特征匹配与运动估计

周晓巍

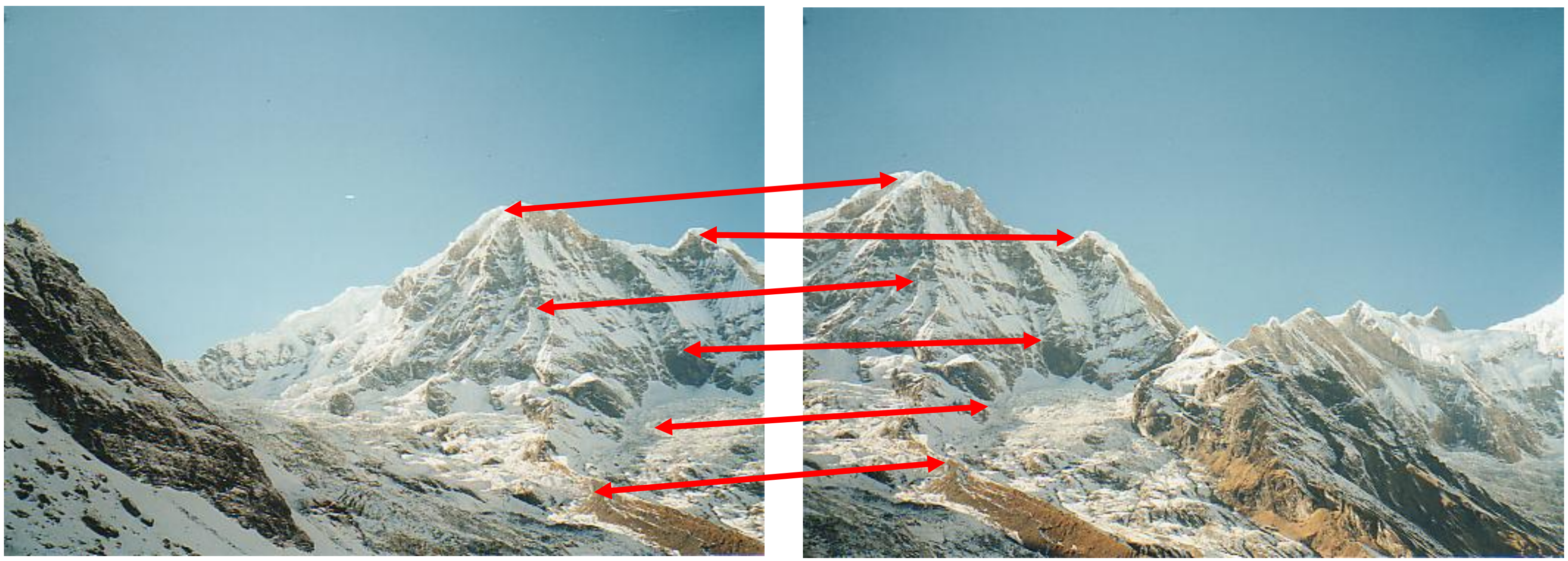
计算摄影学第九节

Slides adapted from Noah Snavely, Jia-Bin Huang and Linda Shapiro


DONE

How to combine two images?

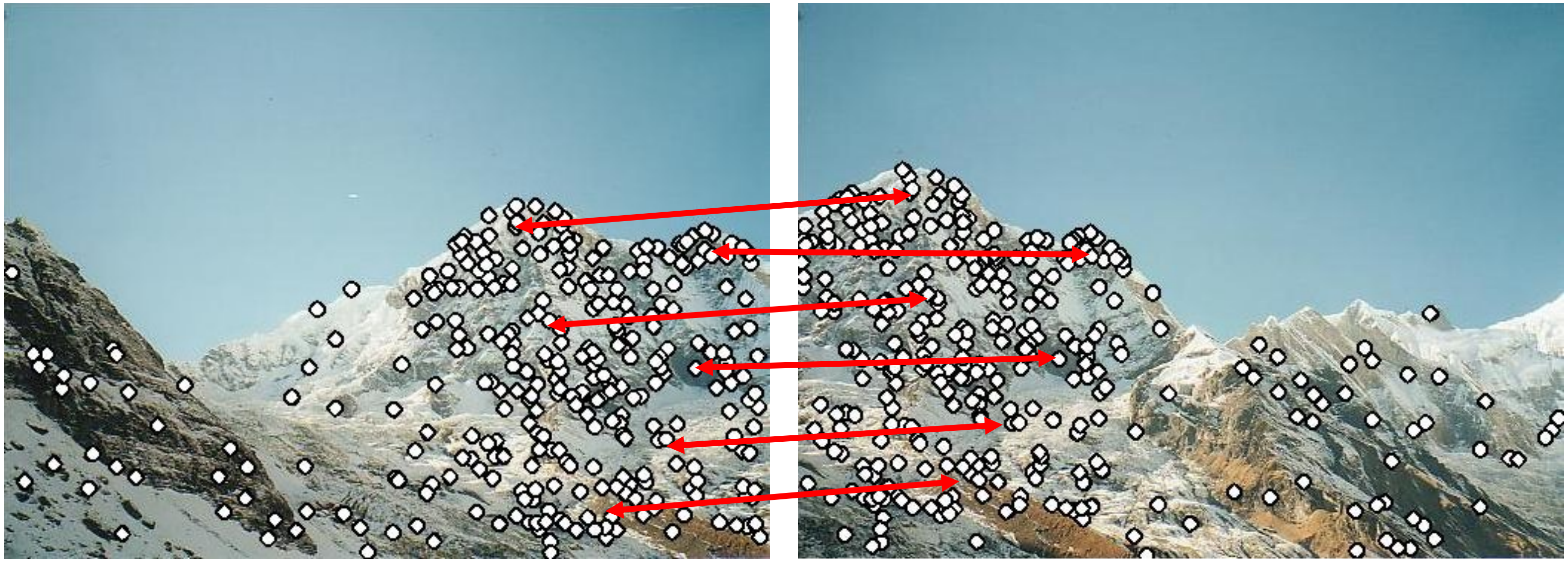
How to combine two images?



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How to combine two images?

How to do feature matching?



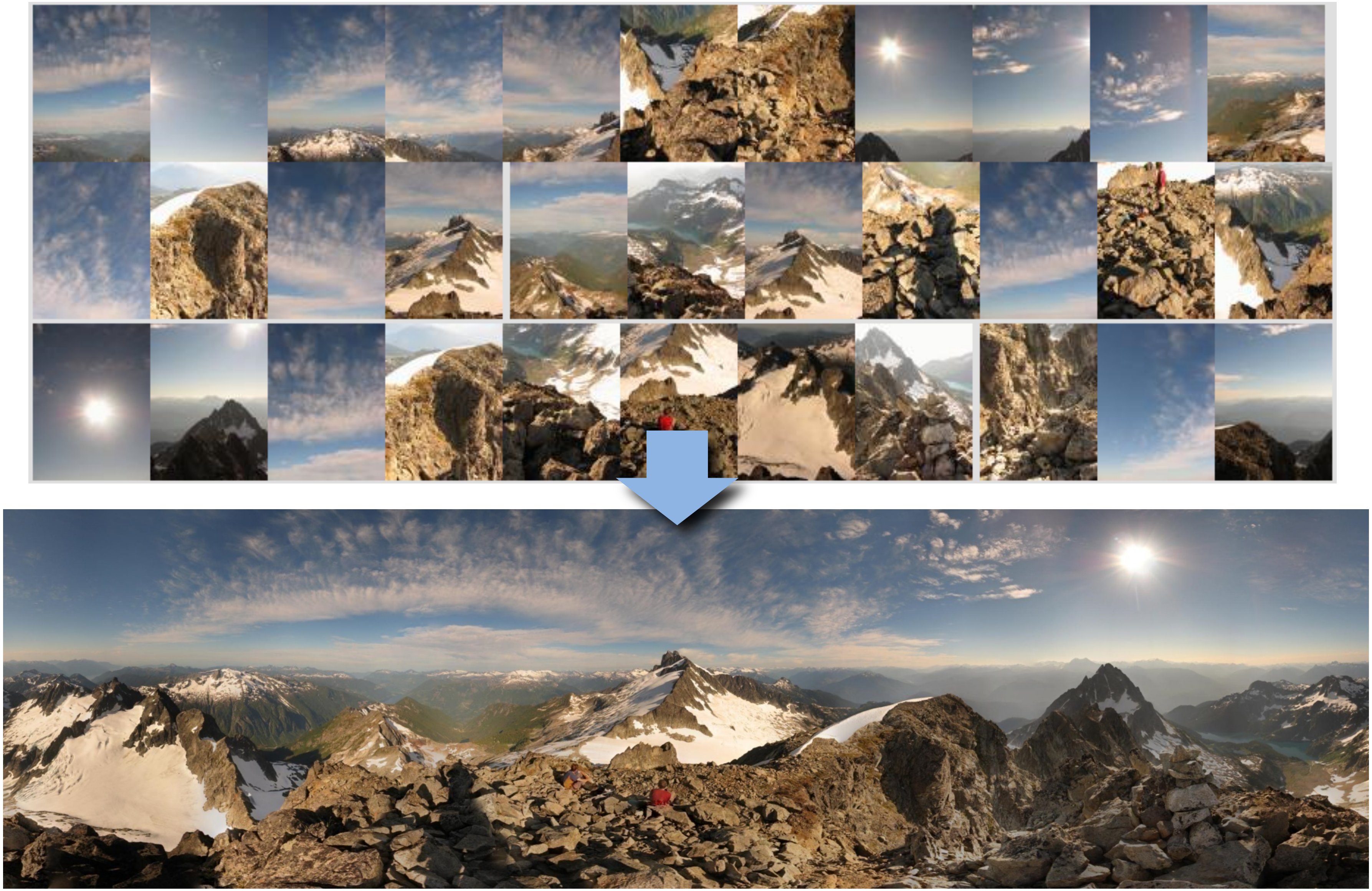
Step 1: extract features
Step 2: match features

Lots of applications

Features are used for:

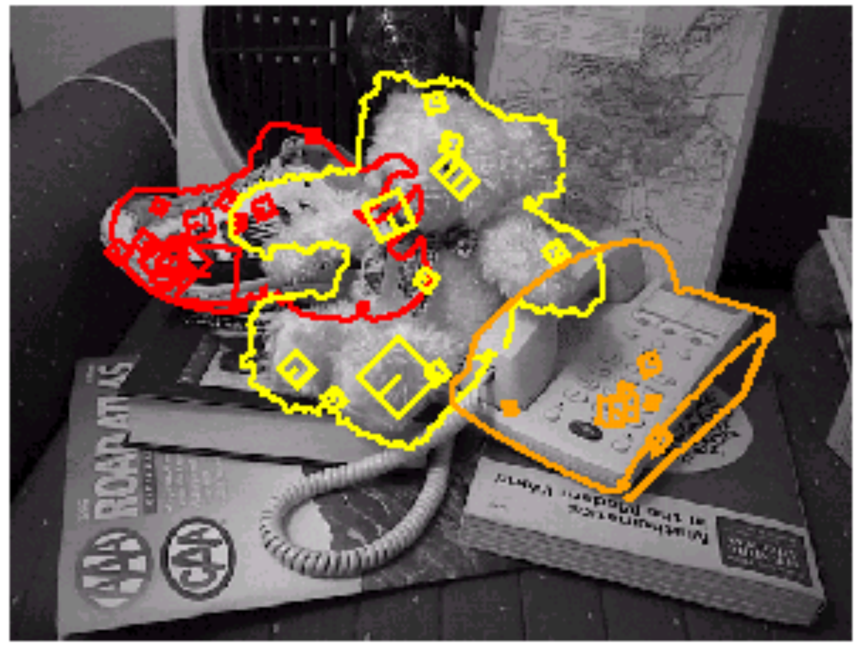
- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

Automatic panoramas（全景）



Credit: Matt Brown

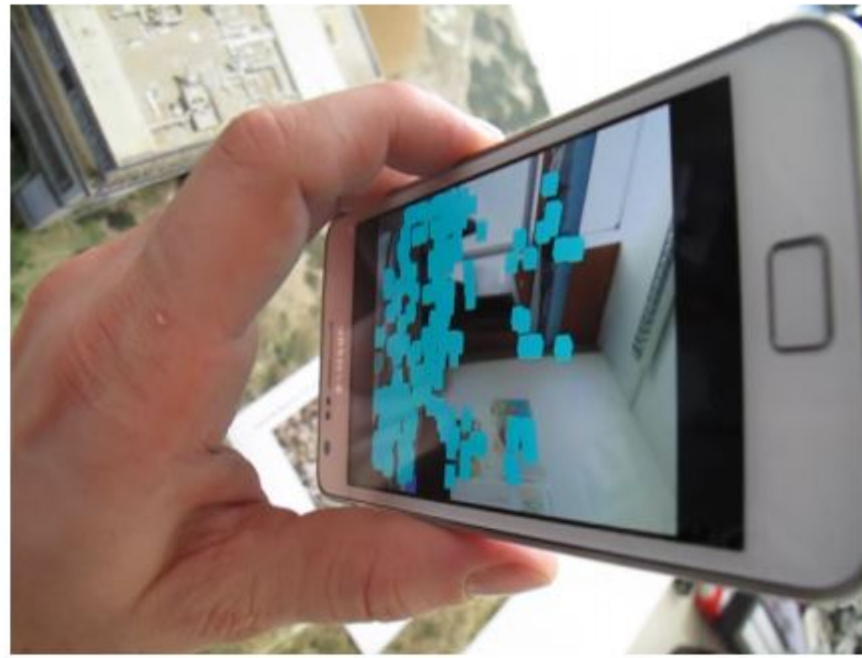
Object recognition



3D Reconstruction

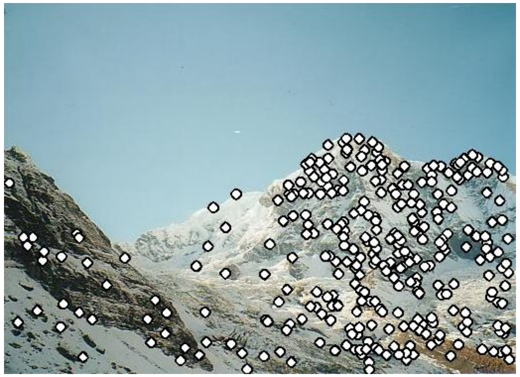
Building Rome in a Day

Visual SLAM



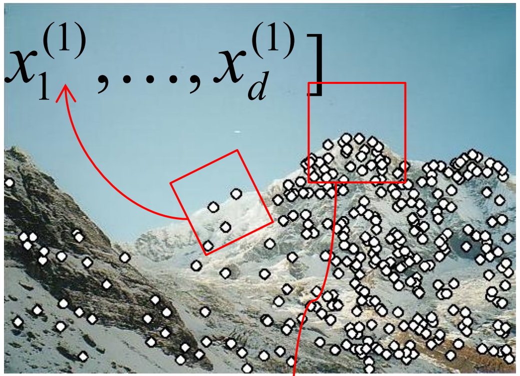
Main Components of Feature matching

1) **Detection:** Identify the interest points



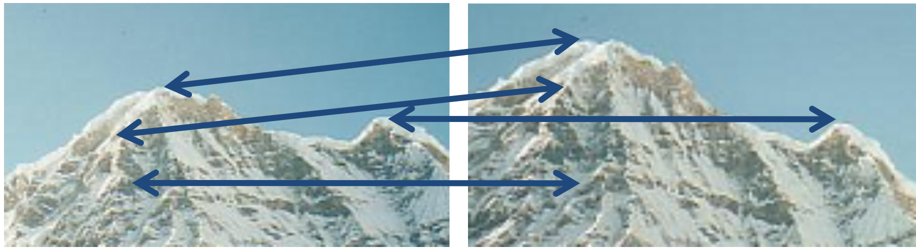
2) **Description**: Extract vector feature descriptor surrounding each interest point.

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



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

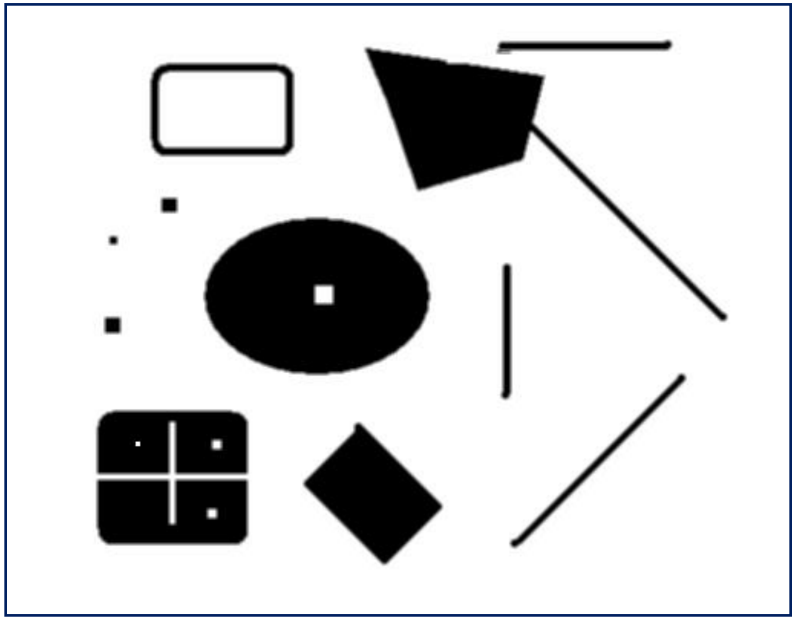
3) **Matching:** Determine correspondence between descriptors in two views

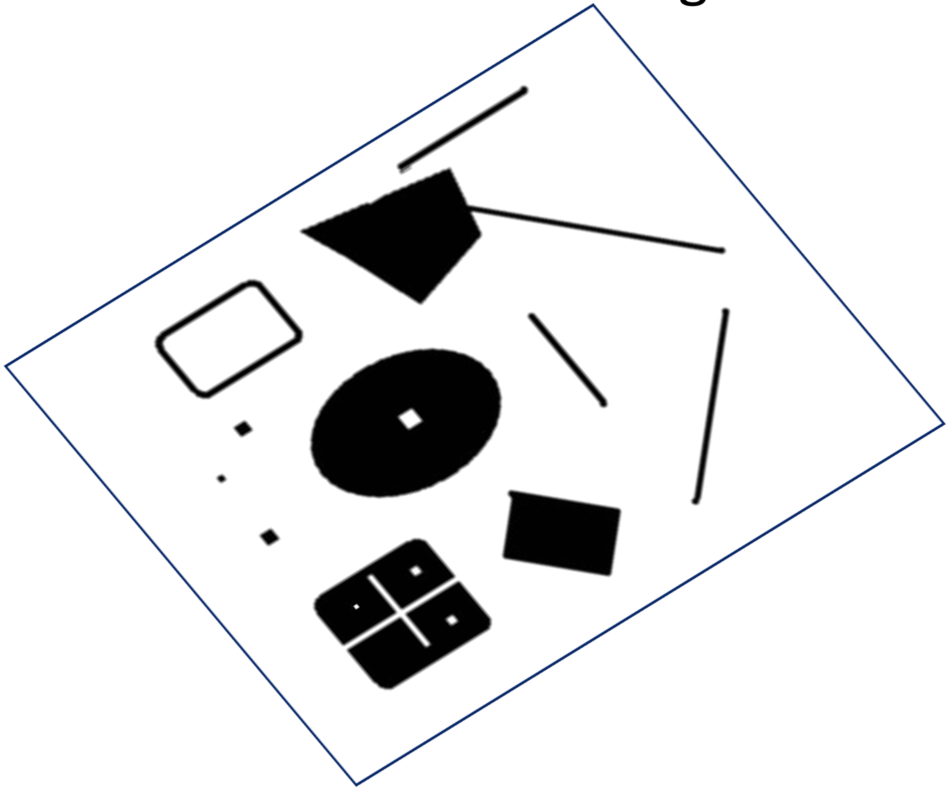


Kristen Grauman

Interest points (兴趣点)

Which points will you choose to match these two images?





- Or called feature points (特征点)

Want uniqueness (唯一性)

Look for image regions that are unique

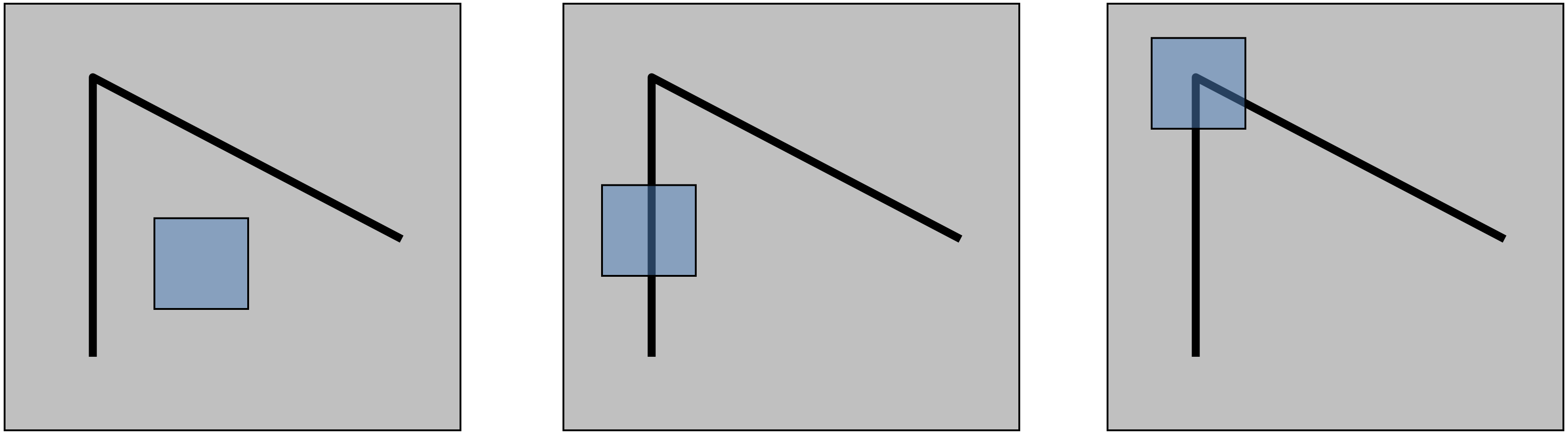
- Lead to unambiguous matches in other images

How to define "uniqueness"?

Local measures of uniqueness

Suppose we consider a small window of pixels (region)

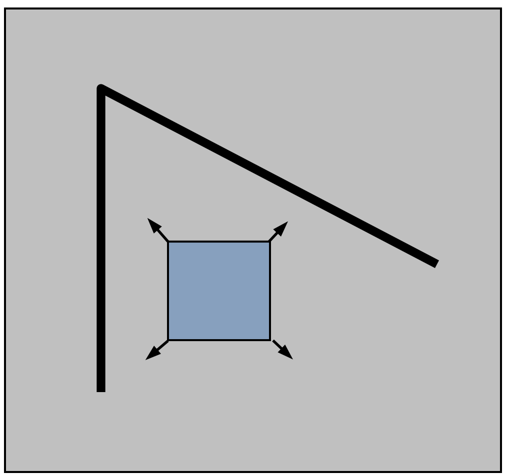
– What defines whether a region is unique?



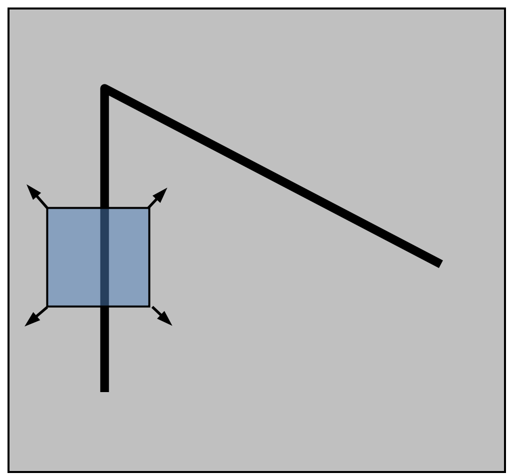
Credit: S. Seitz, D. Frolova, D. Simakov

Local measures of uniqueness

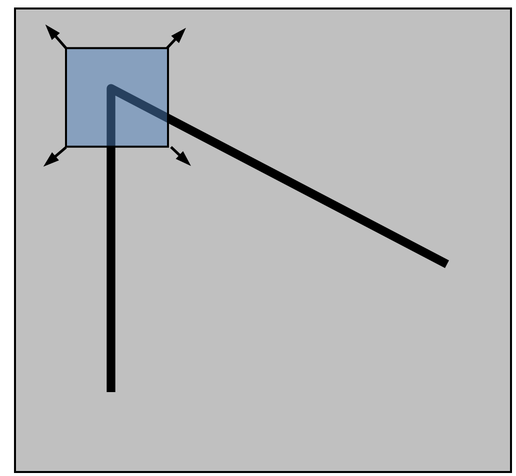
- Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



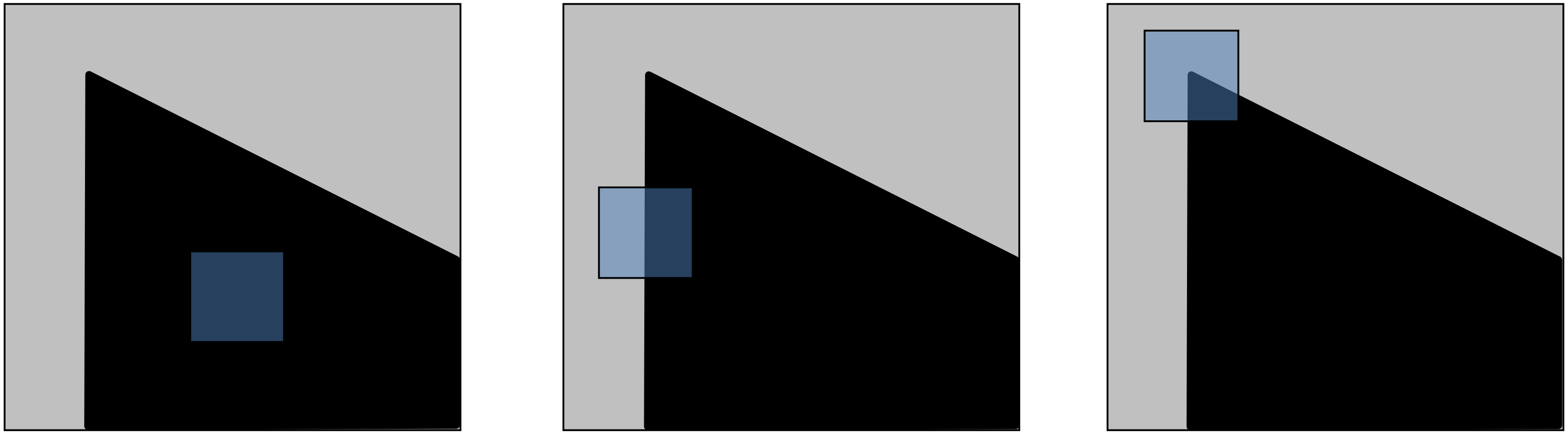
"edge":
no change along the edge direction



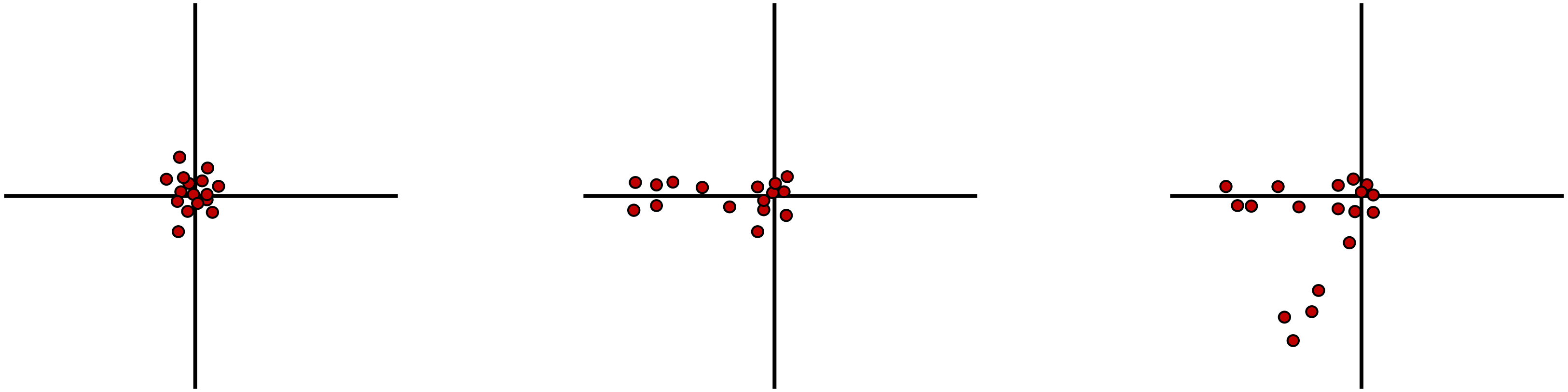
"corner":
significant change in all directions

Credit: S. Seitz, D. Frolova, D. Simakov

How to measure uniqueness mathematically?

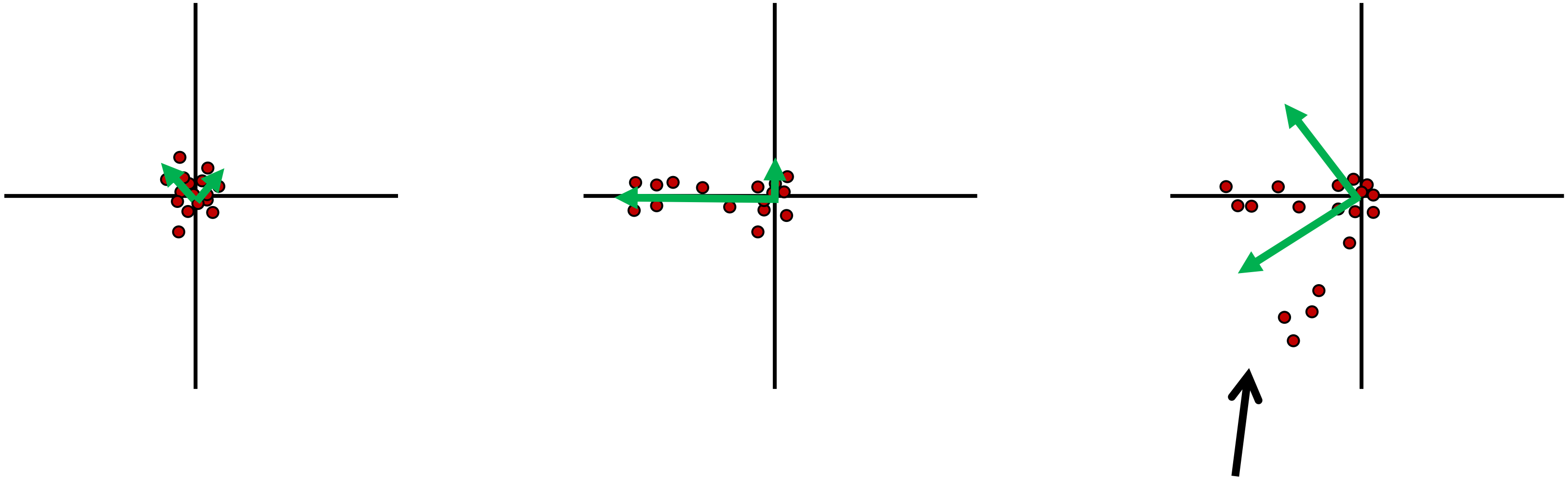


Let‘s look at the distribution of gradients in the region:



Principal Component Analysis
主成分分析

- The 1st principal component is the direction with highest variance.
- The 2nd principal component is the direction with highest variance which is *orthogonal* to the previous components.



Both components are large!

Principal Component Analysis

- How to compute PCA components:

1. Subtract off the mean for each data point.
2. Compute the covariance matrix.
3. Compute eigenvectors and eigenvalues. $Hx = \lambda x$
4. The components are the eigenvectors ranked by the eigenvalues.

Corner detection

1. Compute the covariance matrix at each point

$$H = \sum_{(u,v)} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \qquad I_x = \frac{\partial f}{\partial x}, I_y = \frac{\partial f}{\partial y}$$

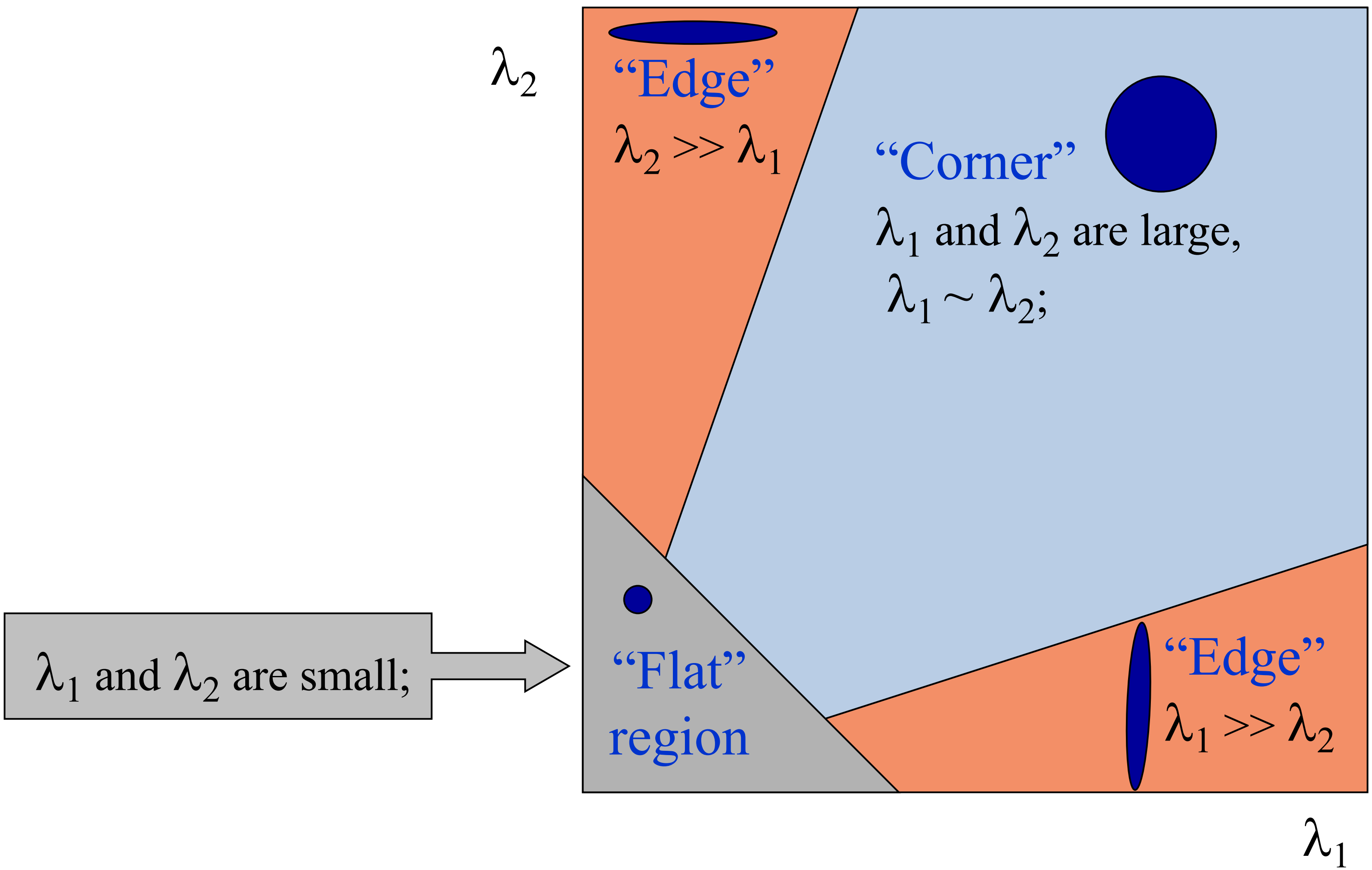
Typically Gaussian weights

2. Compute eigenvalues.

$$H = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \lambda_{\pm} = \frac{1}{2}\left((a+d) \pm \sqrt{4bc + (a-d)^2}\right)$$

Corner detection

3. Classify points using eigenvalues of *H*:



The Harris operator

Computing eigenvalues are expensive
Harris corner detector uses the following alternative

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{determinant(H)}{trace(H)}$$

f is called corner response

Reminder: $det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc \qquad trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$

Harris detector: Steps

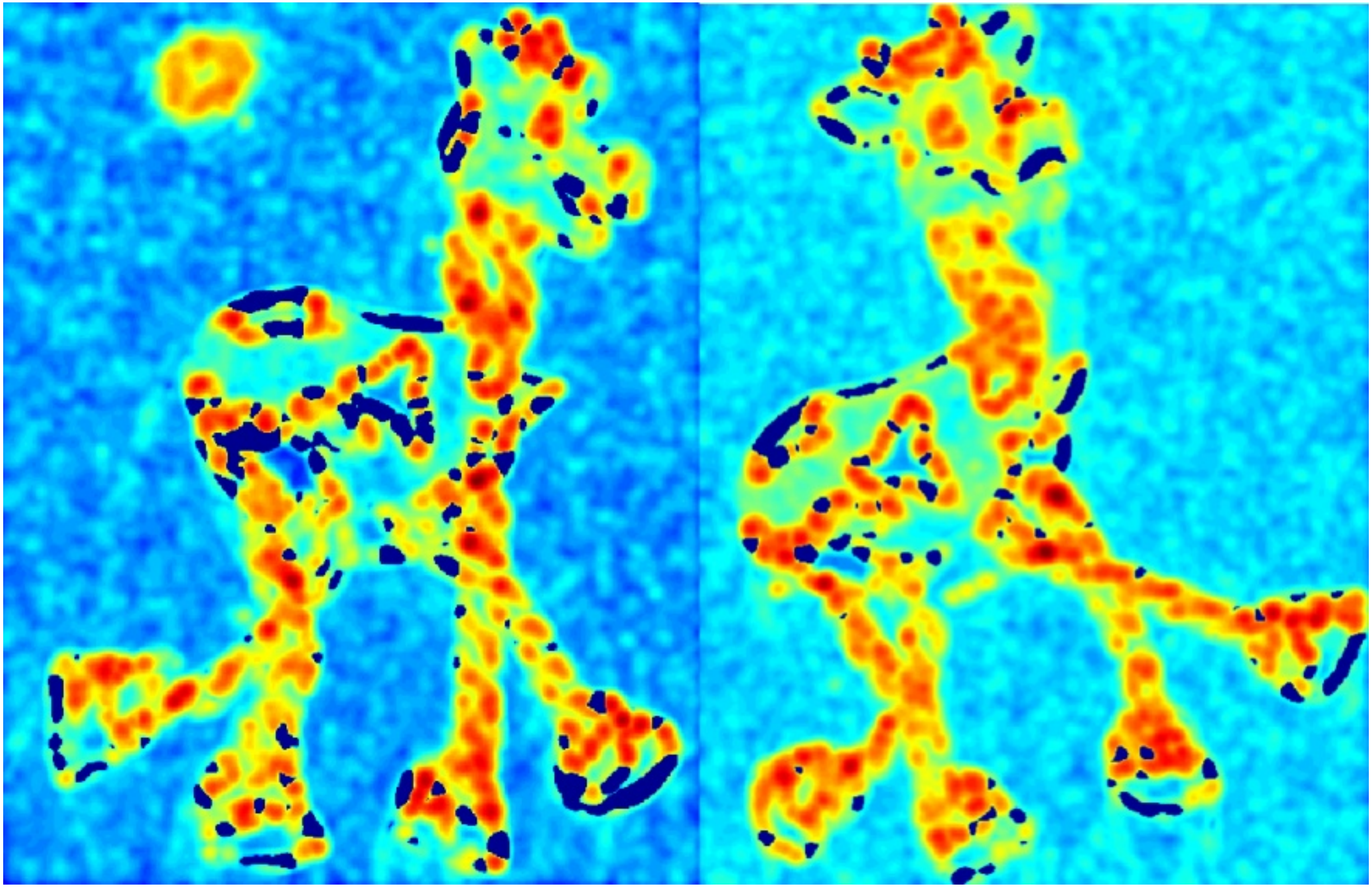
1. Compute derivatives at each pixel
2. Compute covariance matrix H in a Gaussian window around each pixel
3. Compute corner response function f
4. Threshold f
5. Find local maxima of response function (nonmaximum suppression)

C.Harris and M.Stephens. [“A Combined Corner and Edge Detector.”] *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Harris Detector: Steps

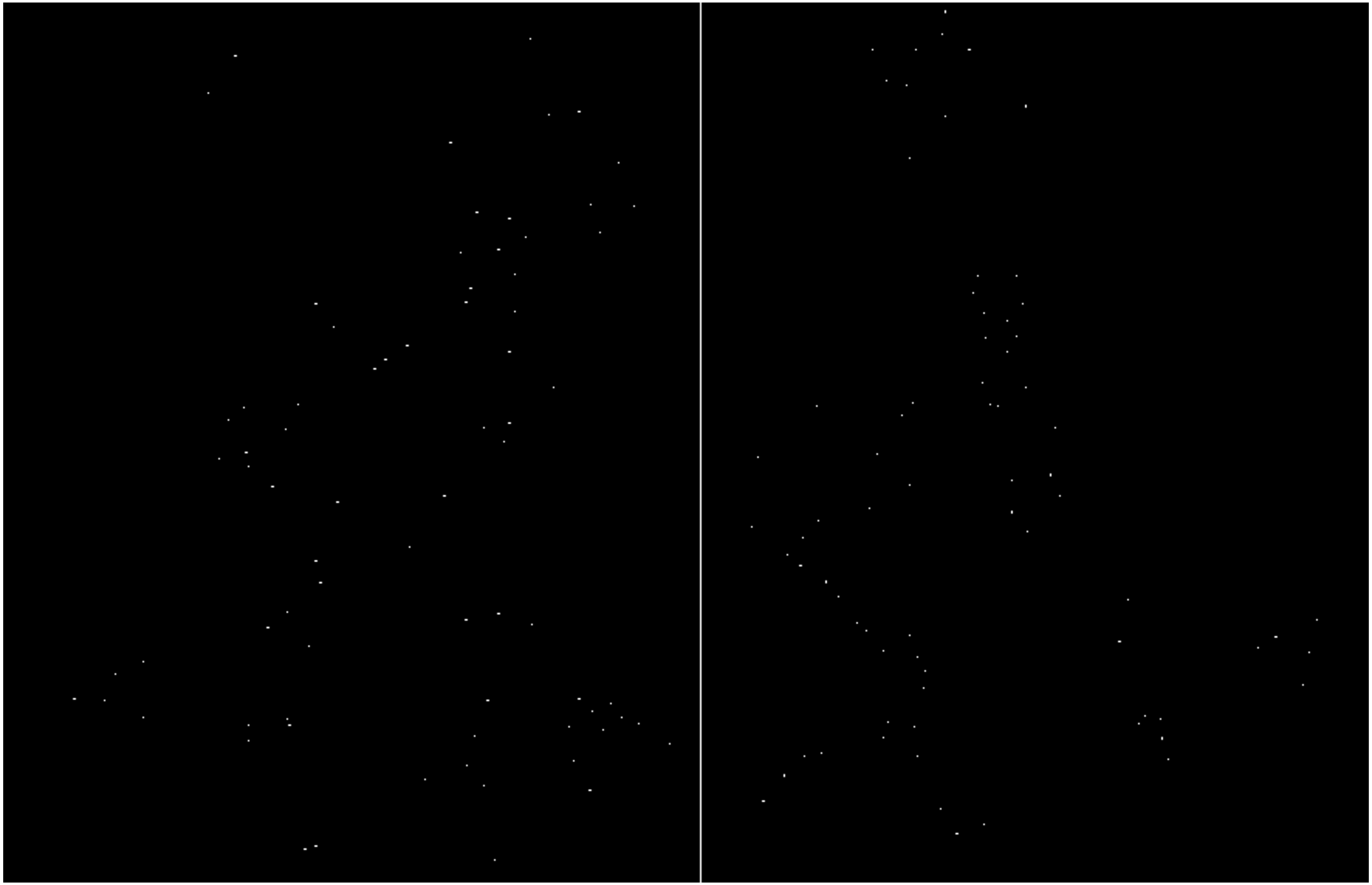
Harris Detector: Steps

Compute corner response f



Harris Detector: Steps

Take only the points of local maxima of *f*



Harris Detector: Steps

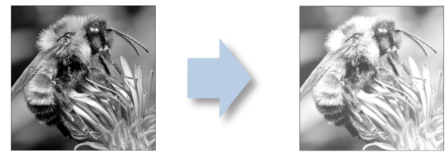
How to make sure the points are **repeatable**？

How to interpret repeatability mathematically: Invariance

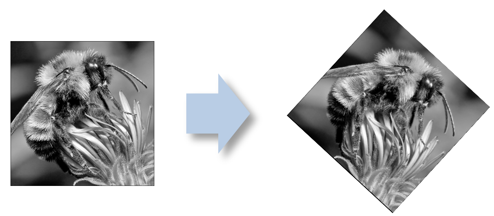
- We want response value *f* at the corresponding pixels to be ***invariant*** to image transformations

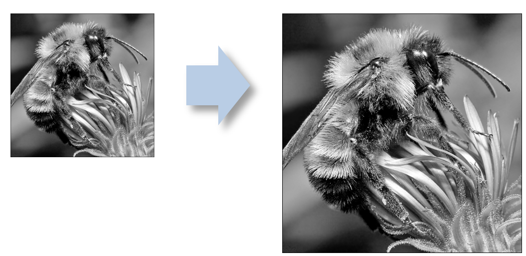
Image transformations

- Photometric （光度）

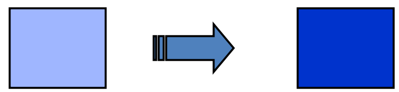
Intensity change  亮度变化

- Geometric （几何）

Rotation  旋转

Scale  尺度

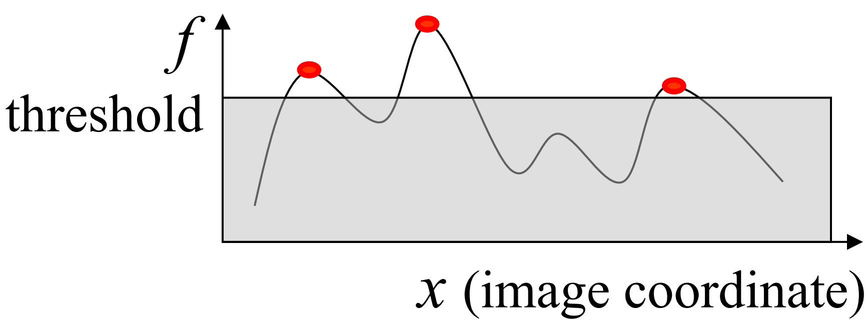
Harris detector: Invariance properties -- photometric transformation

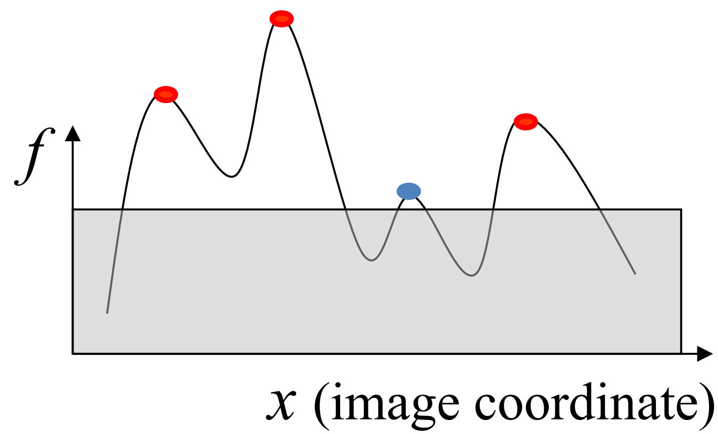


$$I \rightarrow a\,I + b$$

Image derivatives are

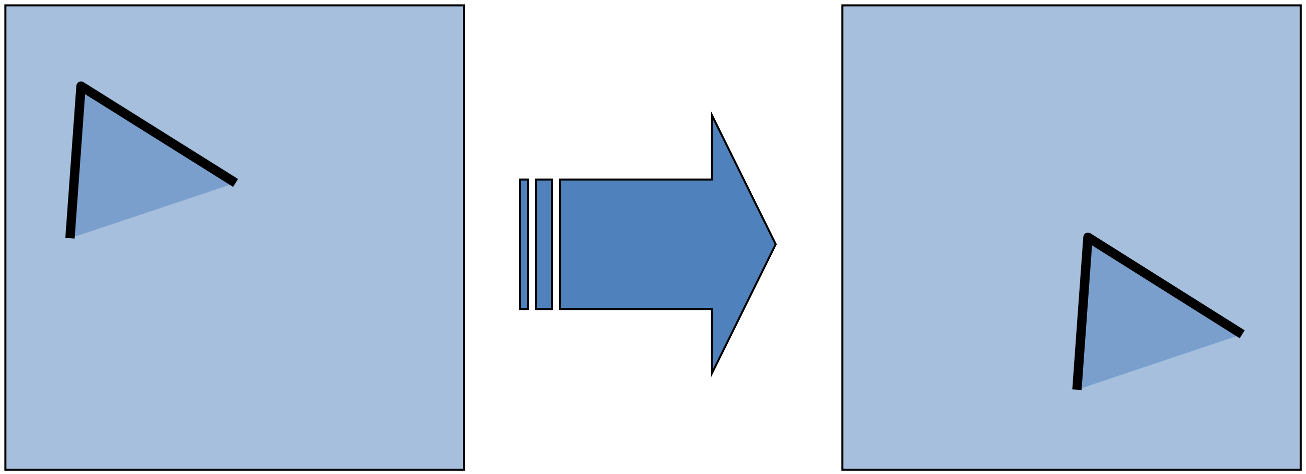
- invariant to intensity shift $I \rightarrow I + b$
- not invariant to intensity scaling: $I \rightarrow a\,I$





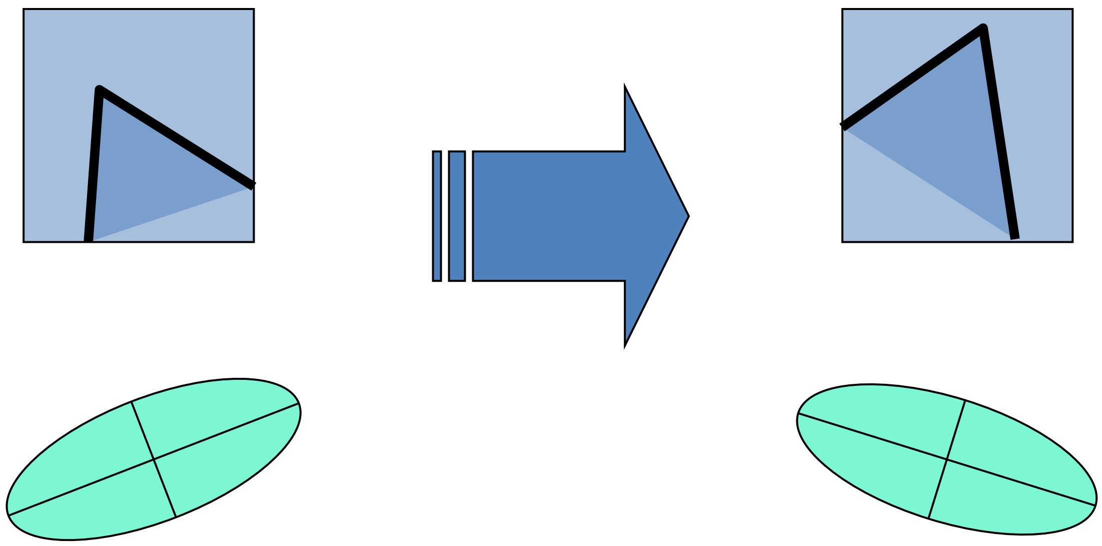
Partially invariant to affine intensity change

Harris detector: Invariance properties -- Image translation



Corner response is invariant w.r.t. translation

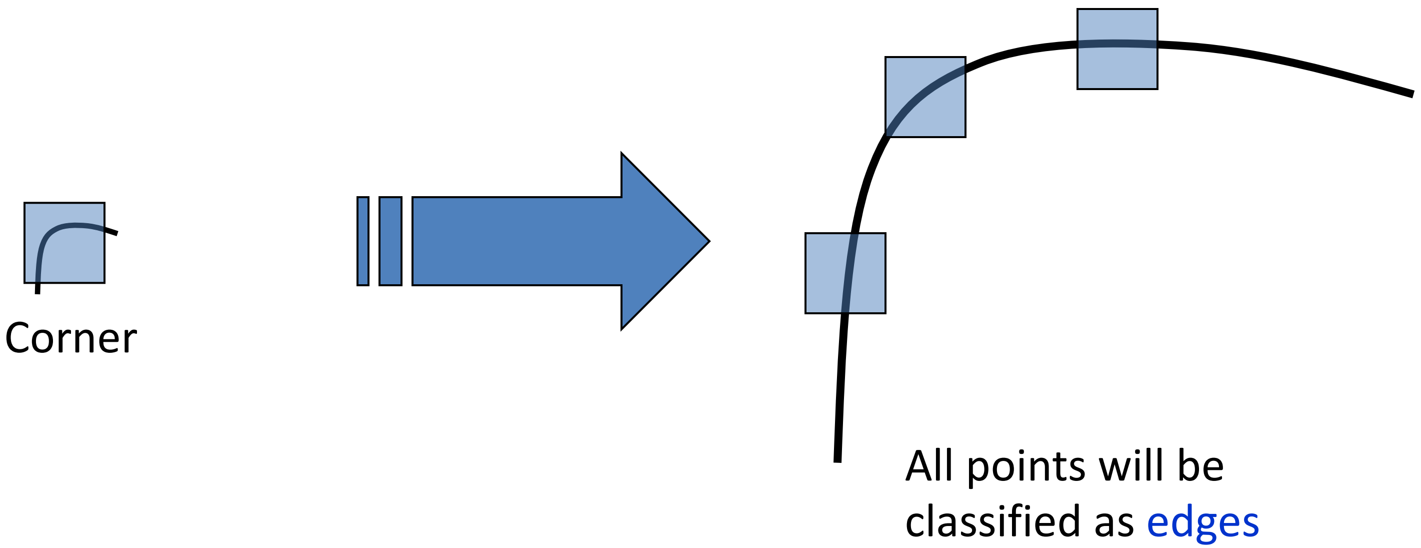
Harris detector: Invariance properties -- Image rotation



Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response is invariant w.r.t. image rotation

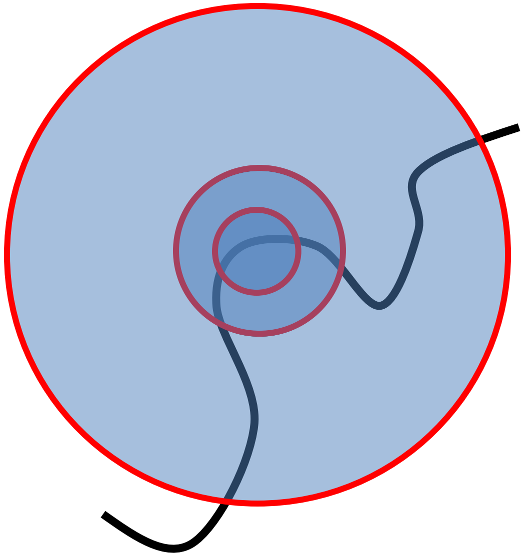
Harris detector: Invariance properties -- Scaling



Corner response is NOT invariant to scaling

Scale invariant detection

Suppose you're looking for corners



Key idea: find scale that gives local maximum of *f*

Automatic Scale Selection

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

How to find corresponding patch sizes?

K. Grauman, B. Leibe

Automatic Scale Selection

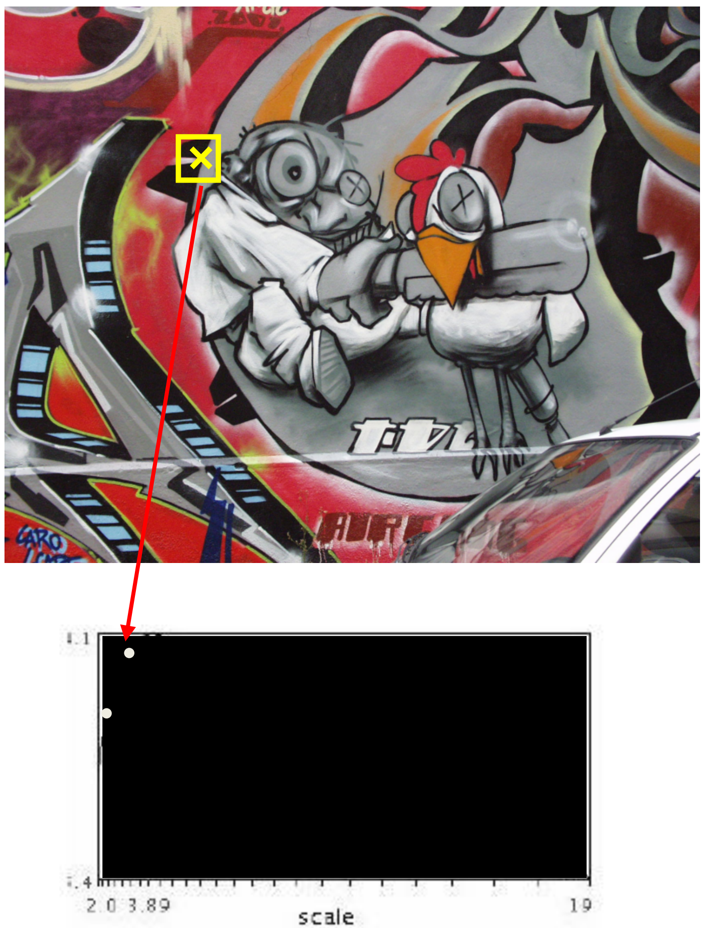
- Function responses for increasing scale

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

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

Automatic Scale Selection

- Function responses for increasing scale



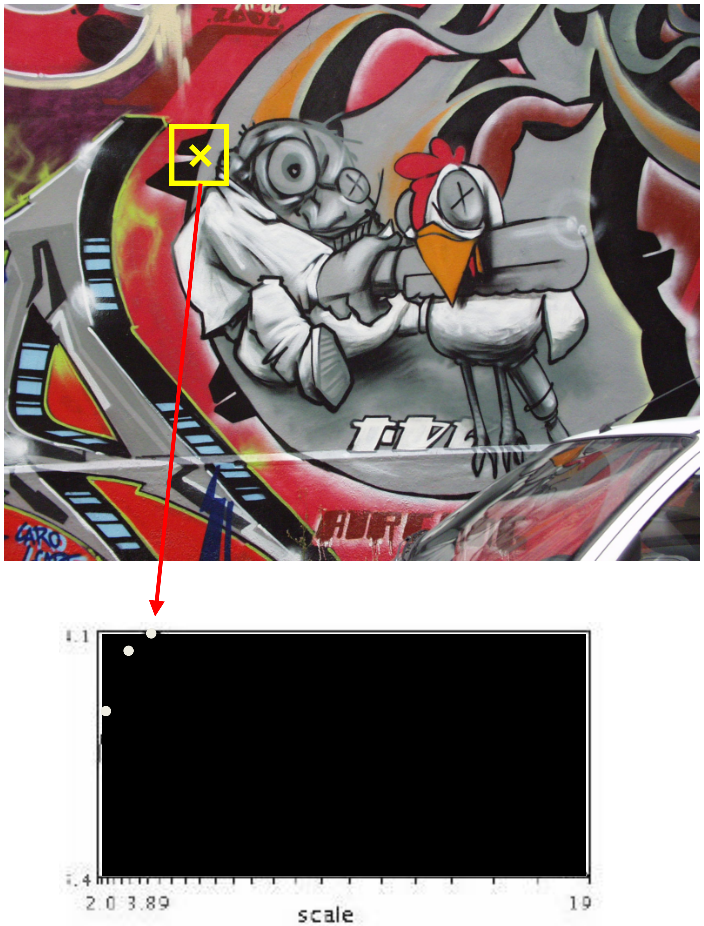
$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale



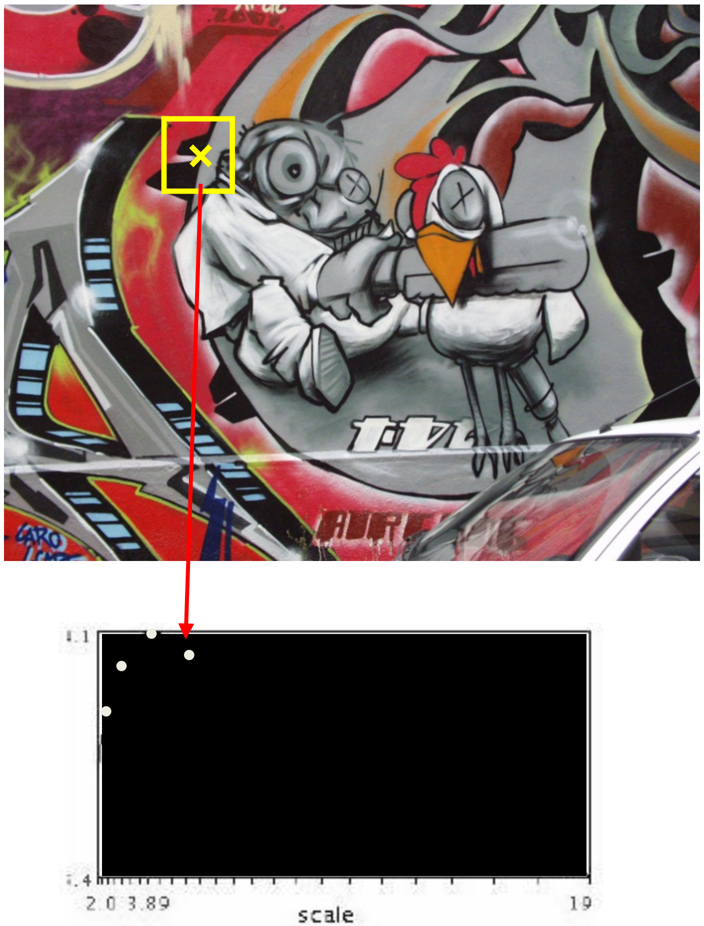
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

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

K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale



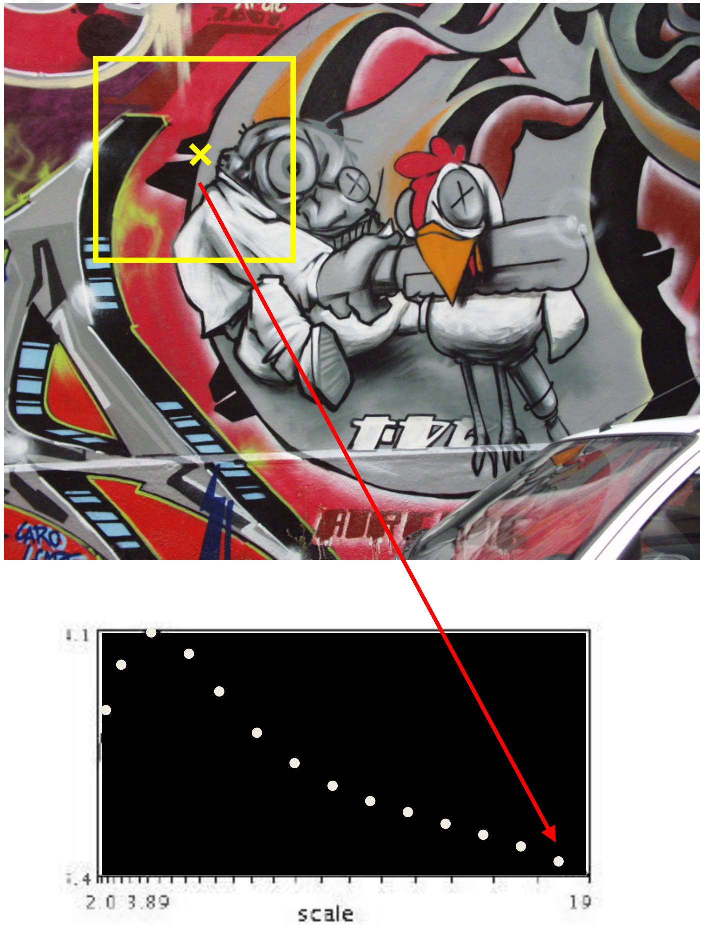
$f(I_{i_1 \ldots i_m}(x,\sigma))$

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

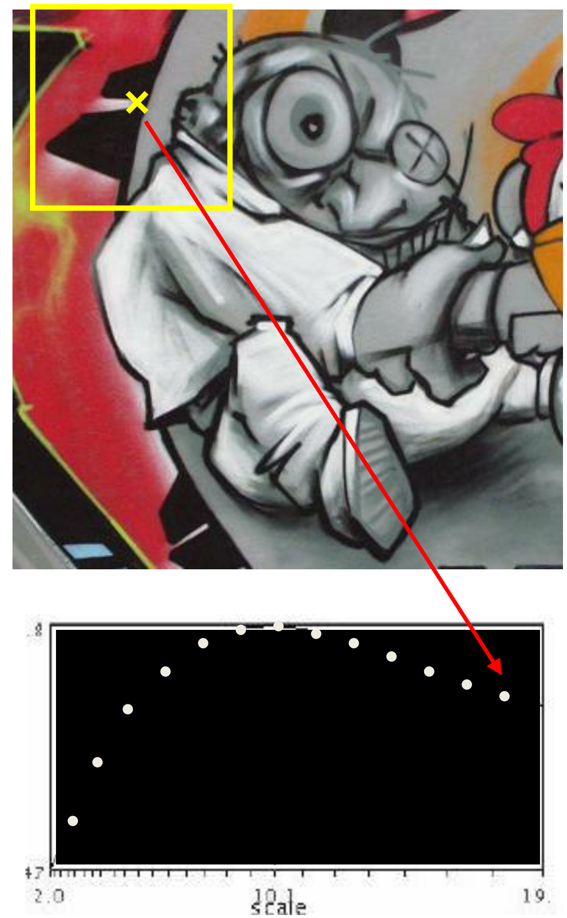
K. Grauman, B. Leibe

Automatic Scale Selection

- Function responses for increasing scale



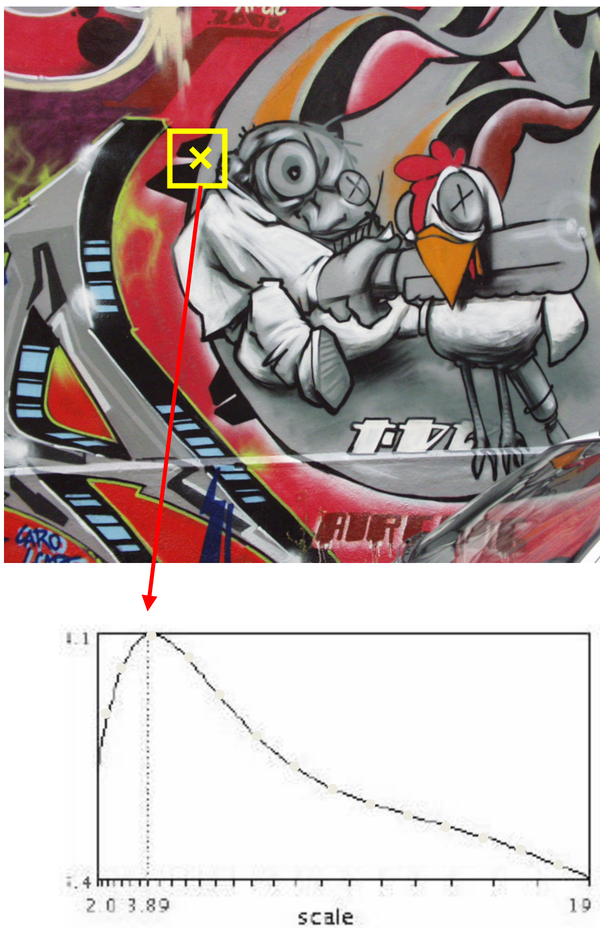
$$f(I_{i_1 \ldots i_m}(x, \sigma))$$



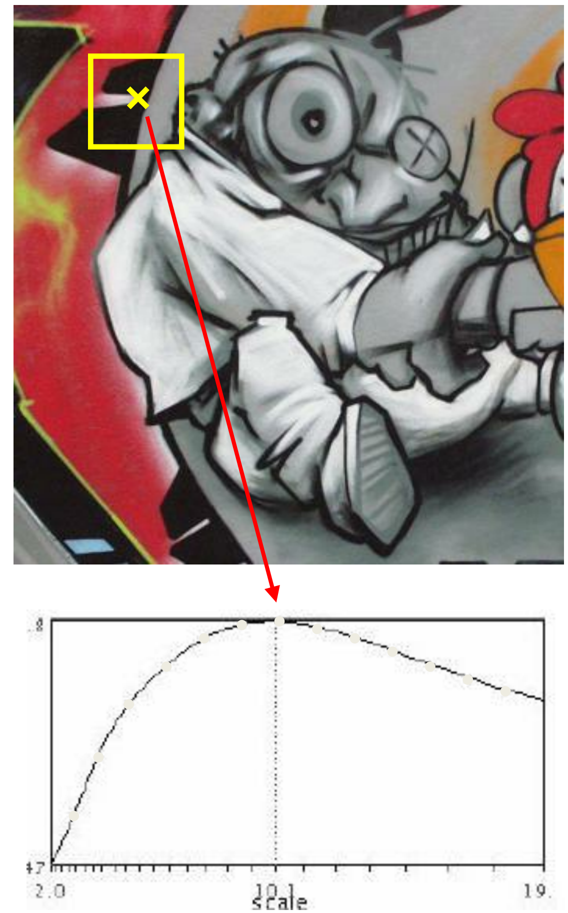
$$f(I_{i_1 \ldots i_m}(x', \sigma))$$

Automatic Scale Selection

- Function responses for increasing scale



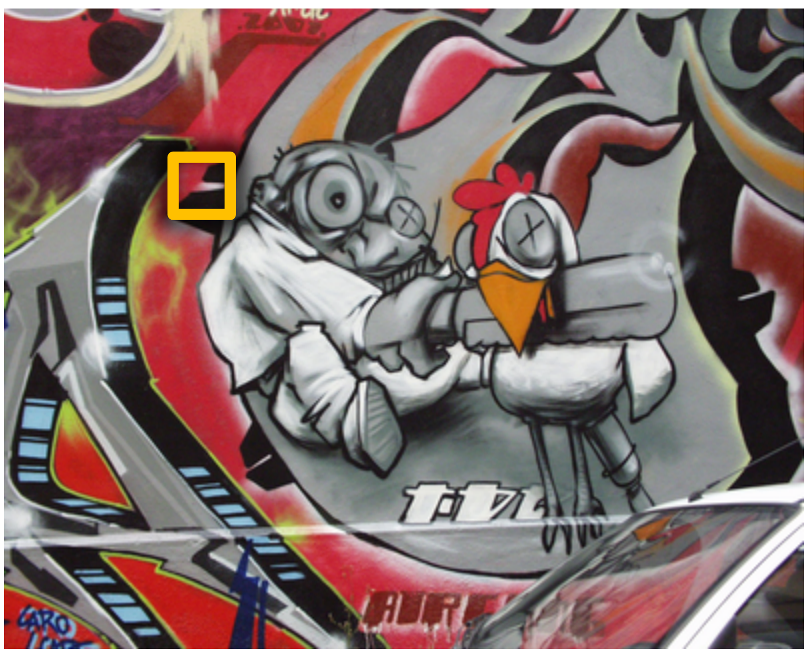
$f(I_{i_1 \ldots i_m}(x, \sigma))$



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

Implementation

- Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with an image pyramid



(sometimes need to create in-between levels, e.g. a ¾-size image)

Blob detector

- Blobs are good features

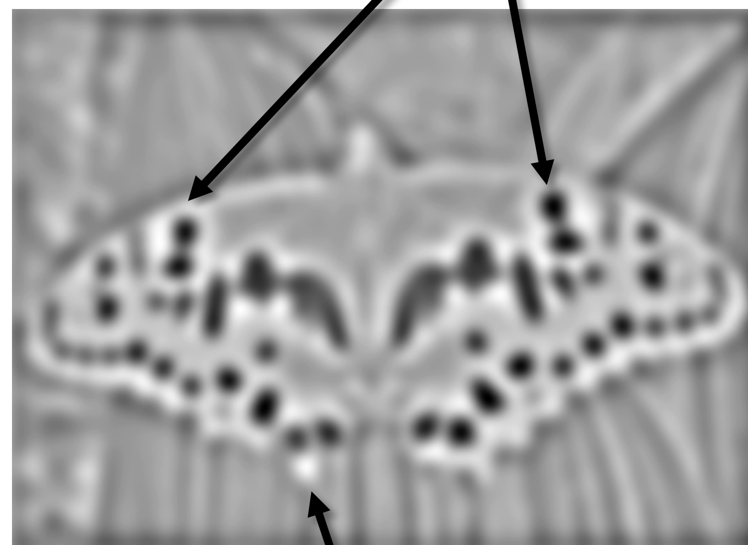
- How to find blobs?

Blob detector

- Blobs are have large second derivatives in image intensity
  - Compute Laplacian of image
  - Find maxima and minima of Laplacian

$I_{xx} + I_{yy}$

minima



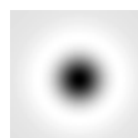
maximum

How to compute Laplacian?

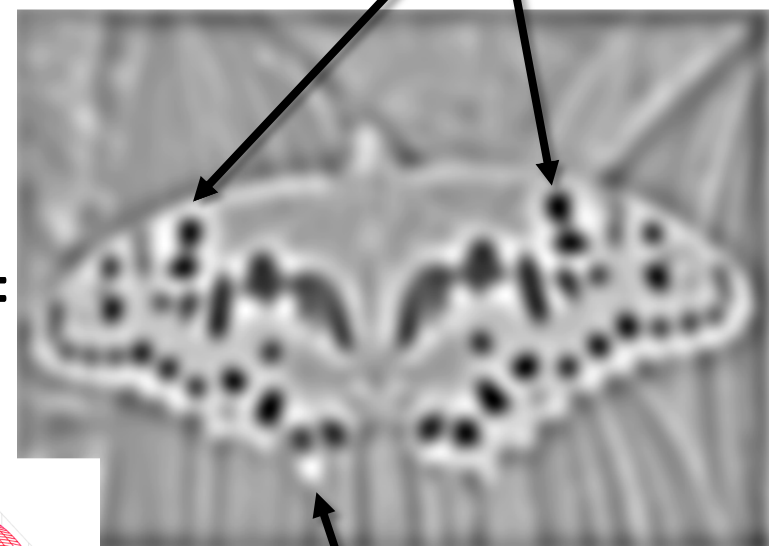
Laplacian of Gaussian Filter

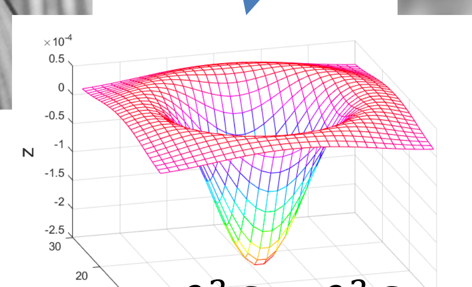
- We usually compute Laplacian of Image using a Laplacian of Gaussian (LoG) filter

$*$



$=$





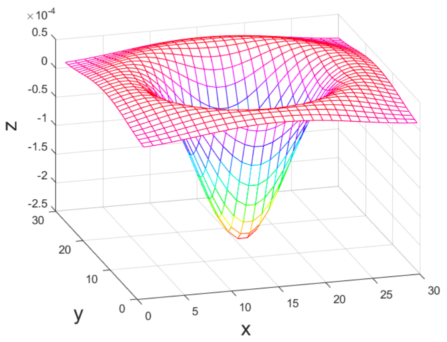
$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

G is 2D Gaussian function

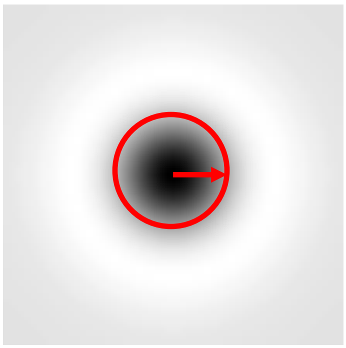
Laplacian of Gaussian Filter

- LoG is equivalent to
  - Smooth image with a Gaussian filter
  - Compute Laplacian

$$\underbrace{\nabla^2 (f(x,y) \otimes G(x,y))}_{\text{Laplacian of Gaussian-filtered image}} = \underbrace{\nabla^2 G(x,y) \otimes f(x,y)}_{\text{Laplacian of Gaussian (LoG) -filtered image}}$$



- The scale of LoG is controlled by σ of Gaussian



Scale selection

- At what scale does the Laplacian achieve a maximum response for a binary circle of radius r?

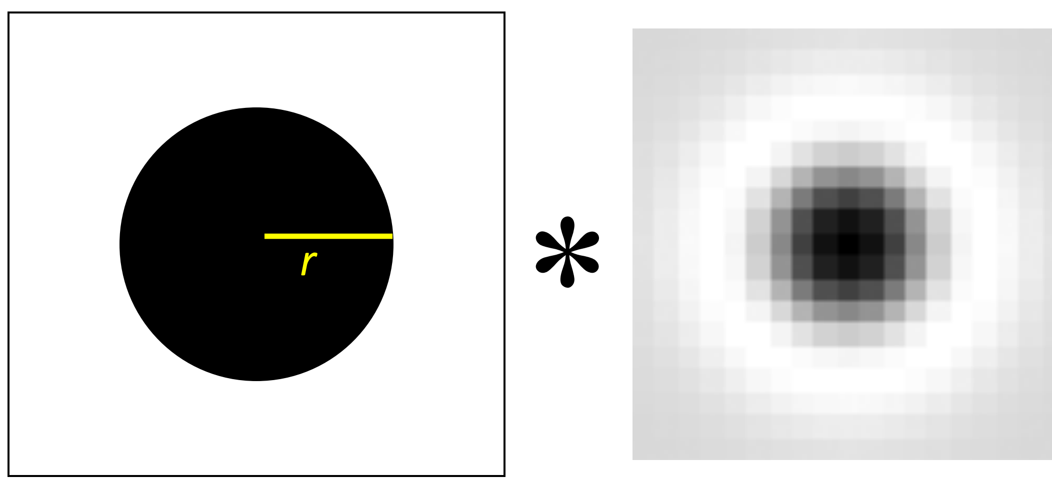
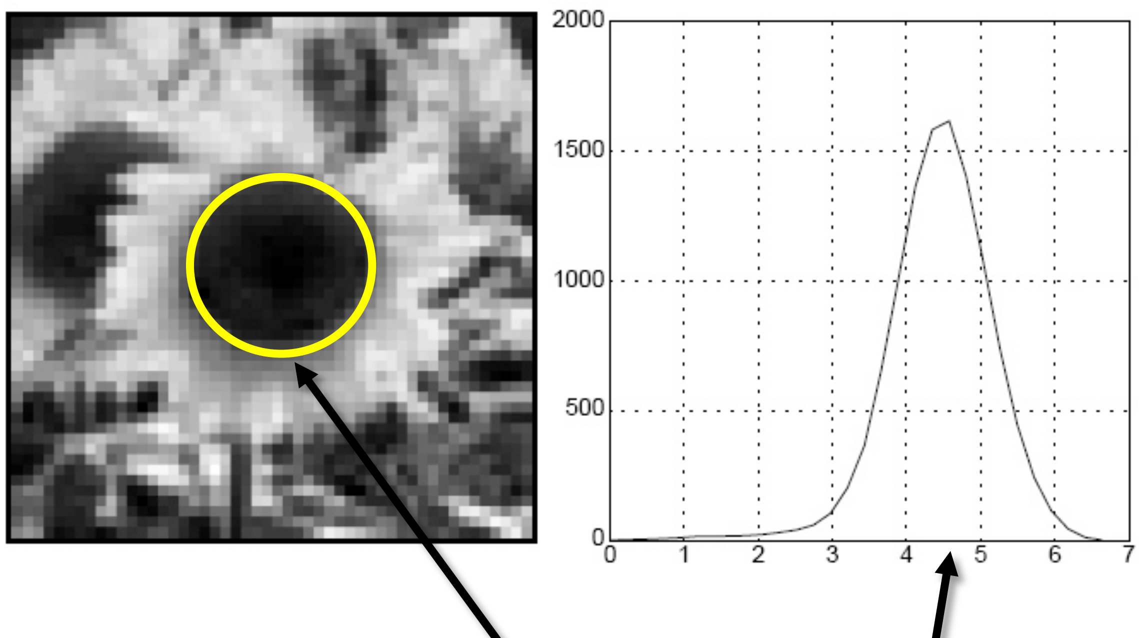


image Laplacian

Scale selection

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

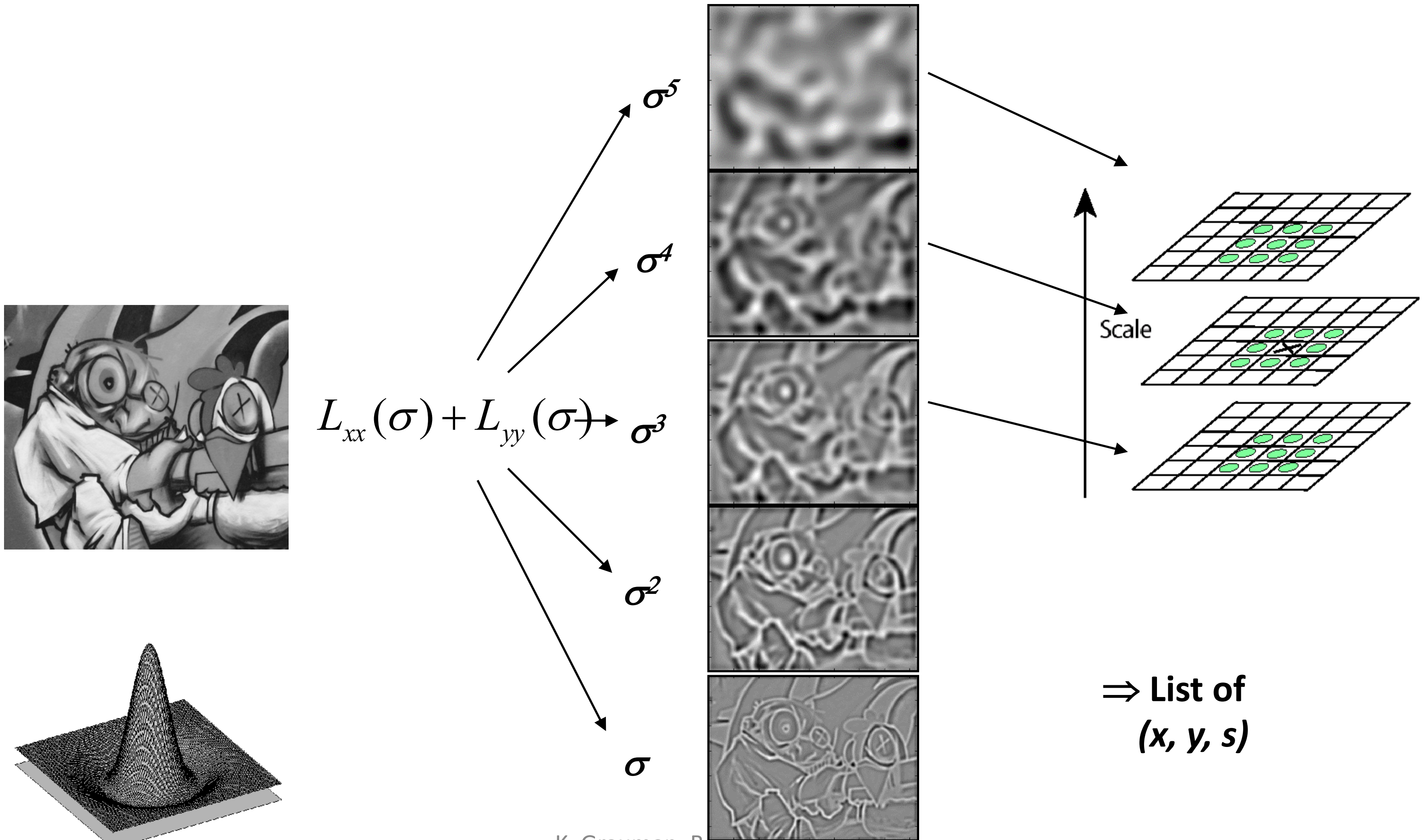


characteristic scale

T. Lindeberg (1998). ["Feature detection with automatic scale selection."]() *International Journal of Computer Vision* **30** (2): pp 77--116.

Scale invariant interest points

Interest points are local maxima in both position and scale



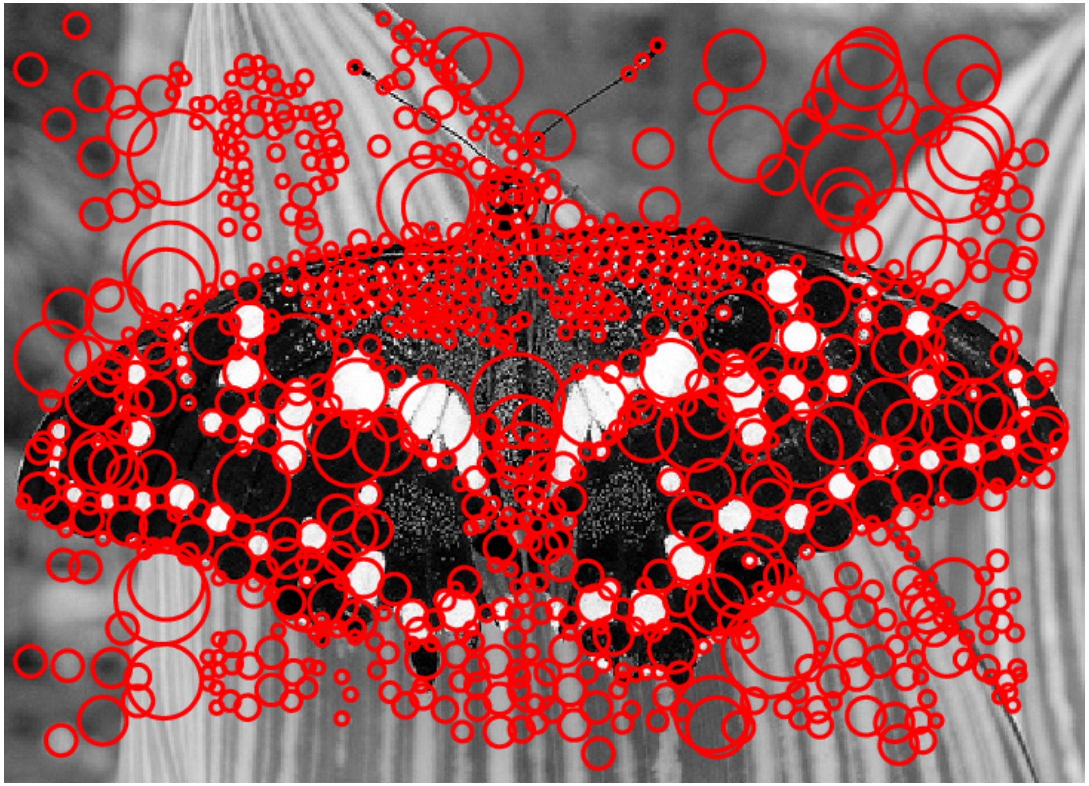
K. Grauman, B. Leibe

Example

Example

sigma = 11.9912

Example

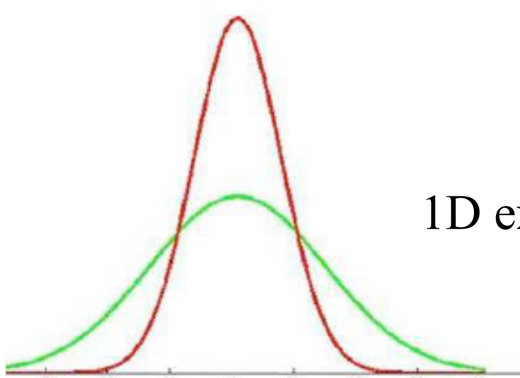


Difference-of-Gaussian (DoG)

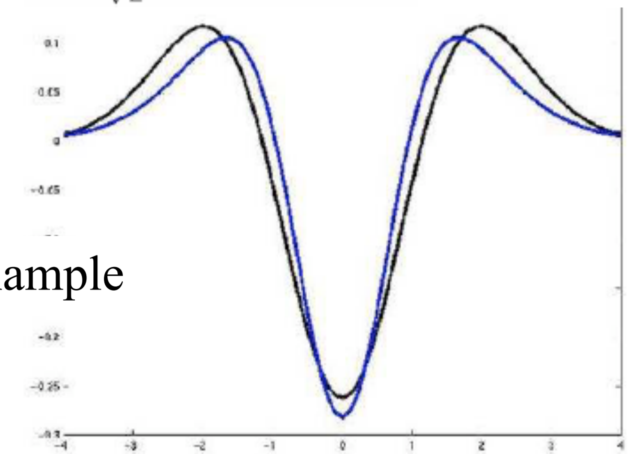
- LoG can be approximated by Difference of two Gaussians (DoG)

$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

Best approximation when:
$\sigma_1 = \frac{\sigma}{\sqrt{2}}$, $\sigma_2 = \sqrt{2}\sigma$



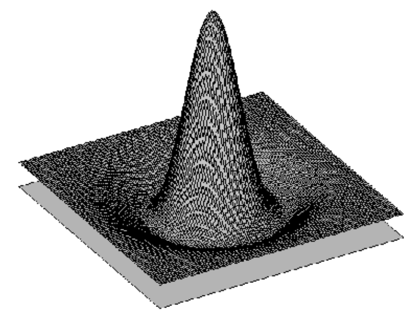
1D example



Slide credit: Robert Collins

Difference-of-Gaussian (DoG)

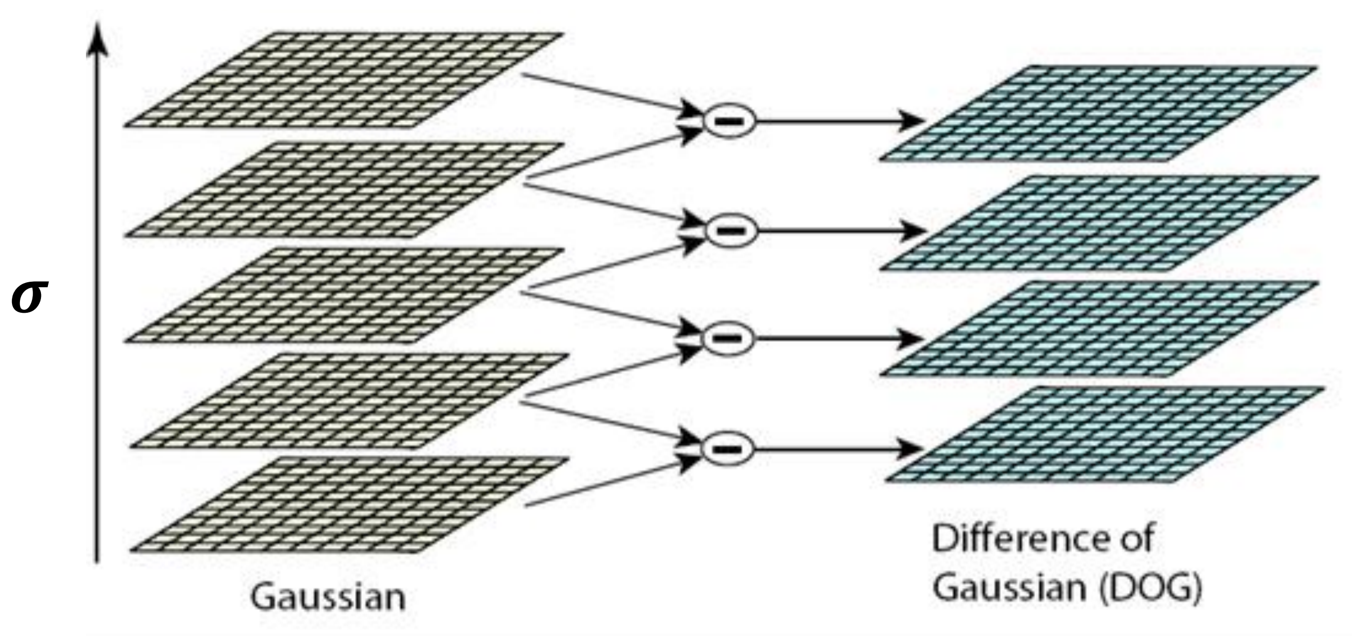
- LoG can be approximated by Difference of two Gaussians (DoG)



K. Grauman, B. Leibe

Difference-of-Gaussian (DoG)

- Computing DoG at different scales



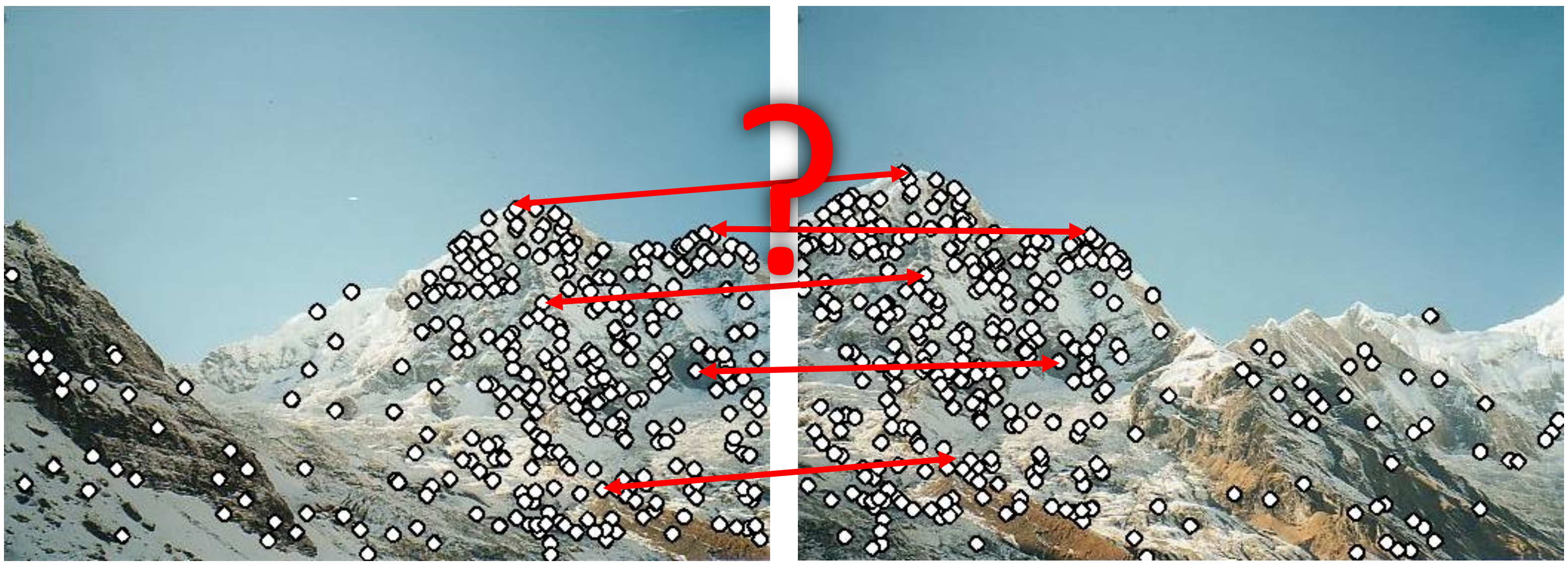
Summary

- What is a good interest point?
  - Unique
  - Invariant to transformations
- Popular detectors
  - Harris corner detector
  - Blob detector (e.g., LoG and DoG)

Feature descriptors （描述子）

We know how to detect good points

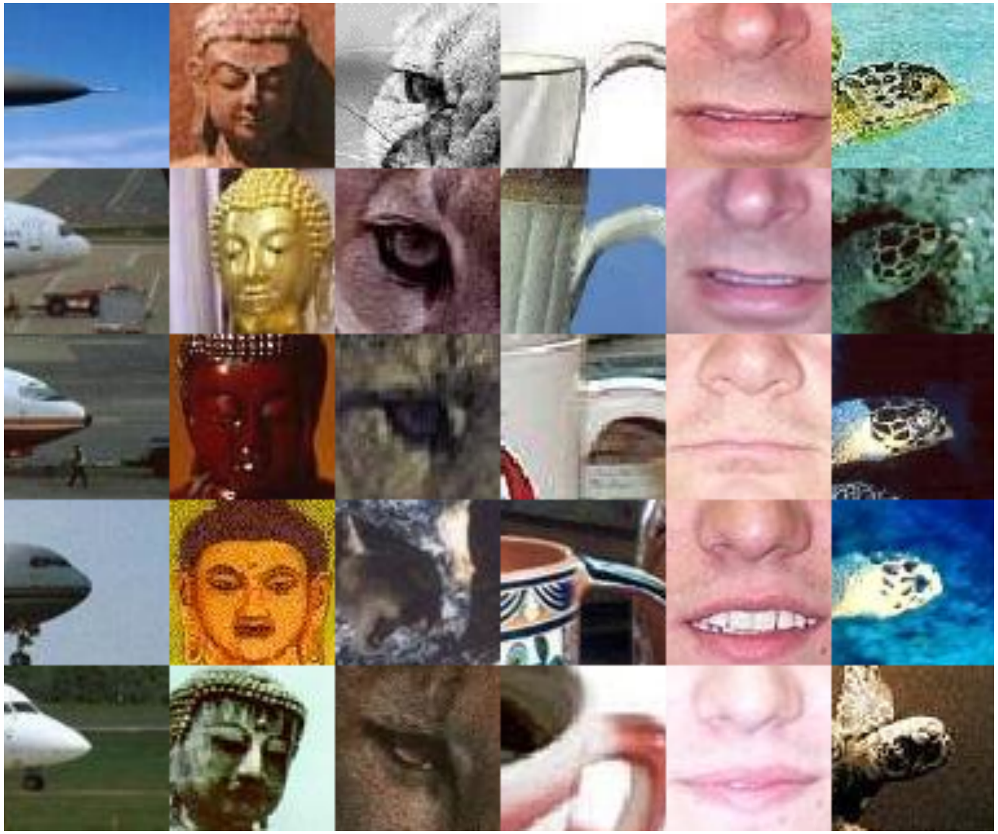
Next question: **How to match them?**



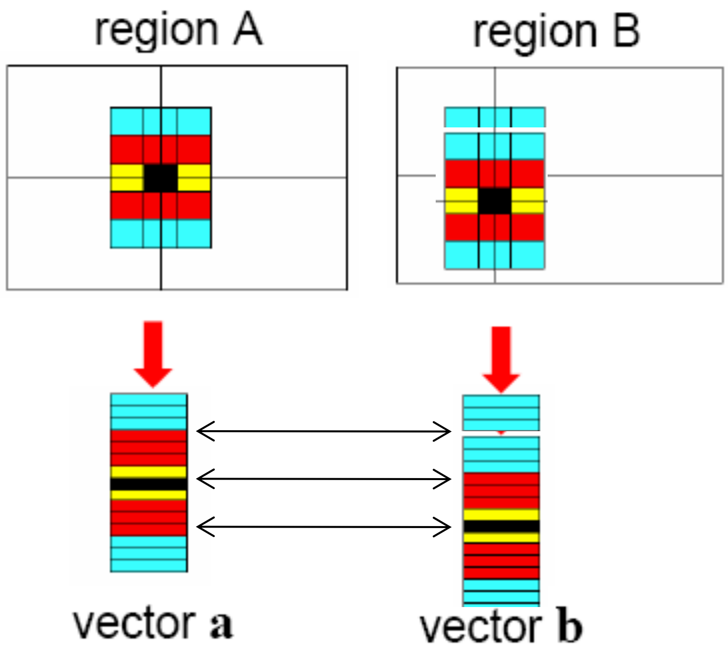
Answer: Extract a *descriptor* for each point, find similar descriptors between the two images

How do we describe an image patch?

Patches with similar content should have similar descriptors.



Raw patches as local descriptors

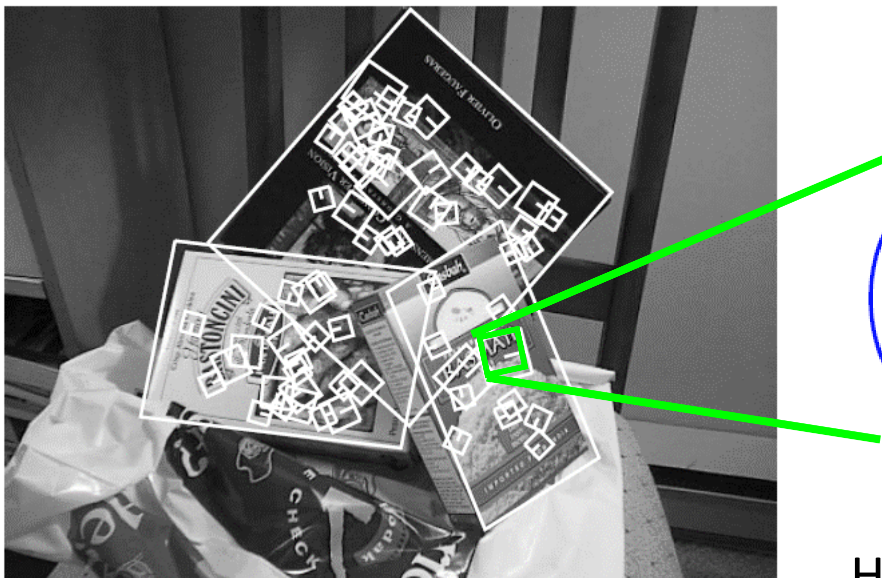


The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector.

But this is very sensitive (not invariant) to even small shifts, rotations.

SIFT descriptor

Scale Invariant Feature Transform (SIFT)



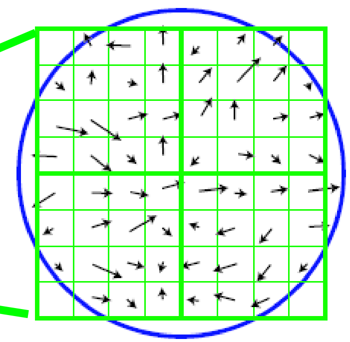
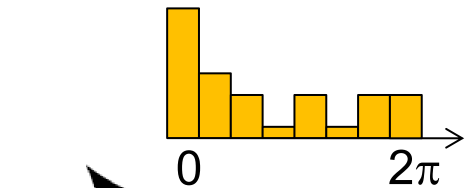
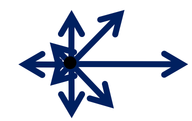


Image gradients





Histogram of oriented gradients

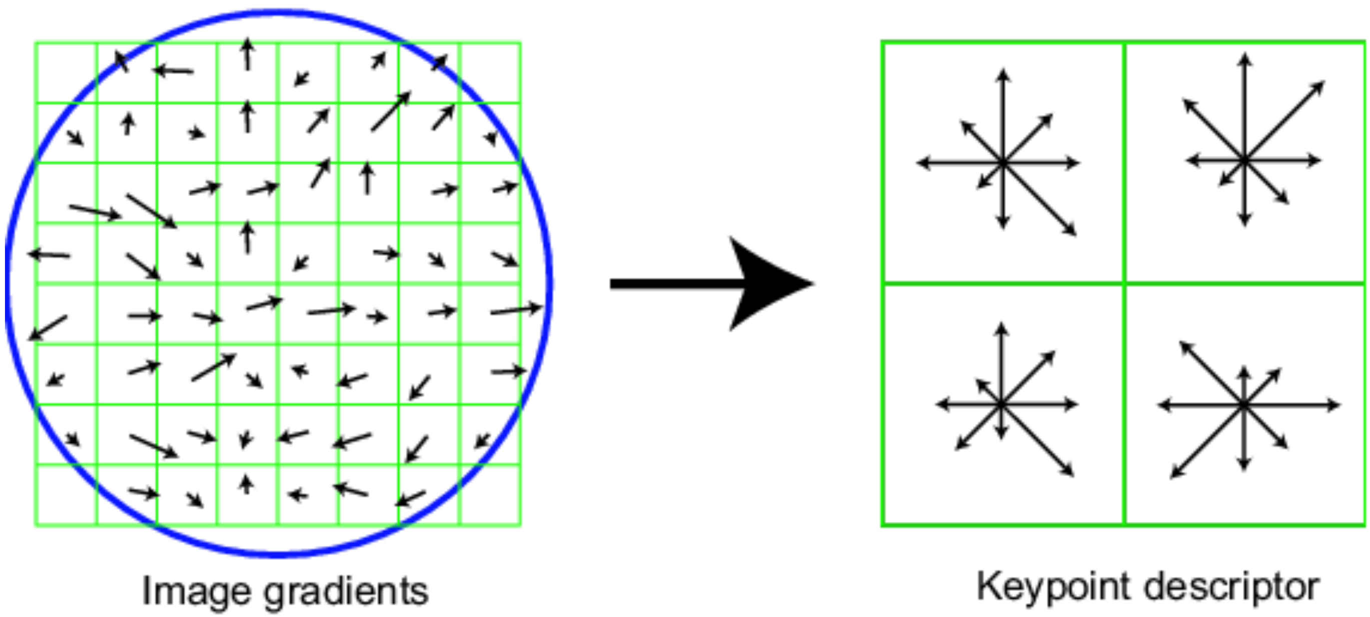
- Captures important texture information
- Robust to small translations / affine deformations

[Lowe, ICCV 1999]

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

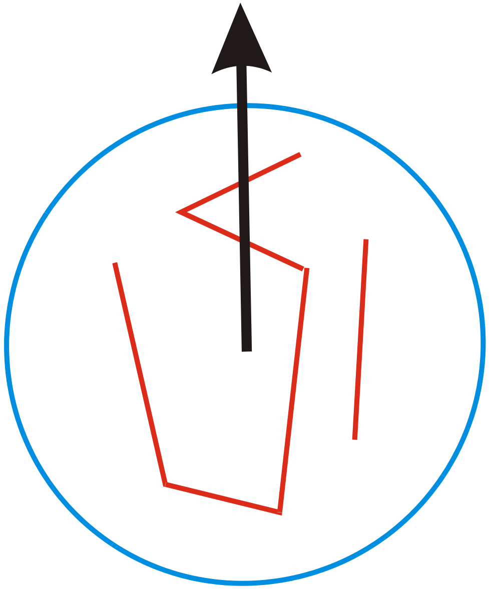


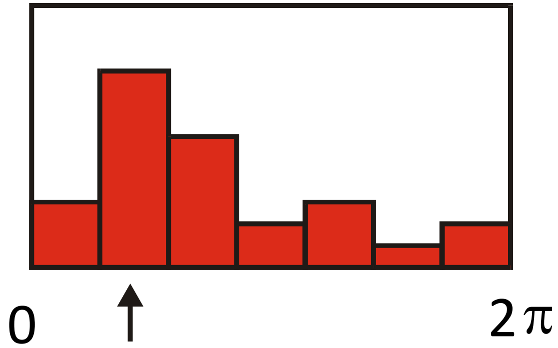
Adapted from slide by David Lowe

Orientation Normalization

[Lowe, SIFT, 1999]

- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation





Lowe's SIFT algorithm

- Run DoG detector
  - Find maxima in location/scale space
  - Remove edge points

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

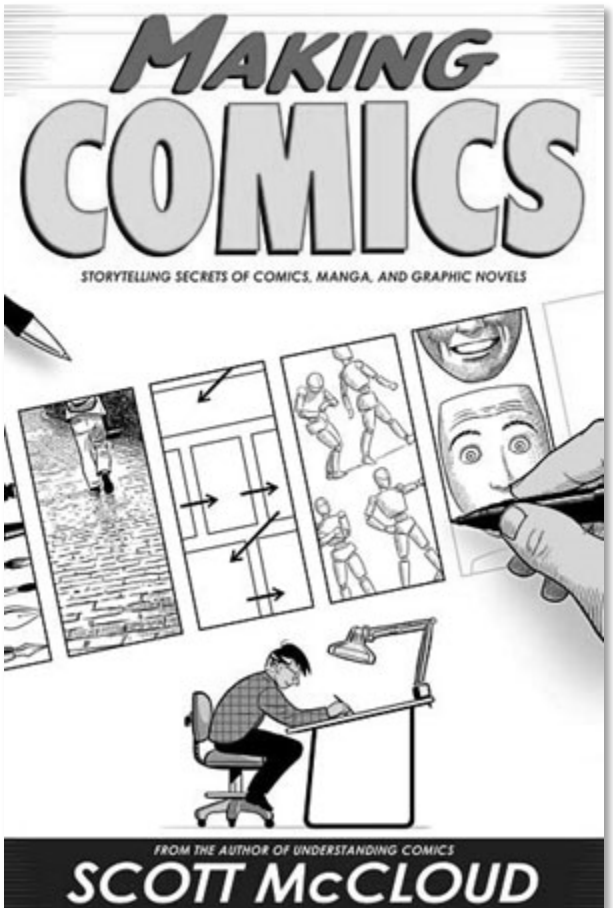
$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

- Find dominate orientation

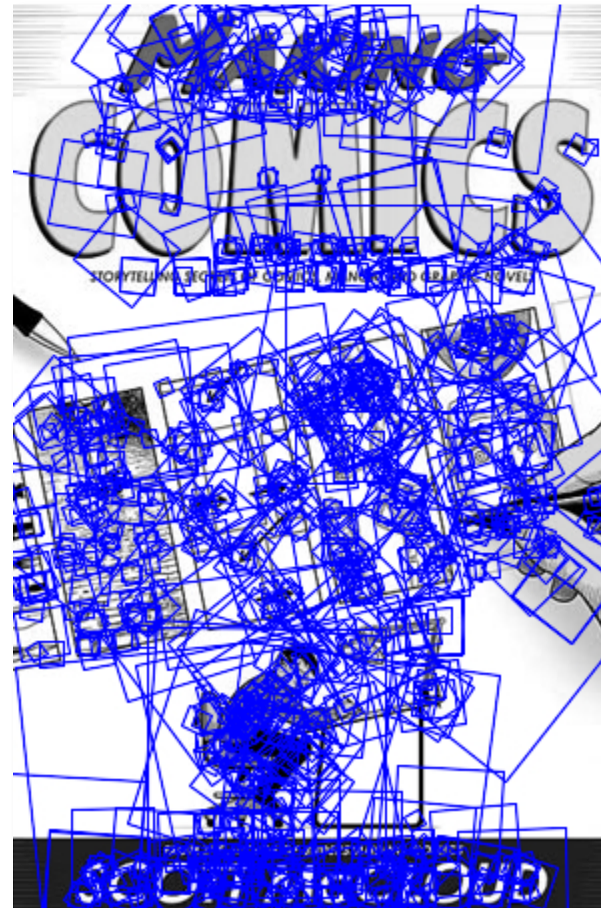
- For each (x,y,scale,orientation), create descriptor

Lowe IJCV 2004

SIFT Example



sift



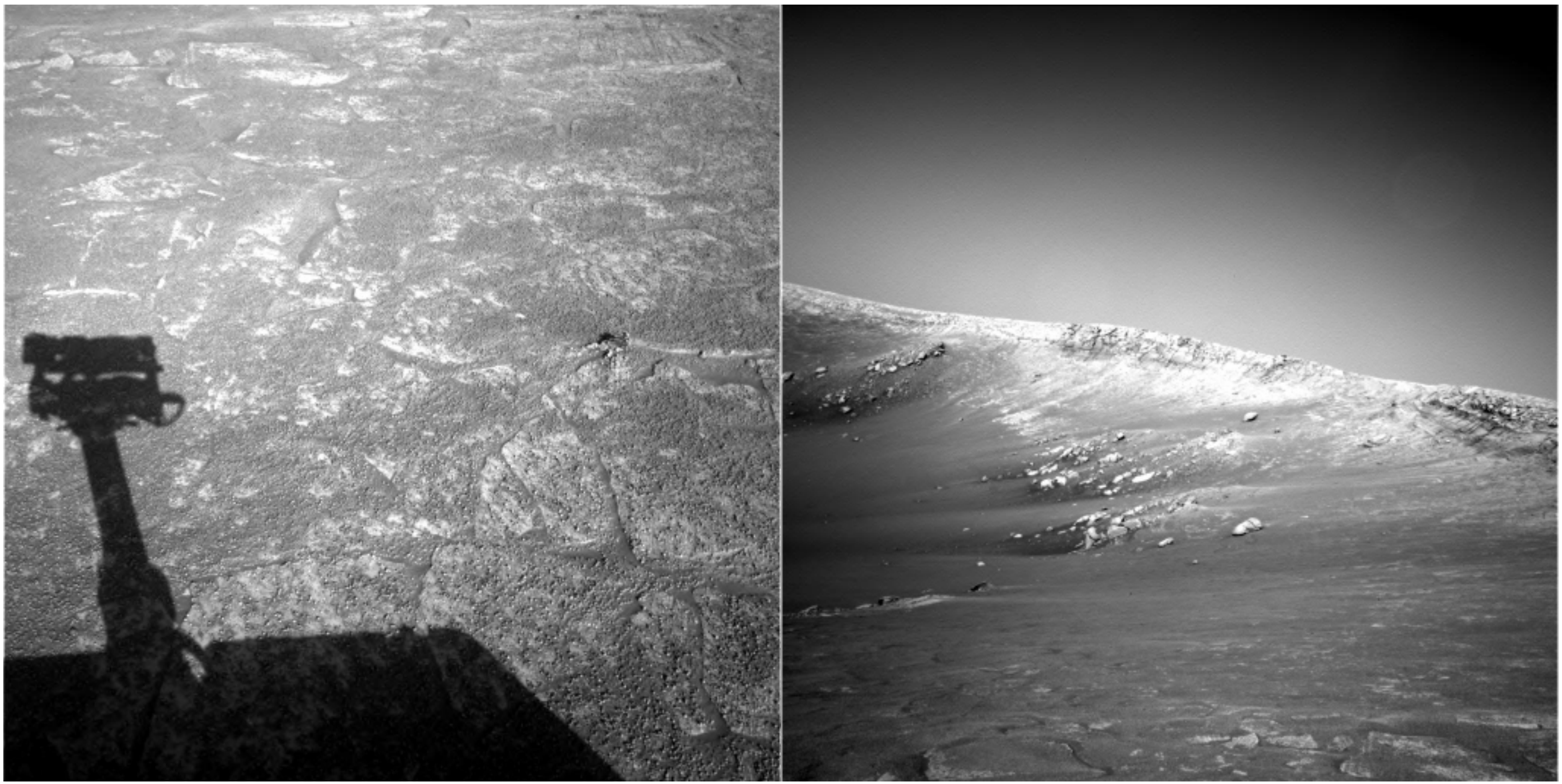
868 SIFT features

Properties of SIFT

Extraordinarily robust matching technique

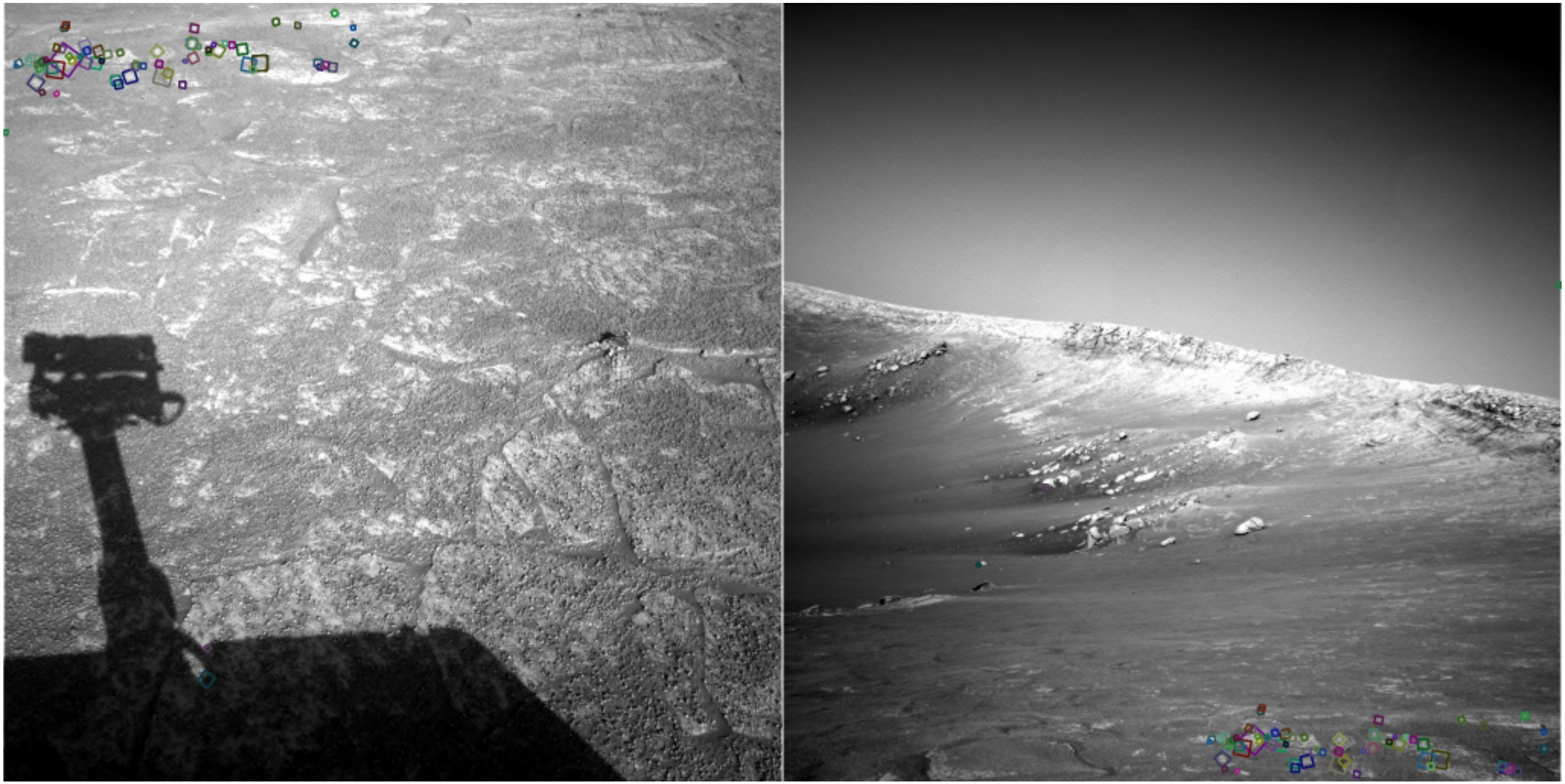
- Can handle changes in viewpoint
  - Theoretically invariant to scale and rotation (why?)
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

Example



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

Example



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

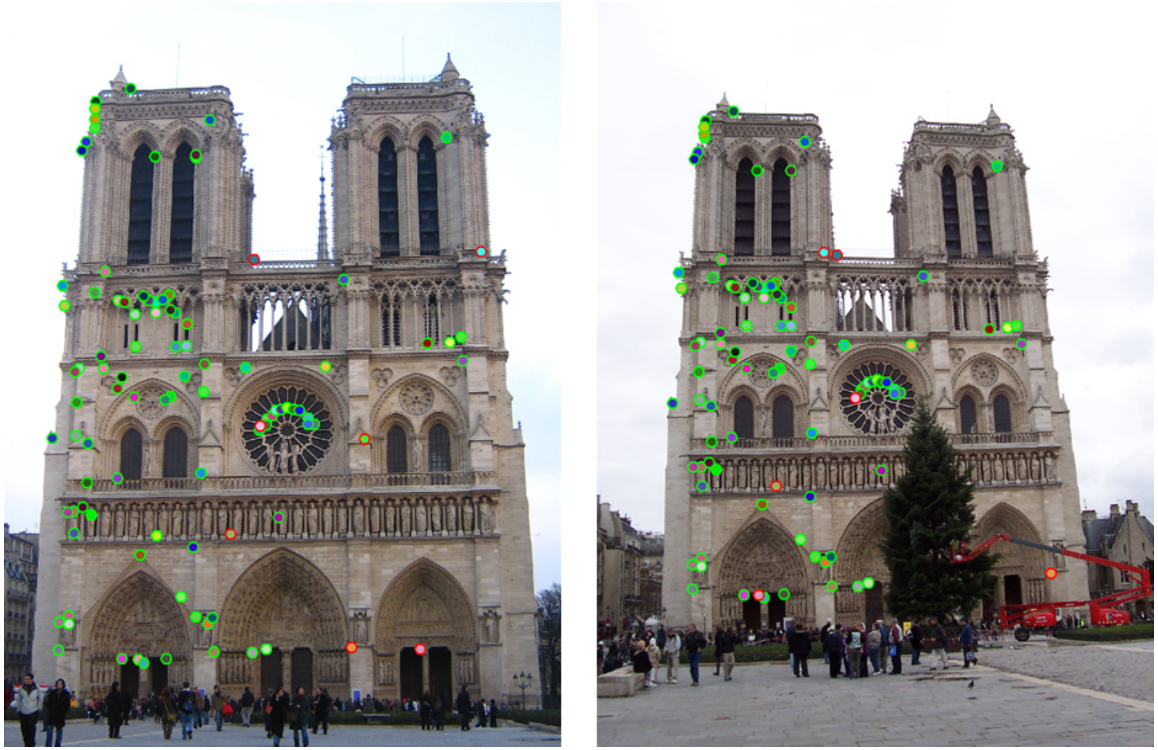
Other detectors and descriptors

- **HOG: Histogram of oriented gradients**
  - Dalal & Triggs, 2005
- **SURF: Speeded Up Robust Features**
  - Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008
- **FAST (corner detector)**
  - Rosten. Machine Learning for High-speed Corner Detection, 2006.
- **ORB: an efficient alternative to SIFT or SURF**
  - Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011
- **Fast Retina Key- point (FREAK)**
  - A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. CVPR 2012 Open Source Award Winner.

Feature matching

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

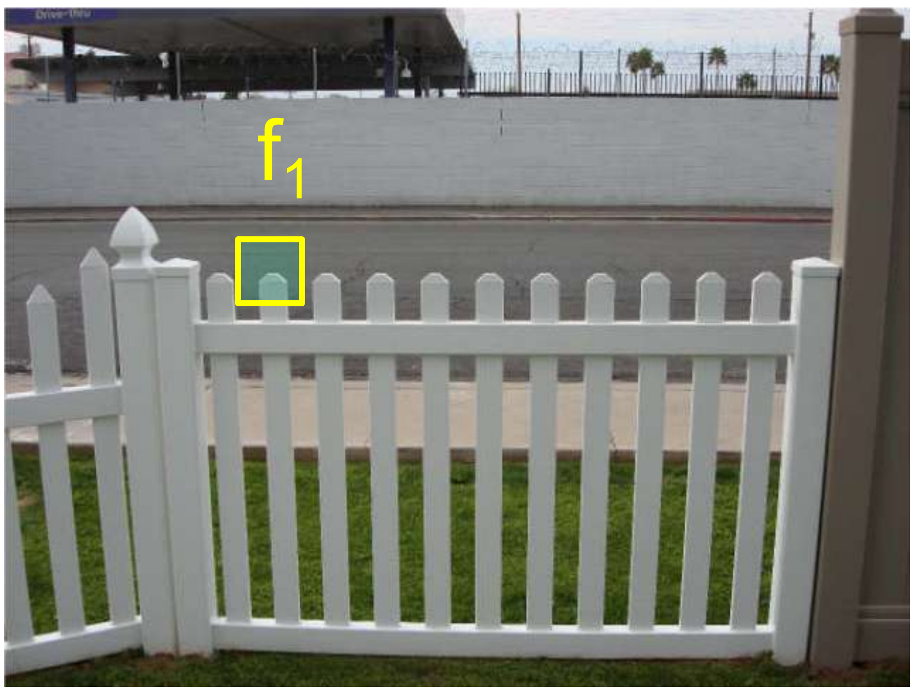
1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance



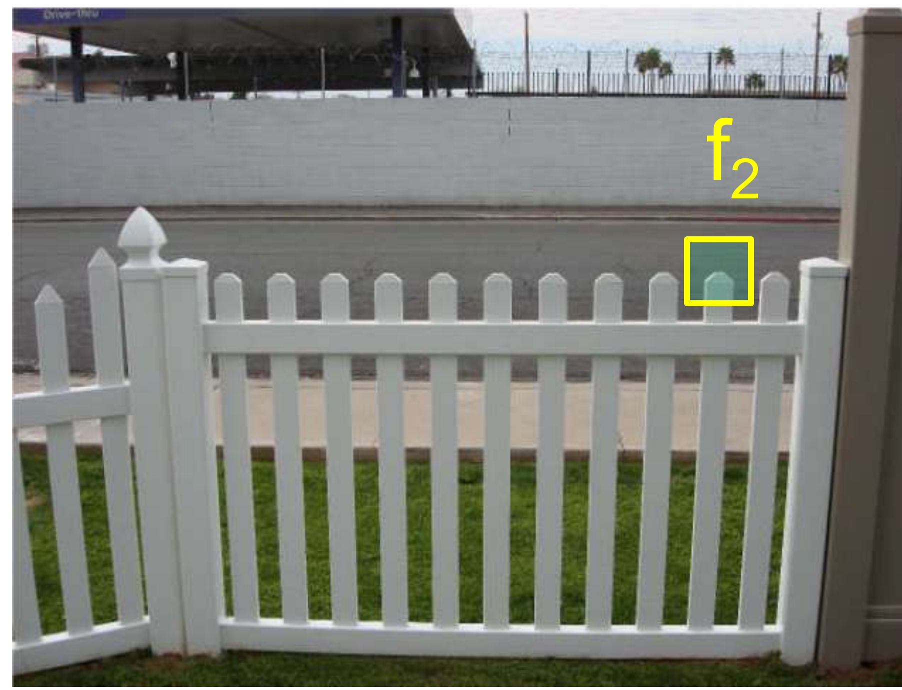
Feature distance

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

- Simple approach: L_2 distance, $||f_1 - f_2||$
- can give small distances for ambiguous (incorrect) matches



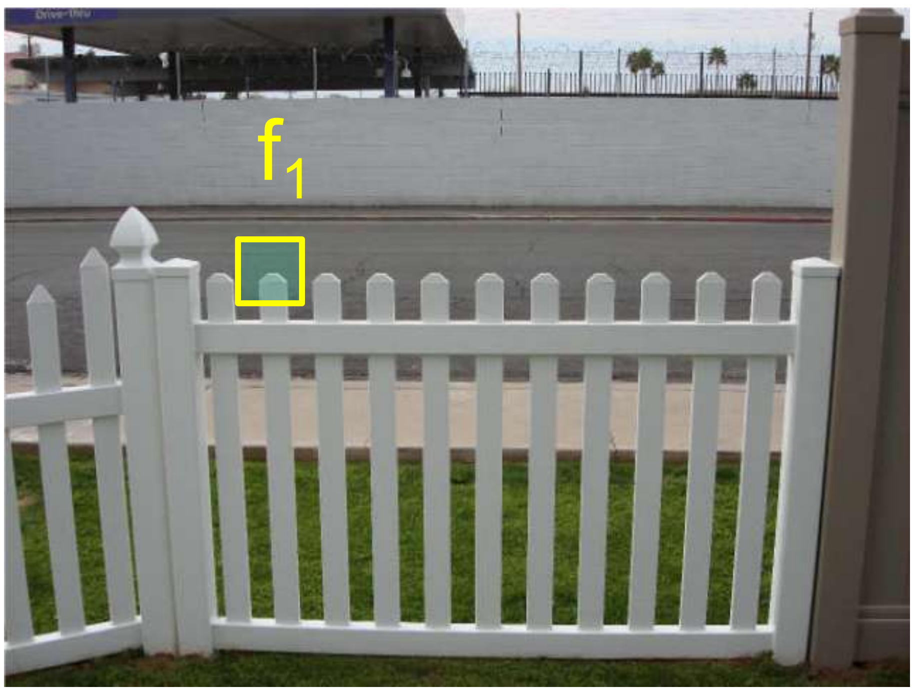
I_1



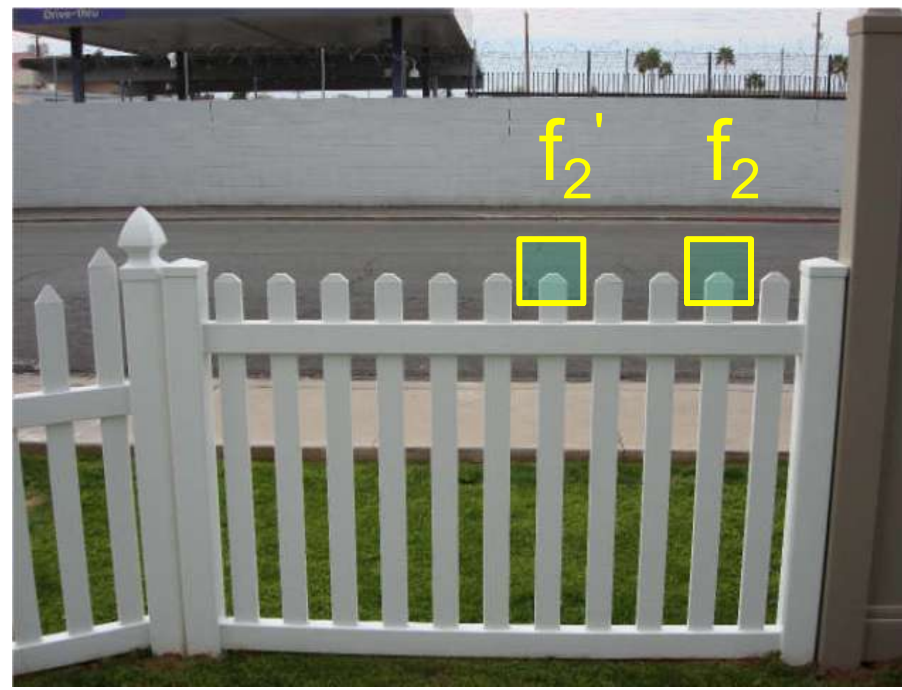
I_2

Ratio test

- Ratio score = $||f_1 - f_2|| / || f_1 - f_2' ||$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2^{nd} best SSD match to f_1 in I_2
- Ambiguous matches have large ratio scores

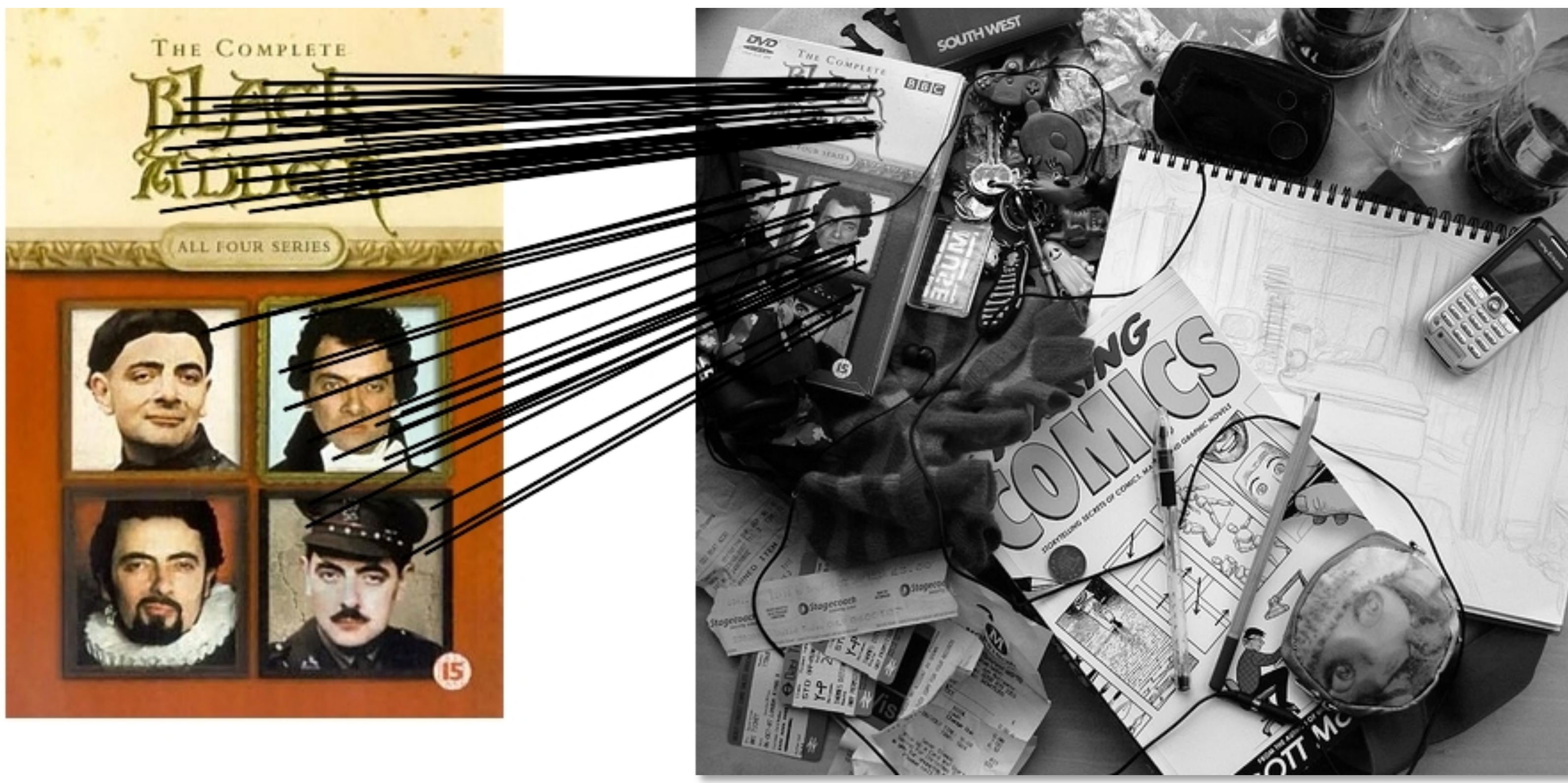


I_1



I_2

Feature matching example



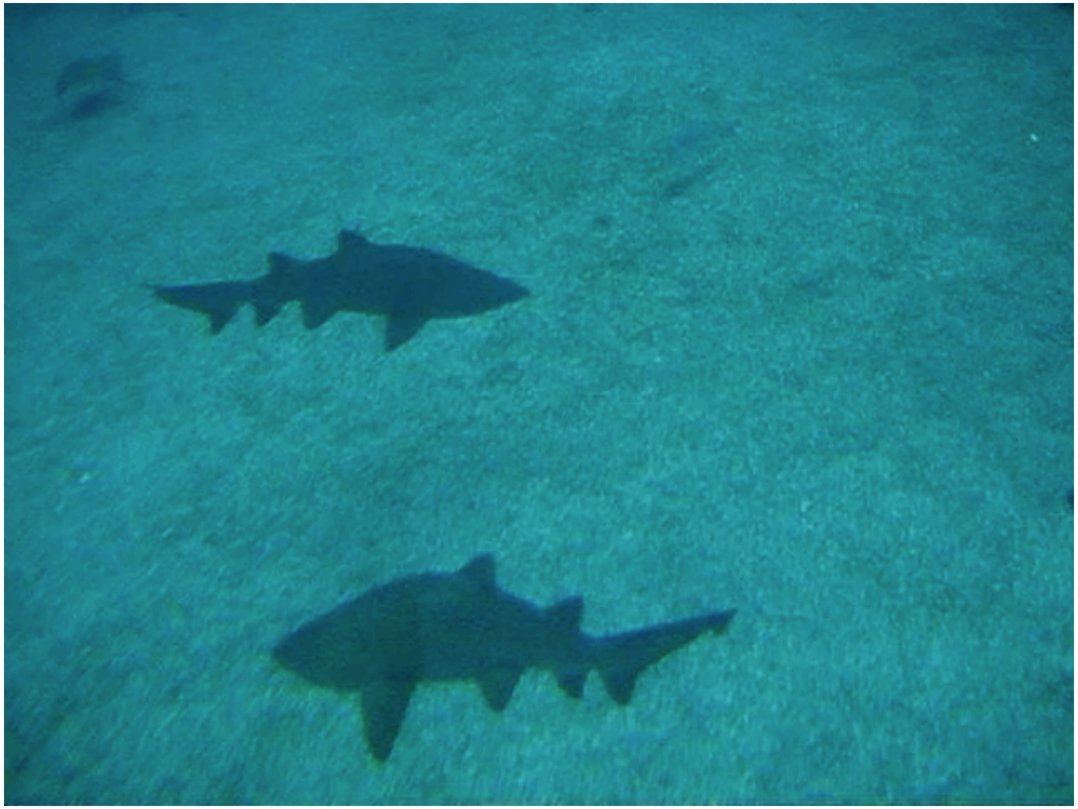
58 matches (thresholded by ratio score)

Motion Estimation

Slides adapted from Linda Shapiro

We live in a moving world

- Perceiving, understanding and predicting motion is an important part of our daily lives

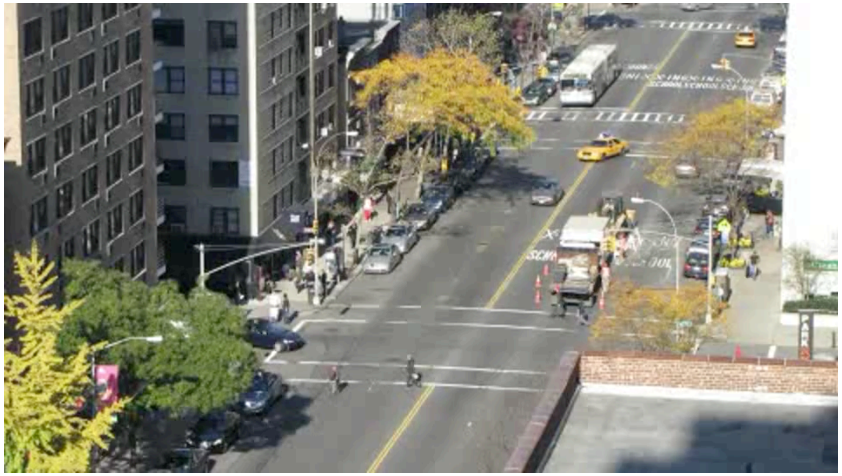


The cause of motion

Static camera, moving scene

Moving camera, static scene

Moving camera, moving scene



Static camera, moving scene, moving light

We still don't touch these areas

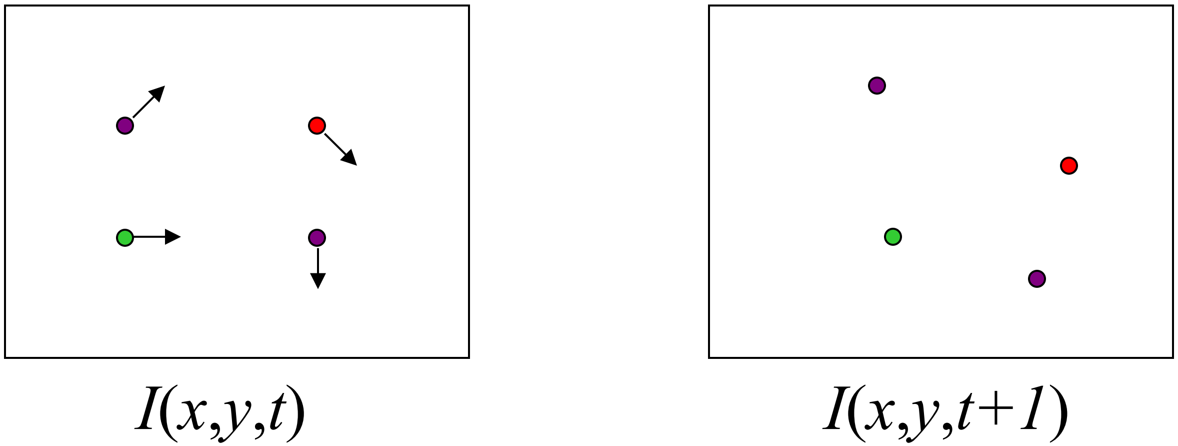
How can we recover motion?

- Feature-tracking
  - Extract feature (interest) points and "track" them over multiple frames
  - Output: displacement of sparse points
- Optical flow
  - Recover image motion at each pixel
  - Output: dense displacement field (optical flow)

Two problems, one registration method:
Lucas-Kanade method

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Feature tracking



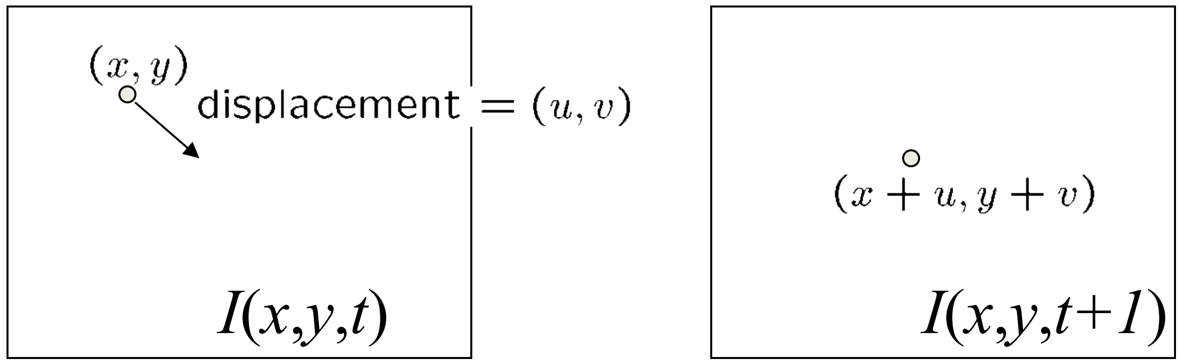
Given two subsequent frames, estimate the point translation

- What is the difference to feature matching?

Key assumptions of Lucas-Kanade Tracker

- **Small motion:** points do not move very far
- **Brightness constancy:** projection of the same point looks the same in every frame
- **Spatial coherence:** points move like their neighbors

The brightness constancy constraint



- Brightness Constancy Equation:

$$\boxed{I(x, y, t) = I(x+u, y+v, t+1)}$$

Take Taylor expansion of $I(x+u,\ y+v,\ t+1)$ at (x,y,t) to linearize the right side:

Image derivative along x   Difference over frames

$$\boxed{I(x+u, y+v, t+1) \approx I(x, y, t) + \boxed{I_x} \cdot u + I_y \cdot v + \boxed{I_t}}$$

$$I(x+u, y+v, t+1) - I(x, y, t) = +I_x \cdot u + I_y \cdot v + I_t$$

So: $$I_x \cdot u + I_y \cdot v + I_t \approx 0 \qquad \rightarrow \nabla \mathrm{I} \cdot [\mathrm{u} \ \ \mathrm{v}]^{\mathrm{T}} + \mathrm{I_t} = 0$$

The brightness constancy constraint

Can we use this equation to recover image motion (u,v) at each pixel?

$$\nabla \mathrm{I} \cdot [\mathrm{u} \;\; \mathrm{v}]^{\mathrm{T}} + \mathrm{I}_{\mathrm{t}} = 0$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns (u,v)

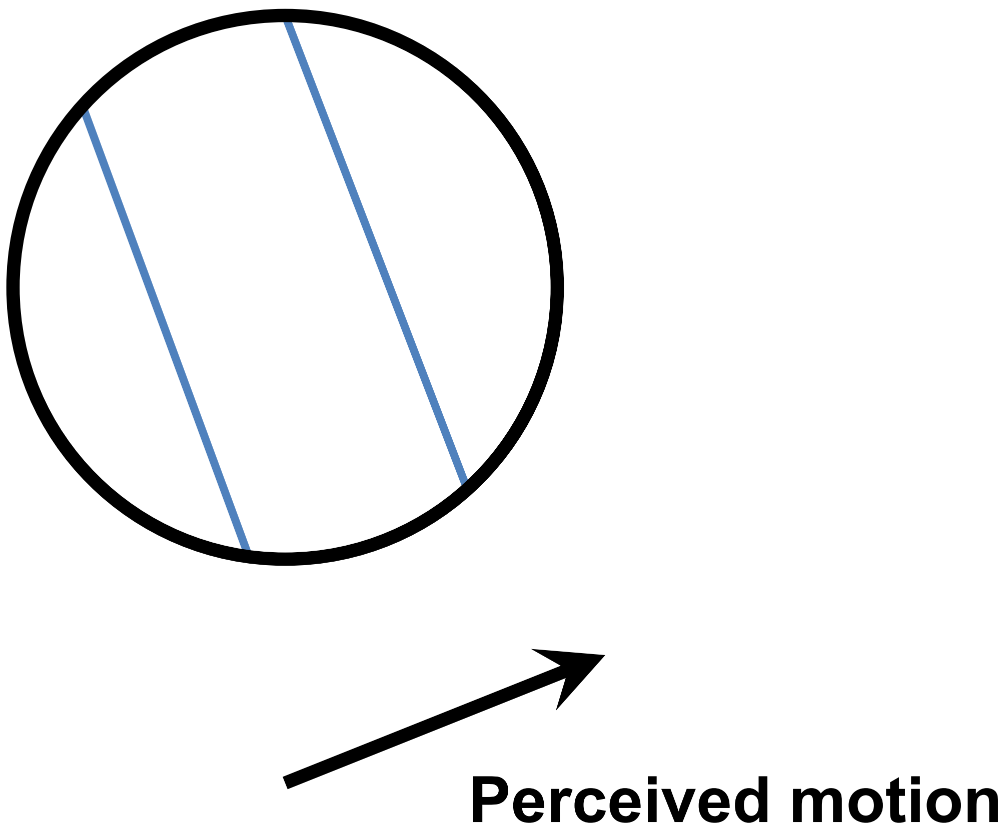
- If (*u*, *v*) satisfies the equation, so does (*u+u'*, *v+v'*) if

$$\nabla \mathrm{I} \cdot [\mathrm{u'} \;\; \mathrm{v'}]^{\mathrm{T}} = 0$$

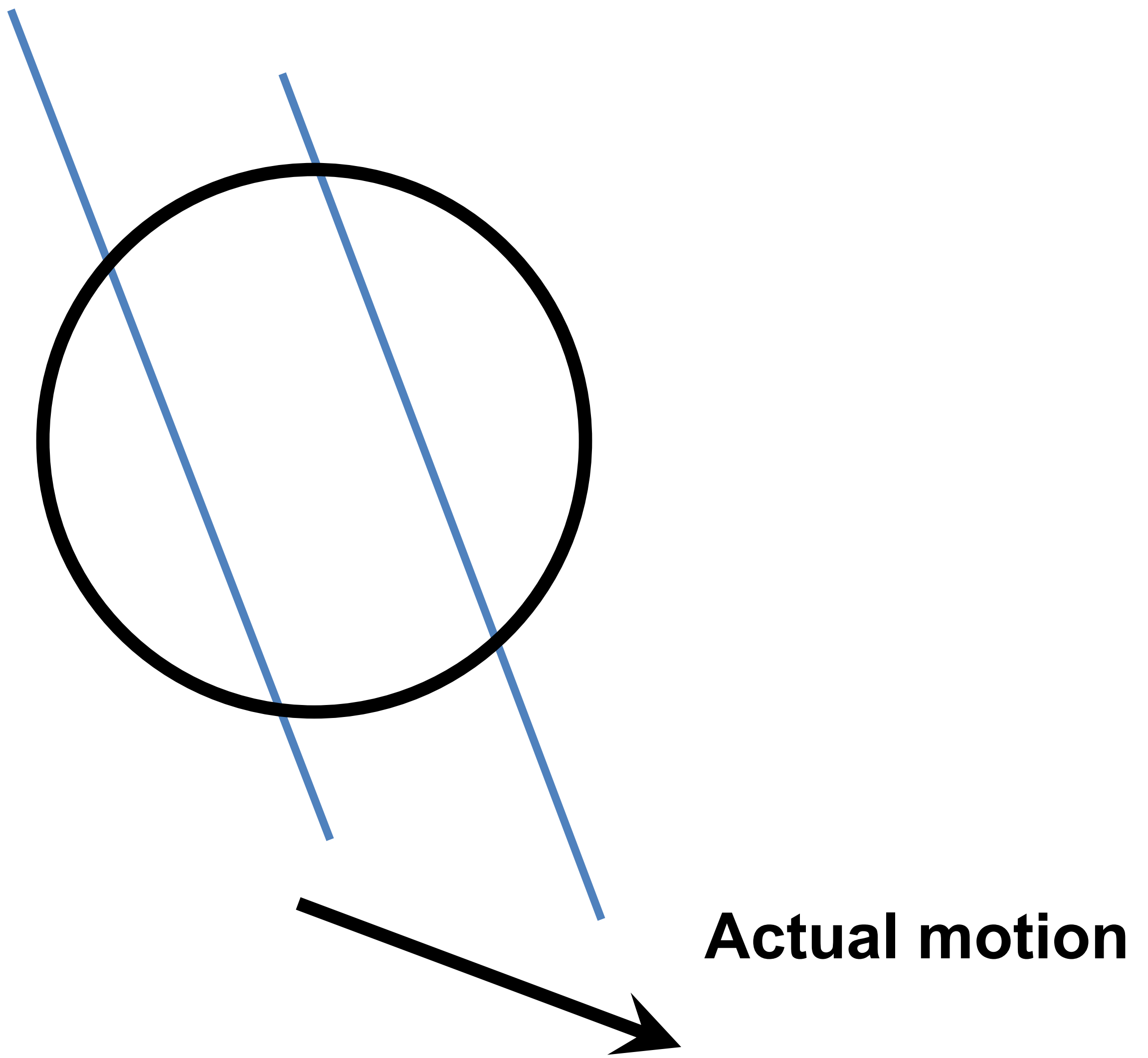
Which means the component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

The aperture problem

孔径问题



The aperture problem



The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Solving the ambiguity…

- Idea: get more equations
- **Spatial coherence constraint**
- Assume the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \;\; v]$$

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Matching patches across images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Least squares solution for *d* given by $\boxed{(A^T A)\ d = A^T b}$

$$\overbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}^{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \overbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}^{A^T b}$$

The summations are over all pixels in the K x K window

Conditions for solvability

Optimal (u, v) satisfies Lucas-Kanade equation

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

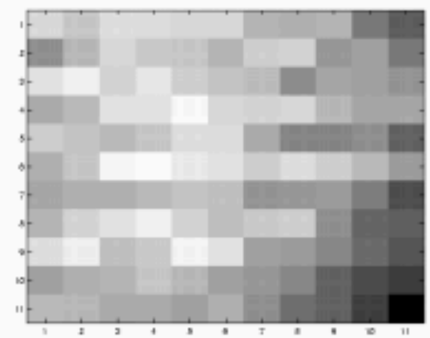
When is this solvable? I.e., what are good points to track?

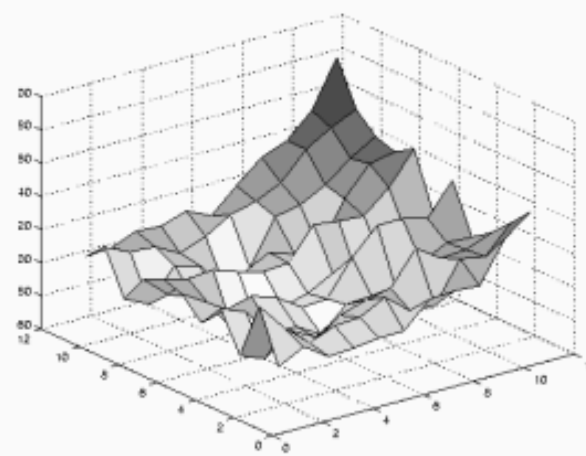
- **A^{T}A** should be invertible and well-conditioned
  - eigenvalues λ_1 and λ_2 of **A^{T}A** should not be too small
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

Low Texture Region



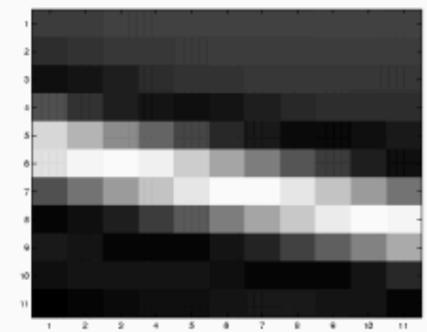


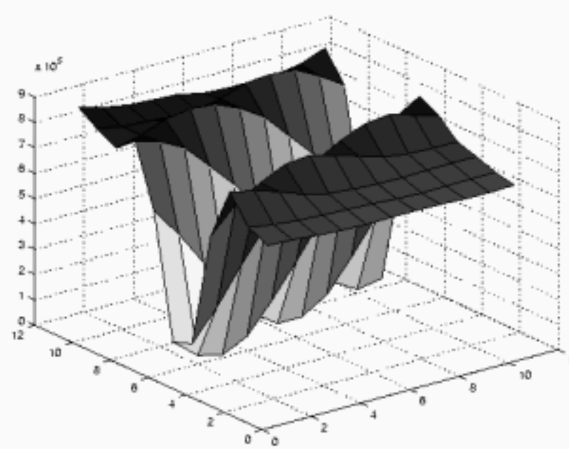
$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

Edge



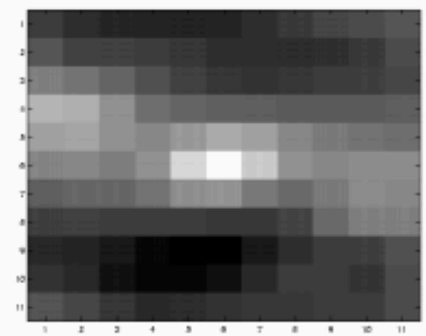


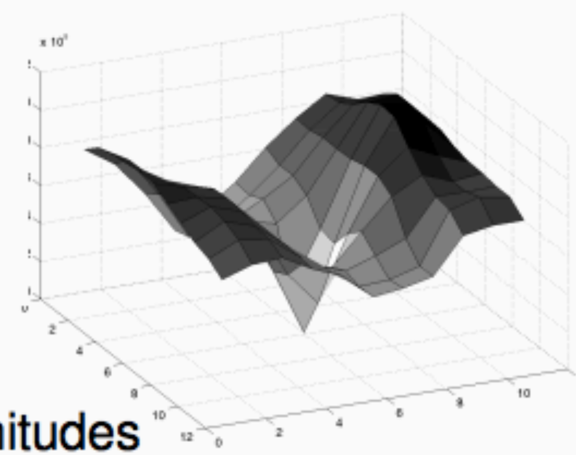
$$\sum \nabla I (\nabla I)^T$$

– large gradients, all the same

– large λ_1, small λ_2

High Texture Region





$$\sum \nabla I (\nabla I)^T$$

– gradients are different, large magnitudes

– large λ_1, large λ_2

Errors in Lukas-Kanade

- What are the potential causes of errors in this procedure?
  - Suppose A^TA is easily invertible
  - Suppose there is not much noise in the image

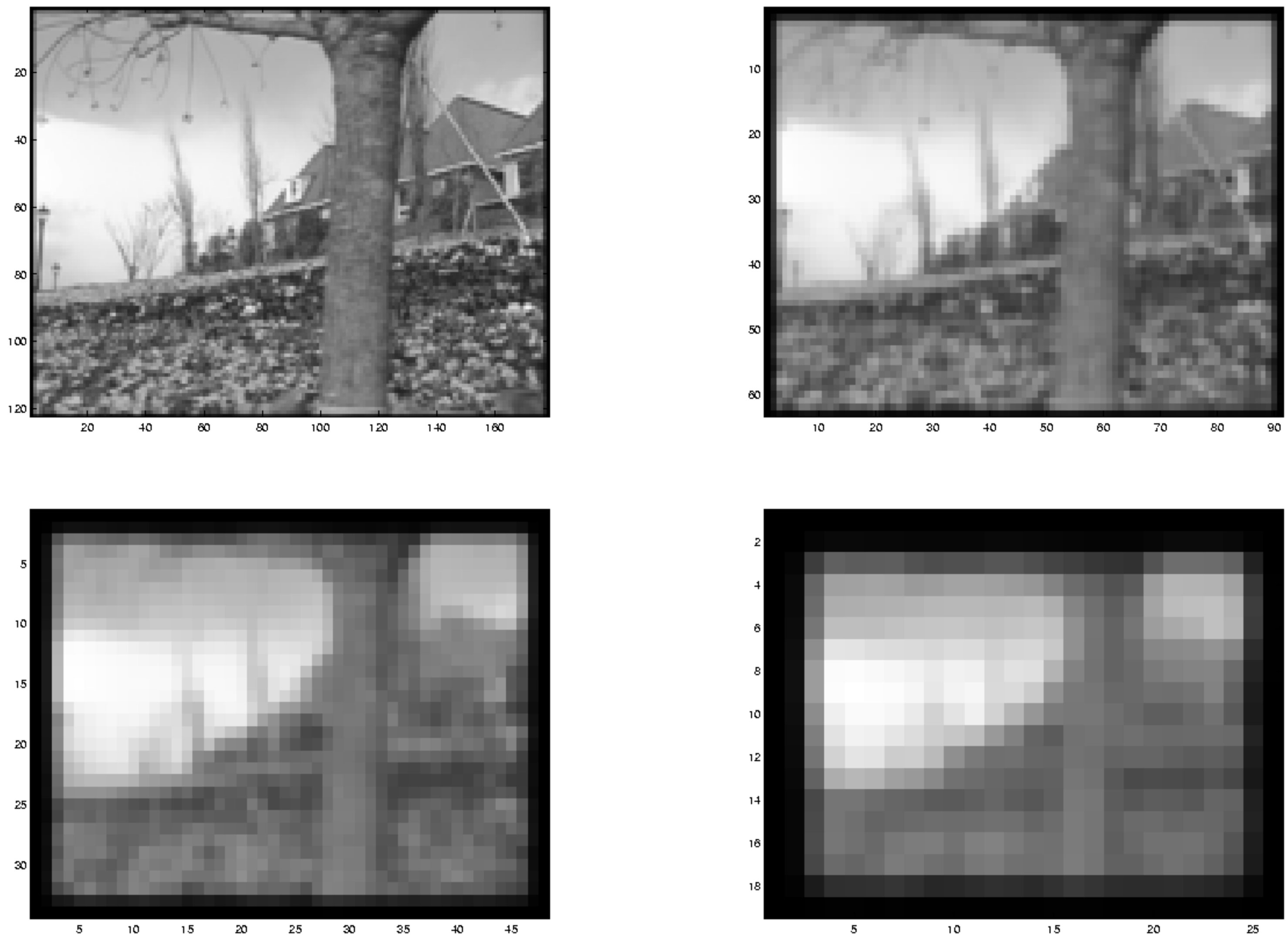
When our assumptions are violated

- Brightness constancy is **not** satisfied
- The motion is **not** small
- A point does **not** move like its neighbors
  - window size is too large
  - what is the ideal window size?

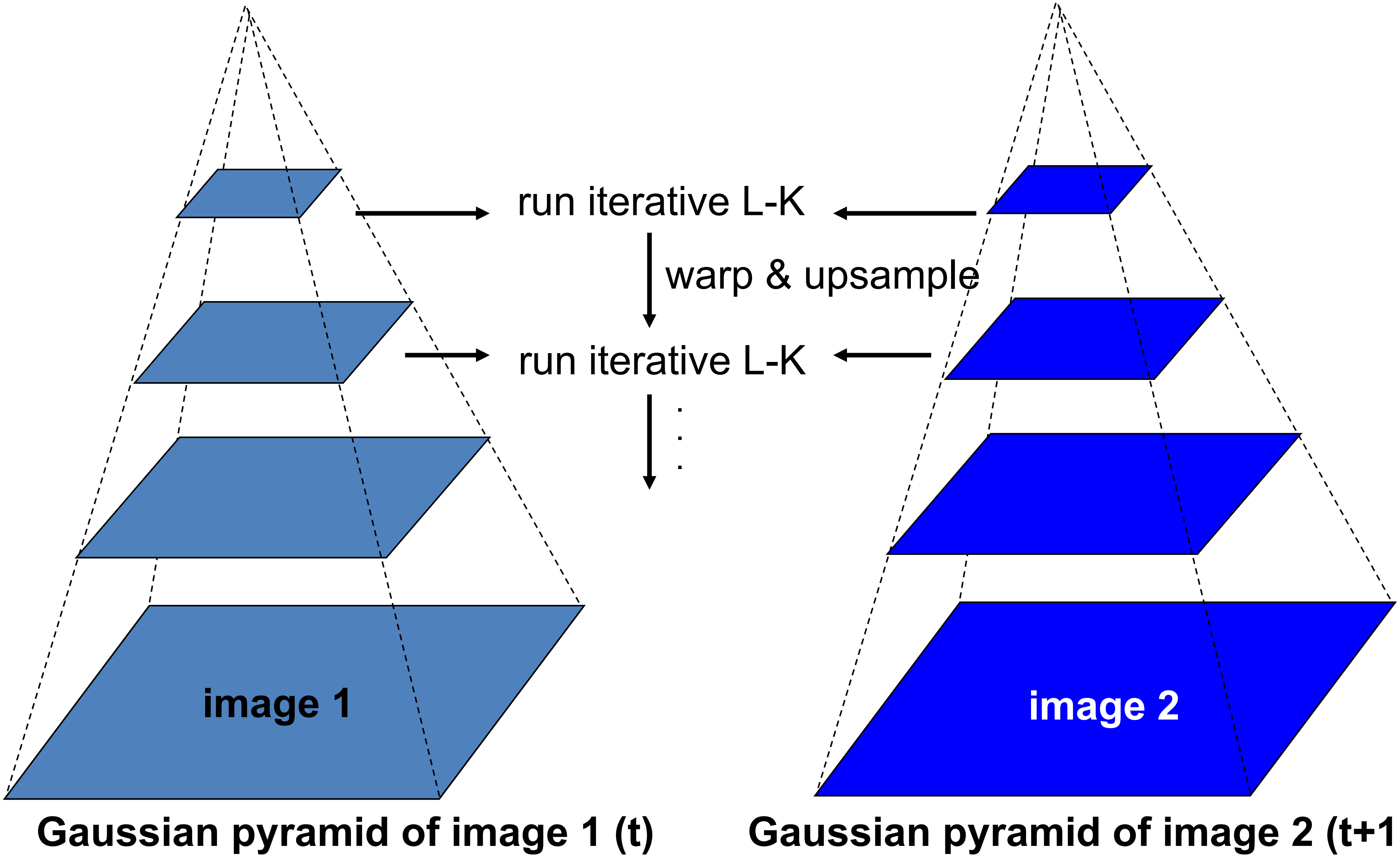
Revisiting the small motion assumption

- Is this motion small enough?
  - Probably not—it's much larger than one pixel
  - How might we solve this problem?

Reduce the resolution!

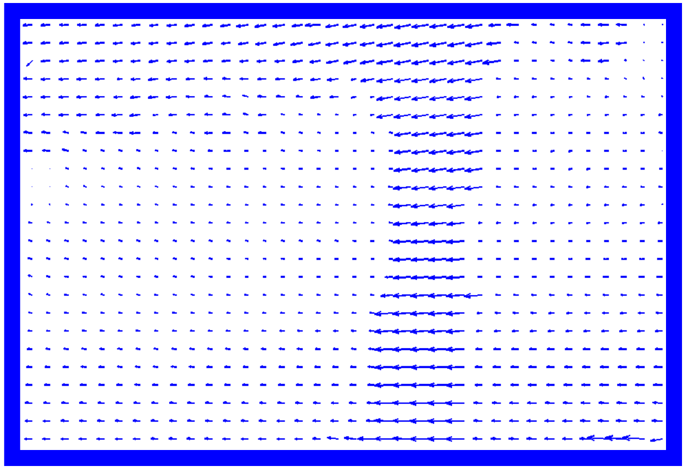


Coarse-to-fine optical flow estimation

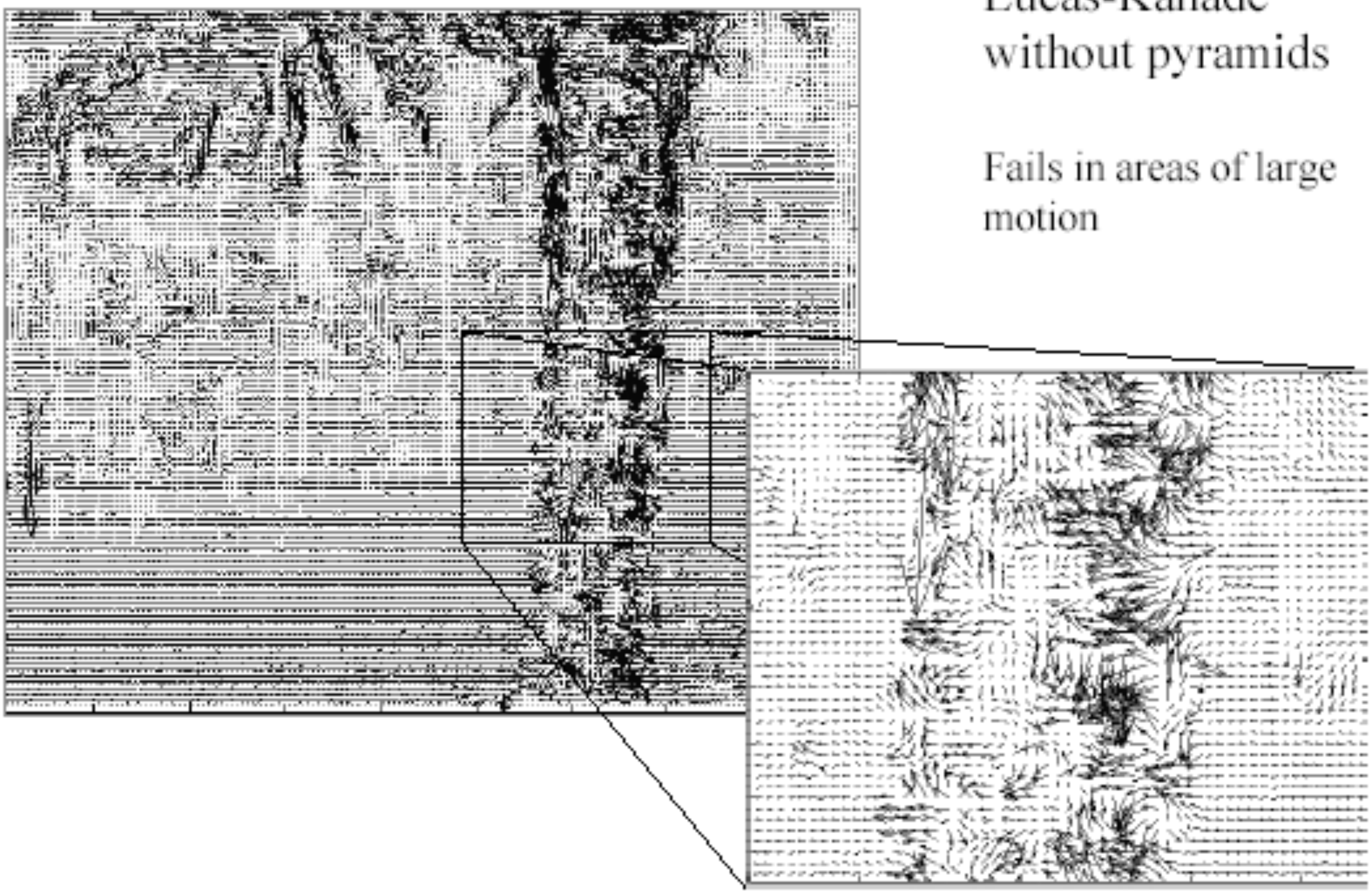


The Flower Garden Video

What should the
optical flow be?



Optical Flow Results

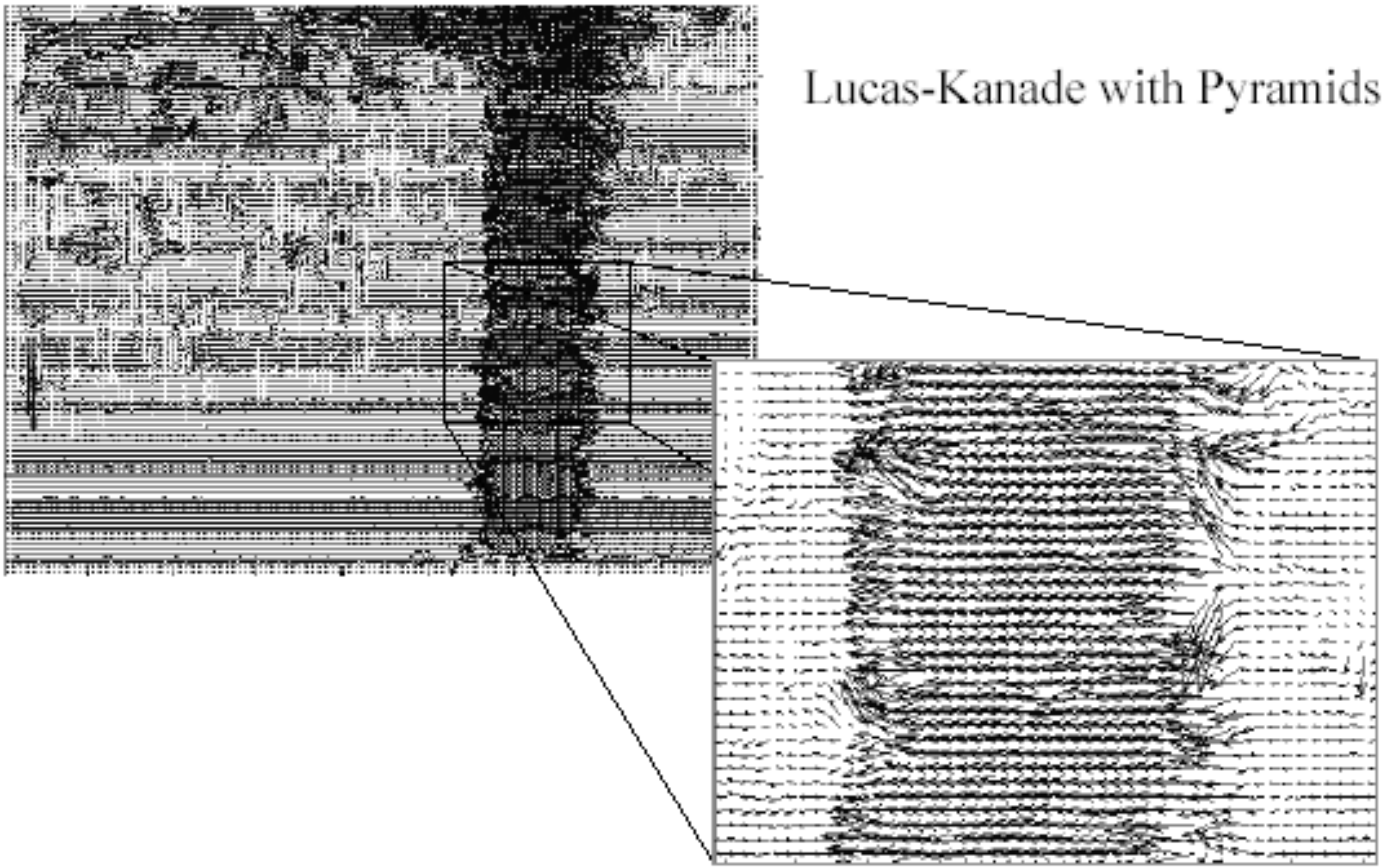


Lucas-Kanade
without pyramids

Fails in areas of large
motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



Lucas-Kanade with Pyramids

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Video stabilization

Video denoising

Things to remember

- Feature matching
  - Detector: Harris corner detector, LoG/DoG
  - Descriptor: SIFT …
  - Matching: ratio test
  - Invariance

- Motion estimation
  - Feature tracking
  - Optical flow
  - Lucas-Kanade
    - Three assumptions

- Both feature matching and motion estimation are called correspondence problem (对应关系问题)

Questions?