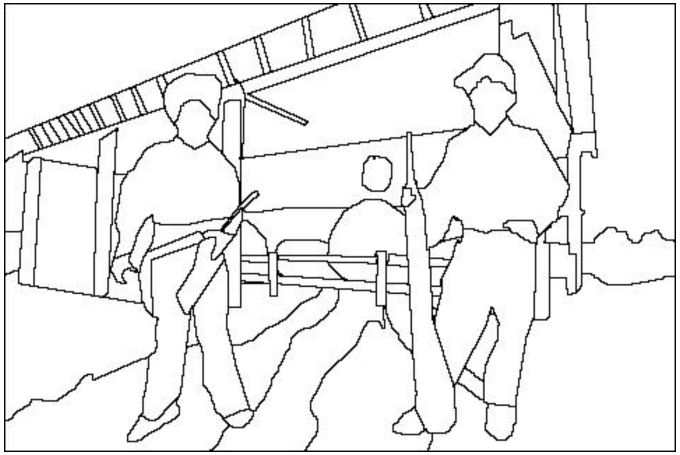
Feature Detection

COS 429
Princeton University

Summary, So Far

Algorithms to extract info from single image

- Frequencies, gradients
- Edges
- Primitives
- Segments
- Symmetries
- Texture



Summary, So Far

Algorithms to extract info from single image

- Frequencies, gradients
- Edges
- Primitives
- Segments
- Symmetries
- Texture

- What else?

Starting Today

Extract info from multiple images

- What kind of info would be useful?

Image Correspondence

Goal: Find map between two images

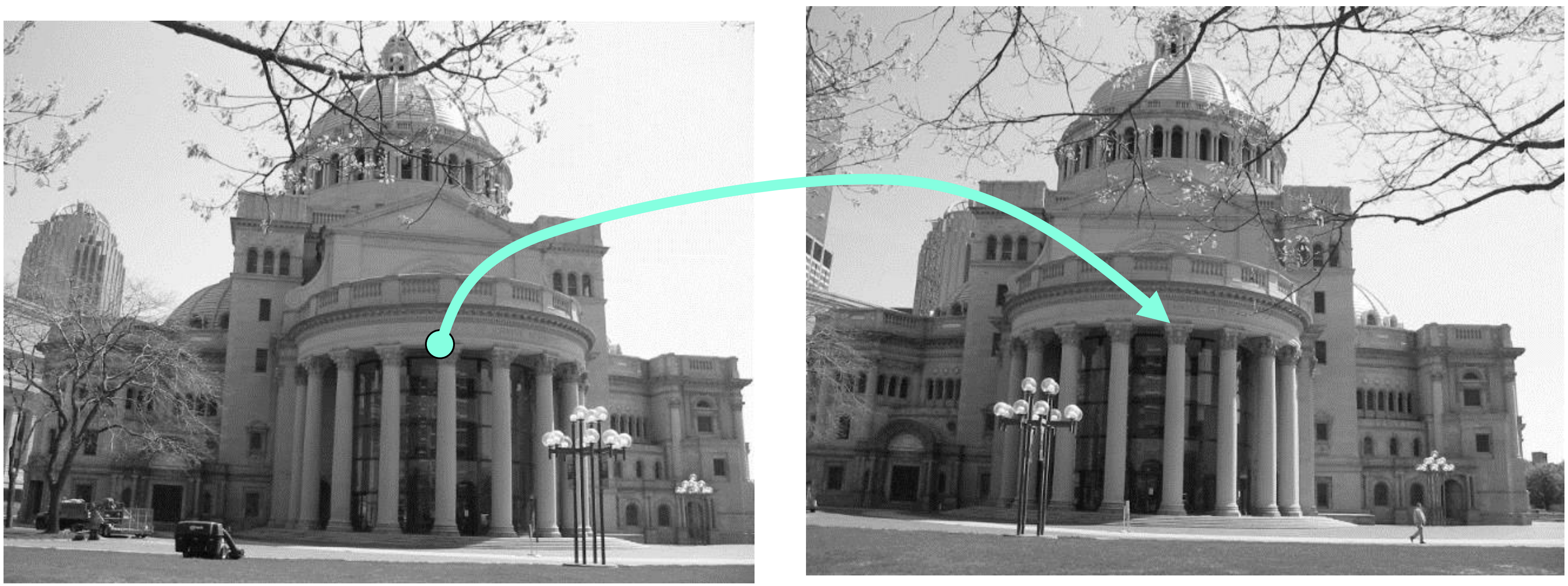
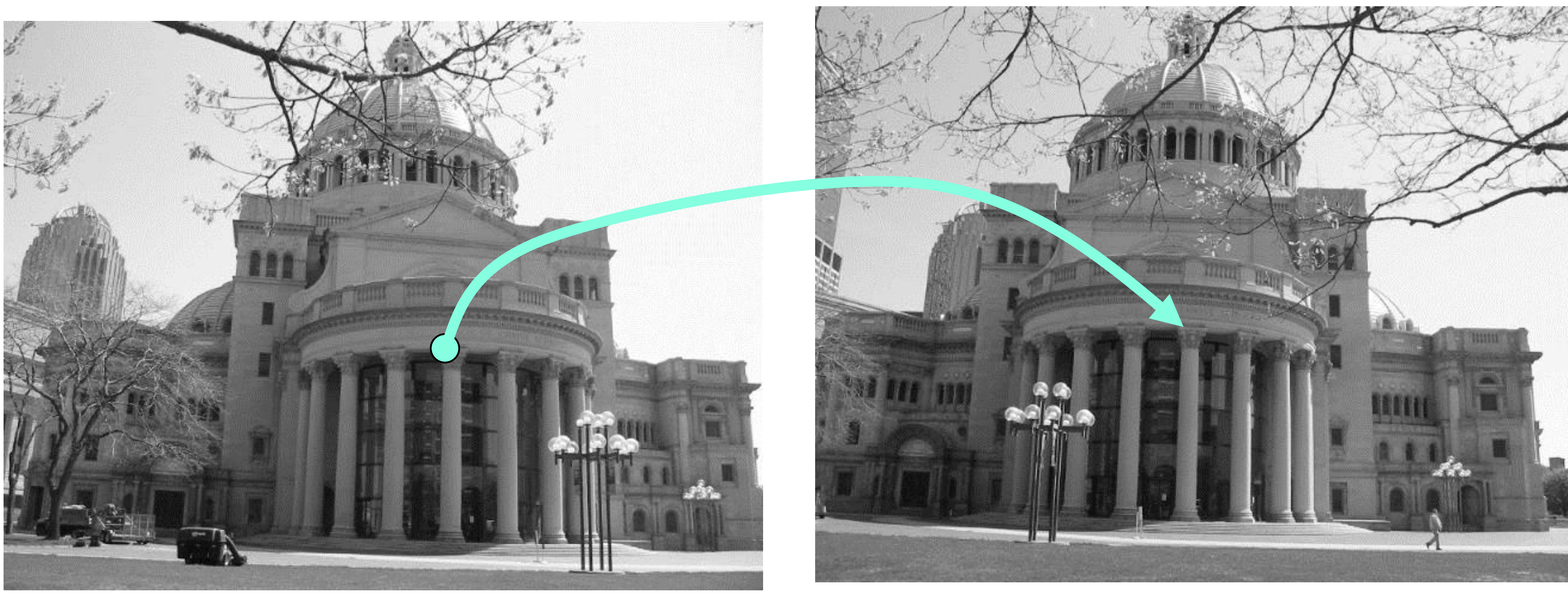


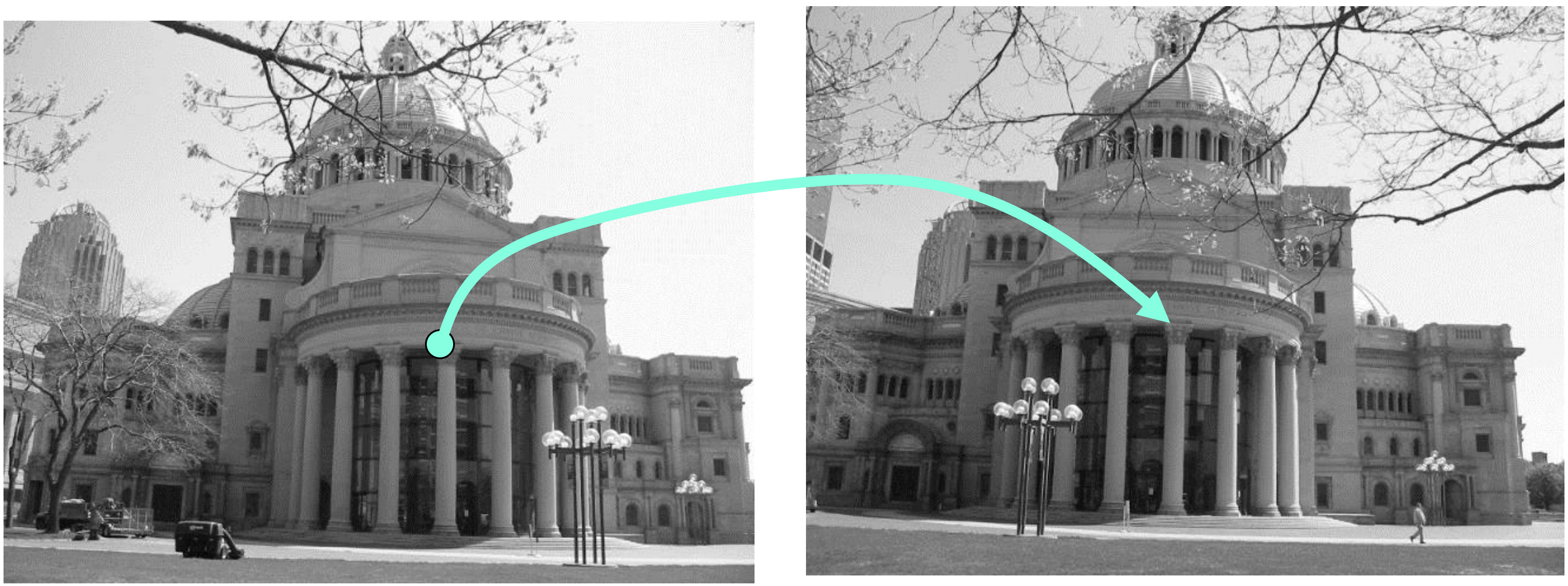
Image Correspondence

Goal: Find map between two images

- Sparse correspondences: map a small number of points
- Dense correspondences: map all points



Applications?



Applications?

Attribute transfer

Mosaics (panoramas)

Motion tracking

3D reconstruction

Recognition

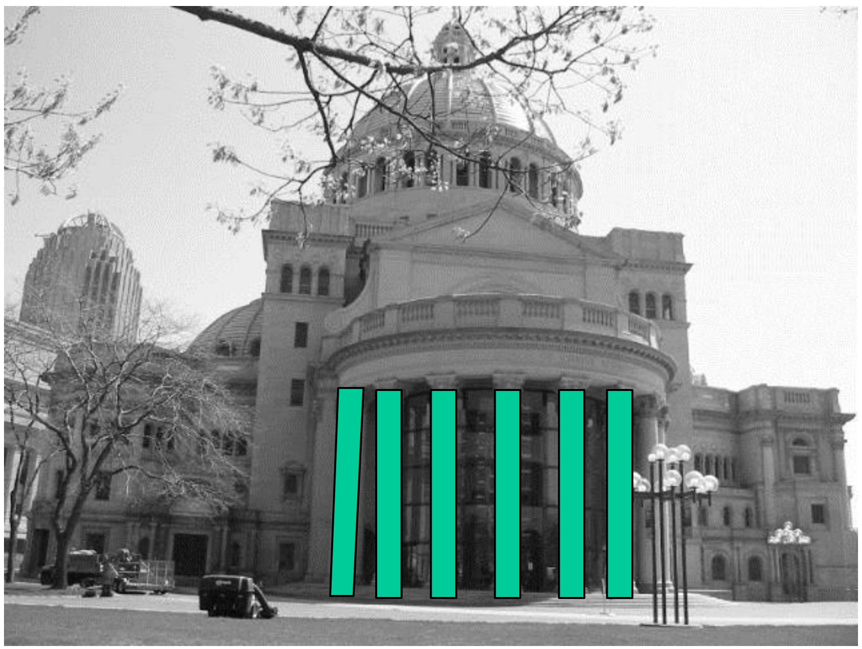
Wide baseline stereo

Mobile robot navigation

…

Attribute Transfer

Transfer properties (e.g., labels)
from one image to another



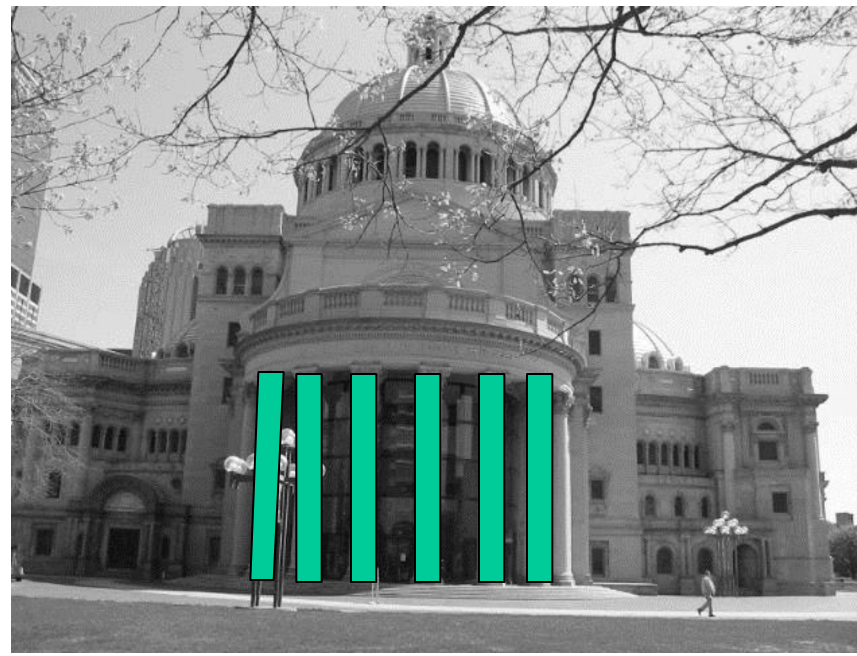
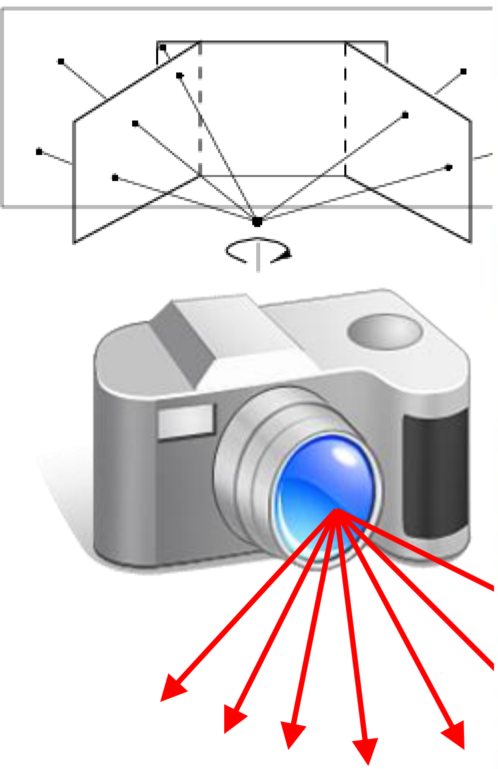


Image Mosaics (Panorama)

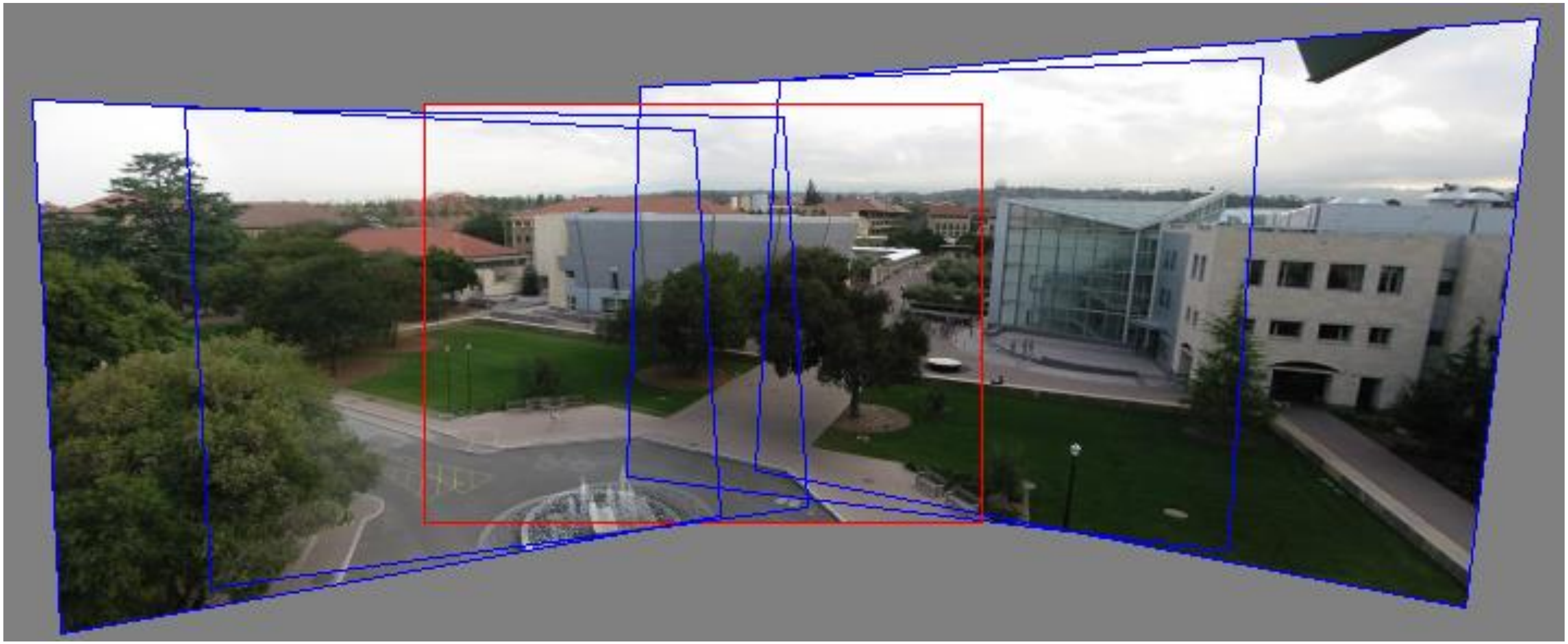


. . .

image from S. Seitz

Obtain a wider angle view by combining multiple images.

Image Mosaics (Panorama)



To construct an image mosaic, we need to find the homographies (projective transformations) that map images onto one another.

Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fal

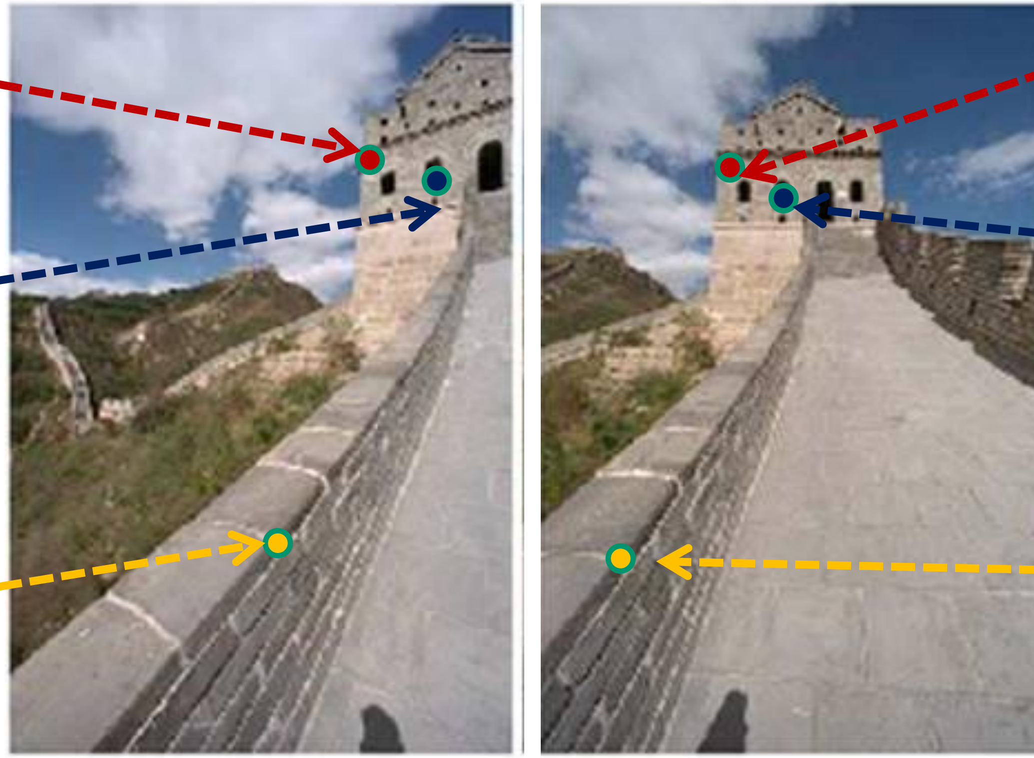
Mosaics: Finding a Homography

(x_1, y_1)

(x_2, y_2)

$\vdots$

(x_n, y_n)



(x'_1, y'_1)

(x'_2, y'_2)

$\vdots$

(x'_n, y'_n)

We can compute the homography between two images using sparse point correspondences

Mosaics: Finding a Homography

Homography (map) can be represented by a projection matrix H

p' = Hp

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Can set scale factor i=1.

So, there are 8 unknowns $[a,b,c,d,e,f,g,h]^T$

N correspondences provides a system of N linear equations:

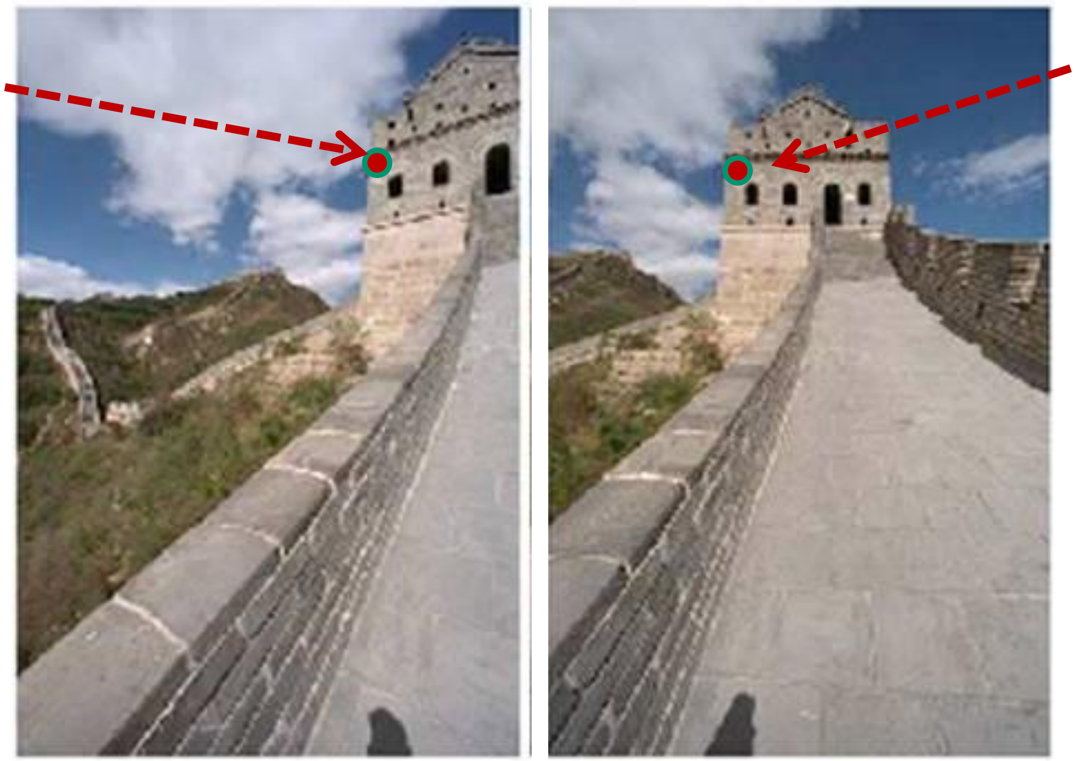
Can solve if have at least 8 eqs, but the more the better…

If overconstrained (N>8), solve using least-squares:

$$\min \; || Hp - p' ||^2$$

Mosaics: Finding a Homography

$$(x, y)$$



$$\left({}^{wx'}\!/_{w}, {}^{wy'}\!/_{w} \right) = (x', y')$$

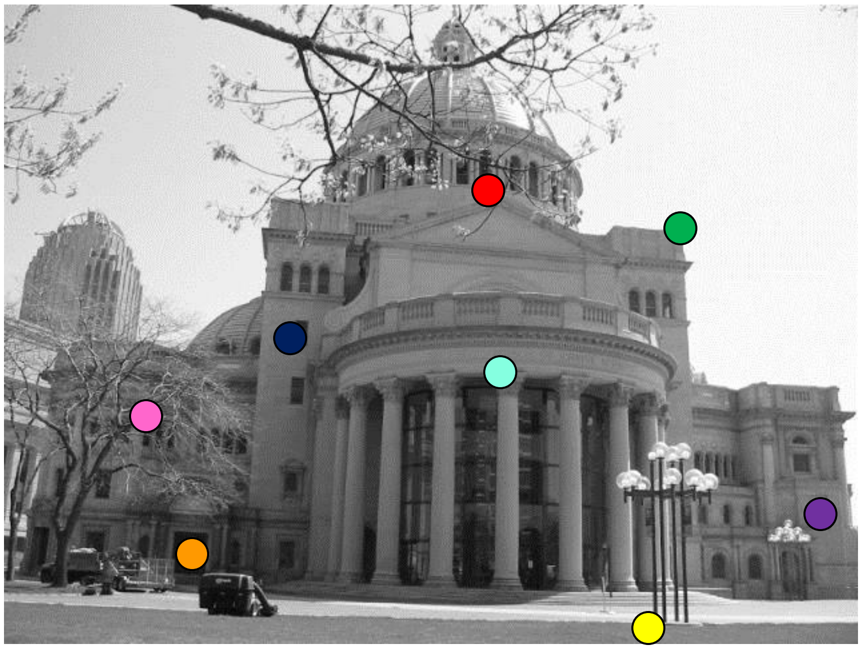
To compute a map from homography **H**

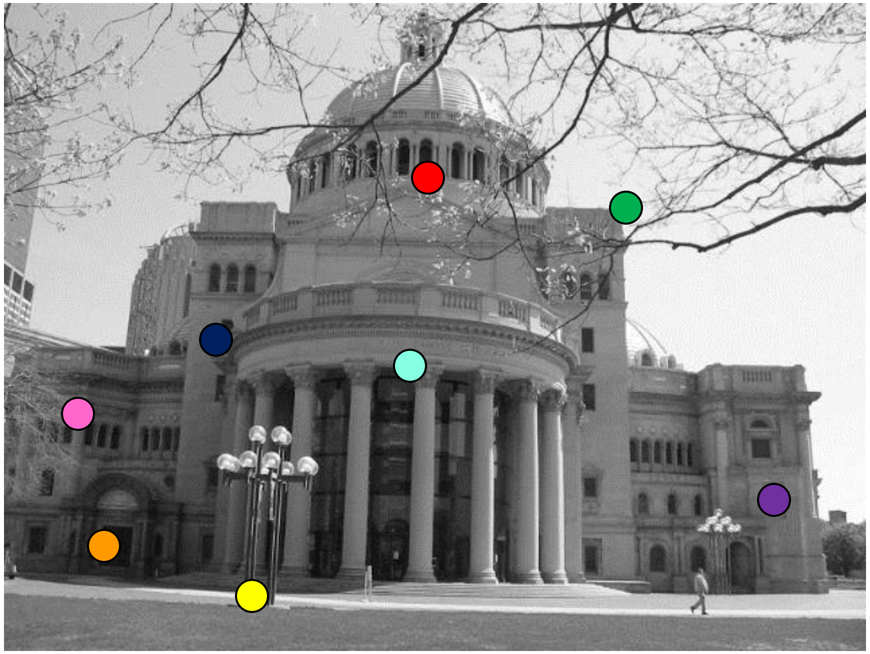
- Compute **p' = Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates

$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

Mosaics: Finding a Homography

So, computing a mosaic (panorama) requires finding a sparse (>=4) set of correspondences



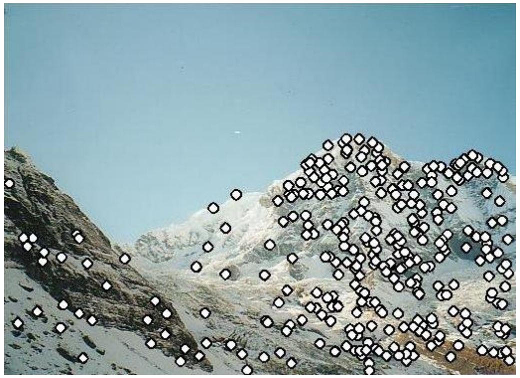


Finding Image Correspondences

How?

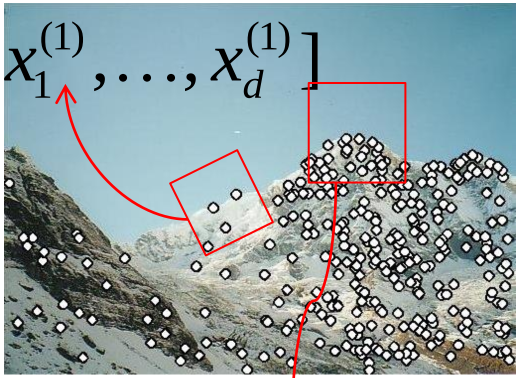
Matching Image Features

1) Feature Detection:
   Identify image features



2) Feature Description:
   Extract feature descriptor for each feature

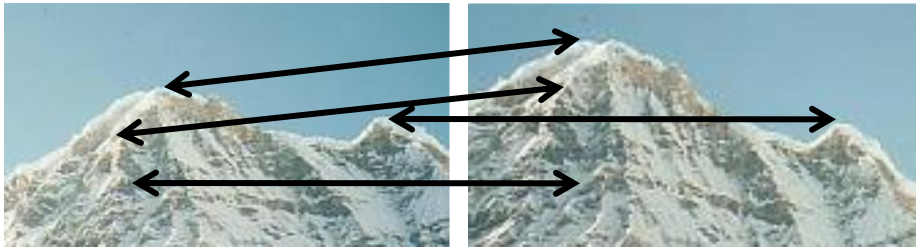
$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



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

3) Feature Matching:
   Find candidate matches between features

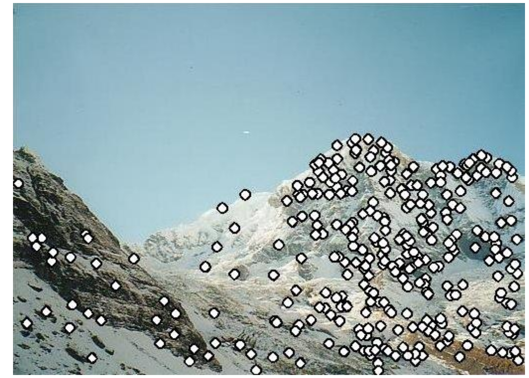
4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features

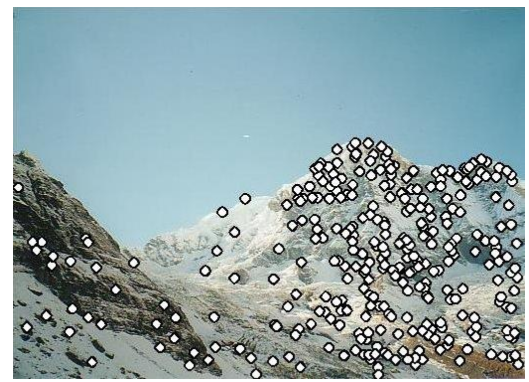


Kristen Grauman

Matching Image Features

1) Feature Detection:
   Identify image features
2) Feature Description:
   Extract feature descriptor for each feature
3) Feature Matching:
   Find candidate matches between features
4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features





Kristen Grauman

Feature Detection

Goals:

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Distinctiveness
  - Each feature has a distinguishing description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Feature Detection: Repeatability

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

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

Feature Detection: Repeatability

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

No chance to find true matches!

- Features should appear at “stable” locations that are invariant to typical image variations

Feature Detection: Distinctiveness

- We want to be able to reliably determine which feature goes with which

- Features should appear at locations with distinctive appearances

Propose a feature detection method?

Feature Detection

Some feature point detection methods:

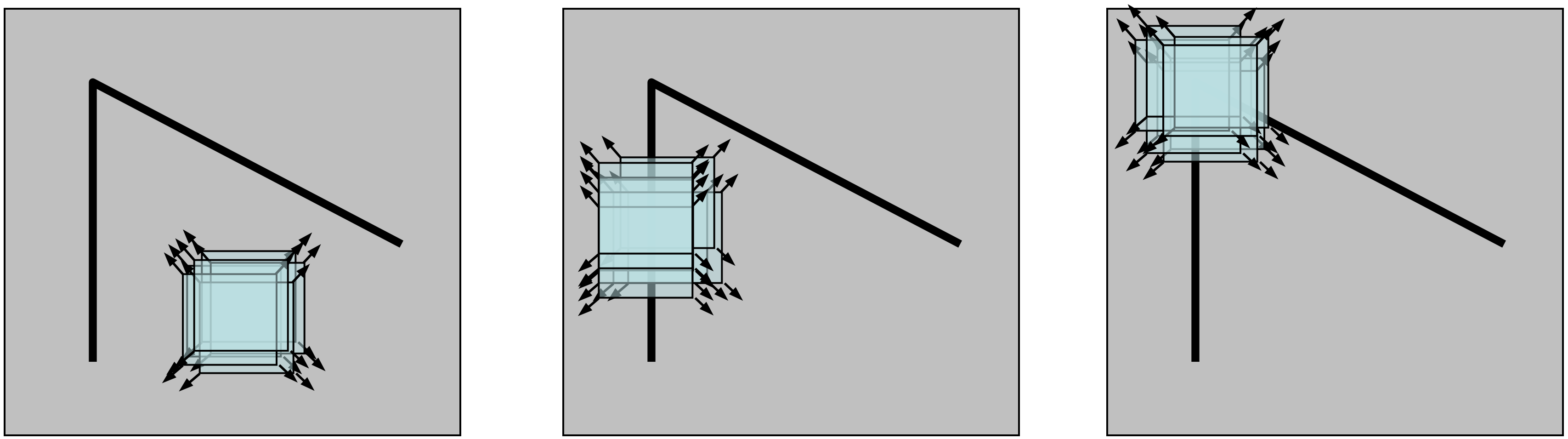
- Corners
  - Harris
- Scale-space blobs
  - SIFT

Feature Detection

Some feature point detection methods:

- Corners ←
  - Harris
- Scale-space blobs
  - SIFT

Corner Detector: Intuition



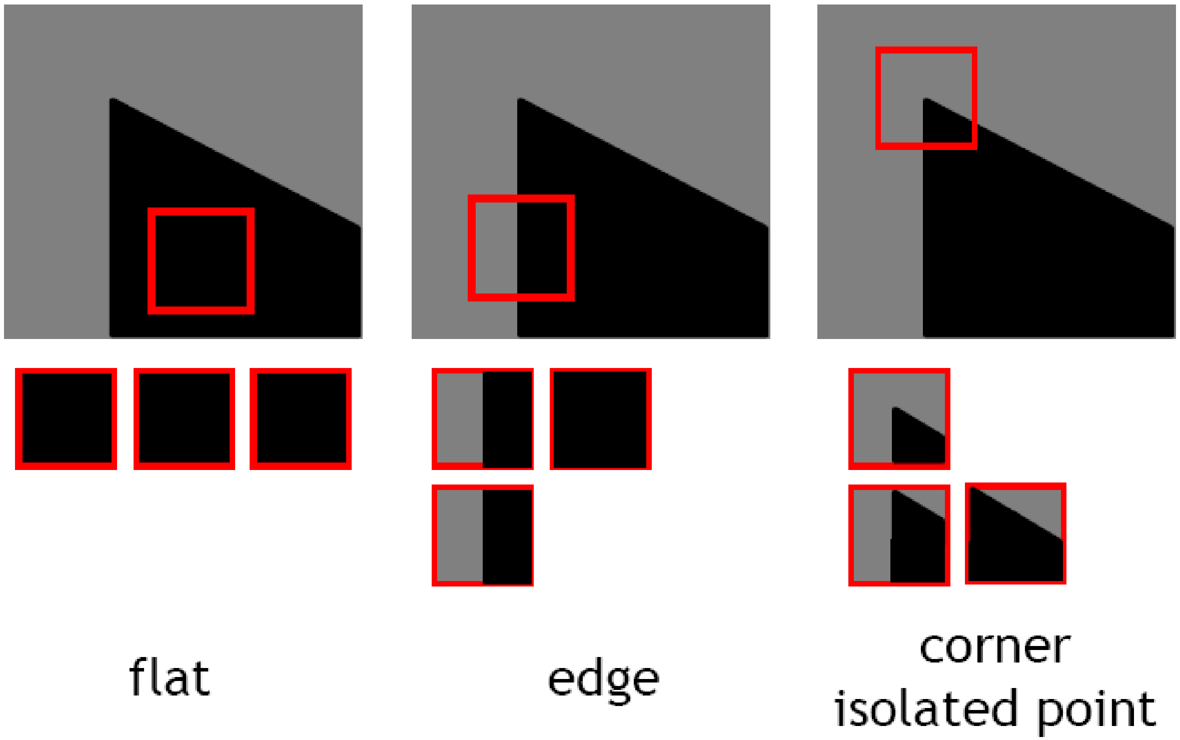
"flat" region:
no change in all
directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Moravec Corner Detector

Shift in any direction would result in a significant change at a corner.



Algorithm:

- Shift in horizontal, vertical, and diagonal directions by one pixel.
- Calculate the absolute value of the MSE for each shift.
- Take the minimum as the cornerness response.

Harris Corner Detector

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

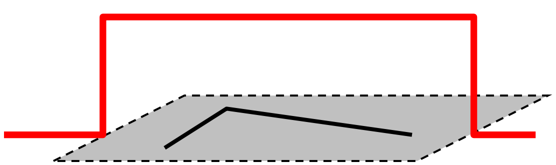
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

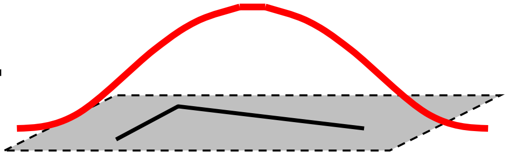
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Harris Corner Detector

Apply Taylor series expansion:

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

$$= \sum_{x,y} w(x,y)\left[I_x u + I_y v + \cancel{O(u^2, v^2)}\right]^2$$

$$E(u,v) = Au^2 + 2Cuv + Bv^2$$

$$A = \sum_{x,y} w(x,y) I_x^2(x,y)$$

$$B = \sum_{x,y} w(x,y) I_y^2(x,y)$$

$$C = \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y)$$

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & C \\ C & B \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Harris Corner Detector

For small shifts [*u*,*v*] we have the following approximation:

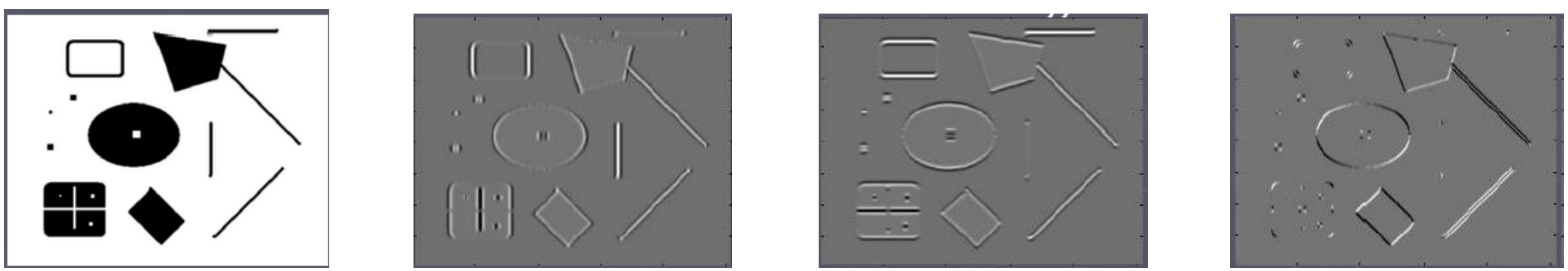
$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

Harris Corner Detector

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



Notation:

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

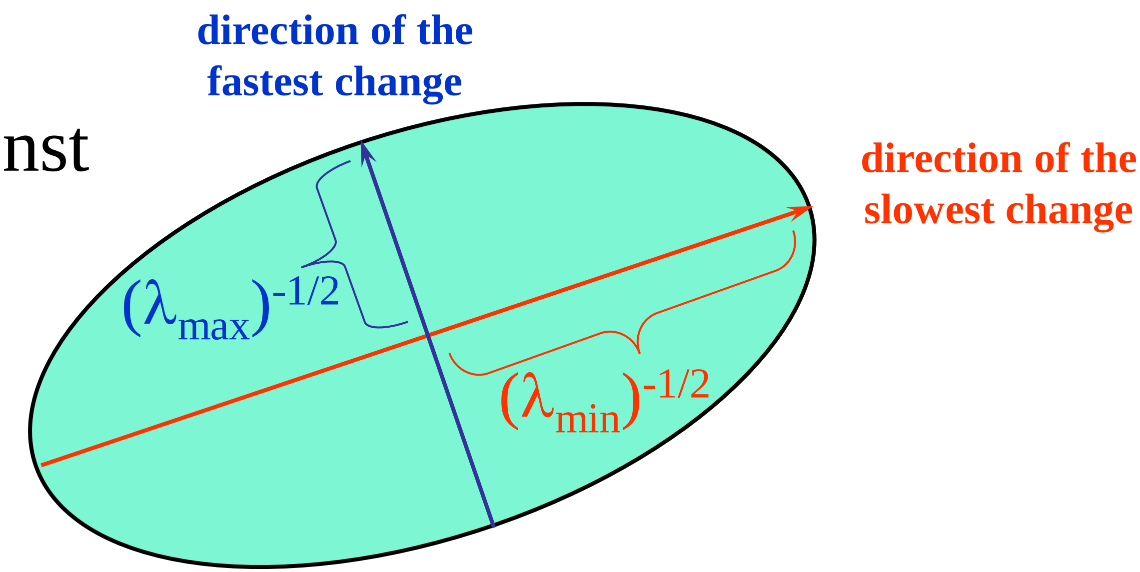
Harris Corner Detector

Intensity change in shifting window: eigenvalue analysis

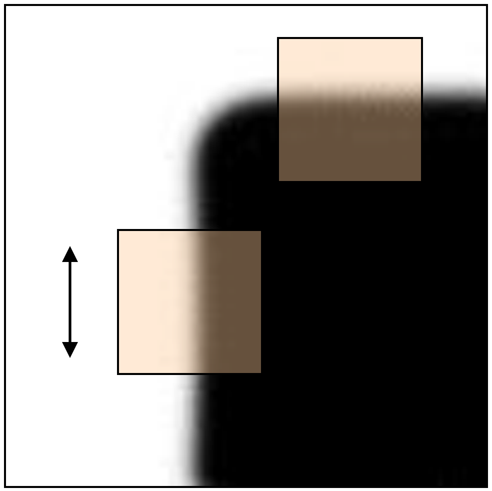
$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const



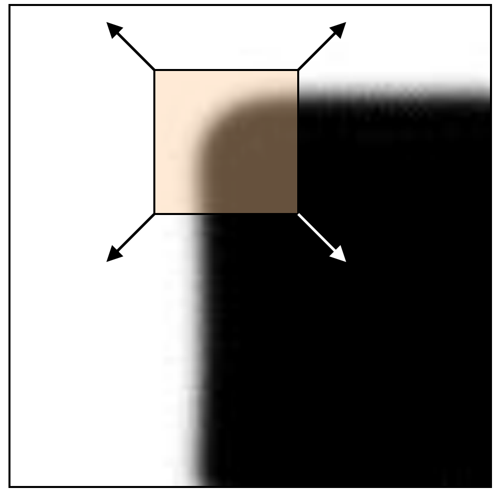
Harris Corner Detector



"edge":

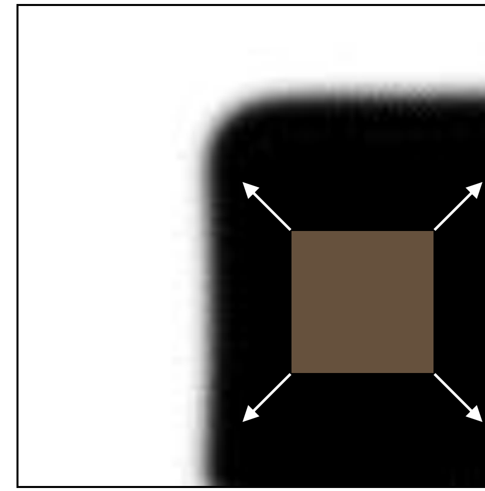
$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$



"corner":

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



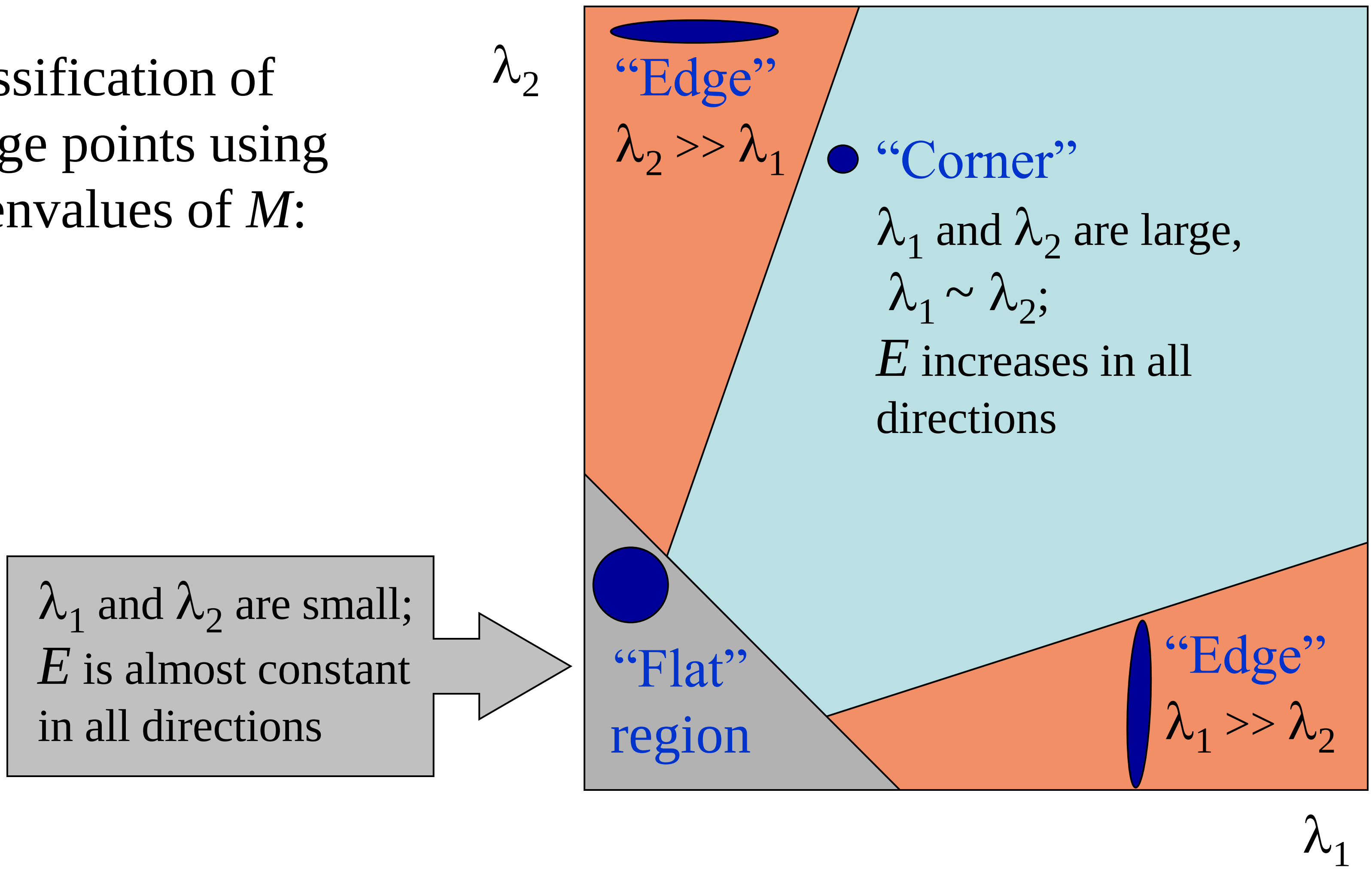
"flat" region

λ_1 and λ_2 are small;

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

Harris Corner Detector

Classification of image points using eigenvalues of M:

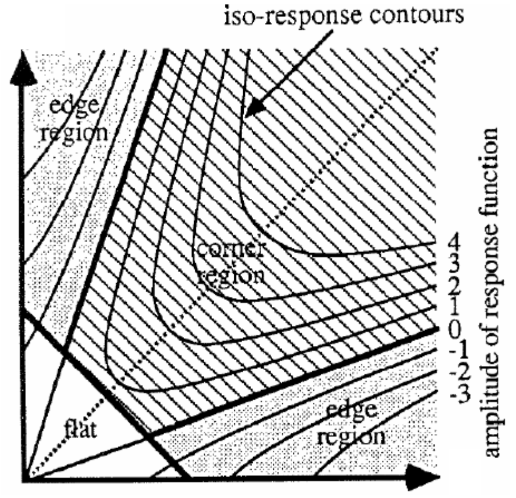


Harris Corner Response (R)

Approximation to eigenanalysis

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

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



(k – empirical constant, k = 0.04-0.06)

No need to compute eigenvalues explicitly!

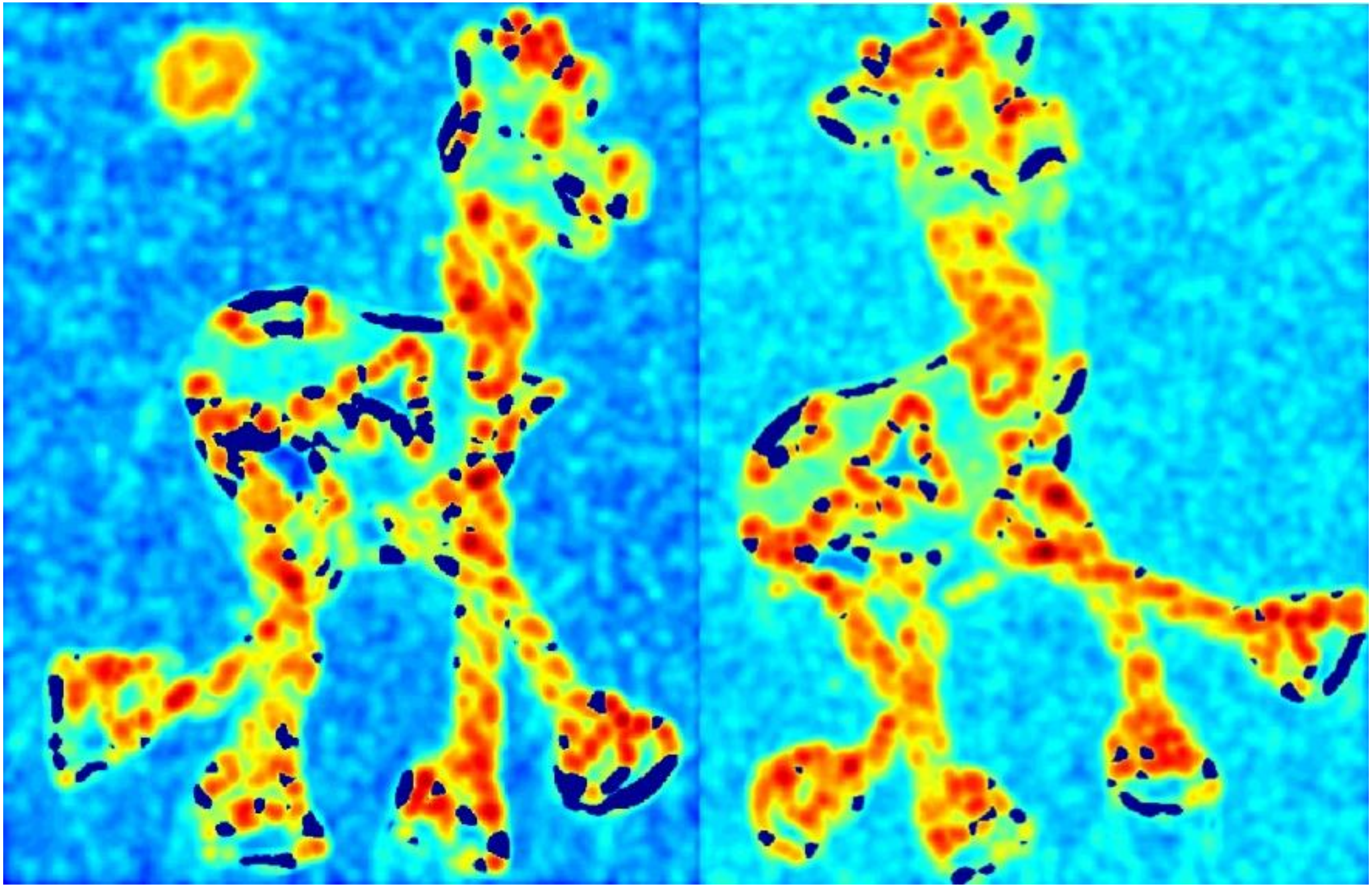
Harris Corner Detector

1) Compute *M* matrix for window around each pixel to get Harris corner responses (R).
2) Find points with large corner responses (R > threshold)
3) Remove points that are not local maxima of R within some neighborhood

Harris Corner Detector: Steps

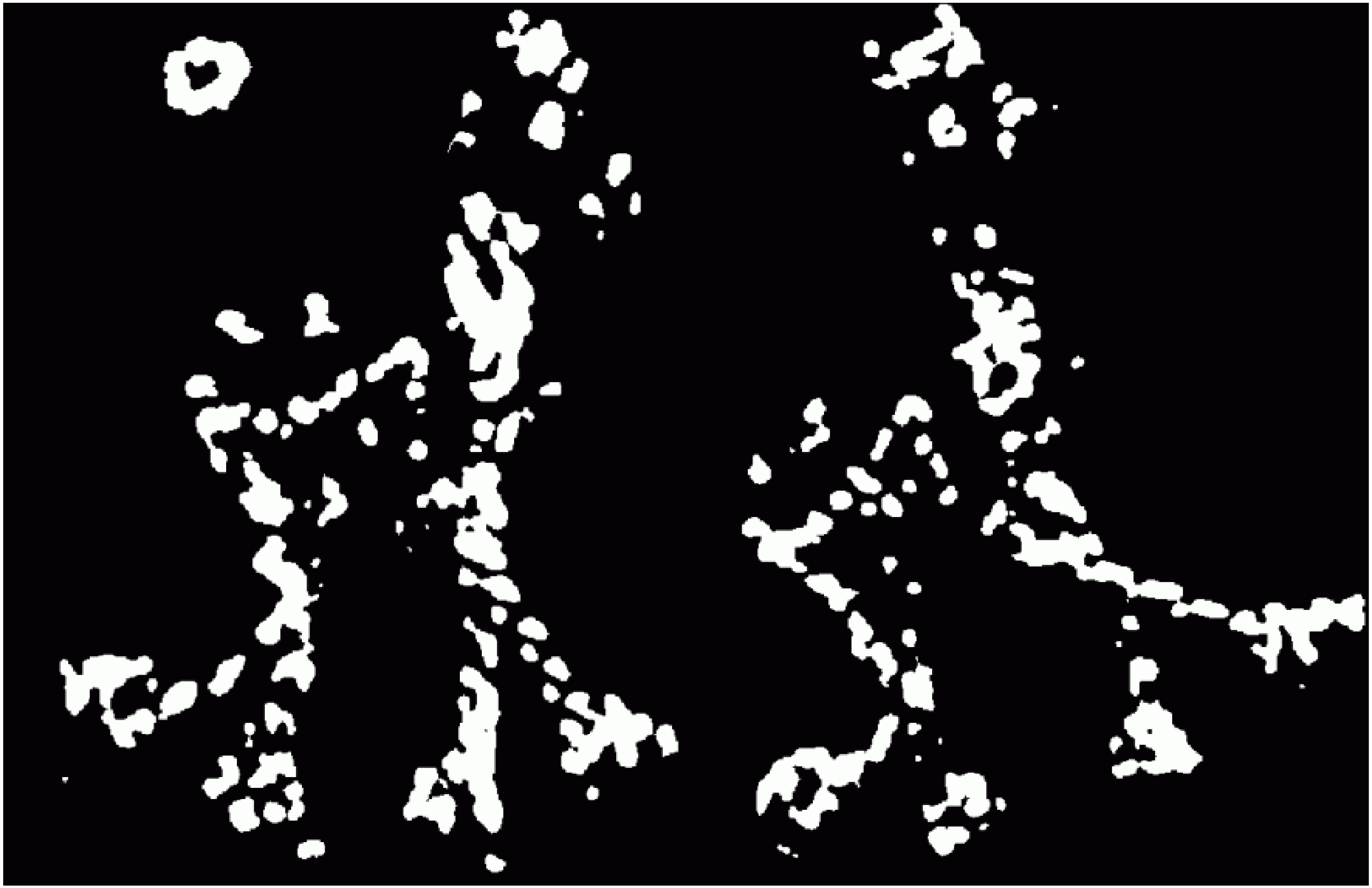
Harris Corner Detector: Steps

Compute corner response R



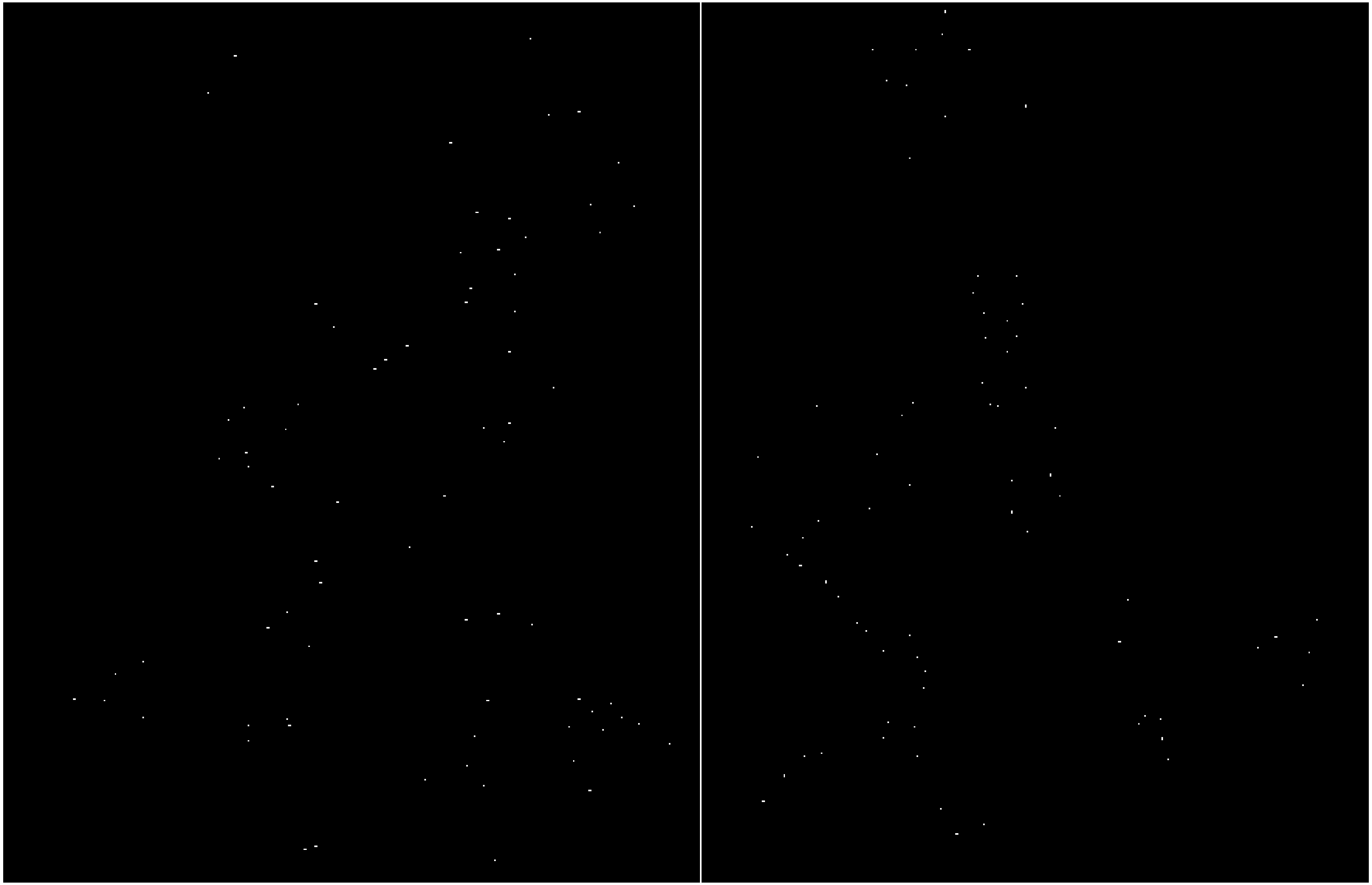
Harris Corner Detector: Steps

Find points with large corner response: R > threshold

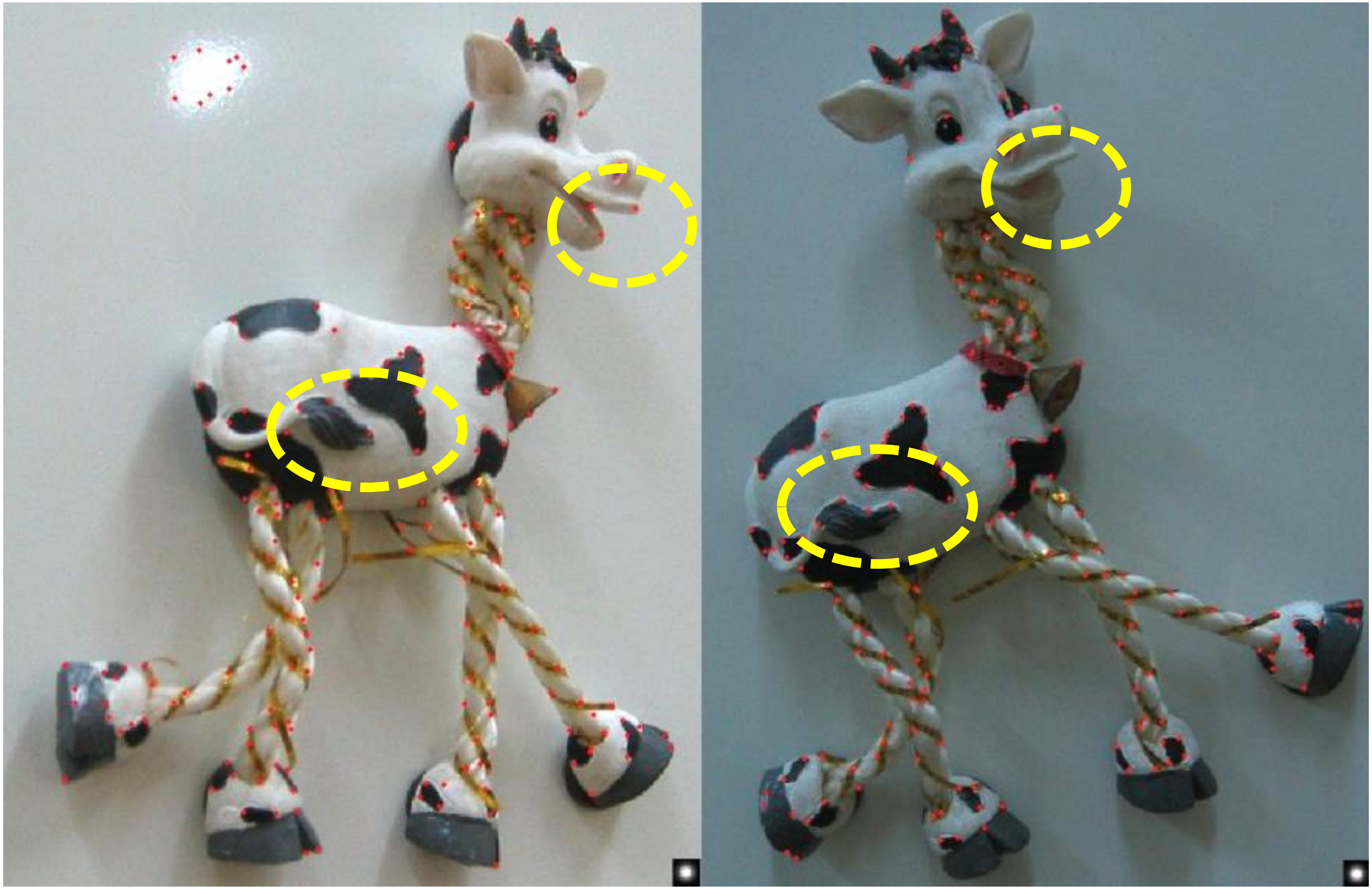


Harris Corner Detector: Steps

Take only the points of local maxima of R



Harris Corner Detector Result

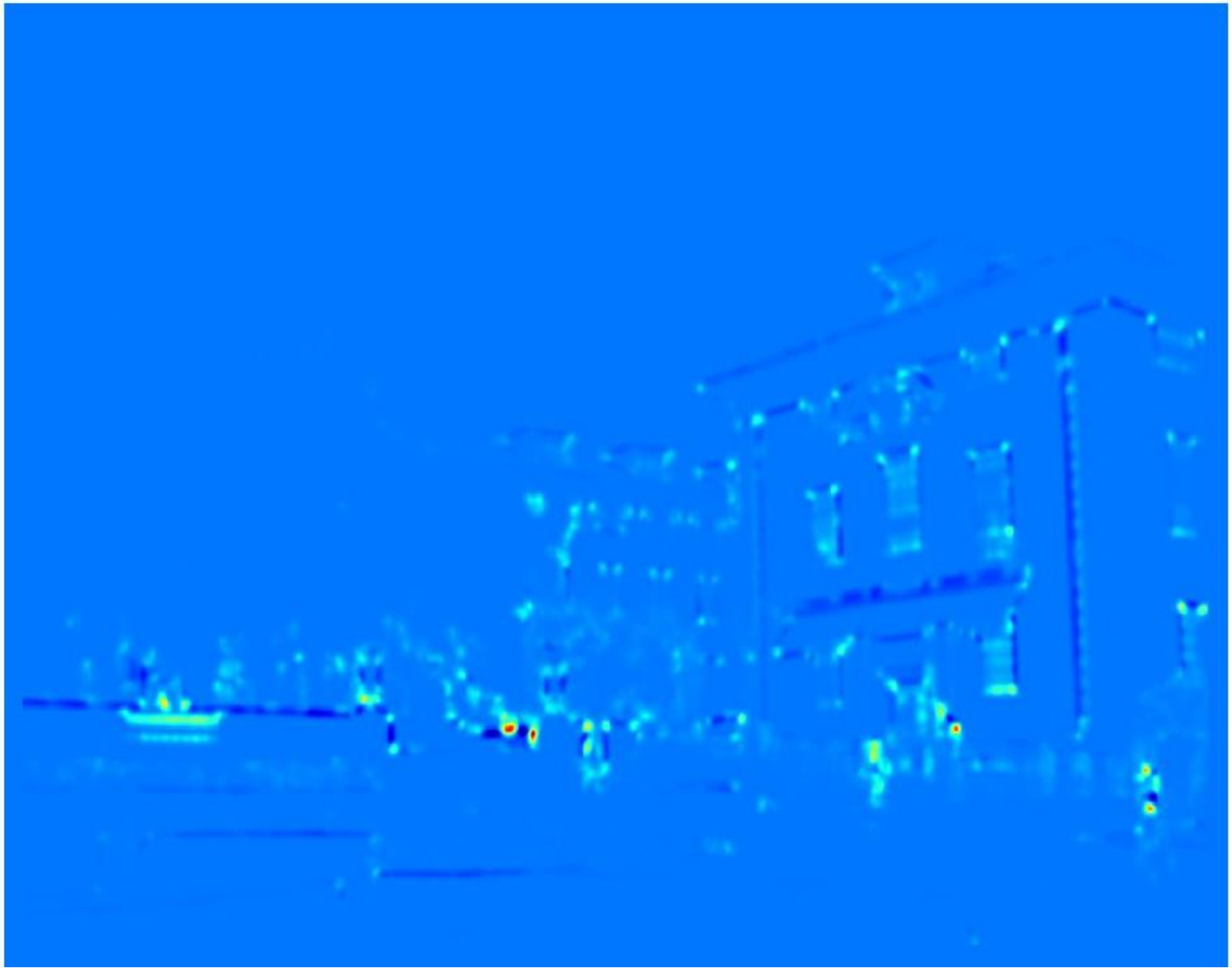


Another Harris Corner Detector Example

Kristen Grauman

Another Harris Corner Detector Example

Compute Harris corner response R at every pixel.



Kristen Grauman

Another Harris Corner Detector Example

Kristen Grauman

Properties of the Harris corner detector

Rotation invariant? Yes

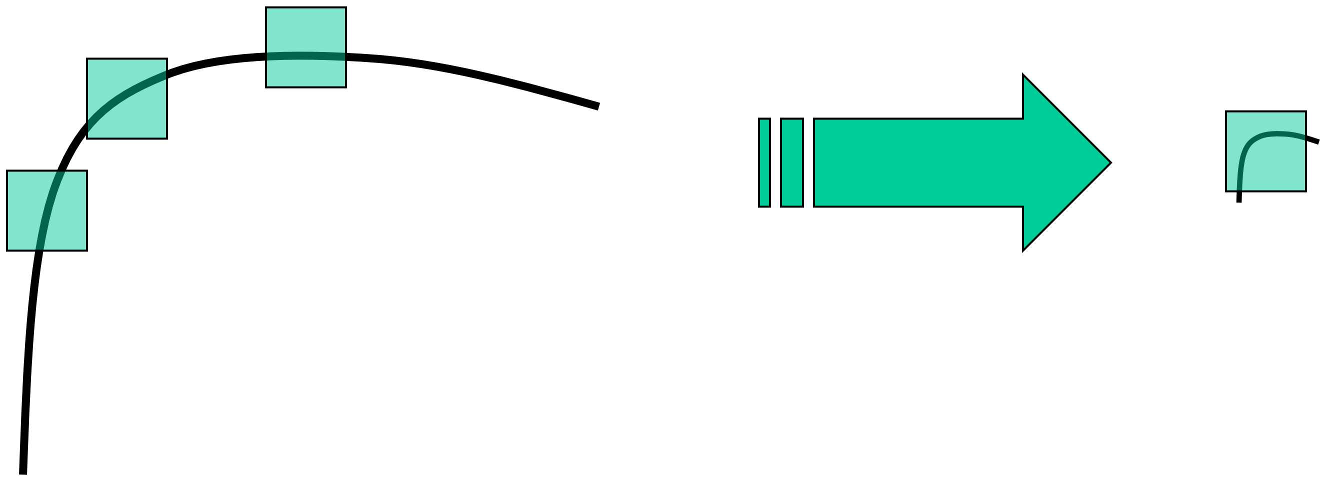
$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

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?

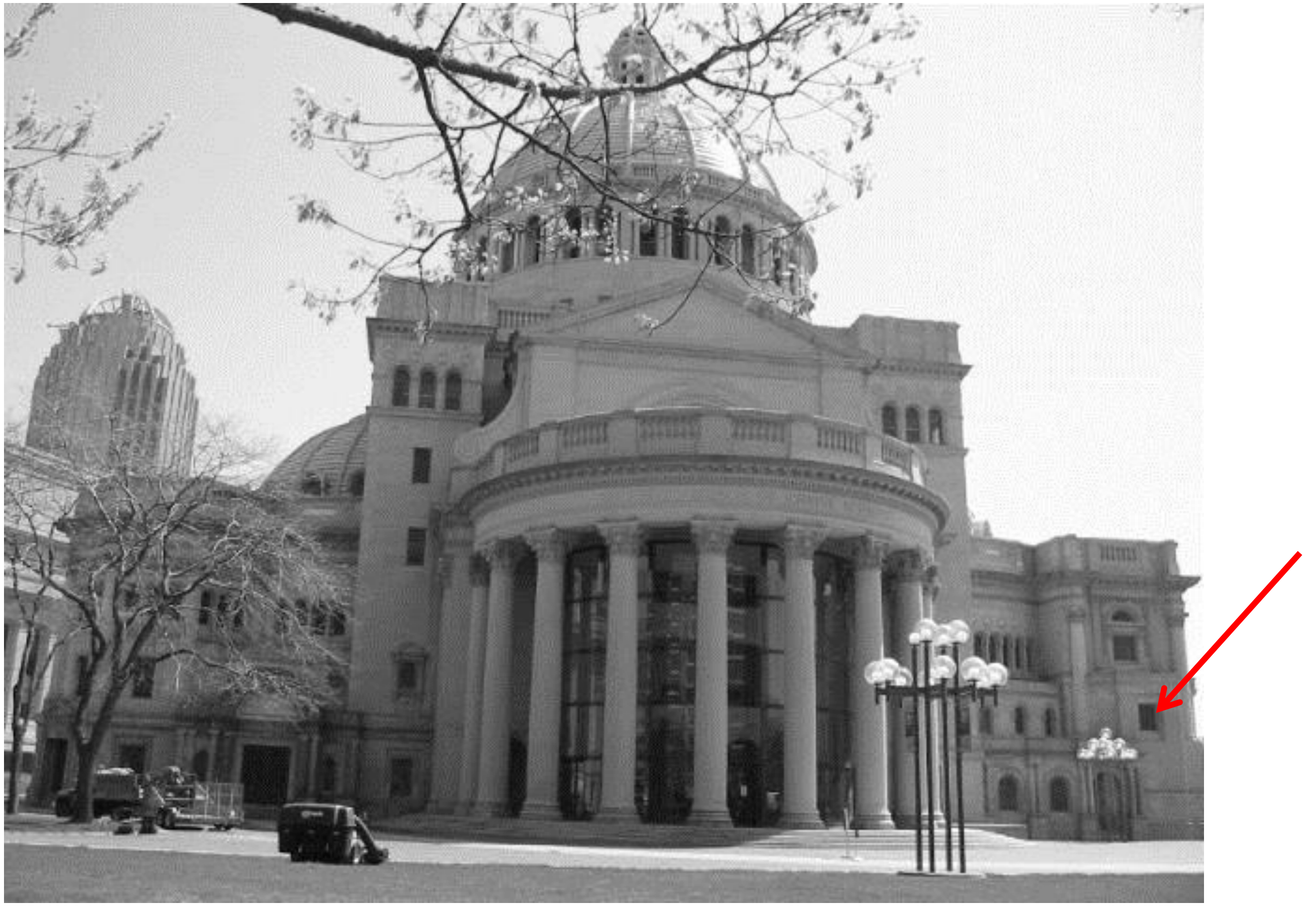
Feature Detection

Some feature point detection methods:

- Corners
  - Harris corner detector
- Scale-space blobs
  - SIFT

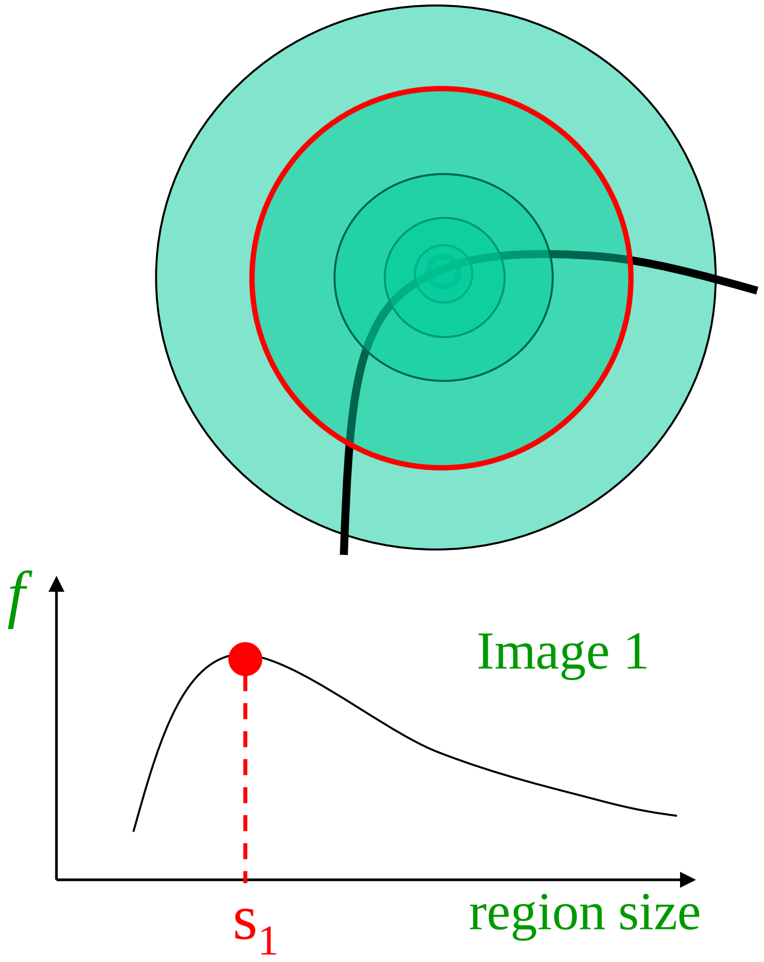
Blob detection

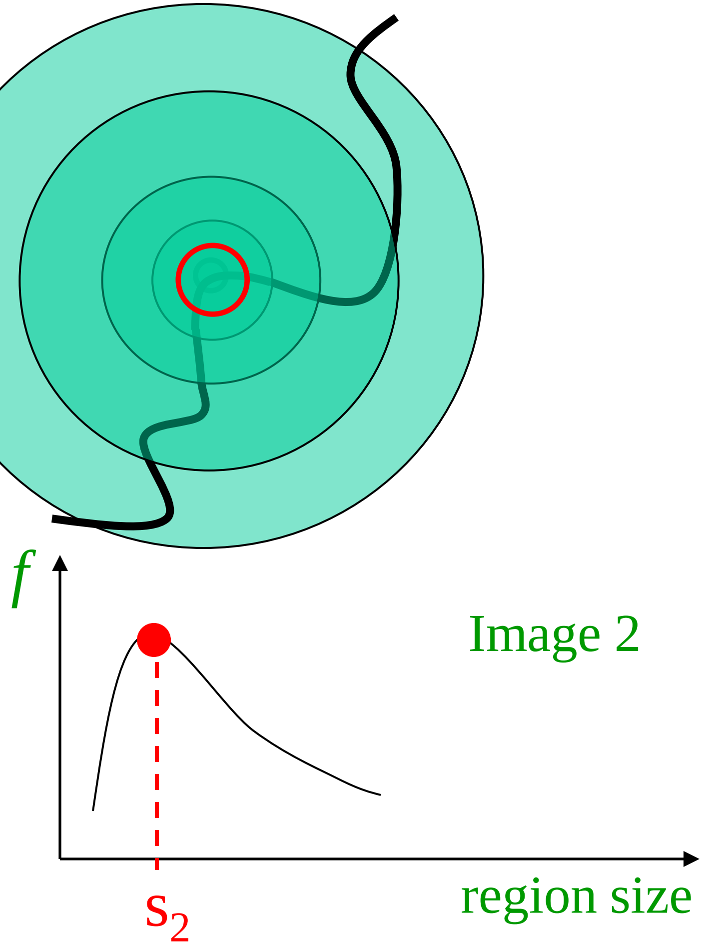
Intuition: centers of “blobs” provide stable feature points



Scale invariant interest points

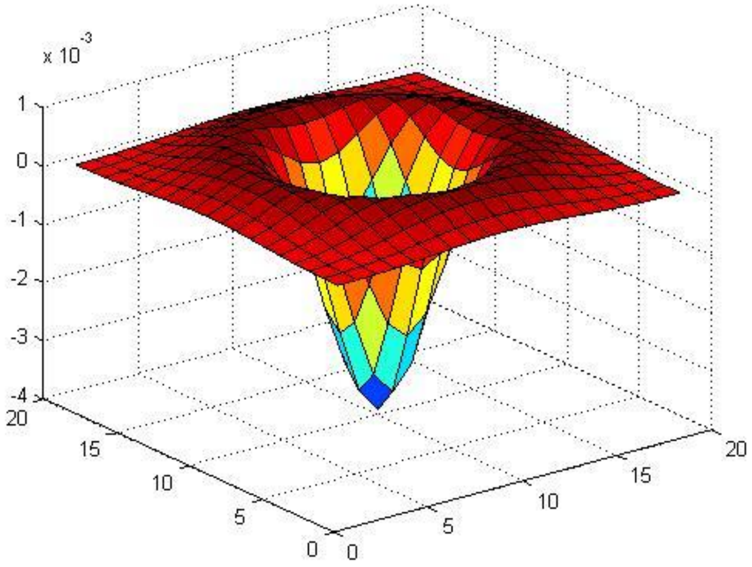
Intuition: size of blobs provide feature scale

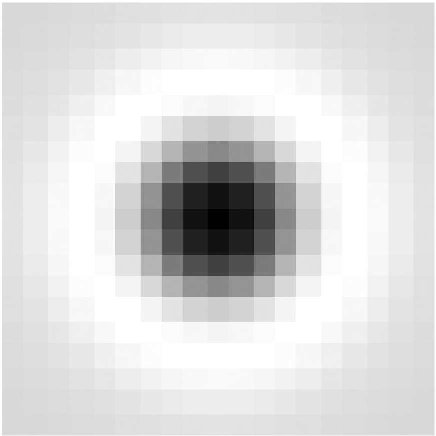




Blob detection

Laplacian of Gaussian: good operator for blob detection

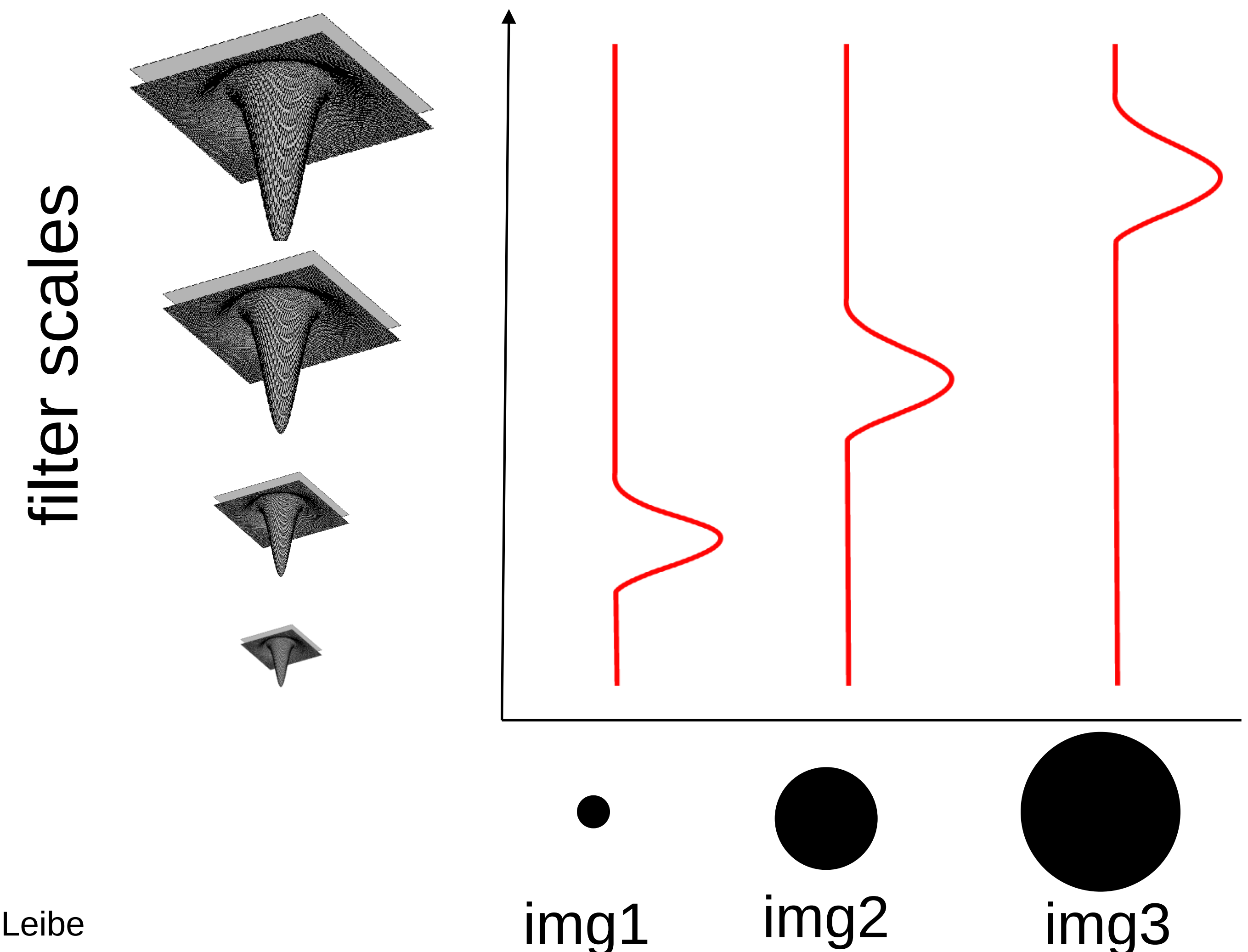




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

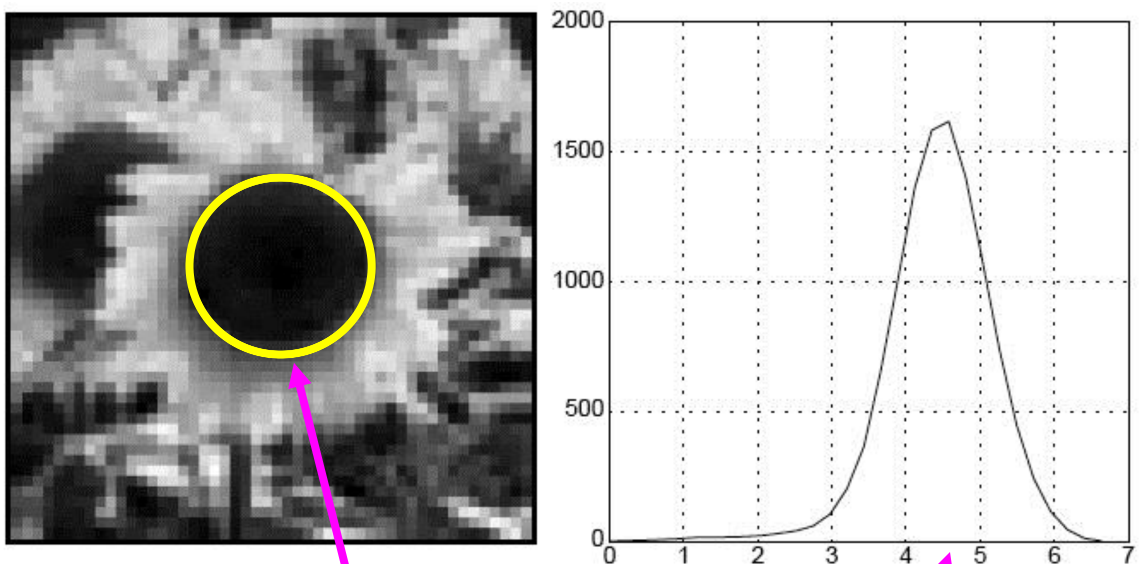
Blob detection: scale selection

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



Blob detection

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



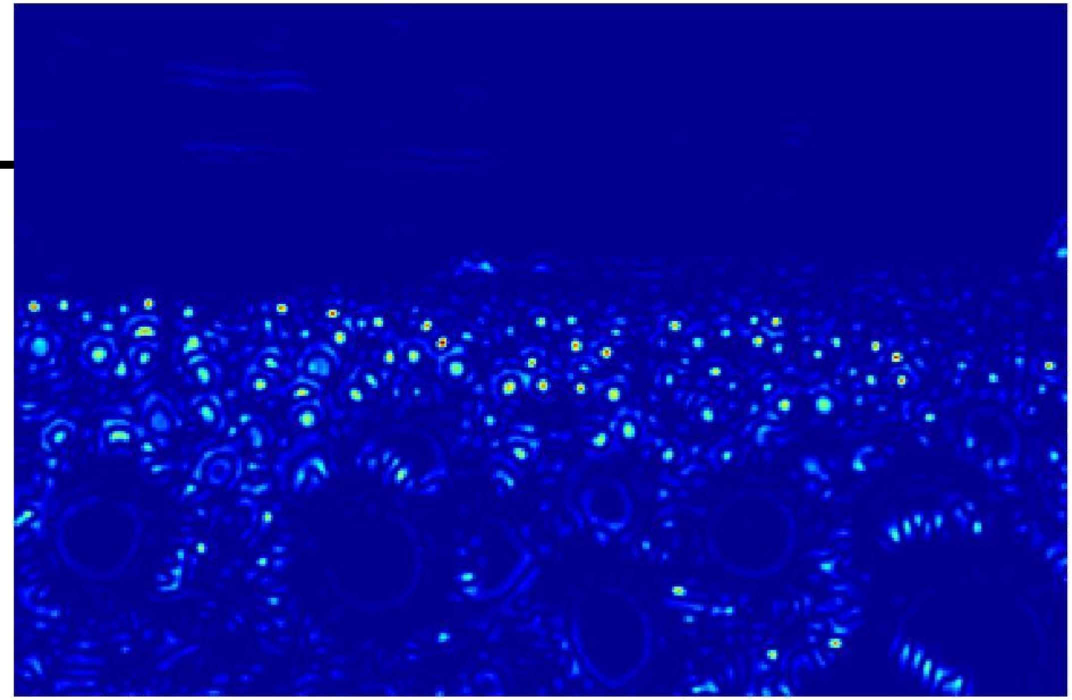
characteristic scale

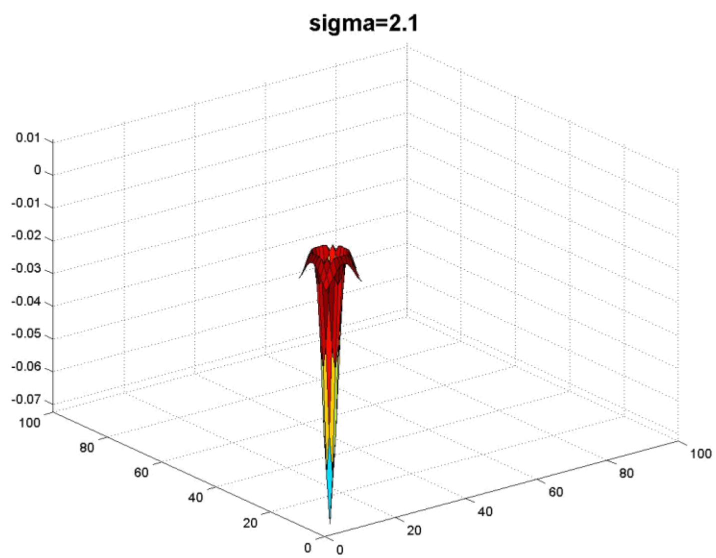
Example

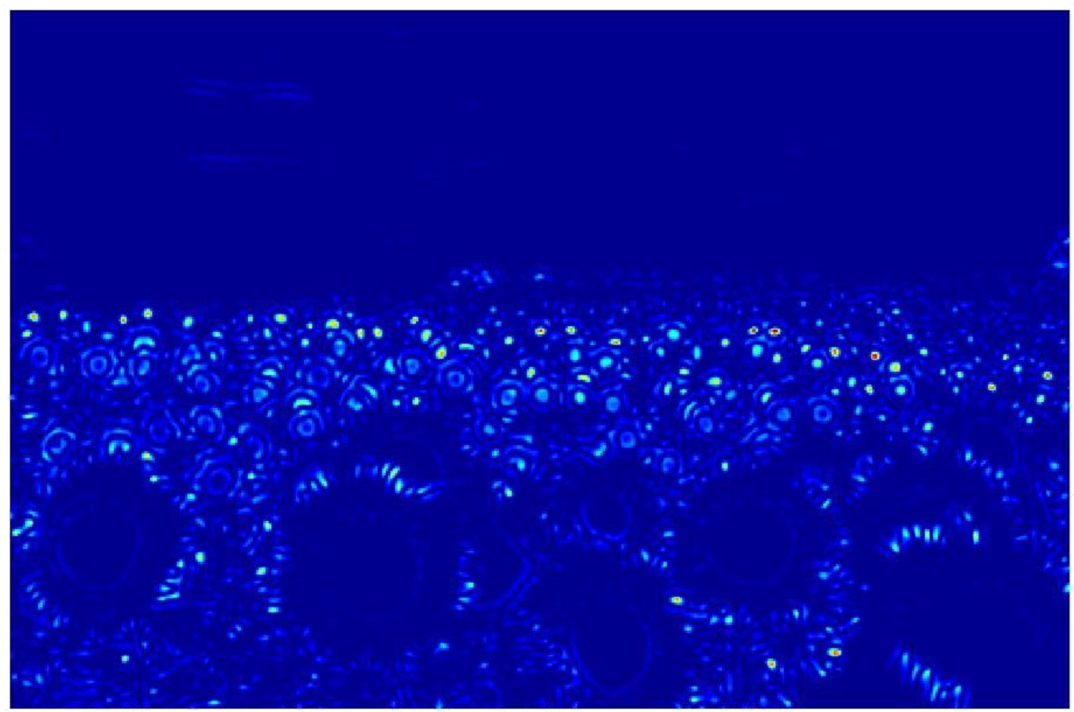
Original image at ¾ the size

Kristen Grauman

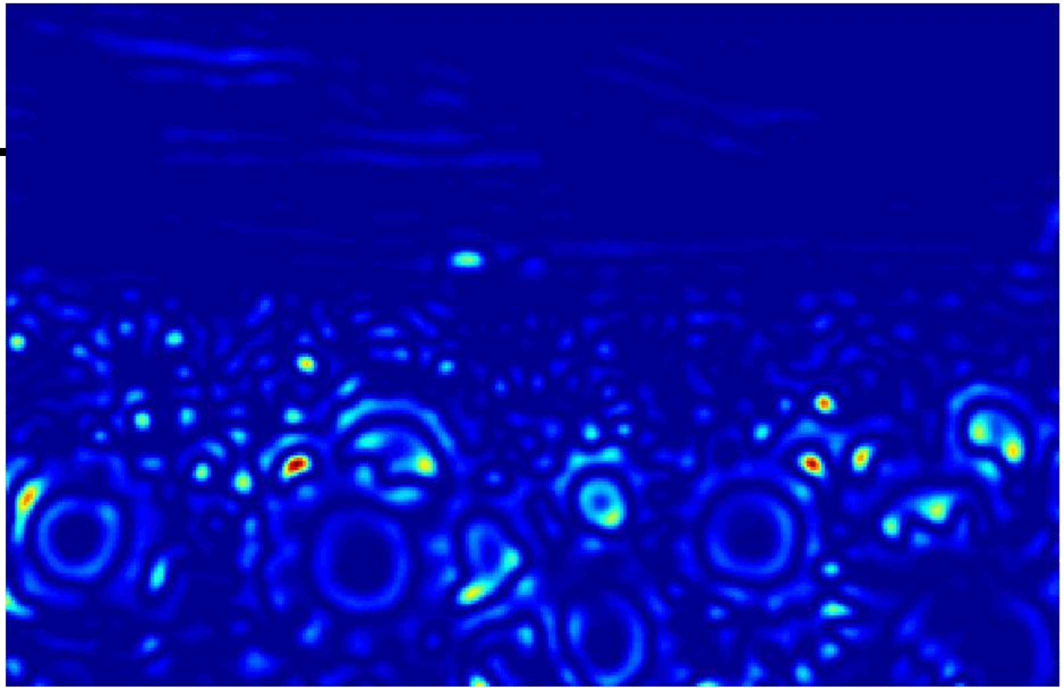
Original image
at ¾ the size

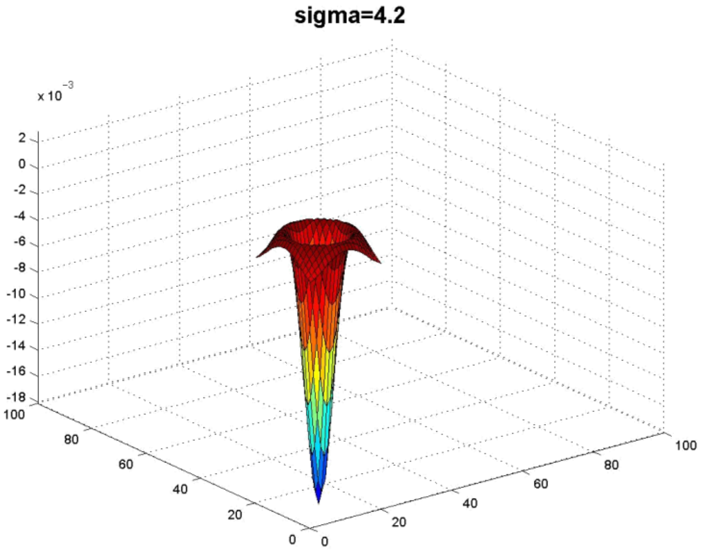


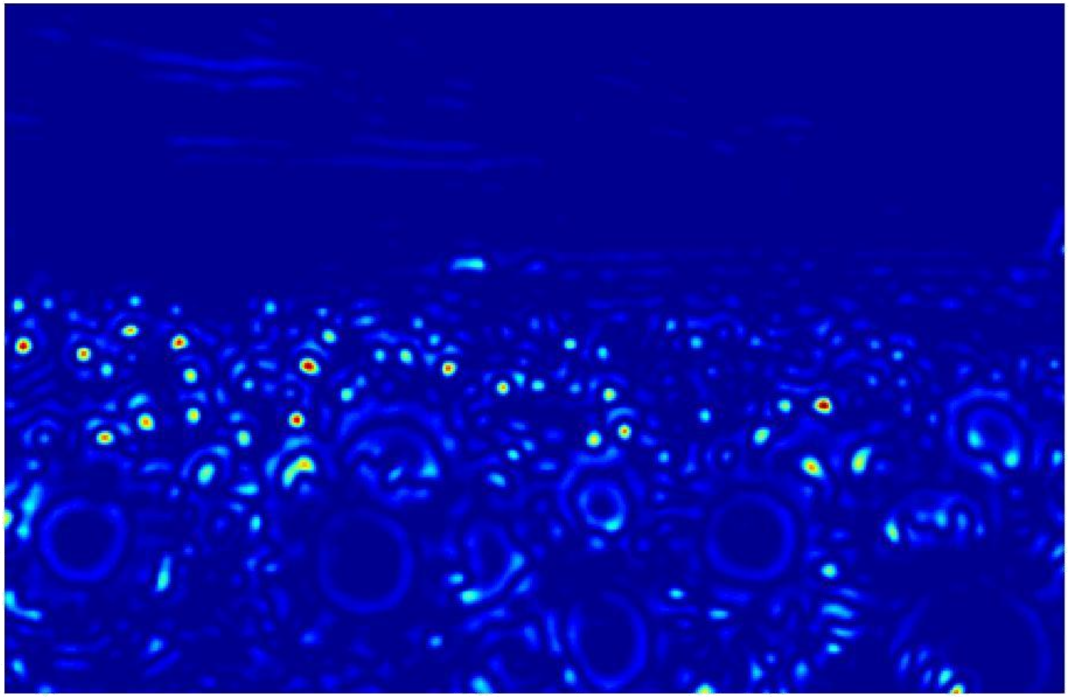




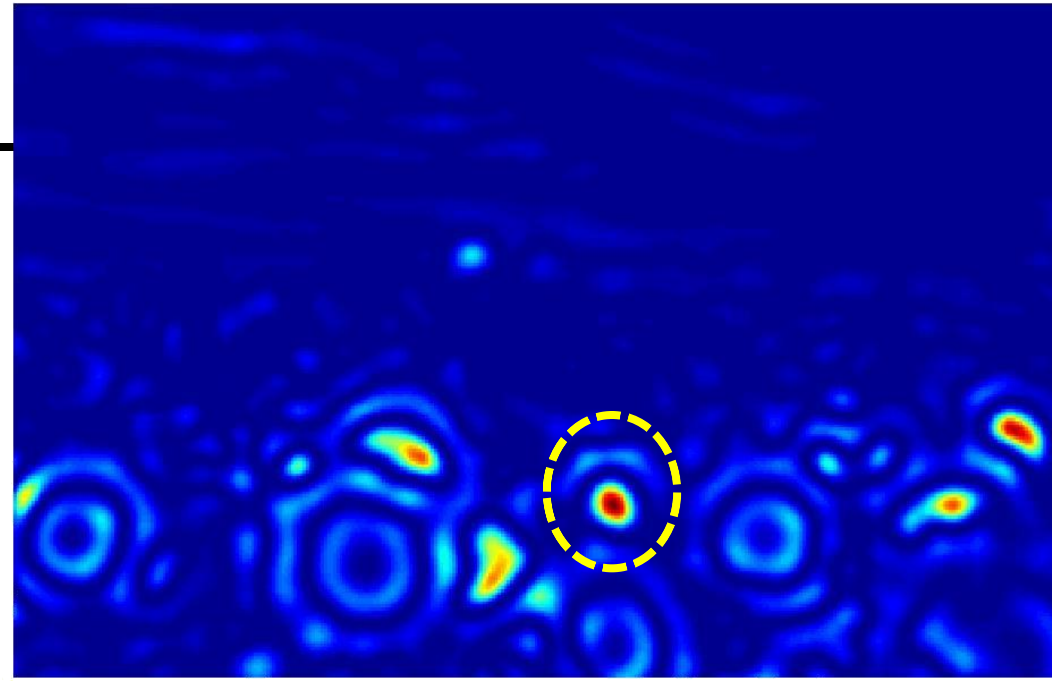
Kristen Grauman

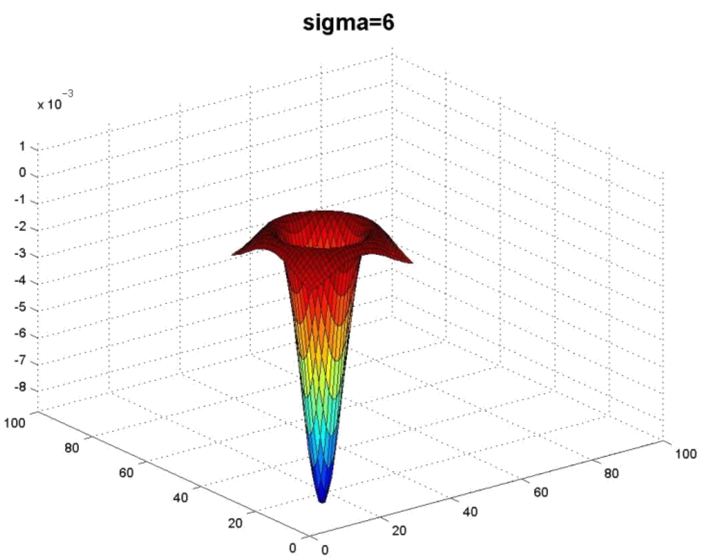


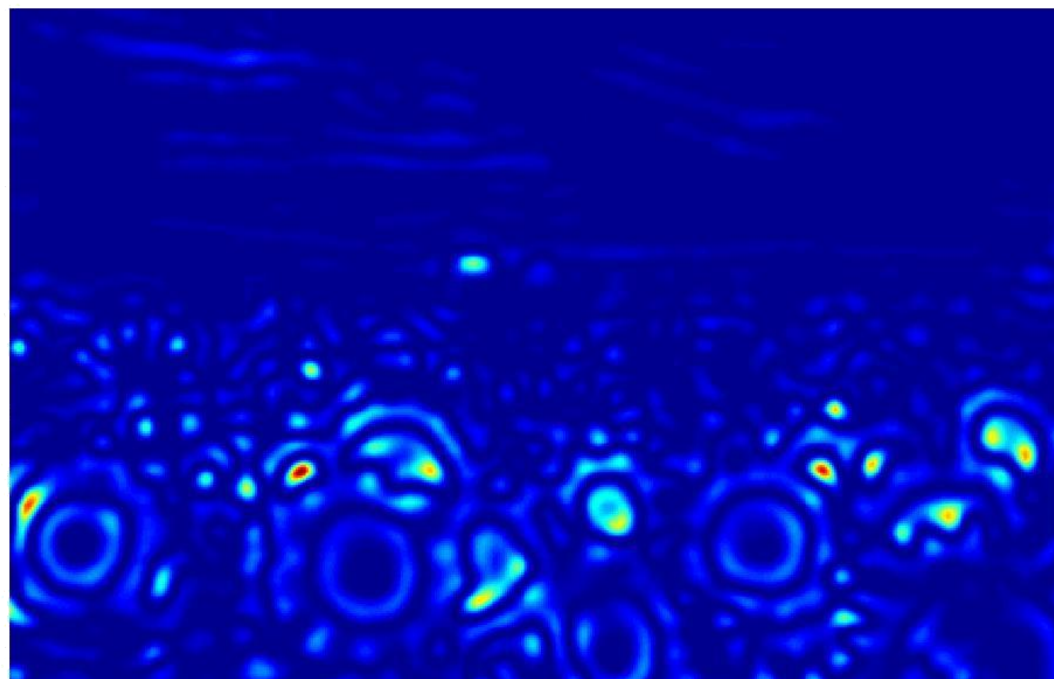




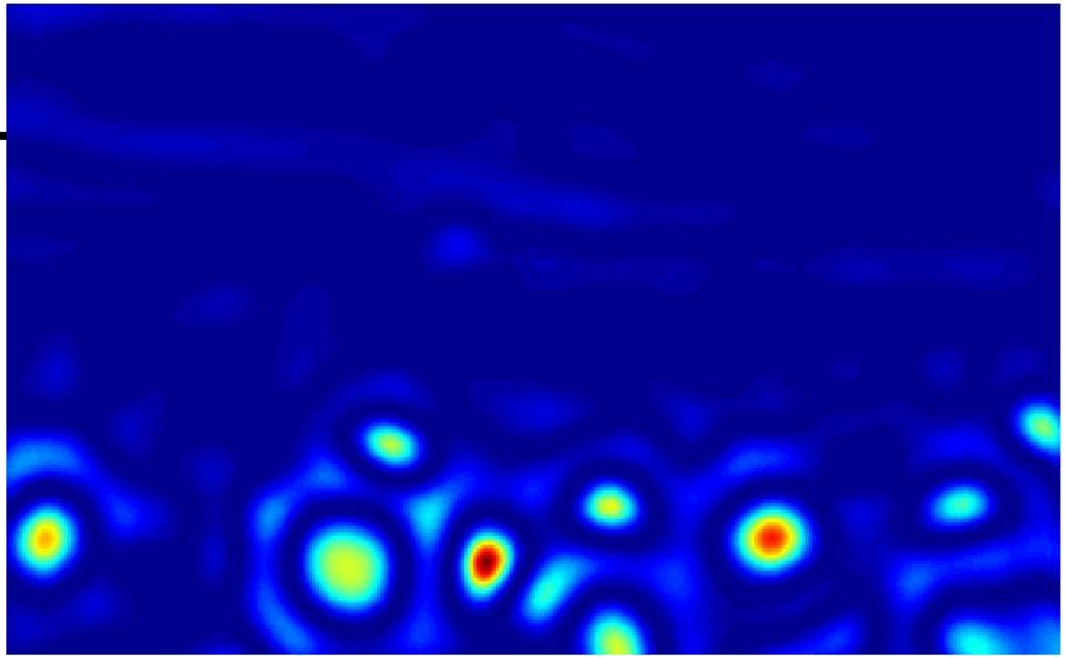
Kristen Grauman

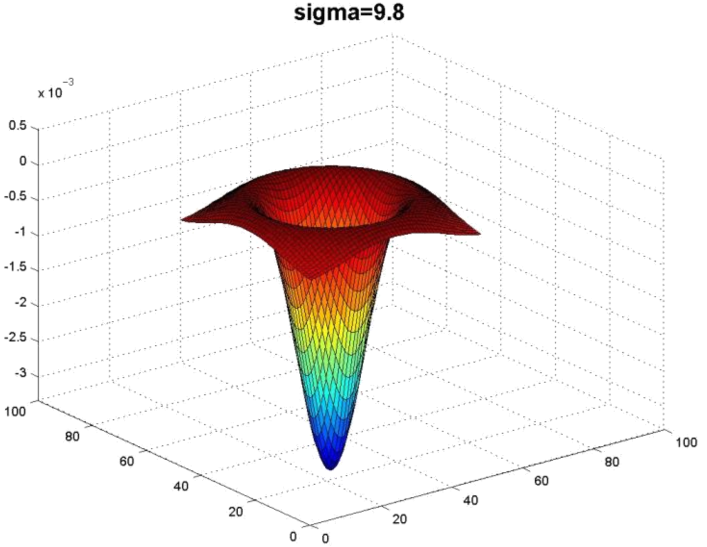


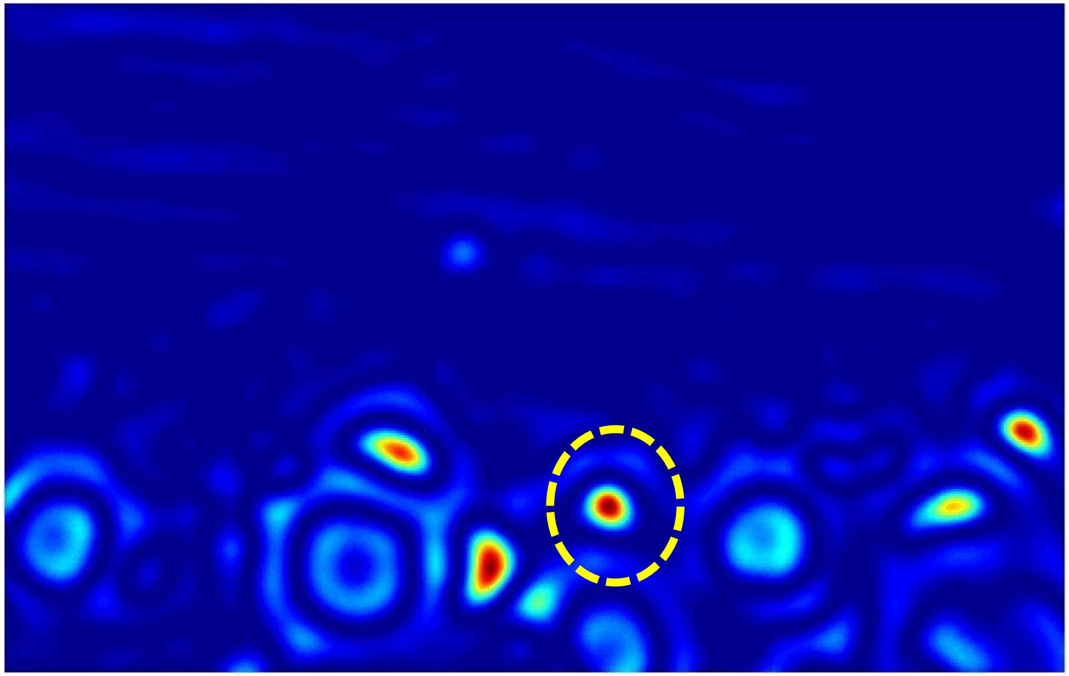




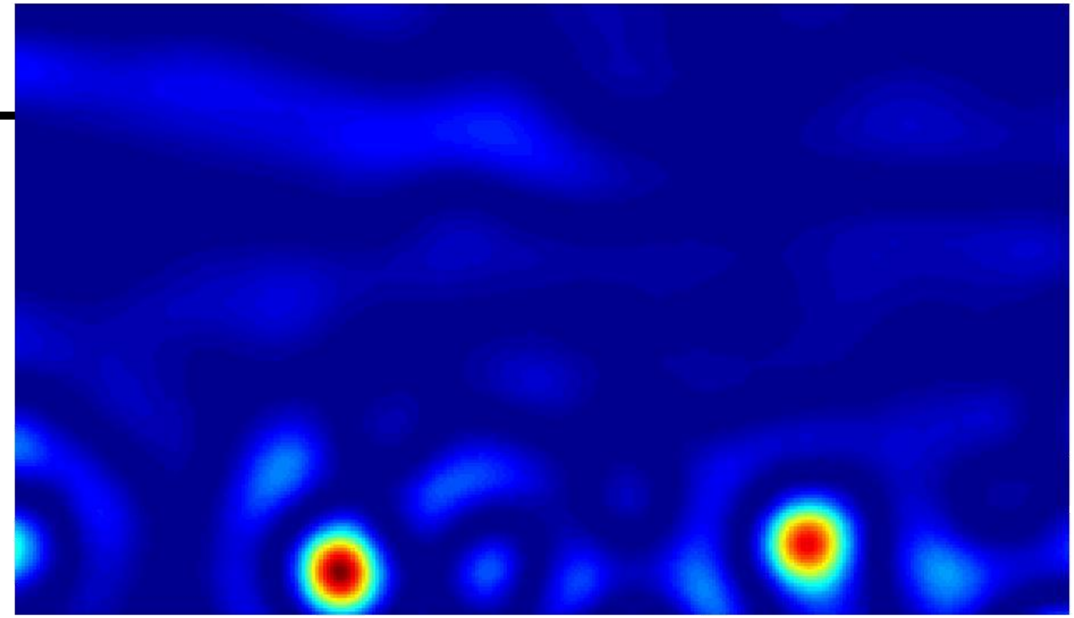
Kristen Grauman

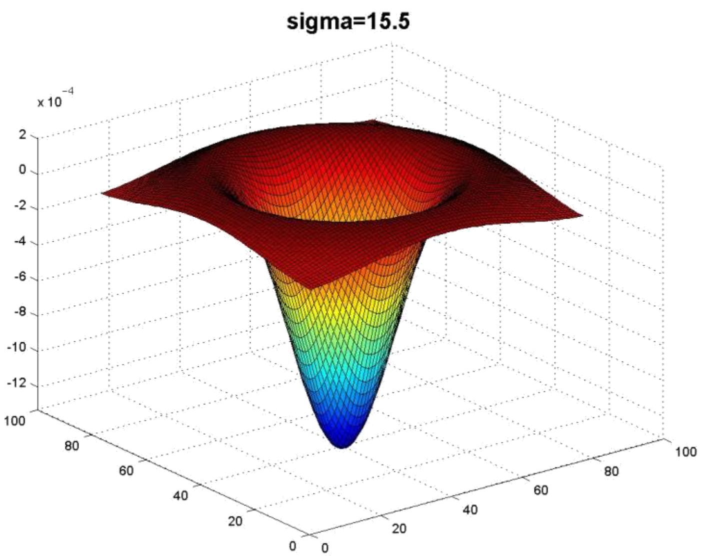


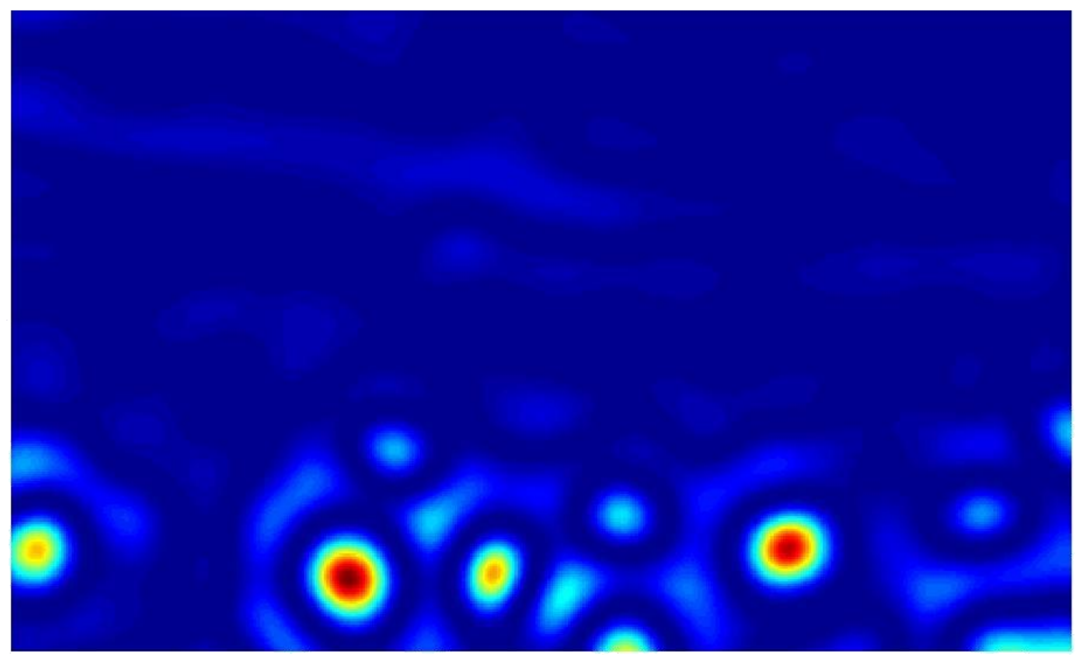




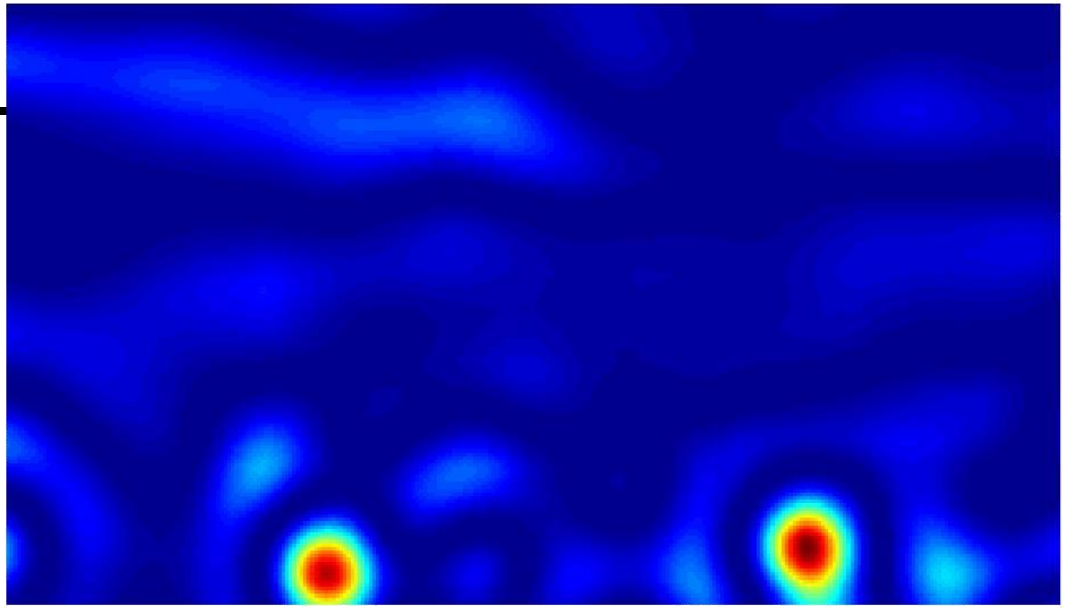
Kristen Grauman

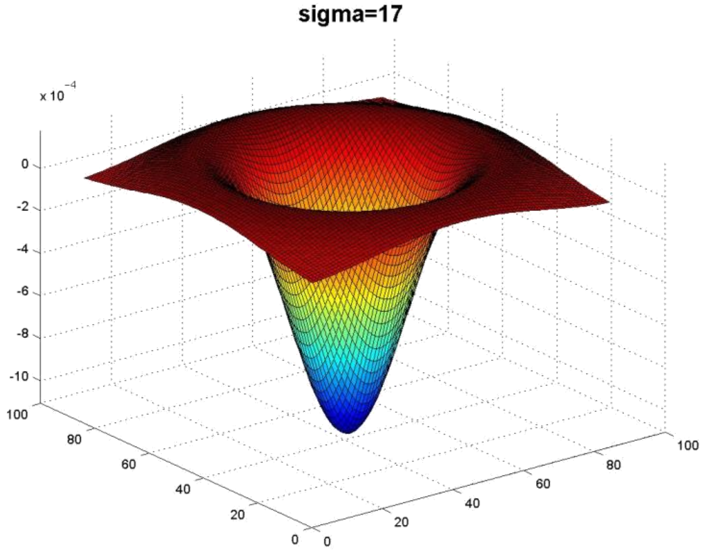


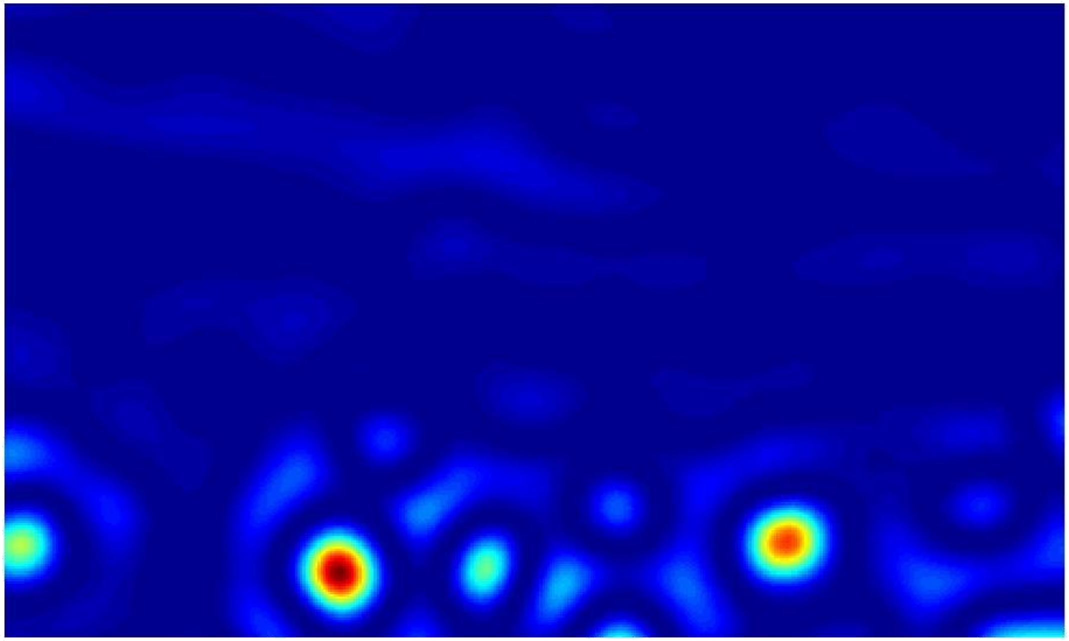




Kristen Grauman





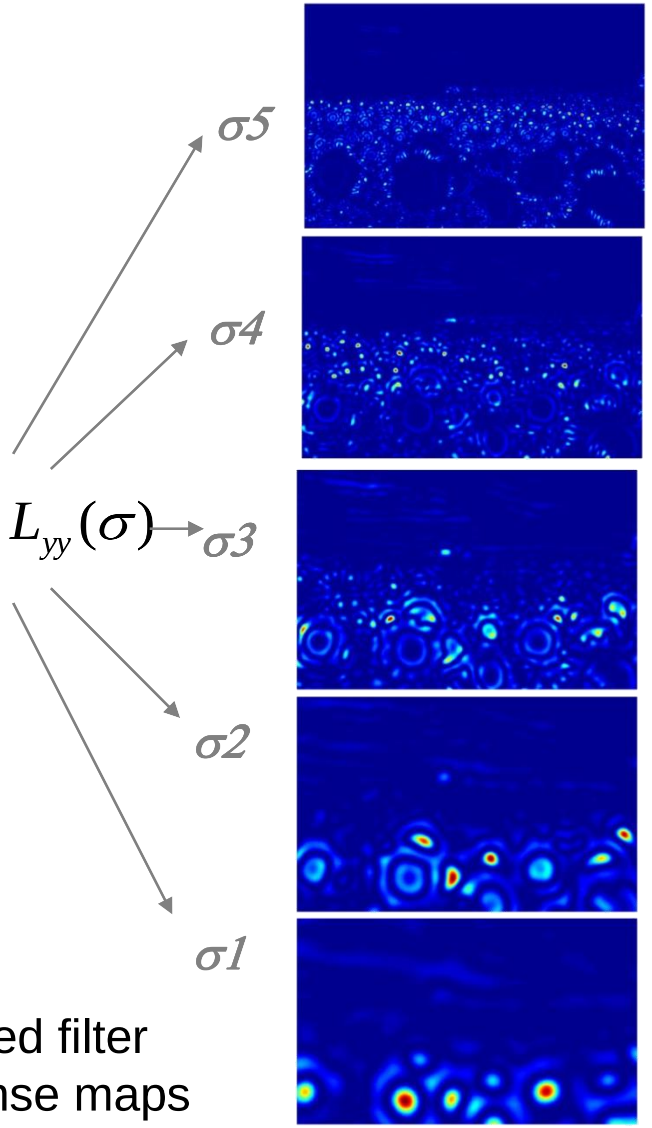


Kristen Grauman

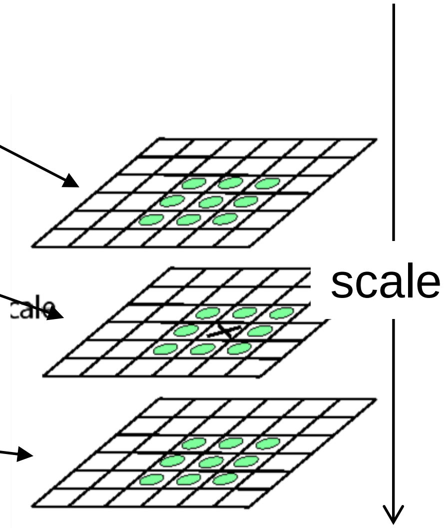
Scale invariant interest points

Interest points are local maxima in both position and scale.

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



Squared filter response maps



⇒ **List of *(x, y,* σ*)***

Scale-space blob detector: Example

Image credit: Lana Lazebnik

SIFT Feature Detector

Scale-space blob detection, but …

- Approximate the Laplacian with a difference of Gaussians (DoG)
  - More efficient to implement

- Reject points with bad contrast:
  - DoG smaller than 0.03 (image values in [0,1])

- Reject edges
  - Similar to the Harris detector; look at the autocorrelation matrix

SIFT Feature Detector

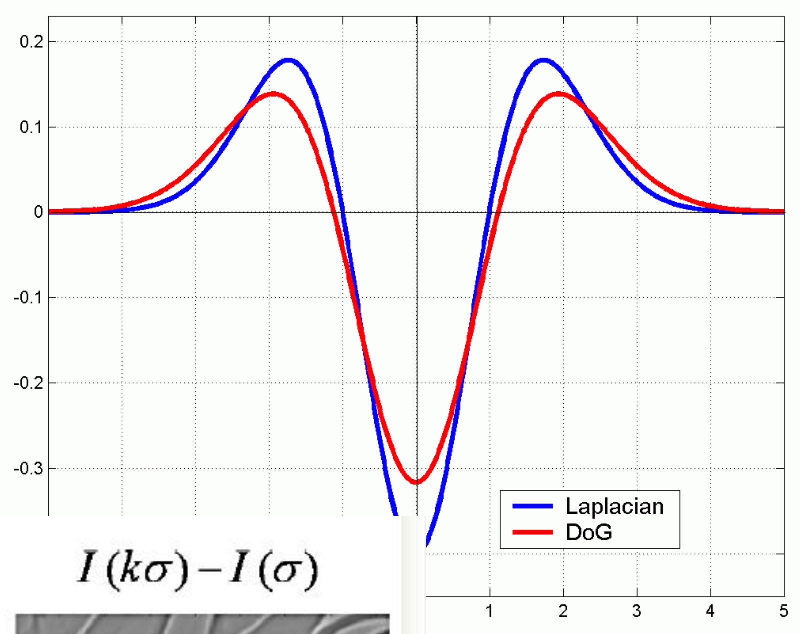
Approximate the Laplacian with a difference of Gaussians; more efficient to implement.

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

(Laplacian)

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

(Difference of Gaussians)

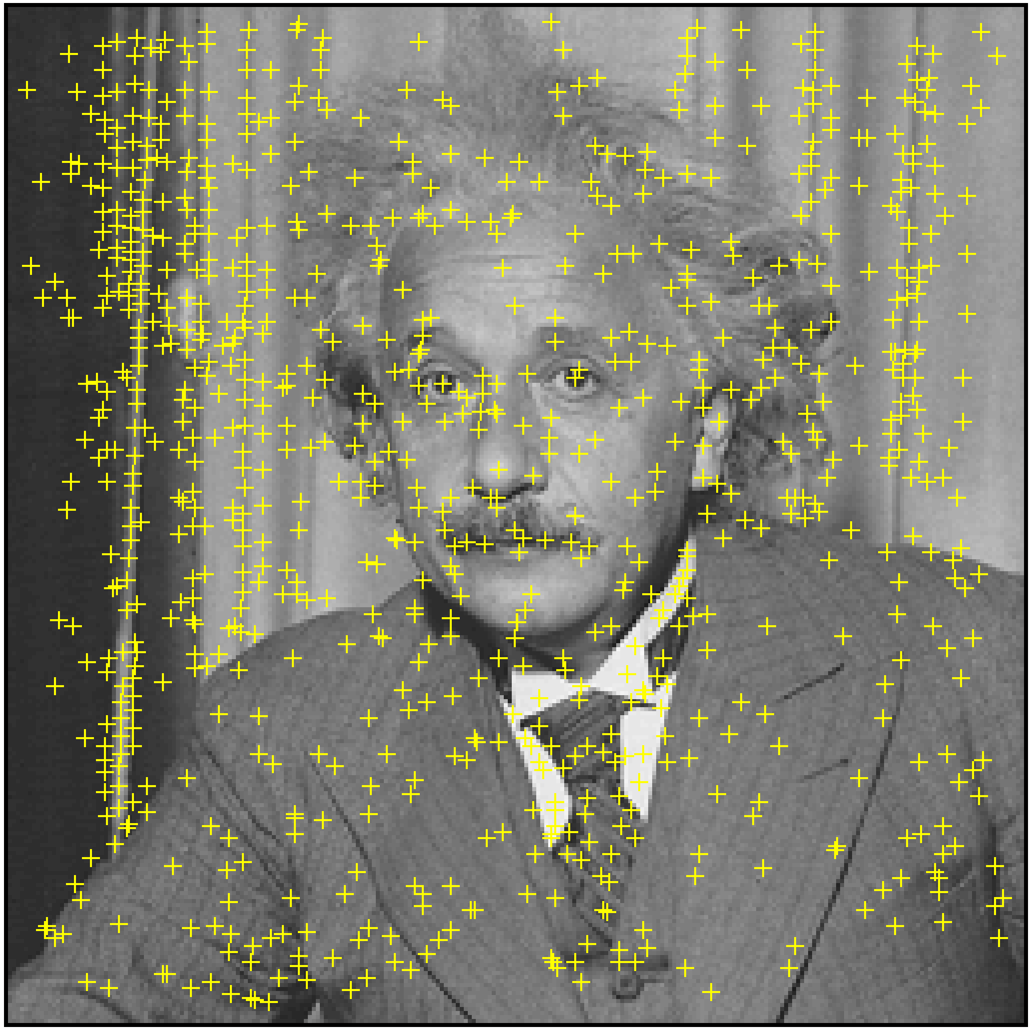


$I(k\sigma)$ $\quad$ $I(\sigma)$ $\quad$ $I(k\sigma) - I(\sigma)$

-

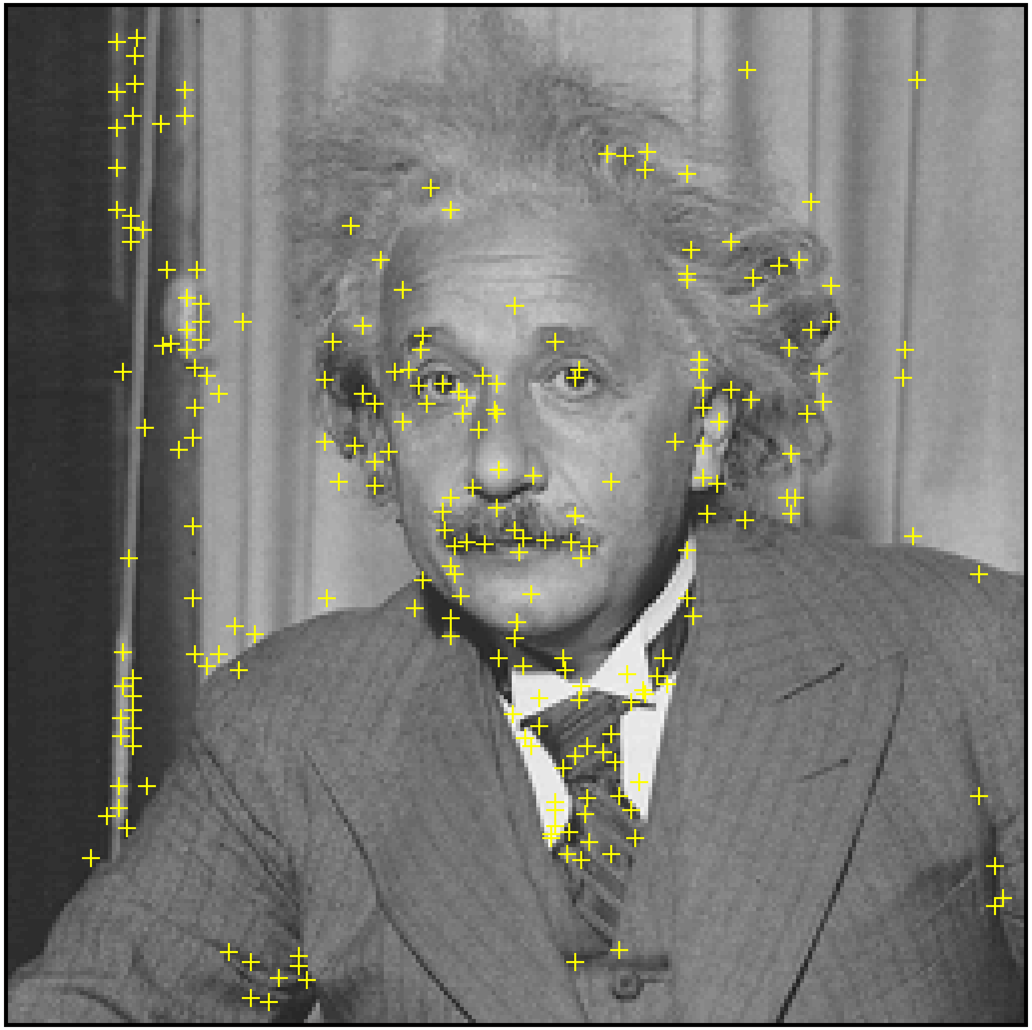
=

SIFT Feature Detector



Maxima of DoG

SIFT Feature Detector



After removing low contrast and edges

SIFT Orientation assignment

By assigning a consistent orientation, the keypoint descriptor can be orientation invariant.

Let, for a keypoint, L is the image with the closest scale.

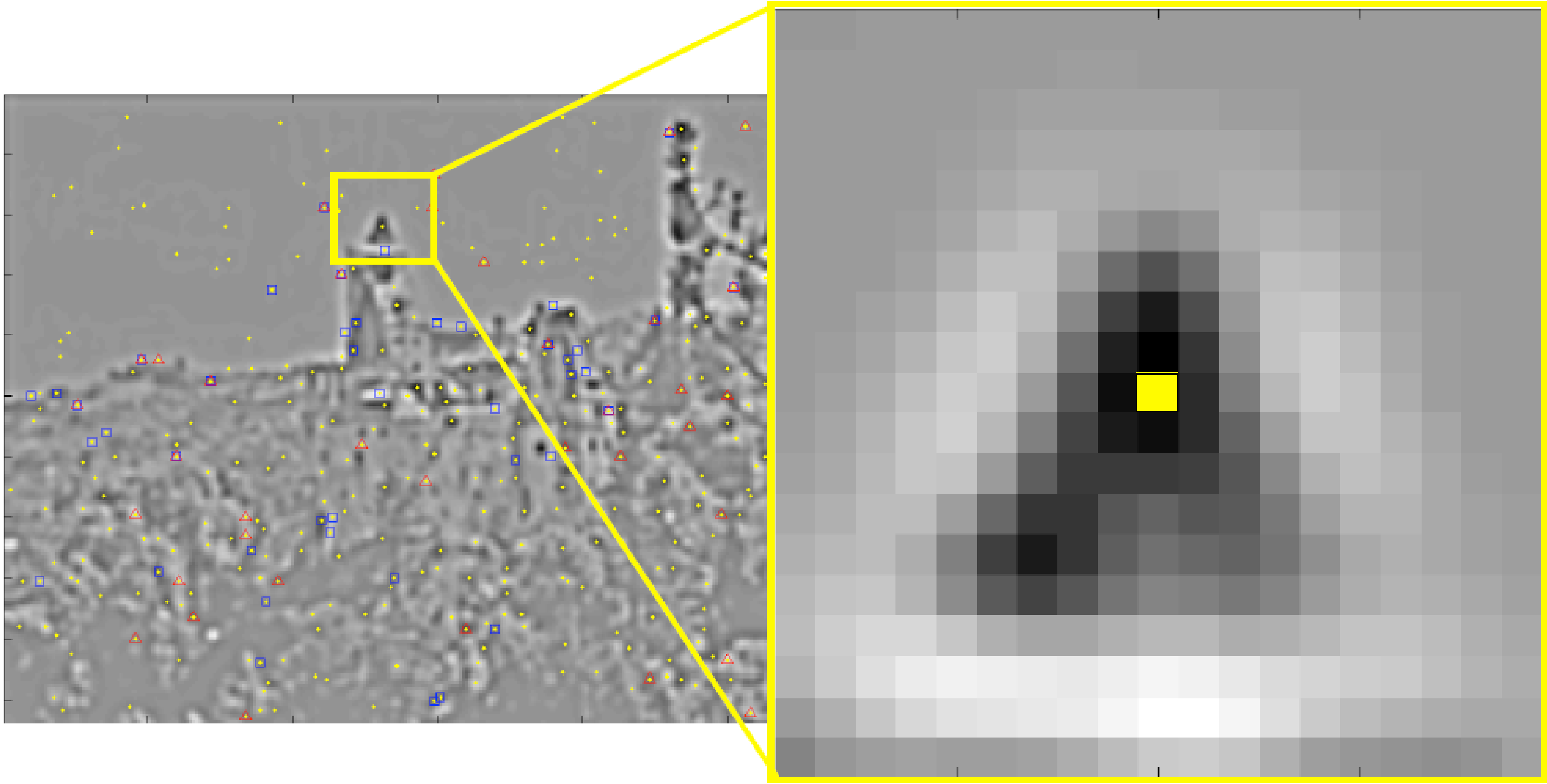
- Compute gradient magnitude and orientation using finite differences:

$$GradientVector = \begin{bmatrix} L(x+1, y) - L(x-1, y) \\ L(x, y+1) - L(x, y-1) \end{bmatrix}$$

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

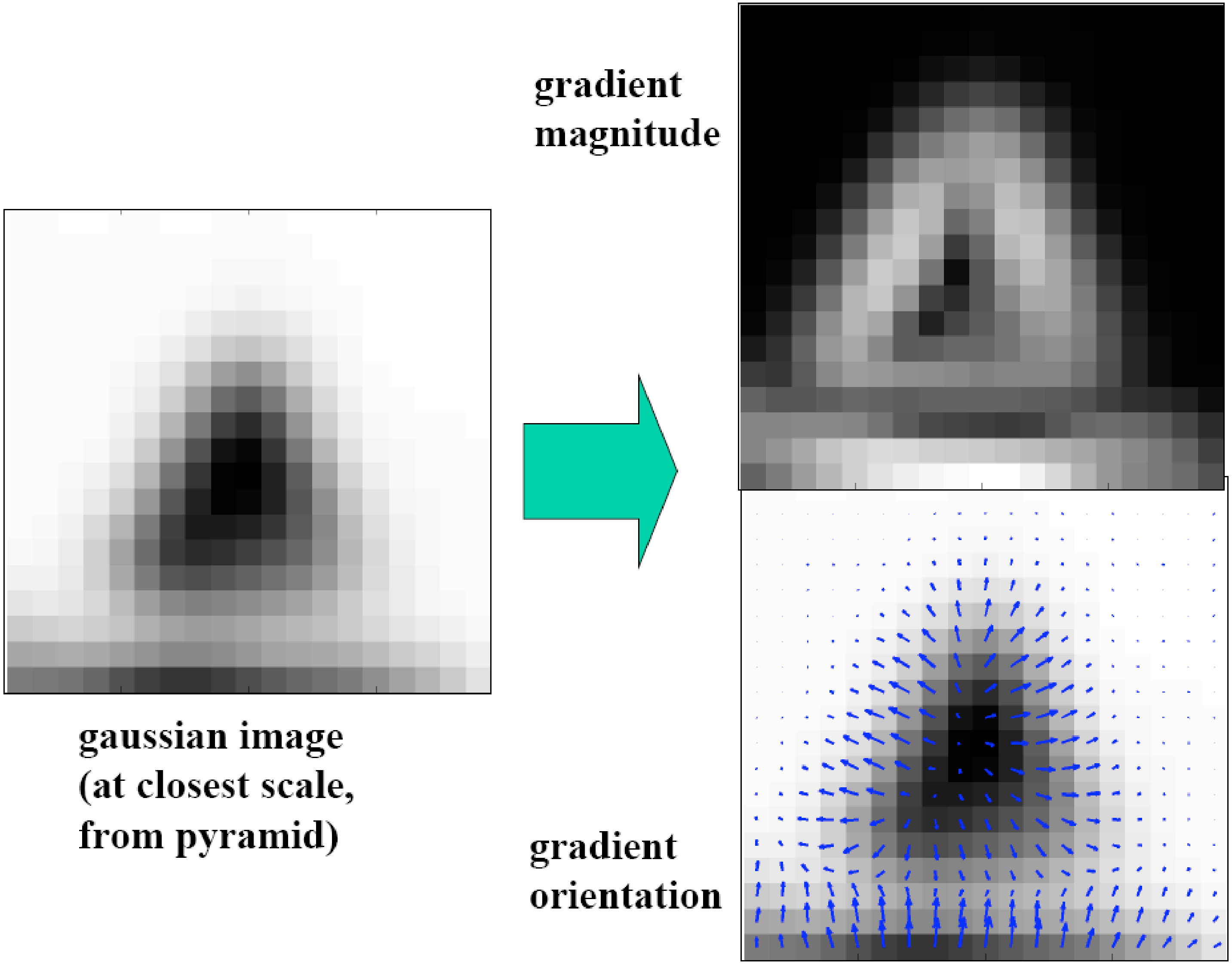
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

SIFT Orientation assignment



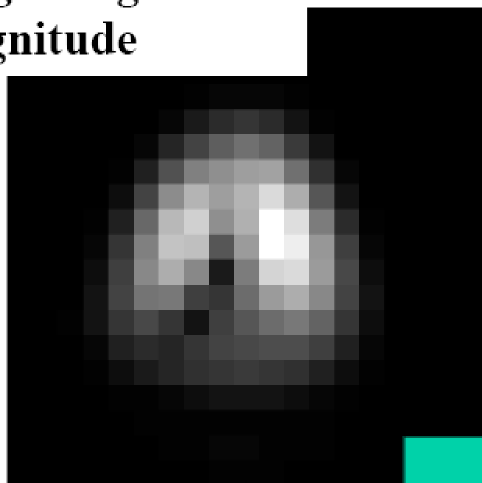
- Keypoint location = extrema location
- Keypoint scale is scale of the DOG image

SIFT Orientation assignment

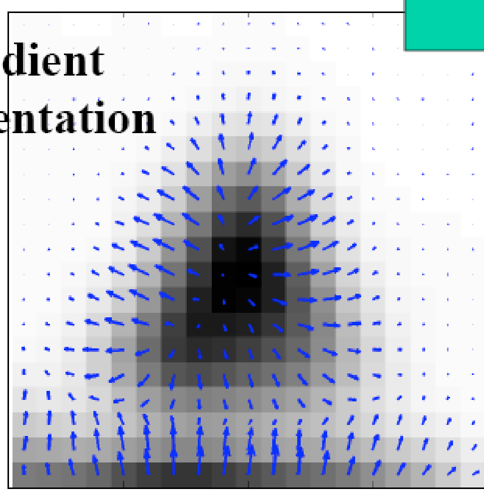


SIFT Orientation assignment

weighted gradient magnitude

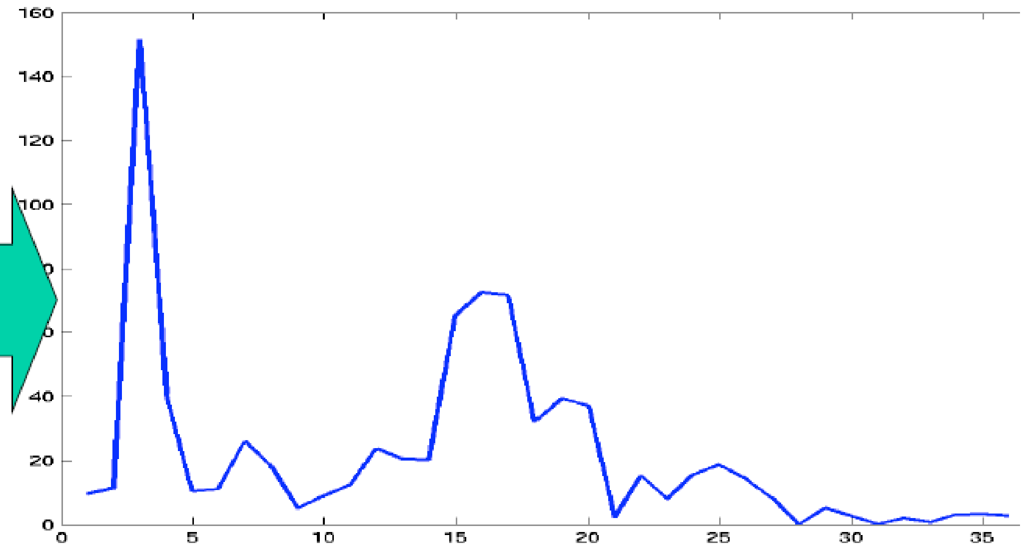


gradient orientation



weighted orientation histogram.

Each bucket contains sum of weighted gradient magnitudes corresponding to angles that fall within that bucket.



36 buckets

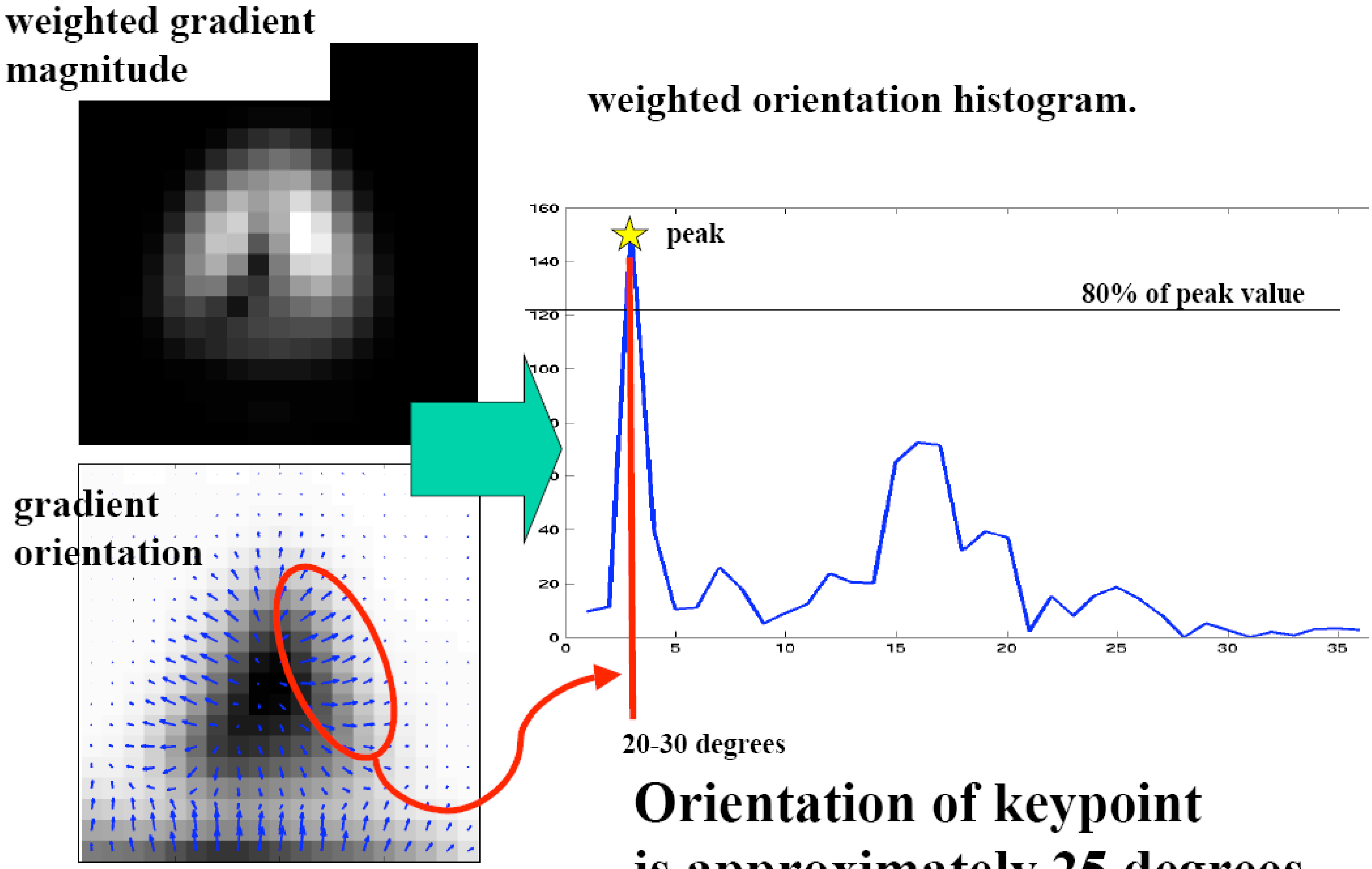
10 degree range of angles in each bucket, i.e.

0 <=ang<10 : bucket 1

10<=ang<20 : bucket 2

20<=ang<30 : bucket 3 …

SIFT Orientation assignment



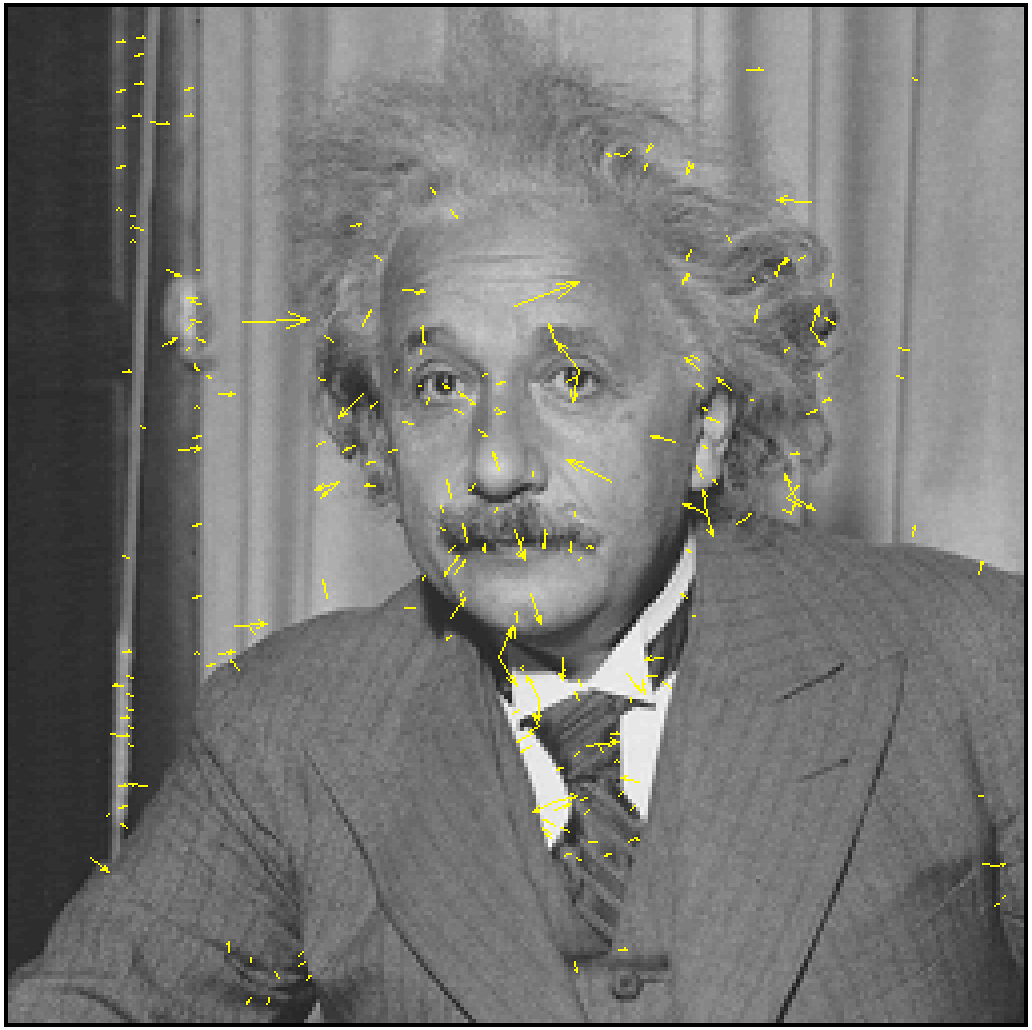
Orientation of keypoint is approximately 25 degrees

SIFT Orientation Assignment

Any peak within 80% of the highest peak is used to create a keypoint with that orientation

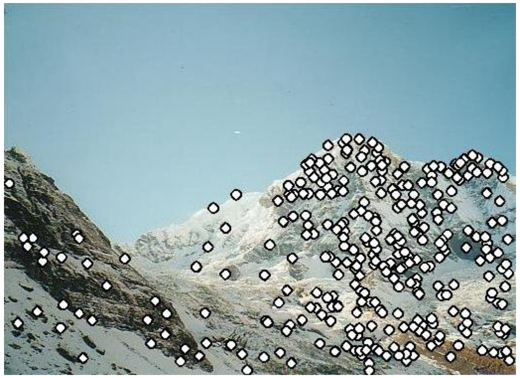
~15% assigned multiplied orientations, but contribute significantly to the stability

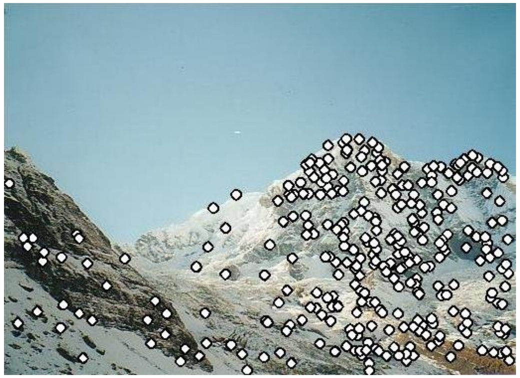
SIFT Feature Points



Matching Image Features

1) Feature Detection:
   Identify image features
2) Feature Description:
   Extract feature descriptor for each feature
3) Feature Matching:
   Find candidate matches between features
4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features





Kristen Grauman

Feature Descriptors

How should we represent the local area around each feature point (for matching)?

Raw Pixel Values?

The simplest way to describe the neighborhood around an interest point is to write down the list of luminances in a k x k window around the point to form a feature vector.

Raw Pixel Values?

The simplest way to describe the neighborhood around an interest point is to write down the list of luminances in a k x k window around the point to form a feature vector.

0.2
0.2
0.2
0.2
0.1
0.3
0.2
0.3
0.3
0.2
0.7
0.4
0.6
0.8
0.6
…

Raw Pixel Values?

Problems?

Window Feature Descriptor

Rotate image region
 around point based on
 the feature orientation

Scale image region
 around the point based on
 the feature scale

Concatenate luminance
 of all pixels in the
 k x k window around
 the point in the
 rotated/scaled window
 into a feature vector

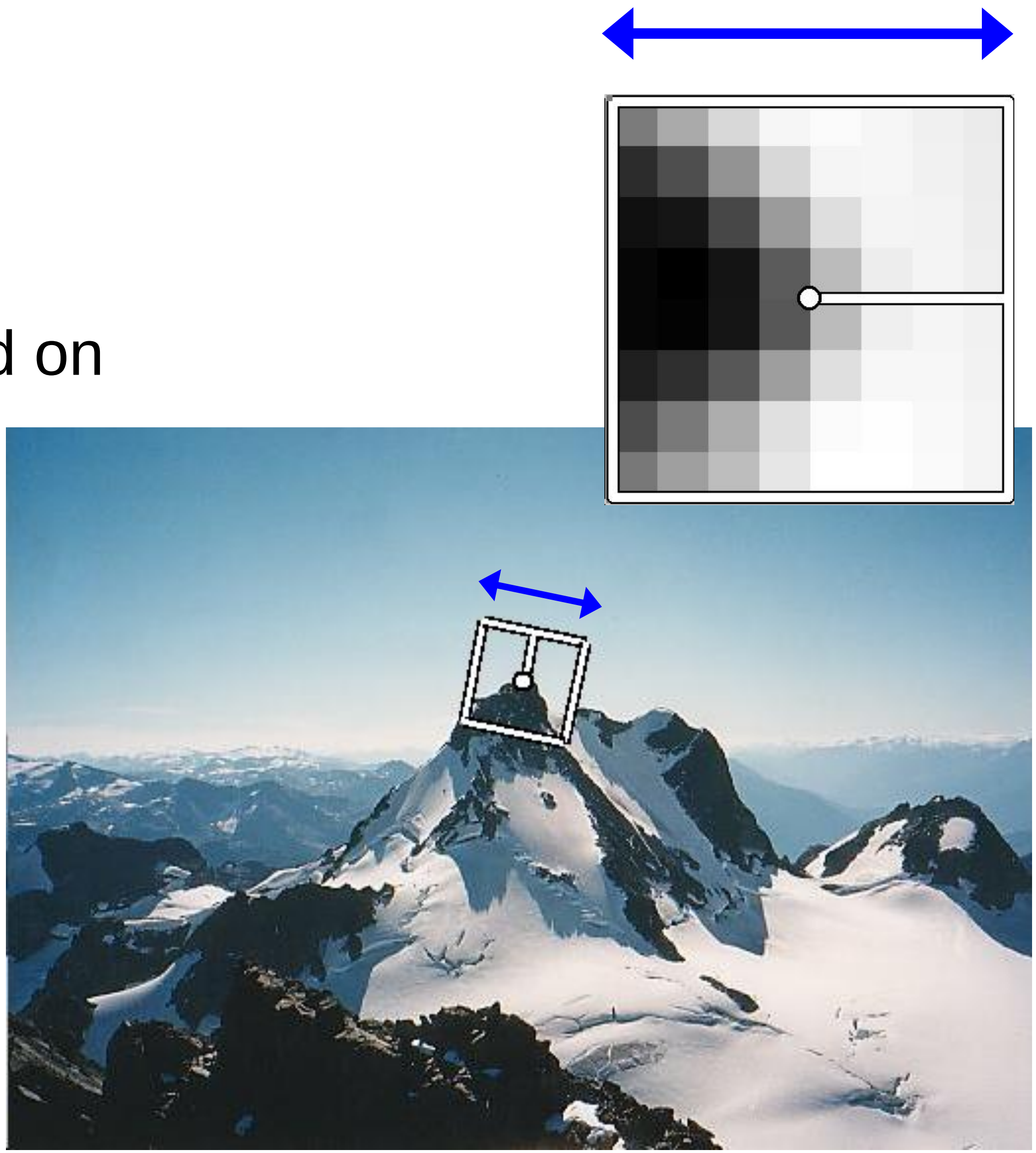
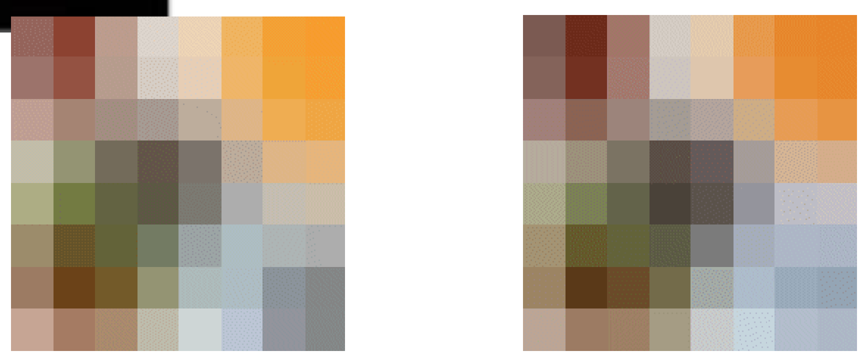


Image from Matthew Brown

Window Feature Descriptor



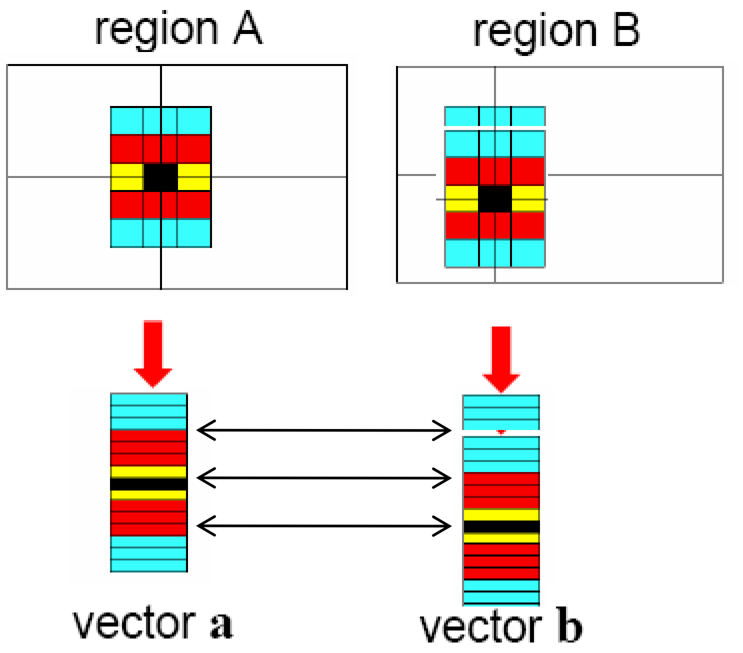
e.g. scale,
translation,
rotation

Window Feature Descriptor

Problems?

Window Feature Descriptor

Sensitive to small shifts or rotations



Window Feature Descriptor

Sensitive to photometric differences

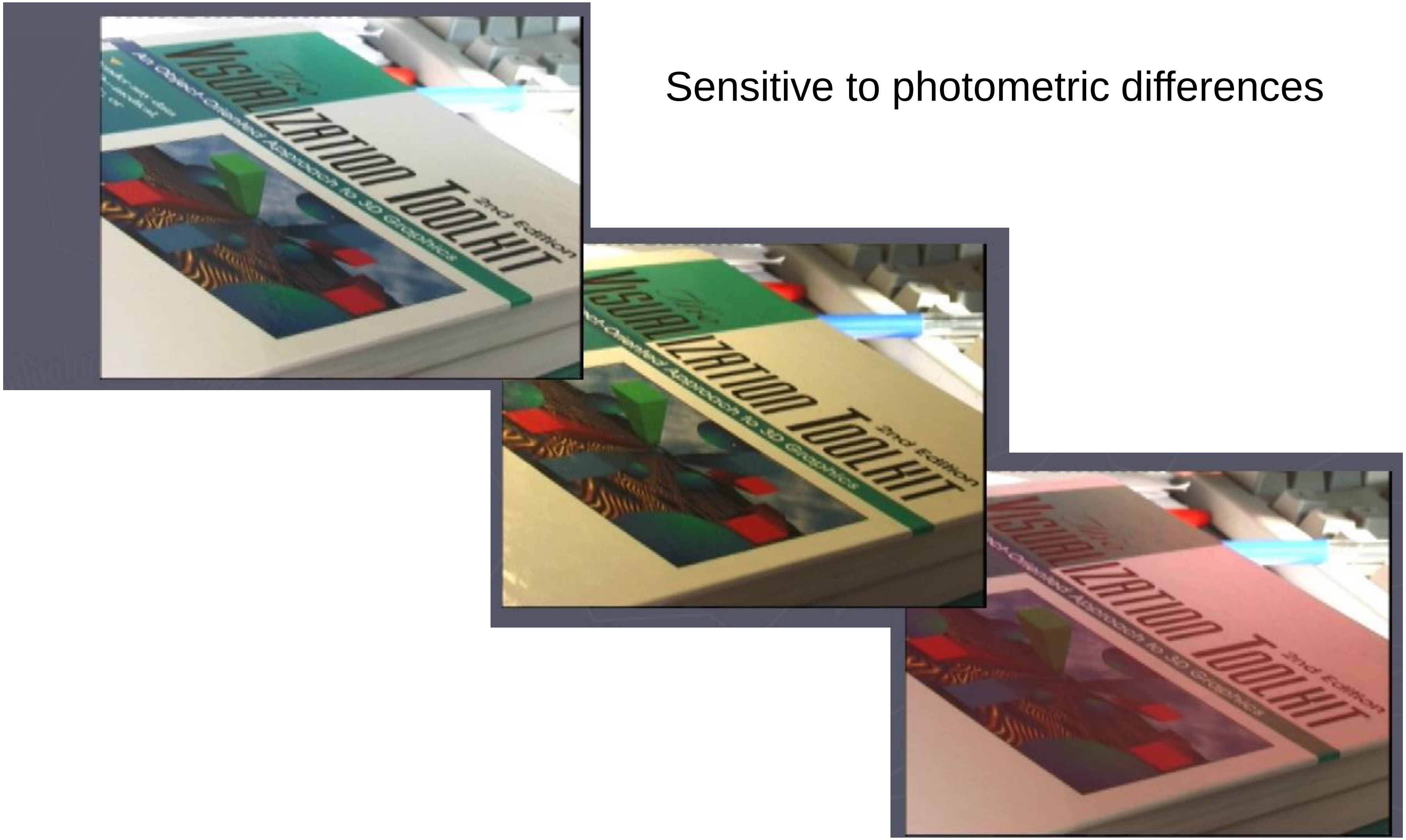
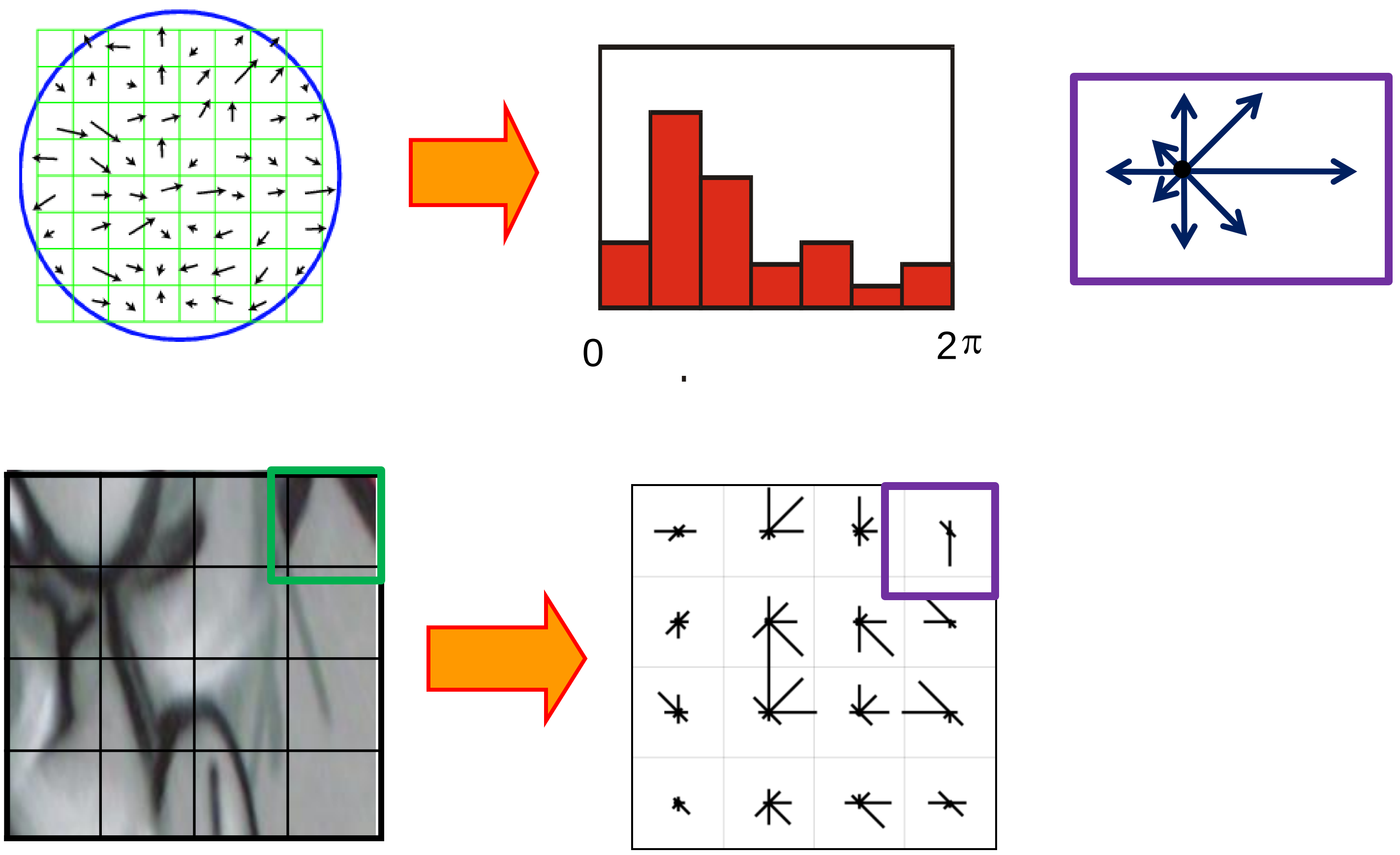


Figure from T. Tuytelaars ECCV 2006 tutorial

SIFT Feature Descriptor [Lowe 2004]

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



SIFT Feature Descriptor

Gradient magnitude and orientation at each point weighted by a Gaussian

Orientation histograms: sum of gradient magnitude at each direction

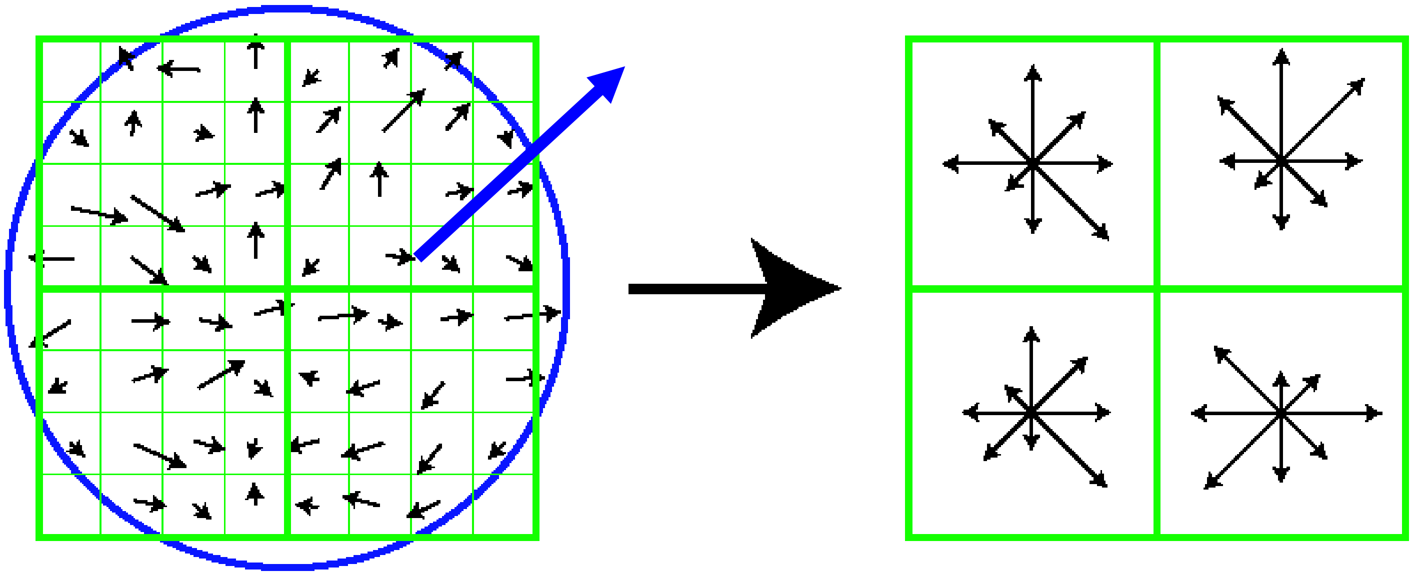


Image gradients

Keypoint descriptor

In practice, 4x4 arrays of 8 bin histogram is used, a total of 128 features for one keypoint

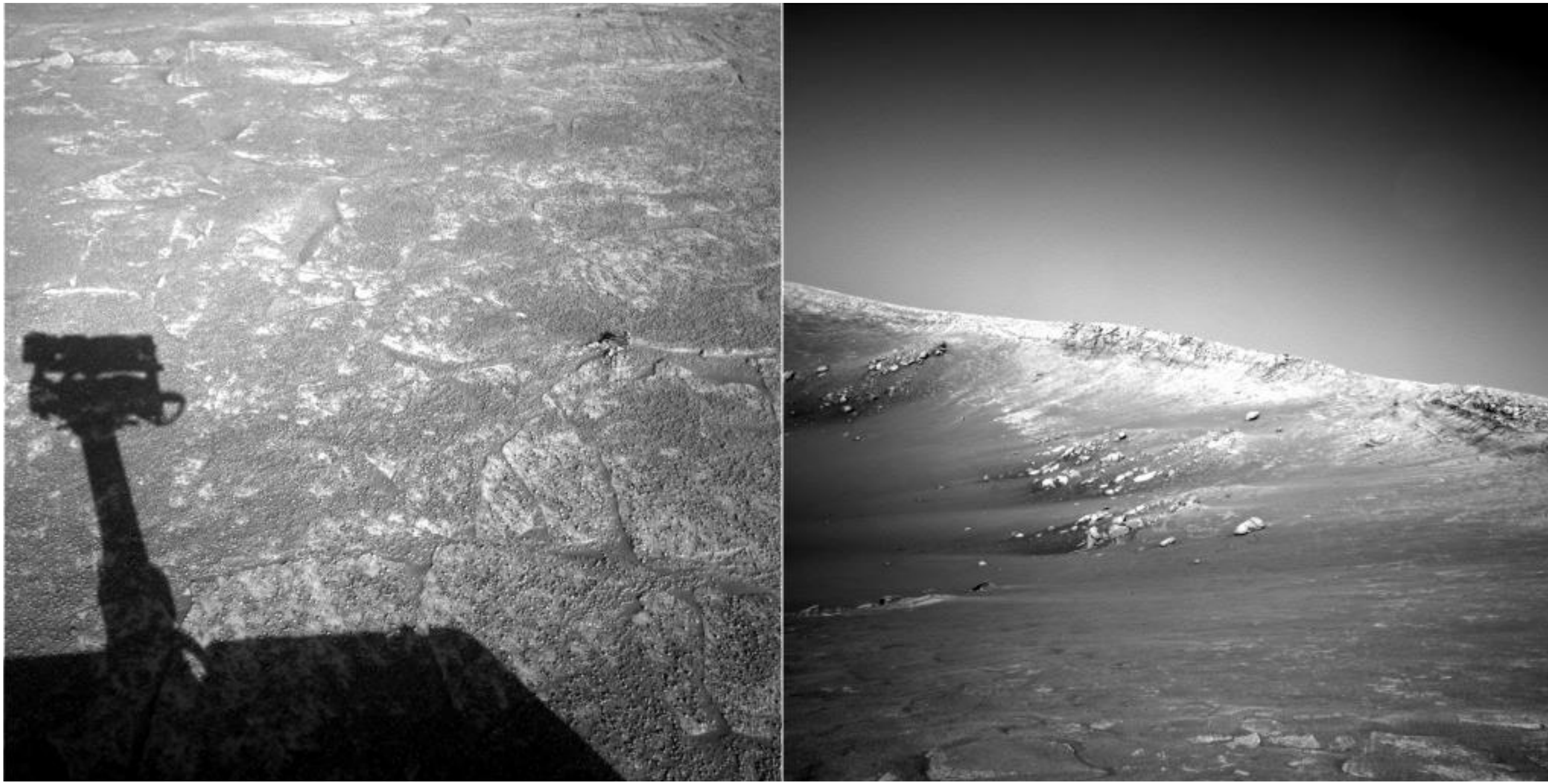
Image from Lowe

SIFT Feature Descriptor

- 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

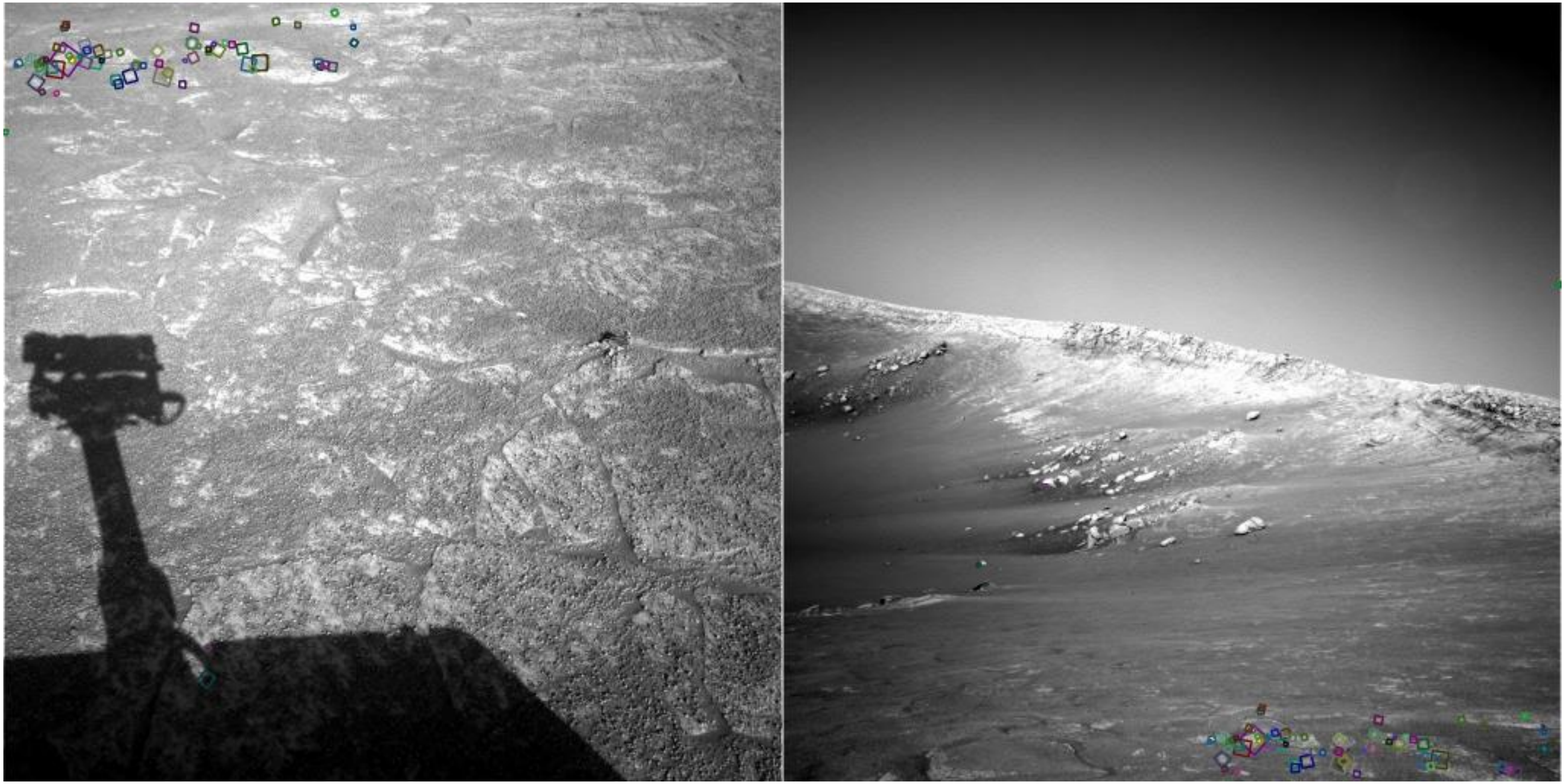
Steve Seitz

SIFT Feature Descriptor



NASA Mars Rover images

SIFT Feature Descriptor



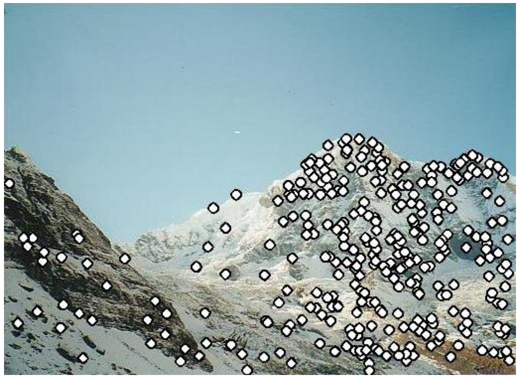
NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Summary

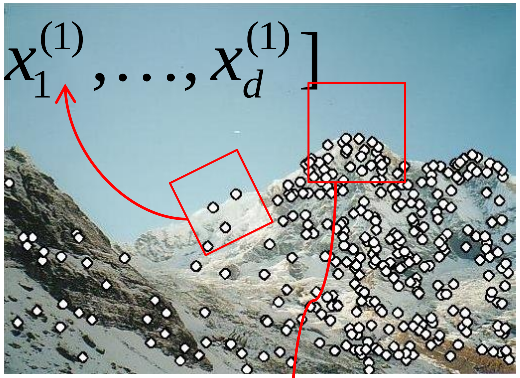
1) Feature Detection:
   Identify image features
2) Feature Description:
   Extract feature descriptor for each feature
3) Feature Matching:
   Find candidate matches between features

Next Time

4) Feature Correspondence:
   Find consistent set of (inlier) correspondences between features



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



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

Kristen Grauman