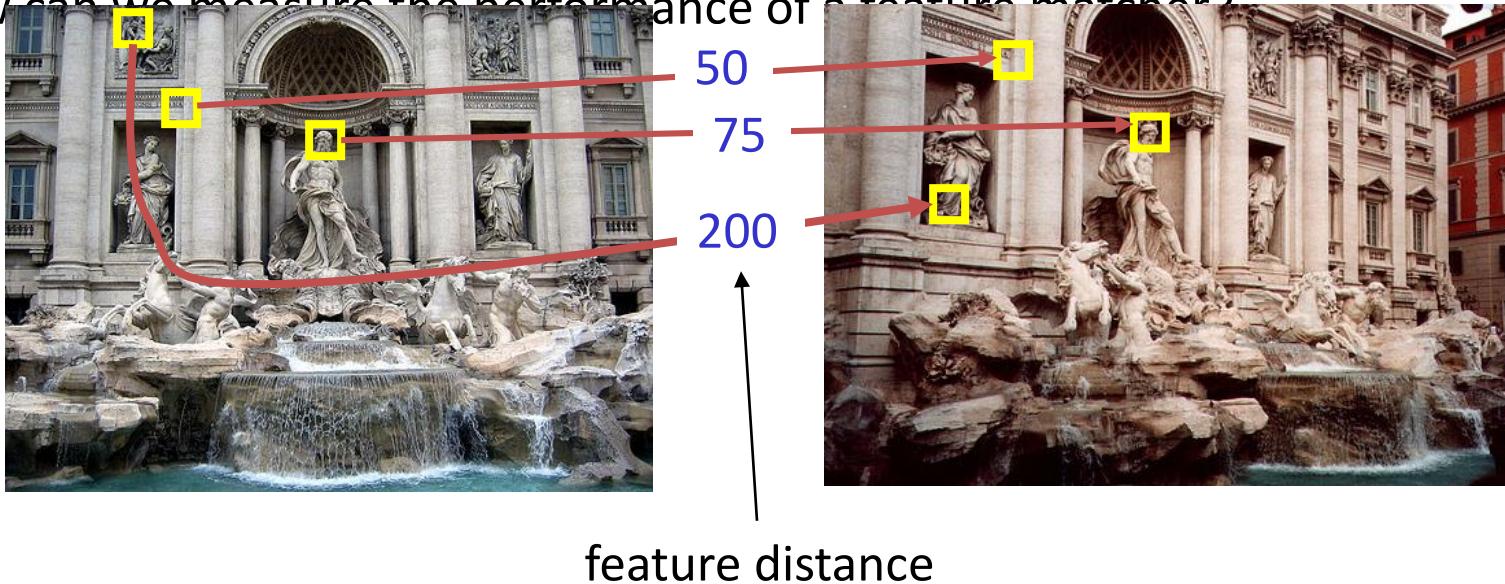


I_2

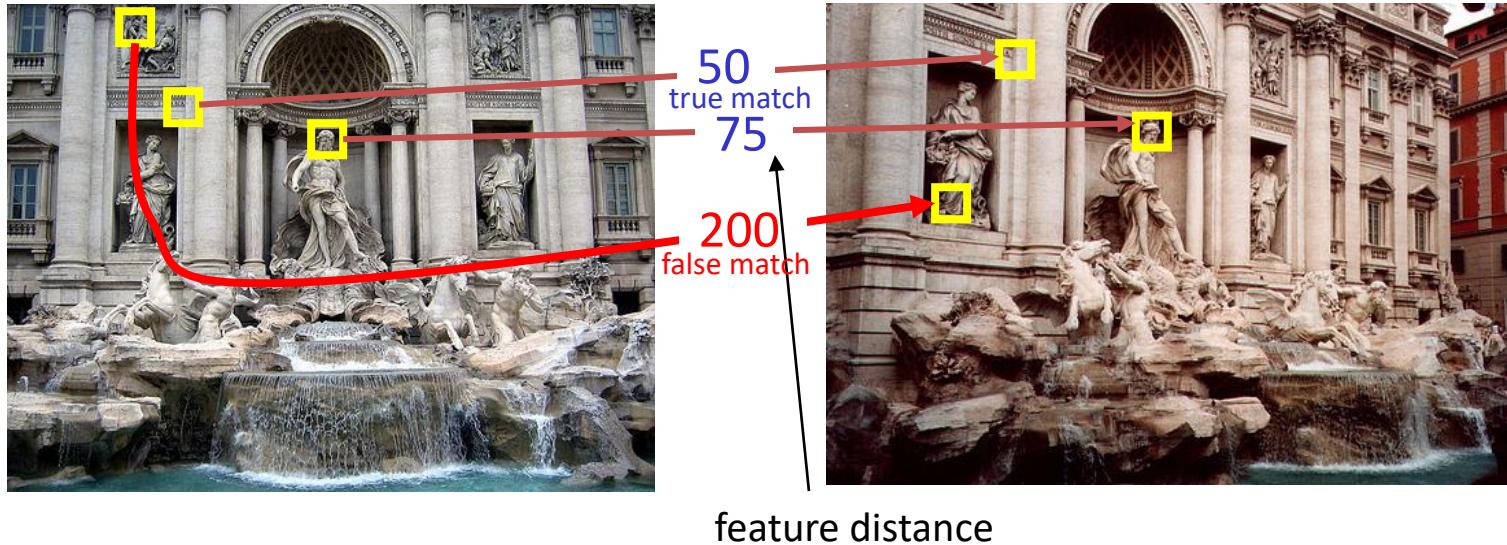
in I_2
to f_1
ambiguous

Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

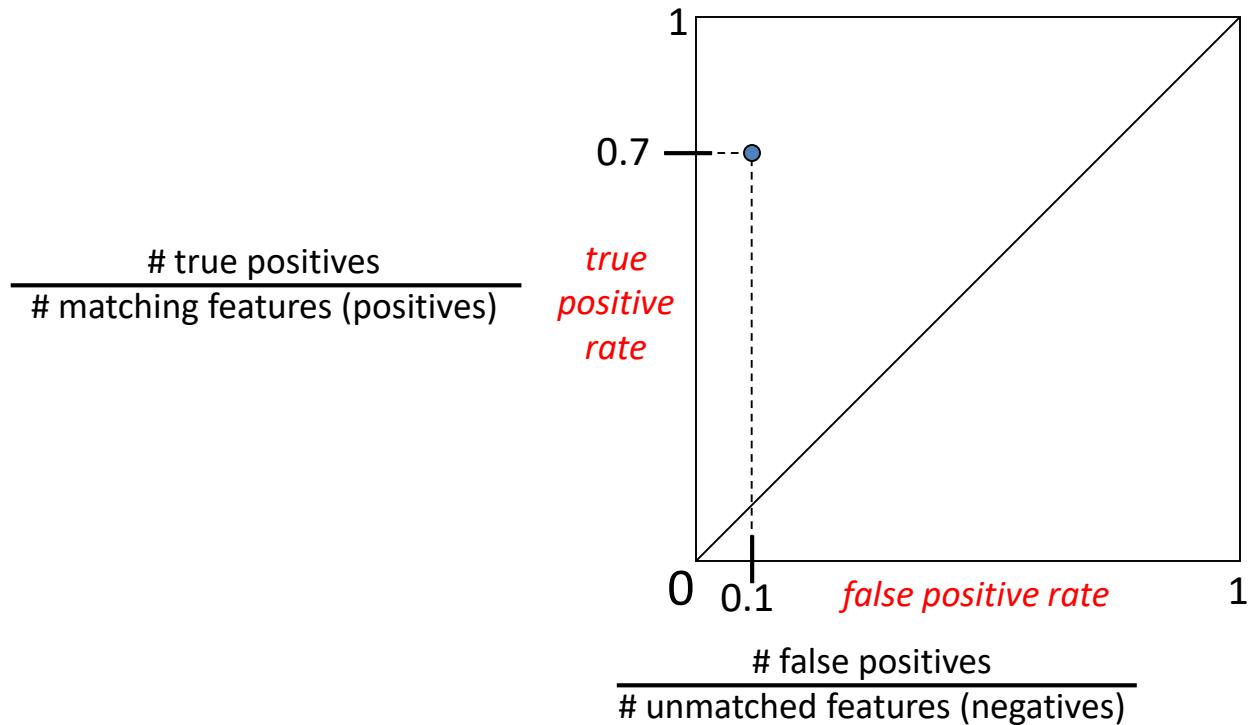


The distance threshold affects performance

- True positives = # of detected matches that are correct
 - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
 - Suppose we want to minimize these—how to choose threshold?

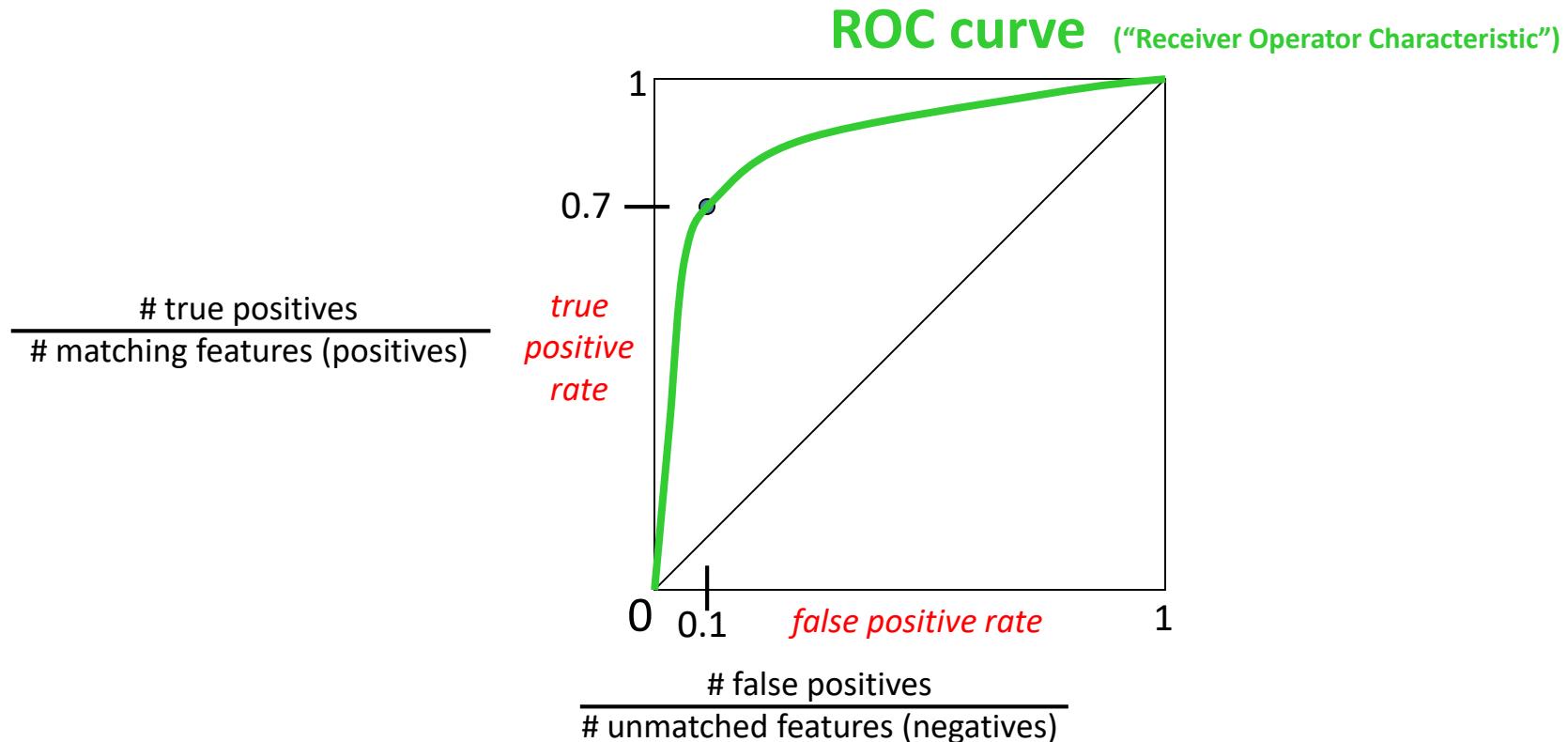
Evaluating the results

How can we measure the performance of a feature matcher?



Evaluating the results

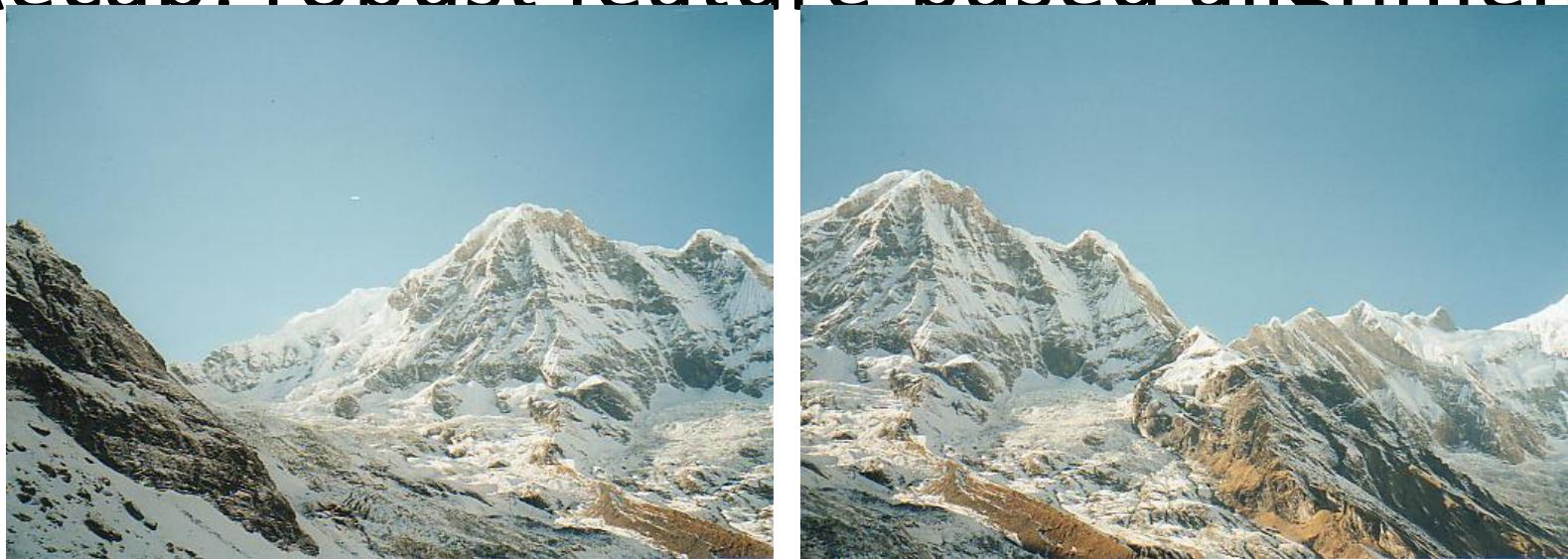
How can we measure the performance of a feature matcher?



ROC Curves

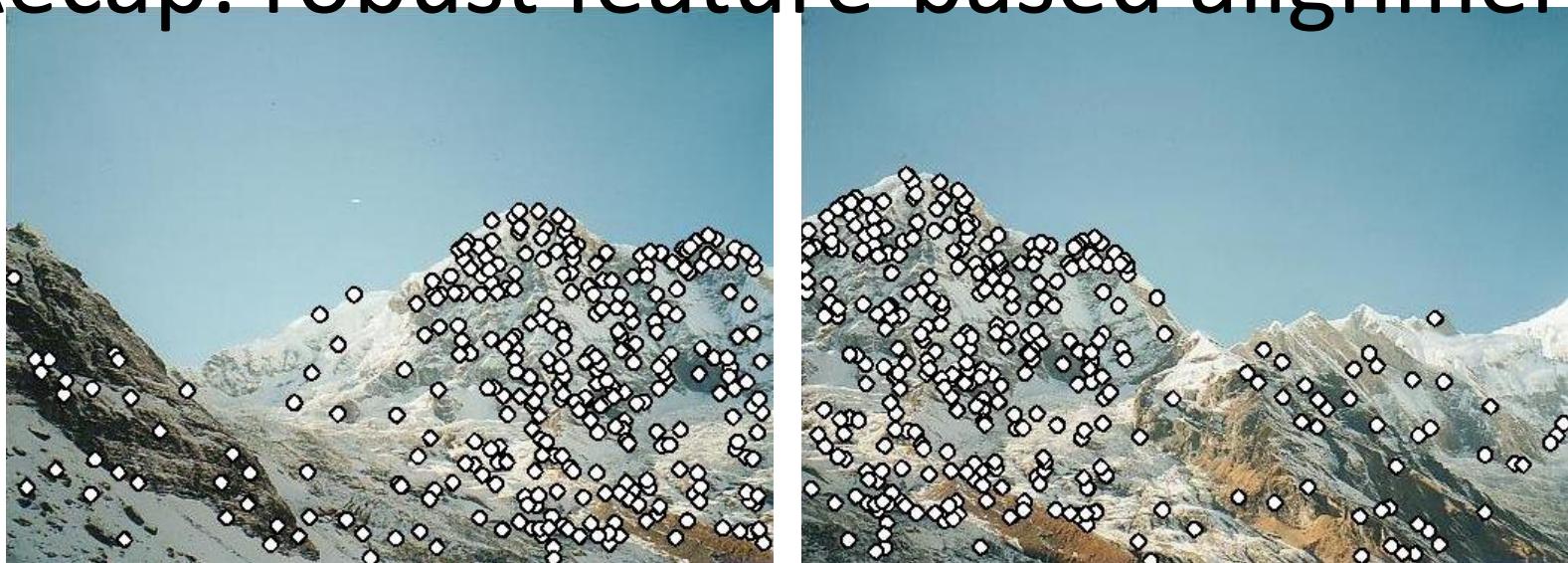
- Generated by counting # current/incorrect matches, for different thresholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods
- For more info: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

Recap: robust feature-based alignment



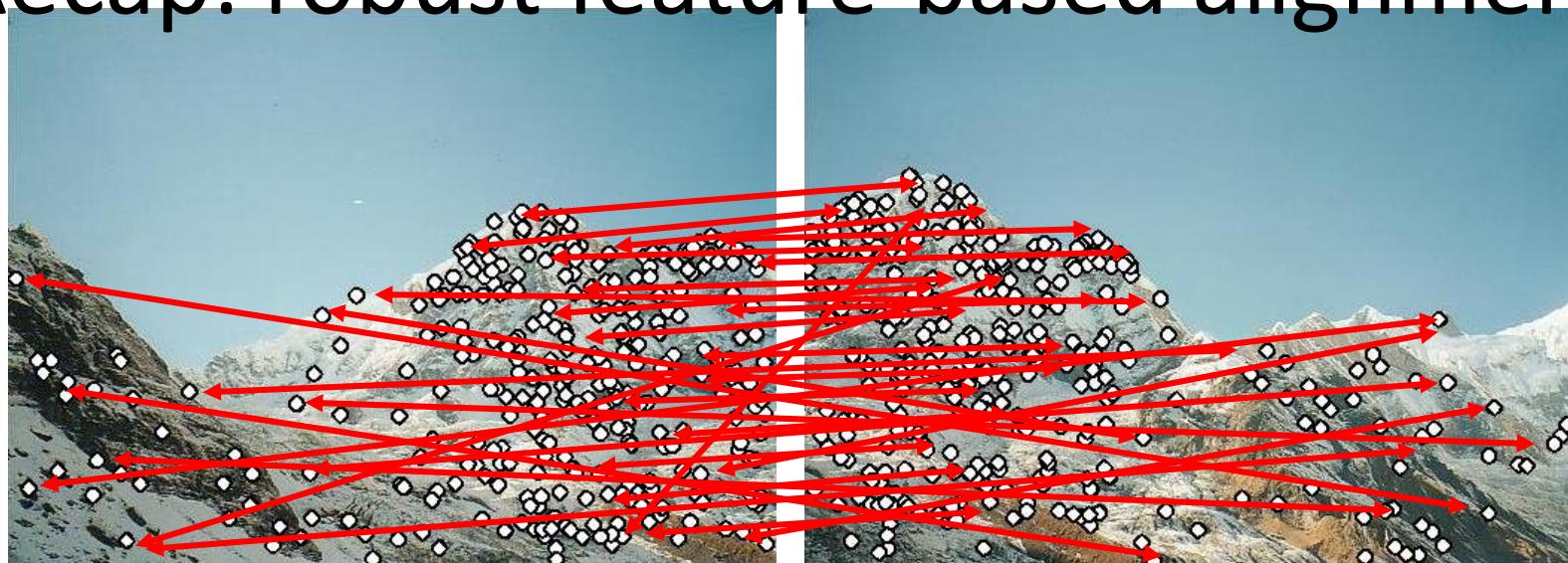
Source: L. Lazebnik

Recap: robust feature-based alignment



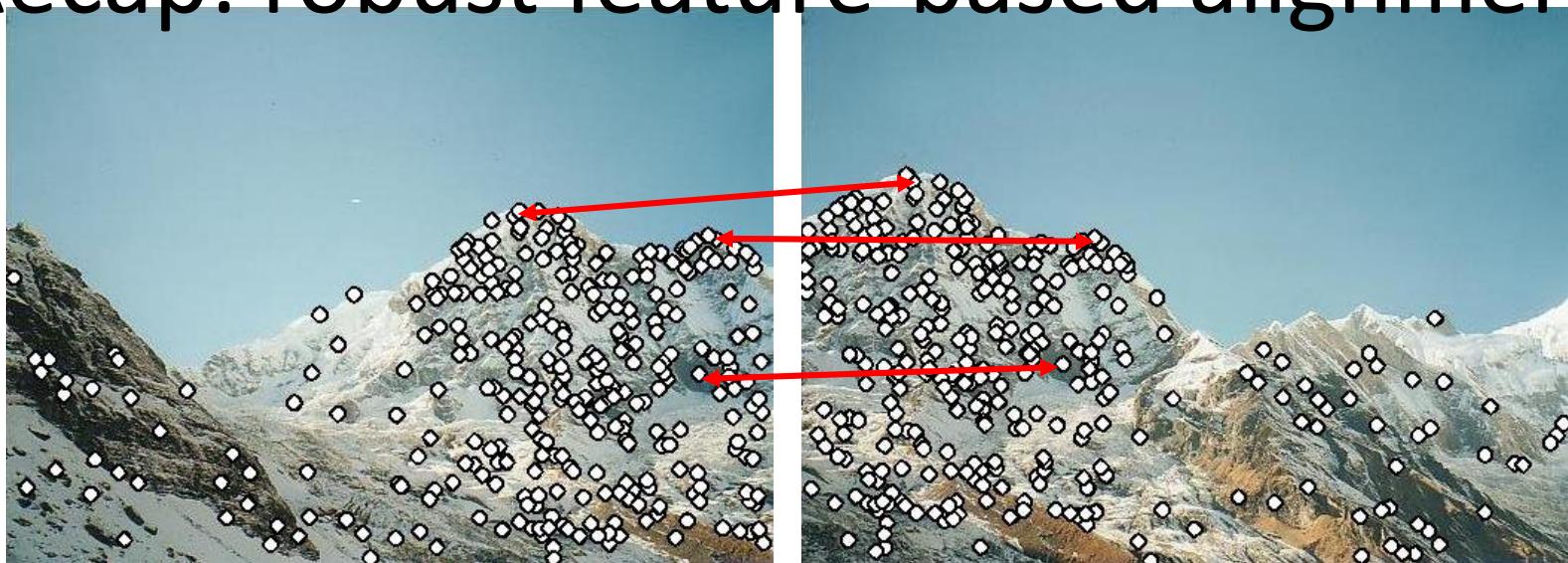
- Extract features

Recap: robust feature-based alignment



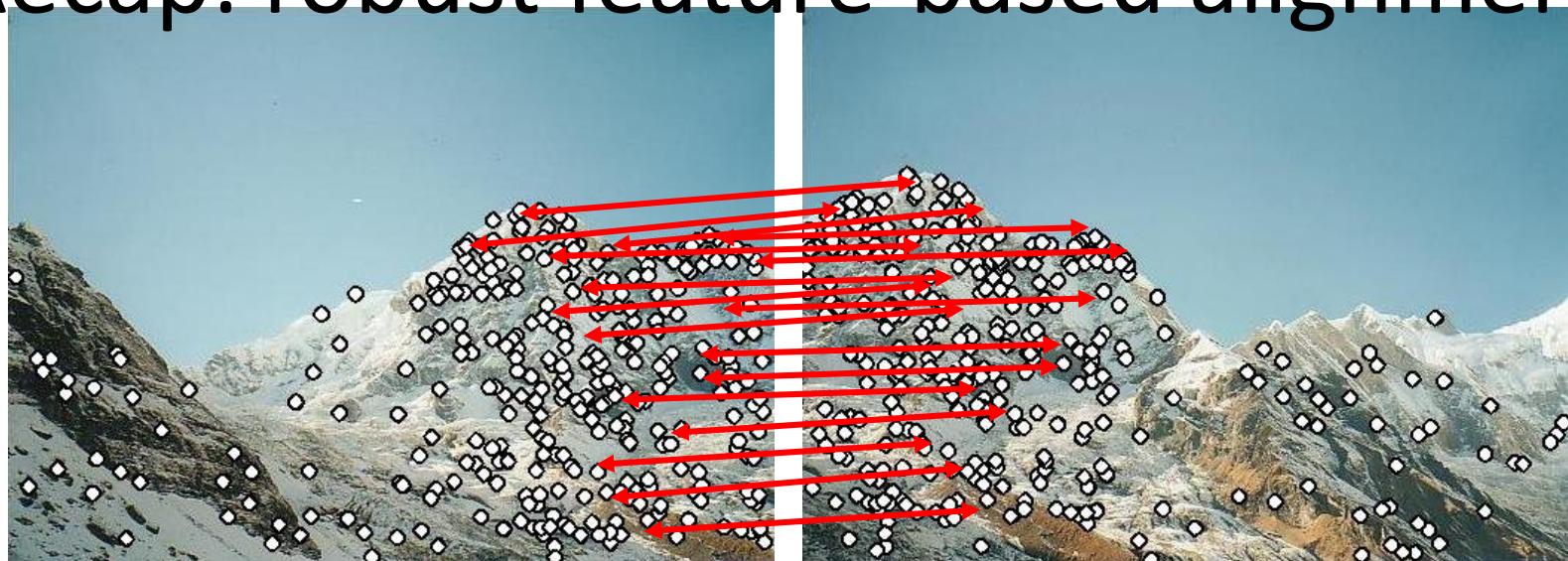
- Extract features
- Compute *putative matches*

Recap: robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
 - *Hypothesize* transformation T (small group of putative matches that are related by T)

Recap: robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
 - *Hypothesize* transformation T (small group of putative matches that are related by T)
 - *Verify* transformation (search for other matches consistent with T)

Recap: robust feature-based alignment

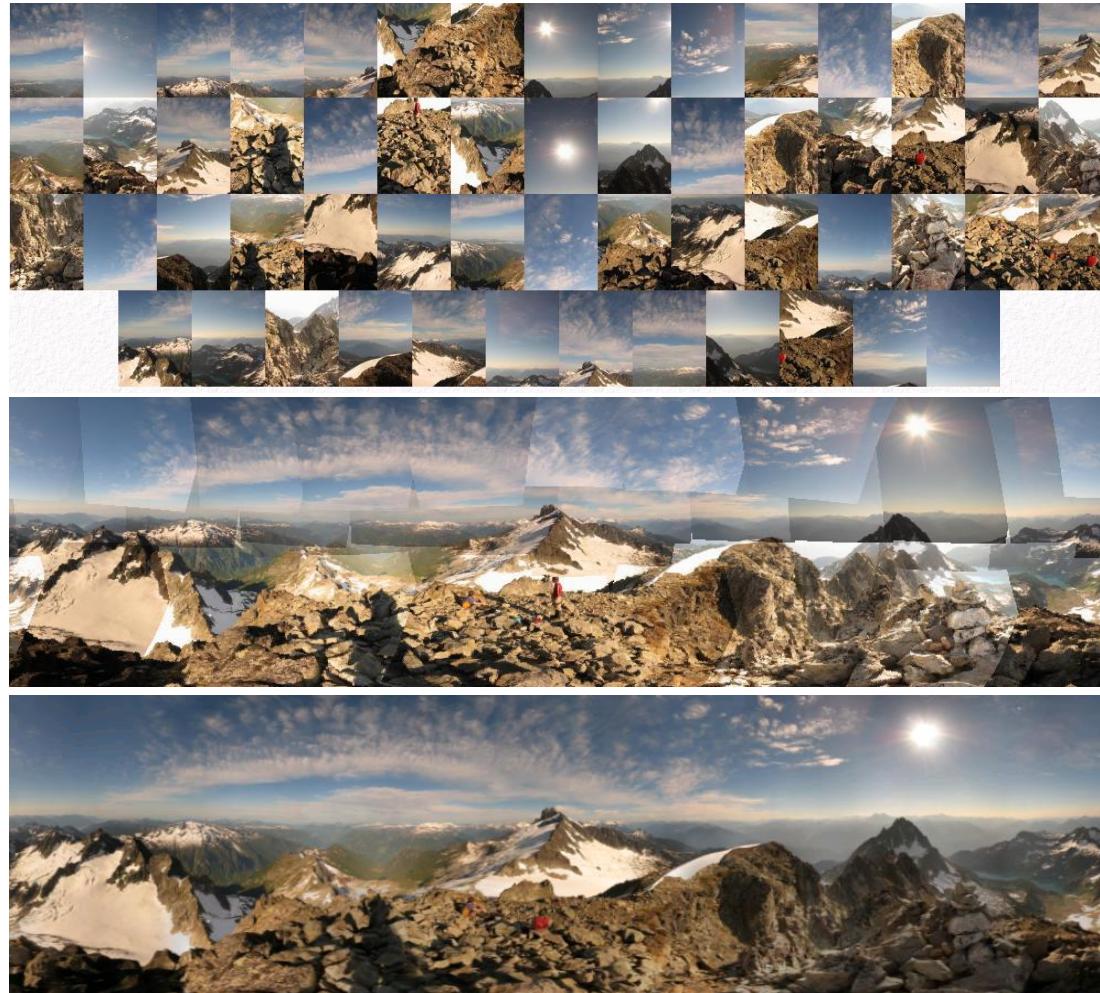


- Extract features
- Compute *putative matches*
- Loop:
 - *Hypothesize* transformation T (small group of putative matches that are related by T)
 - *Verify* transformation (search for other matches consistent with T)

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]

Recognition of specific objects, scenes



Schmid and Mohr 1997



Sivic and Zisserman, 2003

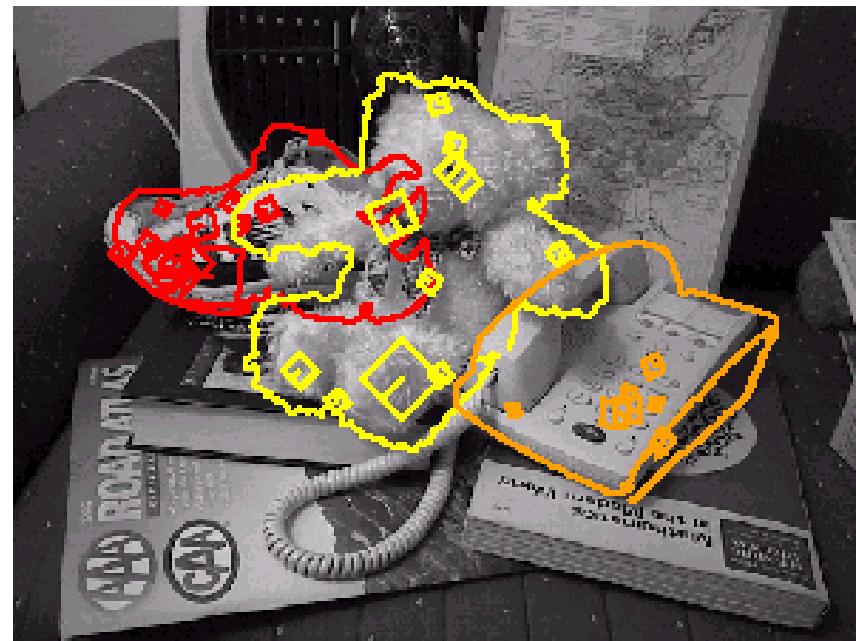


Rothganger et al. 2003

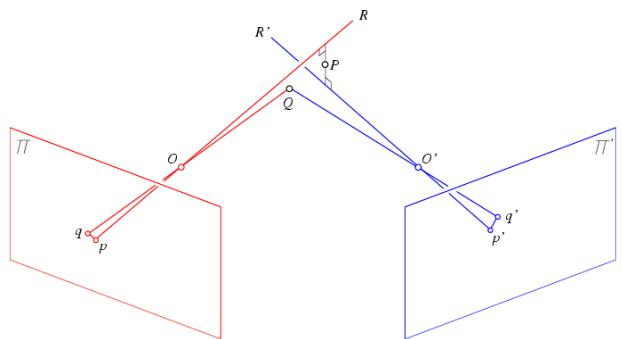


Lowe 2002

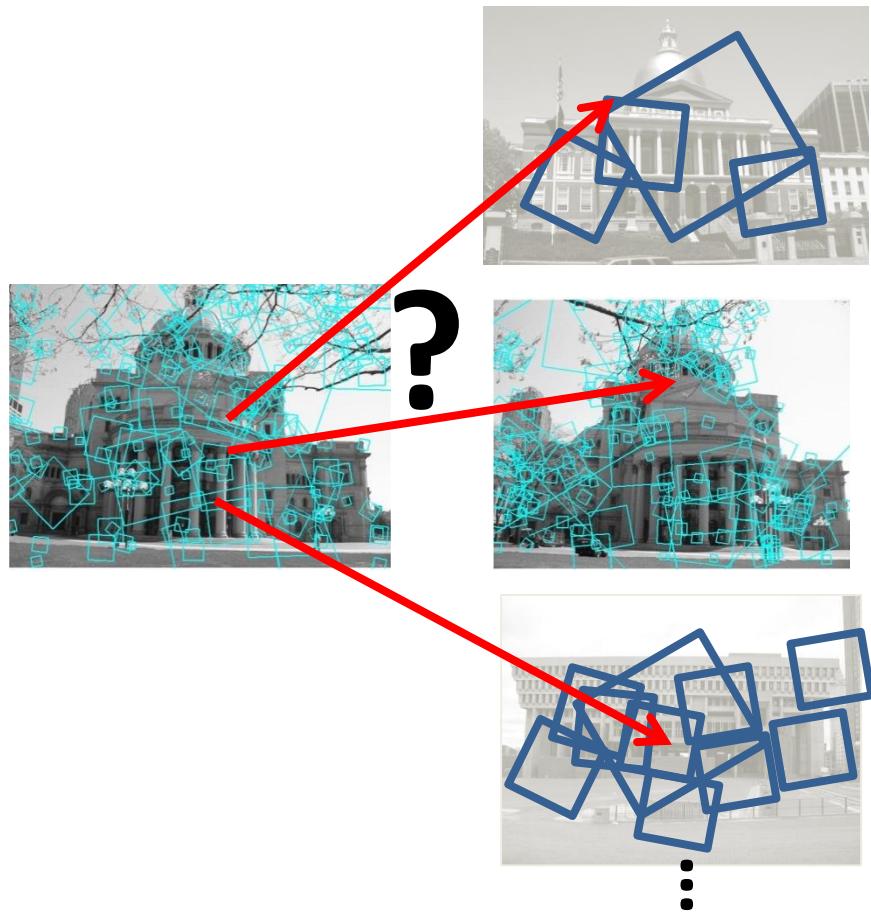
Object recognition (David Lowe)



Multi-view matching



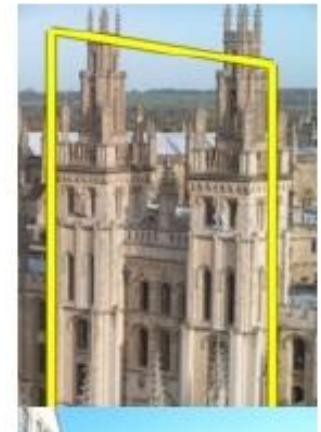
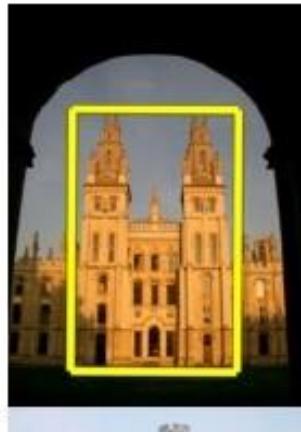
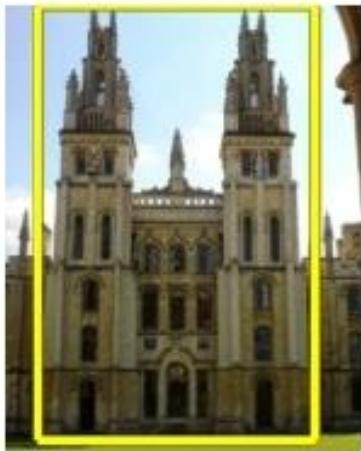
vs



Matching two given
views for depth

Search for a matching view
for recognition

How to quickly find images in a large database that match a given image region?



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>



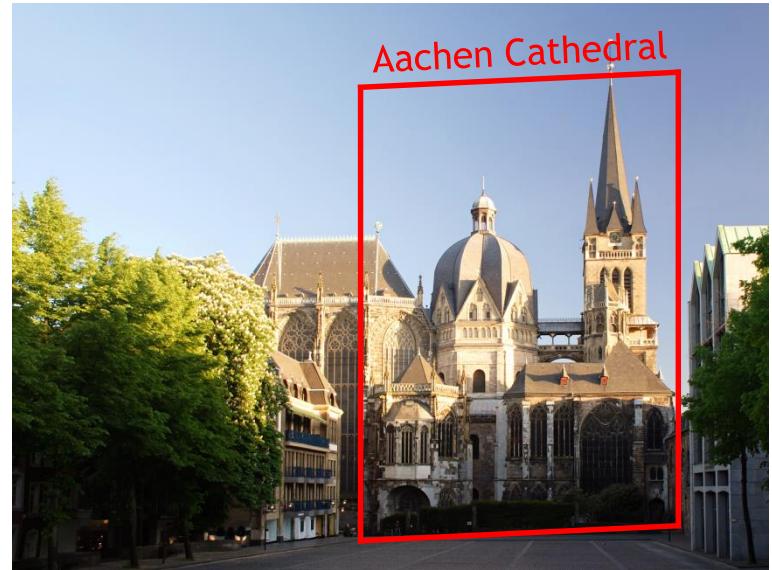
Query region



Retrieved frames

Kristen Grauman

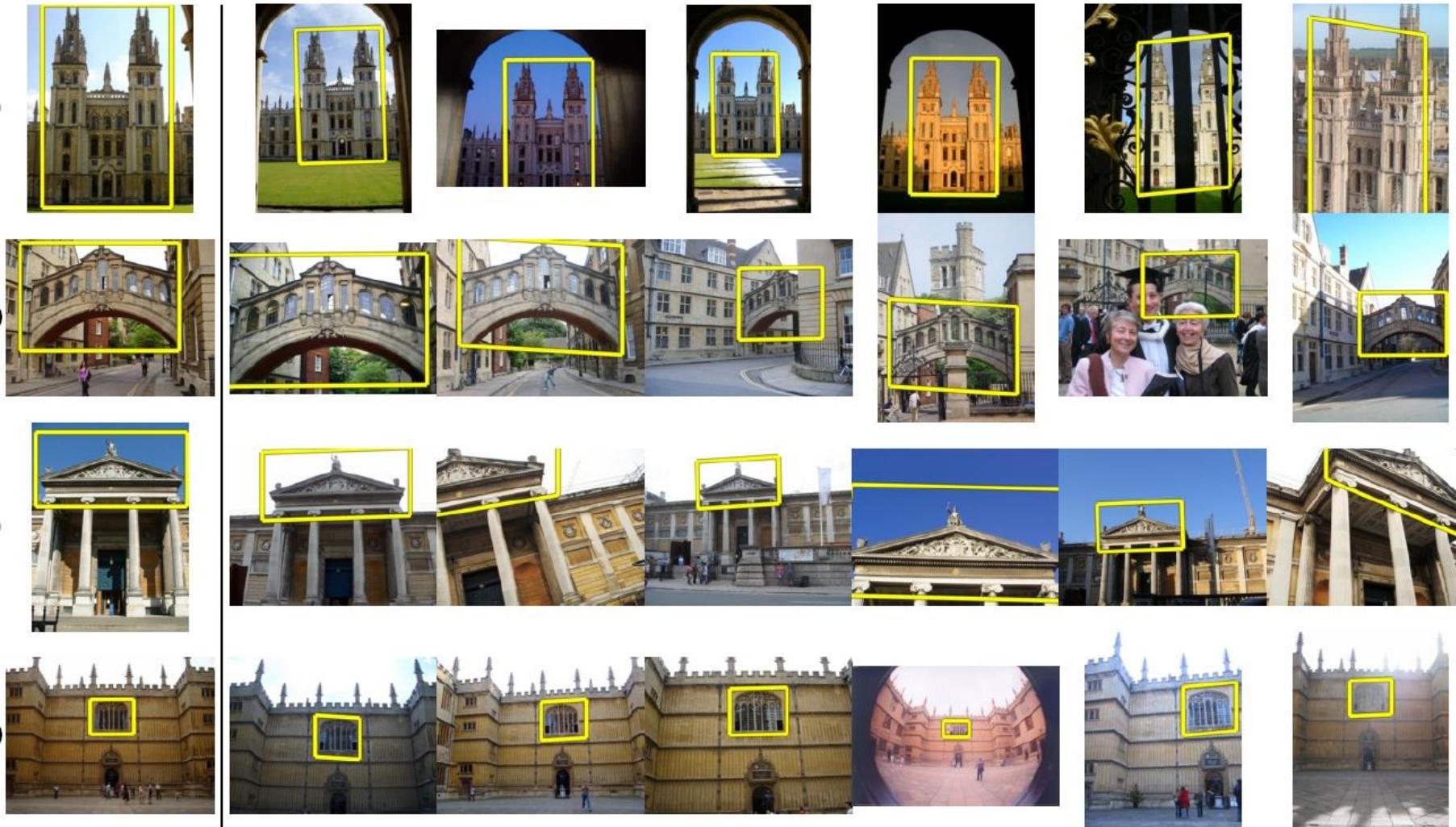
Example Applications



Mobile tourist guide

- Self-localization
- Object/building recognition
- Photo/video augmentation

Application: Large-Scale Retrieval



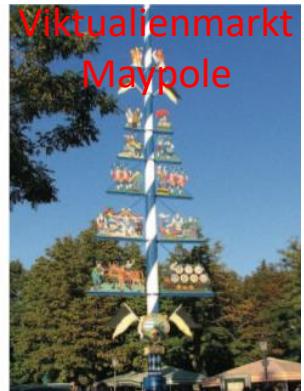
Query

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

Application: Image Auto-Annotation



Left: Wikipedia image
Right: closest match from Flickr





Google Goggles

Use pictures to search the web.

Watch a video



Get Google Goggles

Android (1.6+ required)

Download from [Android Market](#).

[Send Goggles to Android phone](#)

New! iPhone (iOS 4.0 required)

Download [from the App Store](#).

[Send Goggles to iPhone](#)

New!



[Text](#)



[Landmarks](#)



[Books](#)



[Contact Info](#)



[Artwork](#)



[Wine](#)



[Logos](#)



Ihr A

Fruh und

auf Topin



Photo Tourism

The screenshot shows a web browser window for "Photo Tourism" at <http://phototour.cs.washington.edu/>. The page features the University of Washington Computer Science & Engineering logo. The main title is "Photo Tourism: Exploring photo collections in 3D". A Microsoft logo is present. Three sections are displayed: (a) a grid of thumbnail images of a cathedral, (b) a sparse 3D point cloud reconstruction of the cathedral, and (c) a 3D camera view of the cathedral's facade with a user interface for navigation.

Photo Tourism
Exploring photo collections in 3D

Microsoft

(a)

(b)

(c)

Photo tourism is a system for browsing large collections of photographs in 3D. Our approach takes as input large collections of images from either personal photo collections or Internet photo sharing sites (a), and automatically computes each photo's viewpoint and a sparse 3D model of the scene (b). Our photo explorer interface enables the viewer to interactively move about the 3D space by seamlessly transitioning between photographs, based on user control (c).

Live Demo

New! See our work on **Finding Paths through the World's Photos**.

Our structure from motion code is also now available at the **Bundler** homepage.

Slide Credits

- Trevor Darrell
- Bill Freeman
- Kristen Grauman
- Steve Seitz
- Ivan Laptev
- Tinne Tuytelaars
- James Hays
- Svetlana Lazebnik
- Derek Hoiem