Local Features I: Keypoints

CS 4243 Computer Vision & Pattern Recognition

Angela Yao

Recap & Outline

Last Lecture

- Visual textures: property indicative of material and appearance
- Filter banks as feature representations
- Representing texture with textons and histograms of textons
- Perceived contours: texture plays a big role

Today's Lecture

- Keypoints: locations in images for computing descriptors and matching
- Corners & Harris operator
- Equivariance & invariance
- Automatic scale selection
- Laplacian of Gaussian (LoG) operator

Before Panorama Apps on Smartphones…

San Francisco - 35mm film panorama from year 2000
https://www.flickr.com/photos/parkstreetparrot/27691676929/

View from Lookout Mountain, Tennessee (1864)
https://en.wikipedia.org/wiki/Panoramic_photography

How can we combine two images?

1. Match parts which are the same on both images
2. Align images based on the matches

Matching is easy. (sort of)

Trevi Fountain in Rome

What about these images?

NASA Mars Rover images with SIFT feature matches

How can we combine two images?

1. Match parts which are the same on both images
   a. Find locations to match.
   b. Represent surrounding region mathematically.
   c. Do the matching.

Interest point **detection**.
Compute feature **descriptors**.
Feature **matching**.

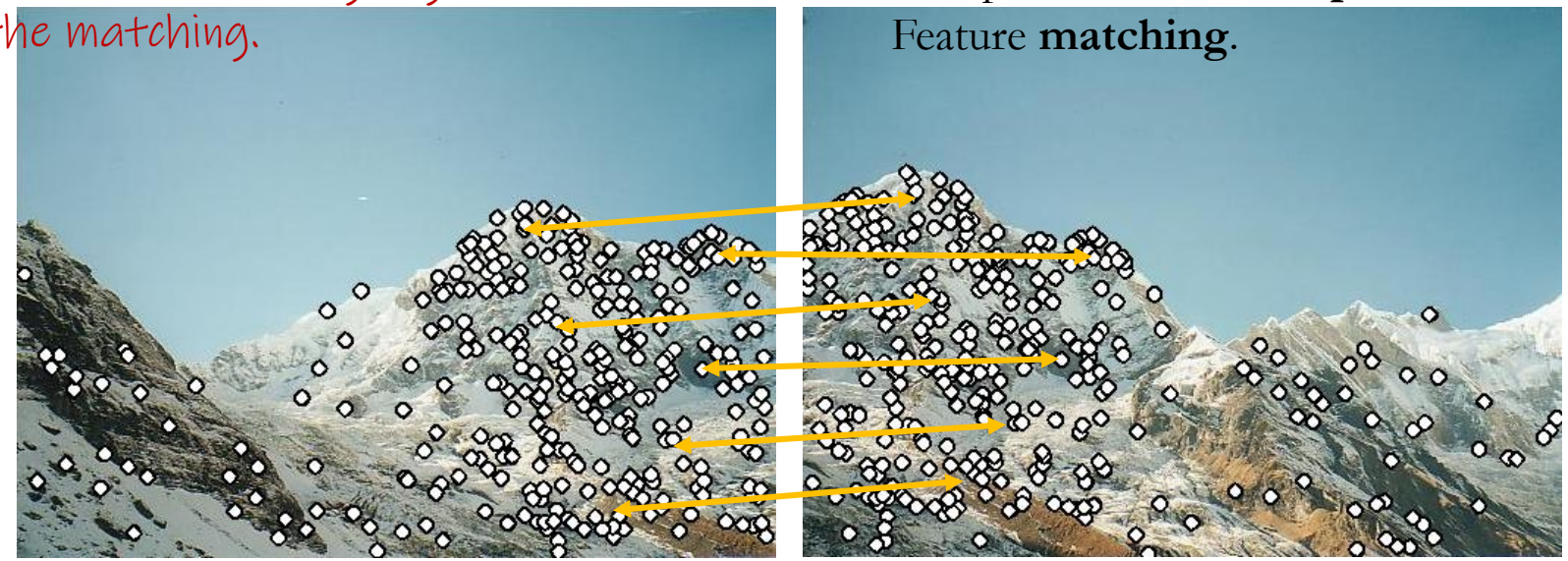
Loaded terminology:
aka keypoint detection; or simply feature detection

We will focus on a specific type of keypoint or interest point called Harris corners.

"Features"
...
Often refers to both detection and representation, i.e. both the keypoint and the descriptor.

today's lecture

next week



Isn't this easy? Just look for stuff that's the same right?

Make a filter or something?

Matching Two Images

Matching Two Images

1) Detection: Identify interest points

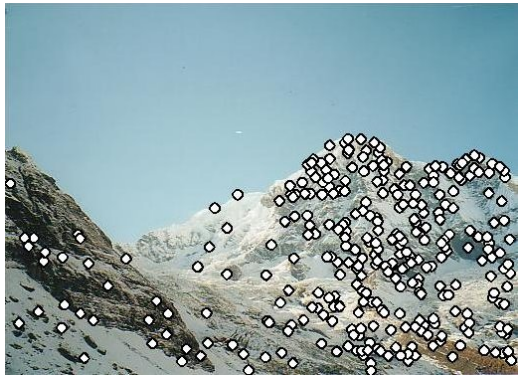
Where should these come from? How to design these so that they will be found in both images?

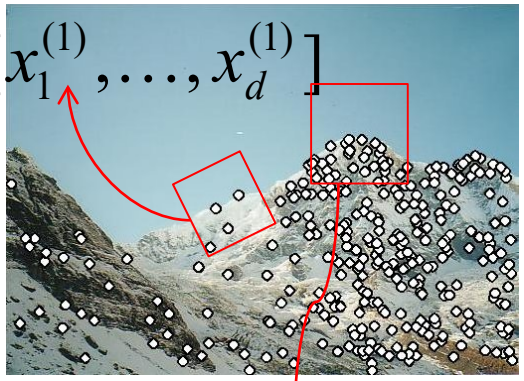
2) Description: Compute vector feature descriptor surrounding each interest point.

How should we characterize a region mathematically? How can we do build in (enough but not too much) uniqueness to facilitate matching?

3) Matching: Determine correspondence between descriptors in two views

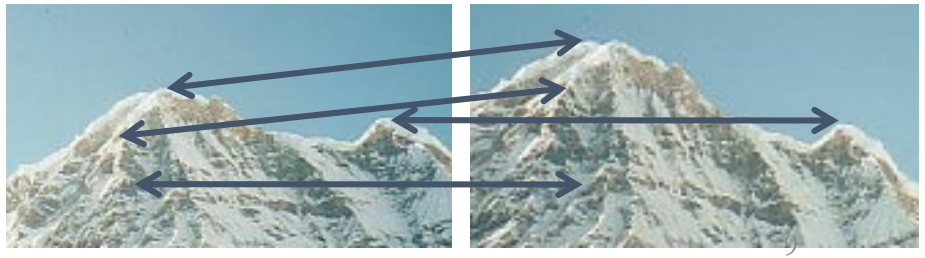
How can we establish correspondence? Efficiency?





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

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



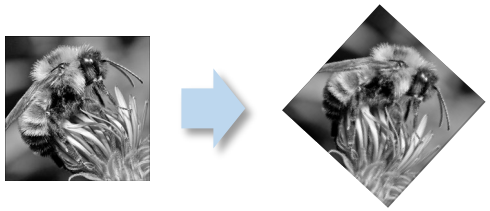
Adapted from K. Grauman

Characteristics of Good Local Features

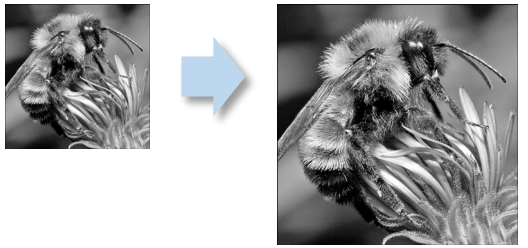
- Repeatable interest points
  - The same point can be found in several images despite geometric and photometric transformations
- Distinct descriptors
  - Captures recognizably different information that allows it to be distinguished
- Efficient
  - Quantity: (number of features << number of image pixels); compact
- Local
  - A feature occupies a relatively small area of the image (therefore robust to clutter and occlusion).

Geometric transformations

Rotation

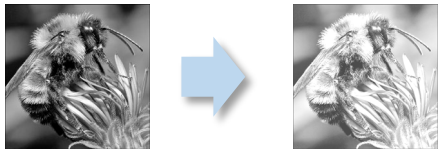


Scale



Photometric transformations

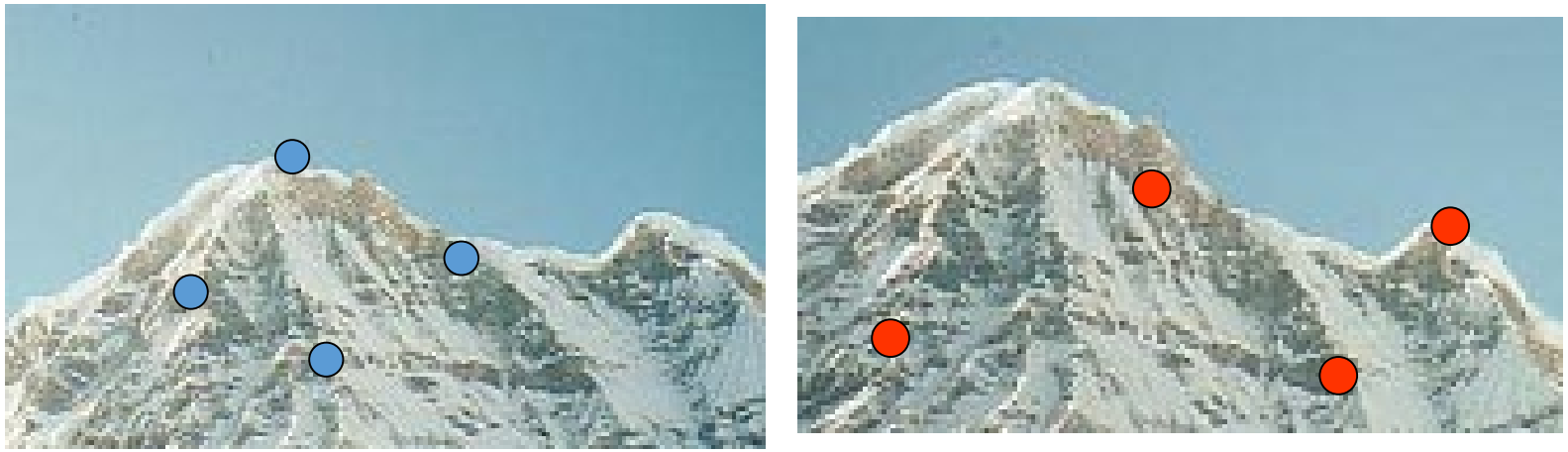
Intensity change



Adapted from K. Grauman

Goal: Repeatability of Interest Point Detector

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



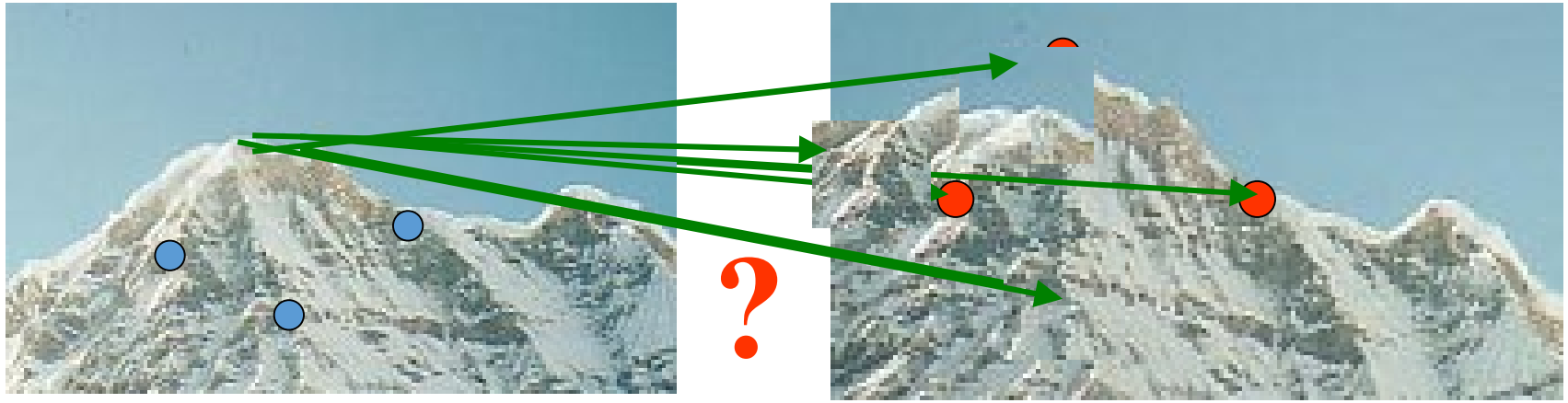
If the points are not the same, there is no chance to find true matches!

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

Adapted from K. Grauman

Goal: Descriptor Distinctiveness

We want to reliably determine which point goes with which.



Descriptors must provide some invariance to geometric and photometric differences between the two views.

Adapted from K. Grauman

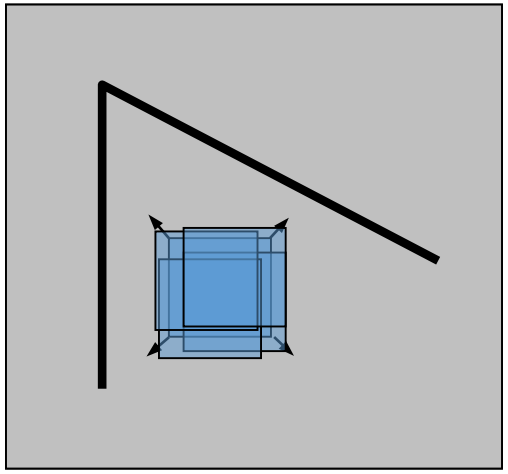
Goal: Descriptor Distinctiveness

How to define?

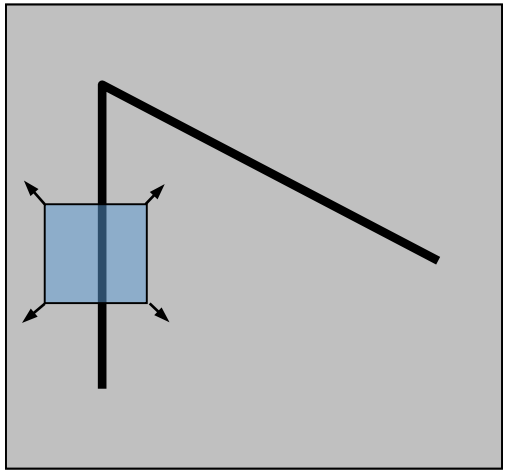
Look for unusual regions → leads to unambiguous matches in other images

Start by considering a small region: What makes a good candidate?

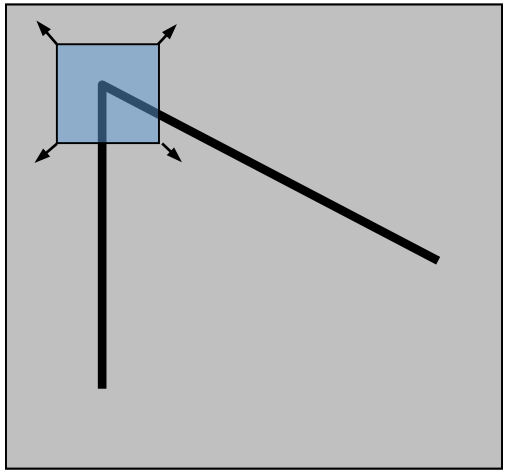
Define a window at the region. How will this window change when you shift it? At unique regions, shifting the window in any direction causes a big change.



"flat" region
no change in all directions



"edge"
no change along the edge direction



"corner"
significant change in all directions

rationale and naming behind Harris corners

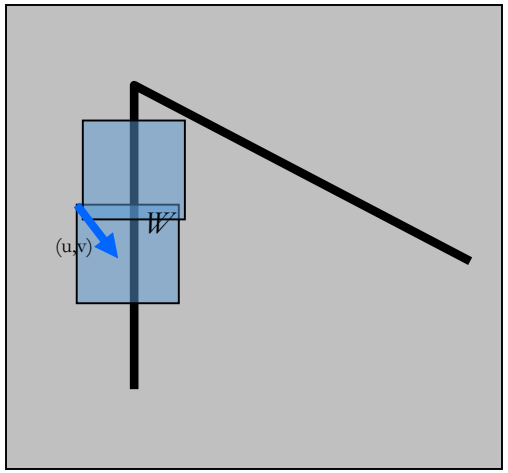
Adapted from: S. Seitz, D. Frolova, D. Simakov

The Math Behind Corner Detection

The Math Behind Corner Detection

Consider shifting the window W by offset (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" $E(u,v)$:



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

- We are happy if this error is high.

Now just search for high error windows all over the image to find the corners. Efficiency issues?

HWQ: How does this scale computationally? Consider for a window of mxm which we shift in by m offsets in both x and y for an image of wxh.

Adapted from: S. Seitz, D. Frolova, D. Simakov

Assumption: consider only small shifts

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

Taylor Series expansion

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

For small shifts, i.e. (u,v) are small, then 1st order approx. is good enough.

$$\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

Image gradients in x- & y-direction!

notation convenience

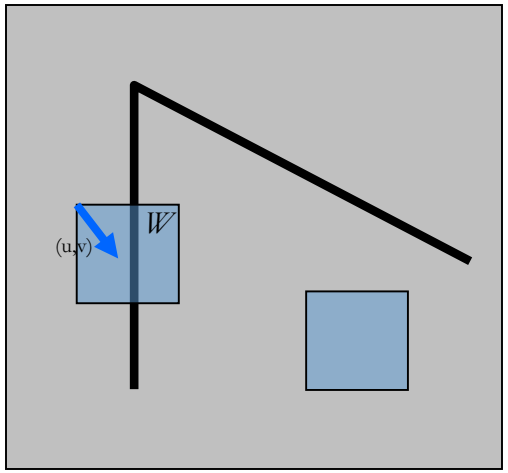
$$\frac{\partial I}{\partial x} = I_x$$

$$= I(x,y) + I_x u + I_y v$$

The Math Behind Corner Detection

Consider shifting the window W by offset (u,v)

- define an SSD "error" $E(u,v)$:



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

$$\approx \sum_{(x,y)\in W} [\cancel{I(x,y)} + I_x u + I_y v - \cancel{I(x,y)}]^2$$

$$= \sum_{(x,y)\in W} [I_x u + I_y v]^2 = Au^2 + 2Buv + Cv^2$$

$$= \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

SSD error is locally approximated as a quadratic function!

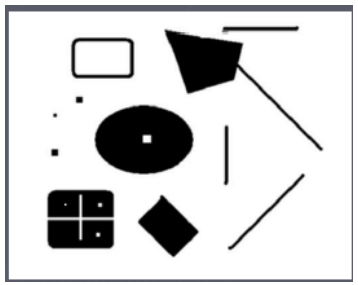
H Second moment matrix

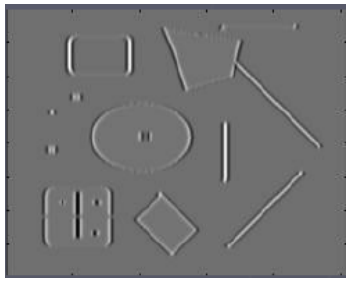
Adapted from: S. Seitz, D. Frolova, D. Simakov

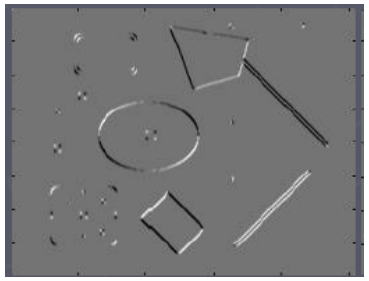
Second Moment Matrix H

$$\mathrm{H}= \begin{bmatrix} A & B \\ B & C \end{bmatrix} \qquad A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

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



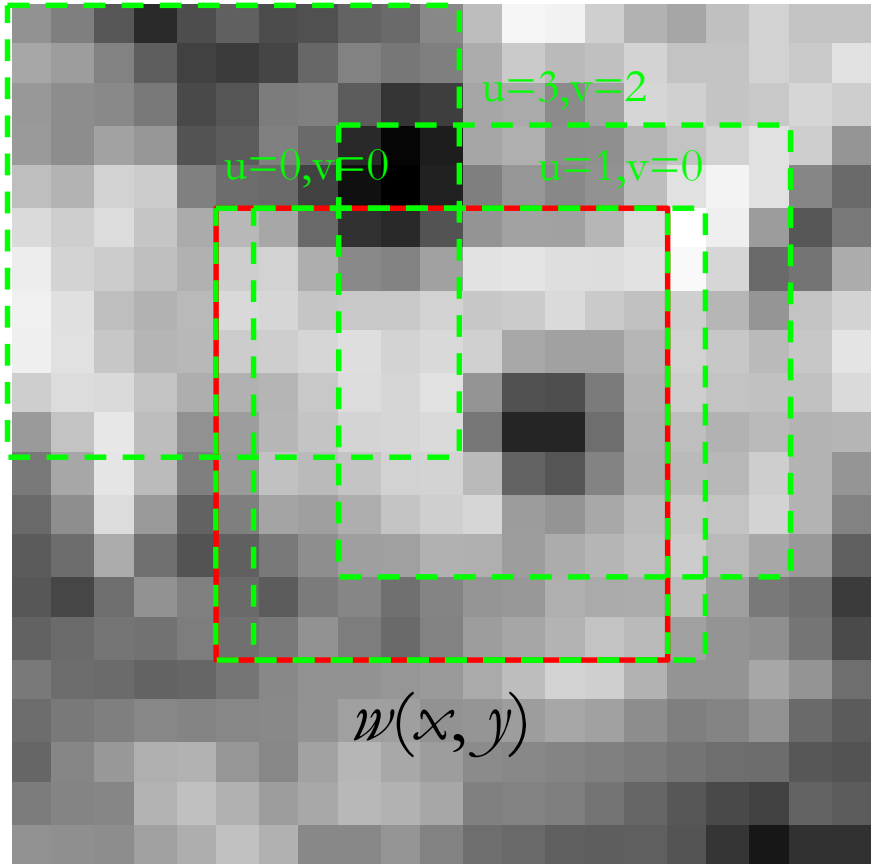




Notation:

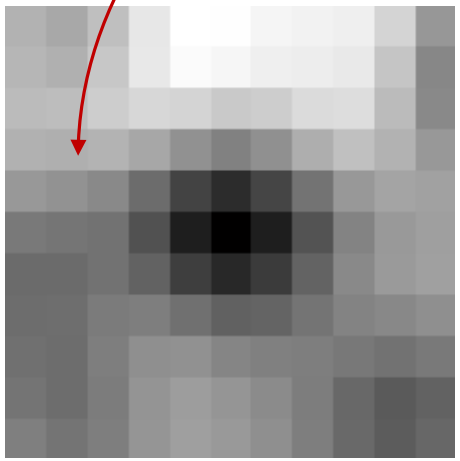
$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Visualizing the SSD Error



$I(x, y)$

quadratic in the error (intensity).



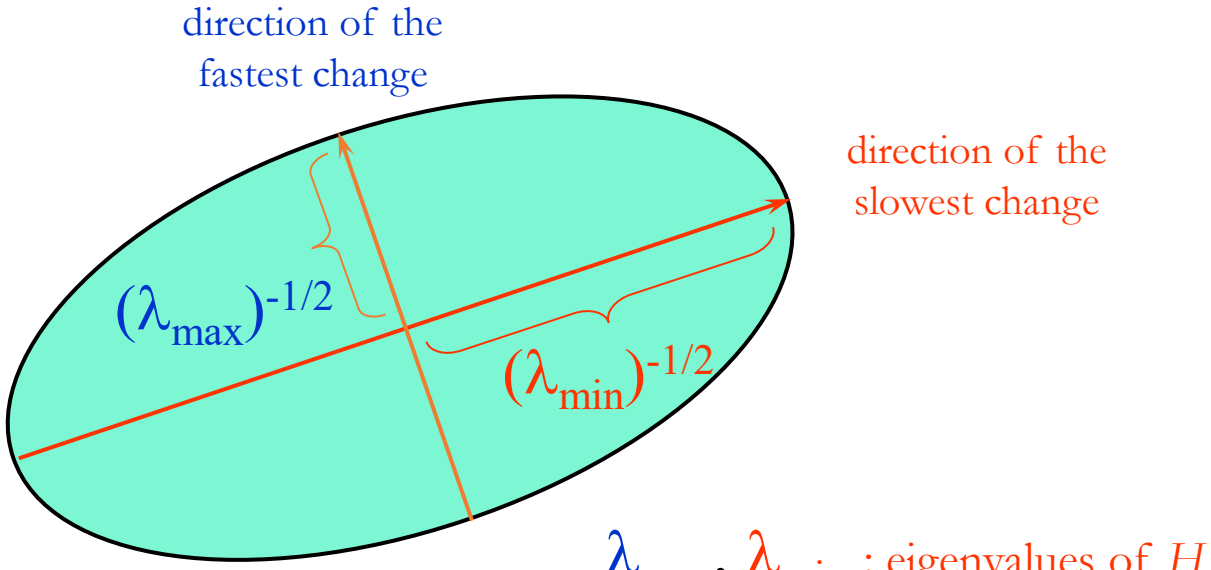
$E(u, v)$

Visualizing H

$$[u \;\; v] \; H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

Quadratic function can be visualized as an ellipse, where coefficients in H controls the shape of the ellipse.

Shape determined by looking at the **eigenvectors** and **eigenvalues**.



λ_{max}, λ_{min} : eigenvalues of H

Ellipse axes orientation determined by the *eigenvectors* of H

Ellipse axes length determined by the *eigenvalues* of H

Solve with SVD (Singular Value Decomposition)! In 2x2 case, directly.

Adapted from: S. Seitz, D. Frolova, D. Simakov

Eigendecomposition of 2x2 Matrices

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar λ is the **eigenvalue** corresponding to **x.**

$$det(A - \lambda I) = 0$$

The eigenvalues are found by solving:

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

For us, $\boldsymbol{A} = \boldsymbol{H}$ is a 2x2 matrix; we can directly solve for the eigenvalues:

$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

Once you know λ, you find **x** by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

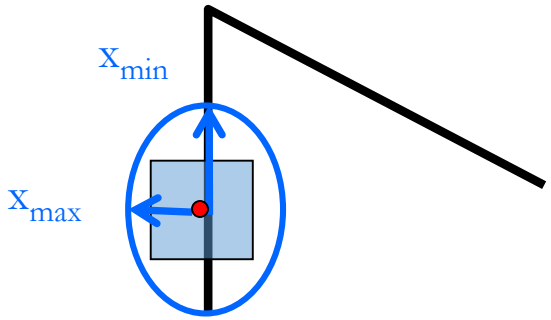
The Math Behind Corner Detection

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



$$Hx_{\max} = \lambda_{\max} x_{\max}$$

$$Hx_{\min} = \lambda_{\min} x_{\min}$$

Eigenvalues and eigenvectors of H define shift directions with the smallest and largest change in error

- x_{max} direction of largest increase in E
- λ_{max} related to amount of increase in direction x_{max}
- x_{min} direction of smallest increase in E
- λ_{min} related to amount of increase indirection x_{min}

If corners represent having large increase in all directions, this suggests that both λ_{max} and λ_{min} should be sufficiently large!

Adapted from: S. Seitz, D. Frolova, D. Simakov

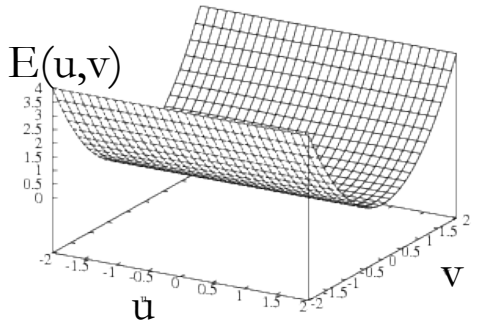
Special Cases

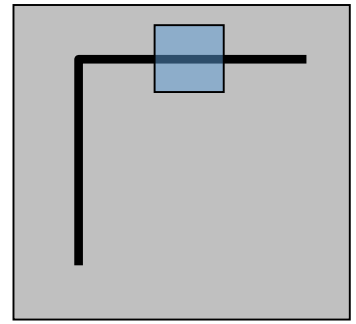
$$A = \sum_{(x,y)\in W} I_x^2 \quad B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \overbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}^{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

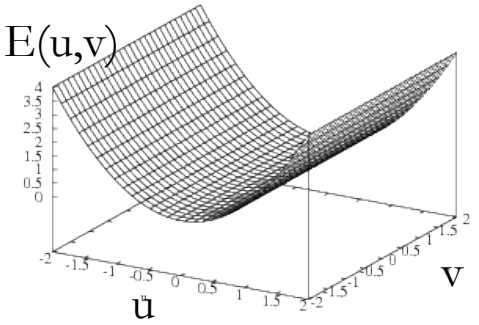
$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

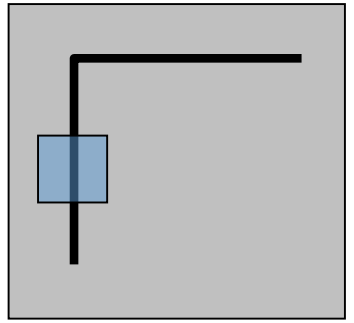




Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

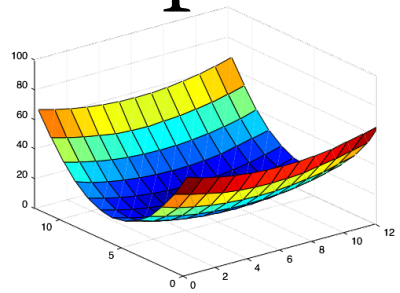


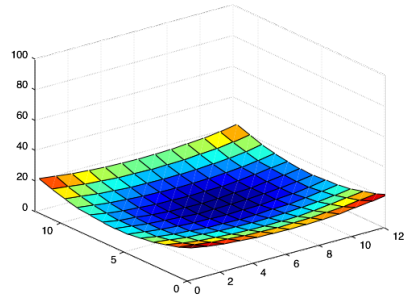


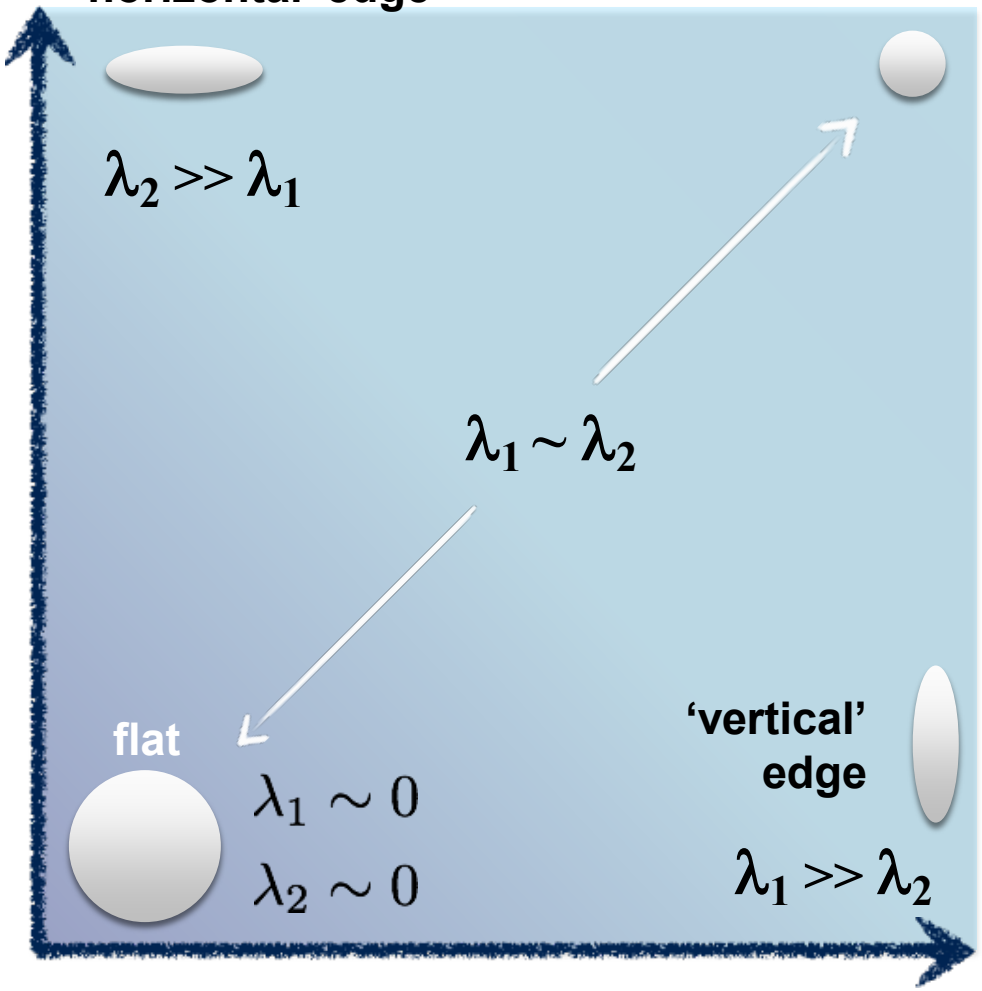
Vertical edge: $I_y = 0$

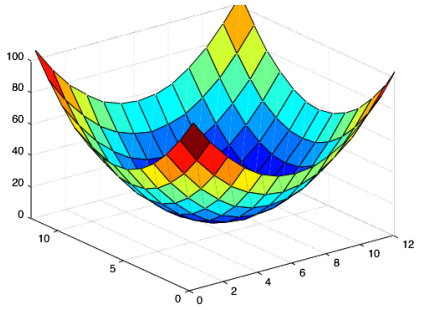
Adapted from: S. Seitz, D. Frolova, D. Simakov

Interpreting the Eigenvalues of H



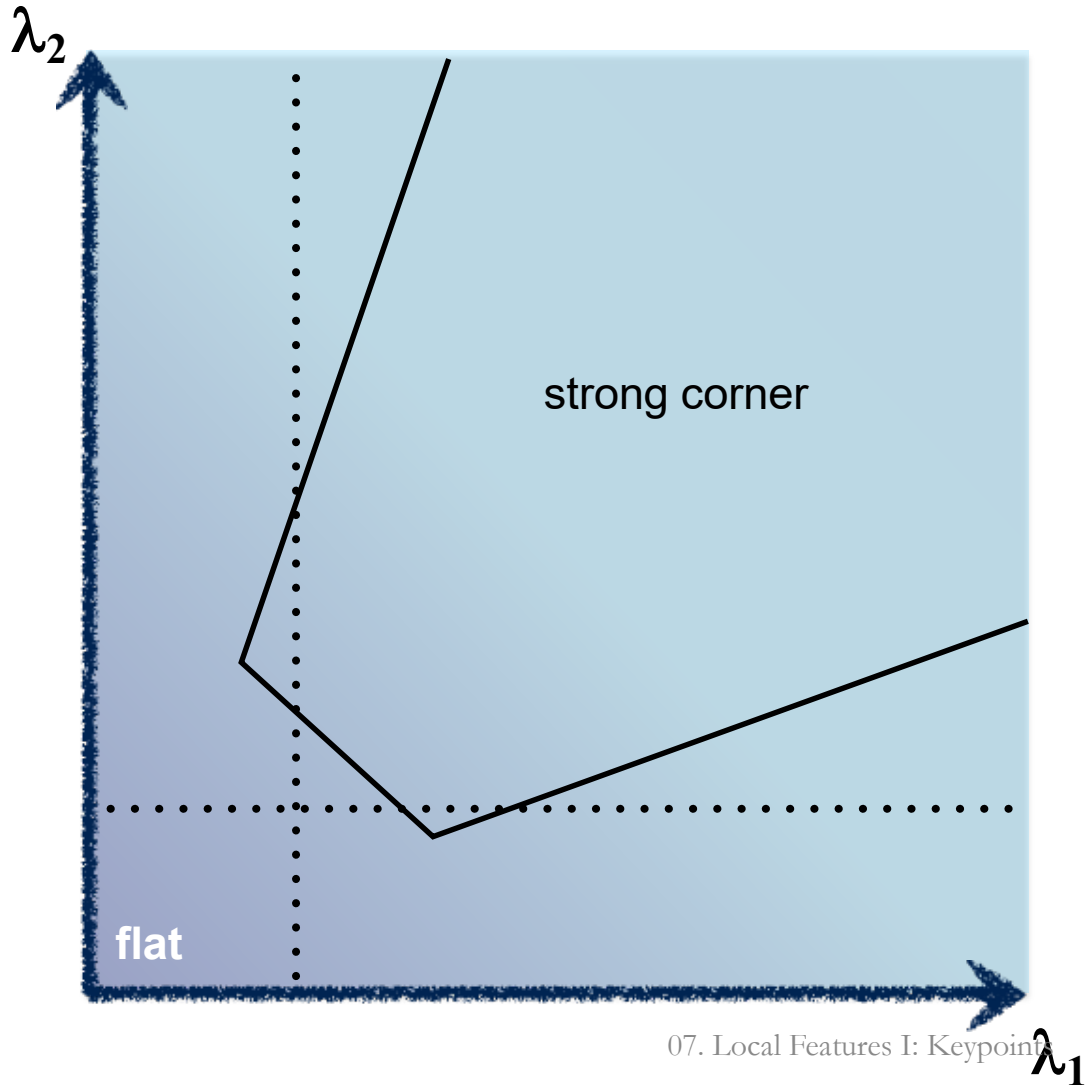






Slide Credit: K. Kitani.

Converting Eigenvalues to Corners



How can we develop a function to score 'cornerness'?

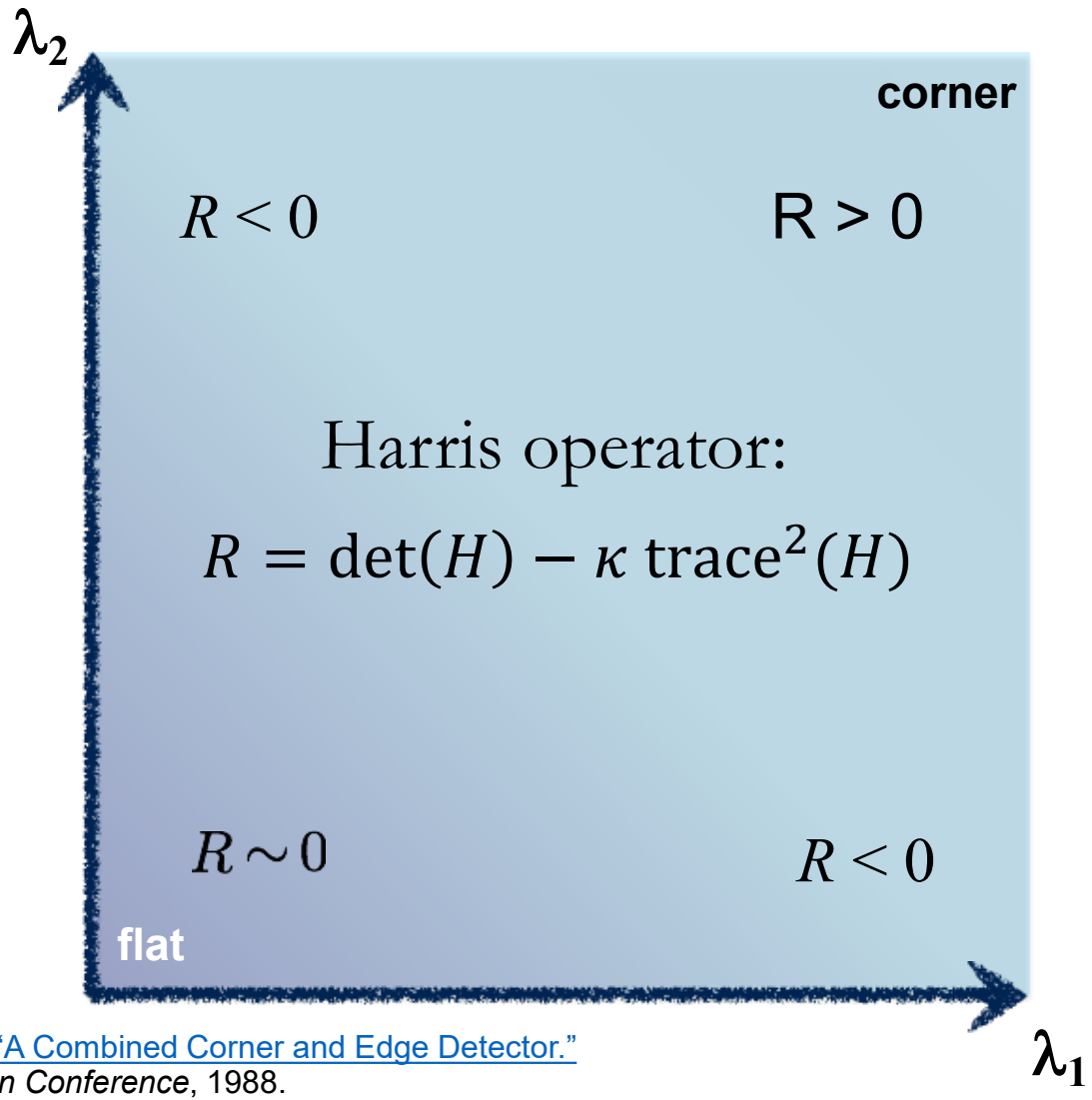
Use the smallest eigenvalue as the response function

$$R = \min(\lambda_1, \lambda_2)$$

Getting eigenvalues is slow (even in 2x2 scenario, because we need to take square roots).

Slide Credit: K. Kitani.

Converting Eigenvalues to Corners (more efficiently)



A more efficient way is

$$R = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

κ ~0.04 to 0.06

$$\det H = \lambda_1\lambda_2$$

$$\mathrm{trace}\ H = \lambda_1 + \lambda_2$$

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

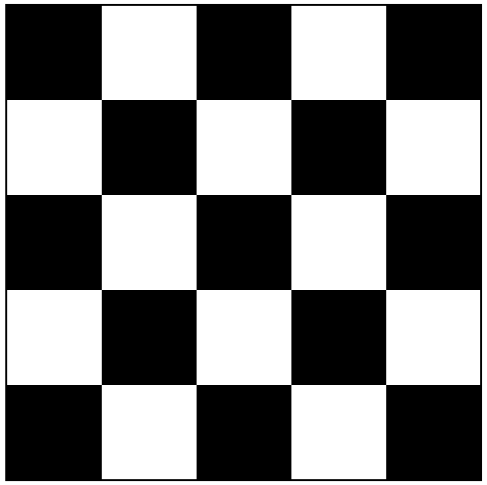
$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

Eigenvalues are always greater than 0 by definition since the H matrix is positive semi-definite.

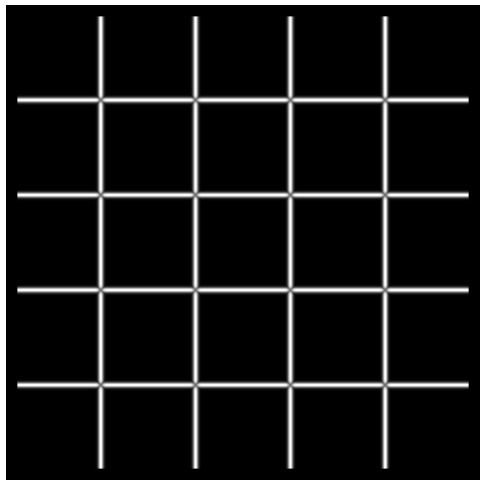
C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proc. of the 4th Alvey Vision Conference, 1988.

Slide Credit: K. Kitani.

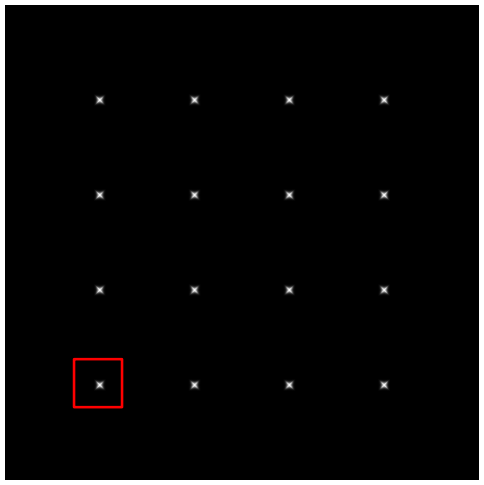
Comparison of Corner Criteria



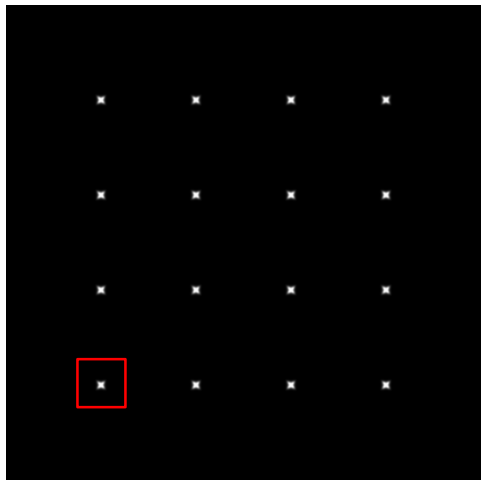
I



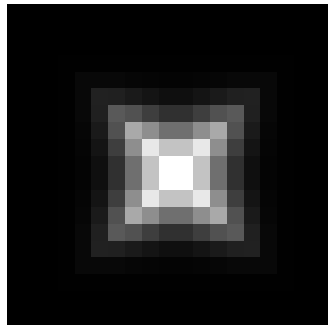
λ_{max}

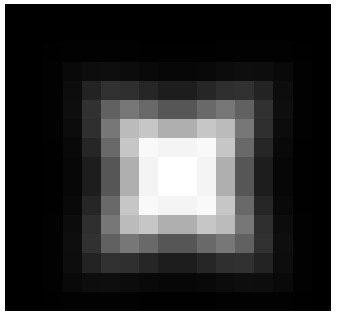


λ_{min}



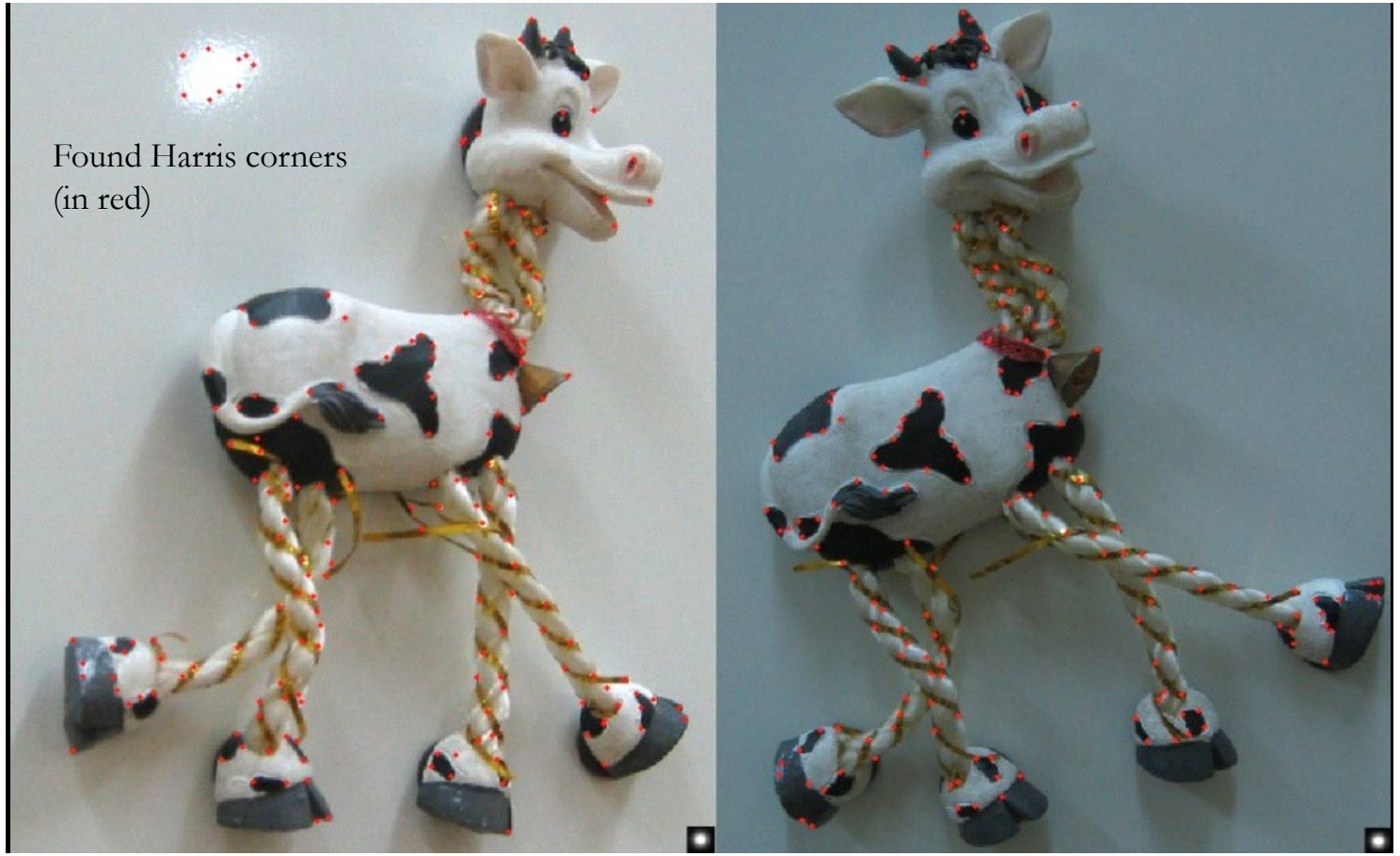
Harris operator



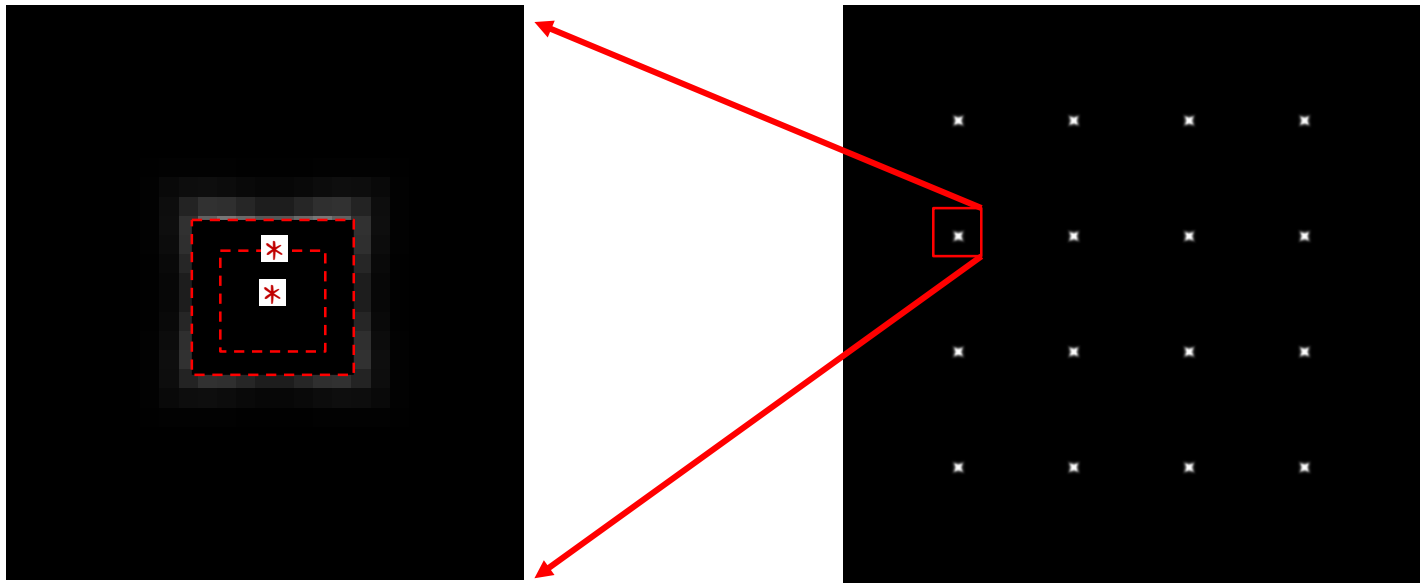


Summary of Harris Corner Detector

1) Compute gradient at each point in the image
2) Compute *H* matrix for each image window and get their *cornerness* scores.
3) Find points whose surrounding window gave large corner response (*R*> threshold)
4) Take the points of local maxima, i.e., perform non-maximum suppression



(Simple) Non-Maximum Suppression



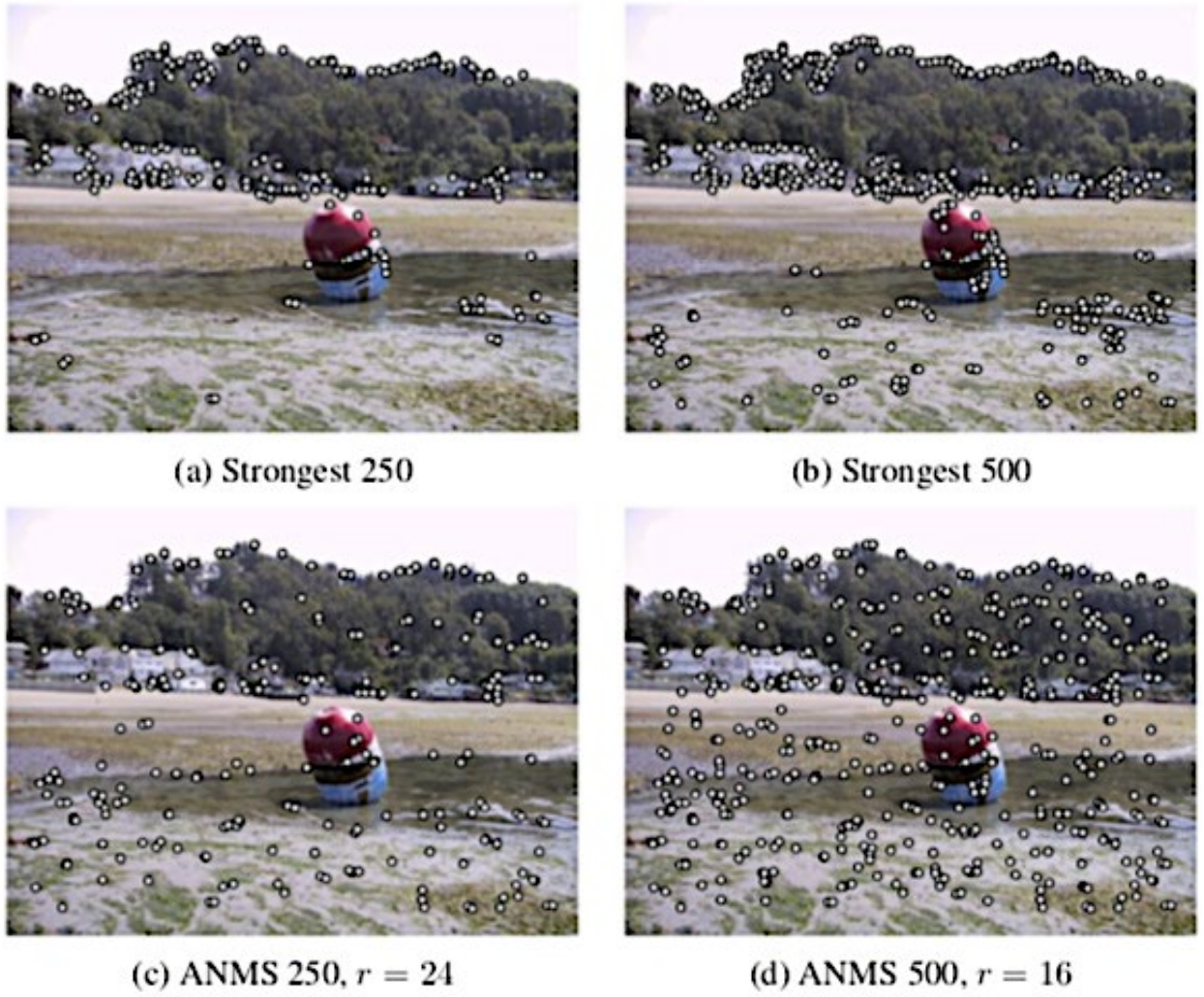
Harris operator

Speed-ups possible to make this more efficient.

Simplest approach: iteratively search for "max" values, then zero-out everything in the surrounding window.

Adaptive Non-Maximum Suppression

- Simple search of maxima leads to an uneven distribution of interest points in areas of higher contrast
- Pick corners which are both local maxima and whose response is significantly greater (e.g. 10%) than all neighbouring local maxima within some radius *r*



(a) Strongest 250 (b) Strongest 500

(c) ANMS 250, $r = 24$ (d) ANMS 500, $r = 16$

Fig. from Szeliski text.

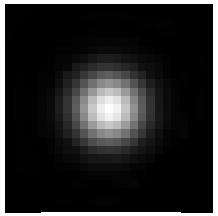
Weighting the Derivatives

In practice, using a simple window W (e.g. 3x3, 5x5) doesn't work too well

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Instead, we *weight* derivatives based on its distance from the center pixel

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



$w_{x,y}$

Harris Corner Invariances

Invariance and equivariance

A criterion of corner detection: repeatability.

Suppose the image is transformed in some way

- translation
- rotation
- intensity scaling

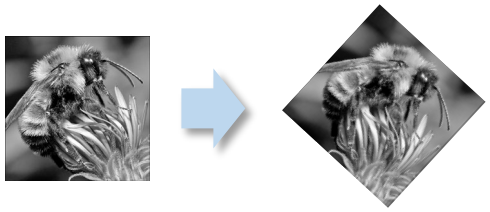
How *should* the corners change?

- Whether a corner is detected or not?
- Location?

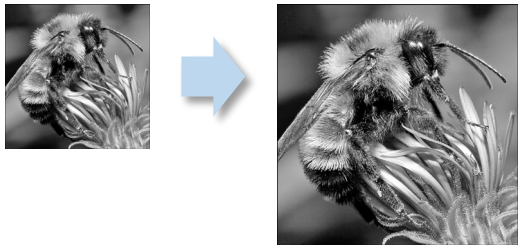
How *does* the output of Harris corner detector change?

Geometric transformations

Rotation

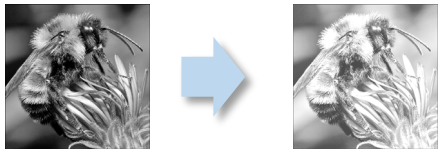


Scale



Photometric transformations

Intensity change



Adapted from K. Grauman

Invariance and Equivariance

Equivariance: if we have two transformed versions of the same image, detection (locations) undergoes a similar transformation

Invariance: image is transformed and detection (score) does not change

"Detection" here can refer either to the corner location as well as the probability of being detected as a corner. Can think of equivariance wrt location and invariance wrt score.

Sometimes invariance and equivariance are both referred to as invariance. ☹

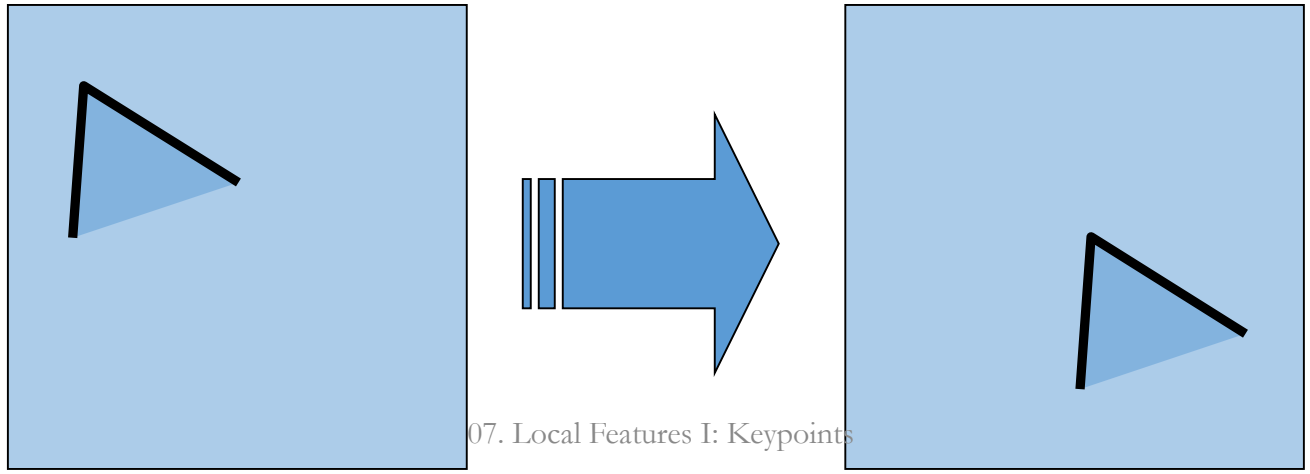
Adapted from K. Grauman

Invariant to translation?

What happens if image is translated?

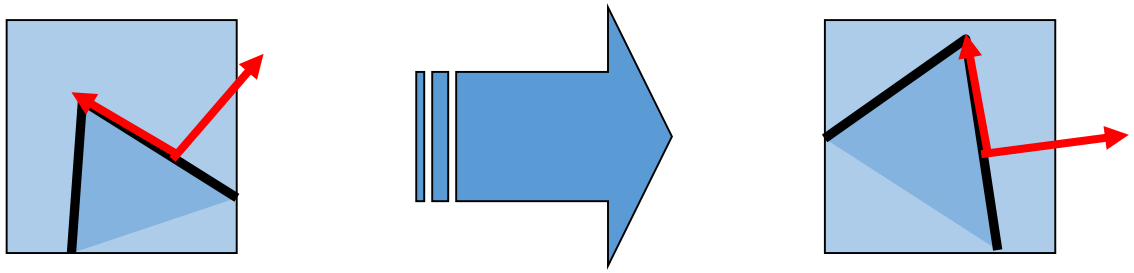
- Derivatives, H obtained through convolution,
- Eigenvalues based only on derivatives

Thus Harris corners; detected location is *equivariant to translation*, and response (R) is *invariant to translation*



Adapted from: S. Seitz, D. Frolova, D. Simakov

Invariant to Rotation?



- Eigenvectors represent the direction of maximum / minimum change in appearance, so they rotate with the patch
- Eigenvalues represent the corresponding magnitude of maximum/minimum change so they stay constant

Corner response is only dependent on the eigenvalues → invariant to rotation

Corner location is as before equivariant to rotation.

Adapted from: S. Seitz, D. Frolova, D. Simakov

Invariant to Photometric Transformations?

Assume linear change: $I' = aI + b$

Change to image gradients?

- $I'_x = aI_x \qquad I'_y = aI_y$

Second Moment matrix?

- $H' = a^2 H$

Eigenvalues & cornerness response function?

- $\lambda'_i = a^2 \lambda_i \qquad R' = a^2 R$

a^2 if R is $\min(\lambda_1, \lambda_2)$, a^4 if R is Harris operator

Does this change probability of detecting a corner?

Yes, since R' is now scaled by a^2

What happens if no scaling (i.e., a = 1)?

R' is not affected.

Adapted from: S. Seitz, D. Frolova, D. Simakov

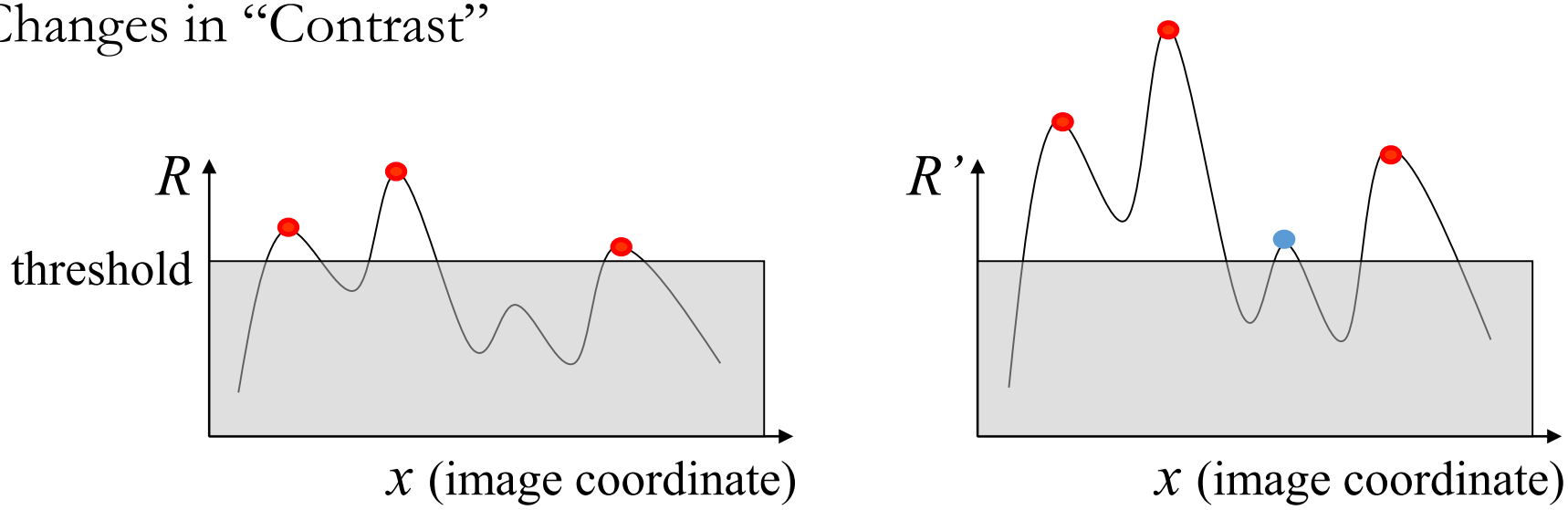
Invariance to photometric transformations?

The Harris corner detector (both locations and probability of detection) is invariant to additive changes in intensity

Changes in overall "Brightness"

It is not invariant to scaling of intensity $I' \rightarrow a\, I$

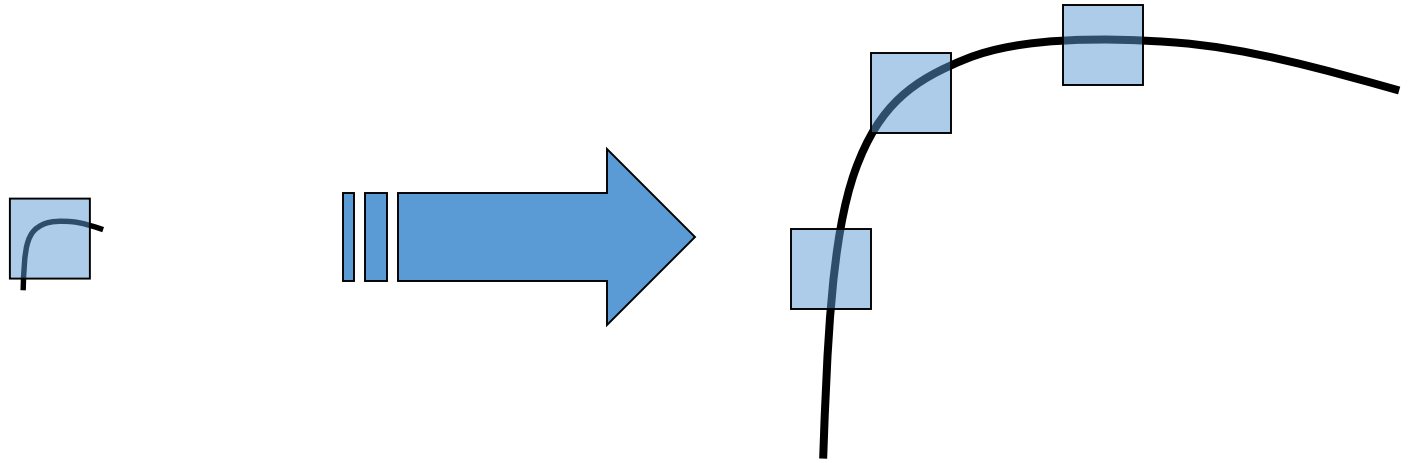
Changes in "Contrast"



Adapted from: S. Seitz, D. Frolova, D. Simakov

Invariance to Scaling?

What was one patch earlier is now many.



Corner

all points will be classified as edges

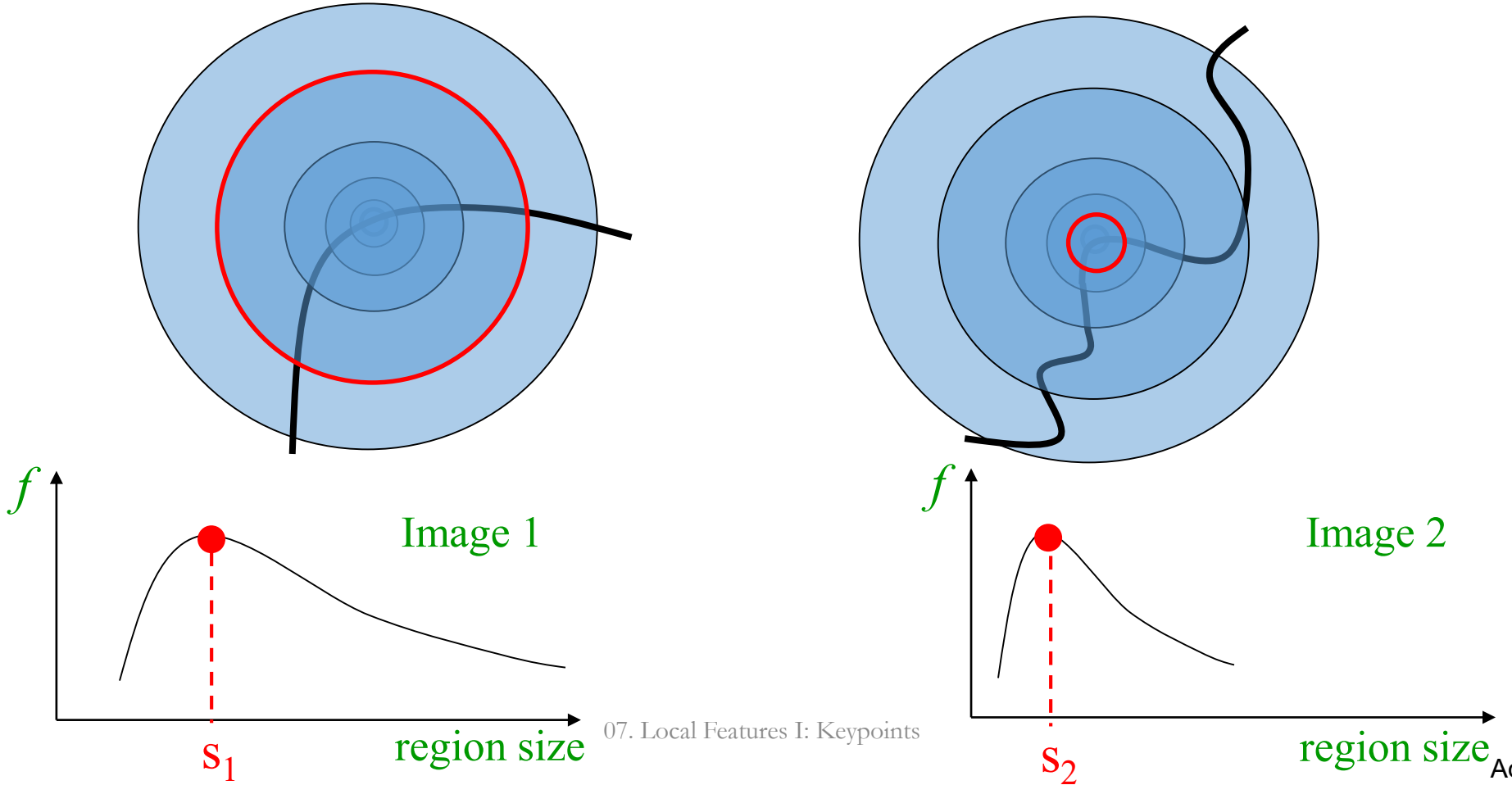
Not equivariant to scaling

Adapted from: S. Seitz, D. Frolova, D. Simakov

Automatic scale selection

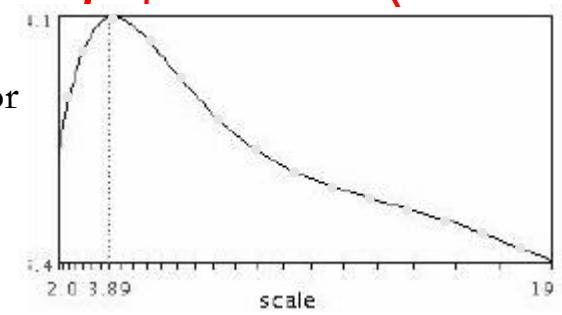
One definition of f could be the Harris operator (R) … another is the Laplacian of Gaussians (coming up)

When looking for keypoints, find the scale that gives a local maximum of f



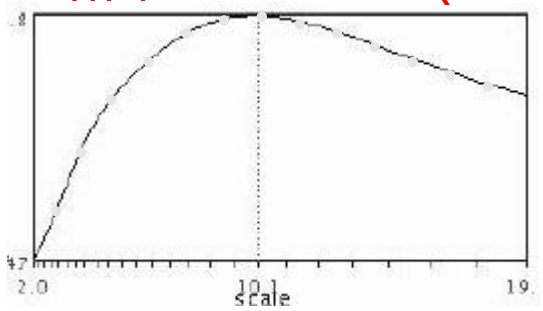
Adapted from: K. Grauman

Automatic Scale Selection



Function responses for increasing scale (scale signature)

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



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

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

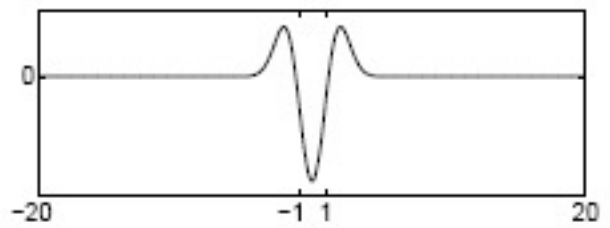
Same operator responses if the patch contains the same image up to some scale factor.

Slide credit: K. Grauman & B. Leibe

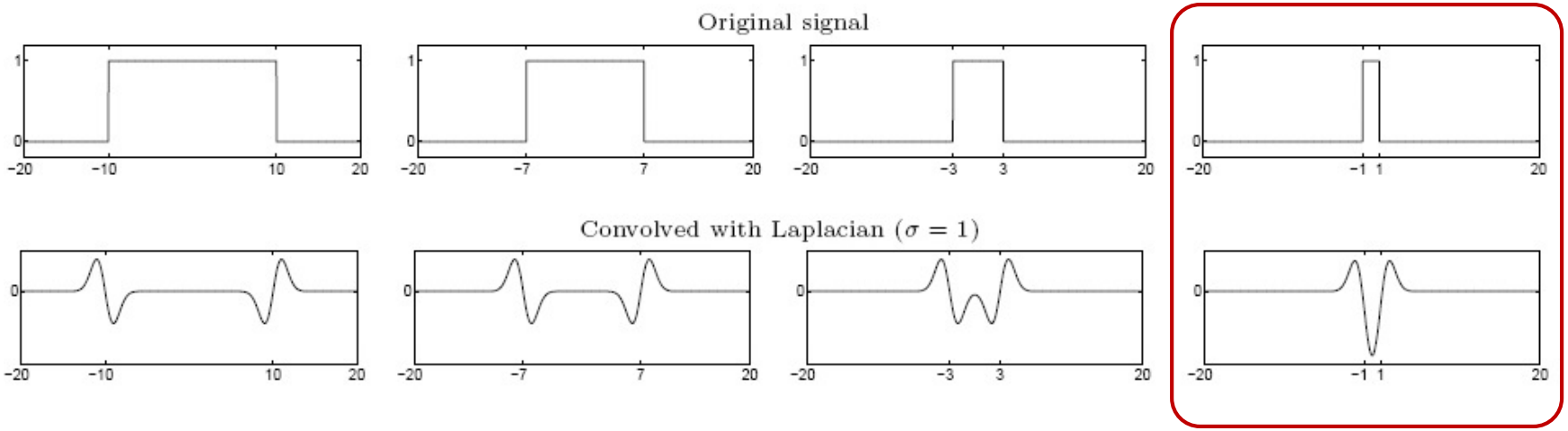
LoG: Laplacian of Gaussians

"Blob detector"

Laplacian of Gaussian (LoG) – 1D

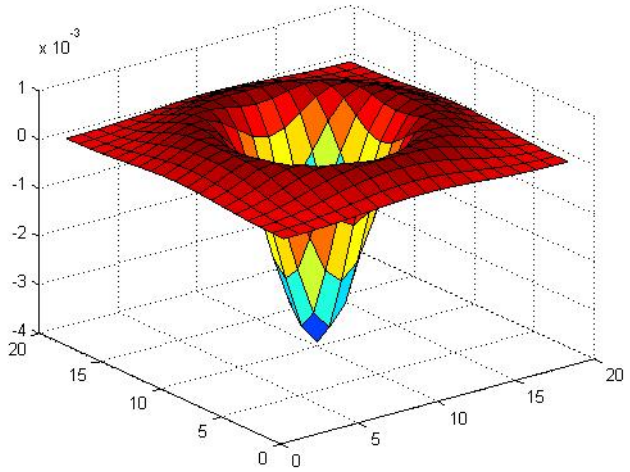


2nd derivative of Gaussian

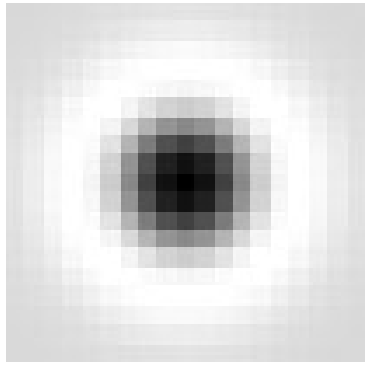


Highest response when the signal has the same **characteristic scale** as the Gaussian.

Laplacian of Gaussian (LoG) – 2D



"Blob detector"



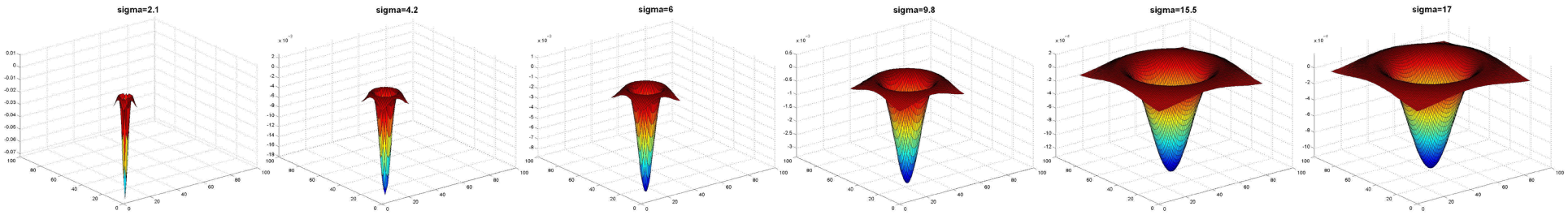
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Laplacian operator

very similar to a difference of Gaussians – i.e. a Gaussian minus a slightly smaller Gaussian (cheaper approximation than taking 2nd derivative)

Adapted from K. Grauman

Applying different Laplacian filters

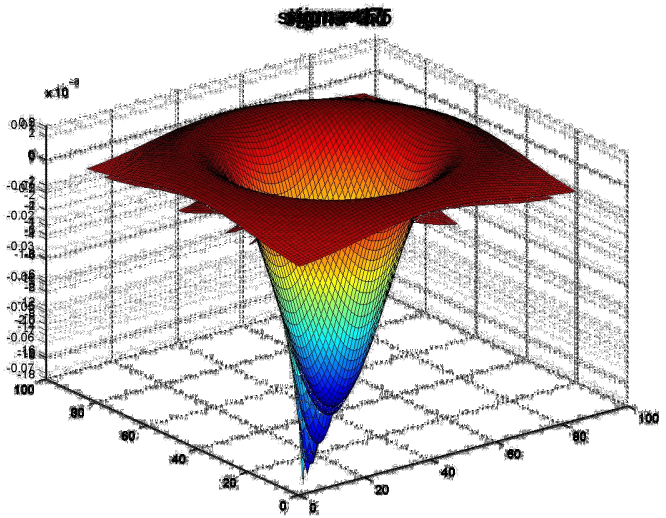


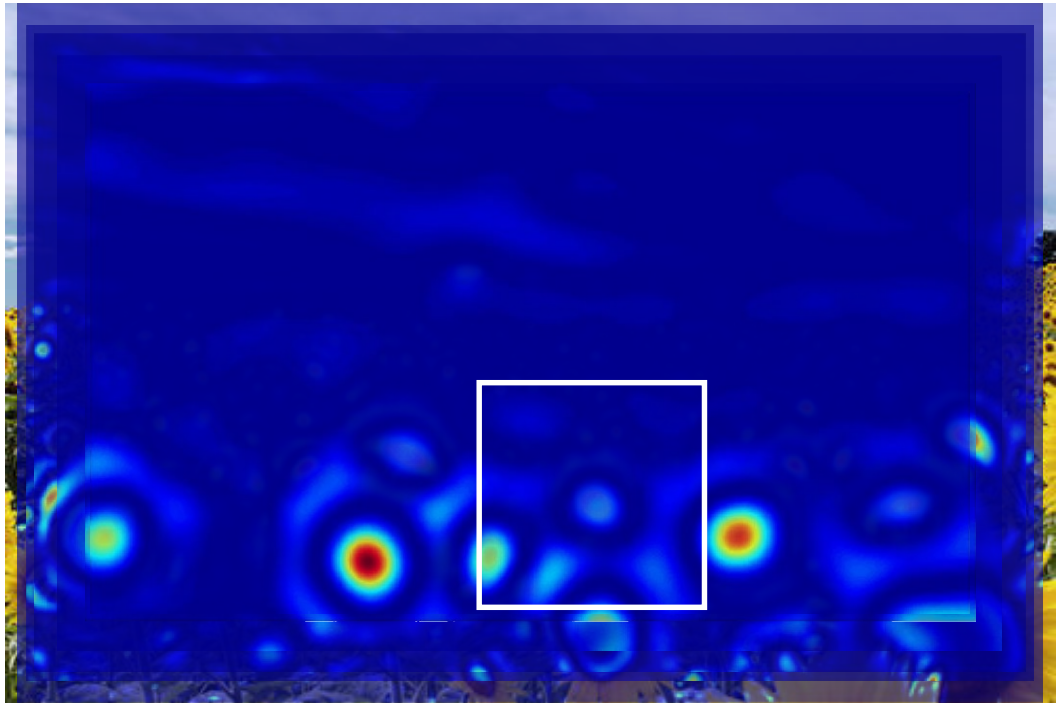
Full size

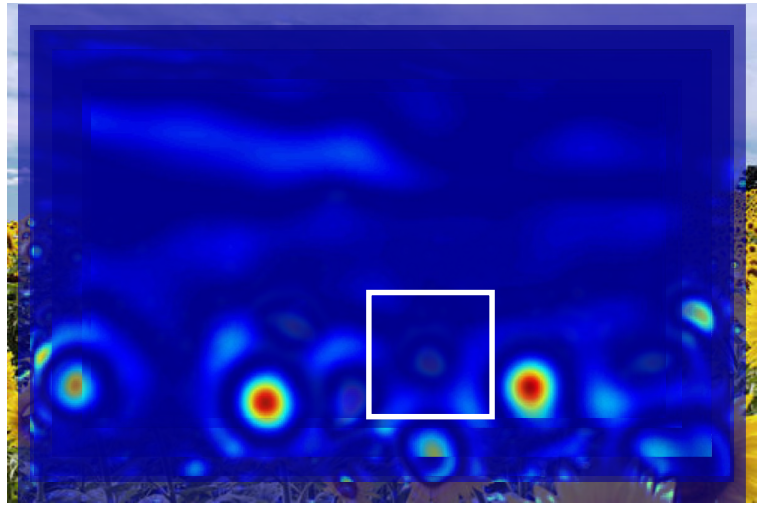
3/4 size

Focus on what happens to these 2 flowers.

Adapted from K. Grauman, K. Kitani



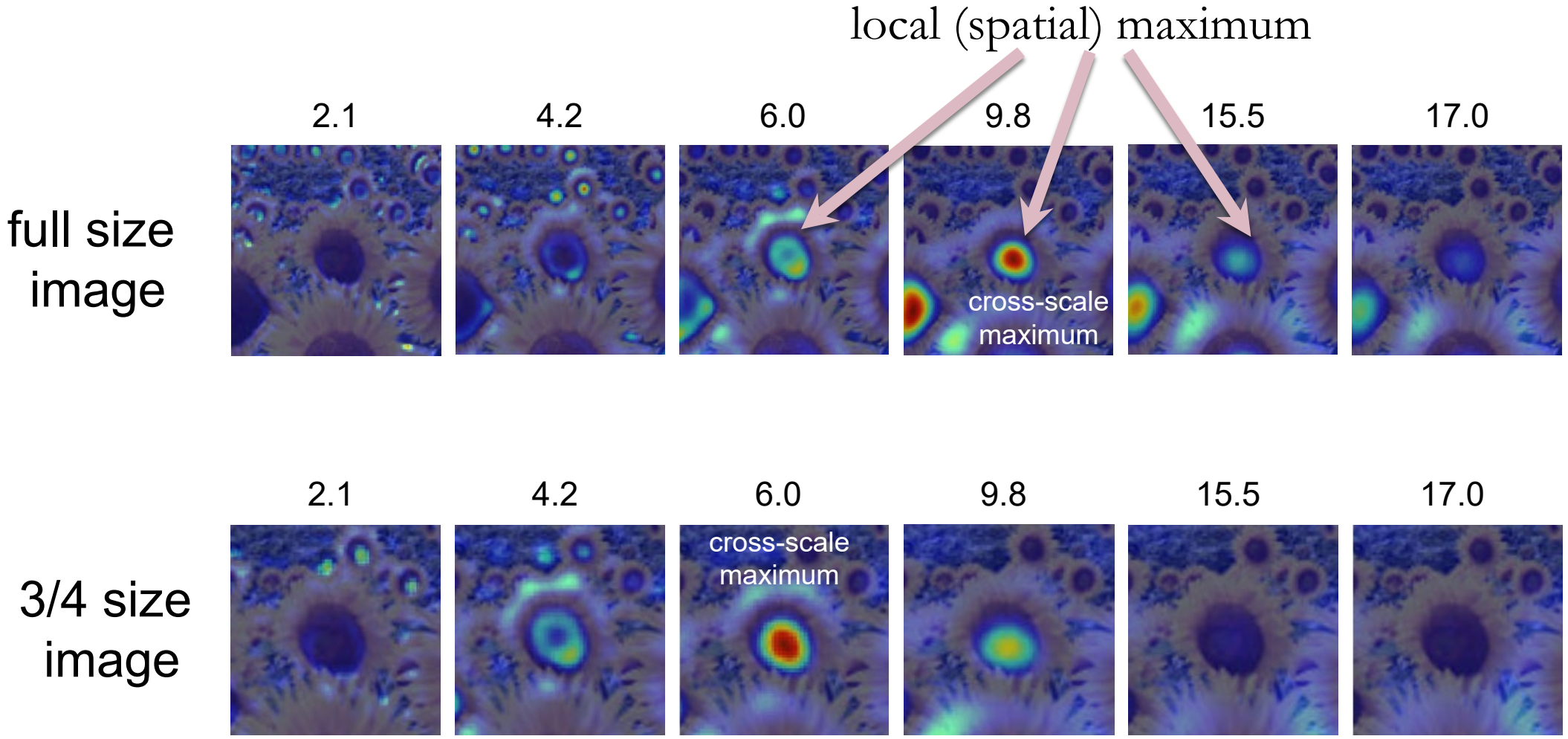




jet color scale
blue: low, red: high

Adapted from K. Grauman, K. Kitani

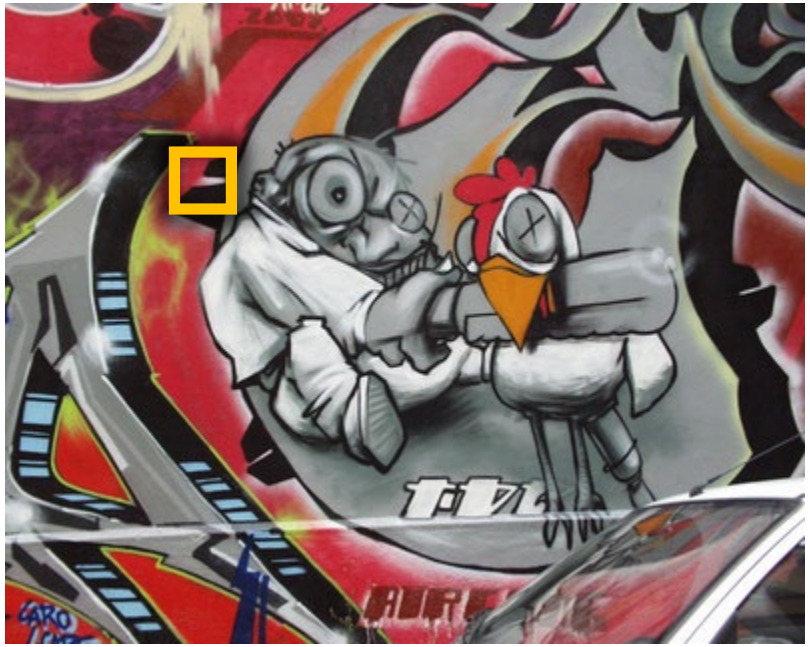
Finding the Optimal Scale



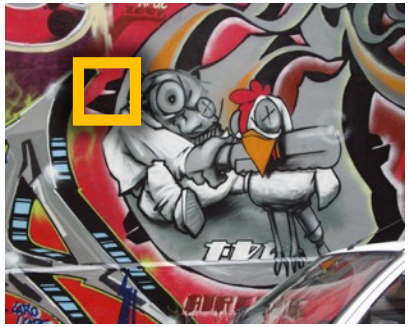
Adapted from K. Grauman, K. Kitani

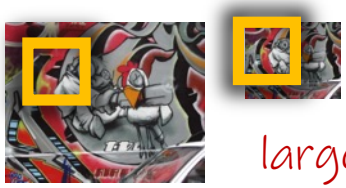
Implementation

Instead of computing $f()$ for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid



smallest scale





largest scale

Slide credit: K. Grauman & B. Leibe

Summary

- "Corners" are good candidates as keypoints; can be found by looking at the Hessian matrix (eigenvalues are both large and approximately equal)
- Harris operator is one (relatively efficient) way of finding these corners
- Harris corners are equivarient to geometric transformations such as translation, rotation, semi-invariant to photometric transformations (brightness changes but not contrast) and not invariant to scaling
- To introduce robustness to changes in scale, we can use automatic scale selection as a part of finding keypoints
- The Laplacian of Gaussian (LoG) operator is an alternative keypoint detector which detects "blobs" and is scale-sensitive