Local Image Features

Dr. Randy C. Hoover

Department of Electrical and Computer Engineering
South Dakota School of Mines and Technology
Rapid City, SD 57701, USA

March 3, 2019

Last few lectures!

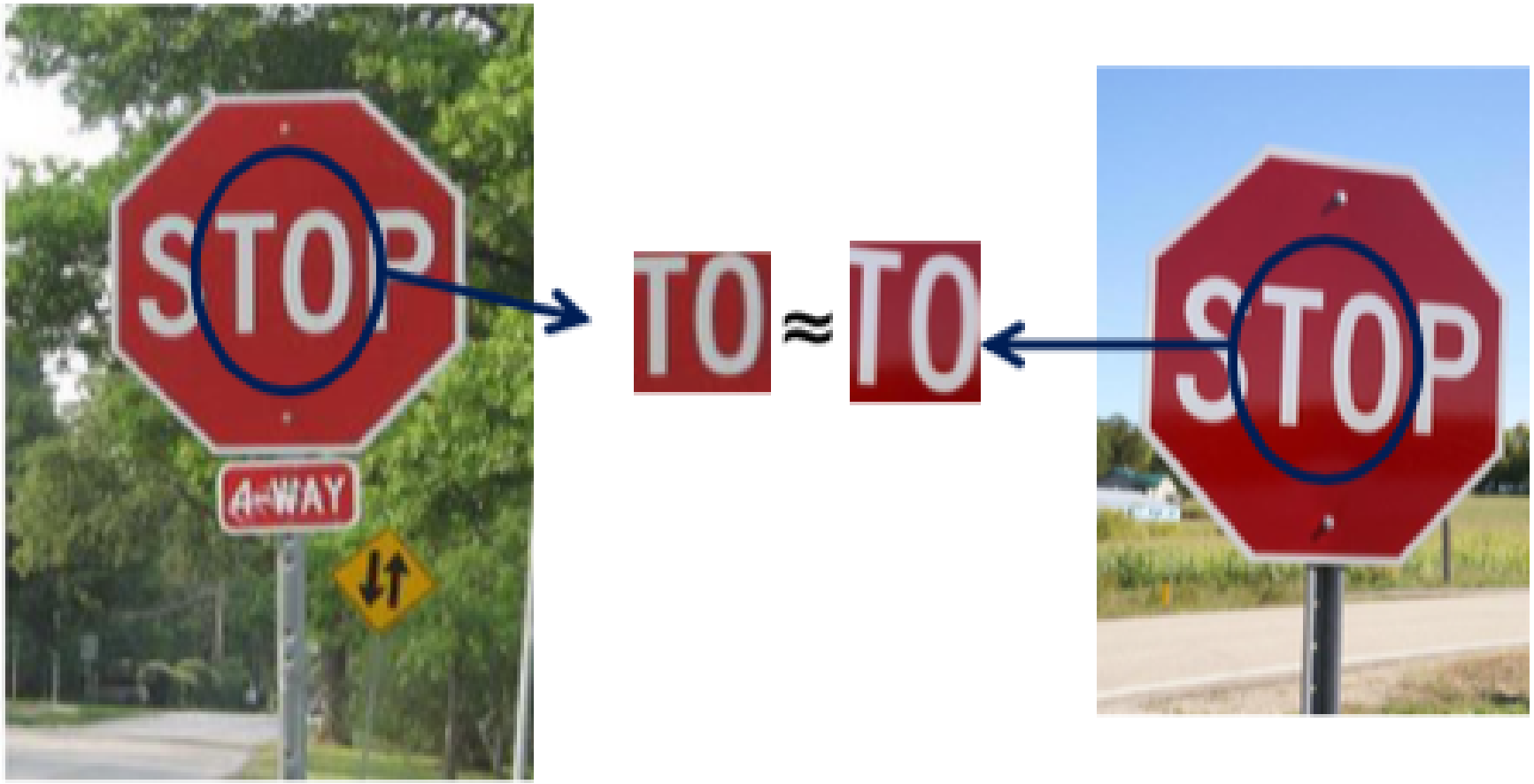
- Image filtering:
  - Smooth/sharpen based upon different kernel
- Template matching:
  - Filter kernels "look like" the features they enhance
  - Use normalized cross-correlation to "match" kernel response
- Image gradients:
  - Can convolve using several finite-difference kernels
  - Noise is amplified in image derivative
  - Smooth with Gaussian $\Longrightarrow$ take a derivative of Gaussian kernel and then convolve to get image derivative
- Edge detection:
  - Magnitude of the image gradient "enhances" large deviations in intensity (edges)
  - Edge direction is determine by the direction of the gradient
  - Use non-maximum suppression and hysteresis thresholding to create a binary edge map
  - Can use other techniques to "pull out" interesting features (Hough transform)

Today! (and next few lectures)

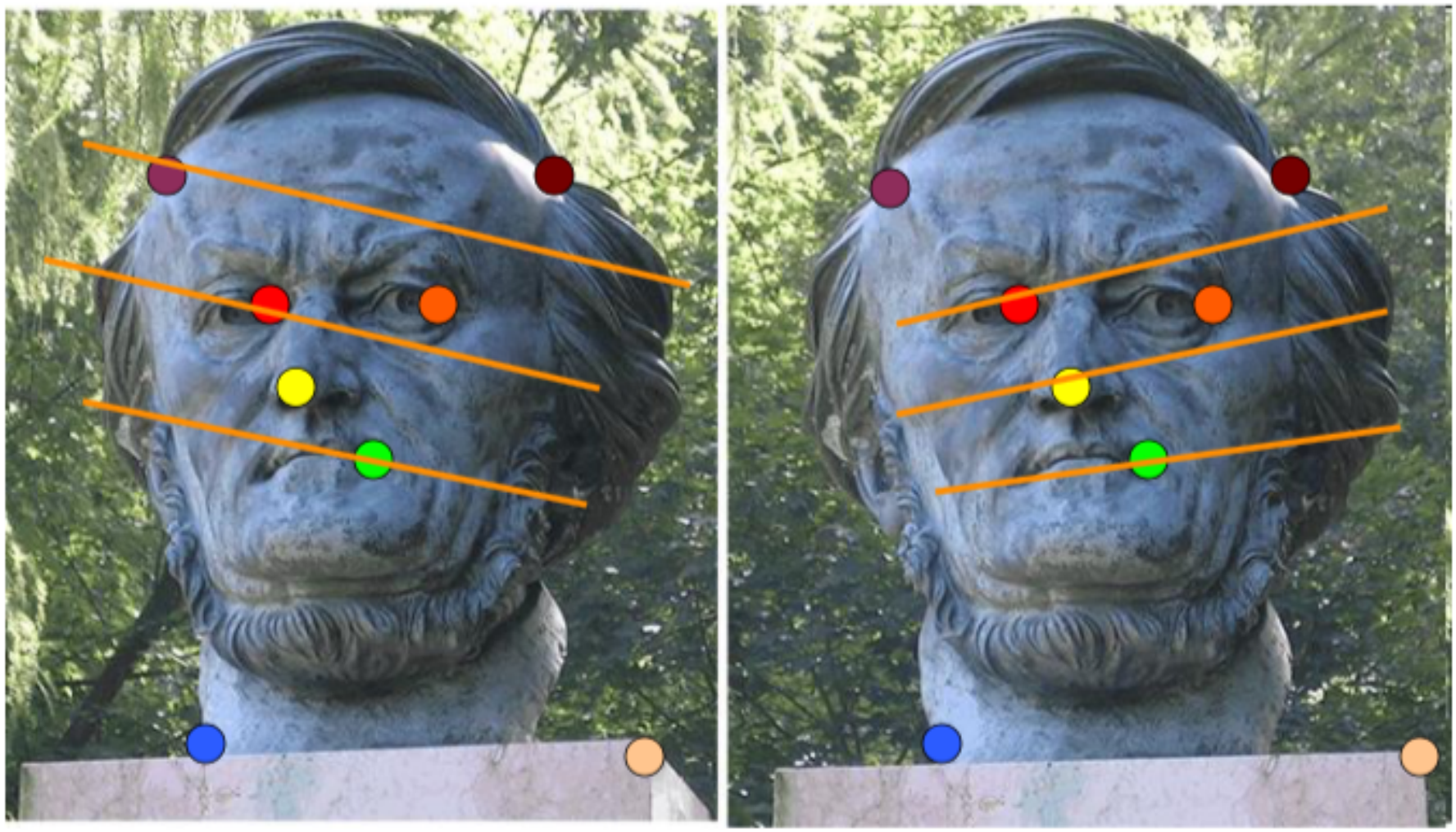
- Image features (filters $\longrightarrow$ edges $\longrightarrow$ corners)
  - How to extract local invariant features
    - Sometimes referred to as: interest points, key points, etc.
- Description:
  - How to develop a "description" of said features for later correspondence
- Feature matching:
  - How to match features across multiple images

Correspondence across different views/scale/aspect/etc.

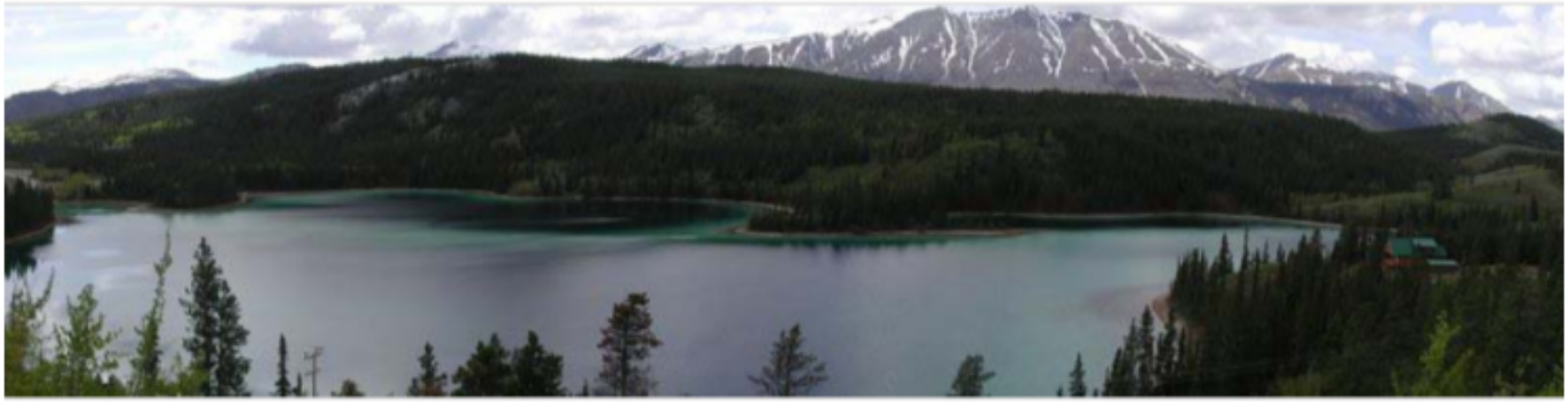
- Correspondence: matching points, patches, edges, or regions across images.



Example: Estimate the "fundamental matrix" between views (3D reconstruction)

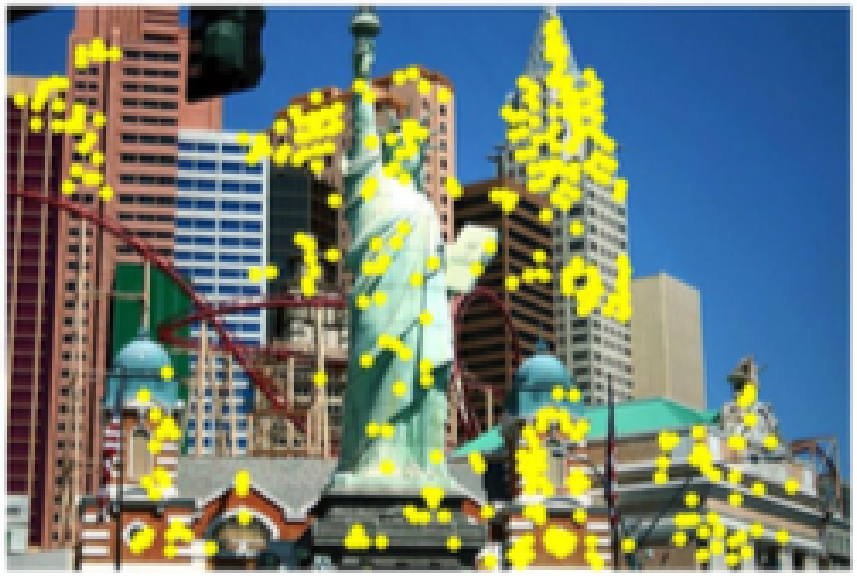


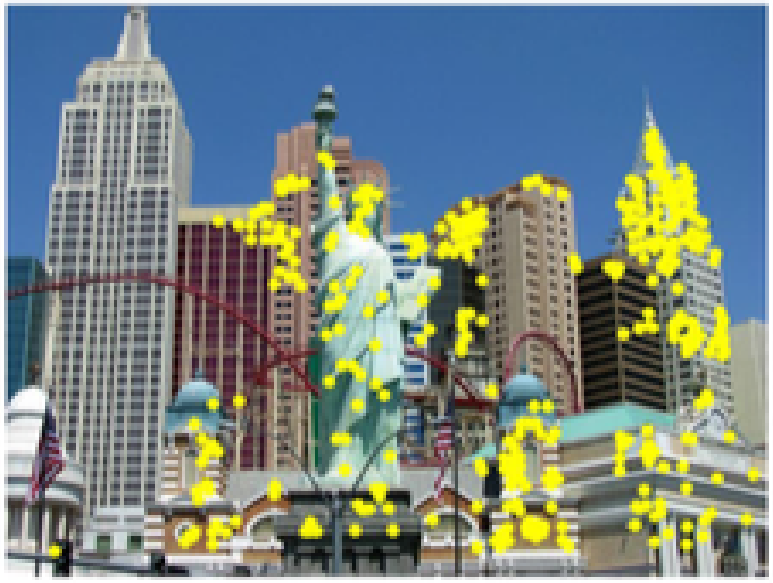
Example: Image alignment (project 3)



Features build the foundation of computer vision

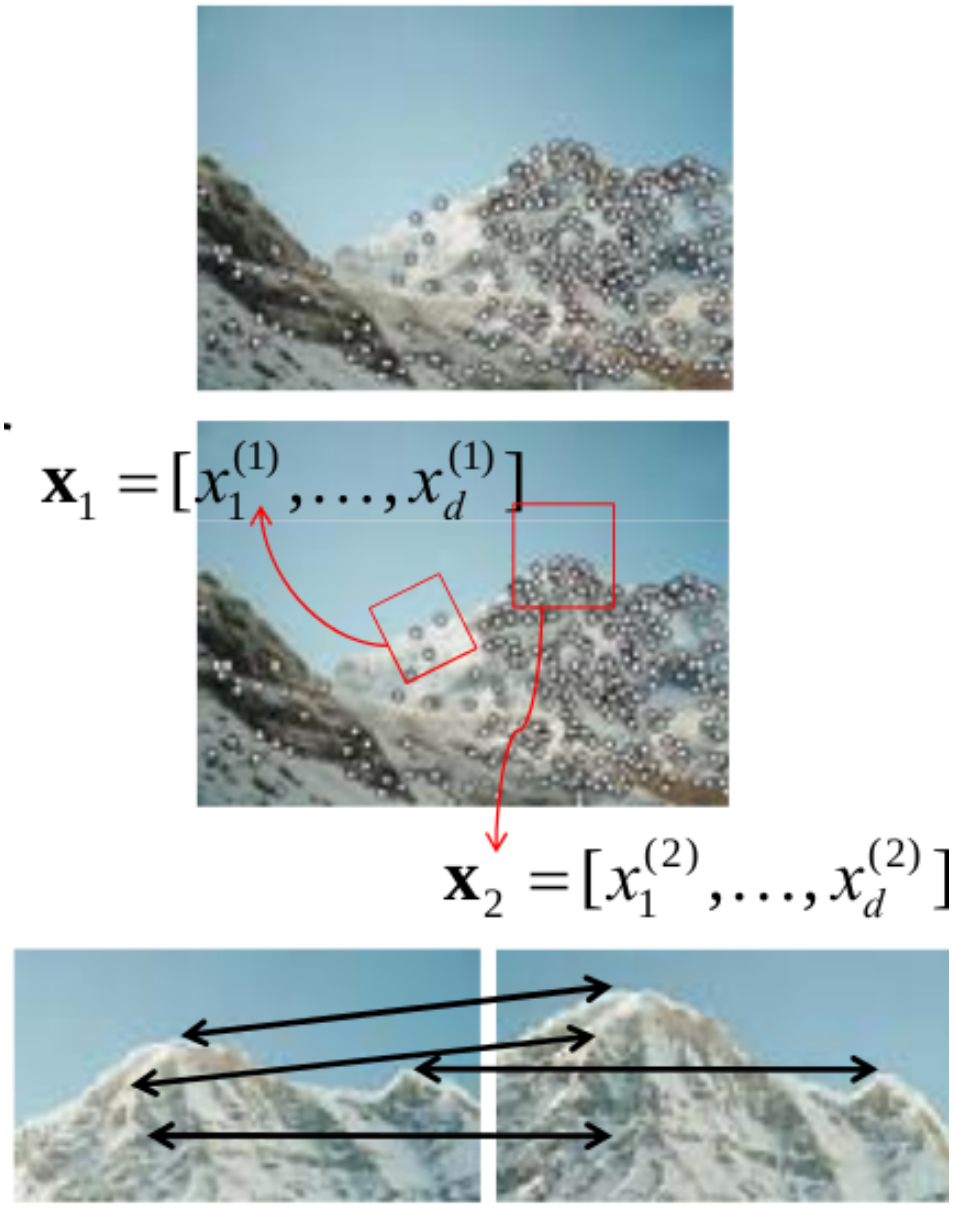
- Many application areas:
  - Motion tracking (surveillance/recon.)
  - Robotics (learning, motion planning, interaction, etc.)
  - Recognition/Classification
  - etc.





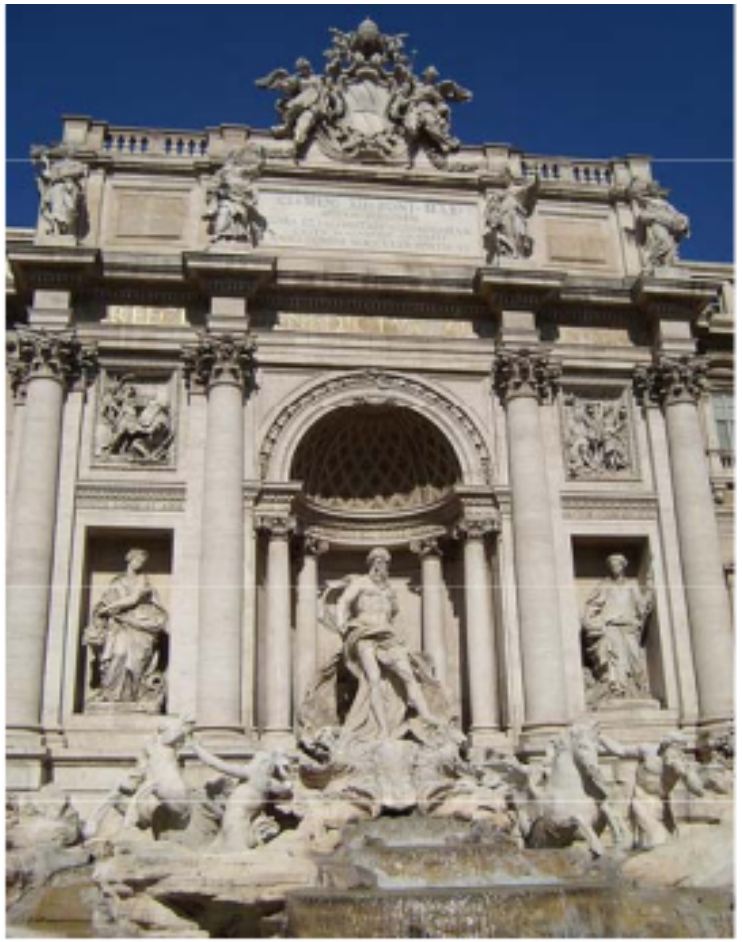
Local features: main components

- Detection:
  - Identify points of interest.
- Description:
  - Extract a feature descriptor surrounding each interest point.
- Matching:
  - Determine correspondence between descriptors from multiple views.



How do we match subsequent images???

by Diva Sian



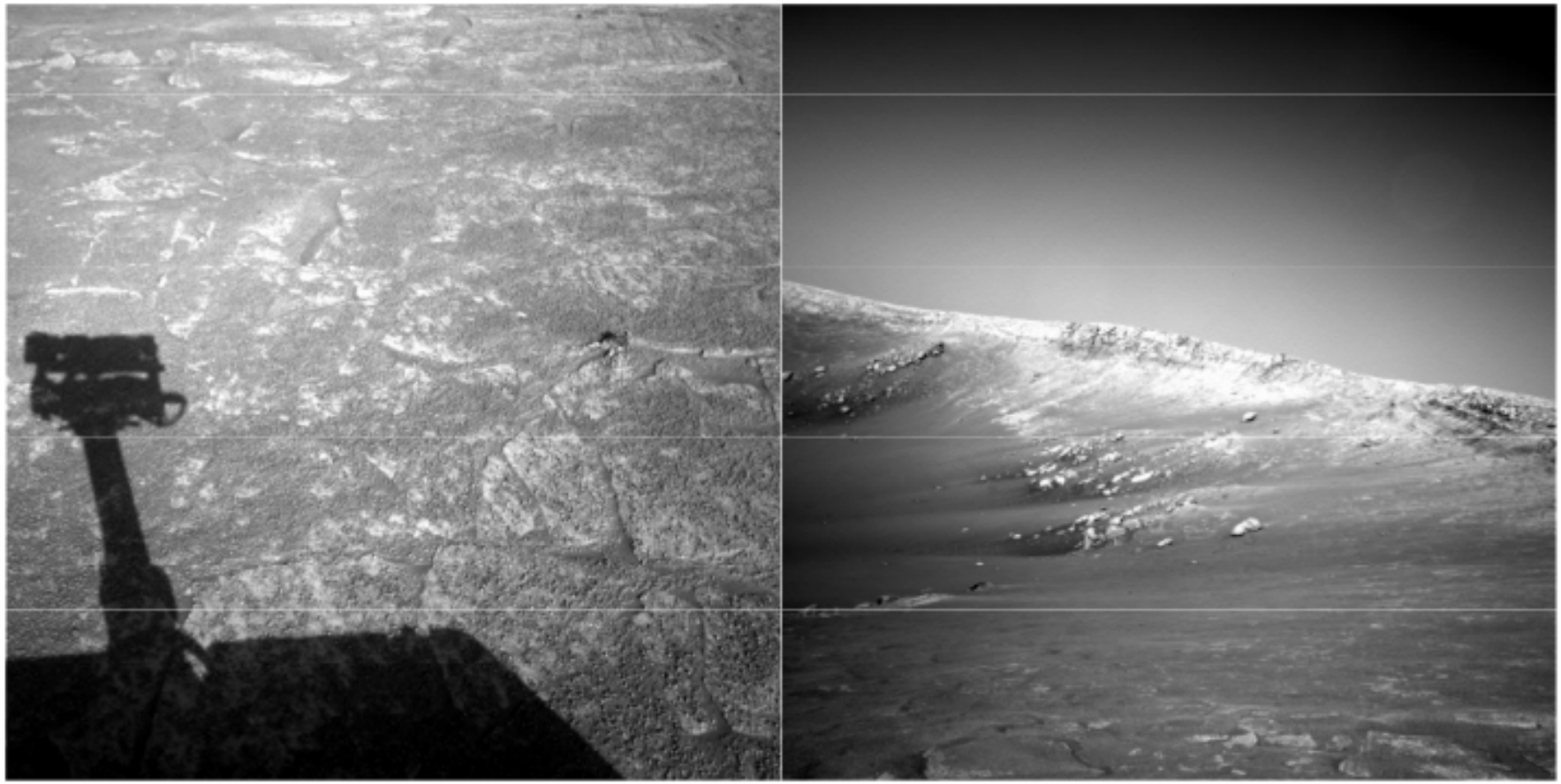
by swashford

How do we match subsequent images???

by Diva Sian

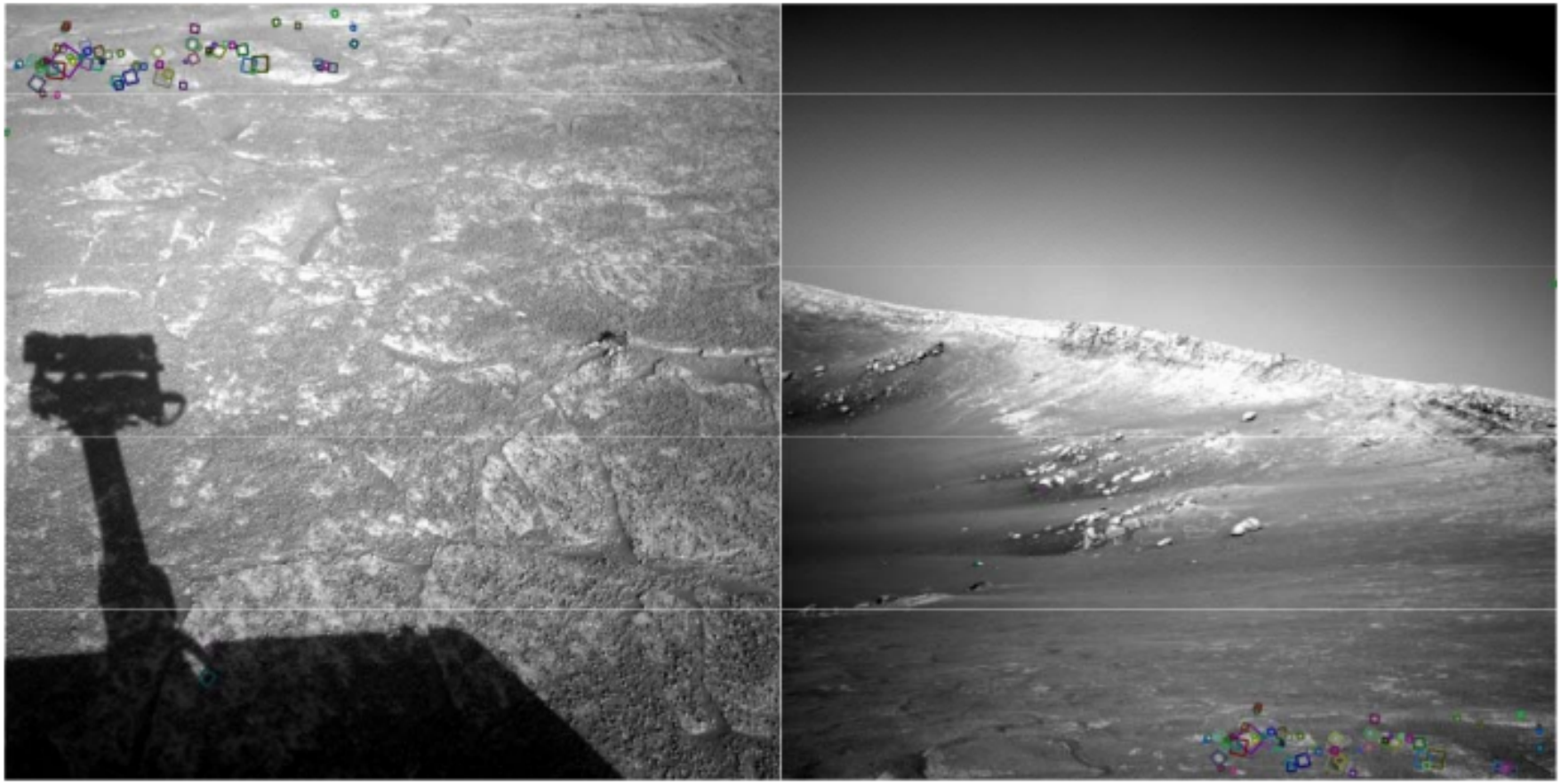
by scgbt

How do we match subsequent images???



NASA Mars Rover images

How do we match subsequent images???



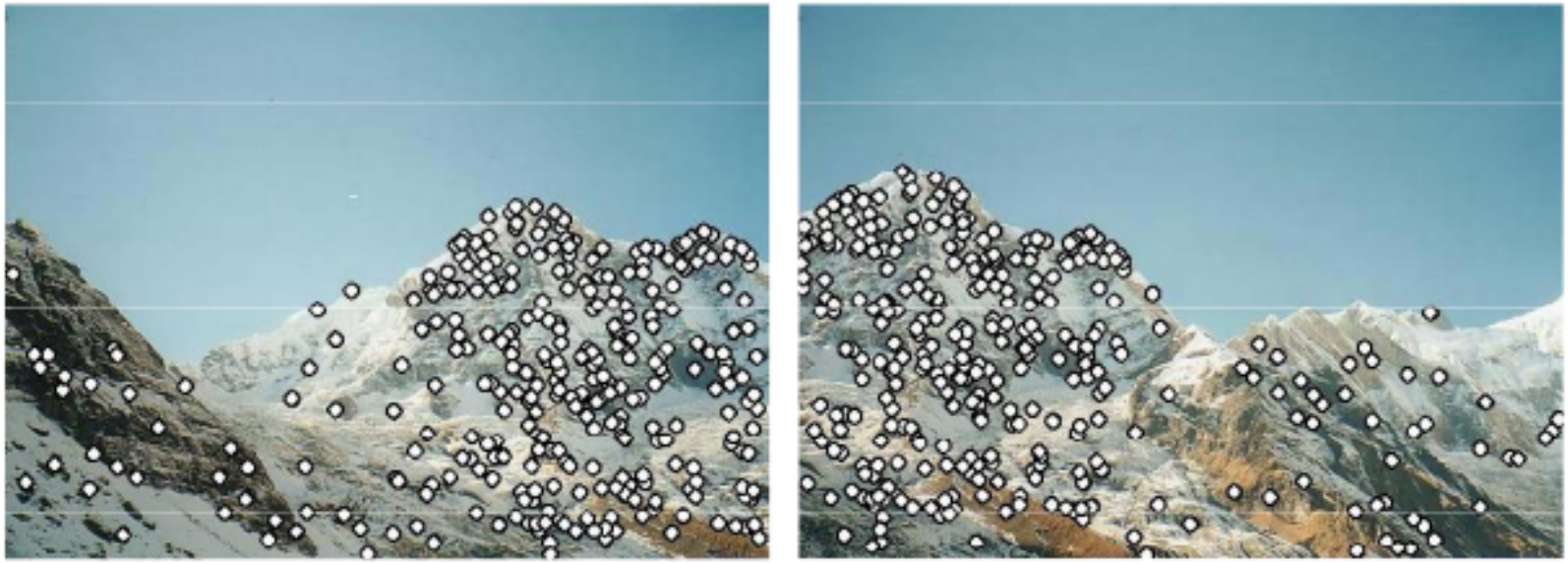
NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Suppose we wish to align two images into a panoramic!

- First, look for local features in the images that match well!
- How would we do this by "eying" it?

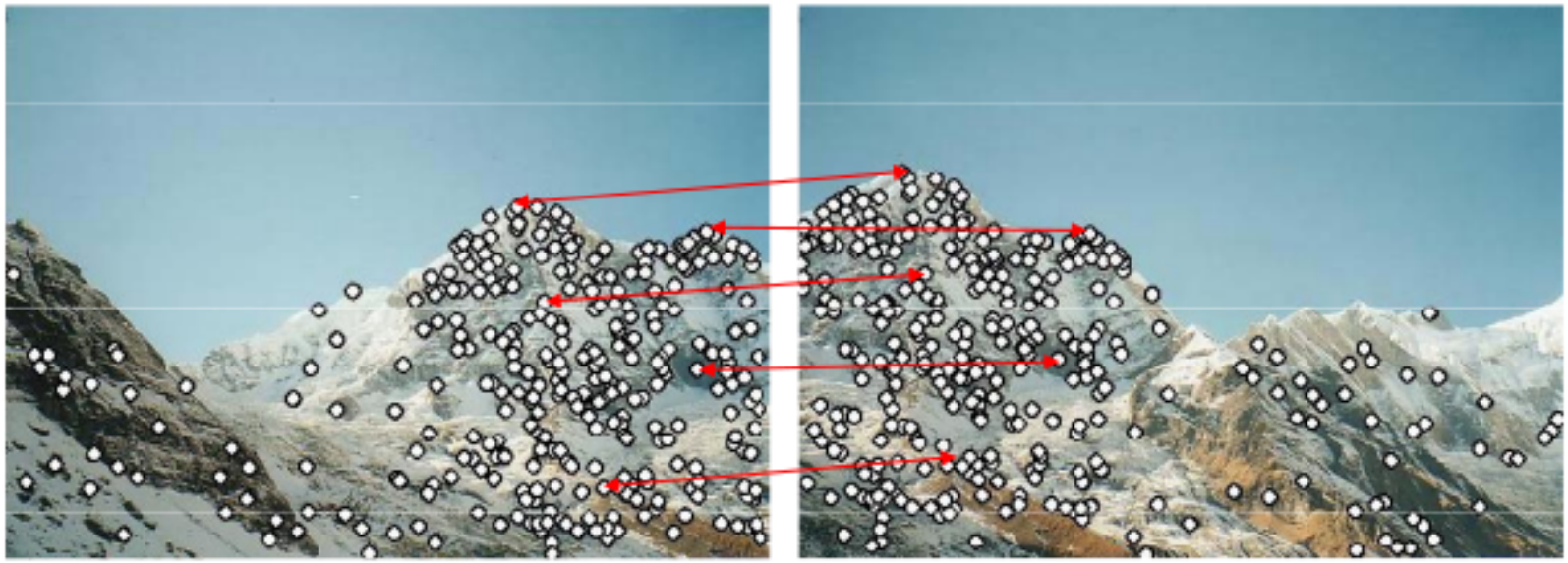
Local features and alignment

1. Detect a lot of features in both images



Local features and alignment

1. Detect a lot of features in both images
2. Find corresponding pairs

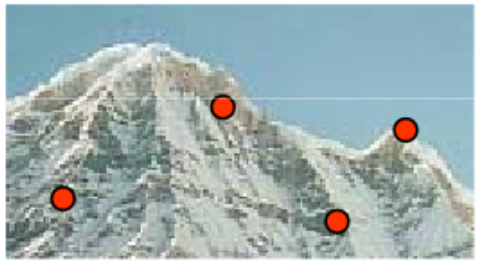


Local features and alignment

1. Detect a lot of features in both images
2. Find corresponding pairs
3. Use these pairs to align the images

Easier said than done!

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



no chance to match!

- We need a repeatable detector

Easier said than done!

- Problem 2:
  - For each feature detected, correctly recognize the corresponding point

- We need a reliable and distinctive descriptor

Our detector and descriptor should have some *invariant* properties!

- Geometric transformations: (rotation, scale, projections, affine shifts, etc.)

Our detector and descriptor should have some *invariant* properties!

- Geometric transformations: (rotation, scale, projections, affine shifts, etc.)
- Photometric properties: (illumination, color space, etc.)

Figure from T. Tuytelaars ECCV 2006 tutorial

Our detector and descriptor should have some *invariant* properties!

- Geometric transformations: (rotation, scale, projections, affine shifts, etc.)
- Photometric properties: (illumination, color space, etc.)
- Other: (noise, blur, compression, etc.)

Figure from T. Tuytelaars ECCV 2006 tutorial

Invariant local features

Subset/subregions within the image designed to be invariant to common geometric and photometric changes.

- Basic steps involved:
  1. Detect distinctive interest points within the image
  2. Extract invariant descriptors to describe said interest points

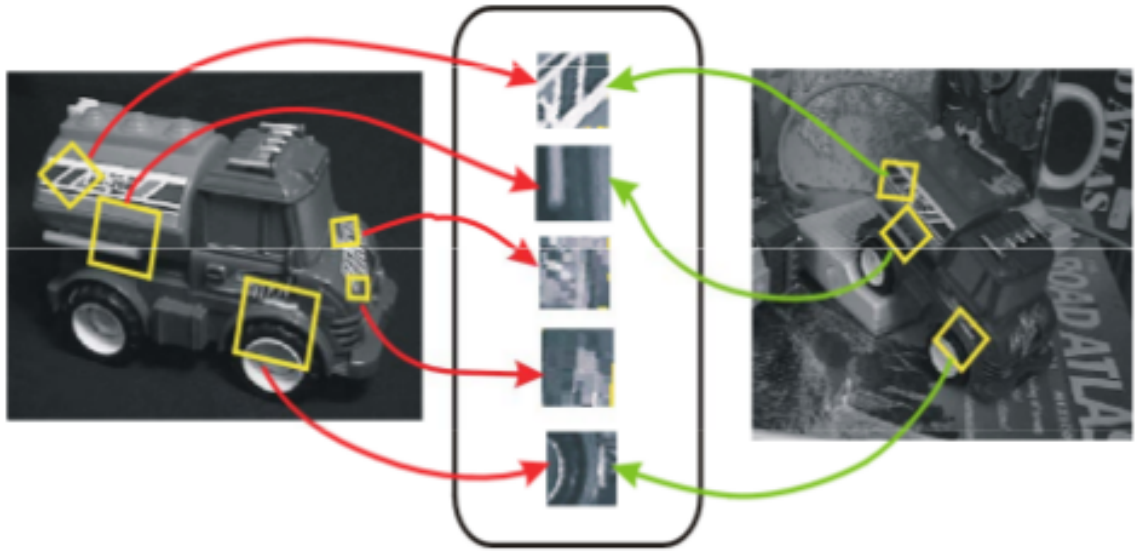


Figure: David Lowe

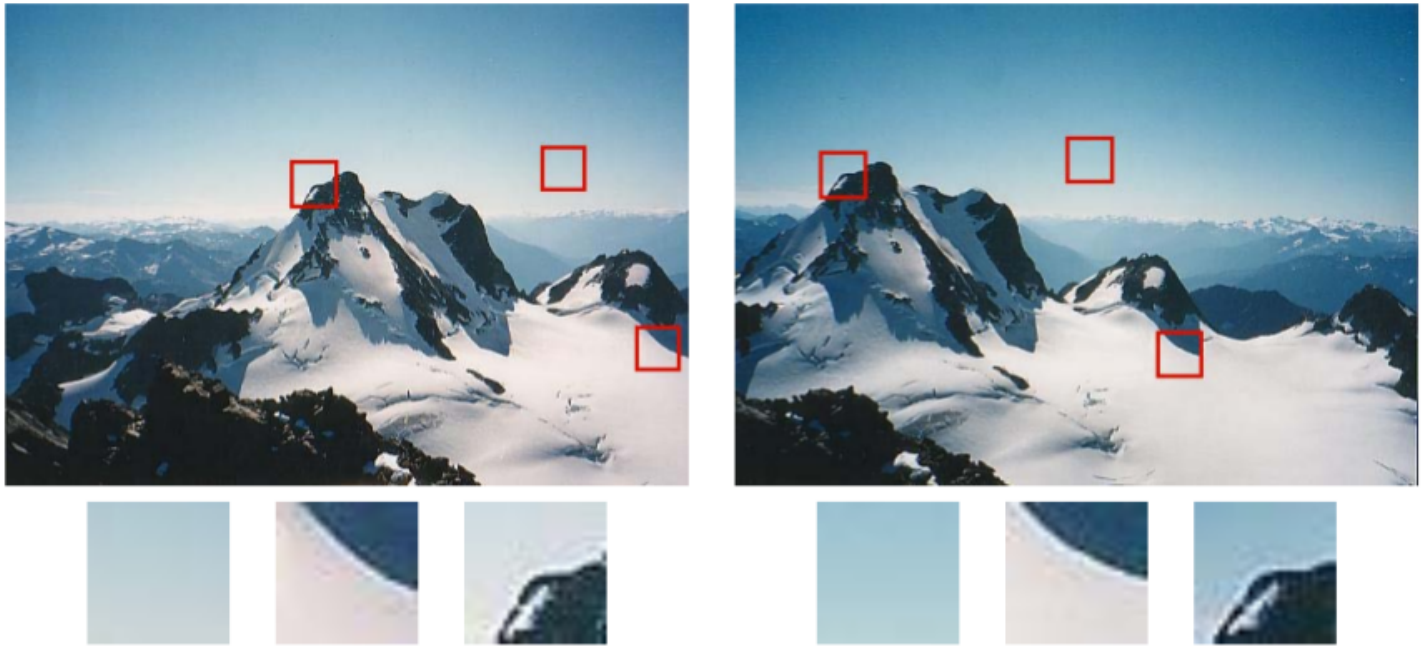
Main questions to be answered

How will these "interest" points be computed?

- What are considered to be salient features that we can **detect** in multiple images?
- How do we **describe** a local region?
- How do we establish **correspondences**, i.e., compute matches?

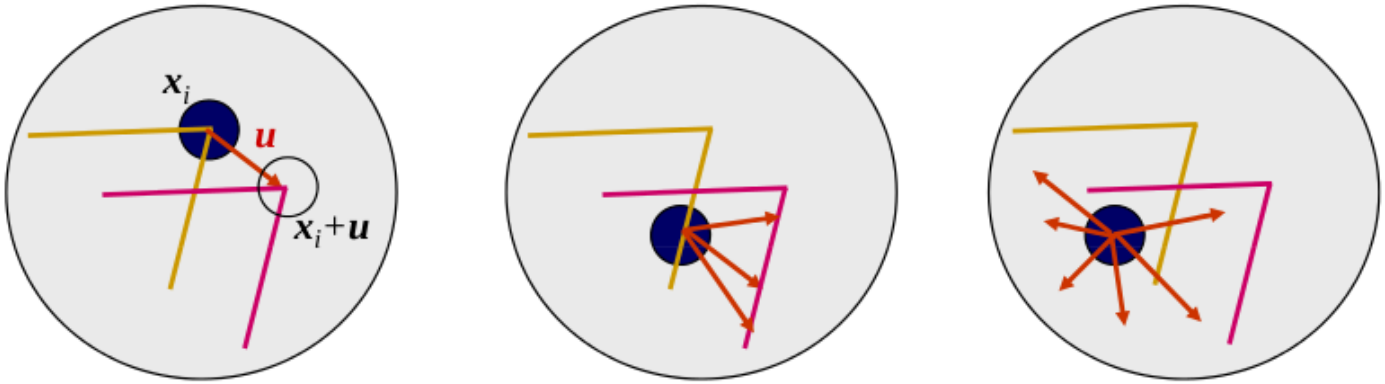
How can we find key features?

- Textureless patches can't be localized!



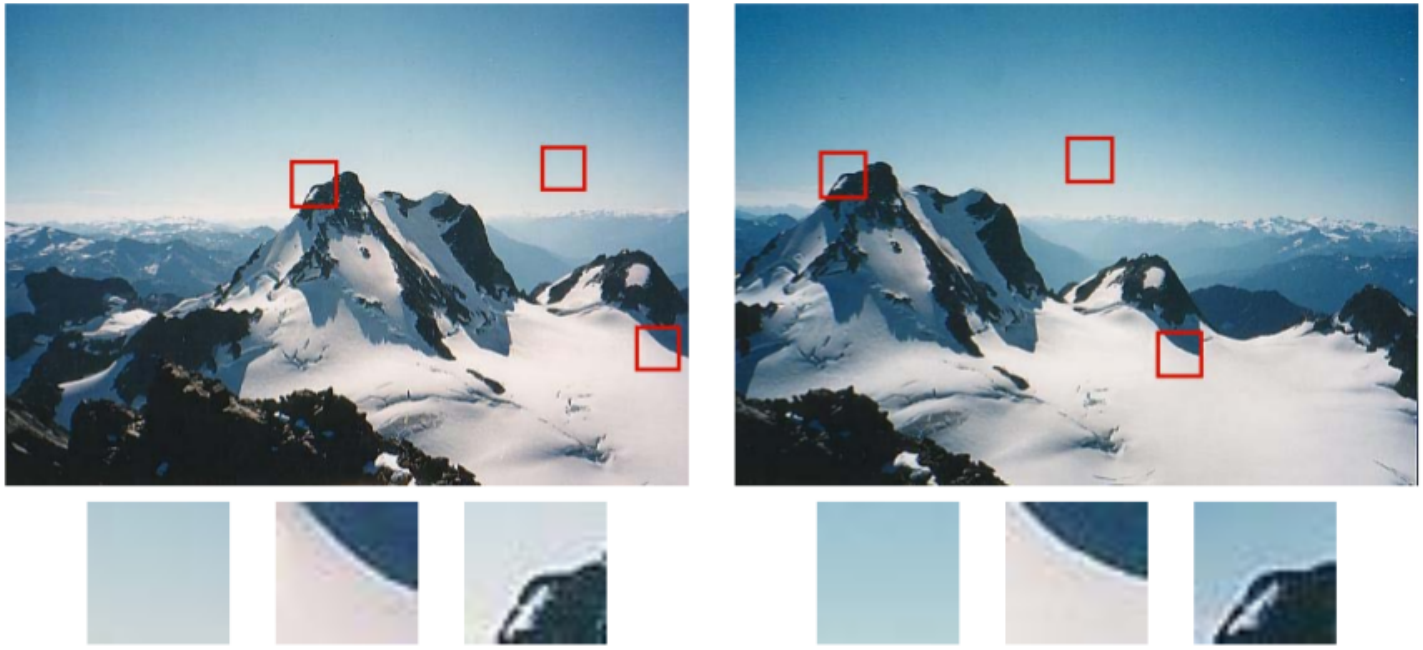
How can we find key features?

- Textureless patches can't be localized!
- Patches with large gradients do better
  - Straight line segments (single gradient directions) suffer from the aperture problem



How can we find key features?

- Textureless patches can't be localized!
- Patches with large gradients do better
  - Straight line segments (single gradient directions) suffer from the aperture problem
- As we just saw, patches with gradients in at least two (significantly) different orientations are a good place to start!



How do we know what a good feature is?

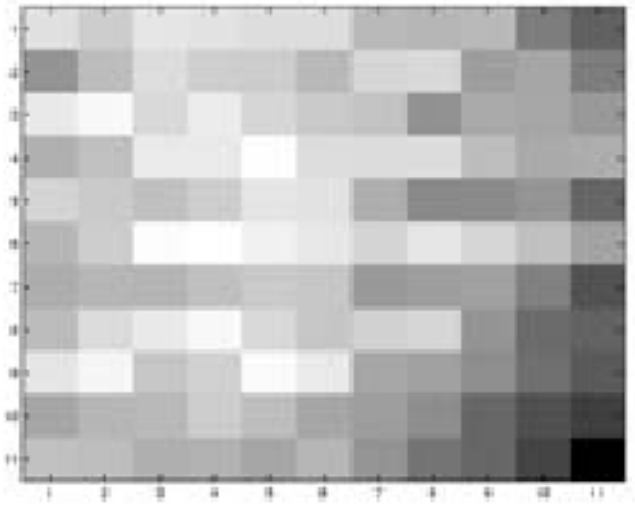
- Consider a small patch I_0 in an image
- We can compare this patch with a translated version of itself (by some Δu)
- Auto-correlation:

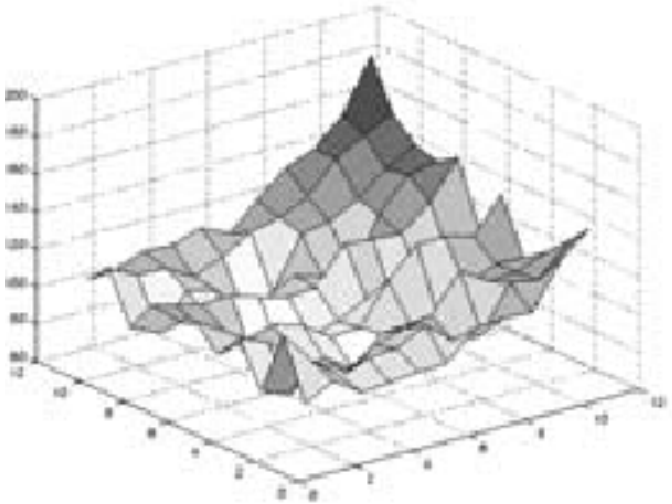
$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i - \Delta u) - I_0(x_i)]^2$$

How do we know what a good feature is?

- Consider a small patch I_0 in an image
- We can compare this patch with a translated version of itself (by some Δu)
- Auto-correlation:

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i - \Delta u) - I_0(x_i)]^2$$

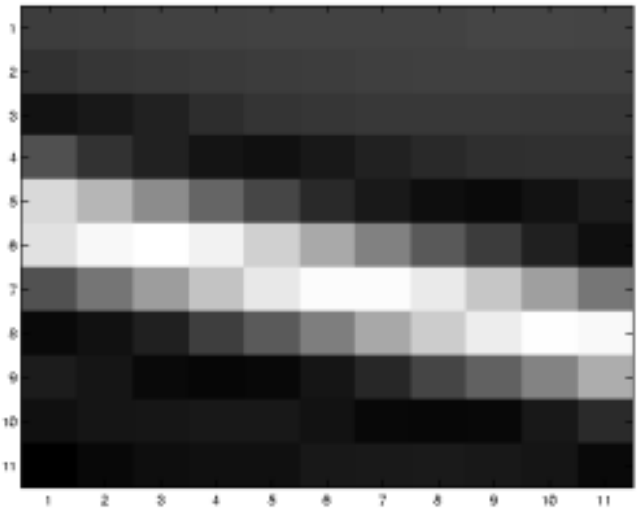


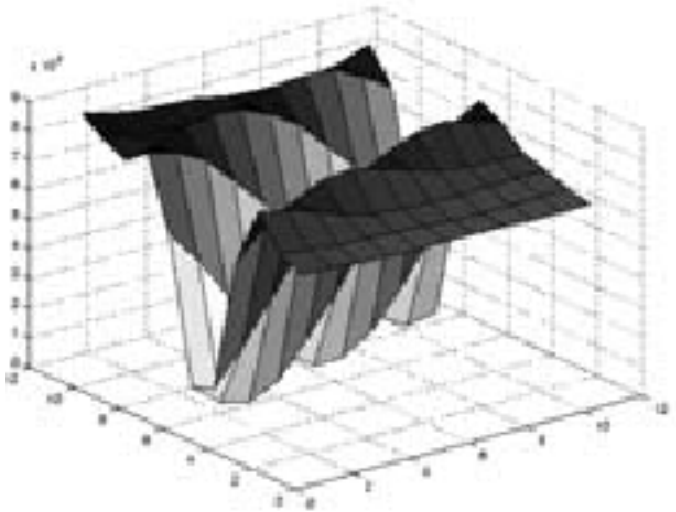


How do we know what a good feature is?

- Consider a small patch I_0 in an image
- We can compare this patch with a translated version of itself (by some Δu)
- Auto-correlation:

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i - \Delta u) - I_0(x_i)]^2$$

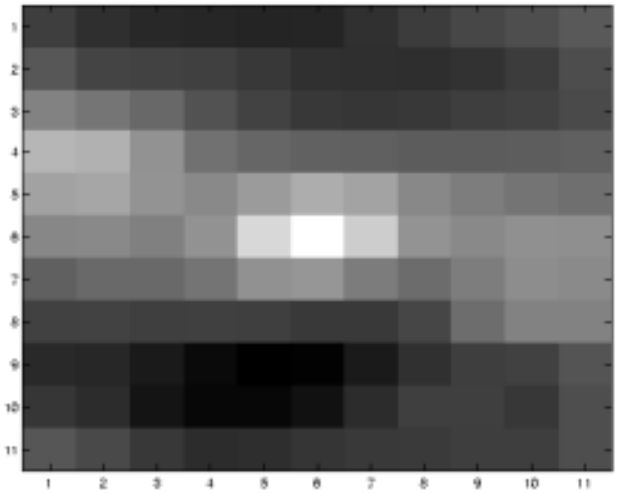


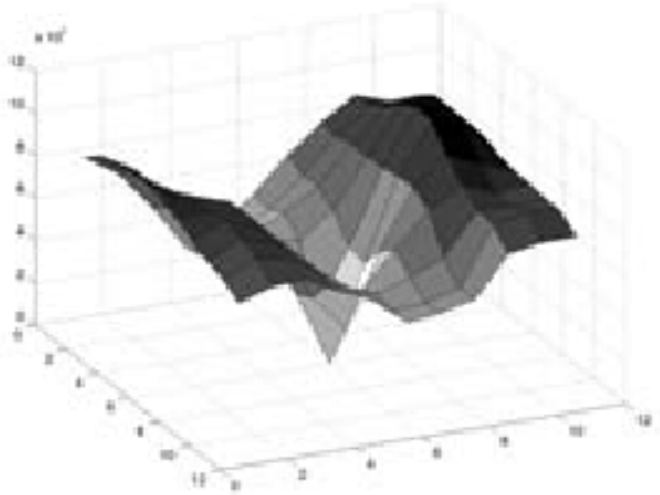


How do we know what a good feature is?

- Consider a small patch I_0 in an image
- We can compare this patch with a translated version of itself (by some Δu)
- Auto-correlation:

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i - \Delta u) - I_0(x_i)]^2$$





How do we know what a good feature is?

- Consider a small patch I_0 in an image
- We can compare this patch with a translated version of itself (by some Δu)
- Auto-correlation:

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i - \Delta u) - I_0(x_i)]^2$$

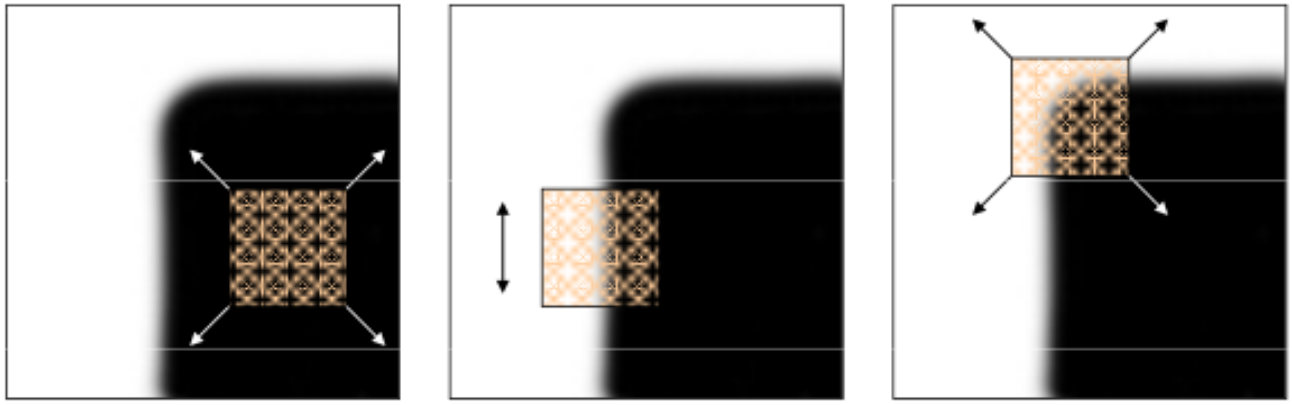
- Good features have gradients in multiple directions!
- Corners are a good place to start!

Finding corners

- **Key property:** in the region around a corner, the image gradient has *at least* two dominant directions!
- Corners are repeatable and distinctive (read the paper by Harris and Stephens)

Corners as distinctive interest points

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



"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

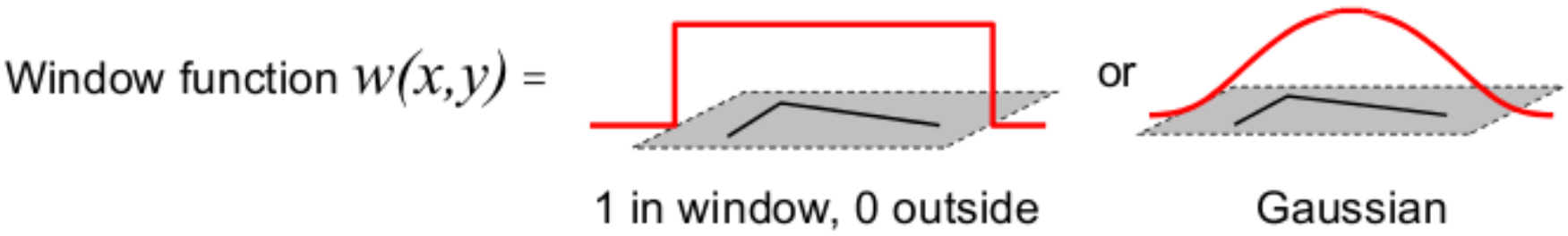
Source: A. Efros

Harris corners

- Being more specific, let Δu in the auto-correlation be represented by a shift $[u, v]$
- Then our auto-correlation becomes:

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window}} [\underbrace{I(x+u, y+v)}_{\text{Shifted intensity}} - \underbrace{I(x,y)}_{\text{Intensity}}]^2$$

where



Harris corners

- Note that a Taylor series expansion of $I(x+u, y+v)$ yields:

$$I(x+u, y+v) \approx I(x,y) + \nabla I(x,y) \cdot [u,v].$$

Recall:

$$\nabla I(x,y) = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right](x,y)$$

is the image gradient.

Plugging this in and simplifying (p.g. 212 in your text):

$$\begin{aligned} E(u,v) &\approx \sum_{x,y} w(x,y)[\nabla I(x,y) \cdot [u,v]]^2 \\ &= [u,v] H [u,v]^T \end{aligned}$$

where H is the 2×2 Harris matrix (a.k.a the auto-correlation matrix)

$$H = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} w * I_x^2 & w * I_x I_y \\ w * I_x I_y & w * I_y^2 \end{bmatrix}$$

Hessian corners

- So, what exactly does this mean and what is this $w(x, y)$?
- First, note that the matrix H looks very similar to the image Hessian [Beaudet78]

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

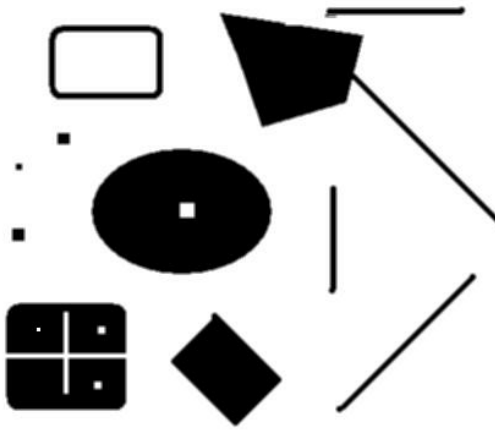
and $\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2$

Hessian corners

- So, what exactly does this mean and what is this $w(x, y)$?
- First, note that the matrix H looks very similar to the image Hessian [Beaudet78]

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

and $\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2$

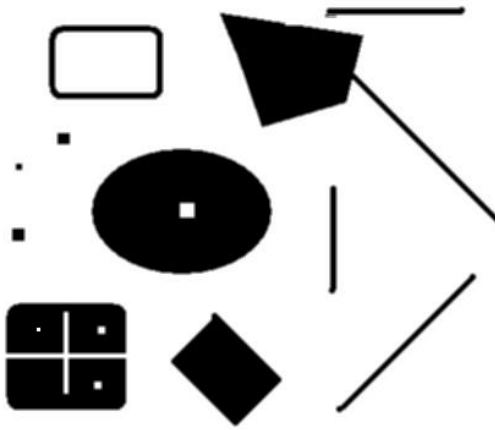


Hessian corners

- So, what exactly does this mean and what is this $w(x, y)$?
- First, note that the matrix H looks very similar to the image Hessian [Beaudet78]

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

and $\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2$

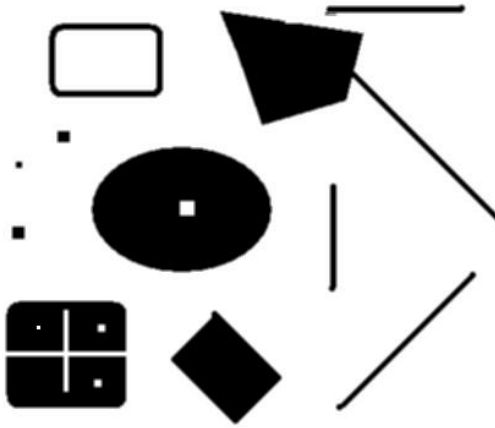


Hessian corners

- So, what exactly does this mean and what is this $w(x, y)$?
- First, note that the matrix H looks very similar to the image Hessian [Beaudet78]

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

and $\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2$

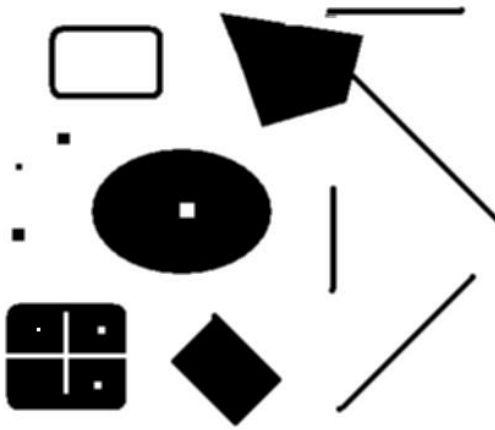


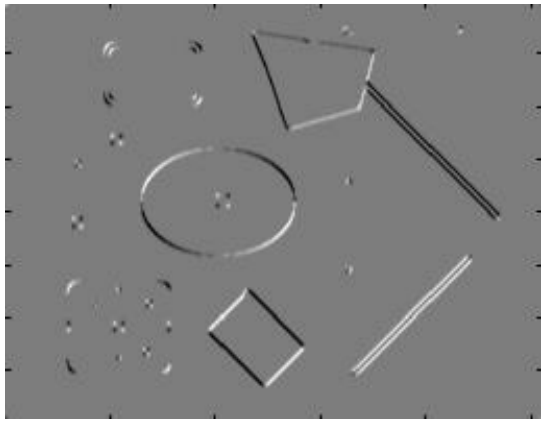
Hessian corners

- So, what exactly does this mean and what is this $w(x, y)$?
- First, note that the matrix H looks very similar to the image Hessian [Beaudet78]

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

and $\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2$



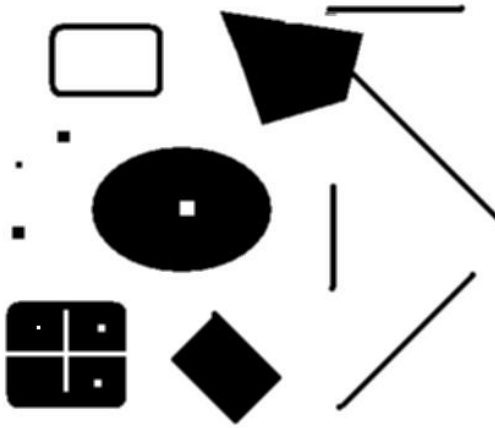


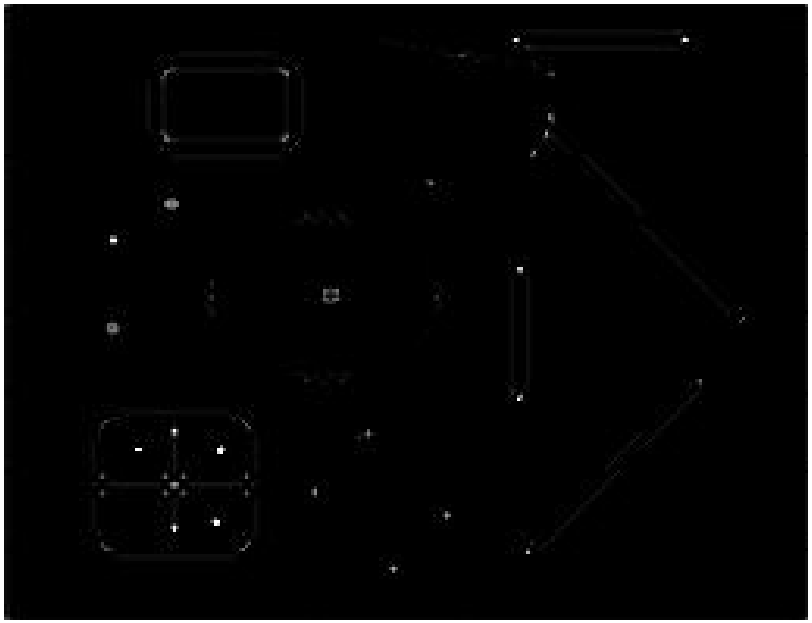
Hessian corners

- So, what exactly does this mean and what is this $w(x, y)$?
- First, note that the matrix H looks very similar to the image Hessian [Beaudet78]

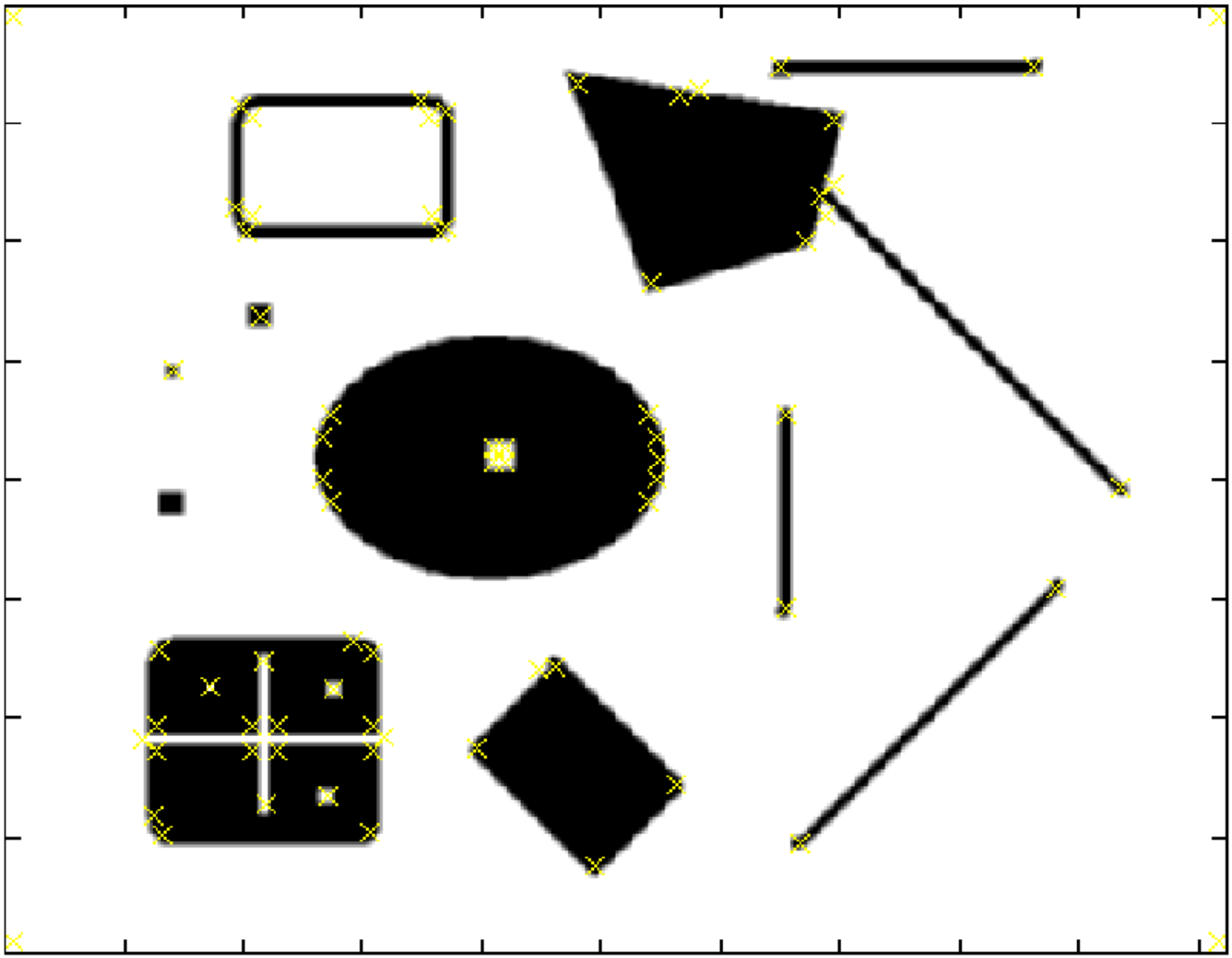
$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

and $\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2$





Hessian corners



Harris corners

- So, how is the Harris detector different?
- The Harris matrix (auto-correlation matrix) is the second moment matrix!

$$H = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- What does this mean??? (example)

Lets formulate this mathematically

1. Given some image $I \in [0,1]^{m \times n}$
2. Define the Gaussian kernel G_{σ_1} and compute the image derivatives as:

$$\left\{ I_x = I * \frac{\partial G_{\sigma_1}}{\partial x}, \quad I_y = I * \frac{\partial G_{\sigma_1}}{\partial y} \right\} \in \mathbb{R}^{m \times n}$$

3. Compute the three images

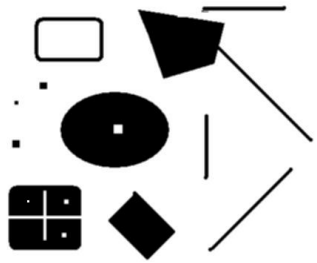
$$\left\{ I_x^2, \ I_y^2, \text{ and } I_x I_y \right\} \in \mathbb{R}^{m \times n}$$

   by performing the **Hadamard product**
4. Filter I_x^2, I_y^2, and $I_x I_y$ by convolving them with some G_{σ_2} where in general $\sigma_2 > \sigma_1$ (this is our window function convolved with H)
5. Compute the corner strength function at each (x, y) (Harris used this but there are others):

$$G_{\sigma_2}(I_x^2)G_{\sigma_2}(I_y^2) - [G_{\sigma_2}(I_x I_y)]^2 - \alpha[G_{\sigma_2}(I_x^2) + G_{\sigma_2}(I_y^2)]^2, \quad \alpha \in [0.04, 0.08]$$

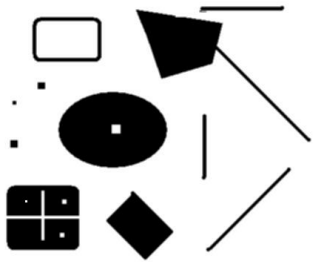
Example of the Harris corner process

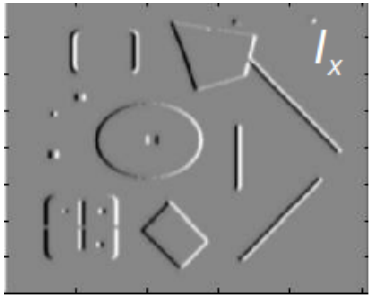
1. Consider our test image
   $I \in [0,1]^{m \times n}$

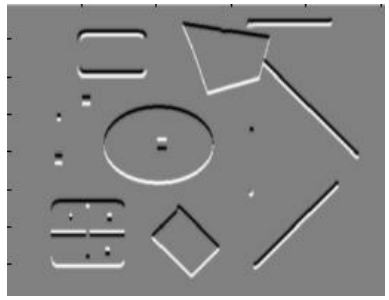


Example of the Harris corner process

1. Consider our test image $I \in [0,1]^{m \times n}$
2. Compute the image derivatives I_x, I_y

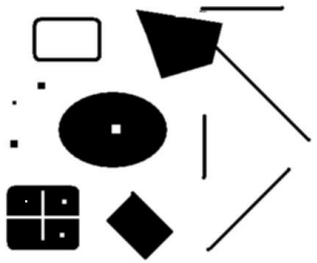


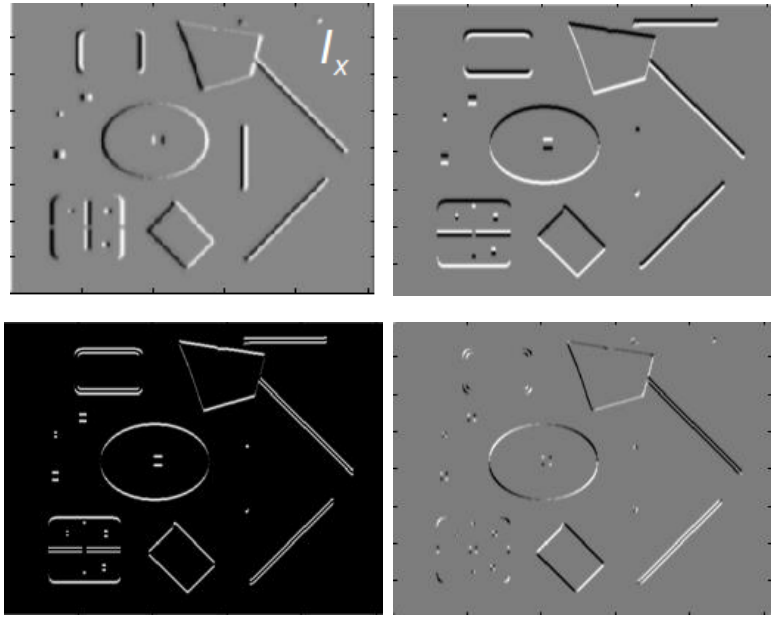




Example of the Harris corner process

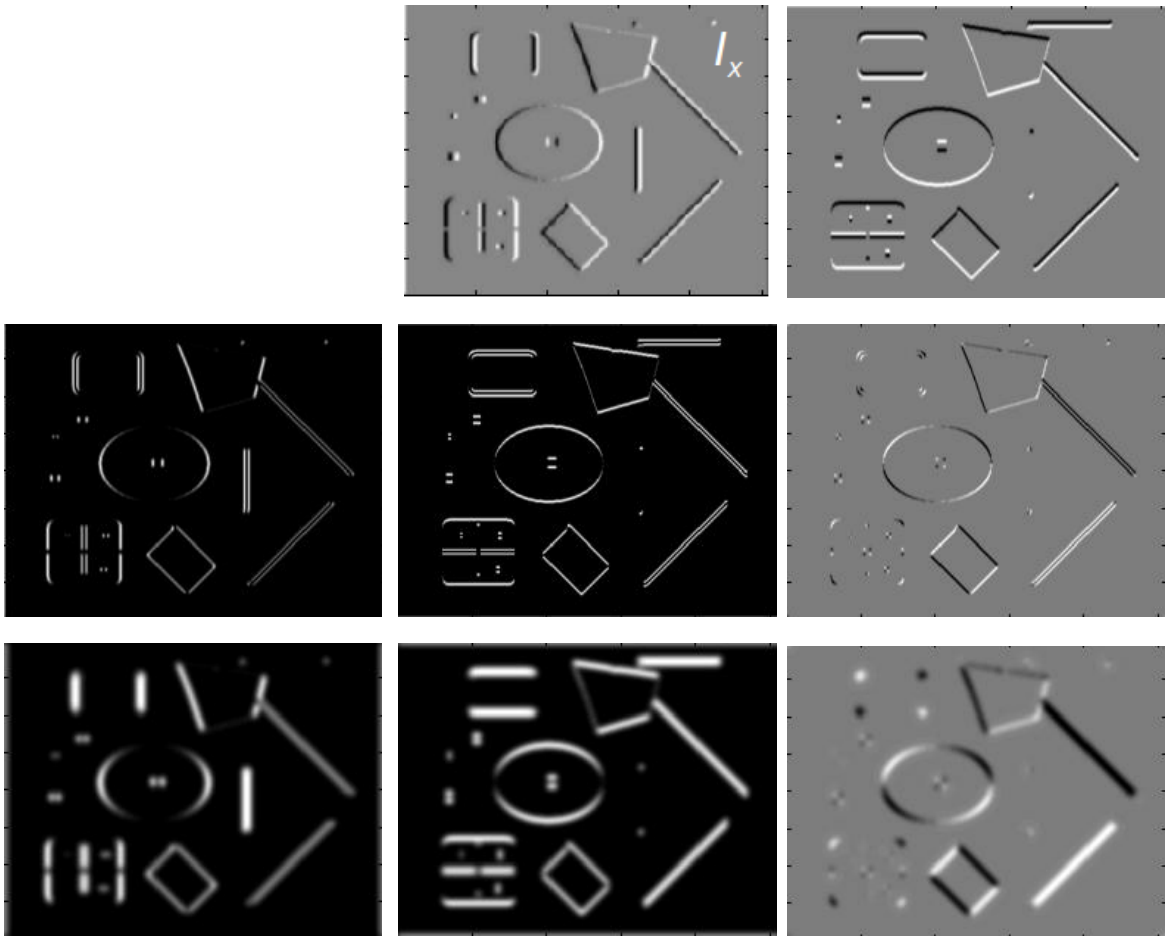
1. Consider our test image $I \in [0,1]^{m \times n}$
2. Compute the image derivatives I_x, I_y
3. Square them and compute $I_x I_y$





Example of the Harris corner process

1. Consider our test image $I \in [0,1]^{m \times n}$
2. Compute the image derivatives I_x, I_y
3. Square them and compute $I_x I_y$
4. Gauss filter I_x^2, I_y^2, and $I_x I_y$

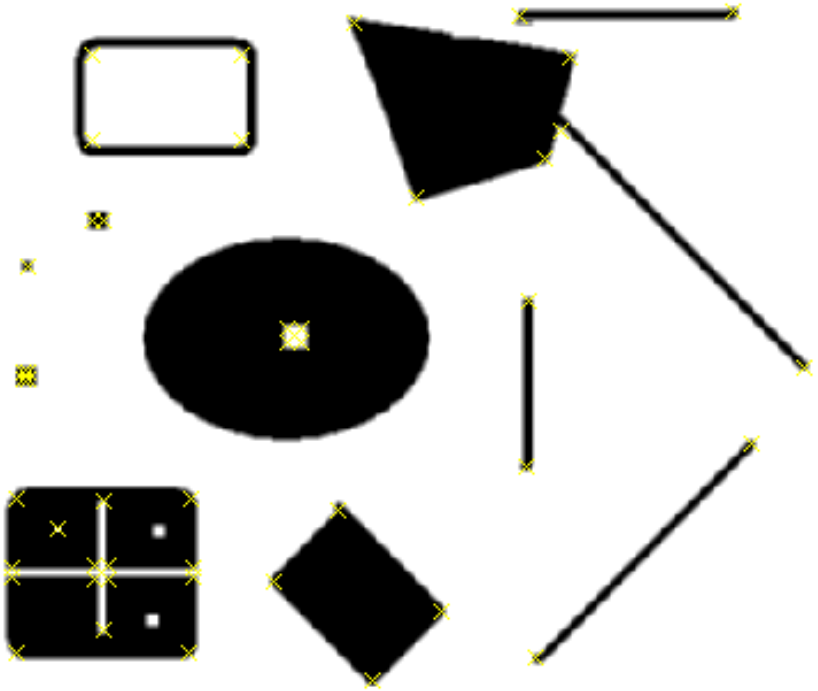


Example of the Harris corner process

1. Consider our test image $I \in [0,1]^{m \times n}$
2. Compute the image derivatives I_x, I_y
3. Square them and compute $I_x I_y$
4. Gauss filter I_x^2, I_y^2, and $I_x I_y$
5. Compute a corner strength function

$$G_{\sigma_2}(I_x^2)G_{\sigma_2}(I_y^2) - [G_{\sigma_2}(I_x I_y)]^2 - \alpha[G_{\sigma_2}(I_x^2) + G_{\sigma_2}(I_y^2)]^2$$

Example of the Harris corner process

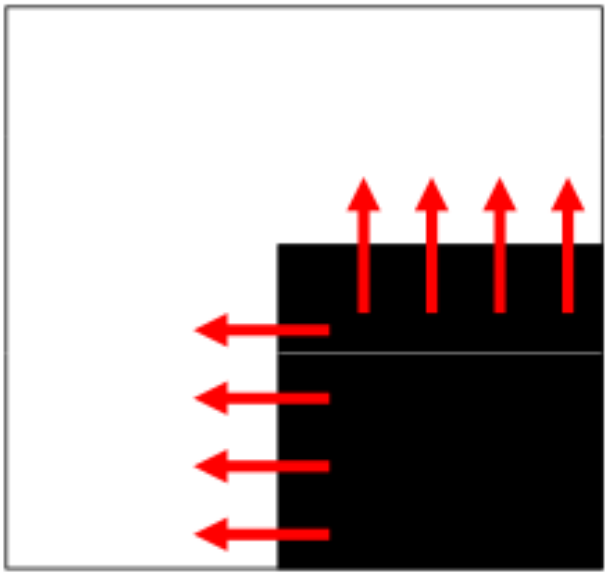


1. Consider our test image $I \in [0,1]^{m \times n}$
2. Compute the image derivatives I_x, I_y
3. Square them and compute $I_x I_y$
4. Gauss filter I_x^2, I_y^2, and $I_x I_y$
5. Compute a corner strength function

$$G_{\sigma_2}(I_x^2)G_{\sigma_2}(I_y^2) - [G_{\sigma_2}(I_x I_y)]^2 - \alpha[G_{\sigma_2}(I_x^2) + G_{\sigma_2}(I_y^2)]^2$$

What does this matrix reveal?

- Consider a corner that happens to exactly align with the image axis



What does this matrix reveal?

- Consider a corner that happens to exactly align with the image axis
  - In this case, there is no $I_x I_y$ component! (why???)

$$H = w * \begin{bmatrix} I_x^2 & 0 \\ 0 & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

- Dominant gradient directions align with the x or y axis
- If λ_i is close to 0, then this is **not** a corner $\rightarrow$ look for locations where both λ_i are large

What does this matrix reveal?

- Consider a corner that happens to exactly align with the image axis
  - In this case, there is no $I_x I_y$ component! (why???)

$$H = w * \begin{bmatrix} I_x^2 & 0 \\ 0 & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

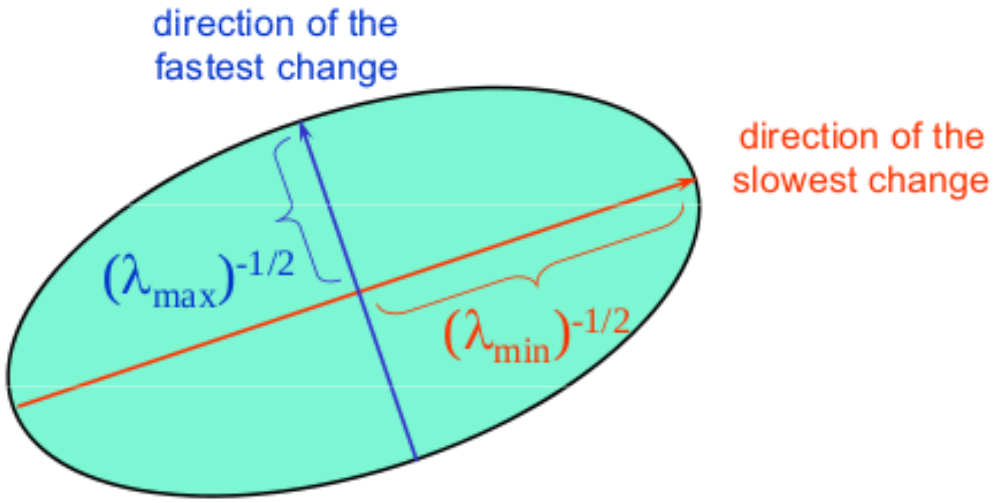
- Dominant gradient directions align with the x or y axis
- If λ_i is close to 0, then this is **not** a corner $\rightarrow$ look for locations where both λ_i are large
- What if the corner is NOT aligned with the image axis?

General case

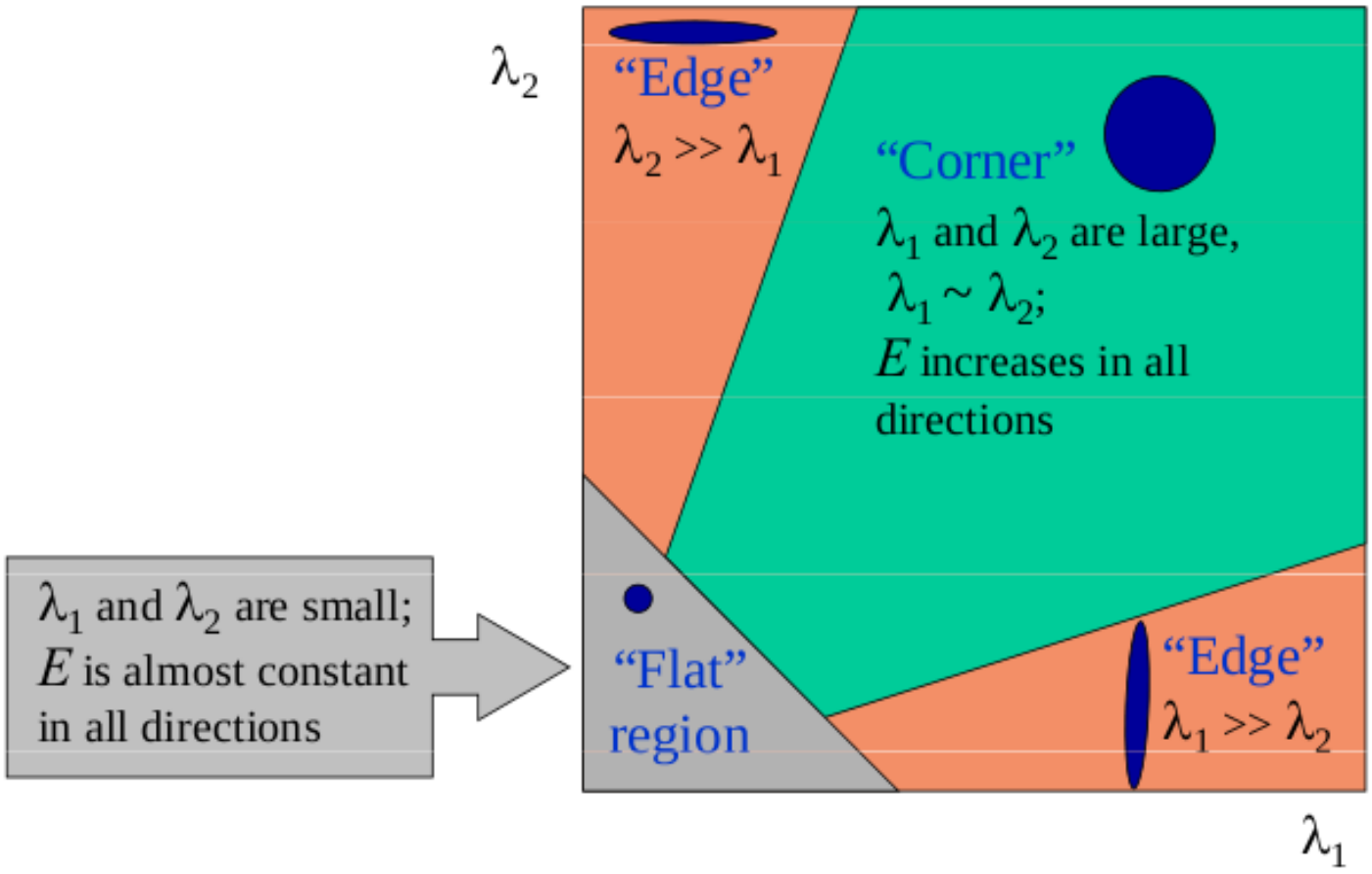
- Note that H is symmetric
  - $\Longrightarrow$ we can visualize H as an ellipse whose axis lengths are determined by the eigenvalues and whose eigenvectors define the principle axis

$$H = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

where R is a rotation matrix (determines the orientation of the ellipse)



How do we interpret the eigenvalues of H?



Corner response function(s)

1. Harris and Stephens:

$$C(H) = \det(H) - \alpha \text{trace}(A)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

$(\alpha \in [0.04 - 0.06])$

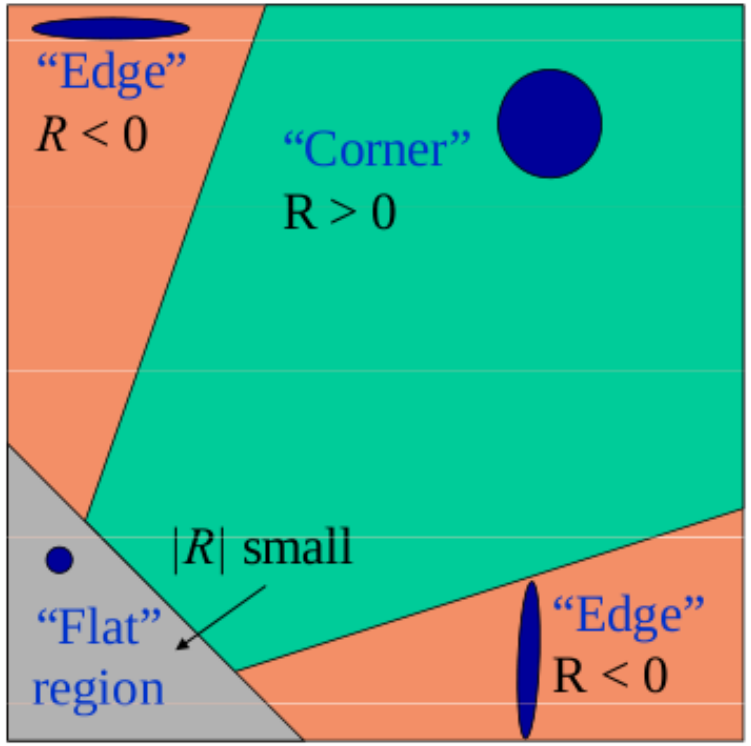
2. Triggs:

$$C(H) = \lambda_1 - \alpha\lambda_2$$

$(\alpha = 0.05)$

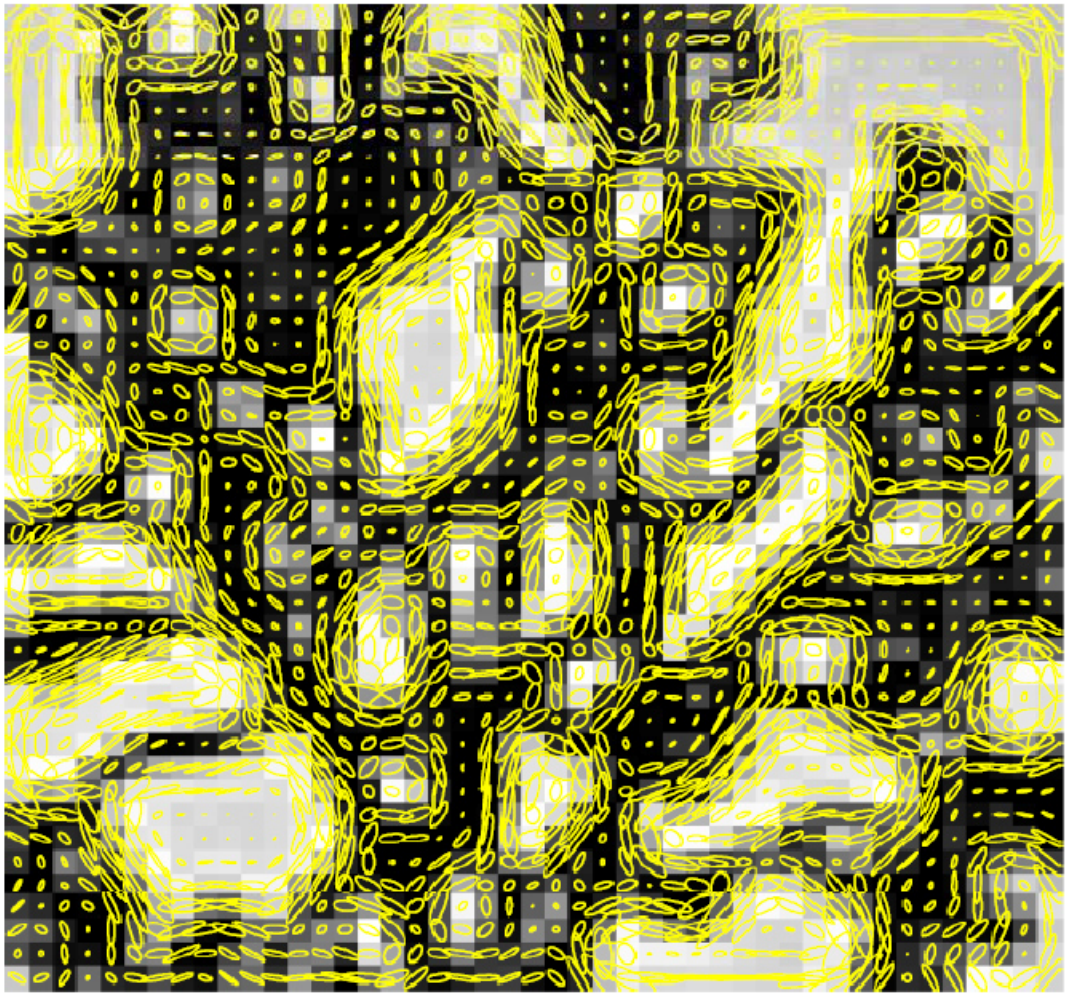
3. Brown, Szeliski, and Winder:

$$C(H) = \frac{\det(H)}{\text{trace}(A)} = \frac{\lambda_1\lambda_2}{\lambda_1 + \lambda_2}$$



Example

Example



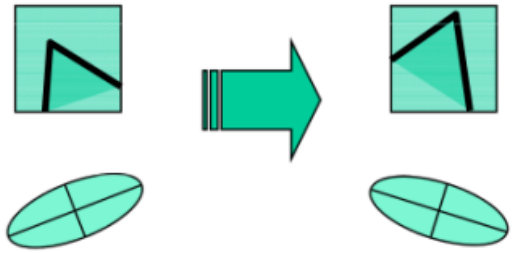
Harris detector Alg. summary

Harris Algorithm:

1. Compute I_x and I_y by convolving the original image with a derivative of Gaussian kernel
2. Compute the three images corresponding to I_x^2, I_y^2, and $I_x I_y$
3. Convolve each of these images with a larger Gaussian
4. Compute the corner strength according to one of the corner strength functions
5. Find the maxima above a threshold and report as detected features
6. (optionally) apply non-maximum suppression to ensure feature is a local maxima

What are the invariance properties of the Harris corner detector?

- Rotation invariance

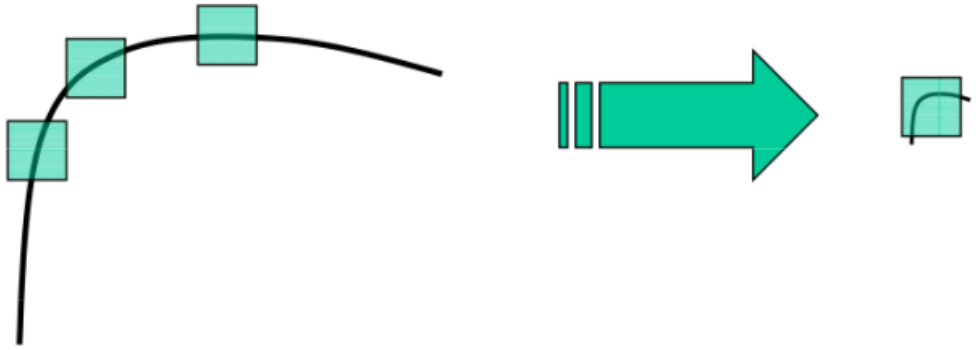


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

- The corner response $C(H)$ is invariant to image rotation!

What are the invariance properties of the Harris corner detector?

- Scale invariance



All points will be classified as edges

Corner !

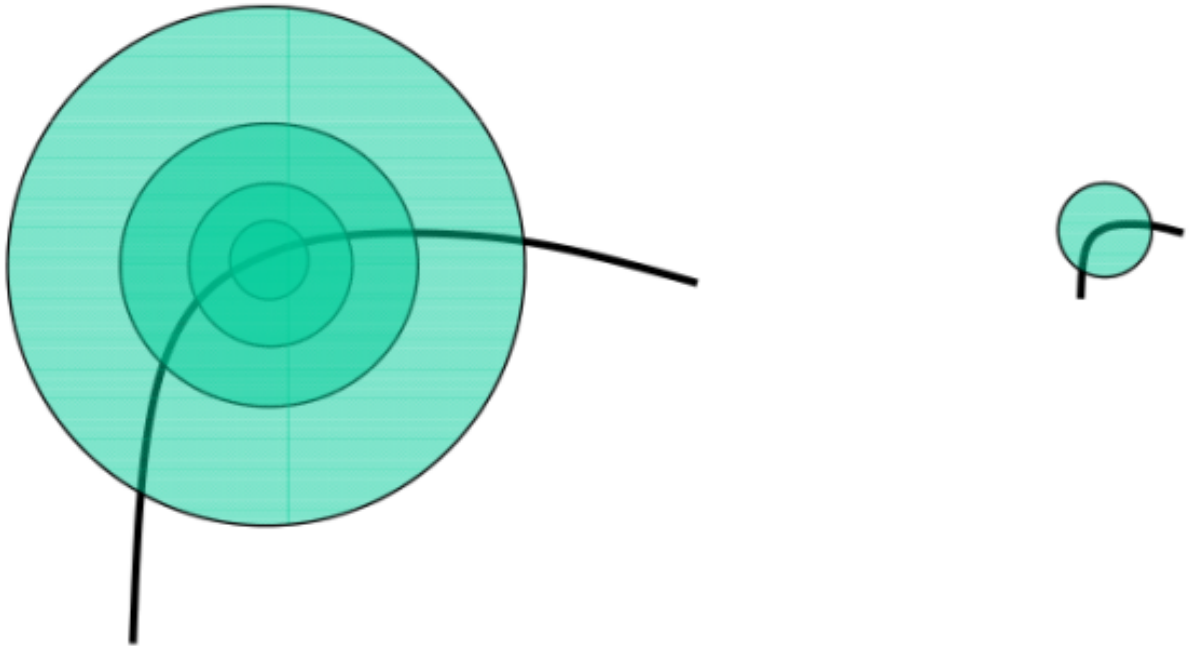
- The corner response $C(H)$ is NOT invariant to image scale!

Scale invariance!

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

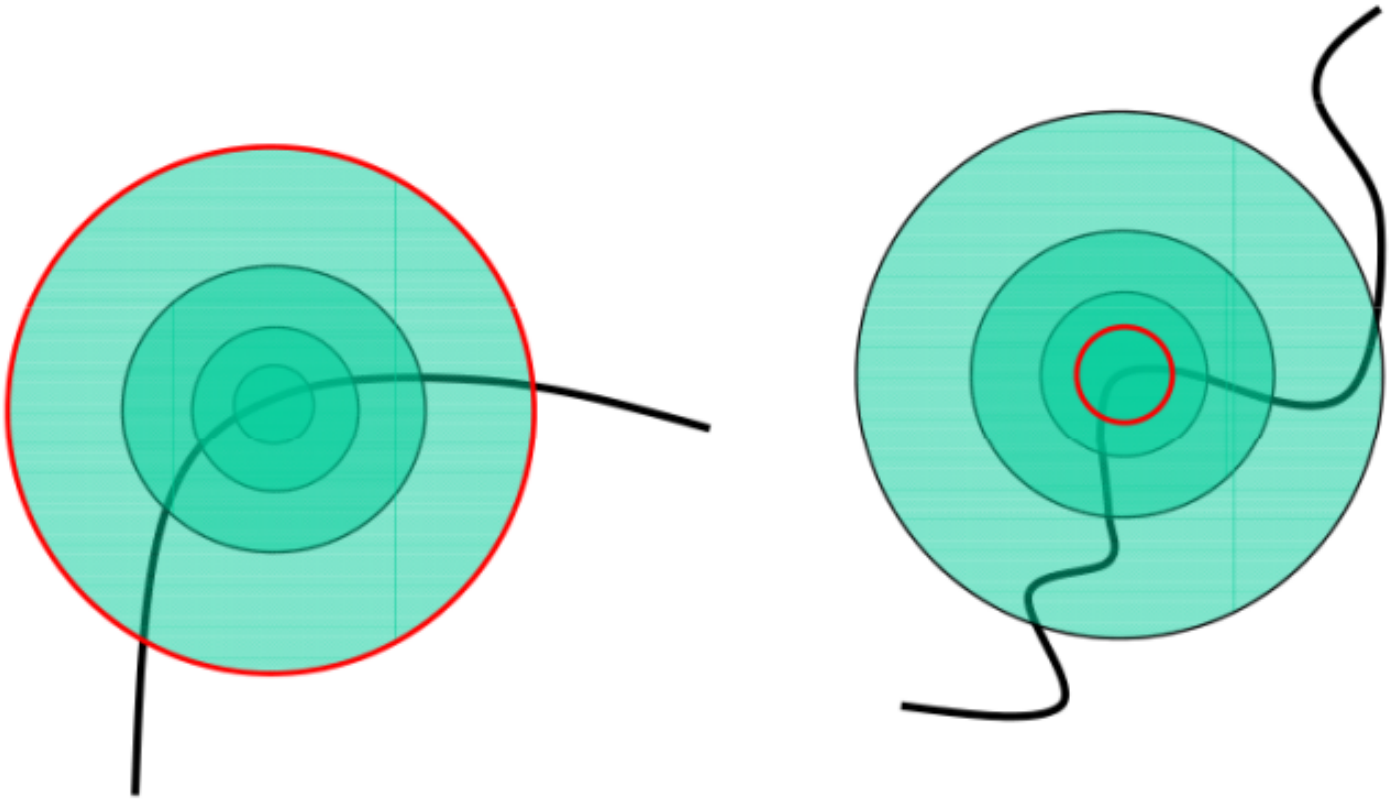
Scale invariance!

- How can we detect **scale invariant** feature points?
- Consider regions of different size patches around the potential feature point
- The regions of corresponding sizes will look the same in both images



Scale invariance!

- Issue: how do we choose the correct patch size in each image independently?



Scale invariance!

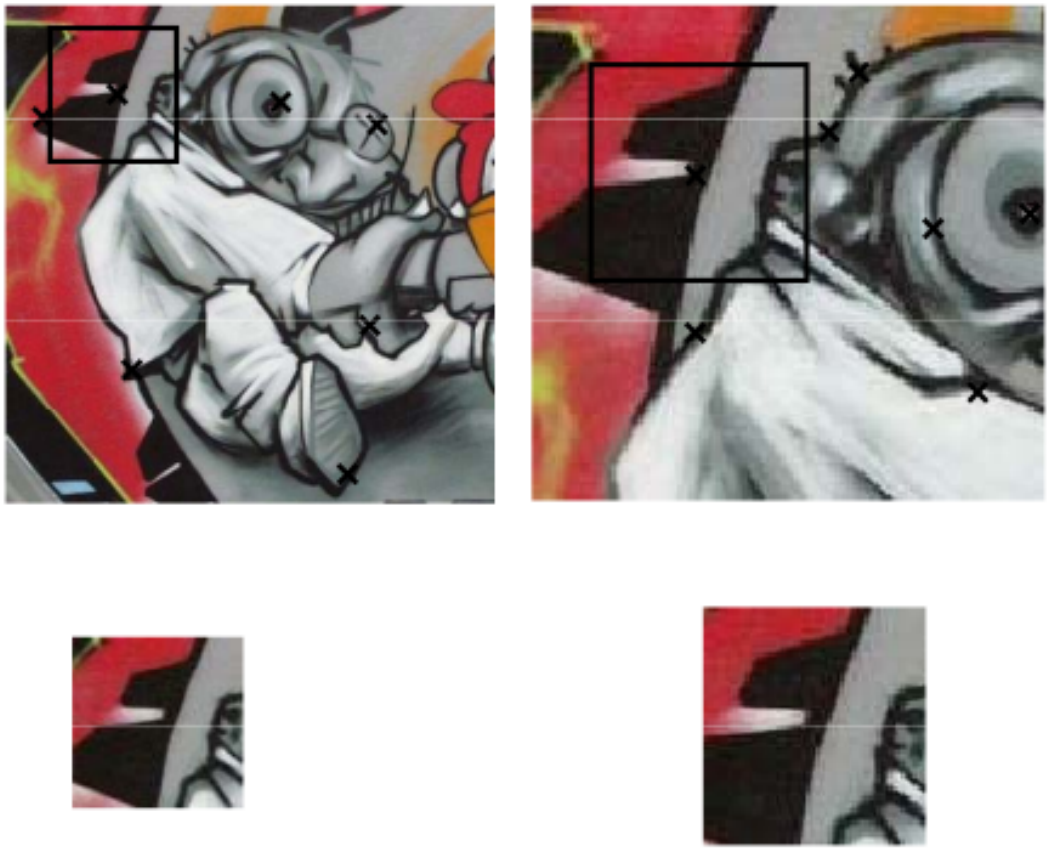
- Solution 1: Exhaustive search across all scales

Scale invariance!

- Solution 1: Exhaustive search across all scales

Scale invariance!

- Solution 1: Exhaustive search across all scales



Scale invariance!

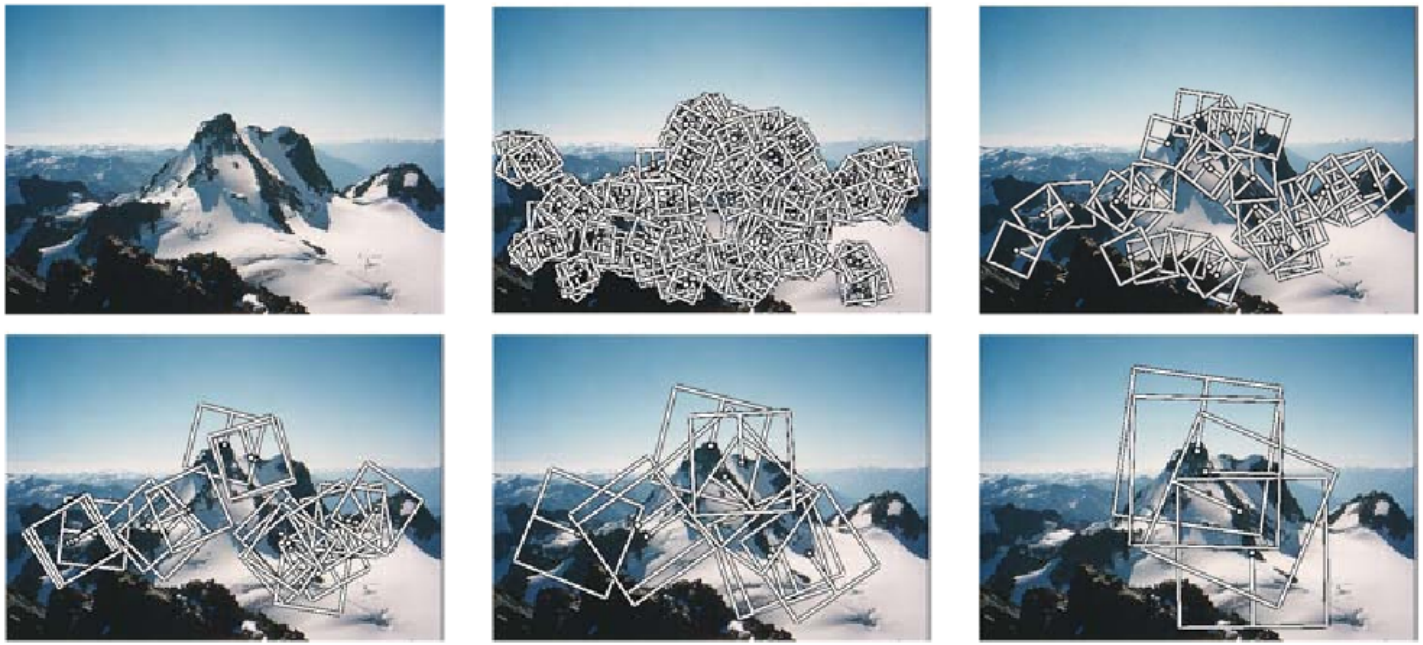
- Solution 1: Exhaustive search across all scales

Scale invariance!

- Solution 1: Exhaustive search across all scales
- Extract a patch from each image individually (your feature list now becomes FL$= [x, y, \text{scale}]$)

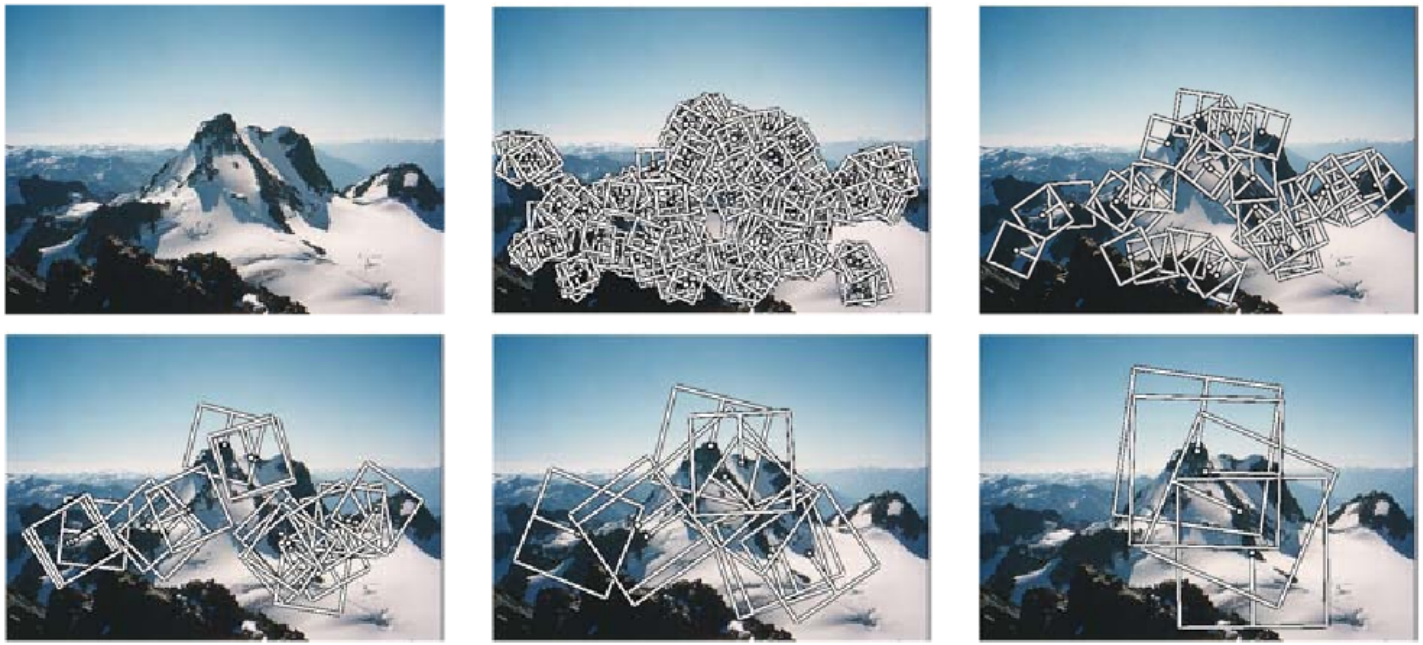
Scale invariance!

- Solution 1: Exhaustive search across all scales



Scale invariance!

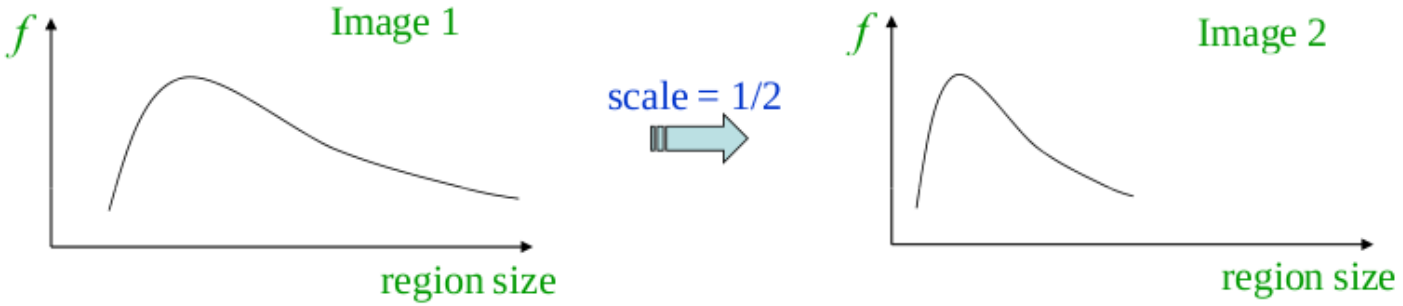
- Solution 1: Exhaustive search across all scales



- Issue: we have to search across all possible scales! We would prefer to select the scale automatically.

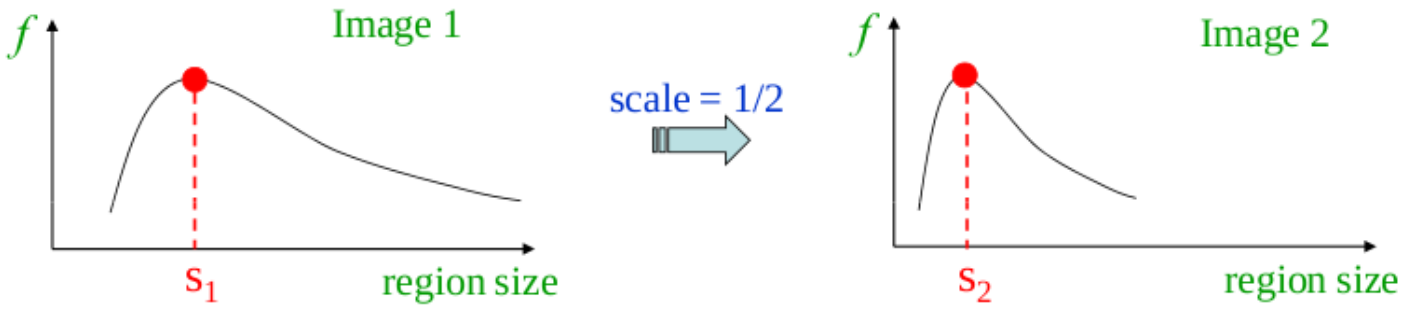
Scale invariance!

- Solution 2: Automatic scale selection
  - Design a function on the patch which is scale invariant (the same for corresponding regions, even at different scales)
    - A good example of such a function is the average intensity. For corresponding regions (even at different scales) this will remain the same.
  - For any given point in an image, we can compute this function as we vary the size of the region (patch) surrounding the interest point



Scale invariance!

- Solution 2: Automatic scale selection
  - Common approach:
    - Take a local max of this function $\Longrightarrow$ the region size for which the maximum is achieved should be invariant to image scale!
  - Important: this scale invariant region size is found in each image **independently**!



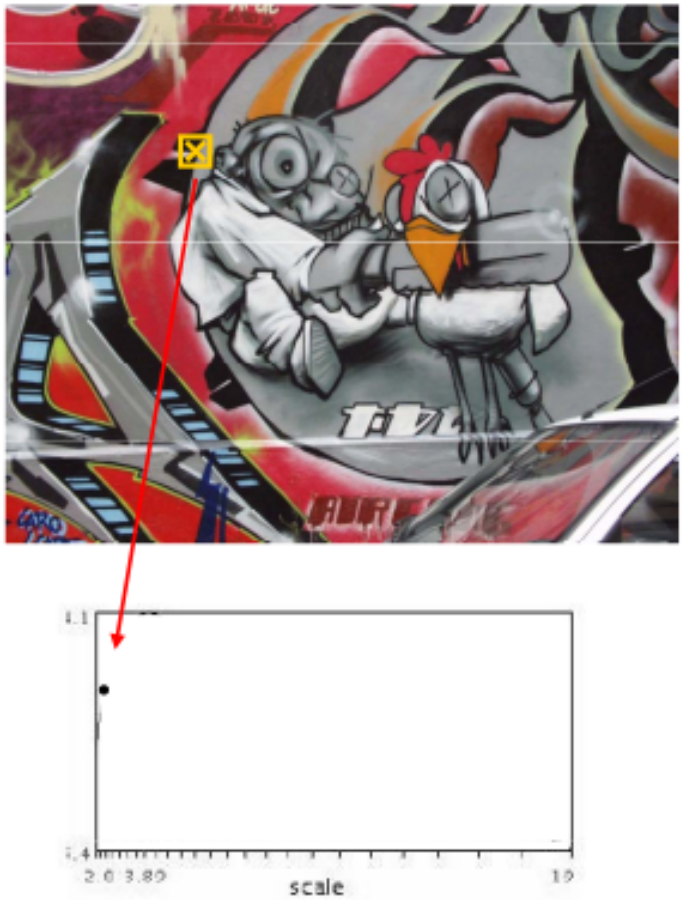
Automatic scale selection

- The same operator response if the patch contains the same image up to a scale factor!

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

Automatic scale selection

- Function response for increasing scale (scale signature)

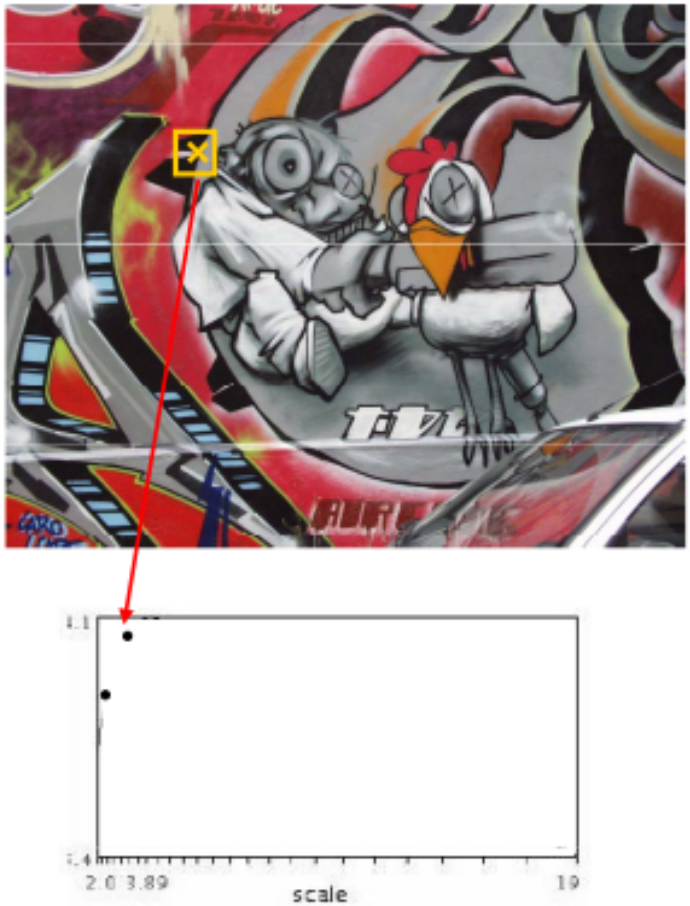


$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

Automatic scale selection

- Function response for increasing scale (scale signature)

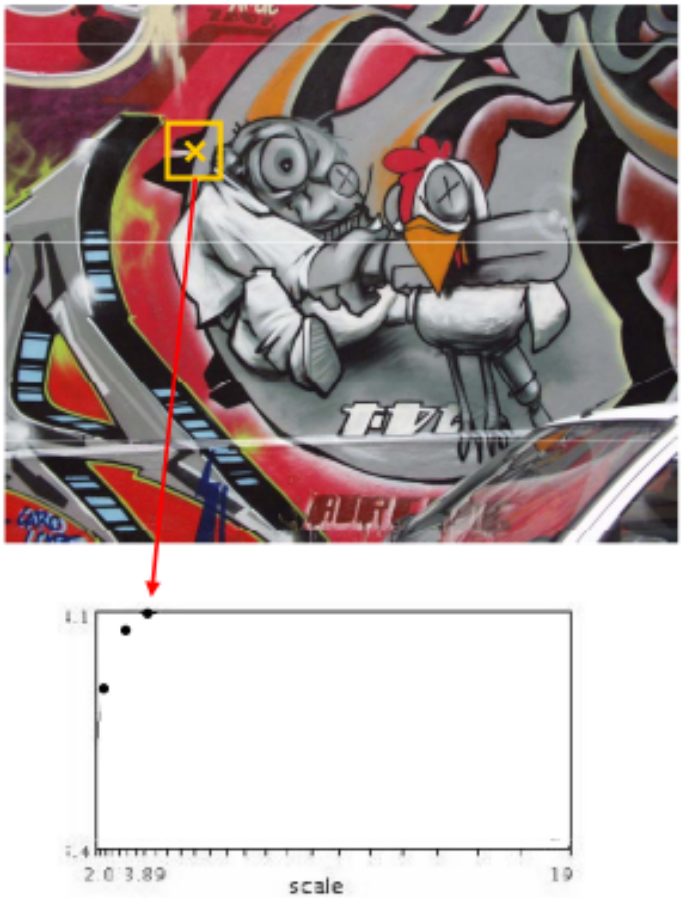


$f(I_{i_1 \ldots i_m}(x, \sigma))$

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

Automatic scale selection

- Function response for increasing scale (scale signature)

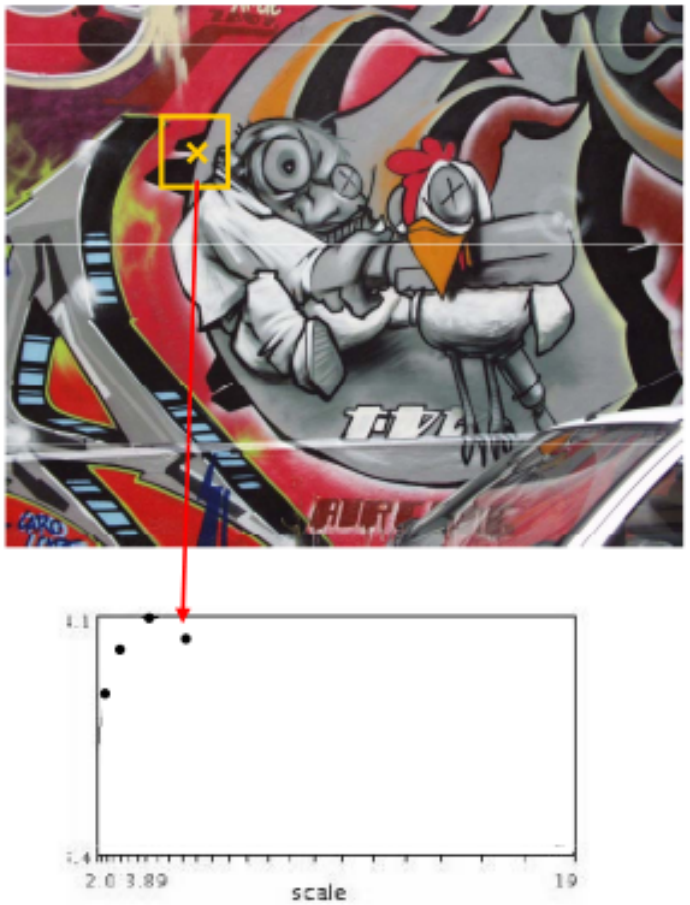


$f(I_{i_1 \ldots i_n}(x, \sigma))$

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

Automatic scale selection

- Function response for increasing scale (scale signature)

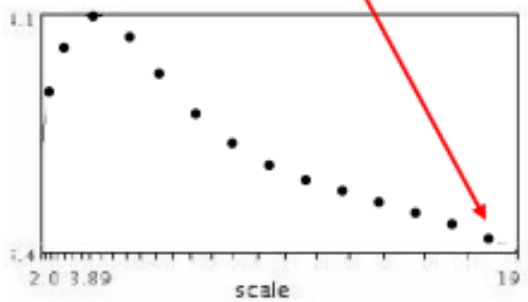


$f(I_{i_1 \ldots i_n}(x, \sigma))$

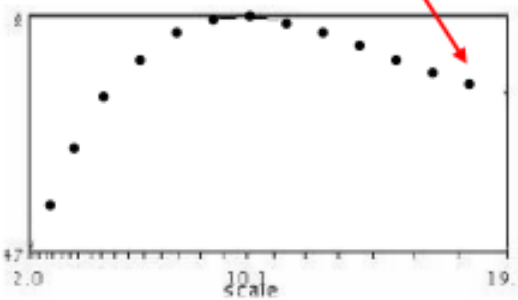
$f(I_{i_1 \ldots i_m}(x', \sigma))$

Automatic scale selection

- Function response for increasing scale (scale signature)



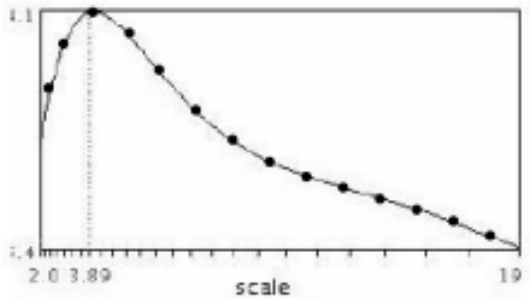
$f(I_{i_1 \ldots i_n}(x, \sigma))$



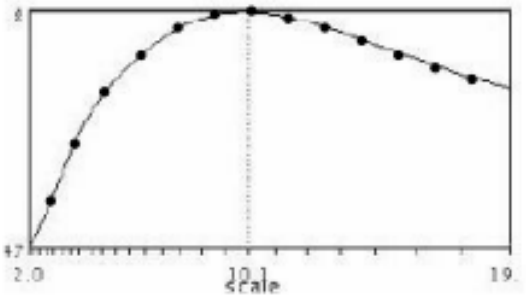
$f(I_{i_1 \ldots i_m}(x', \sigma))$

Automatic scale selection

- Function response for increasing scale (scale signature)



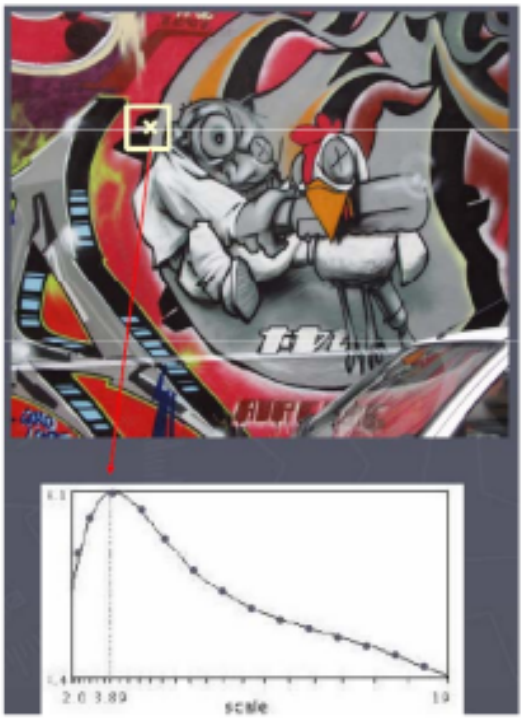
$f(I_{i_1 \ldots i_m}(x, \sigma))$

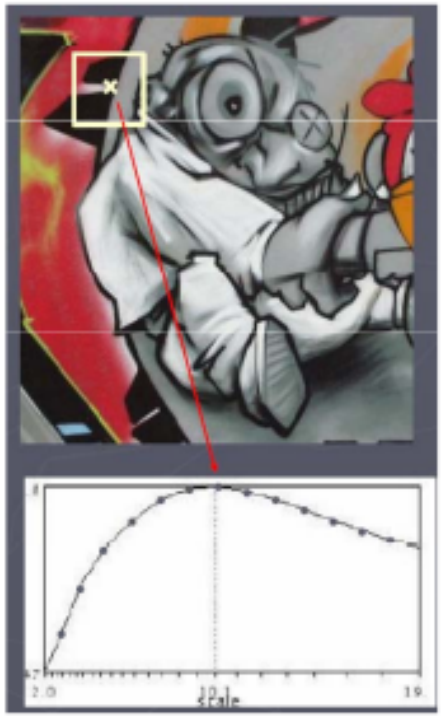


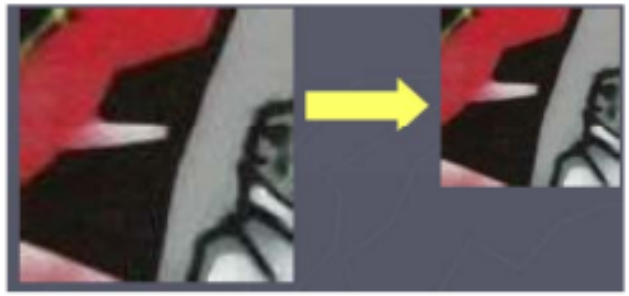
$f(I_{i_1 \ldots i_m}(x', \sigma'))$

Scale selection

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

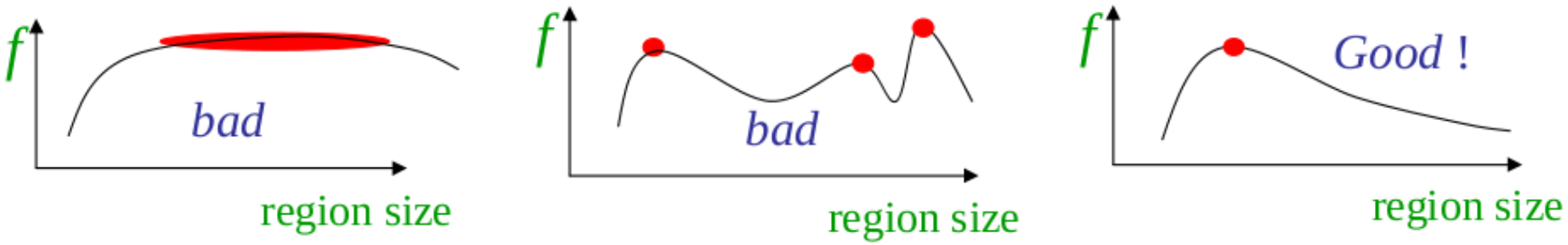






So what function do we use? i.e., what is a good signature function?

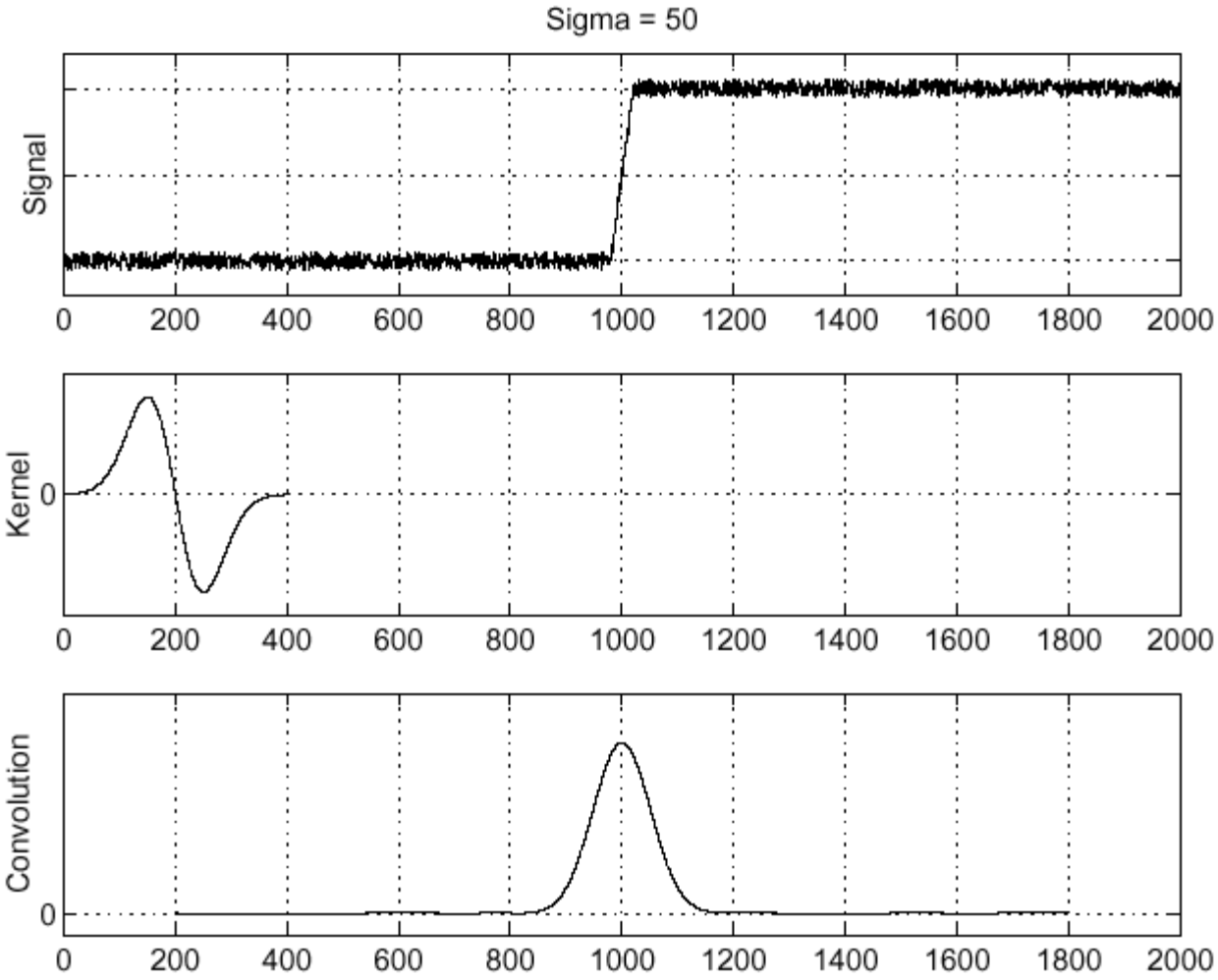
- A "good" function for scale detection is one that has a single **stable** sharp peak



- For usual images: a good function would be one that responds to contrast (sharp local intensity change)

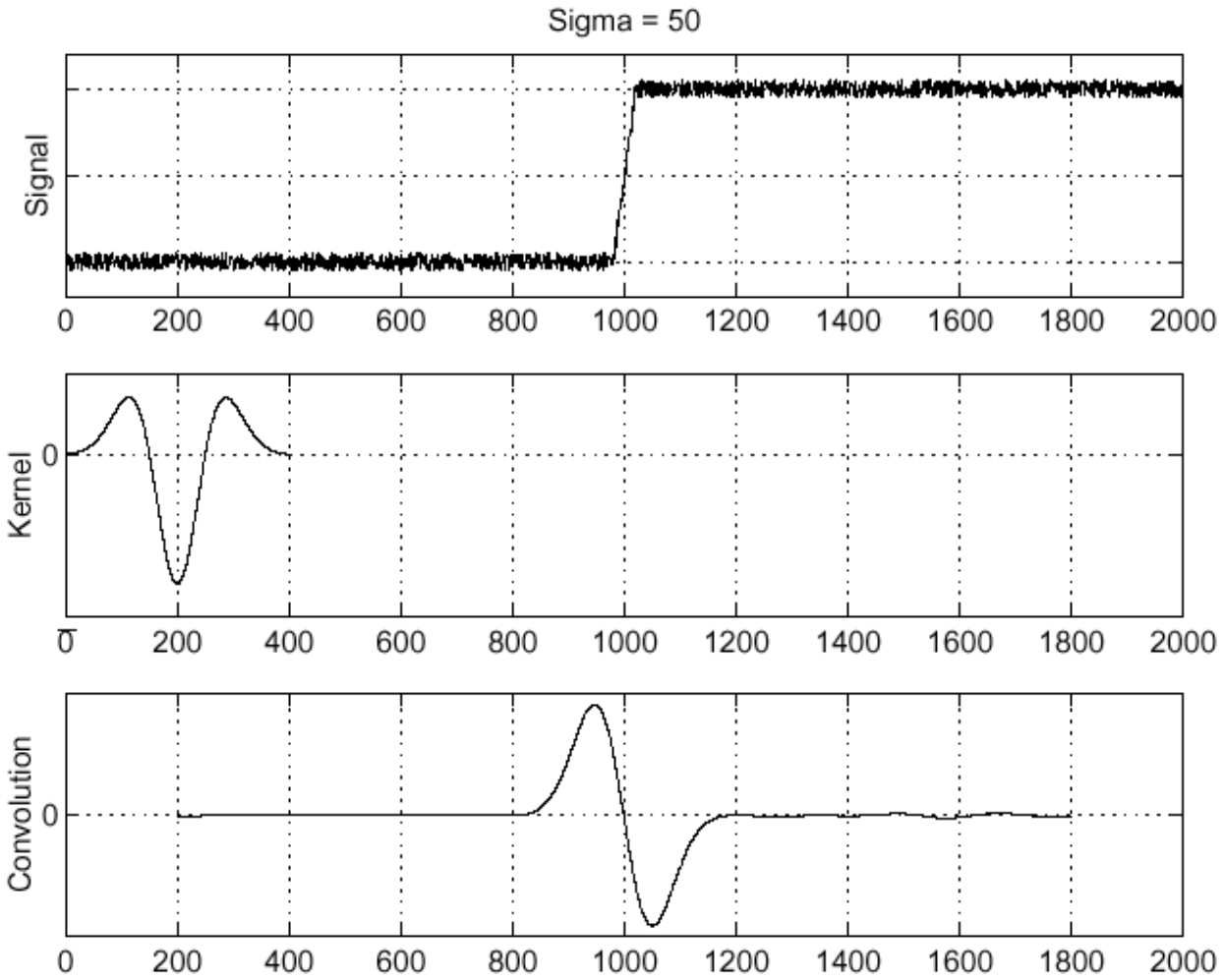
So what function do we use? $\rightarrow$ Recall edge detection!

- f
- $\frac{\partial}{\partial x} g$
- $f * \frac{\partial}{\partial x} g$



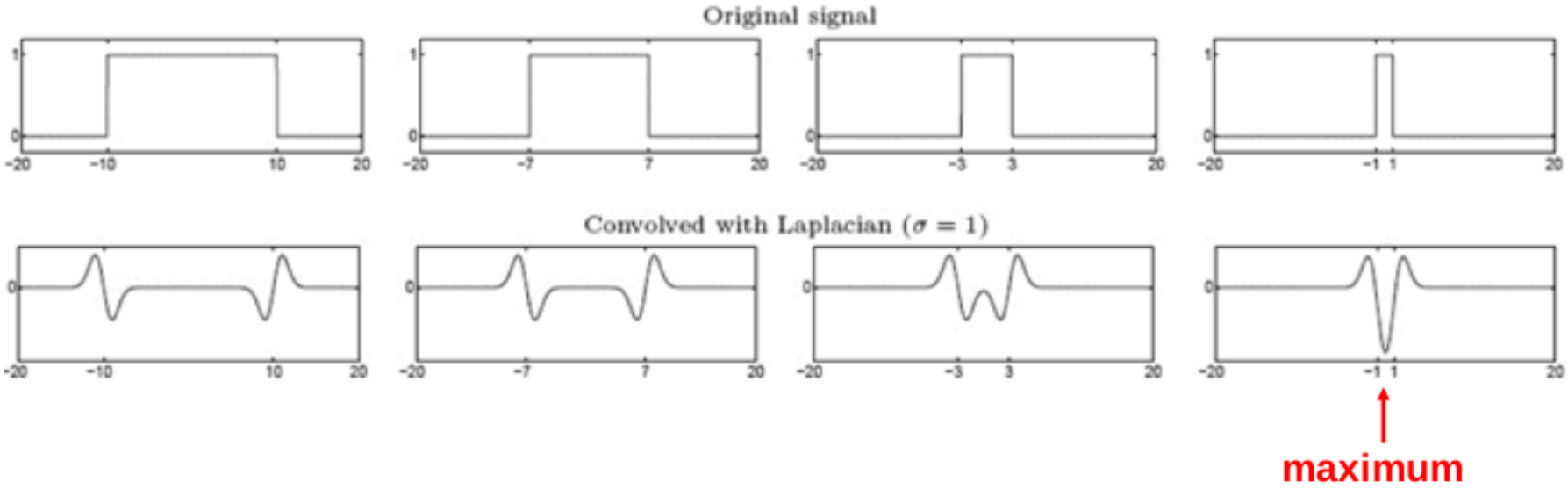
So what function do we use? $\rightarrow$ Recall edge detection!

- f
- $\frac{\partial^2}{\partial x^2} g$
- $f * \frac{\partial^2}{\partial x^2} g$



From edge detection to ripple detection

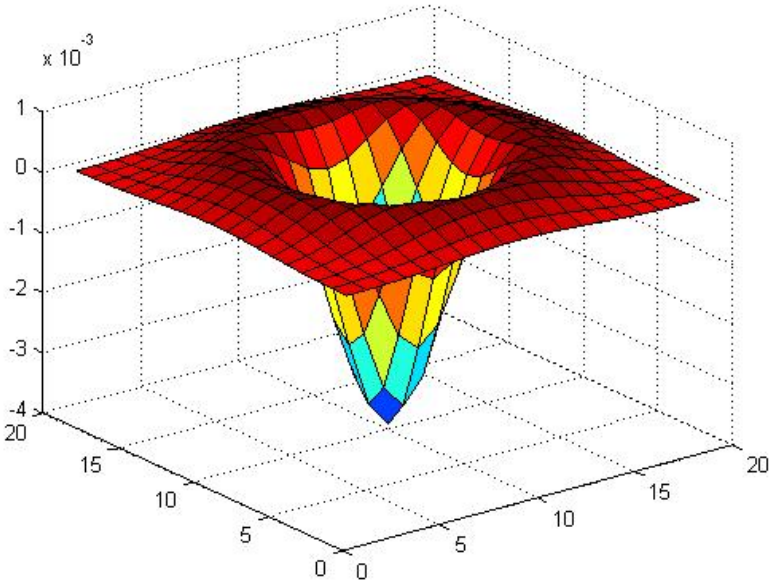
- Edge = ripple
- Blob = superposition of two ripples

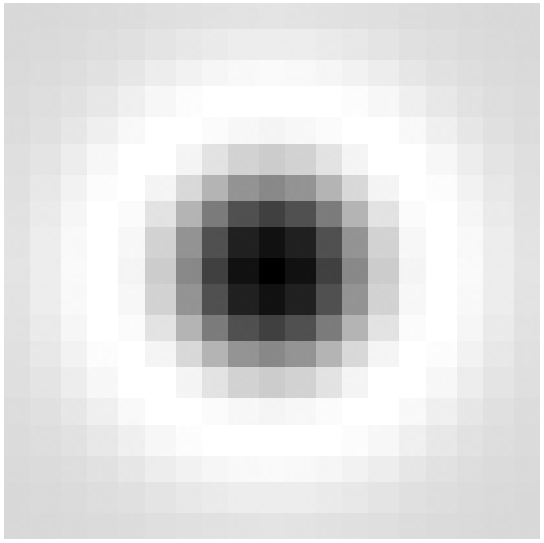


- **Spatial selection:** the **magnitude** of the Laplacian response will achieve a maximum at the center of the blob (provided the scale of the Laplacian is "matched" to the scale of the blob)

Blob detection - 2D

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

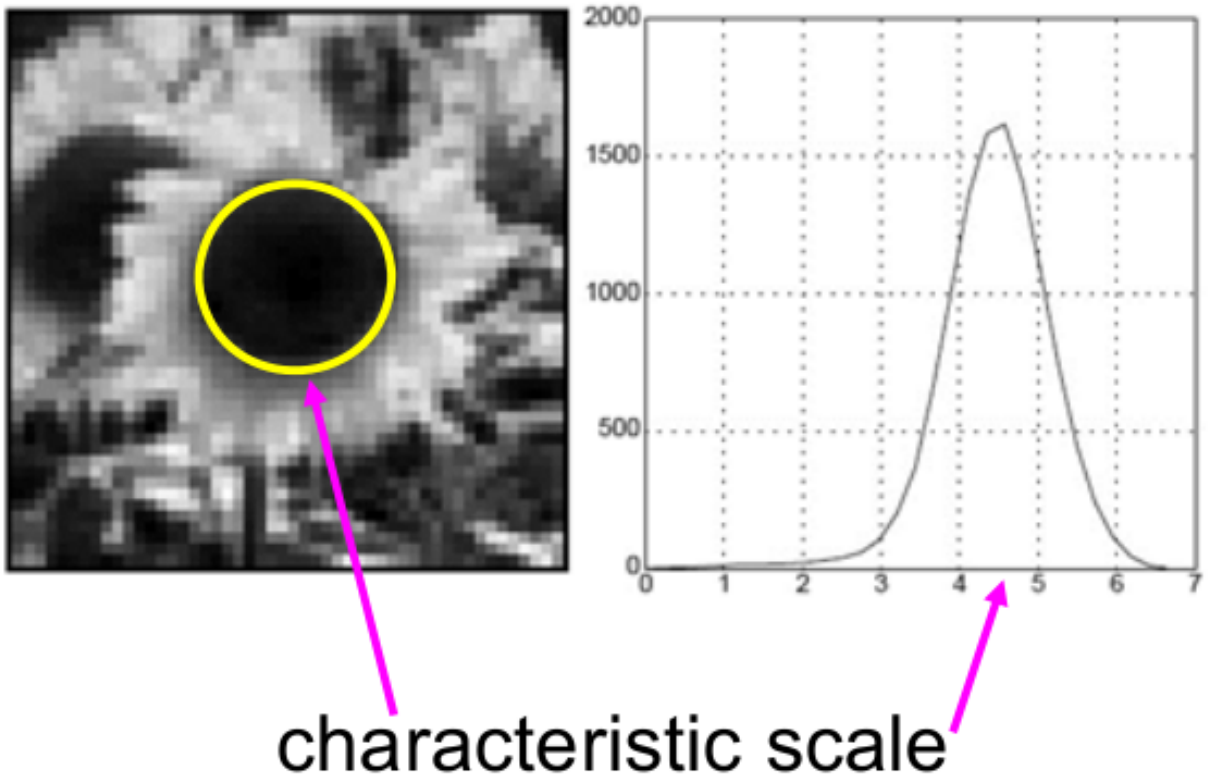




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Blob detection - 2D

- Define the *characteristic scale* as the scale that produces a peak of the Laplacian response (squared!)

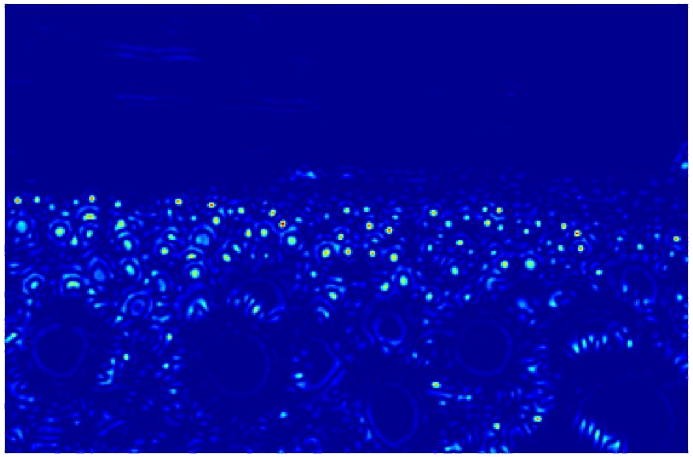


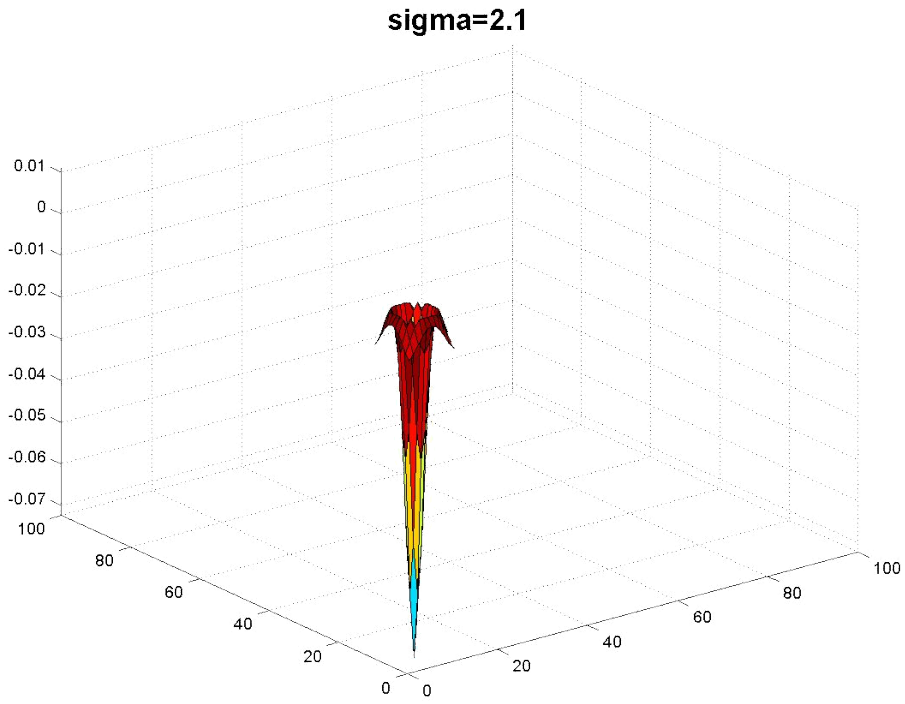
Example

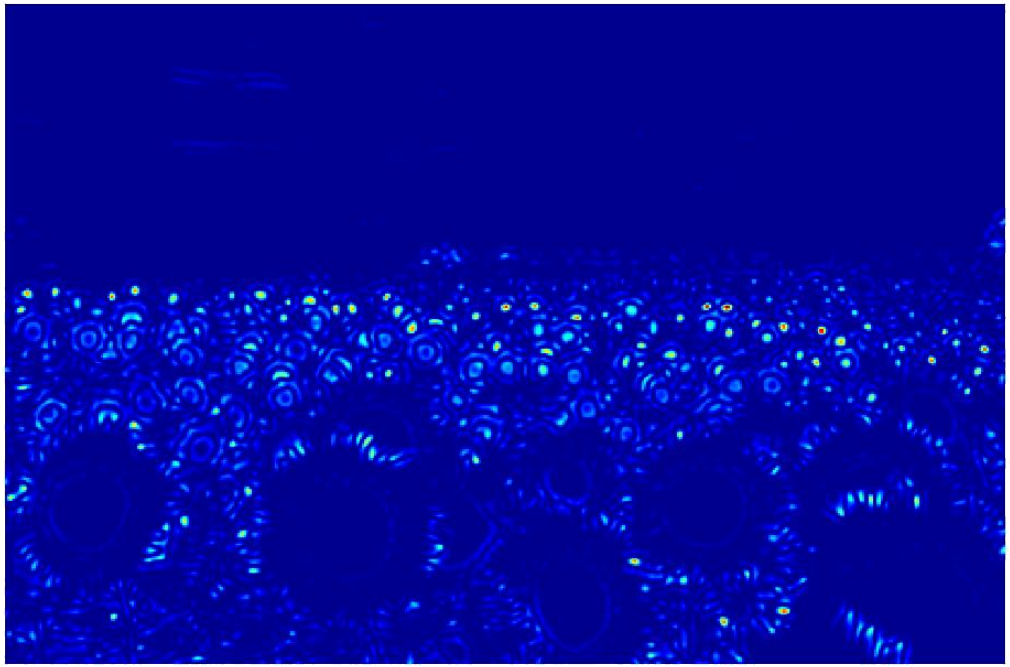
- Original image at 3/4 scale!

Example

- Original image at 3/4 scale!

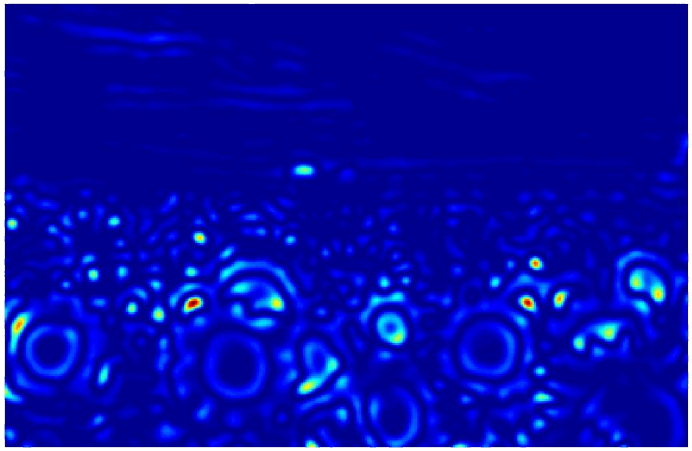


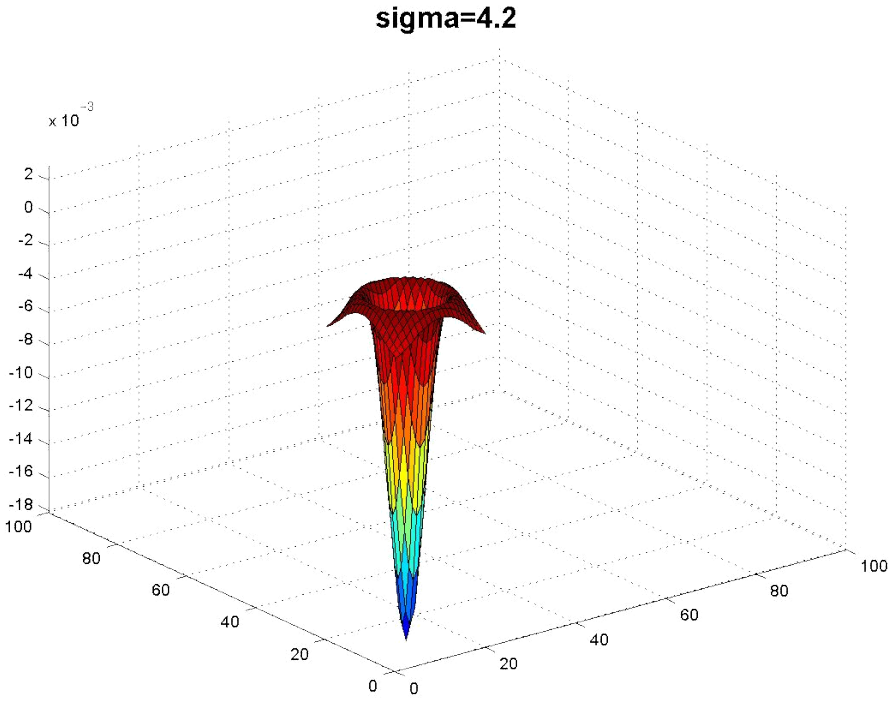


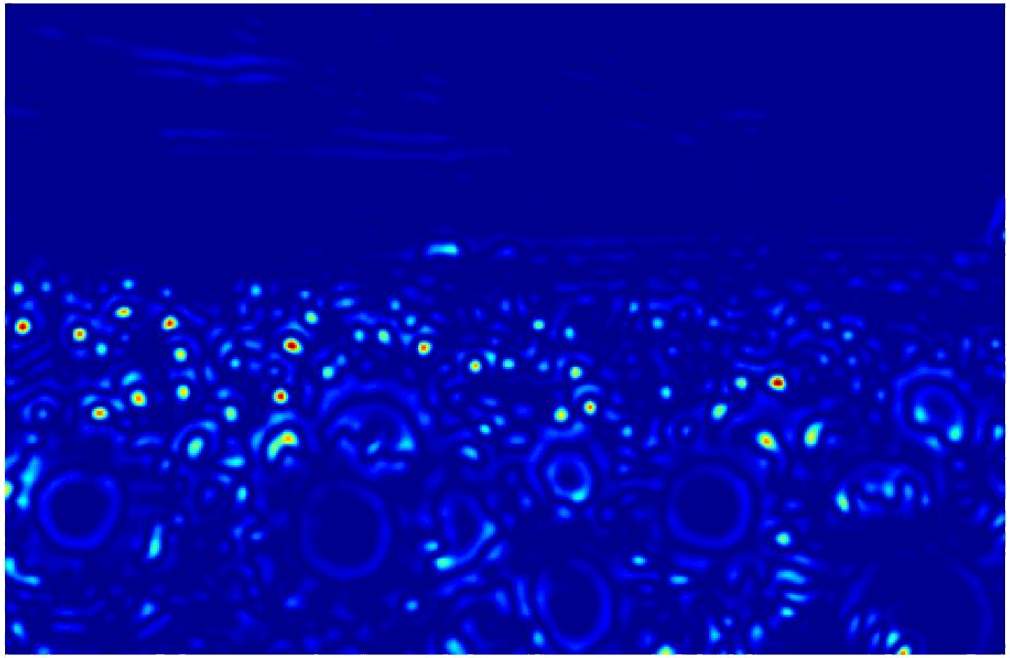


Example

- Original image at 3/4 scale!

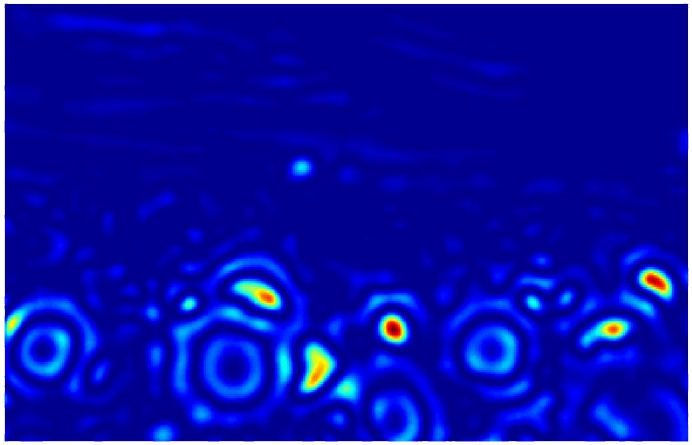


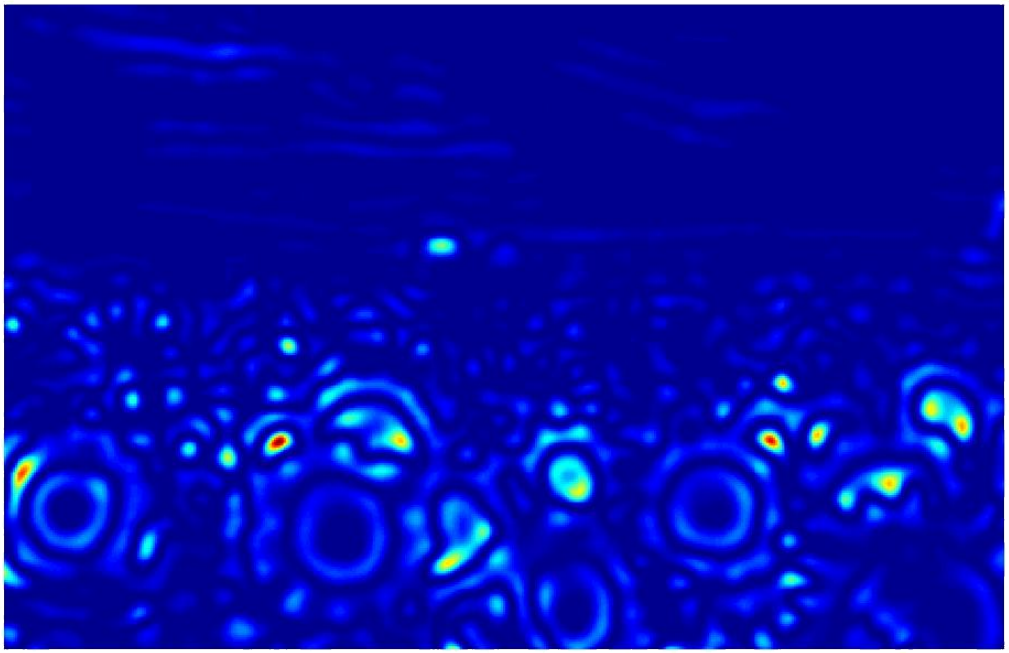


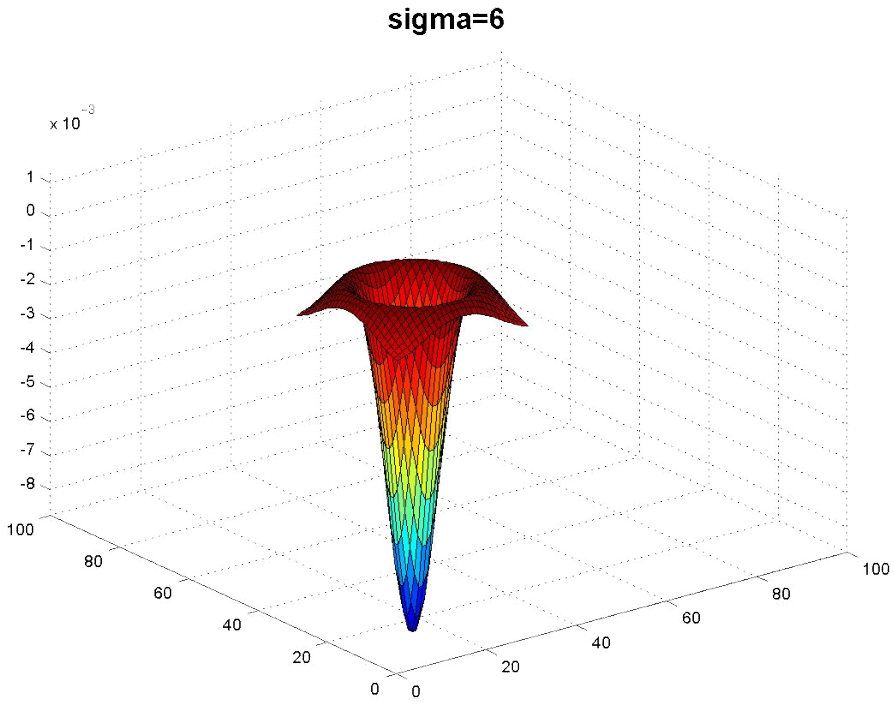


Example

- Original image at 3/4 scale!

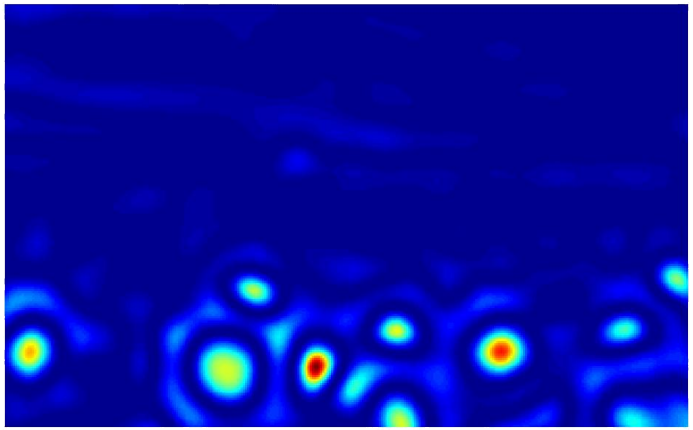


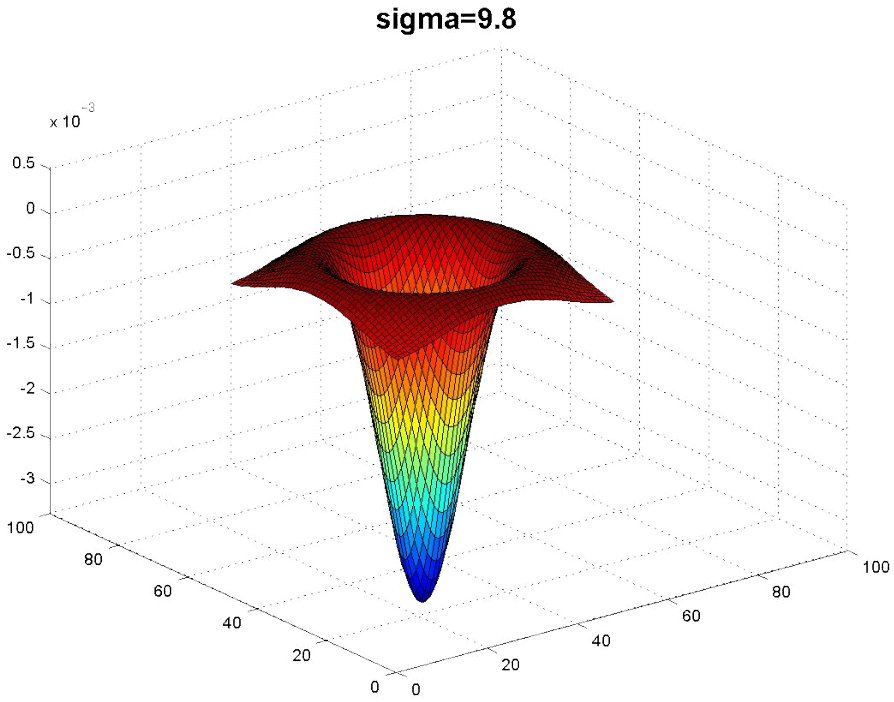


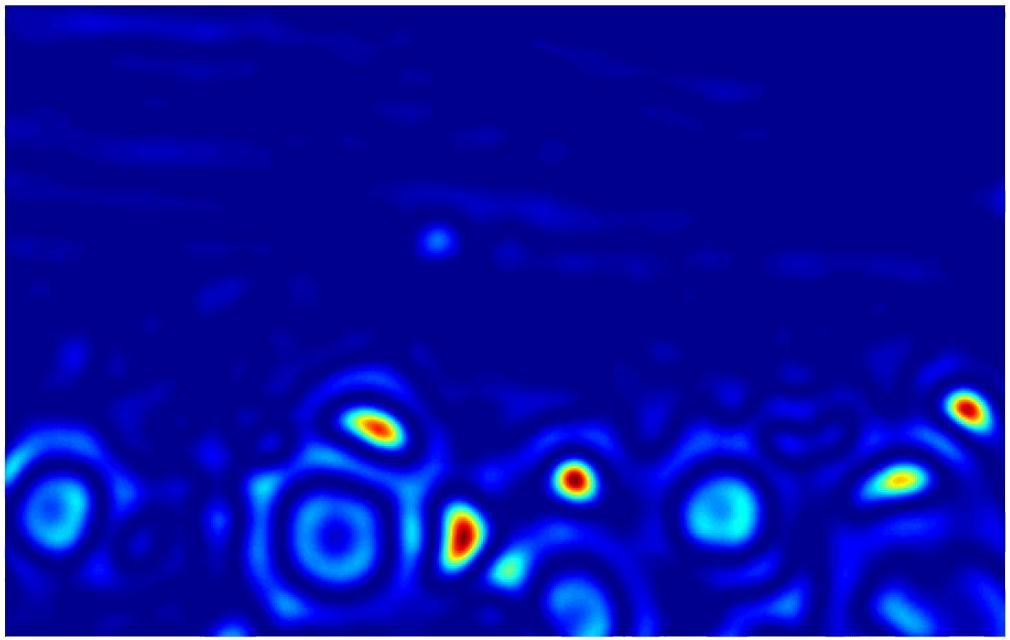


Example

- Original image at 3/4 scale!

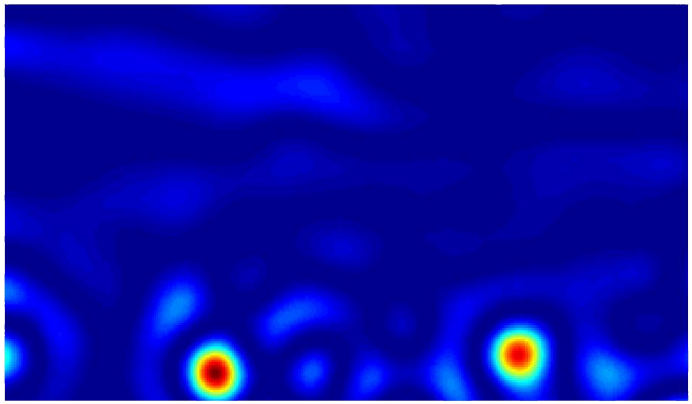


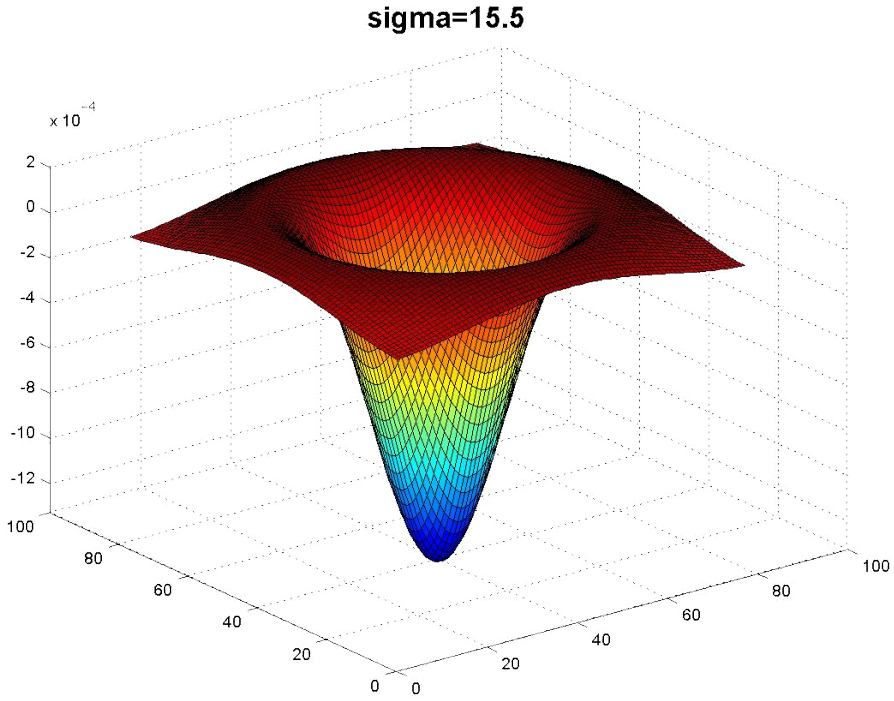


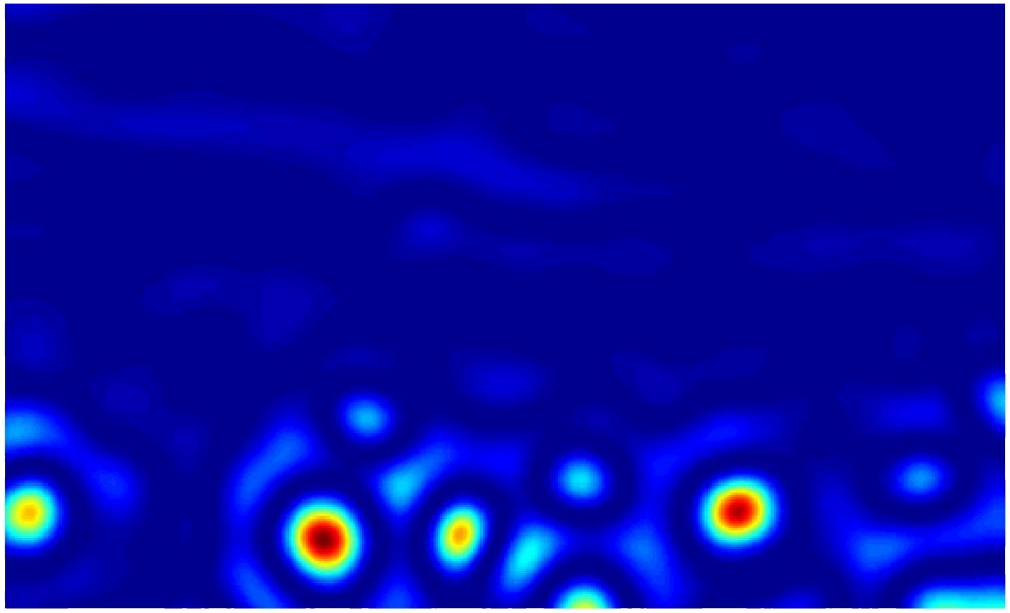


Example

- Original image at 3/4 scale!

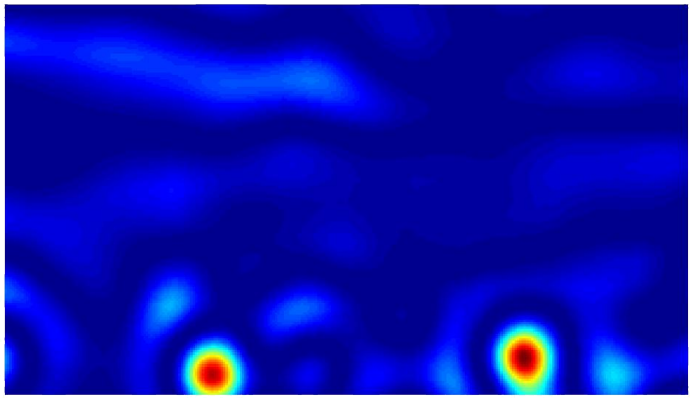


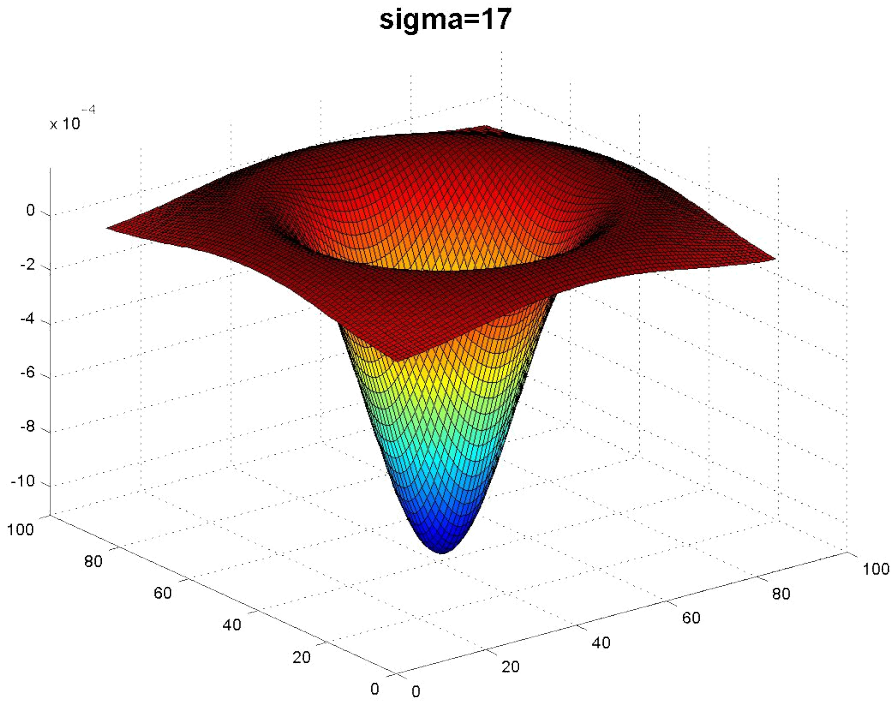


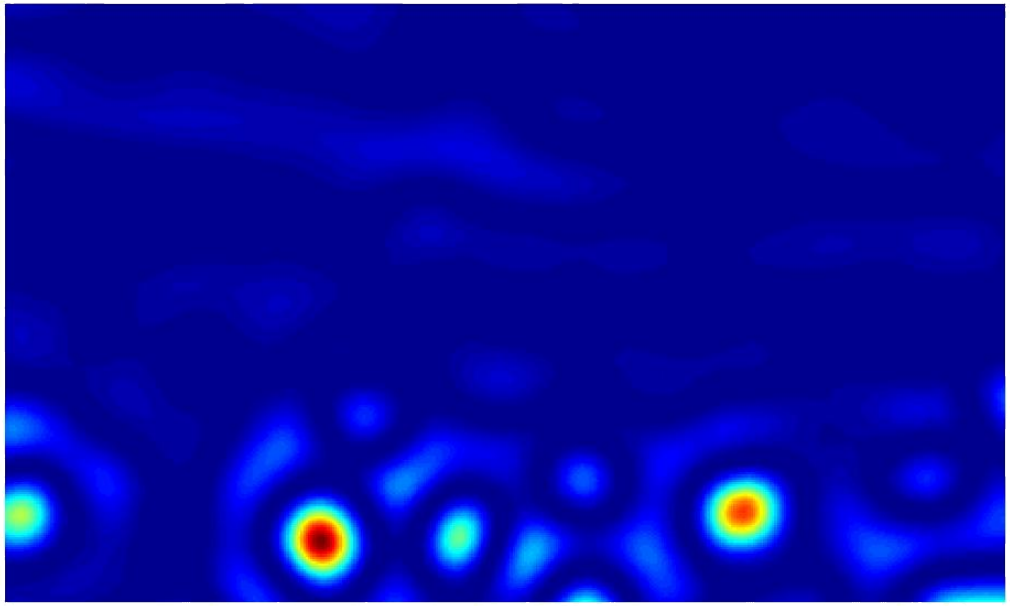


Example

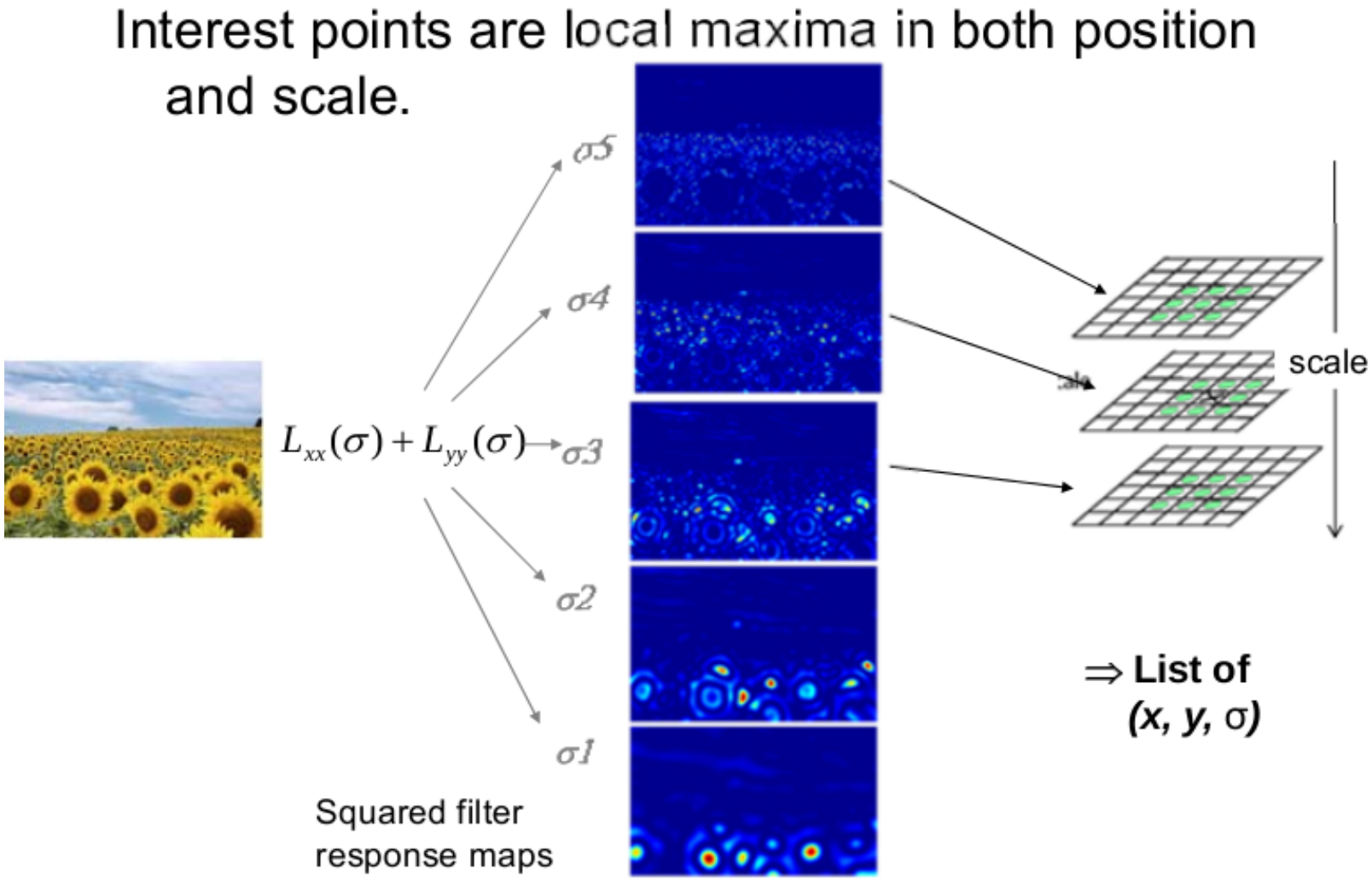
- Original image at 3/4 scale!







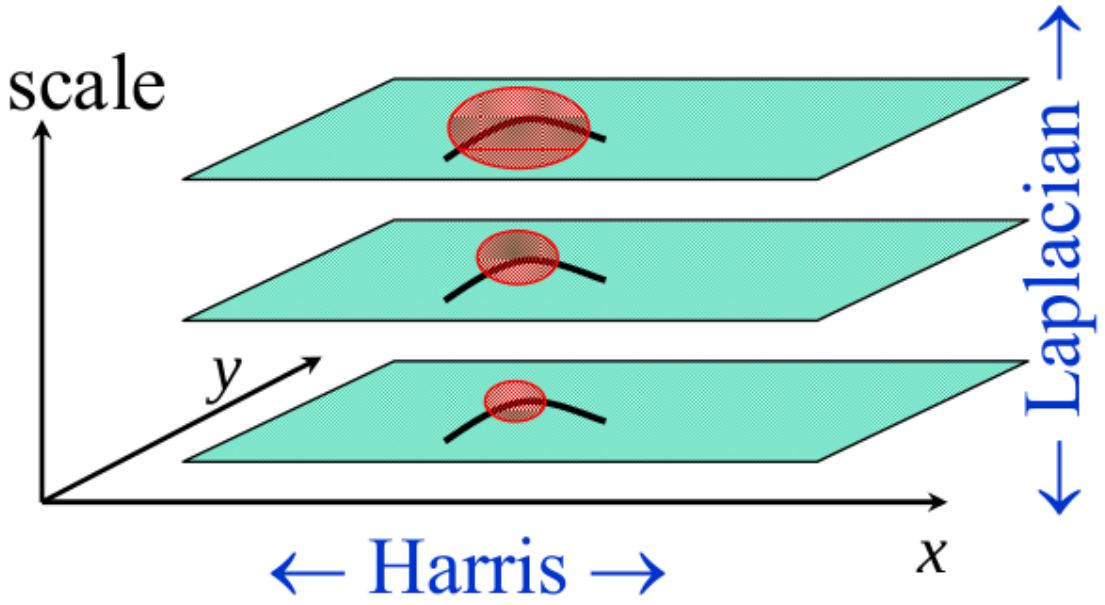
Scale invariant interest points → Harris-Laplacian



Scale invariant feature detector → Harris-Laplacian

Harris-Laplacian

- Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



Scale space feature detector: Example

Let's have a look at this Laplacian kernel!

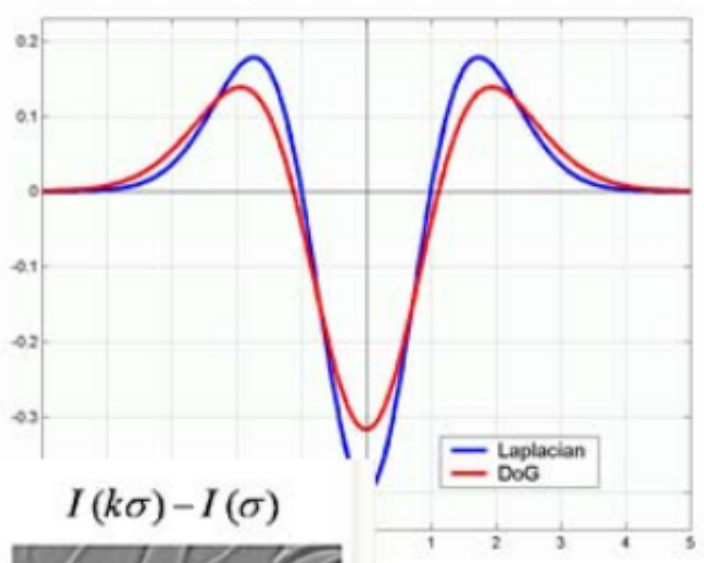
We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

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

(Laplacian)

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

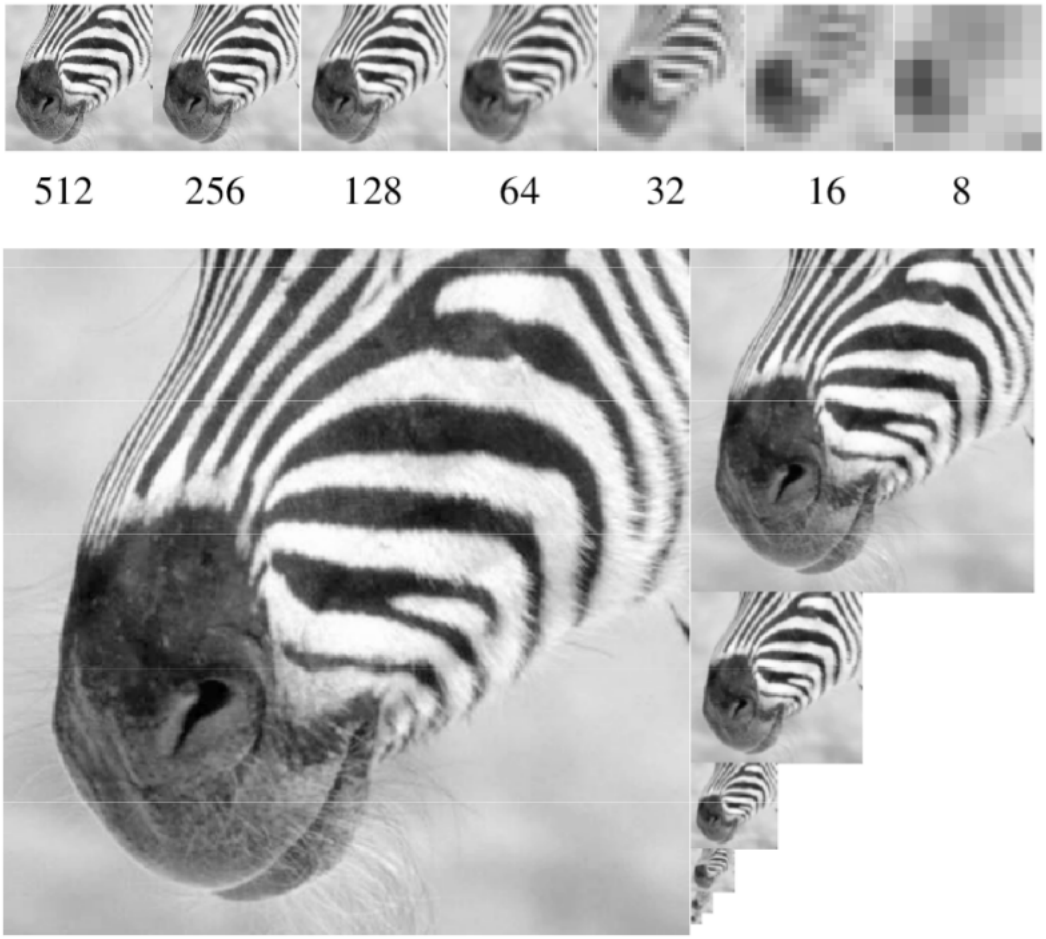
(Difference of Gaussians)



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

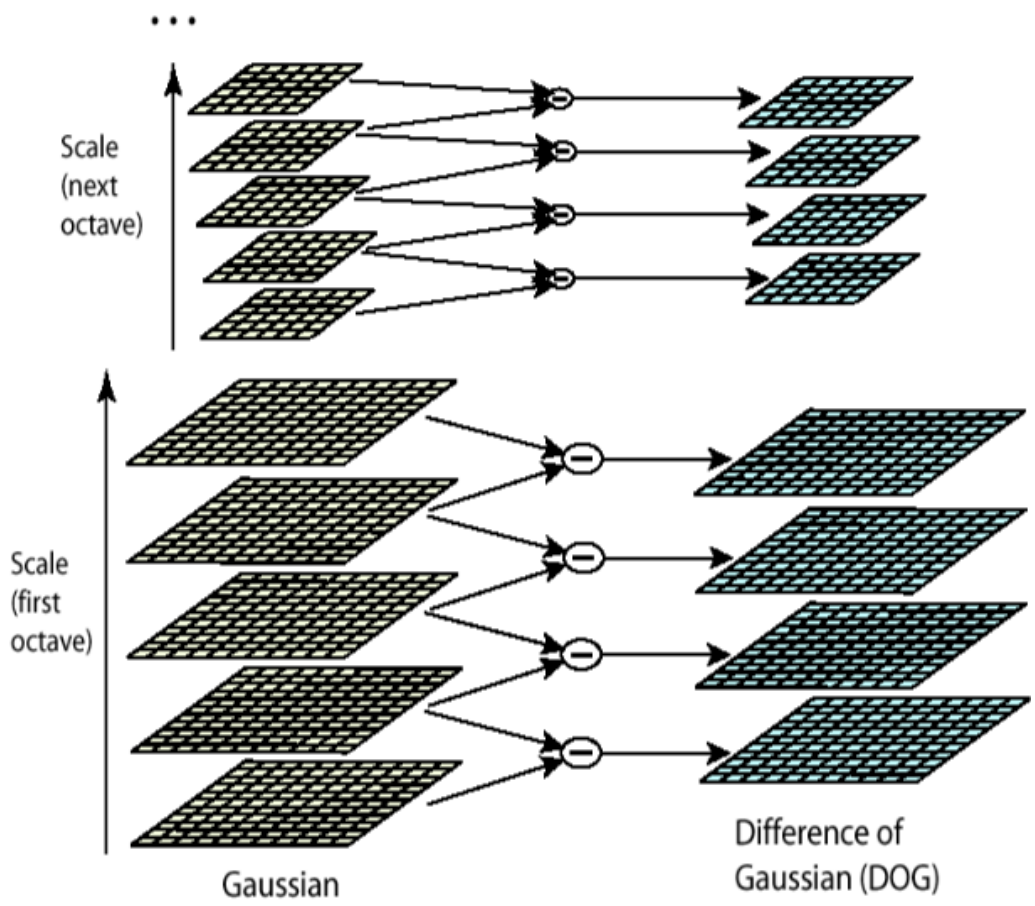
The Scale Invariant Feature Transform (SIFT)

1. Construct an image pyramid at 3-4 scales (octaves - refer to your text)



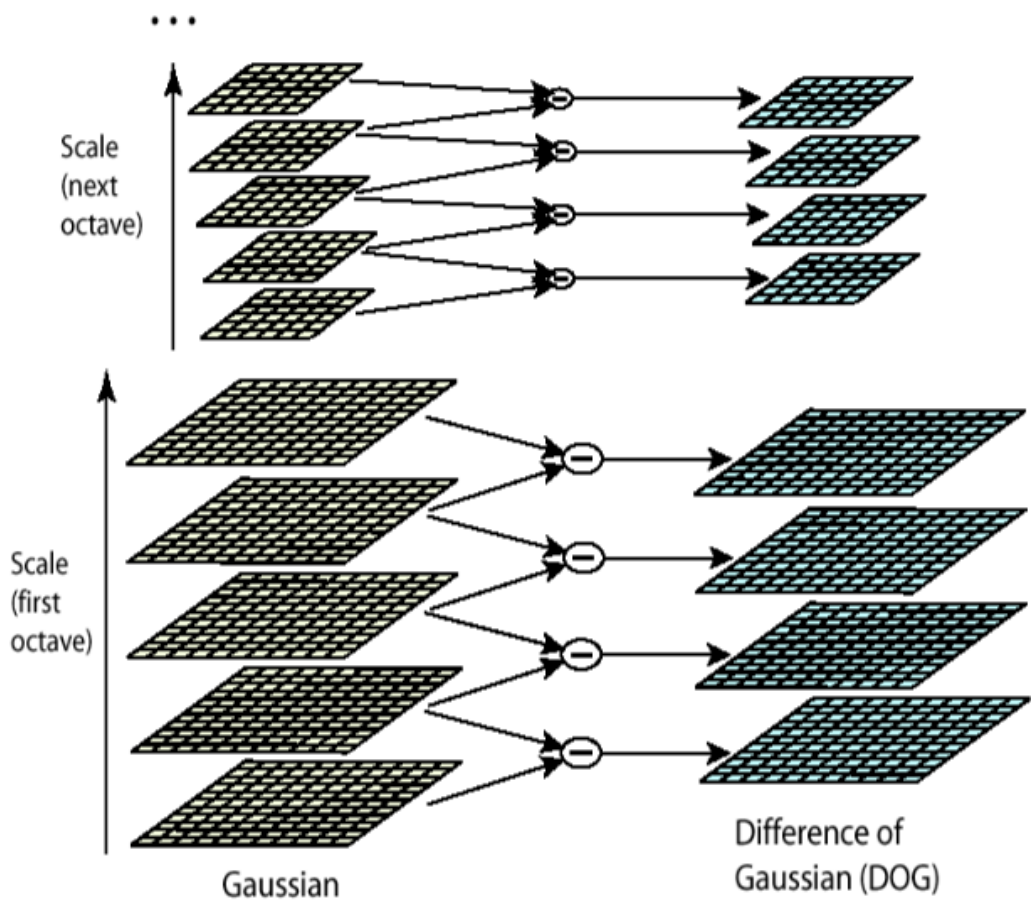
The Scale Invariant Feature Transform (SIFT)

1. Construct an image pyramid at 3-4 scales (octaves - refer to your text)
2. Convolve each image at different scales with a Gaussian kernel $G(x, y, k\sigma)$ (k is selected so we get a fixed number of blurred images/octave)



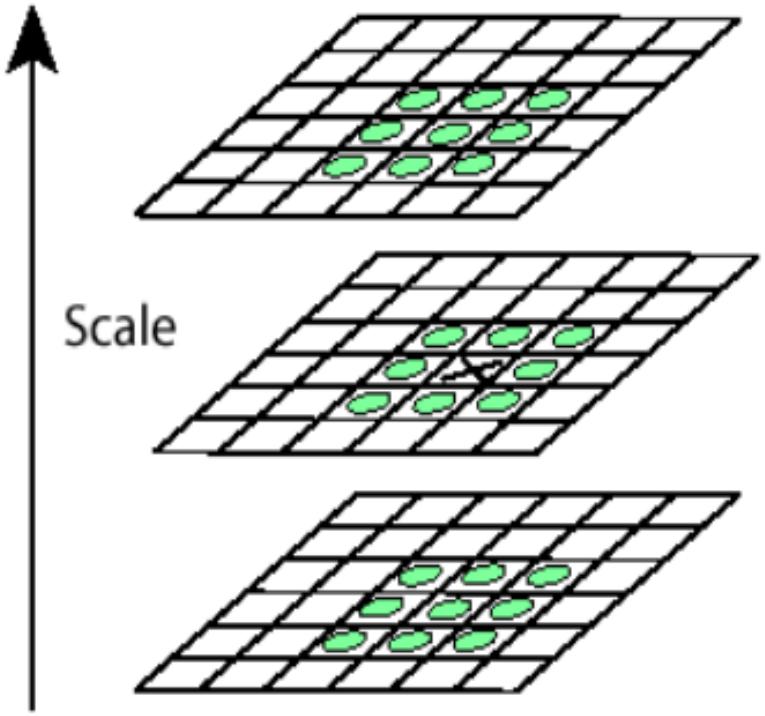
The Scale Invariant Feature Transform (SIFT)

1. Construct an image pyramid at 3-4 scales (octaves - refer to your text)
2. Convolve each image at different scales with a Gaussian kernel $G(x, y, k\sigma)$ (k is selected so we get a fixed number of blurred images/octave)
3. At each octave, compute the set of DoG images



The Scale Invariant Feature Transform (SIFT)

1. Construct an image pyramid at 3-4 scales (octaves - refer to your text)
2. Convolve each image at different scales with a Gaussian kernel $G(x, y, k\sigma)$ (k is selected so we get a fixed number of blurred images/octave)
3. At each octave, compute the set of DoG images
4. Find local maxima or minima in the DoG images across scale
   - Compare to 8-neighbors + 18 neighbors at diff. scales
   - If local max/min = candidate keypoint



The Scale Invariant Feature Transform (SIFT)

1. For each candidate point:
   - Interpolate for sub-pixel localization
     - Taylor series expansion, differentiate, set to zero and solve for the true max/min
     - Improves robustness and stability of the feature

$$
\begin{aligned}
I(x,y) \quad \approx \quad & I(x_0,y_0) + [(x-x_0),(y-y_0)]\,\nabla I(x_0,y_0)^T \\
& + \frac{1}{2}\left([(x-x_0),(y-y_0)]\,\nabla^2 I(x_0,y_0)\,[(x-x_0),(y-y_0)]^2\right) \\
& + o(\cdot)
\end{aligned}
$$

   - Remove low-contrast keypoints (using the Gaussian image at the closest scale set a threshold - again at the sub-pixel level!)
   - Remove edge keypoints (the same as the Harris detector - 1 big gradient and 1 small gradient $\implies$ edge)

The Scale Invariant Feature Transform (SIFT)

- Assign the keypoint an orientation
  - Keypoint orientation: Gradient orientation histogram computed in the neighborhood (using the Gaussian image at the closest scale)
  - The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window ($\sigma = 1.5$ times the scale)
- Keep keypoints for maximum hist direction as well as directions within 80% of the max (relative to the keypoint orientation $\rightarrow$ rotation invariance)
- In Lowe's paper, he used 10° bins at this stage

Note: This is only the keypoint (feature) not the descriptor!

SIFT example - single scale!

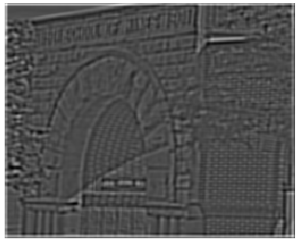
SIFT example - single scale!

SIFT example - single scale!

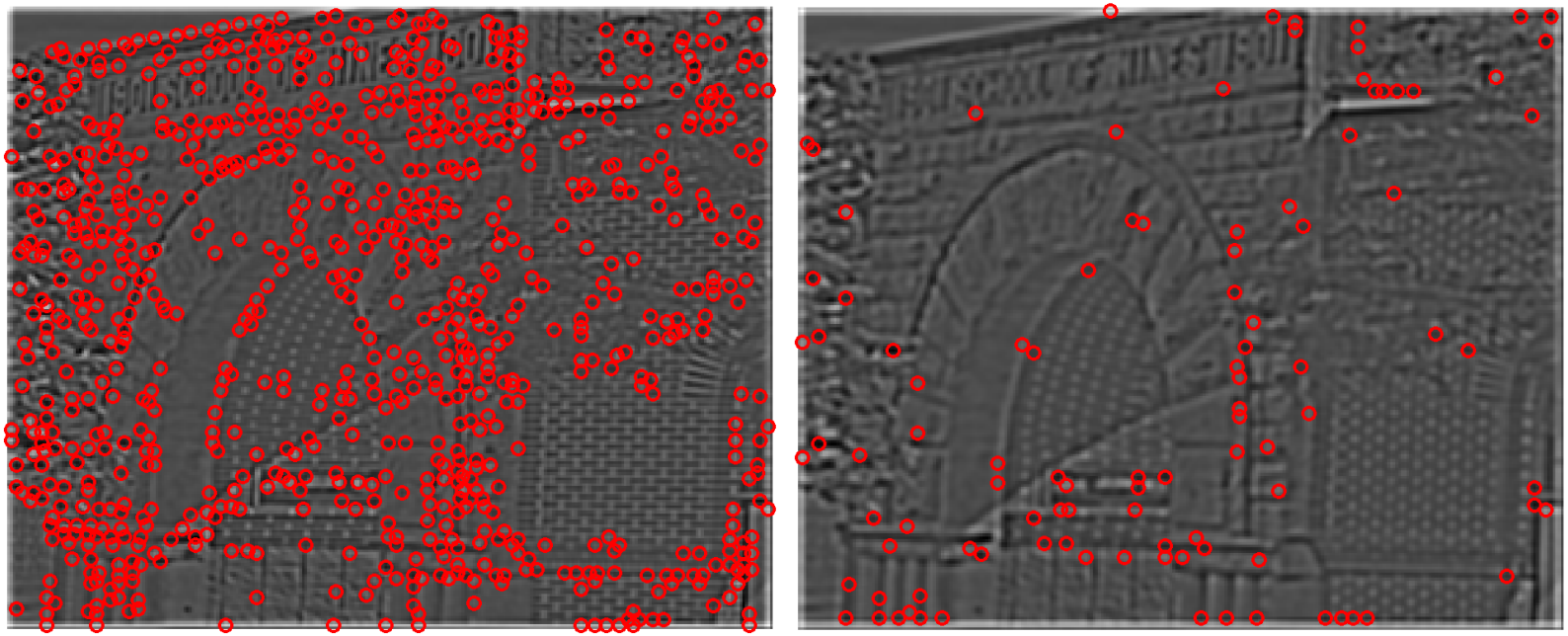




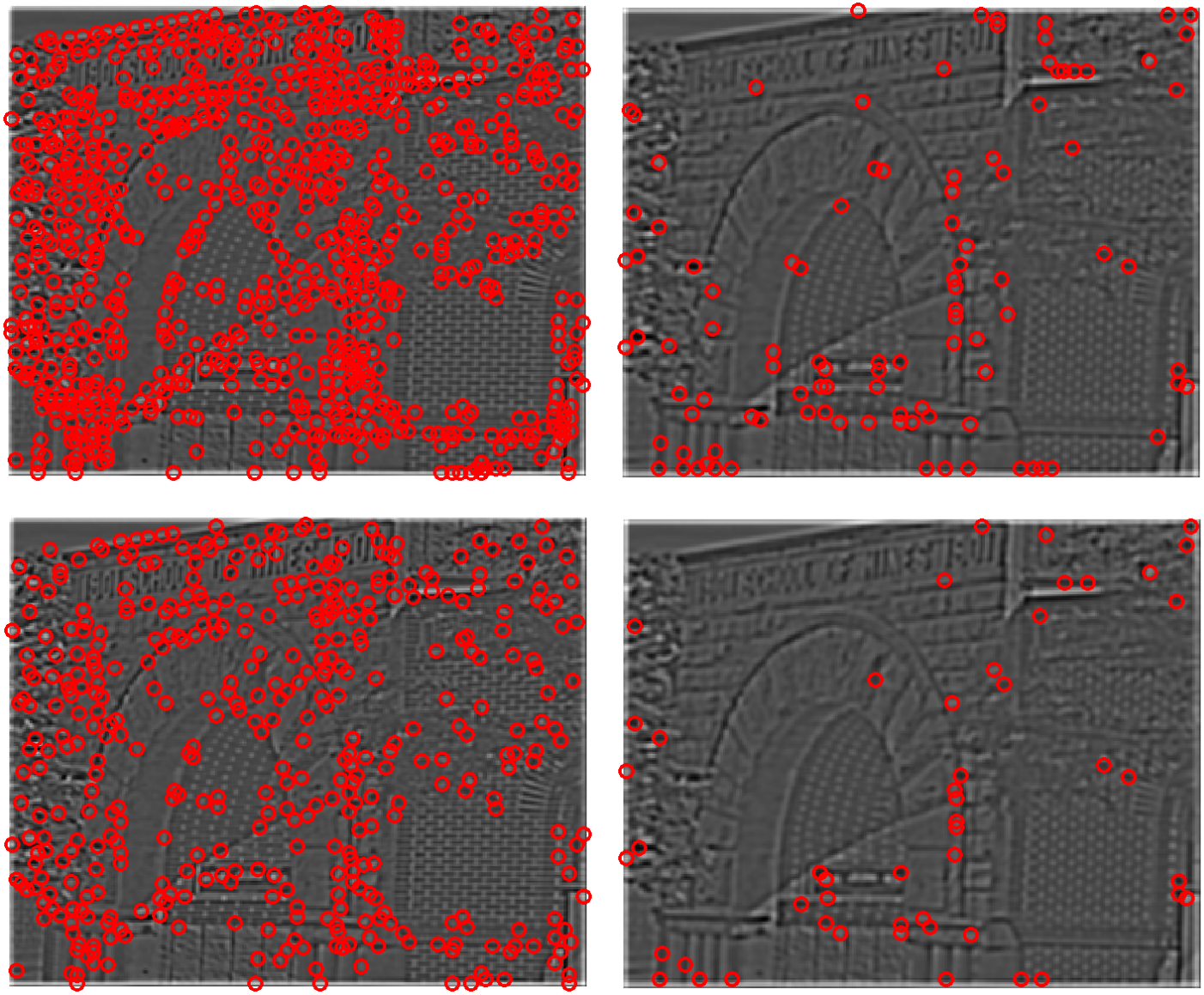


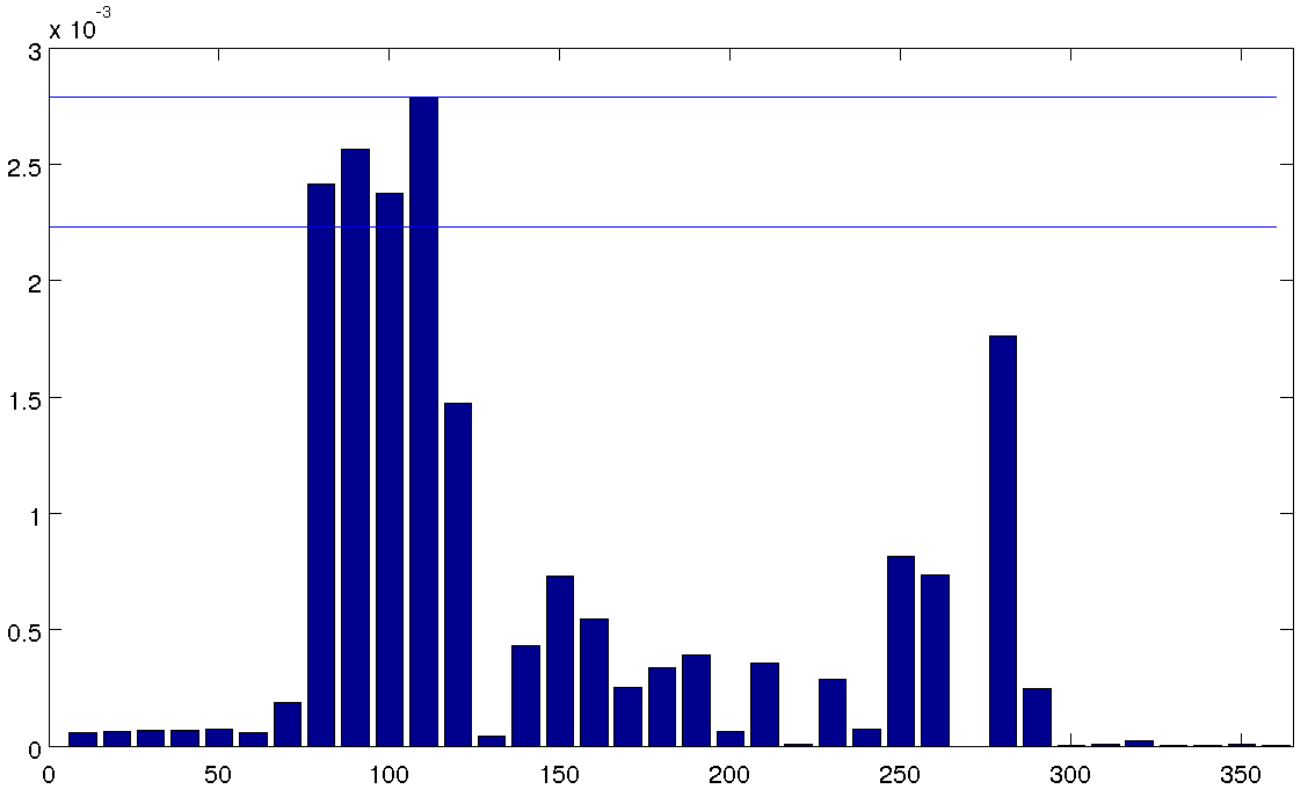

SIFT example - single scale!



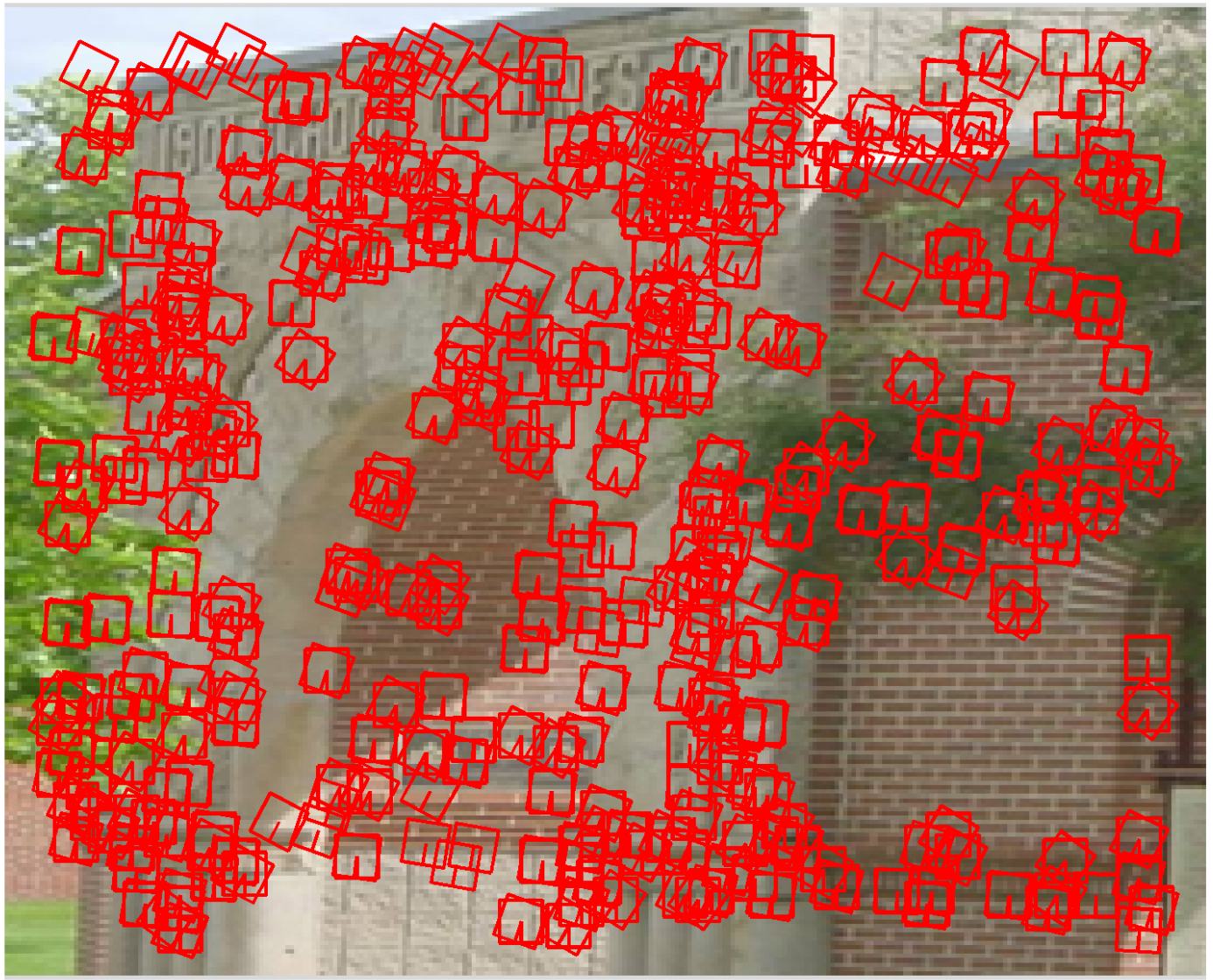
SIFT example - single scale!



SIFT example - single scale!



SIFT example - single scale!



Slide credits

- Prof. Trever Darrell
- Prof. Kristen Grauman
- Forsyth and Ponce
- Rich Szeliski