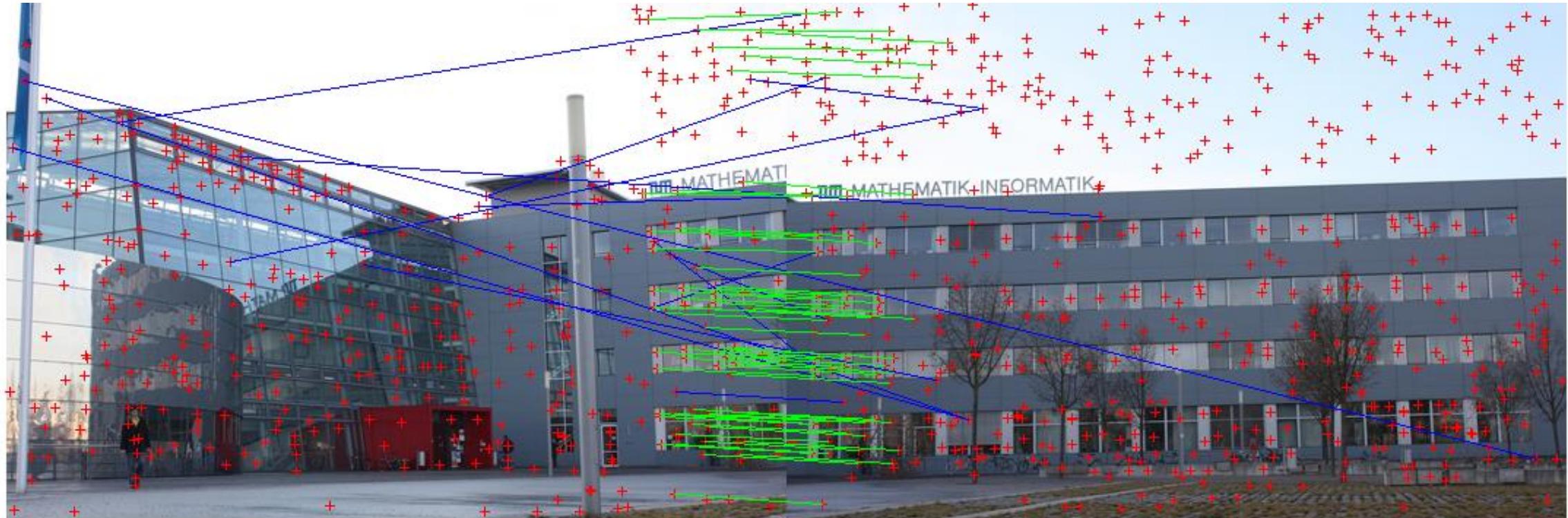


Features



References

- <http://szeliski.org/Book/>
- <http://www.cs.cornell.edu/courses/cs5670/2019sp/lectures/lectures.html>
- <http://www.cs.cmu.edu/~16385/>
- <https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37>

contents

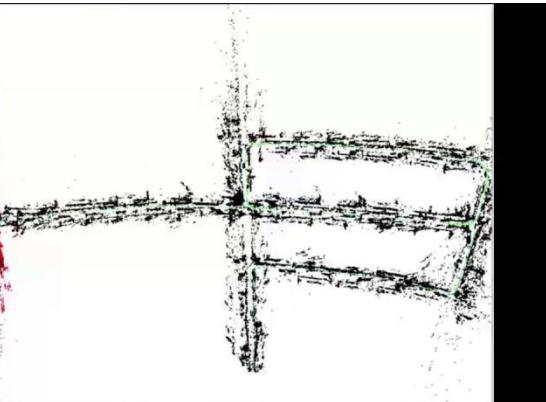
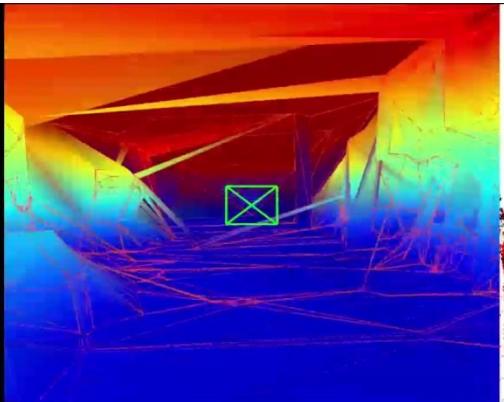
- **What and why we need features detection?**
- Feature detection
 - Blob detection
 - Harris corner detection
 - SIFT detector
- Feature description
 - HOG
 - SIFT descriptor
- SIFT feature matching
- Panoramas

What is a feature?

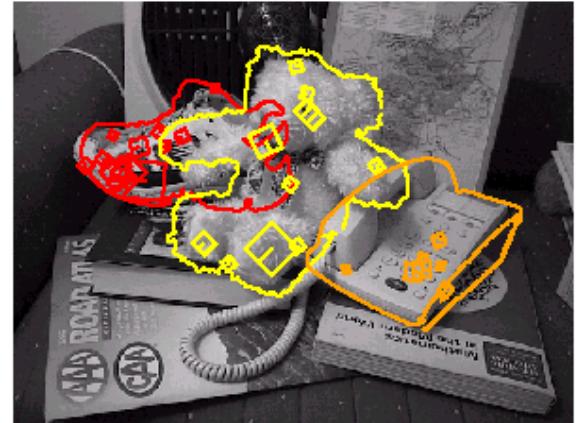
- There is no universal or exact definition of what constitutes a feature, and the exact definition often depends on the problem or the type of application.
Given that, a feature is defined as an "**interesting**" part of an image.
 - [from: wikipedia]

What can we do with features?

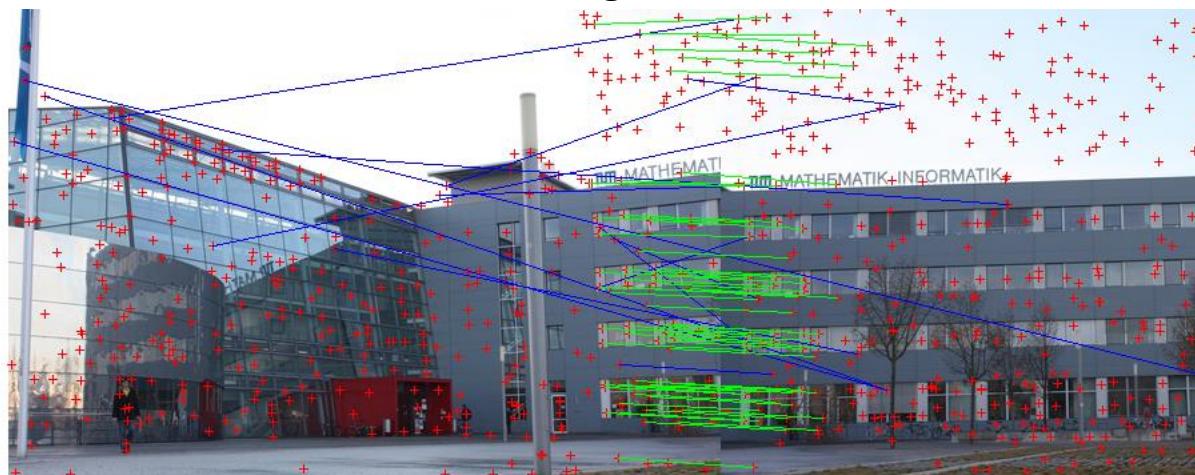
6x



Robot navigation



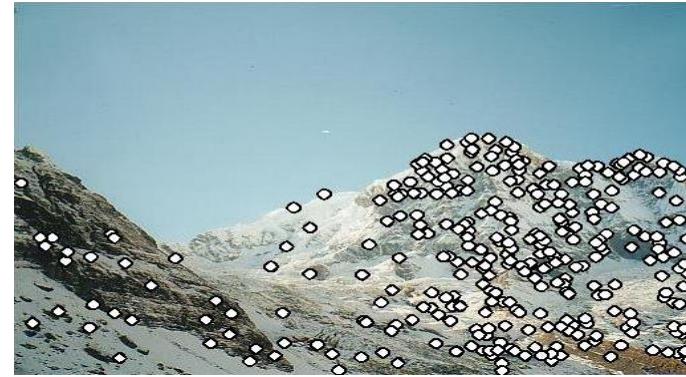
Object recognition



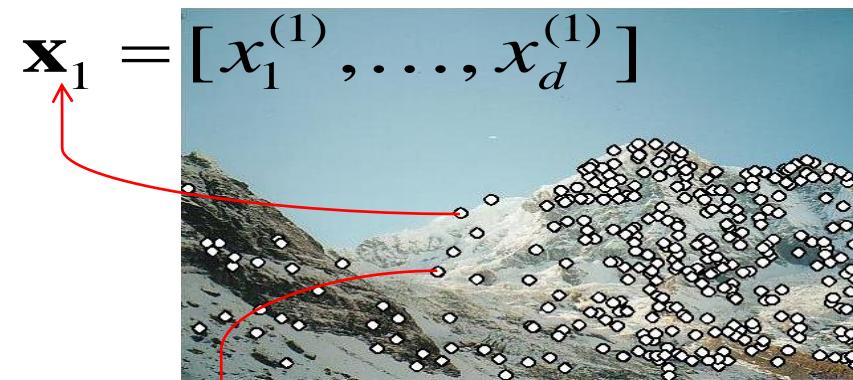
Panorama stitching

Local features: main components

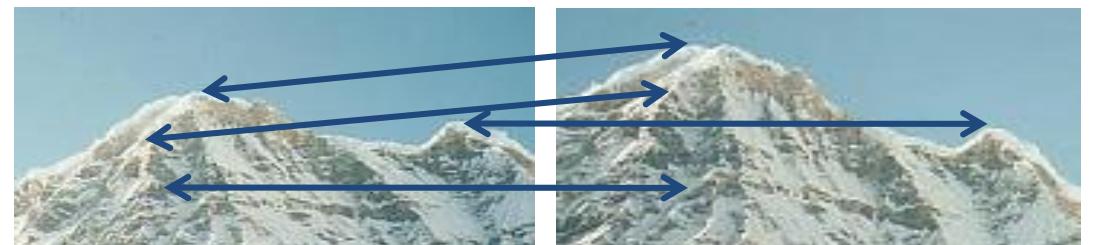
1. Detection: Identify the interest points (also called **keypoints**).



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



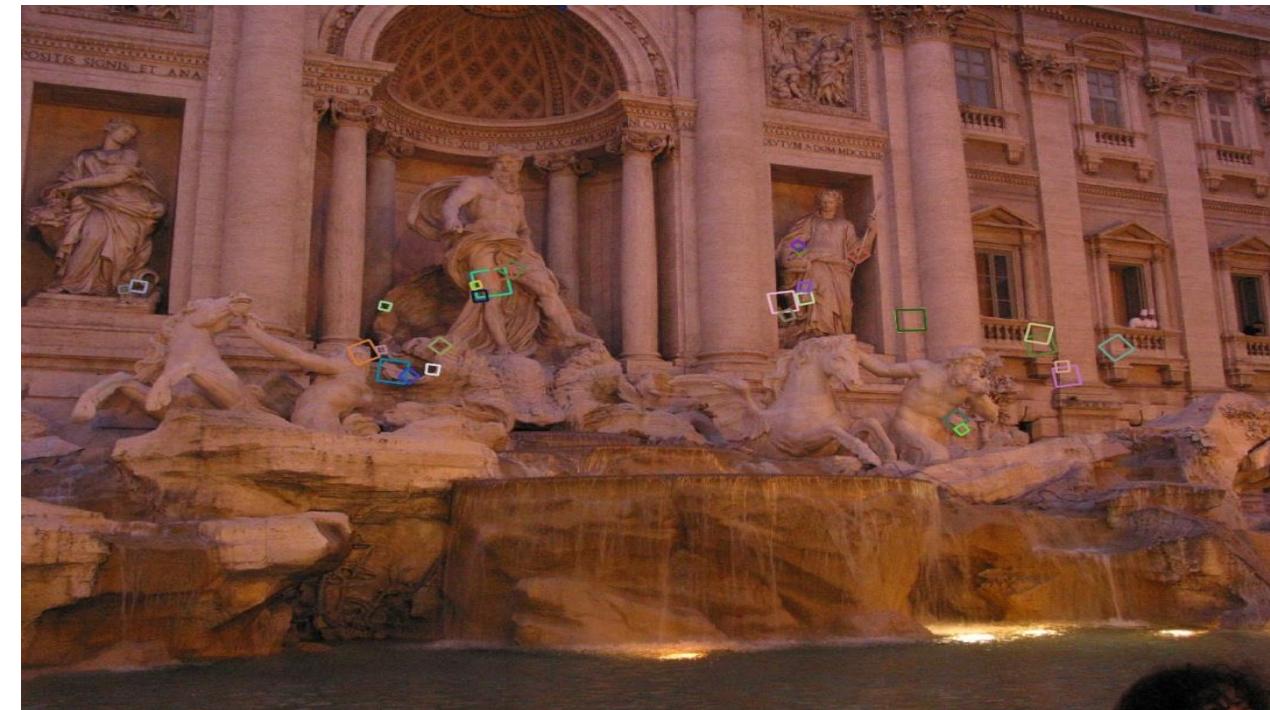
3. Matching: Determine correspondence between descriptors in two views.



Properties of SIFT

- **SIFT: scale invariant feature transform.**
- Extraordinarily robust matching technique for local keypoint detection description and matching.
- Can handle changes in viewpoint: 3D change of POV, scale, rotation and translation.
 - Up to about 60 degree out of plane rotation.
- Can handle significant changes in illumination.
 - Sometimes even day vs. night.
- Fast and efficient—can run in real time.

SIFT example



contents

- What and why we need features detection?
- **Feature detection**
 - Blob detection
 - Harris corner detection
 - SIFT detector
- Feature description
 - HOG
 - SIFT descriptor
- SIFT feature matching
- Panoramas

Advantages of local keypoints

Locality:

- features are local, so robust to occlusion and clutter => invariance to 3D change of viewpoint.

Quantity:

- hundreds or thousands in a single image

Distinctiveness:

- can differentiate a large database of objects

Efficiency

- real-time performance achievable

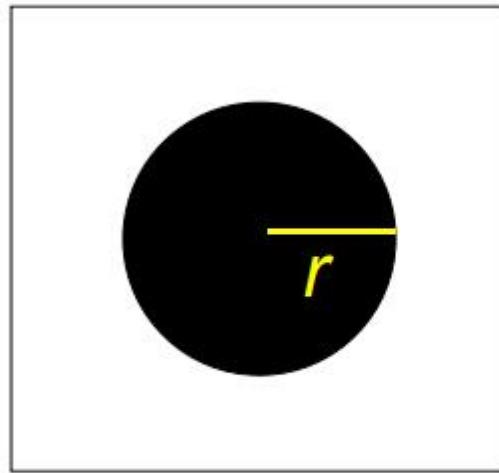
contents

- What and why we need features detection?
- Feature detection
 - **Blob detection**
 - Harris corner detection
 - SIFT detector
- Feature description
 - HOG
 - SIFT descriptor
- SIFT feature matching
- Panoramas

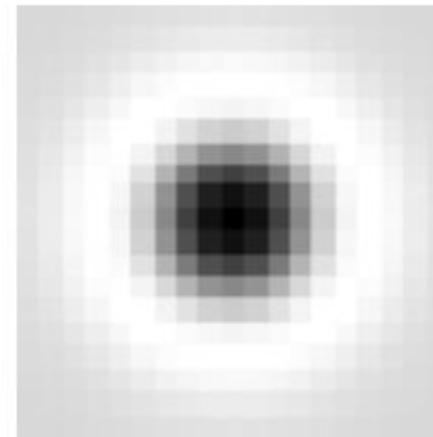
Blob detection

- Like template matching- we are convolving a signal with a template of a LoG (Laplacian of Gaussian) to get the highest response when the template matches the signal.

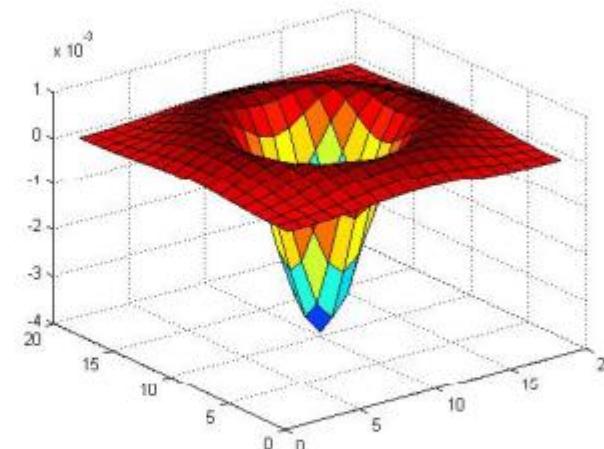
$$\text{LoG}(x, y; \sigma) = \Delta_{(x,y)} G(x, y; \sigma) = \frac{\partial^2 G(x, y; \sigma)}{\partial x^2} + \frac{\partial^2 G(x, y; \sigma)}{\partial y^2} = \frac{1}{\pi \sigma^4} \left(\frac{x^2 + y^2}{2\sigma^2} - 1 \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



image



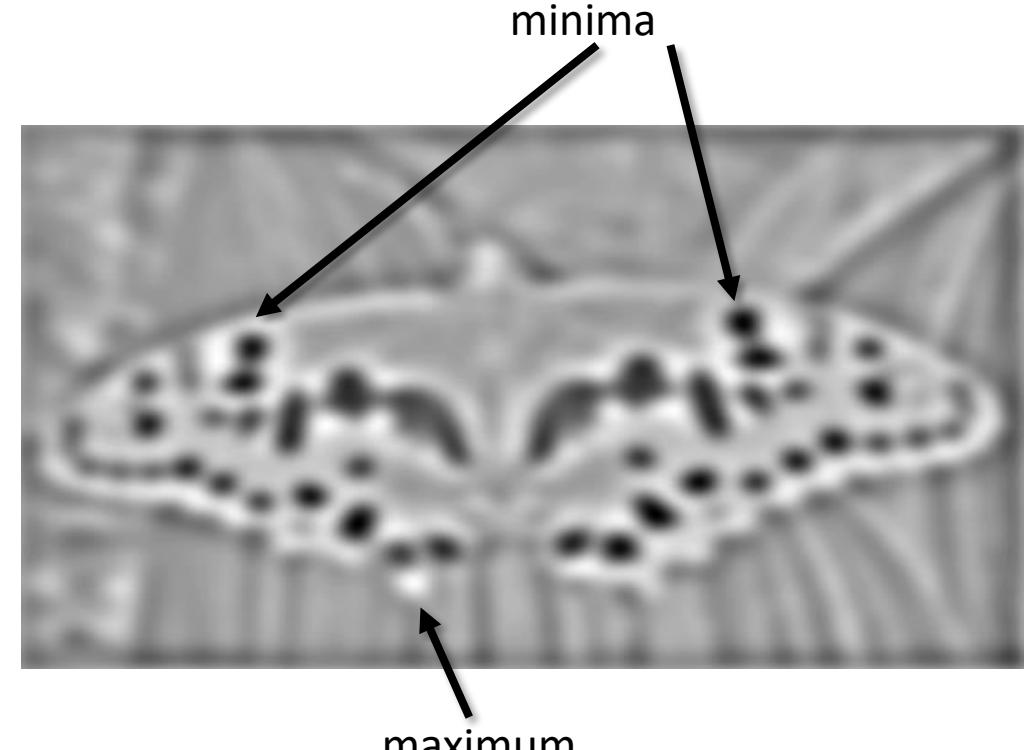
Laplacian



Blob detection



$$* \quad =$$

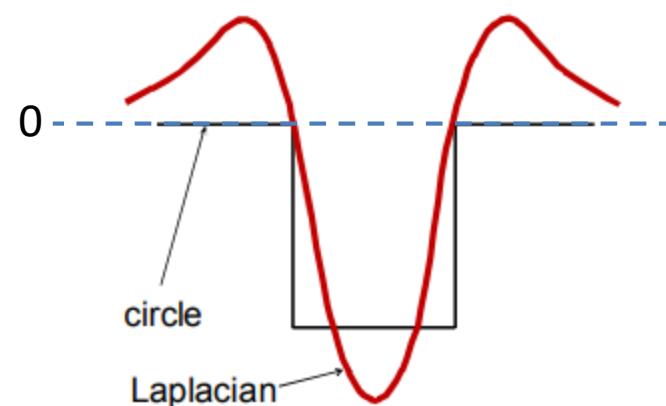
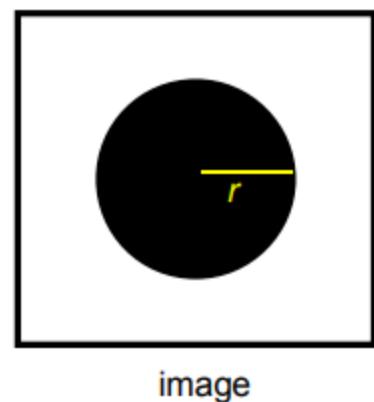


- Find maxima *and minima* of LoG operator in space and scale

Blob detection

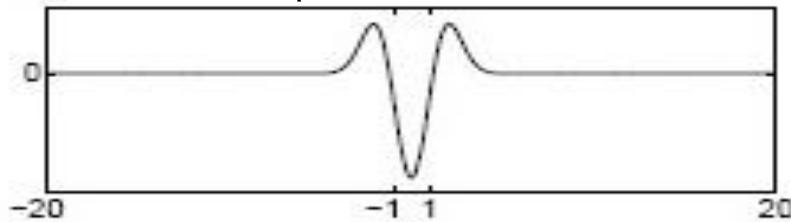
$$\text{LoG}(x, y; \sigma) = \Delta_{(x,y)} G(x, y; \sigma) =$$
$$\frac{\partial^2 G(x, y; \sigma)}{\partial x^2} + \frac{\partial^2 G(x, y; \sigma)}{\partial y^2} = \frac{1}{\pi \sigma^4} \left(\frac{x^2 + y^2}{2\sigma^2} - 1 \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Highest response occurs when the signal is exactly the width of the positive part of the LoG => $r = \sqrt{2}\sigma$.

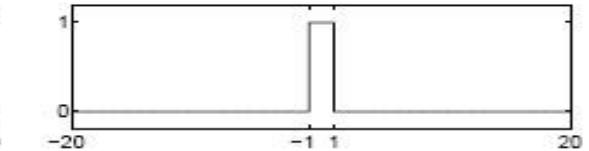
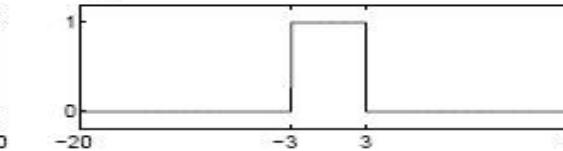
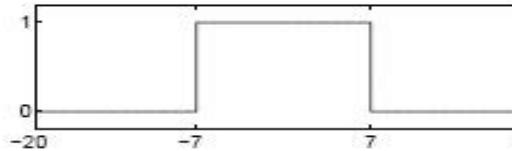
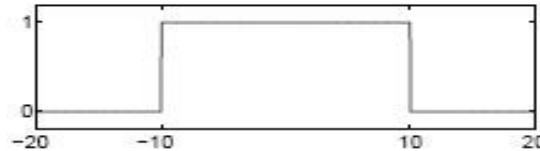


Blob detection

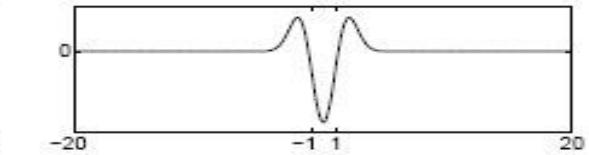
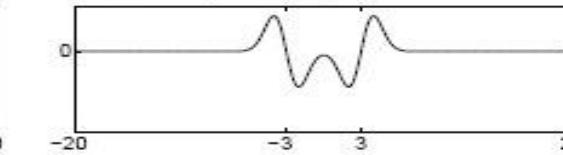
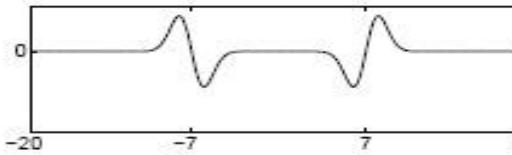
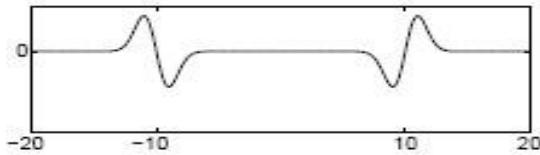
Laplacian filter



Original signal



Convolved with Laplacian ($\sigma = 1$)

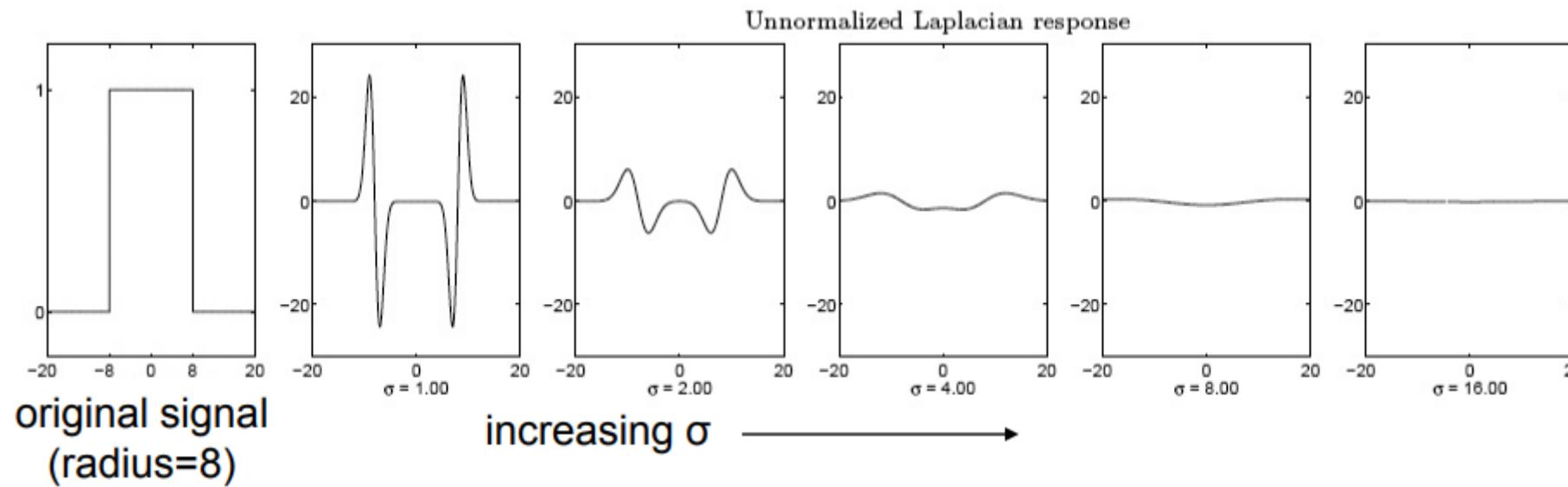


Highest absolute response when the signal has the same **characteristic scale** as the filter



Blob detection- normalized LoG

- We want to find the characteristic scale of the blob by convolving it with LoGs at several scales and looking for the maximum response.
- However, LoG response decays as scale increases.



Blob detection- normalized LoG

- We want that the maximum of the LoG will be always at the same value, so we are using the scale normalized LoG:

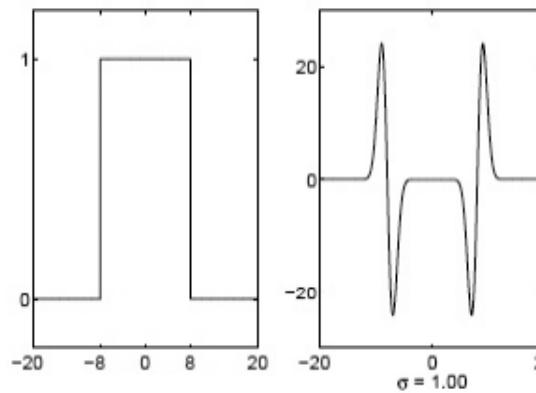
$$\text{normLoG}(x, y; \sigma) = \sigma^2 \Delta_{(x,y)} G(x, y; \sigma) = \sigma^2 \left(\frac{\partial^2 G(x, y; \sigma)}{\partial x^2} + \frac{\partial^2 G(x, y; \sigma)}{\partial y^2} \right)$$

- Full derivation is available here:

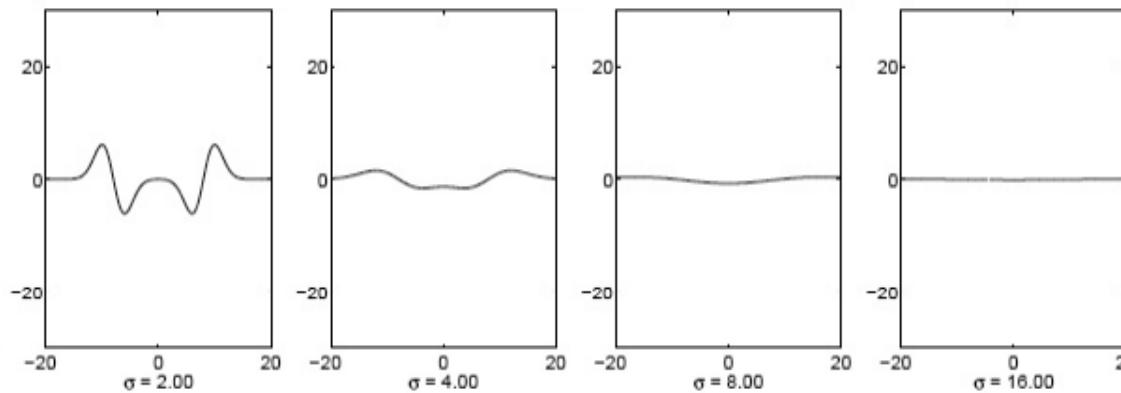
<http://www.cim.mcgill.ca/~langer/558/2009/lecture11.pdf>

Blob detection- normalized LoG

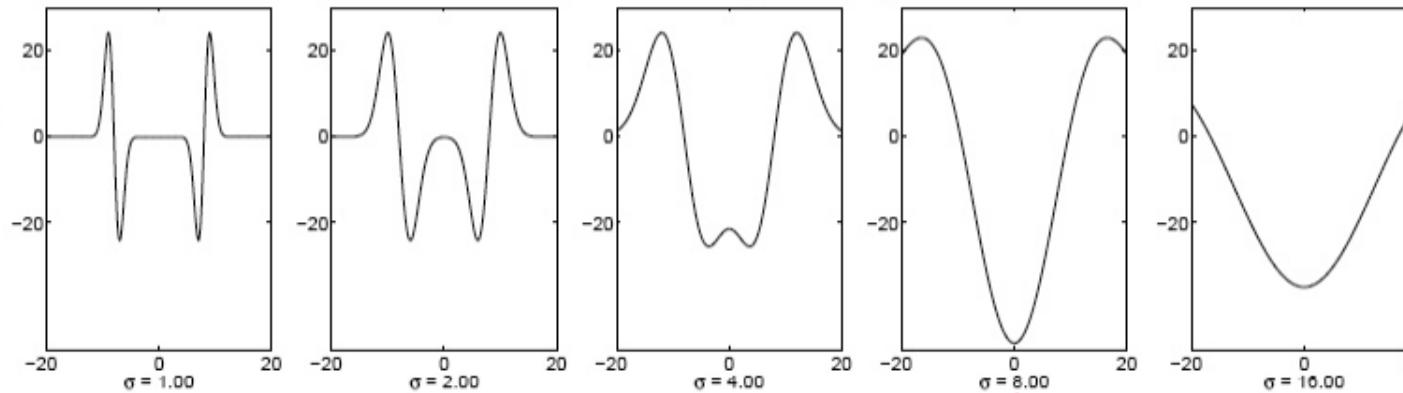
Original signal



Unnormalized Laplacian response



Scale-normalized Laplacian response

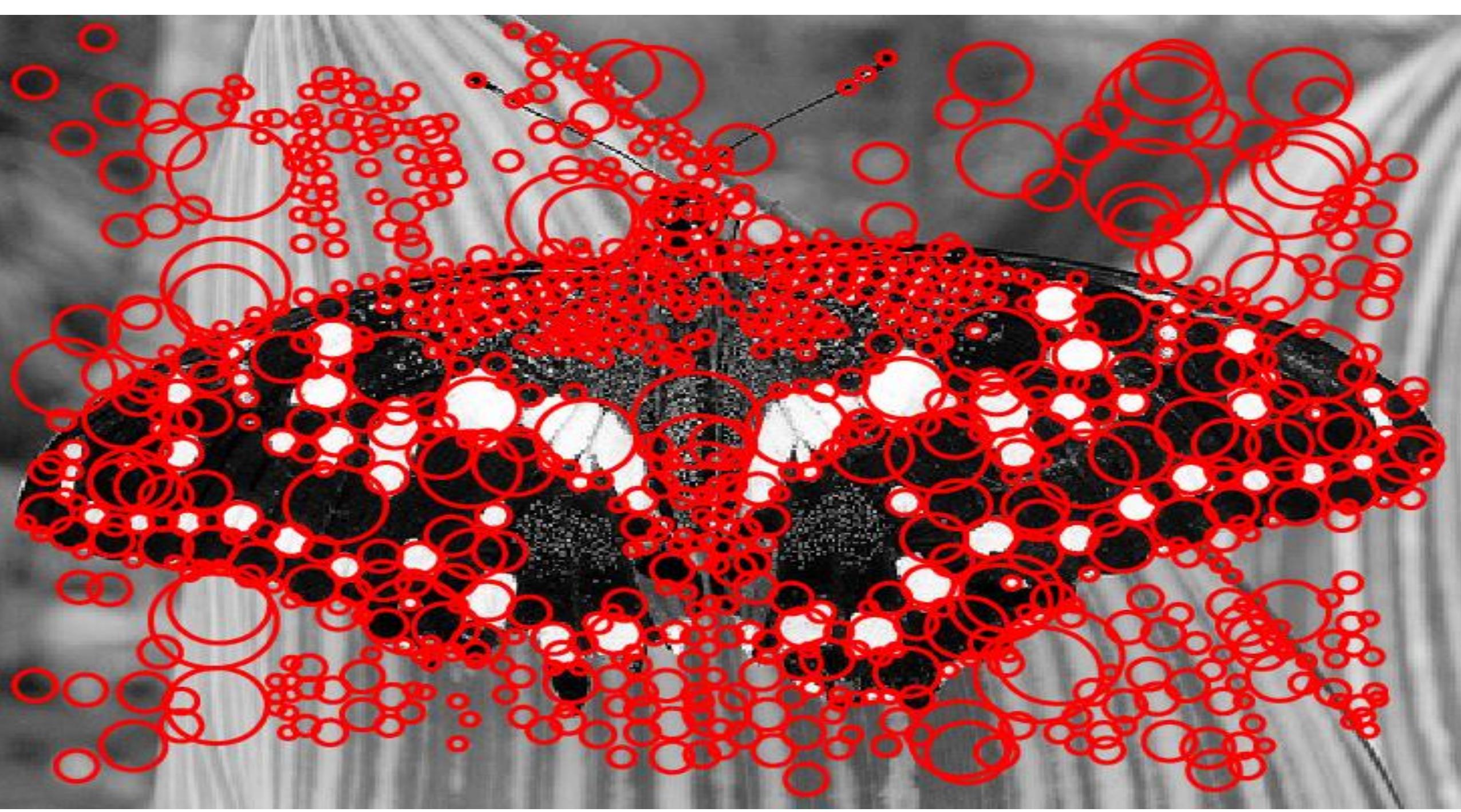


maximum

Blob detection- normalized LoG

- An example of max responses:





Normalized LoG

- responses of the same features at different scales are the same!

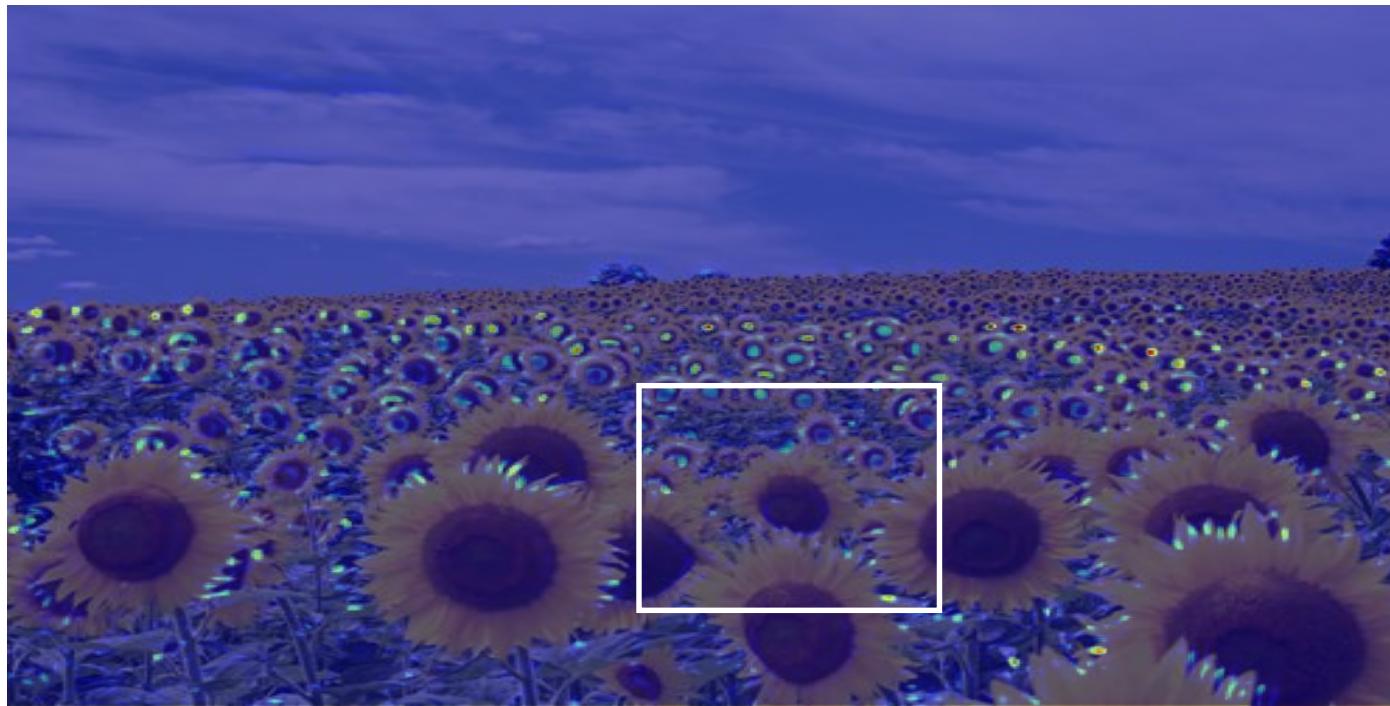
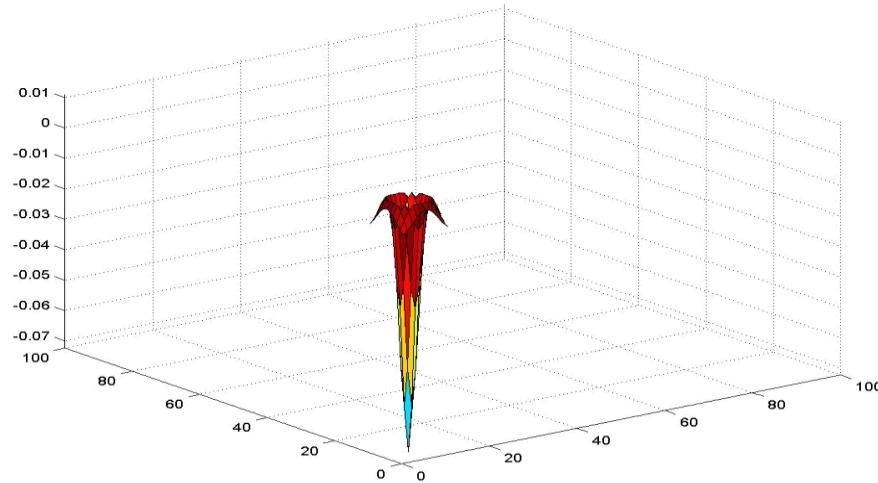
Full size



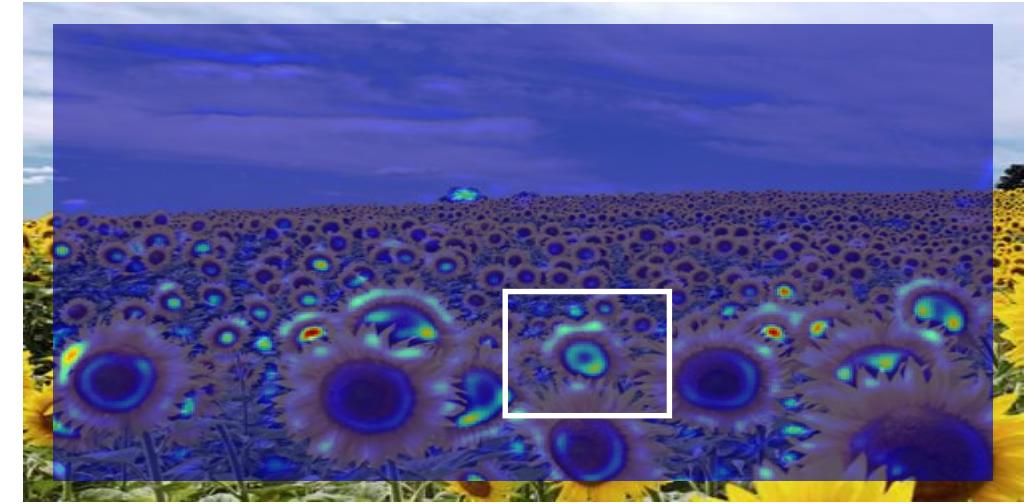
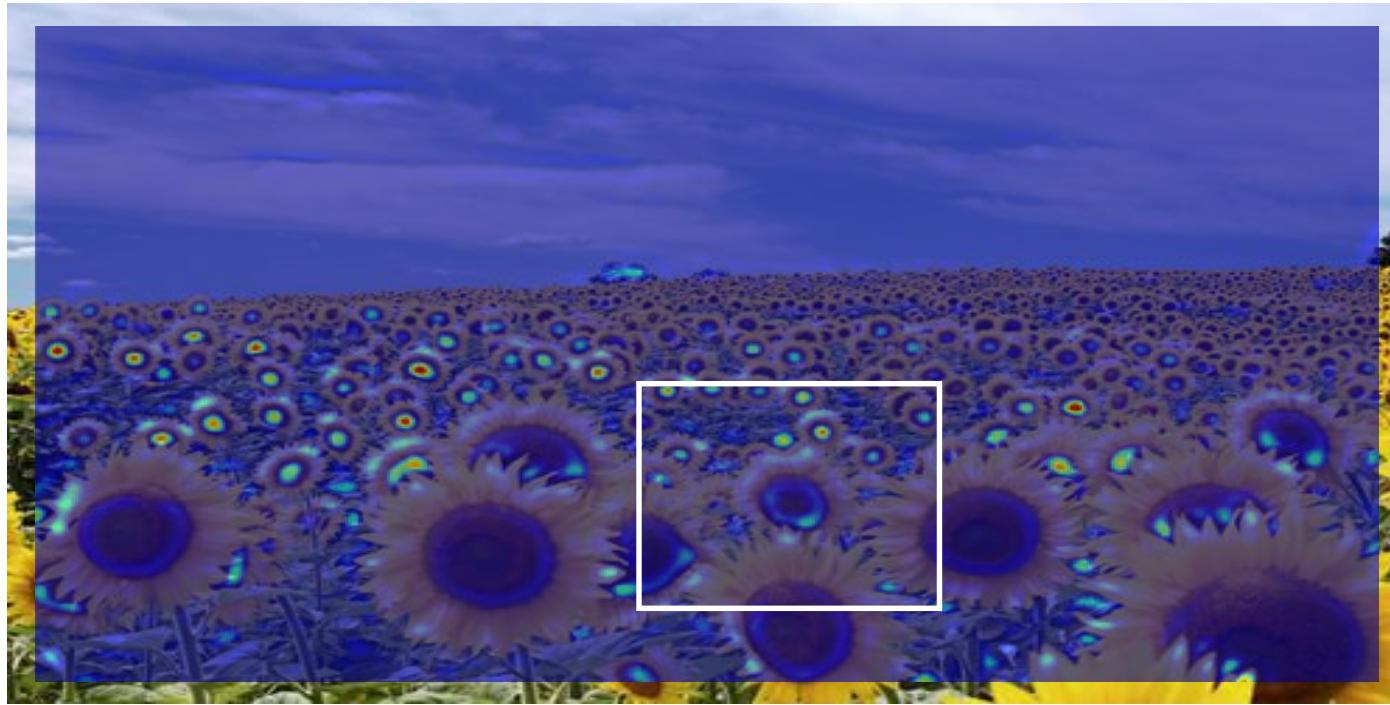
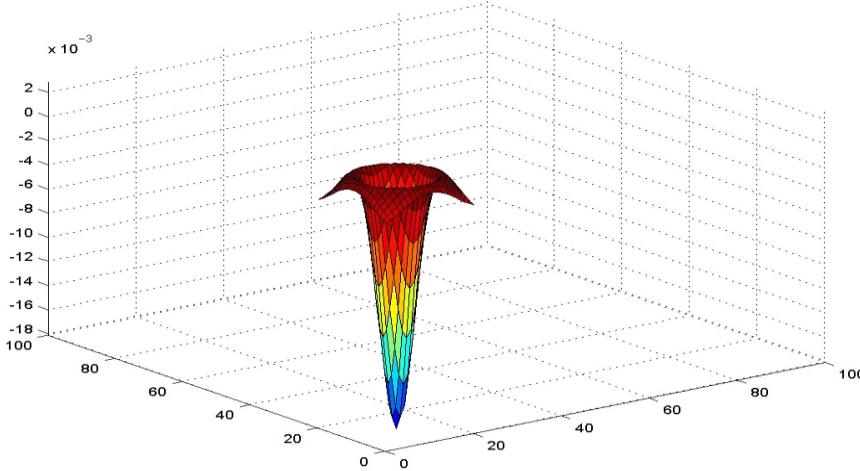
3/4 size

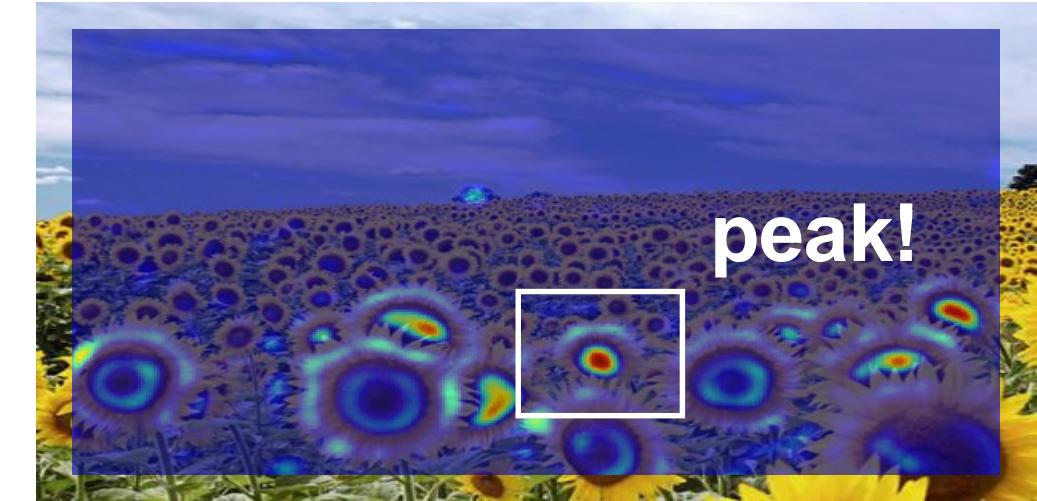
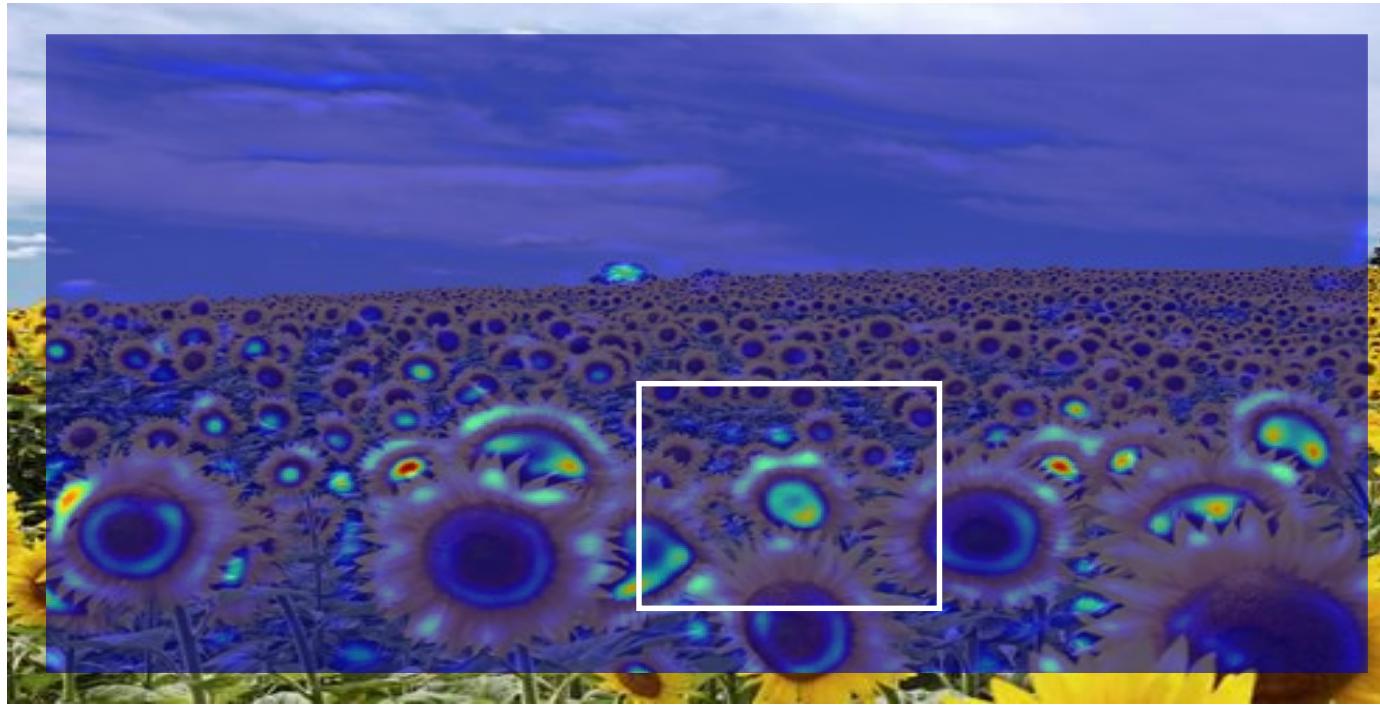
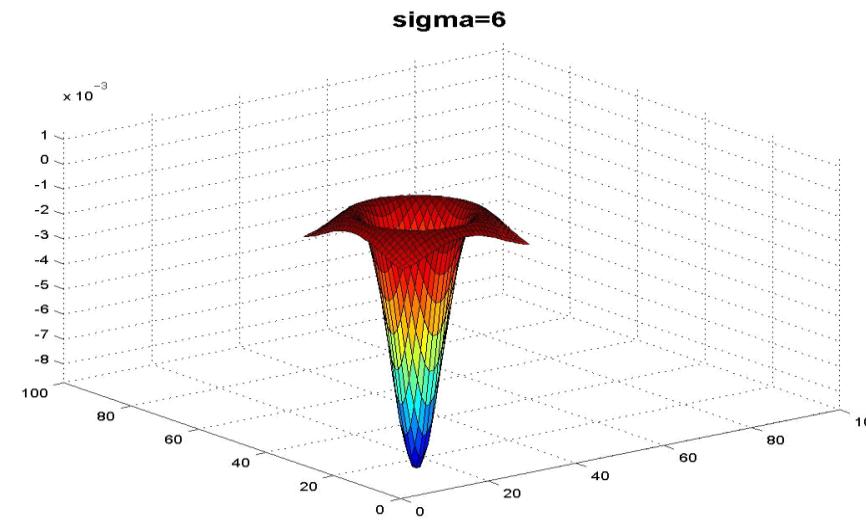


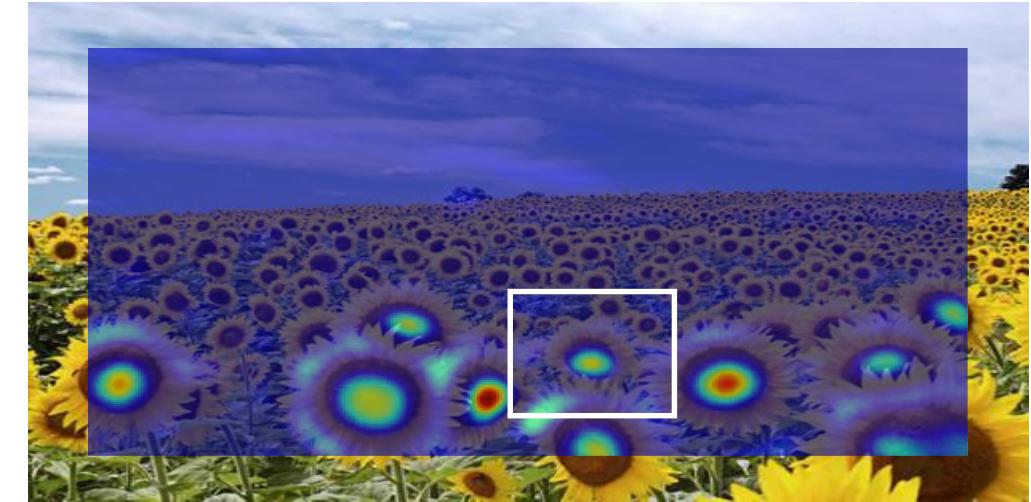
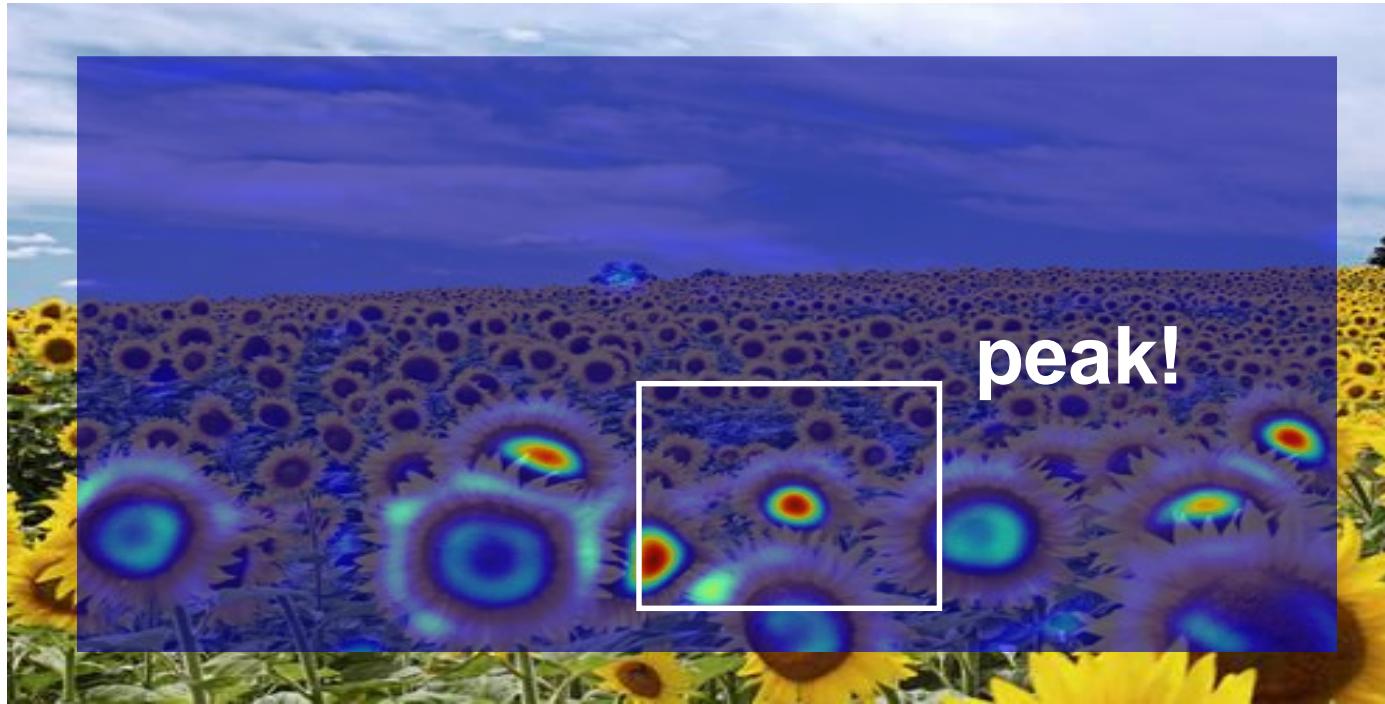
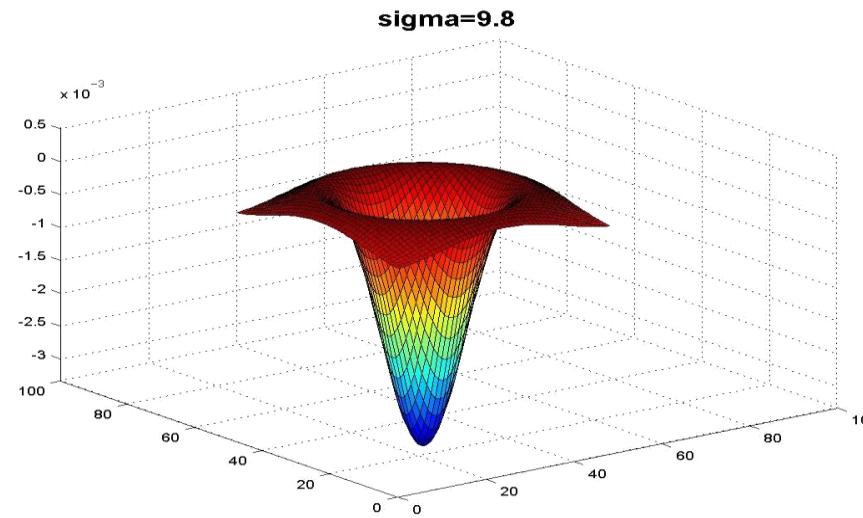
sigma=2.1



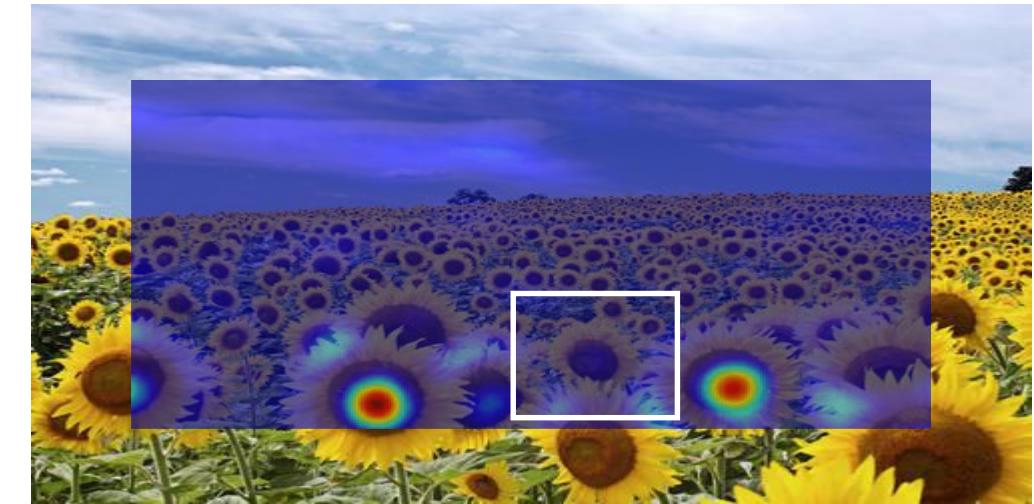
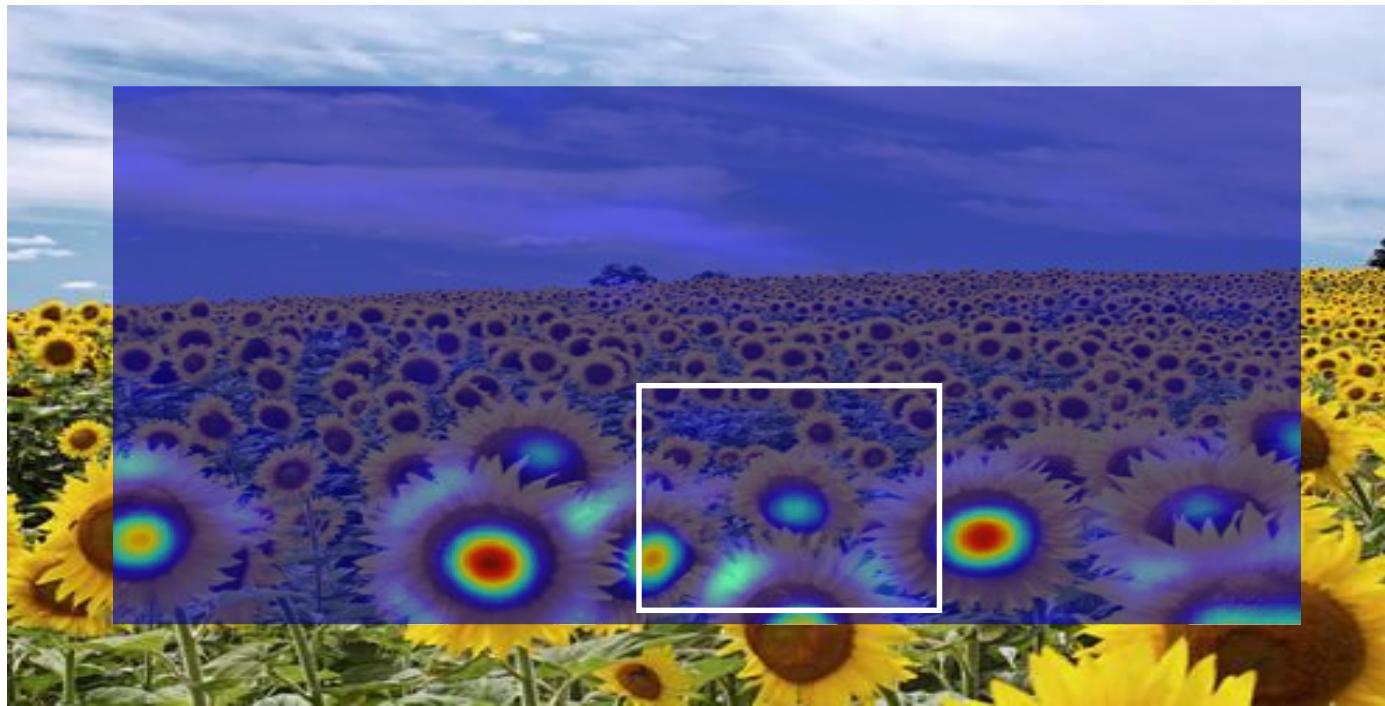
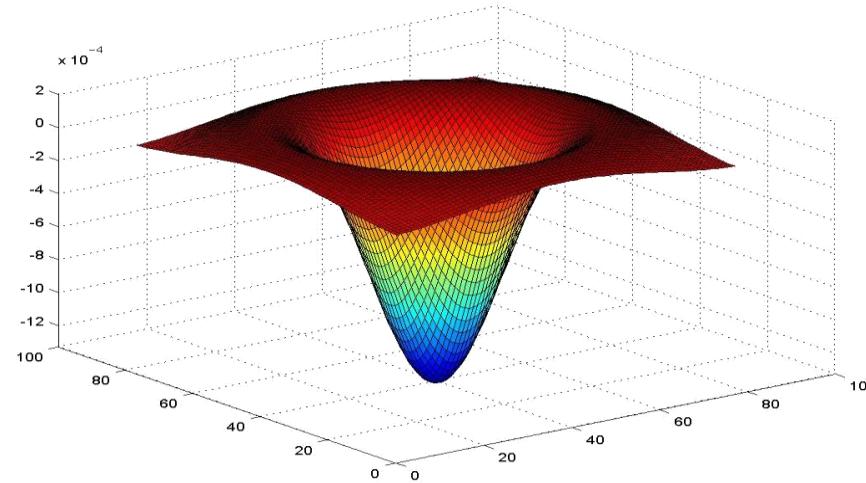
sigma=4.2

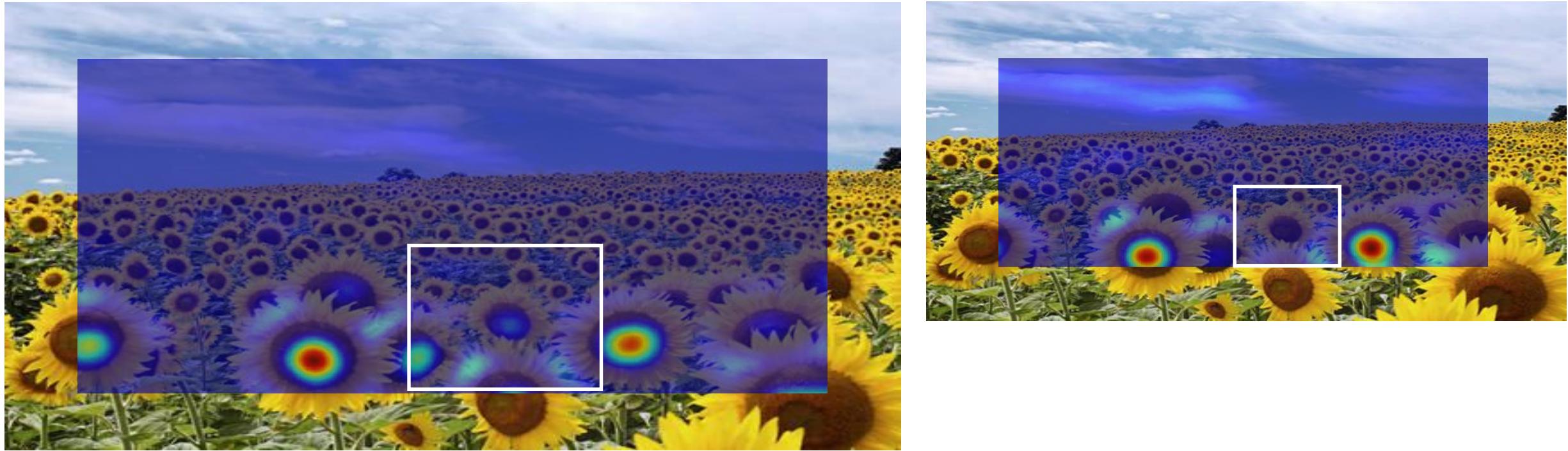
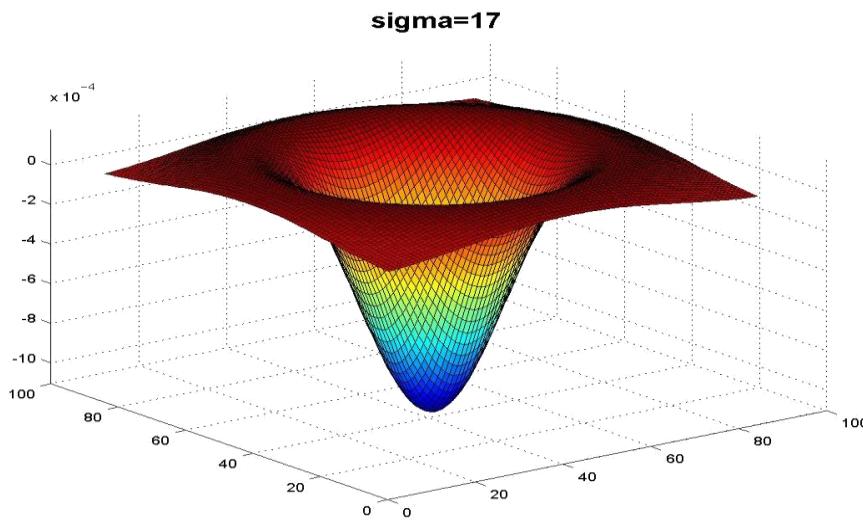






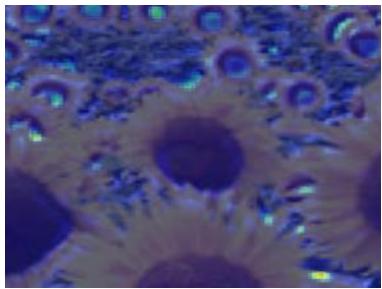
sigma=15.5



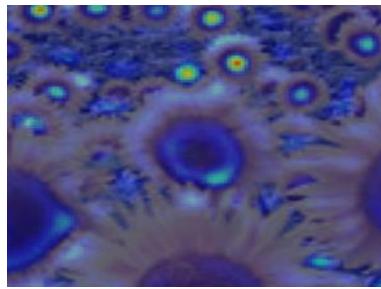


optimal scale

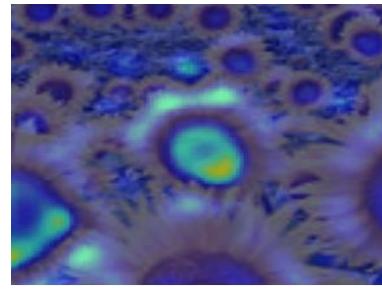
2.1



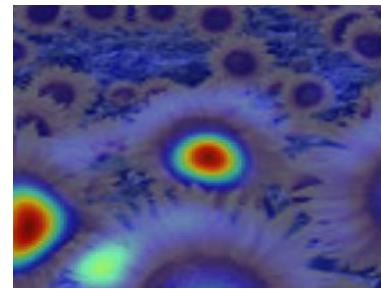
4.2



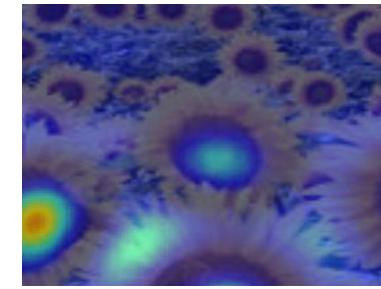
6.0



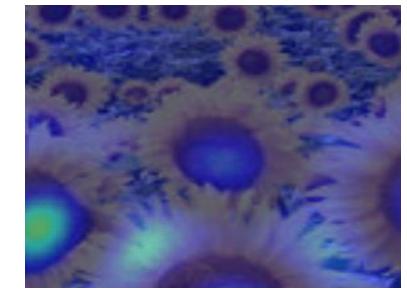
9.8



15.5

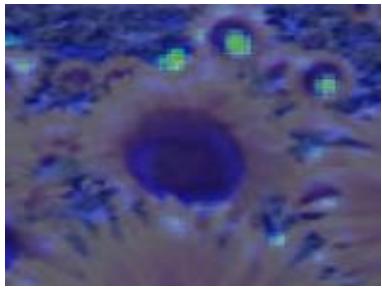


17.0

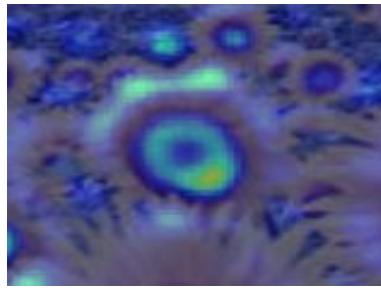


Full size image

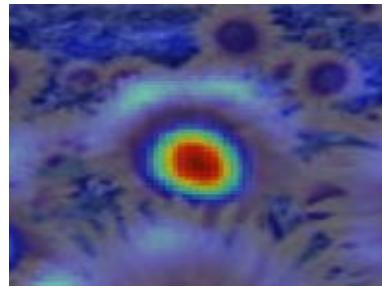
2.1



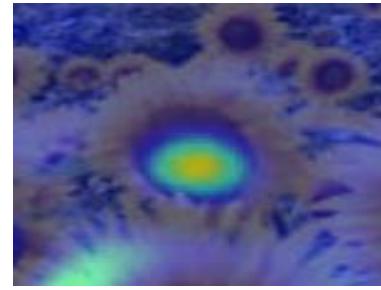
4.2



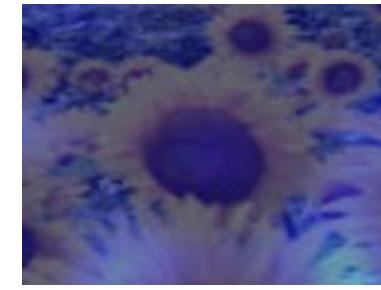
6.0



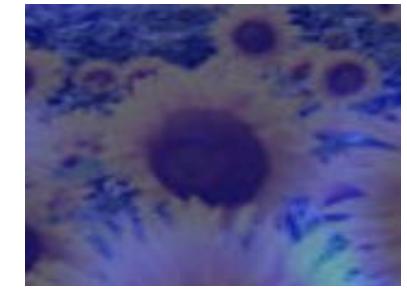
9.8



15.5



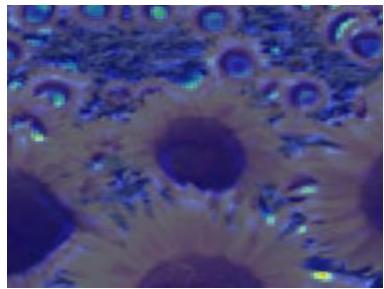
17.0



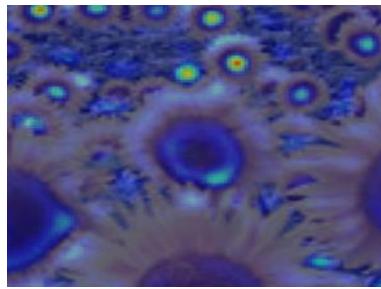
3/4 size image

optimal scale

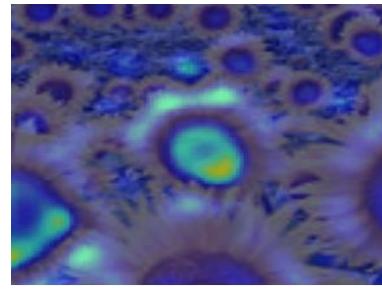
2.1



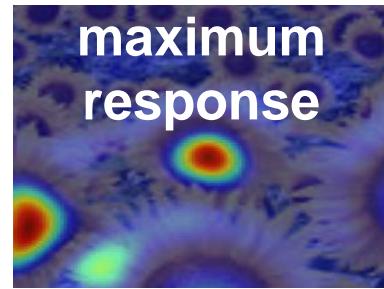
4.2



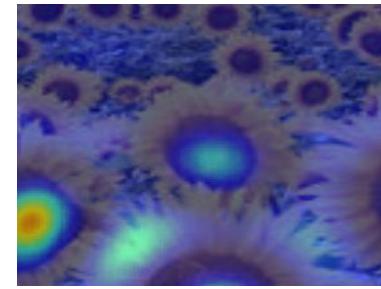
6.0



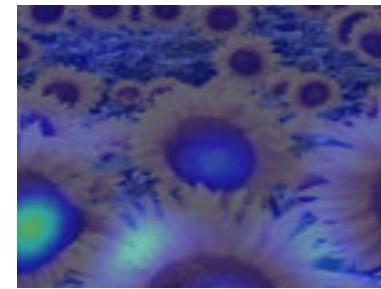
9.8



15.5

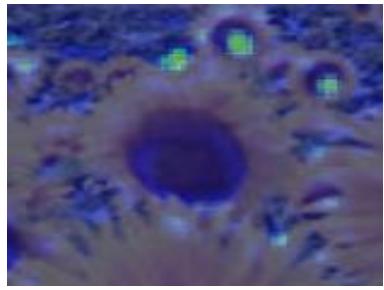


17.0

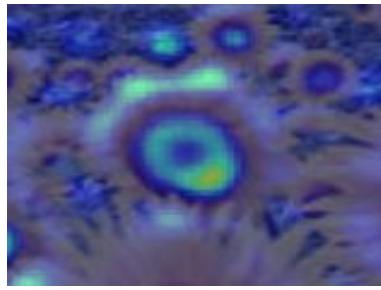


Full size image

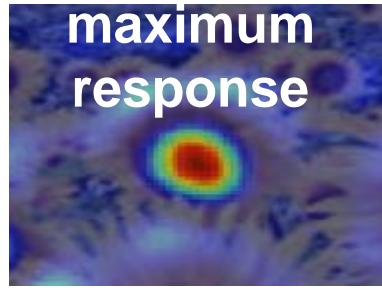
2.1



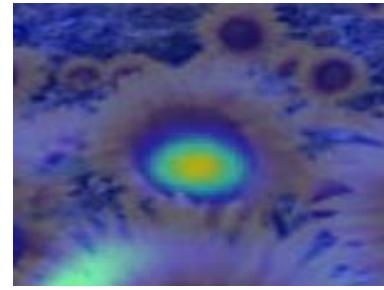
4.2



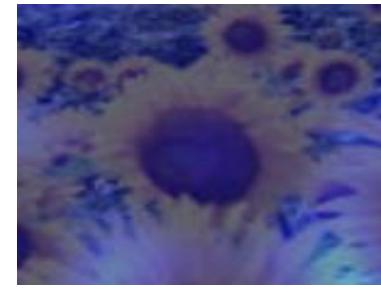
6.0



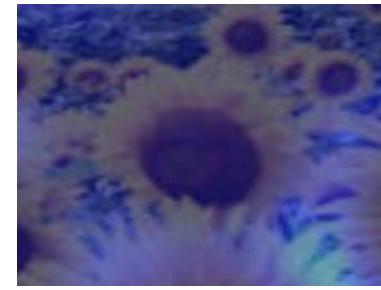
9.8



15.5

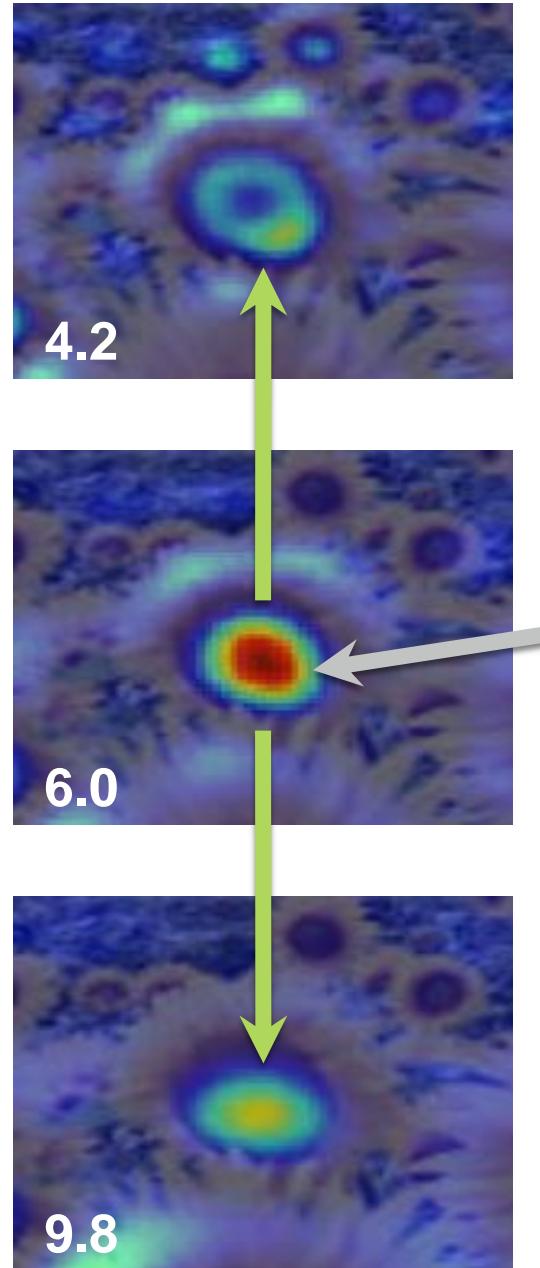


17.0



3/4 size image

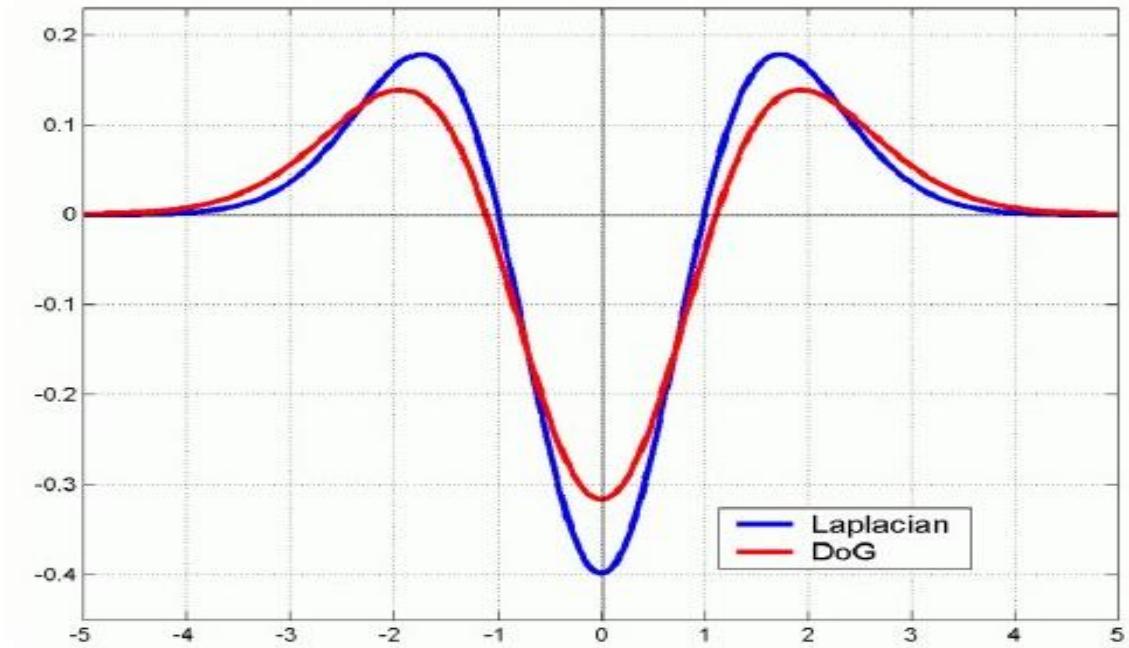
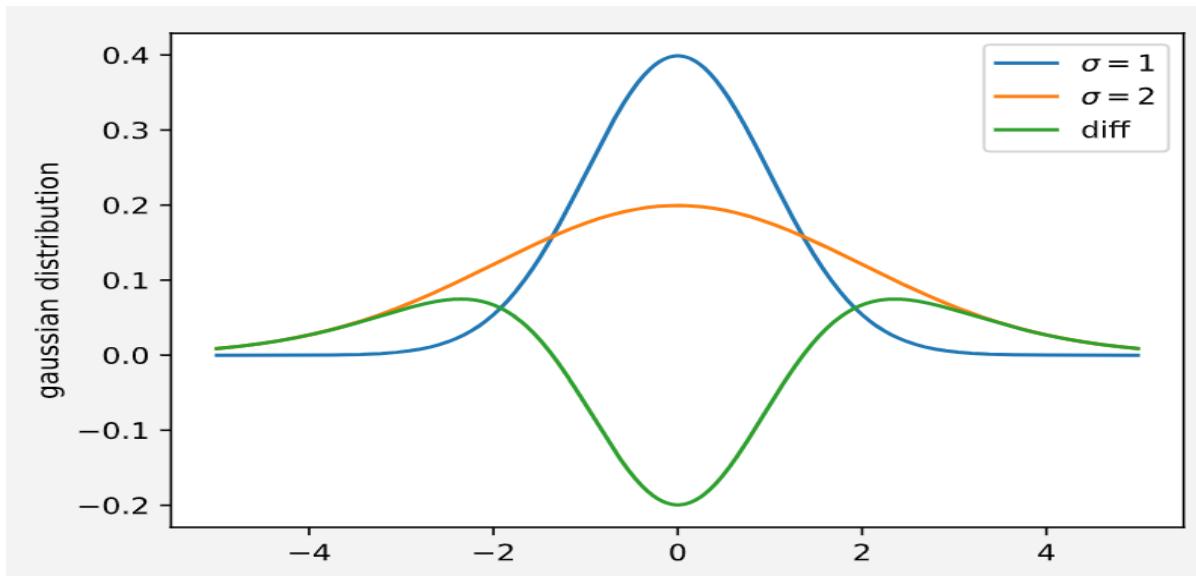
cross-scale maximum



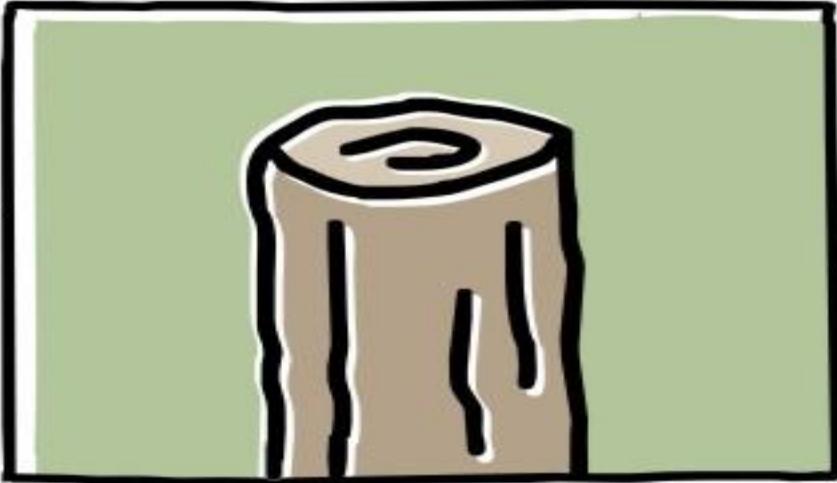
local maximum

As seen in class- edges: DoG

- Can also use difference of Gaussians (DoG) to mimic LoG.
- Why do we want to do this? Faster computationally (explained here: <https://dsp.stackexchange.com/a/37675>)



log vs. dog



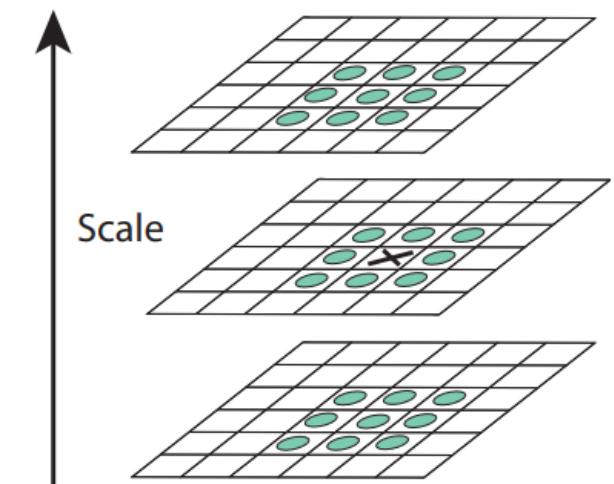
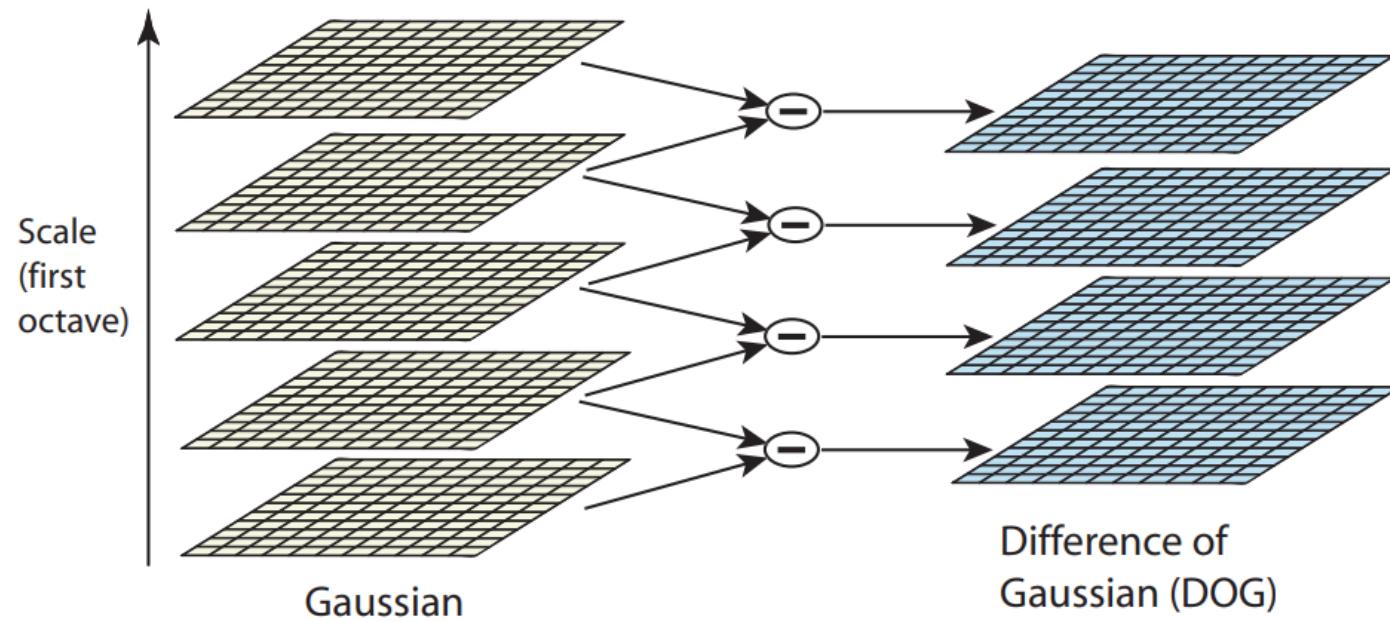
likes to be outside
found near the fireplace
gives warmth and comfort
plays dead
bark
doesn't have a tail



likes to be outside
found near the fireplace
gives warmth and comfort
plays dead
bark
can't be made into shelves

Blob detection algorithm

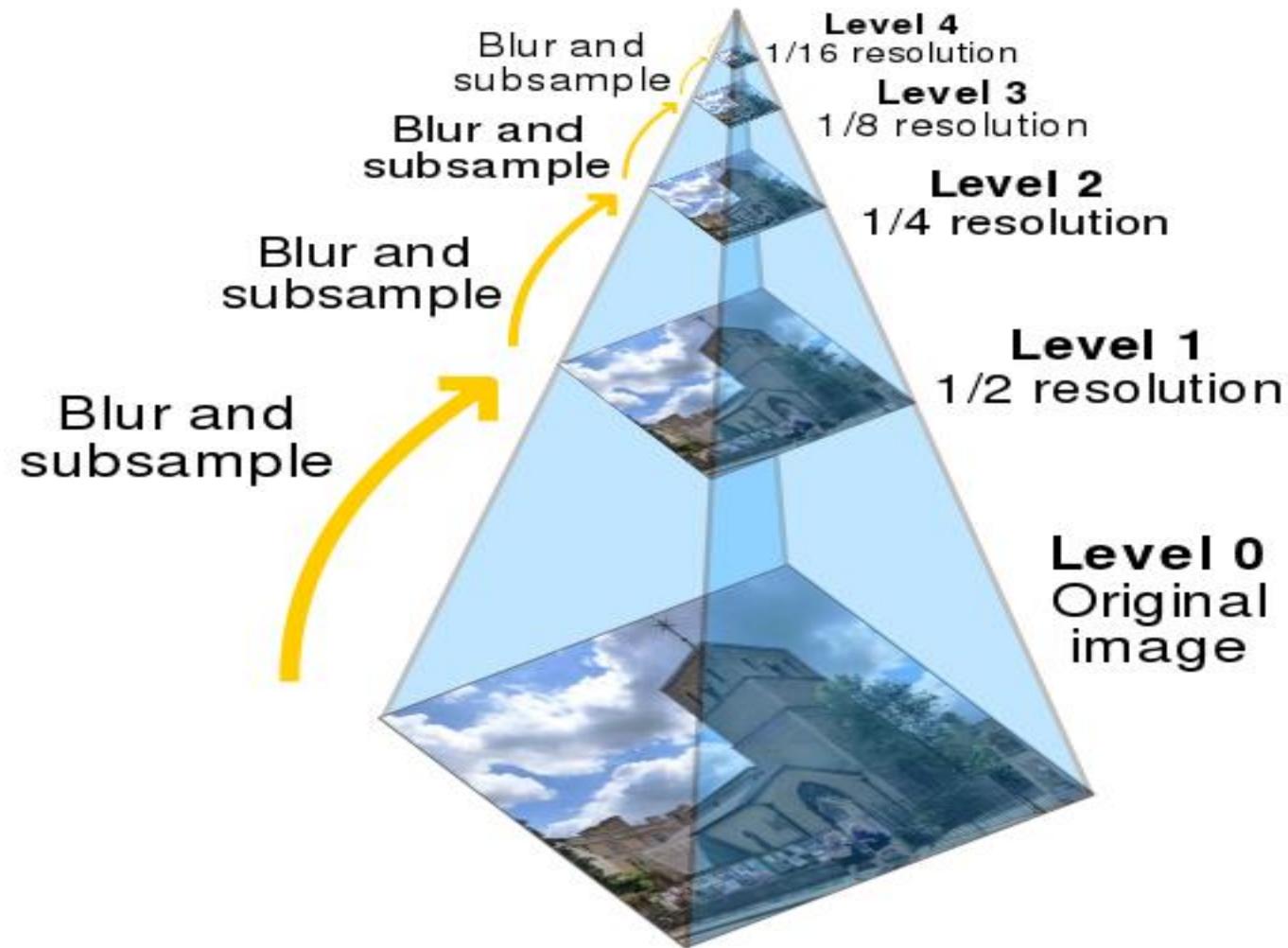
- Build DoG images.
- Search across different image scales the max response.



Blob detection: summary

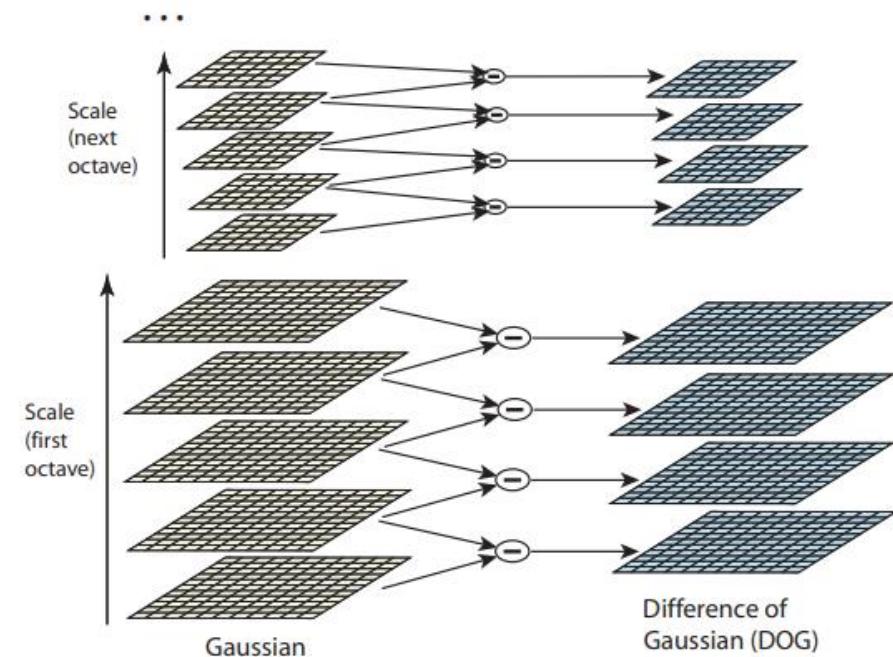
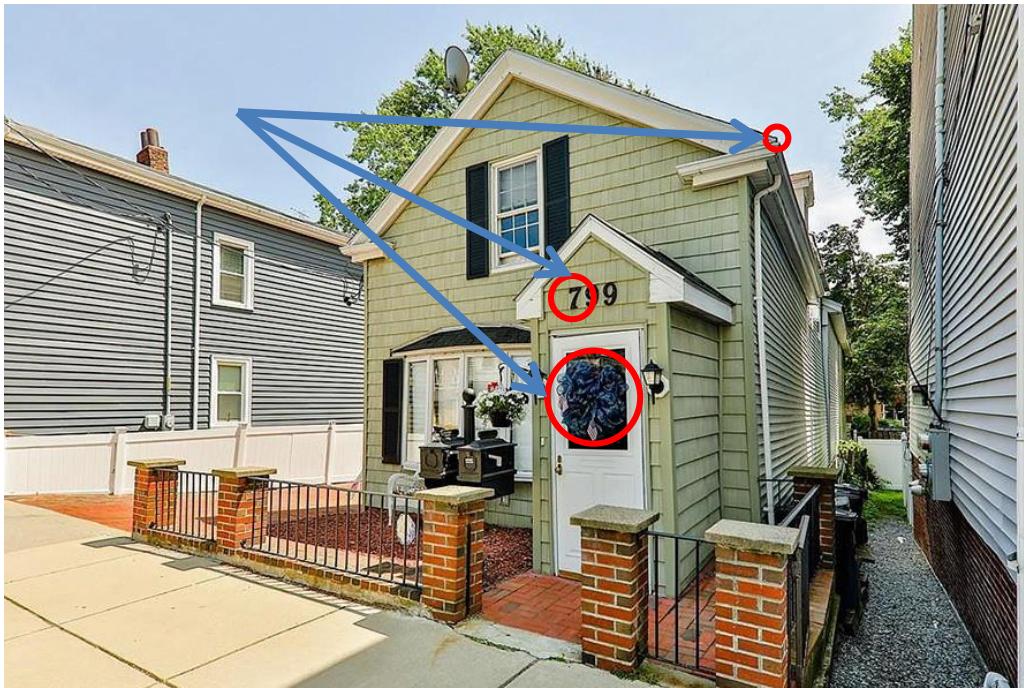
- Advantages:
 - Invariant to translation, rotation, scale and intensity shift $I \rightarrow I + b$ (because we use only the derivatives).
- Disadvantages:
 - Can also find edges and not only corners (why is it bad? Next).

Reminder: Gaussian pyramid



Improved LoG blob detector

- Since the images is low-passed filter so much, we can decimate the image and not loose data in the process!
- We can build a Gaussian pyramid (as taught in image processing recap class) and run the entire algorithm on the different **octave scales**.
- **When using different octave scales we can get features from different image scale.**



contents

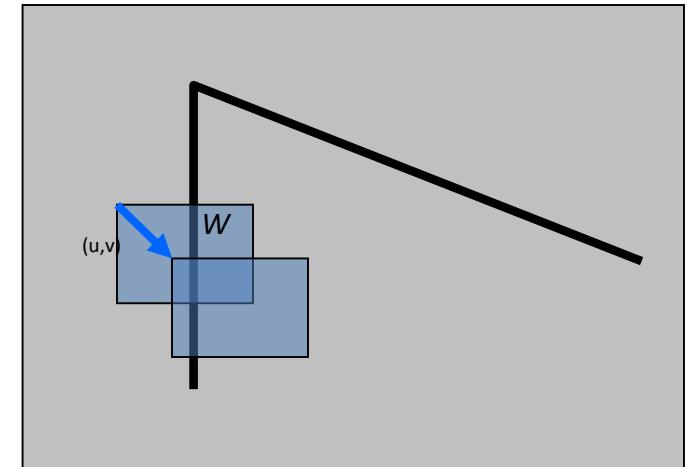
- What and why we need features detection?
- Feature detection
 - Blob detection
 - **Harris corner detection**
 - SIFT detector
- Feature description
 - HOG
 - SIFT descriptor
- SIFT feature matching
- Panoramas

Harris corner detection

- Consider shifting the window W by (u, v)
 - compare each pixel before and after by summing up the squared differences (SSD).
 - this defines an SSD “error” $E(u, v)$:

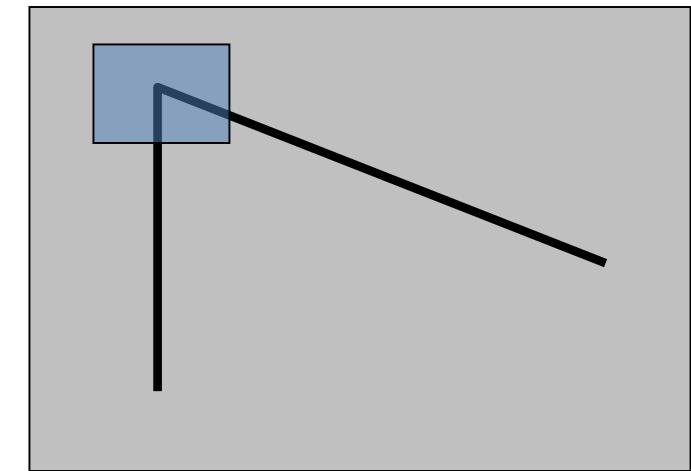
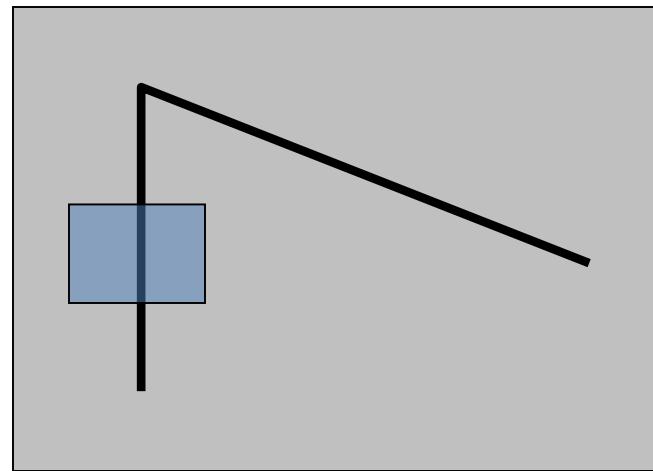
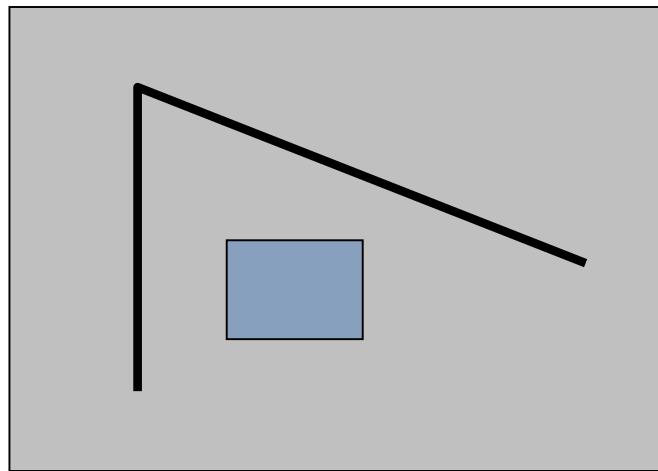
$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

- **We are happy if this error is high for all $(u, v) \neq (0, 0)$**



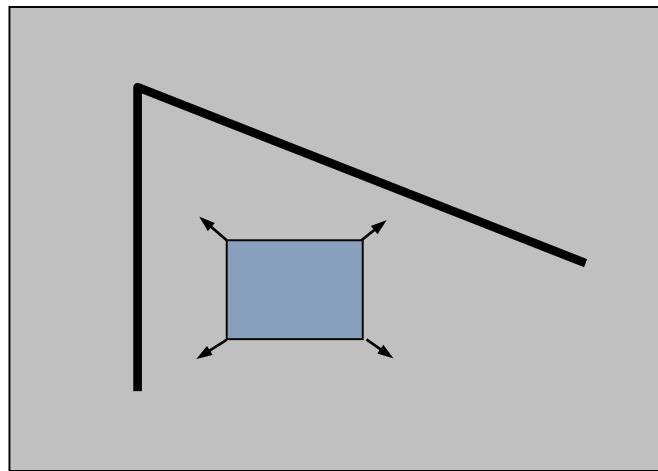
Local measures of uniqueness

- Suppose we only consider a small window of pixels.
- How does the window change when you shift it?

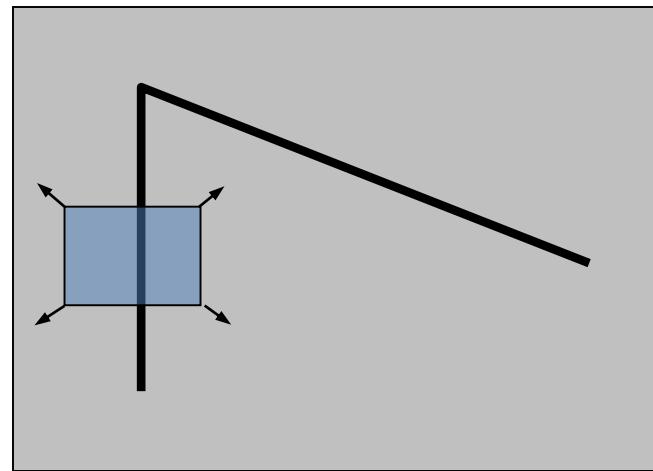


Local measures of uniqueness

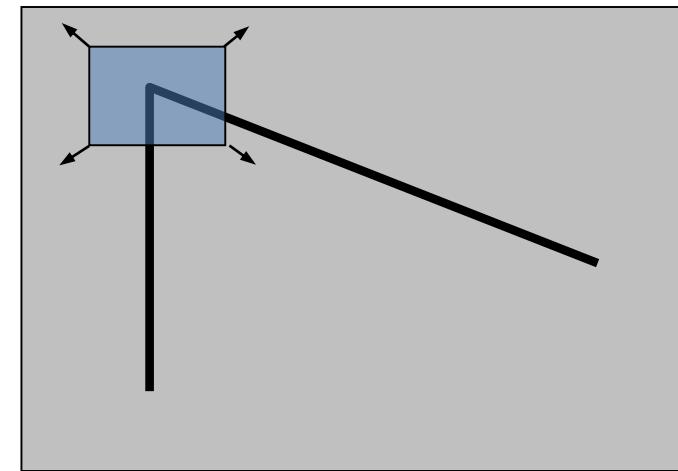
- Suppose we only consider a small window of pixels-
- How does the window change when you shift it?



“flat” region:
no change in all
directions



“edge”:
no change along the edge
direction



“corner”:
significant change in all
directions

Harris corner detection

- Taylor Series expansion of I :

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

- If the motion (u, v) is small, then first order approximation is good

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

- Plug it into the SSD error term:

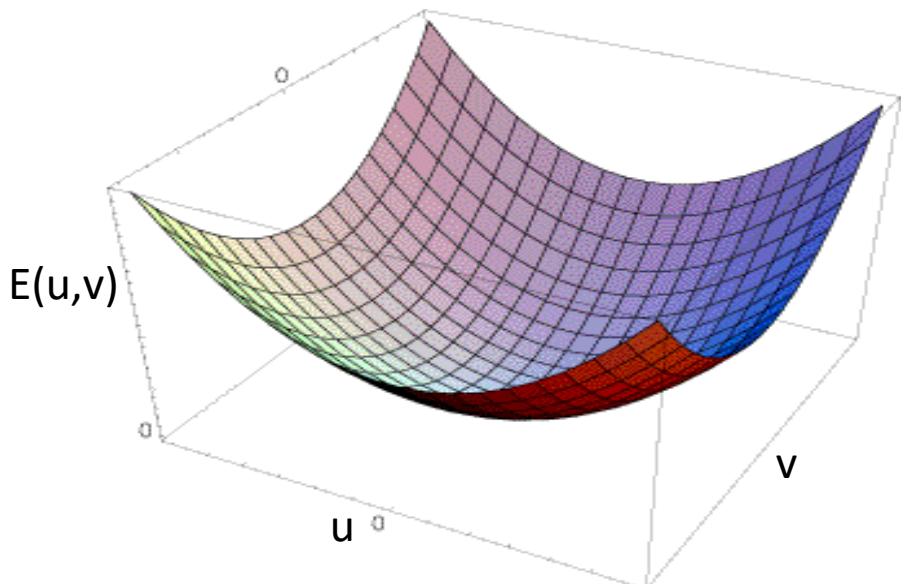
$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I(x, y) + I_x u + I_y v - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I_x u + I_y v]^2 \end{aligned}$$

Harris corner detection

$$\begin{aligned} E(u, v) &\approx \sum_{(x,y) \in W} [I_x u + I_y v]^2 \\ &\approx A u^2 + 2Buv + C v^2 \\ &\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

Also called **second-moments matrix**

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



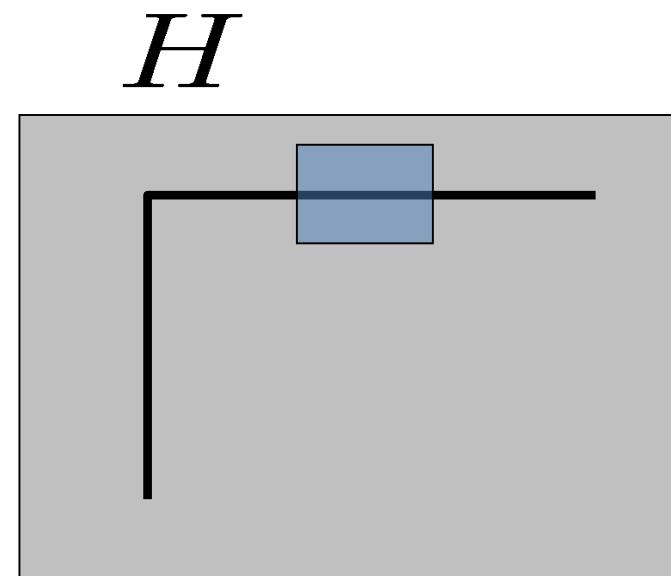
Harris corner detection

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

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

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

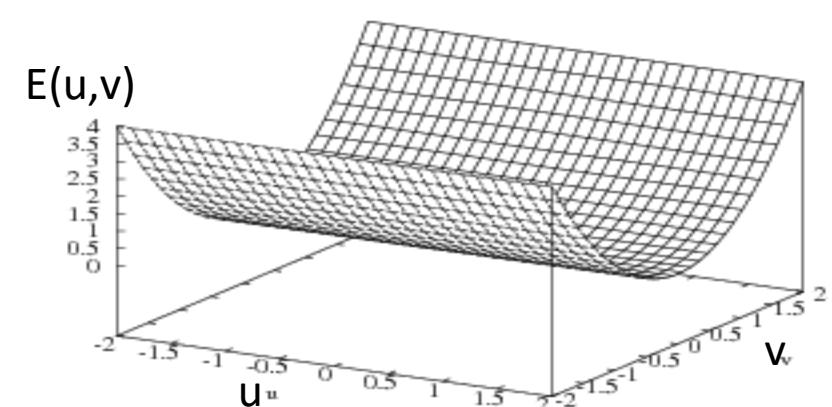
$$C = \sum_{(x,y) \in W} I_y^2$$



Horizontal edge:

$$I_x = 0$$

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



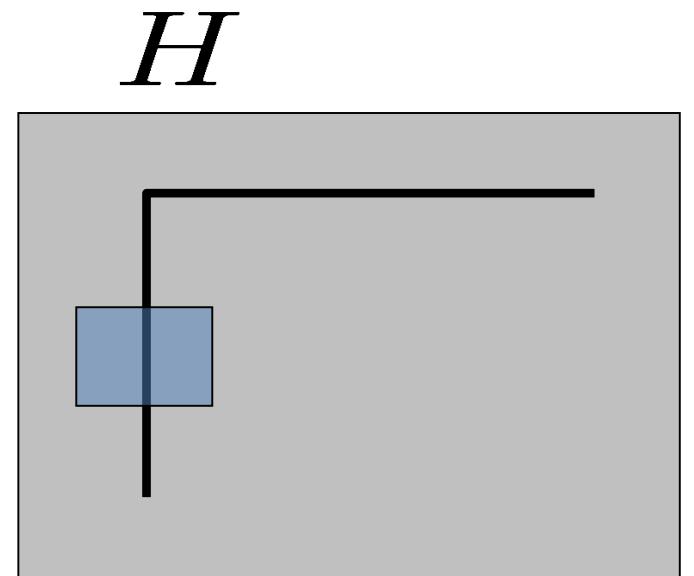
Harris corner detection

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

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

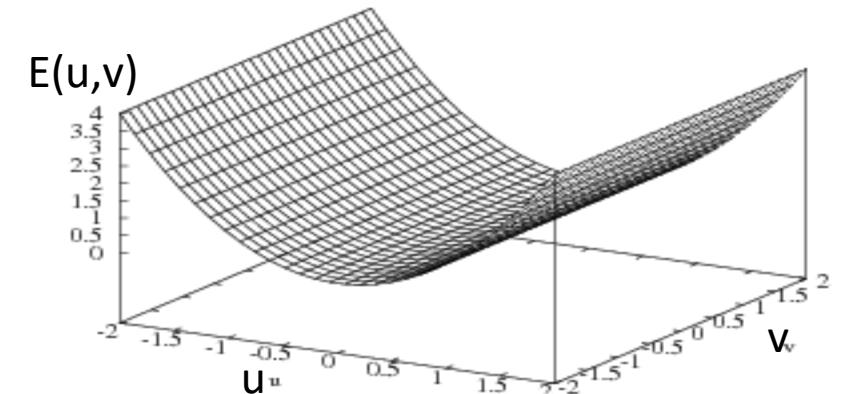
$$B = \sum_{(x,y) \in W} I_x I_y$$

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



Vertical edge: $I_y = 0$

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



Harris corner detection: probabilistic interpretation

- A real symmetric matrix has an eigendecomposition of:

$$Av = \lambda v$$

$$AQ = Q\Lambda$$

$$A = Q\Lambda Q^{-1}$$

A is real symmetric matrix

$$\xrightarrow{} A = Q\Lambda Q^T$$

$$A = \begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix}$$

- Bonus: eigenvectors are orthonormal if A is real and symmetric.

Harris corner detection

- An ellipse can have a matrix form of:

$$x^T (e_1 e_2) \begin{pmatrix} \lambda_1 & \\ & \lambda_2 \end{pmatrix} \begin{pmatrix} e_1^T \\ e_2^T \end{pmatrix} x = 1$$

$$\lambda_1 x^T e_1 e_1^T x + \lambda_2 x^T e_2 e_2^T x = 1$$

$$\frac{(e_1^T x)^2}{\left(\frac{1}{\sqrt{\lambda_1}}\right)^2} + \frac{(e_2^T x)^2}{\left(\frac{1}{\sqrt{\lambda_2}}\right)^2} = 1$$

- Which is exactly as a rotated ellipse with a center of (0,0):

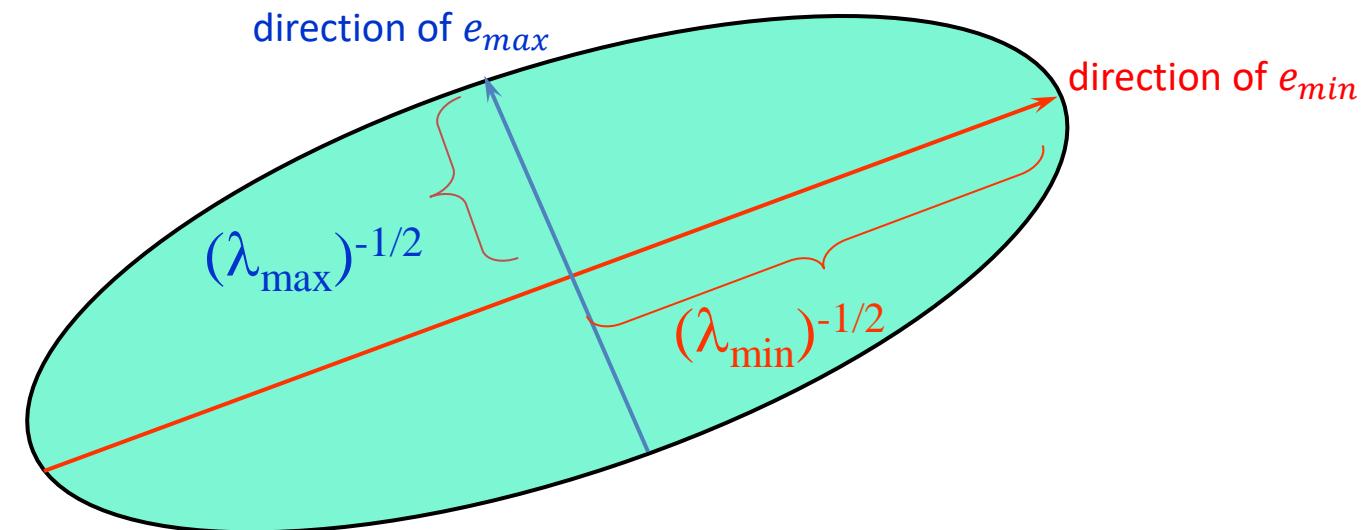
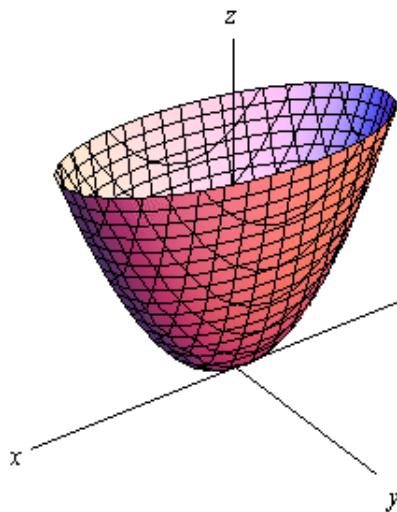
$$\frac{(x \cos(\theta) + y \sin(\theta))^2}{a^2} + \frac{(x \sin(\theta) - y \cos(\theta))^2}{b^2} = 1$$

Harris corner detection

- Combining the two equations seen before we can conclude that when taking a cross-section from the error function, we can get an ellipsoid.

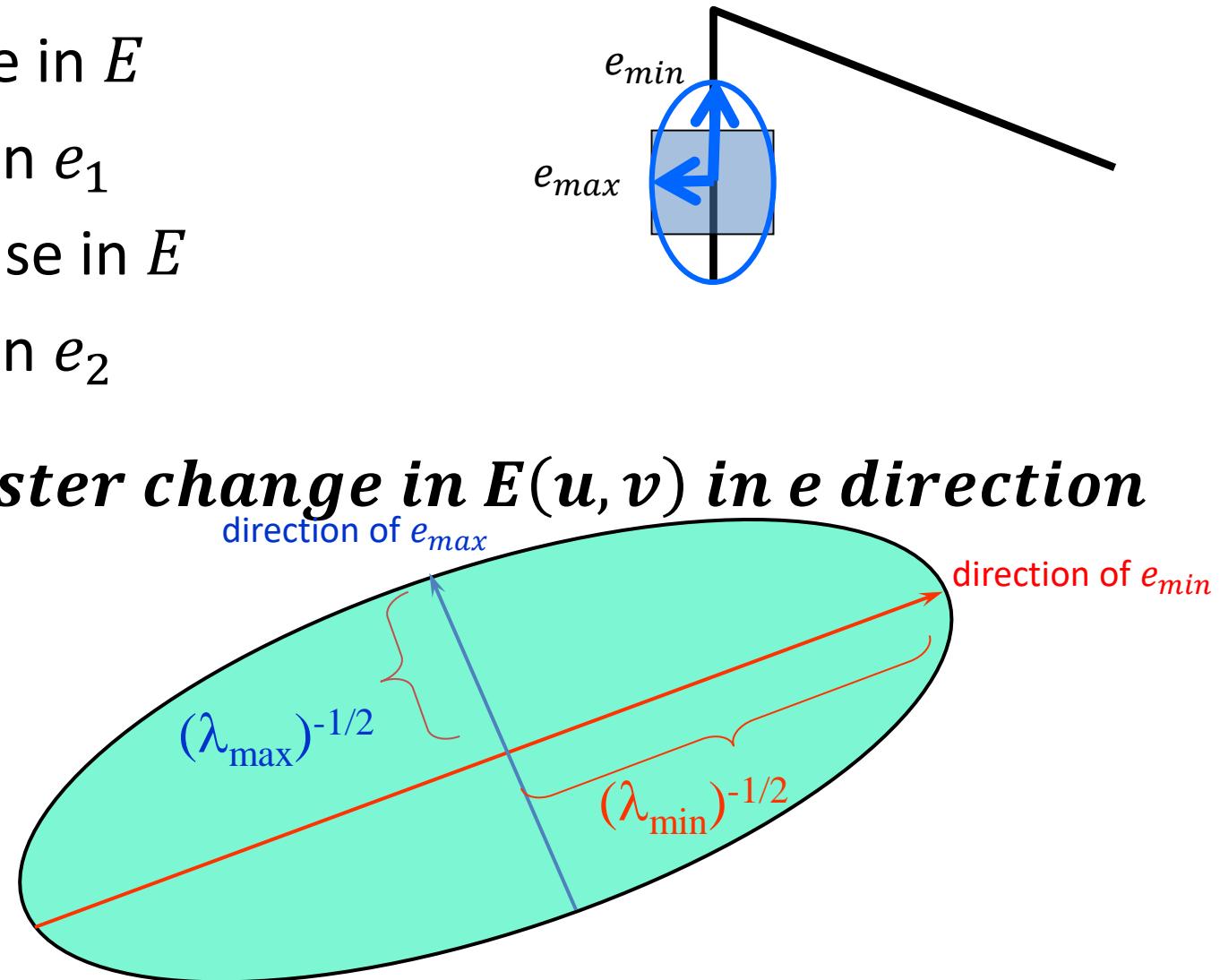
$$-[u \ v]H \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

- Assume $\lambda_1 > \lambda_2$



Harris corner detection

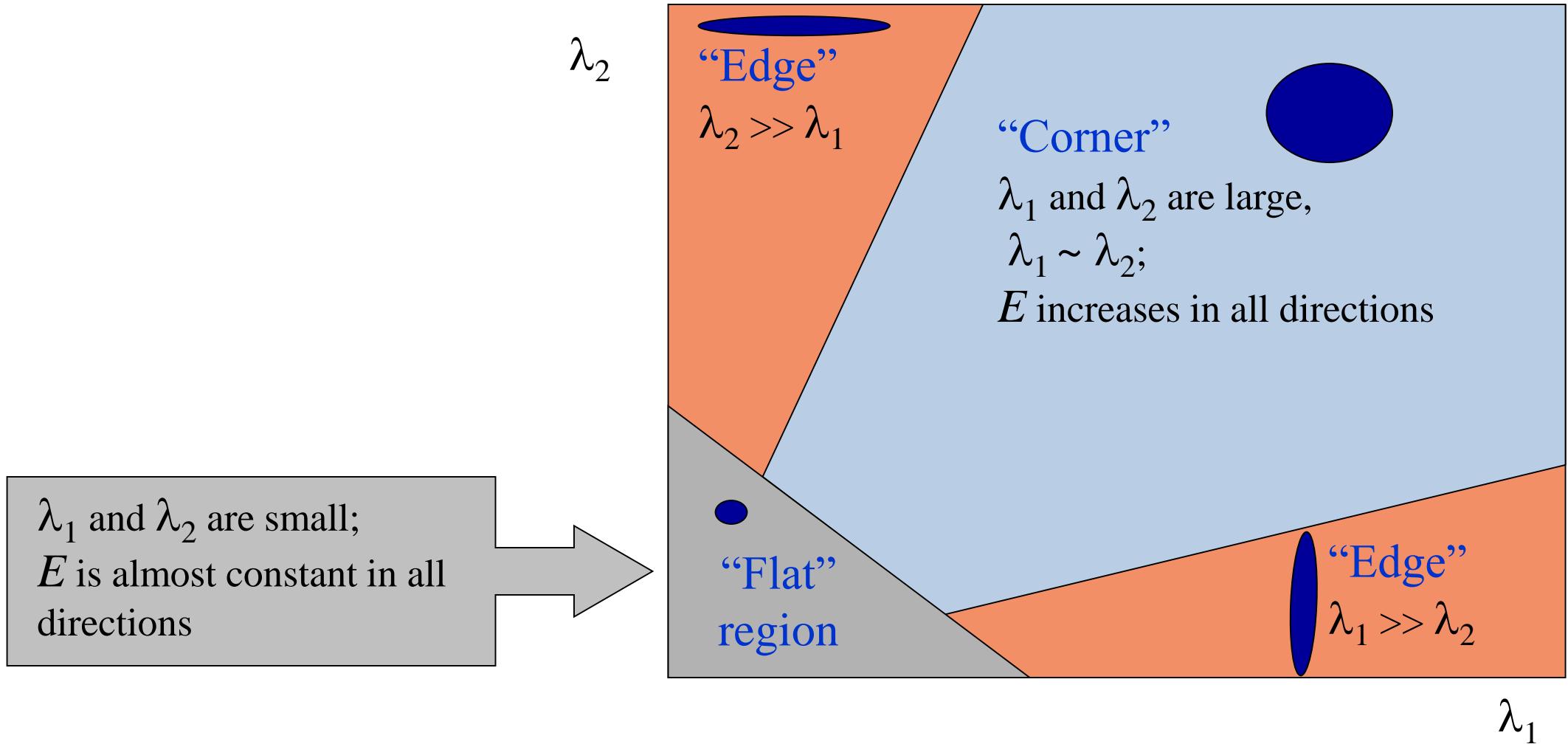
- Eigenvalues and eigenvectors of H
 - e_1 = direction of largest increase in E
 - λ_1 = relative increase in direction e_1
 - e_2 = direction of smallest increase in E
 - λ_2 = relative increase in direction e_2
- λ larger $\leftrightarrow \lambda^{-\frac{1}{2}}$ smaller \leftrightarrow faster change in $E(u, v)$ in e direction



Interpreting the eigenvalues

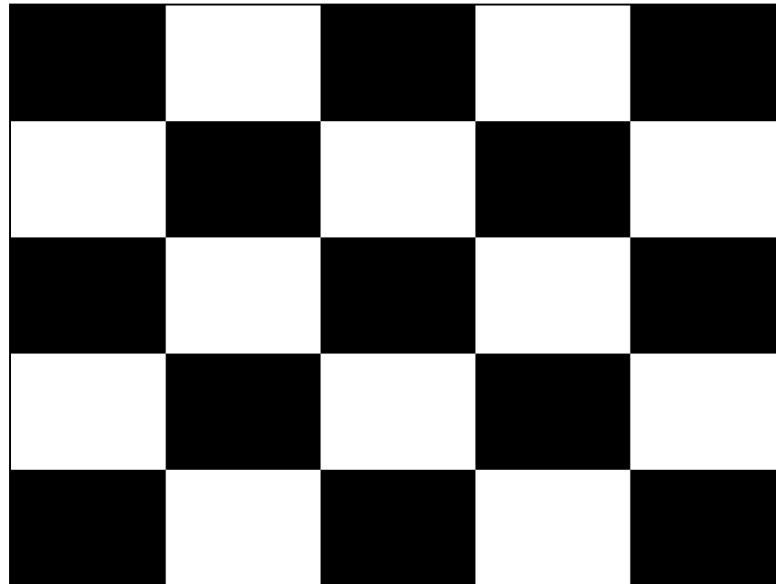
- A “good” corner will have a large $R = \lambda_{min}$, which means big change of E in both axis.
- Getting the eigenvectors and eigenvalues is computationally inefficient.
- Instead, use two tricks:
 - $\prod_i \lambda_i = \det(A)$
 - $\sum_i \lambda_i = \text{trace}(A)$
- Then we can more easily compute R :
 - $R = \det(A) - \kappa * \text{trace}(A)^2 \quad (\kappa \in [0.04, 0.06])$
 - $R = \frac{\det(A)}{\text{trace}(A)+\epsilon} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2 + \epsilon}$

Interpreting the eigenvalues

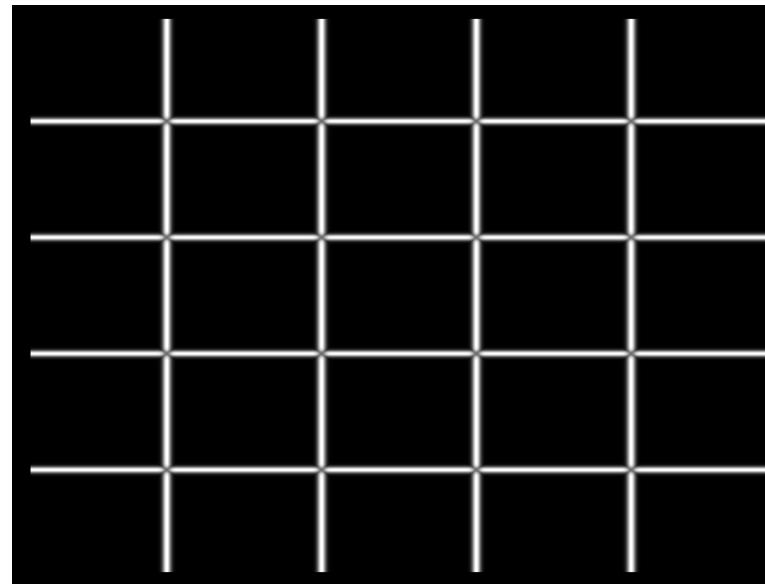


Interpreting the eigenvalues

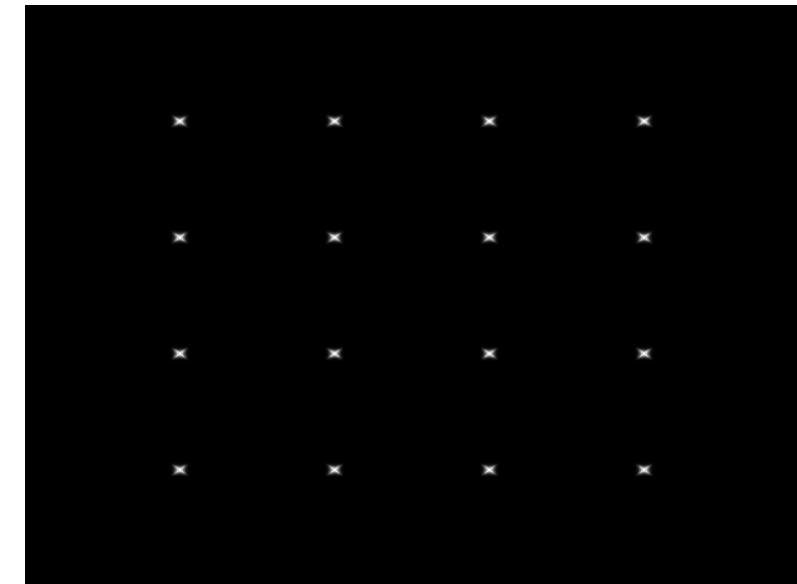
- A binary threshold of pixels above λ_{max} and λ_{min}



I



λ_{max}



λ_{min}

Harris corner detection

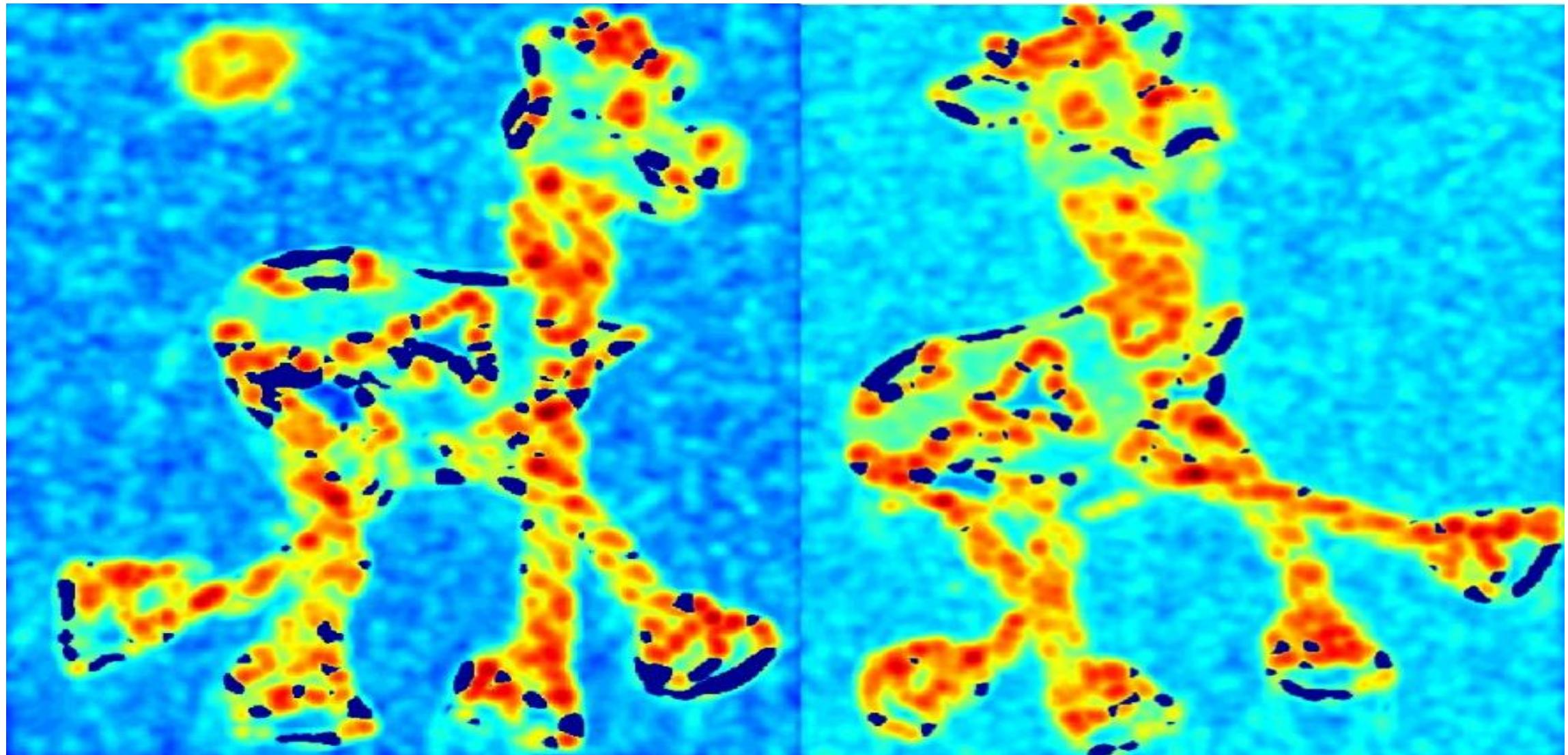
- Compute gradients of patch around each pixel.
- ~~Subtract the mean from each patch gradient.~~
- Compute the second-moment matrix.
- Compute eigendecomposition of covariance matrix.
- Use eigenvalues to find corners.

This is PCA (principal component analysis). Out of scope but really interesting!

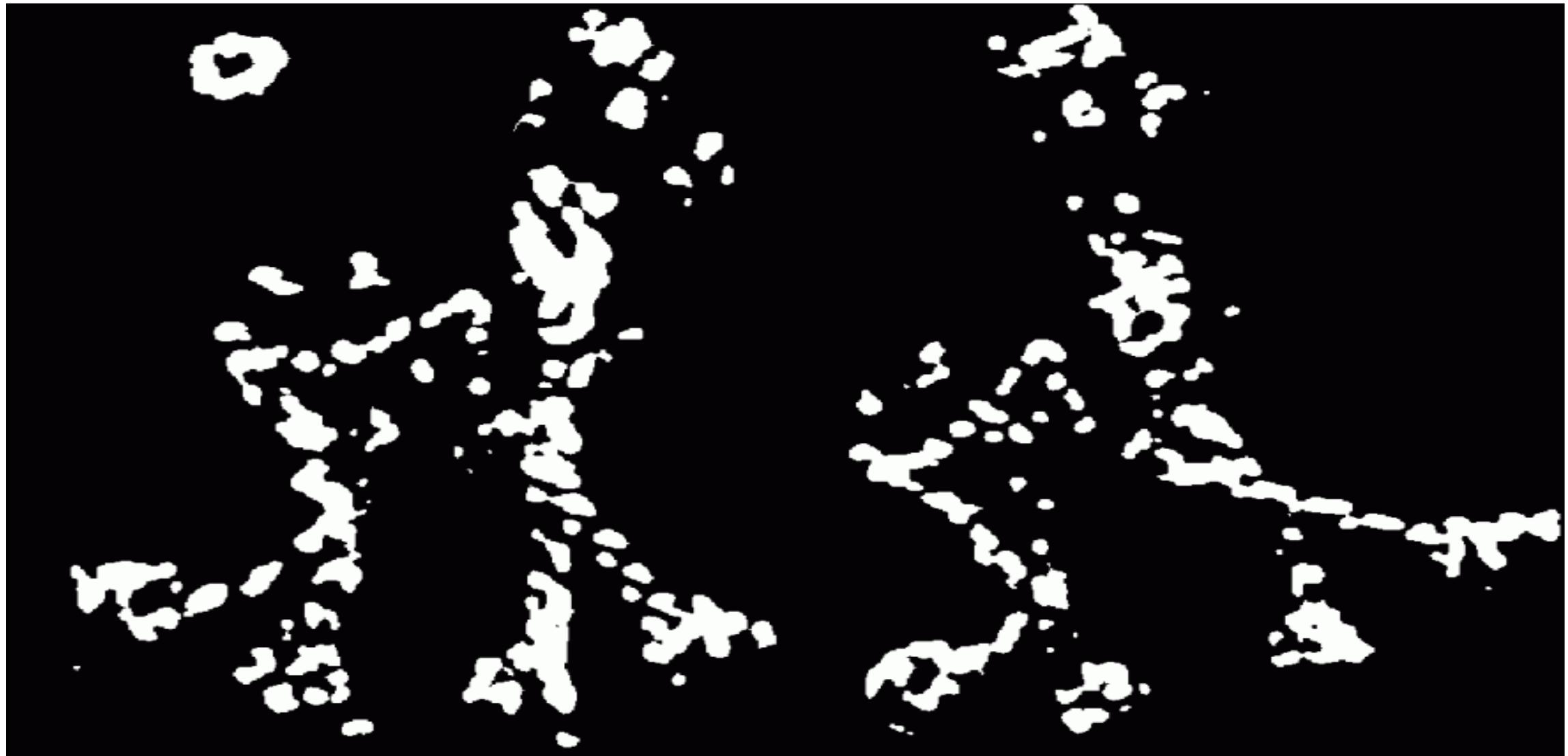
Harris detector example



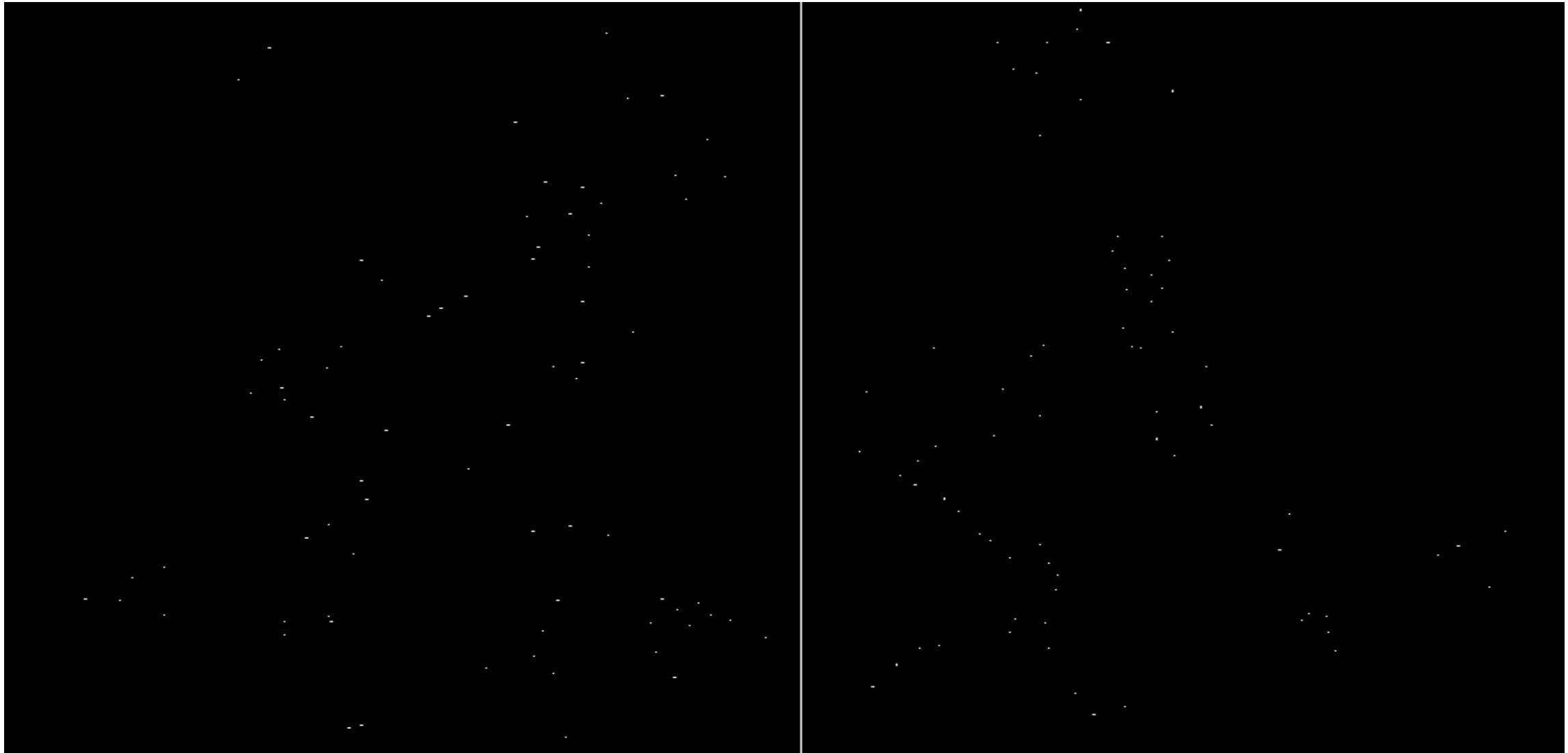
f value (red high, blue low)



Threshold ($f > \text{value}$)



Find local maxima of f



Harris features (in red)



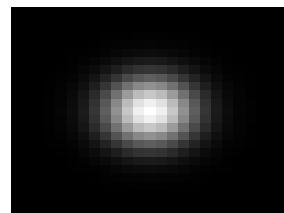
Weighting the derivatives

- In practice, using a simple window W doesn't work too well

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

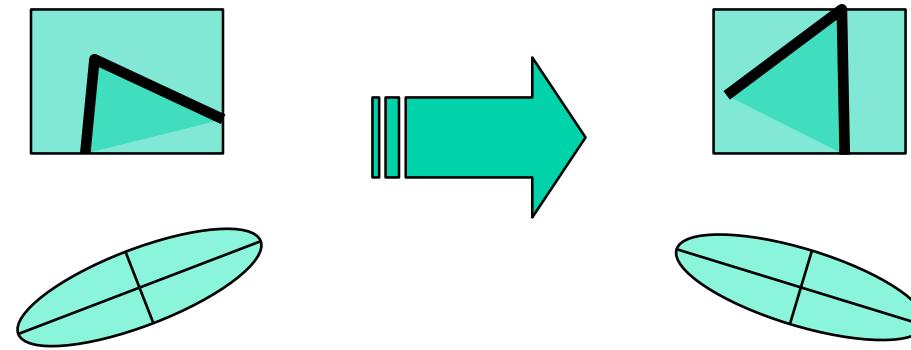
- Instead, we'll *weight* each derivative value based on its distance from the center pixel

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



$w_{x,y}$

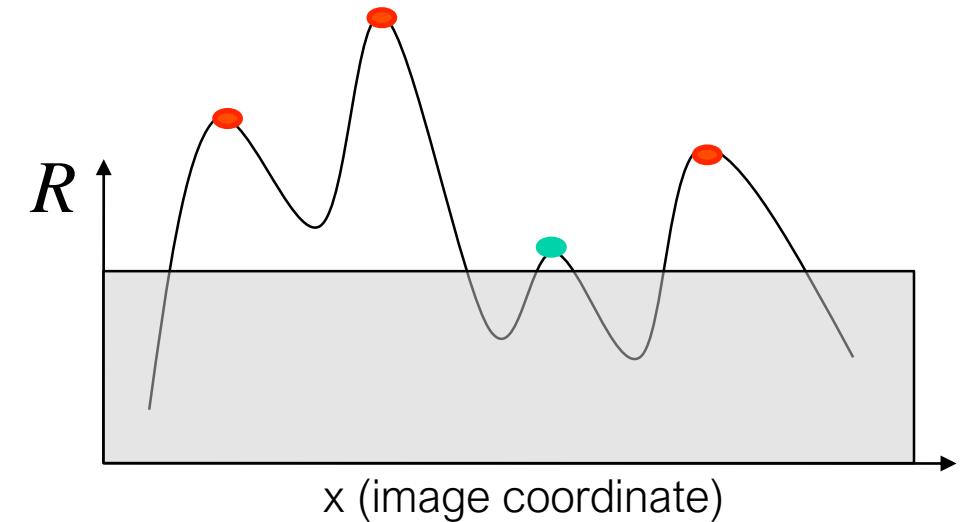
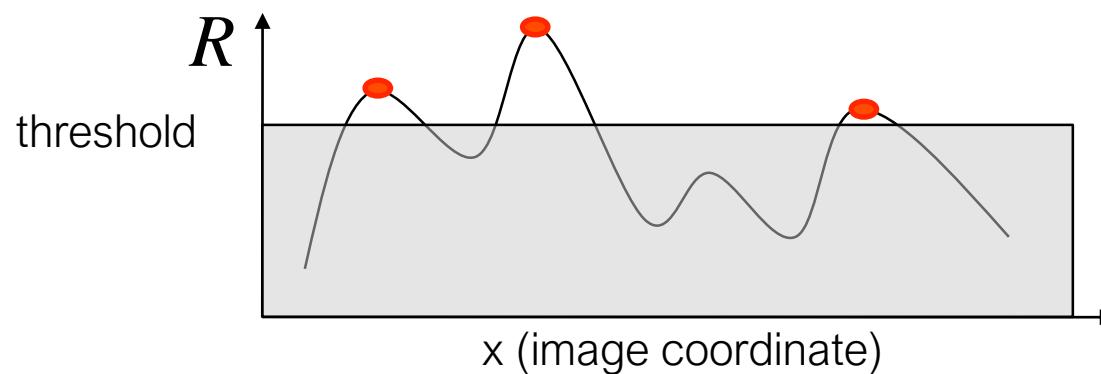
Harris corner detector- rotation and translation



- Ellipse rotates but its shape (**eigenvalues**) remains the same => invariant to rotation!
- The feature is also translation invariant (easy to see).

Harris corner detector- intensity

- Partial invariance to *affine intensity* change
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Not completely invariance to Intensity scale: $I \rightarrow a \cdot I$



The Harris corner detector is not invariant to scale



contents

- What and why we need features detection?
- Feature detection
 - Blob detection
 - Harris corner detection
 - **SIFT detector**
- Feature description
 - HOG
 - SIFT descriptor
- SIFT feature matching
- Panoramas

SIFT keypoint detection

- Find blobs using the improved blob detection (across different octave scale).
- Use interpolation to find exact peak of keypoint.
 - The interpolation takes place in **x, y and scale dimension.**
- Eliminate edge response with Harris corner detector variant (called **principal curvature**) around temp keypoints in interpolated space and scale.
- SIFT has the advantages of both previous technics and is invariance to: rotation, translation, scale, illumination shift and partially to 3D change of viewpoint since it is a local keypoint detector.

contents

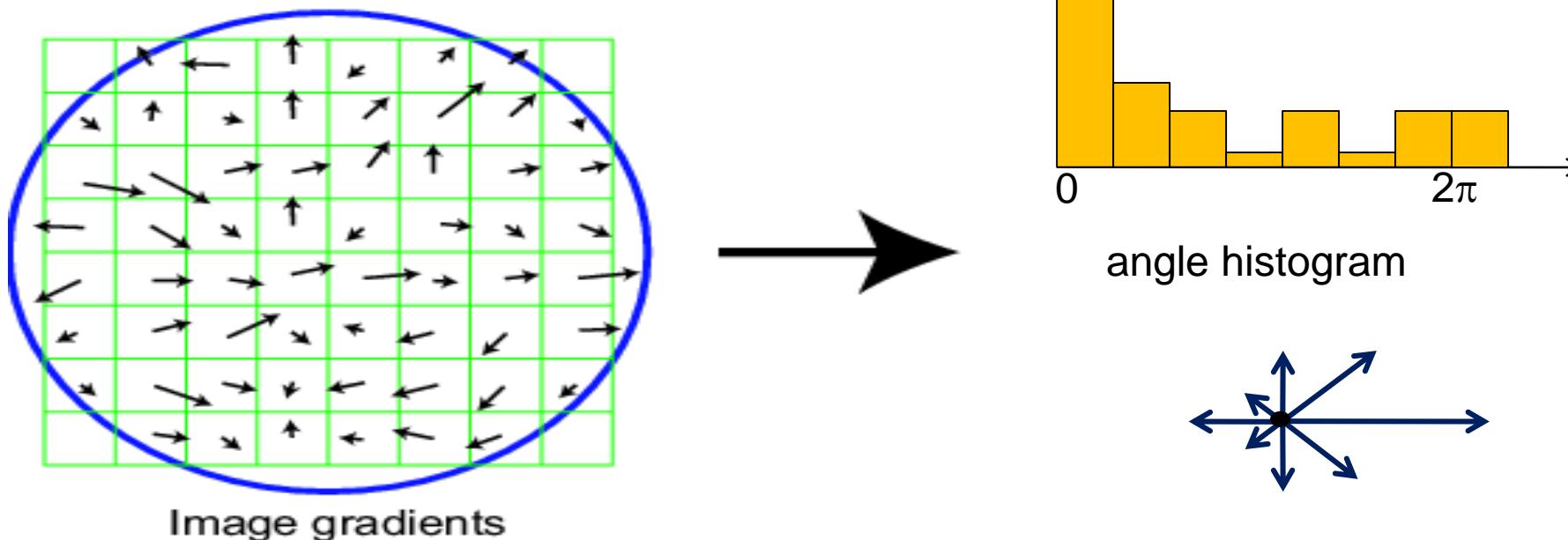
- What and why we need features detection?
- Feature detection
 - Blob detection
 - Harris corner detection
 - SIFT detector
- **Feature description**
 - HOG
 - SIFT descriptor
- SIFT feature matching
- Panoramas

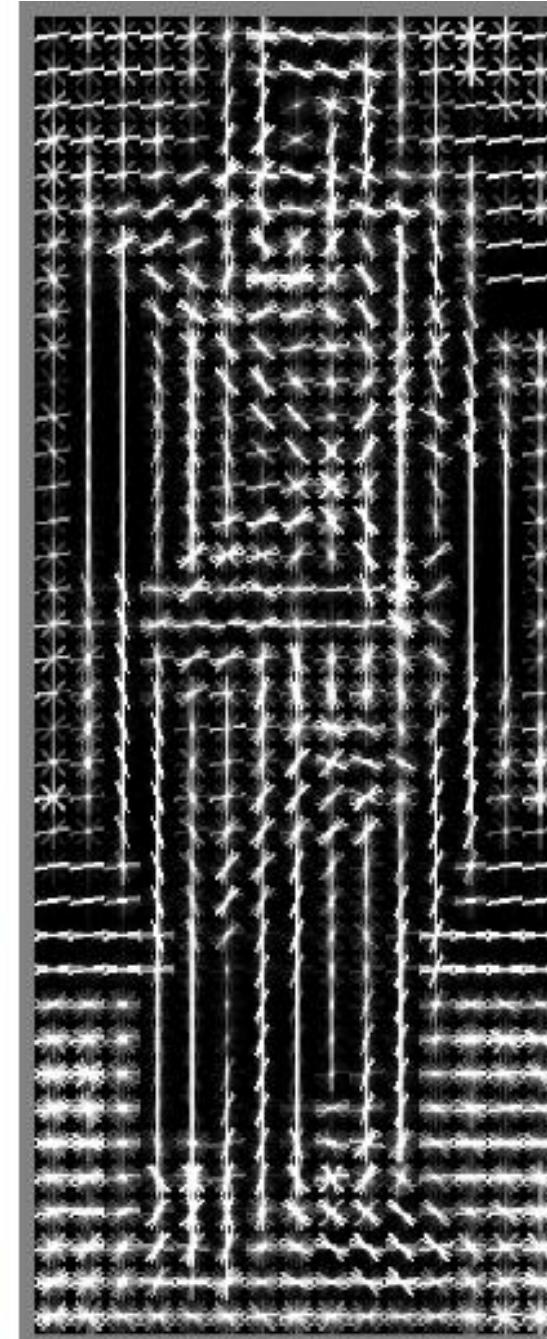
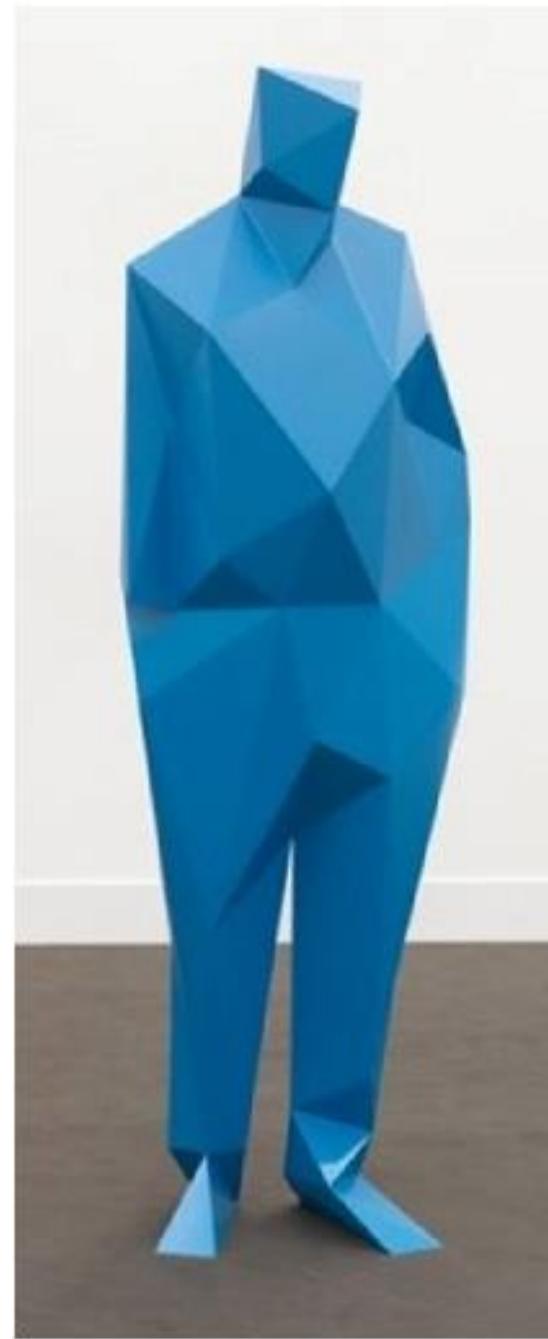
HOG- Histogram of Oriented Gradients

- A dense representation of image as blocks, each with its own histogram of gradient directions, weighted by the gradient magnitude.
- Originally used to detect humans in images.
- Equations for gradient magnitude and orientation:

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

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





contents

- What and why we need features detection?
- Feature detection
 - Blob detection
 - Harris corner detection
 - SIFT detector
- Feature description
 - HOG
 - **SIFT descriptor**
- SIFT feature matching
- Panoramas

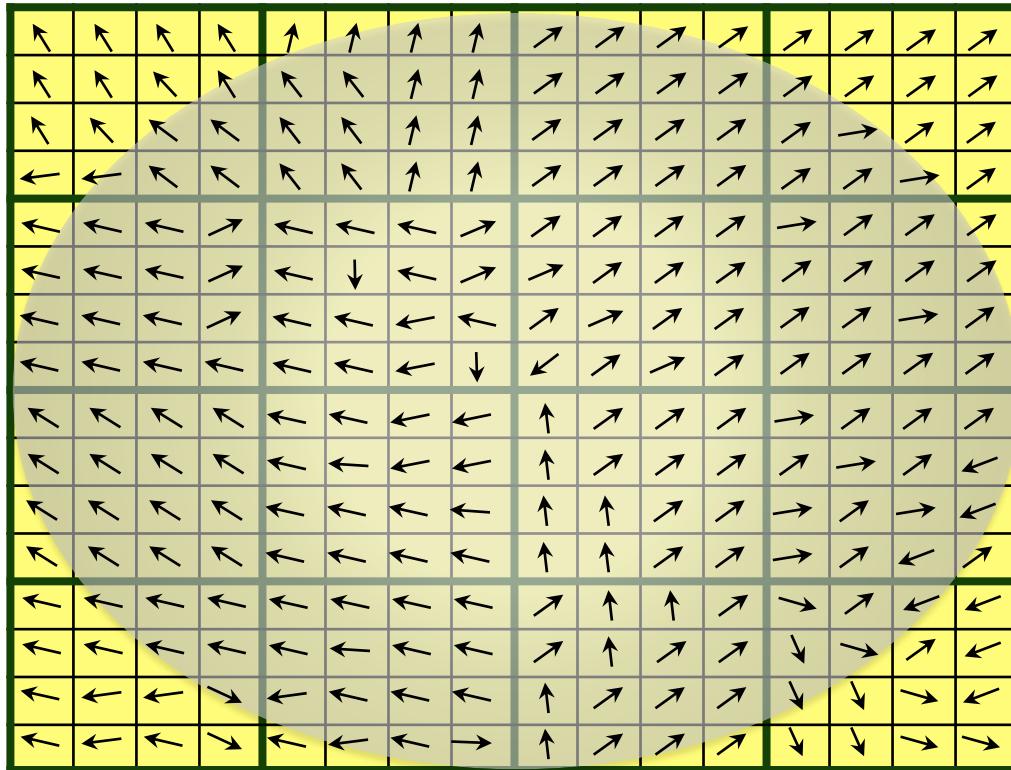
HOG variation as SIFT keypoint descriptor

- First, find the main patch direction using a HOG on all gradients around keypoint at the selected scale, all further calculation is done around this direction (this is how to get a rotation invariance descriptor).
- Take 16x16 patch around detected feature, and calculate gradient orientation and magnitude to each pixel.
- Magnitude is also weighted by a Gaussian around the keypoint.
- Build sub-blocks of 4X4 of the patch.
- Create 8-bin histogram of edge orientations (weighted by Gaussian and magnitude of gradient) to each sub-block.
- Take the $4 \times 4 \times 8 = 128$ results of histograms and concatenate them to a single feature vector. Normalize this vector to be invariance to illumination scale.

HOG variation as SIFT keypoint descriptor

Image Gradients

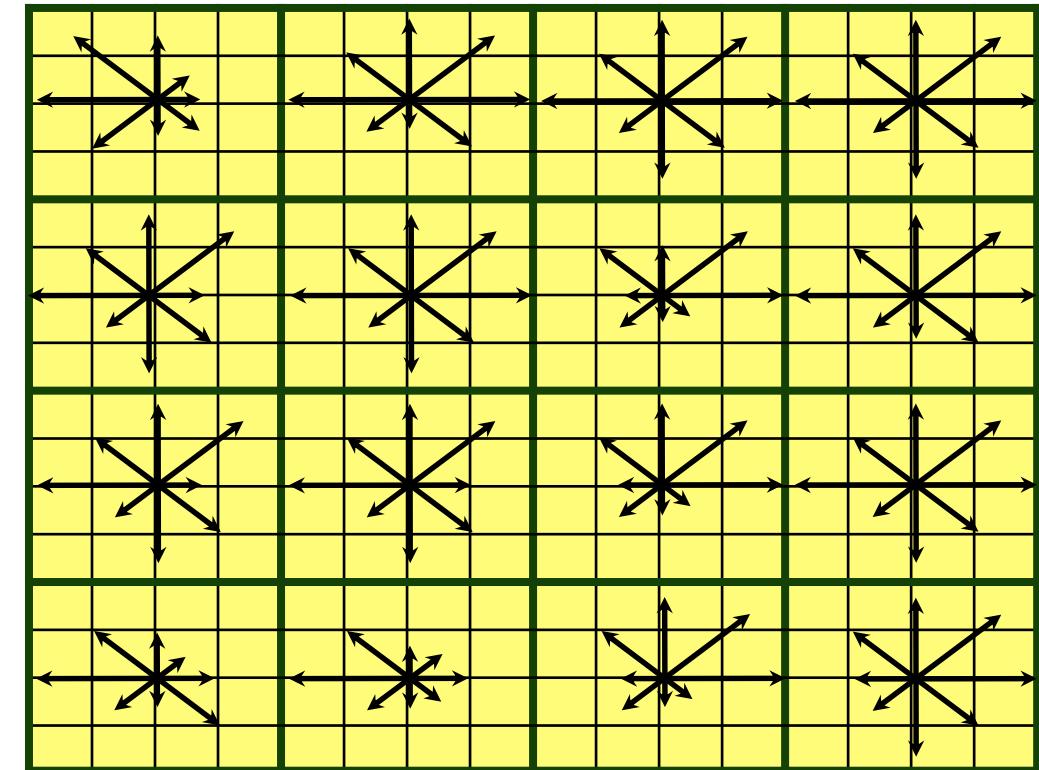
(4 x 4 pixel per cell, 4 x 4 cells)



Gaussian weighting

SIFT descriptor

(16 cells x 8 directions = 128 dims)



HOG variation as SIFT keypoint descriptor



HOG variation as SIFT keypoint descriptor

- Invariant to:
 - Rotation (due to shifting of the histograms around the main direction).
 - Translation (easy to see).
 - Scale (build at a specific scale).
 - Illumination shift (only derivatives are used).
 - Illumination scale (normalize the feature vector).
 - 3D change of view (the descriptor is of local keypoints).

contents

- What and why we need features detection?
- Feature detection
 - Blob detection
 - Harris corner detection
 - SIFT detector
- Feature description
 - HOG
 - SIFT descriptor
- **SIFT feature matching**
- Panoramas

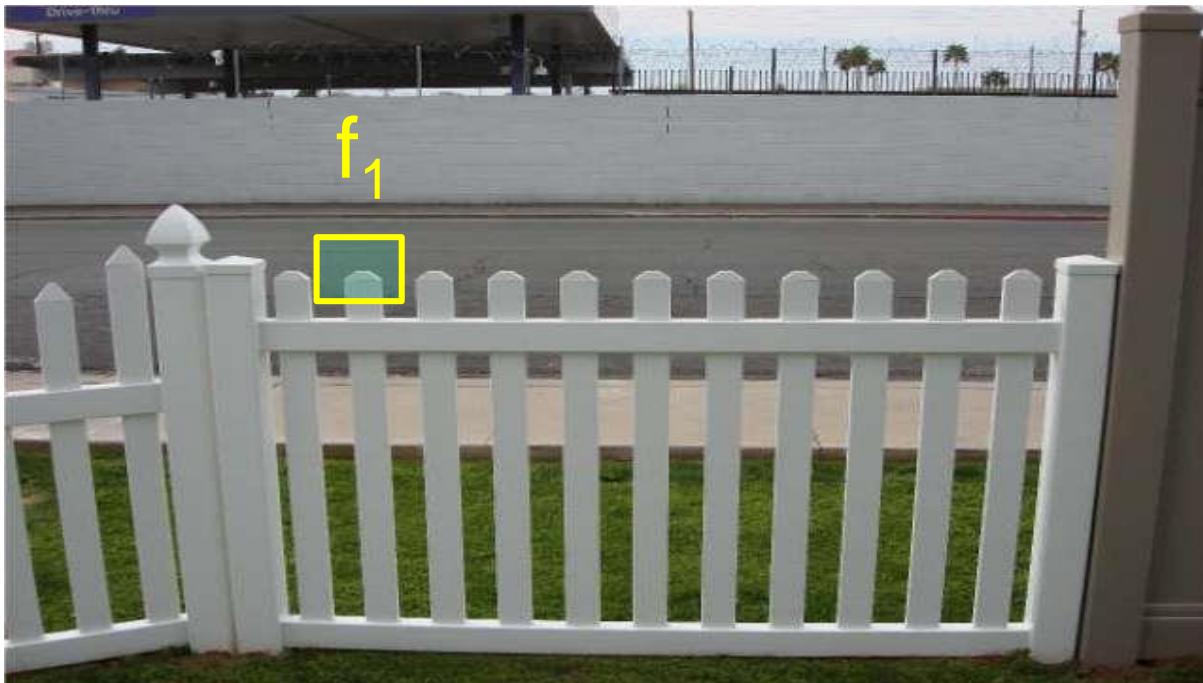
SIFT Feature matching

- 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
- What distance function to use?

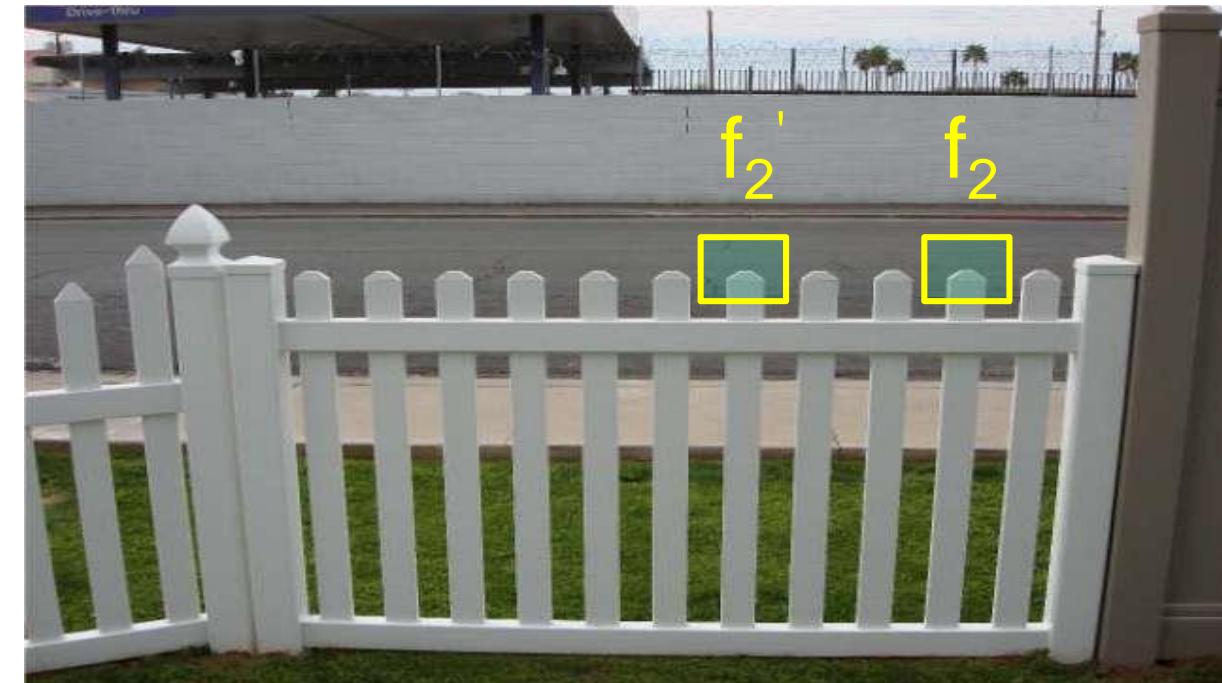
SIFT Feature matching

- What distance function to use?

- Simple approach: L₂ distance, $L_2 = ||f_1 - f_2|| = \sqrt{\sum_i (f_{1i} - f_{2i})^2}$
- can give small distances for ambiguous (incorrect) matches.



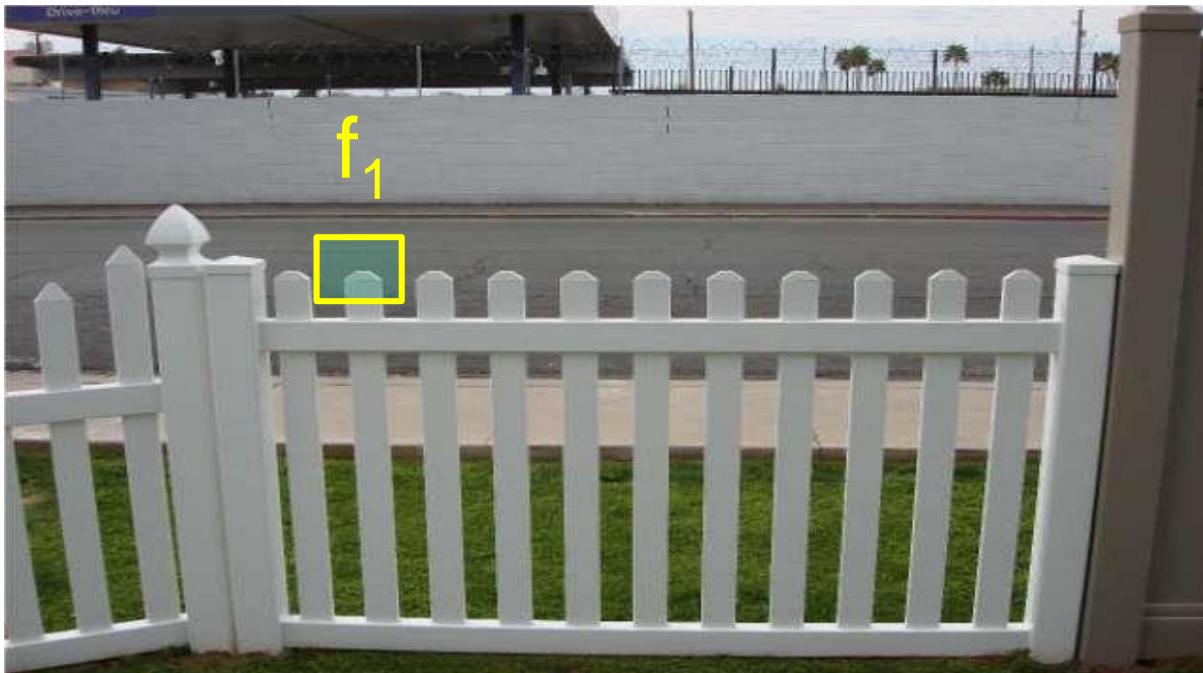
I_1



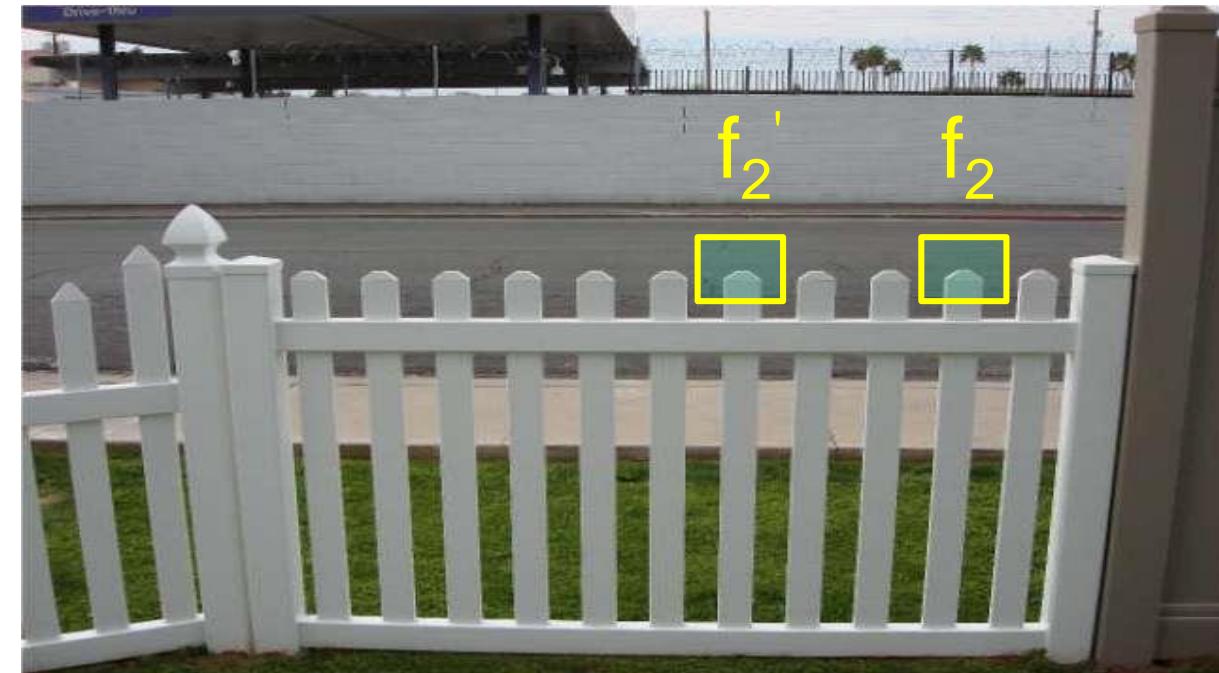
I_2

SIFT Feature matching

- Better approach: ratio distance: $\|f_1 - f_2\| / \|f_1 - f_2'\| < TH$
 - f_2 is best match to f_1 in I_2
 - f_2' is 2nd best match to f_1 in I_2
 - gives larger values for distinct matches.



I_1



I_2

What haven't been covered about SIFT

- Computational fast search of descriptors.
- A lot of minor engineering steps (e.g. thresholding of features).
- And more... this algorithm is 28 pages long article (with 52000 citations!!!)
 - <https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>

contents

- What and why we need features detection?
- Feature detection
 - Blob detection
 - Harris corner detection
 - SIFT detector
- Feature description
 - HOG
 - SIFT descriptor
- SIFT feature matching
- **Panoramas**

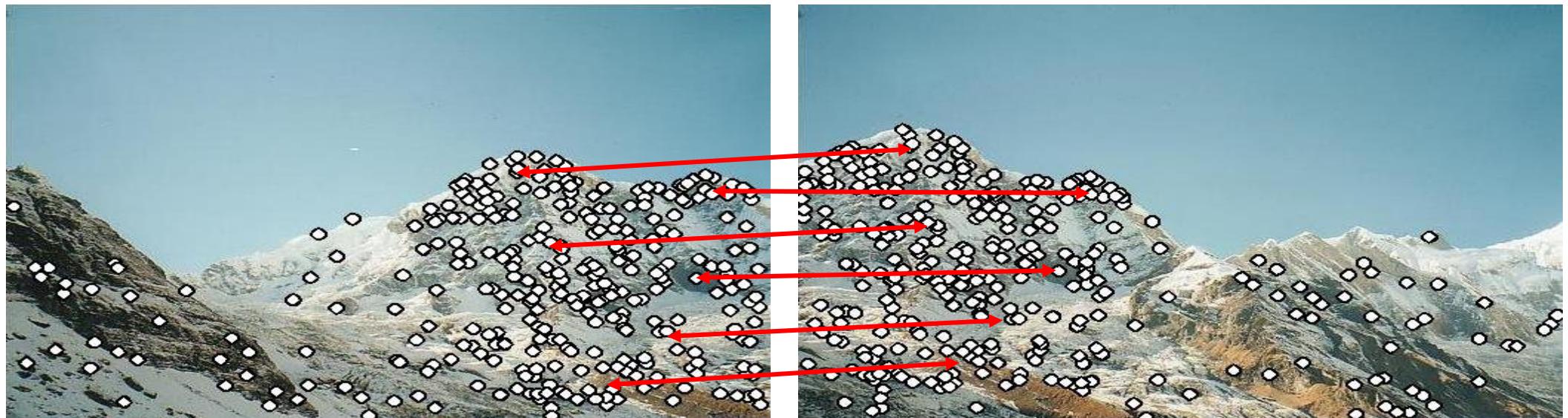
Panoramas

- We have two images – how do we combine them?



Panoramas

- Use SIFT to match descriptors of the two images.
- Between the two images there can be an unknown homographic transformation.
 - How do we align the two images?



Panoramas

- Find the best homographic projection from I_2 to I_1 using RANSAC.
 - Finding this homographic projection is the same as finding the camera calibration matrix that we've seen, only with 3X3 matrix.
 - RANSAC is used to drop wrongly matched points (outliers). Only 3 points needed to be chosen at random to find a RANSAC projection.

