Computer Vision
INM460/IN3060

Lecture 5
Image matching:
interest point detection and feature descriptors

Dr Giacomo Tarroni
Slides credits: Giacomo Tarroni, Wenjia Bai

Recap from the previous lecture

Image segmentation:

- Intensity-based methods
  - Thresholding
  - K-means clustering
- Watershed
- Active contours
  - Snakes
  - Level-sets
- Graph cuts
- Superpixels
- Quantifying results: how to tell if we achieved a good segmentation

Students surveys

Survey	Programmes	Timing of survey	Access
National Student Survey (NSS)	All final year undergraduate students	NSS campaign live from 6th January to 30th April. Core survey campaign live from 20th January to 31st January	http://www.thestudentsurvey.com
Your Voice 1 (questions are similar to NSS)	Undergraduate Year 1 students	20th January to 30th April. Core survey campaign live from 20th January to 31st January	Online access via www.city.ac.uk/feedback
Your Voice 2 (questions are similar to NSS)	Undergraduate Year 2 (Year 3 for some programmes e.g. BSc SLT) students	20th January to 30th April. Core survey campaign live from 20th January to 31st January	Online access via www.city.ac.uk/feedback https://studenthub.city.ac.uk/student-administration/student-survey/take-your-survey
Postgraduate Taught Experience Survey (PTES)	Postgraduate taught students	3rd February to 6th March 2020	Accessible via www.city.ac.uk/feedback https://studenthub.city.ac.uk/student-administration/student-survey/take-your-survey
Postgraduate Research Experience Survey (PRES)	Research students	Not running - runs every two years.	

Overview of today's lecture

Image matching:

- Interest point detection
  - Harris detector
  - Scale-adapted Harris detector
  - Harris-Laplace detector
  - Scale-invariant feature transform (SIFT)
- Feature descriptors
  - Basic features
  - SIFT features
  - Speeded-Up Robust Features (SURF)
- Matching algorithm

Image matching

- Let's assume to have two images showing the same object but in different conditions (e.g. camera angle & position, lighting, etc.)
- How do we perform image matching/alignment/registration, i.e. how do we identify the (rigid? affine? non-rigid?) transformation which aligns them?
  - By pixels: using all the available information
  - By edges: using some information
  - By interest points (also keypoints): using very little information
- All three approaches work and have their applications

Matching

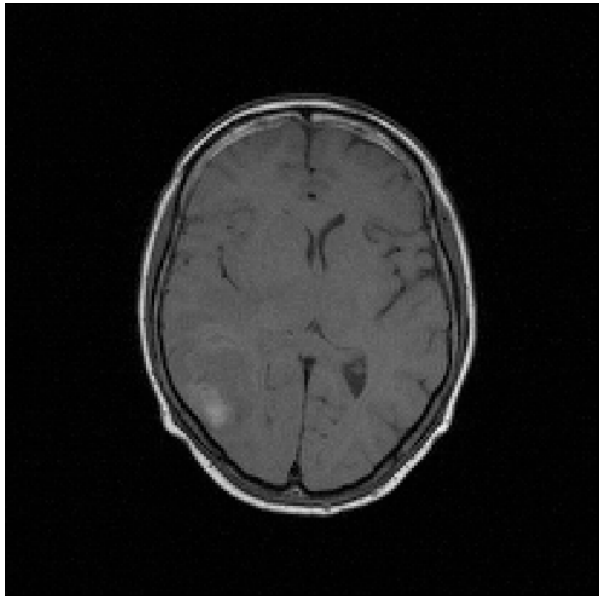
M. Brown. Automatic panoramic image stitching using invariant features, IJCV 2007

Matching by pixels

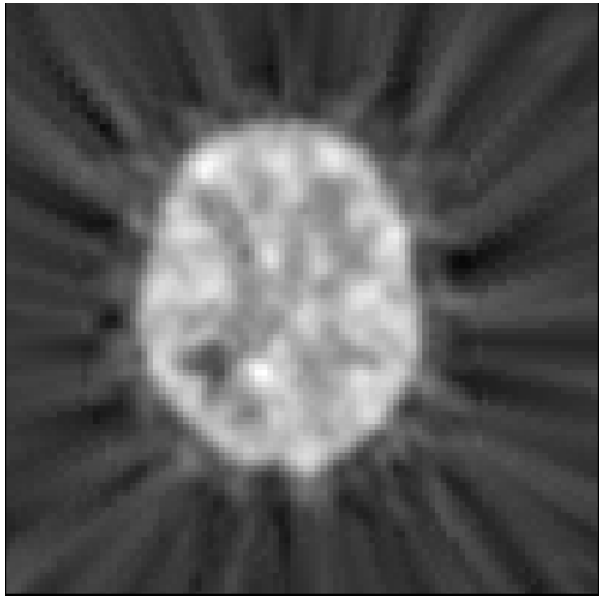
- Many medical image registration algorithms work by finding a transformation T^* which optimises a **similarity metric** based on image pixels:

$$T^* = \underset{T}{\operatorname{argmax}}\, Similarity(I_1, T(I_2))$$

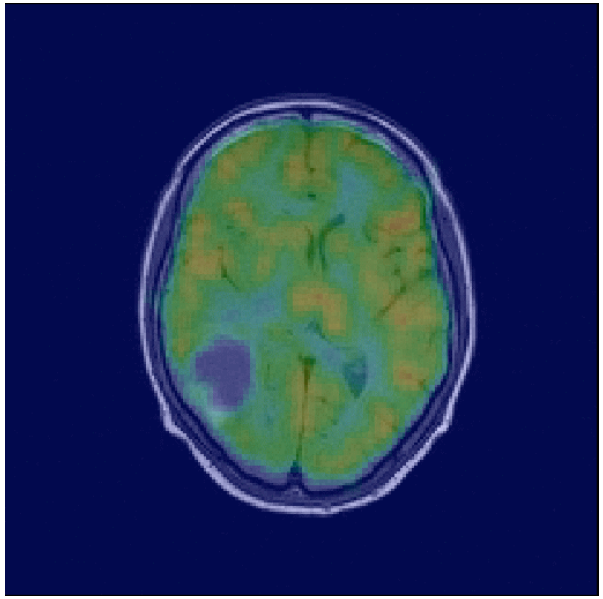
- Similarity metrics: sum of squared distances (SSD), mutual information (MI), etc.
- This enables image fusion (i.e. the combination of images from different modalities) for better clinical assessment



MR image



PET image



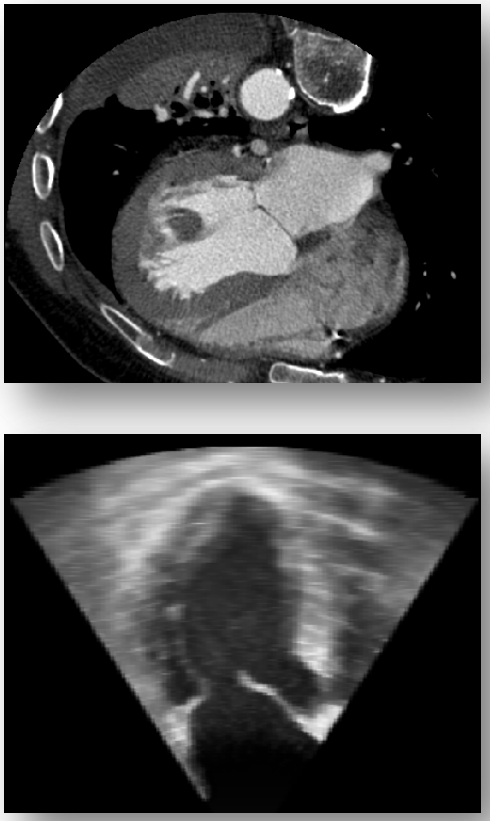
PET/MR image fusion

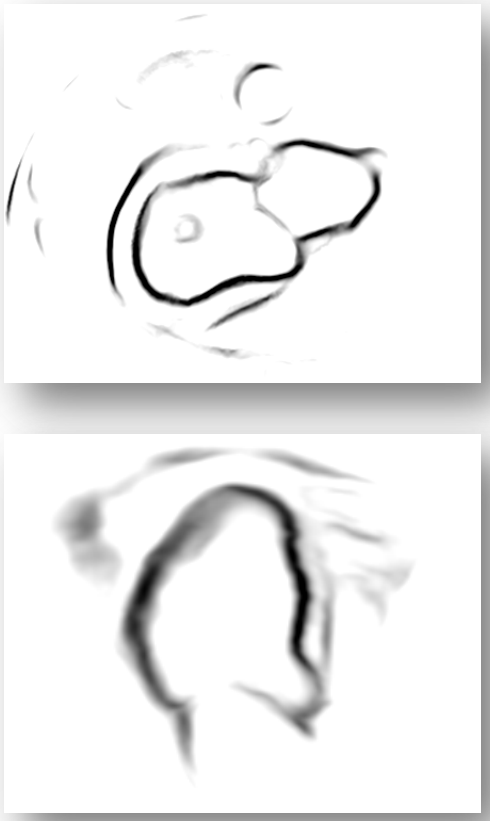
Matching by edges

Edge maps can first be extracted and then used to perform the alignment

3D CT

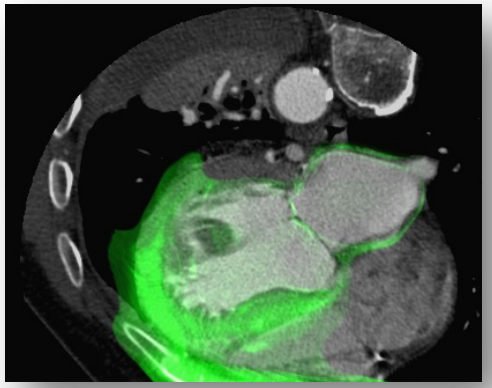
3D US





Input images

Edge maps



CT/US image fusion

O. Oktay et al. Probabilistic edge map for 3D ultrasound image registration, FIMH 2015

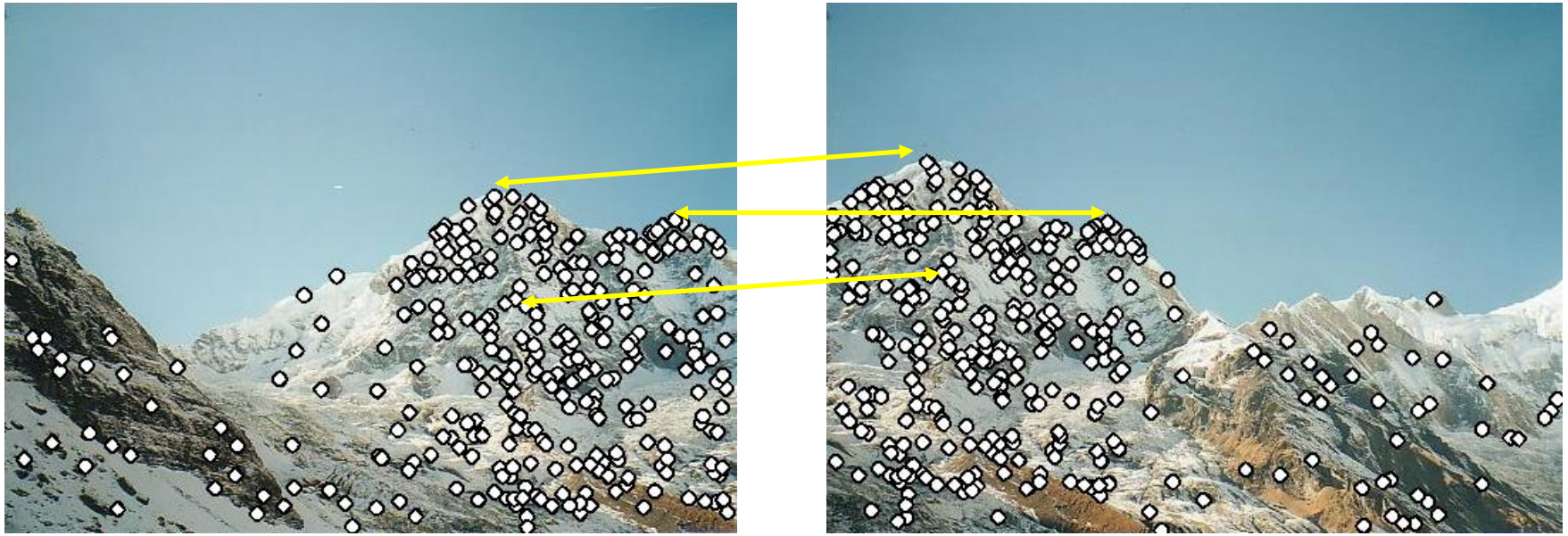
Matching by interest points

- Some “interest points” can be extracted from both images and used to perform the alignment
- Interest points matching has the advantage of being efficient, since we only need to deal with some points and not all the image pixels
- Interest points can be thought to be identified after object detection:
  - First detect the object in the two images
  - Then identify some “homologous” points to perform the alignment

Matching by interest points

Interest points can also be extracted separately from both images **without performing object detection**:

- Identify points that are easy to recognize
- Find potential correspondences between the two images
- Estimate the transformation that best aligns them



M. Brown. Automatic panoramic image stitching using invariant features, IJCV 2007

Matching by interest points

Interest points can also be extracted separately from both images **without performing object detection**:

- Identify points that are easy to recognize
- Find potential correspondences between the two images
- Estimate the transformation that best aligns them

Image stitching / panorama

M. Brown. Automatic panoramic image stitching using invariant features, IJCV 2007

What defines an interest point?

An **interest point** is a point in the image which

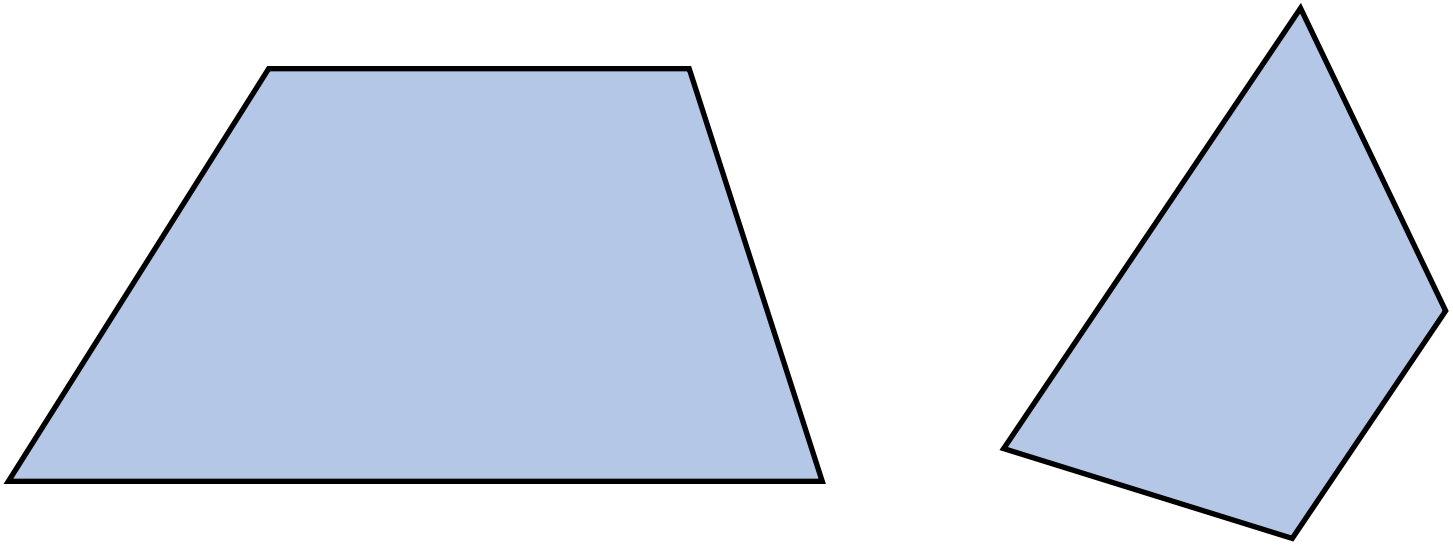
- has a well-defined position in the image
- is positioned where the **local image structure is rich** in terms of information contents (e.g. corner point, significant 2D texture), such that we will be able to distinguish it from other points easily
- is **stable under local and global perturbations** in the image domain (e.g. illumination variations), such that we will be able to identify it reliably with a high degree of repeatability

There are several approaches to interest point detection, including:

- Harris detector
- SIFT detector
- SURF detector

What defines an interest point?

- Pick a point in the left image and find it again in the right image
- What type of point would you select?

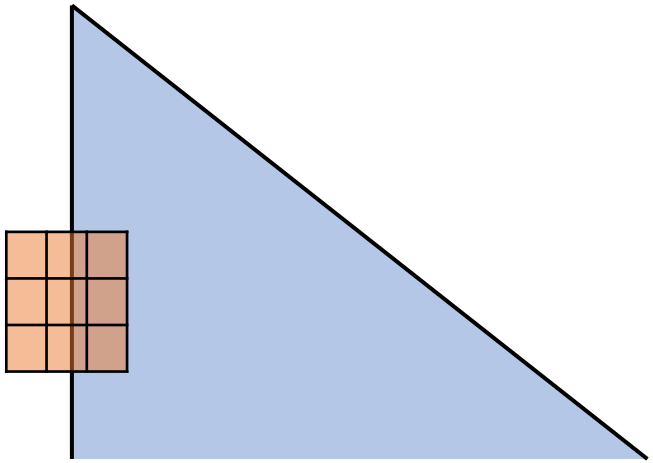


- It's likely that you picked a **corner**
- Corners are good candidates for interest points

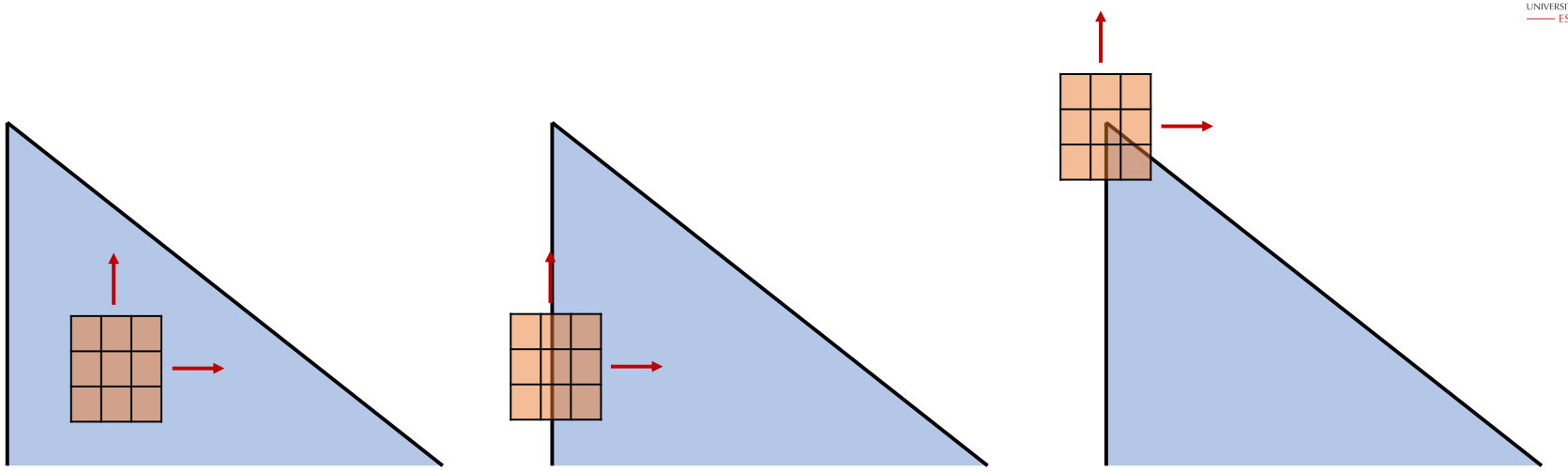
Credit: Ioannis Gkioulekas

Corner detection

- A corner is an intersection of two edges
- Edge detection was performed looking at the magnitude of the image gradient **at each pixel**. Consequently, the gradient magnitude is not enough to differentiate between an edge point and a corner point
- However, if we look at **a small window**, we can tell the difference between edge and corner by shifting the window and measuring the change of intensities for each pixel of the window
  - Edge: change of intensity along just one direction
  - Corner: change of intensity along both directions



Corner detection



flat region
no change in any direction

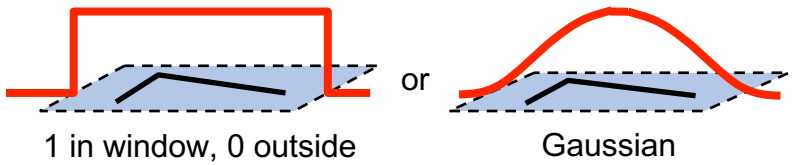
edge
change along one direction

corner
change along both directions

If the shift is $[u, v]$ and I is the image intensity, the change of intensities can be defined using the sum of squared differences:

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

where $w(x,y)$ is a window function defining the extent of the comparison:



$w(x,y)$

C. Harris and M. Stephens. A combined corner and edge detector, AVC, 1988

Change of intensities

For a small shift $[u, v]$, we can approximate the change of intensities using the first derivatives of image intensity:

- Taylor expansion, first order approximation:

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y)$$

- Therefore, substituting in the previous equation:

First derivatives for image intensity

$$E(u,v) = \sum w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

$$\approx \sum w(x,y)\left[uI_x(x,y) + vI_y(x,y)\right]^2$$

$$= \sum w(x,y)(u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2)$$

Rewrite as matrix multiplication

$$= \sum w(x,y)[u,v]\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$= [u,v]\sum w(x,y)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$= [u,v] \cdot M \cdot \begin{bmatrix} u \\ v \end{bmatrix}$$

Change of intensities

For a small shift $[u, v]$, we have the following bilinear approximation,

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

where M is a 2x2 matrix, computed from local image derivatives,

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

Compute different "sub-matrices" for each (x, y) in window and sum them

Product between image derivatives

- M is a matrix, and it contains information about both gradient components (both on their own and multiplied between each other) for the pixels in the window
- We will get a large $E(u, v)$ where image derivatives are large
- All the information regarding the directions of intensity changes is in M

Interpretation of M

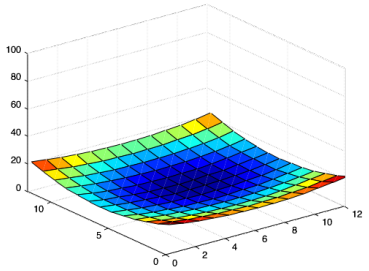
$$E(u,v) \approx [u,v] \cdot M \cdot \begin{bmatrix} u \\ v \end{bmatrix}$$

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

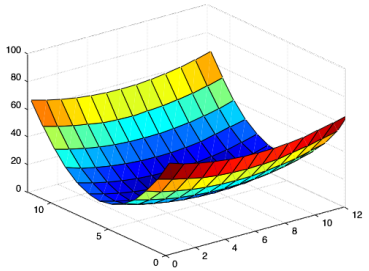
All the information regarding the directions of intensity changes is in M:

- If $M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, we are in a **flat** region
- If $M = \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}$, there will be a large change if we shift along u, but little change if we shift along v. We are on an **edge**
- If $M = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$, there will be a large change whichever way we shift the window. We are on a **corner**

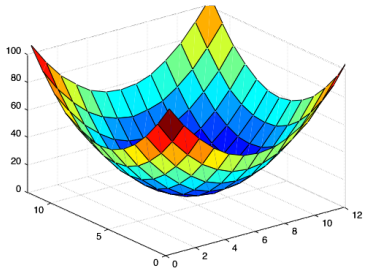
$E(u,v)$



flat



edge



corner

Interpretation of M

In general, M is not a diagonal matrix and is thus more complicated to interpret

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

- To make M simpler, it is possible to perform **eigen decomposition**
- Since M is a real symmetric matrix, we can decompose M by

$$M = P \Lambda P^T$$

$$\text{with } \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

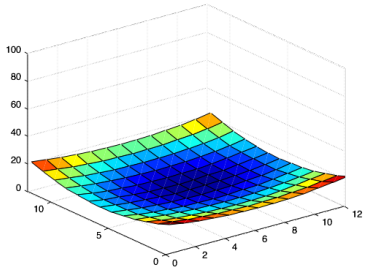
- P is a matrix whose columns are eigenvectors (all orthogonal to each other)
- $\boldsymbol{\Lambda}$ **is a diagonal matrix** containing the eigenvalues of M (with $\lambda_1 \geq \lambda_2$). Since it is diagonal, it can be interpreted using the rationale of the previous slide
- Now, however, we won't be moving along u and v, but instead along
  - the direction of maximum intensity change (i.e. gradient), with intensity change represented by λ_1
  - the one perpendicular to that, with intensity change represented by λ_2

Interpretation of Λ

$$E(u,v) \approx [u,v] \cdot P\Lambda P^T \cdot \begin{bmatrix} u \\ v \end{bmatrix}$$

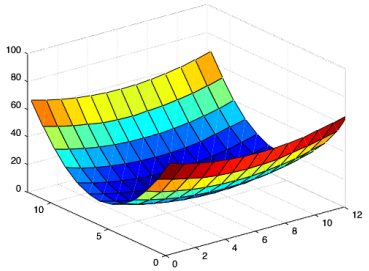
$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$E(u,v)$



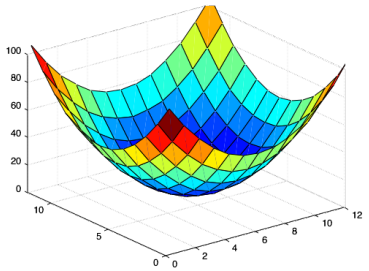
flat

- If $\Lambda = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, we are in a **flat** region



edge

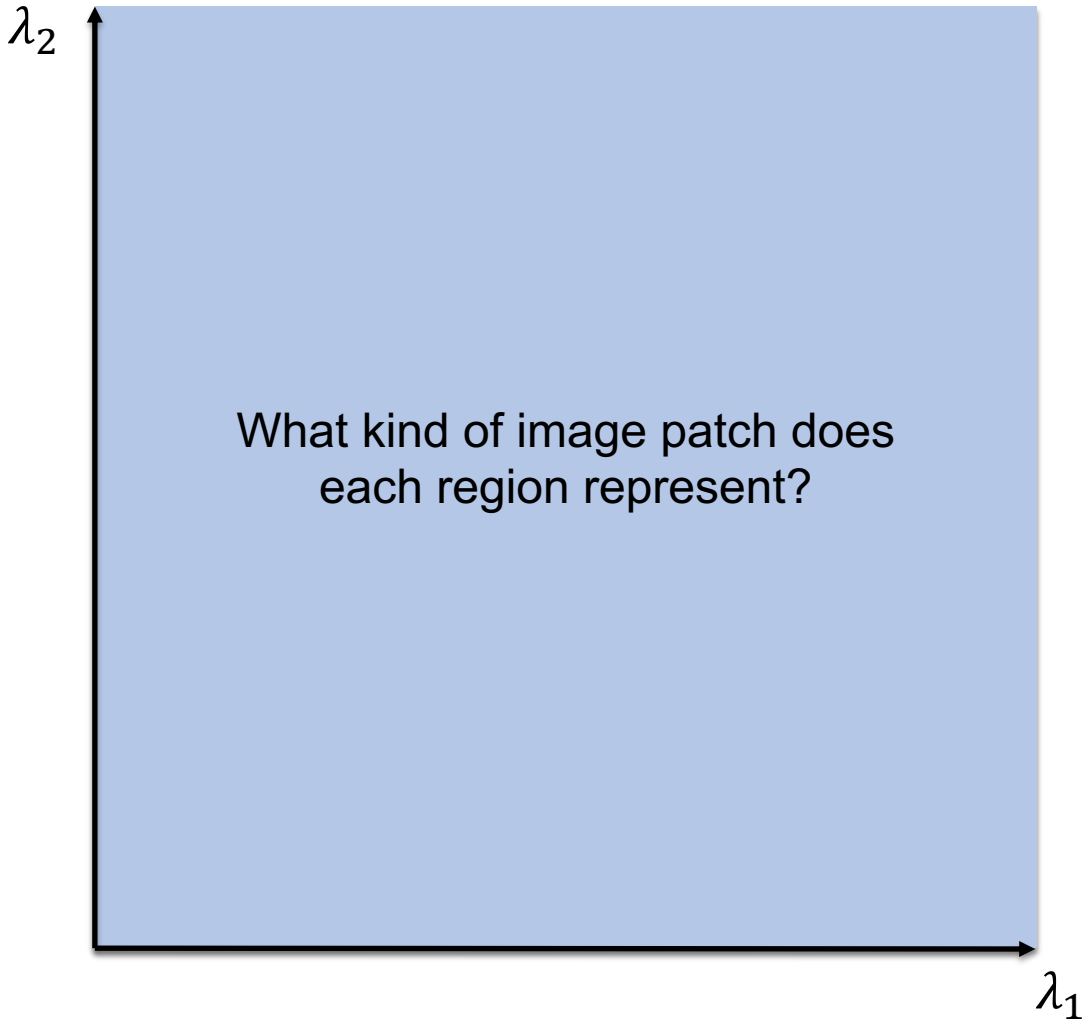
- If $\Lambda = \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}$, there will be a large change if we shift along the gradient direction, but little change if we shift along the perpendicular one. We are on an **edge**



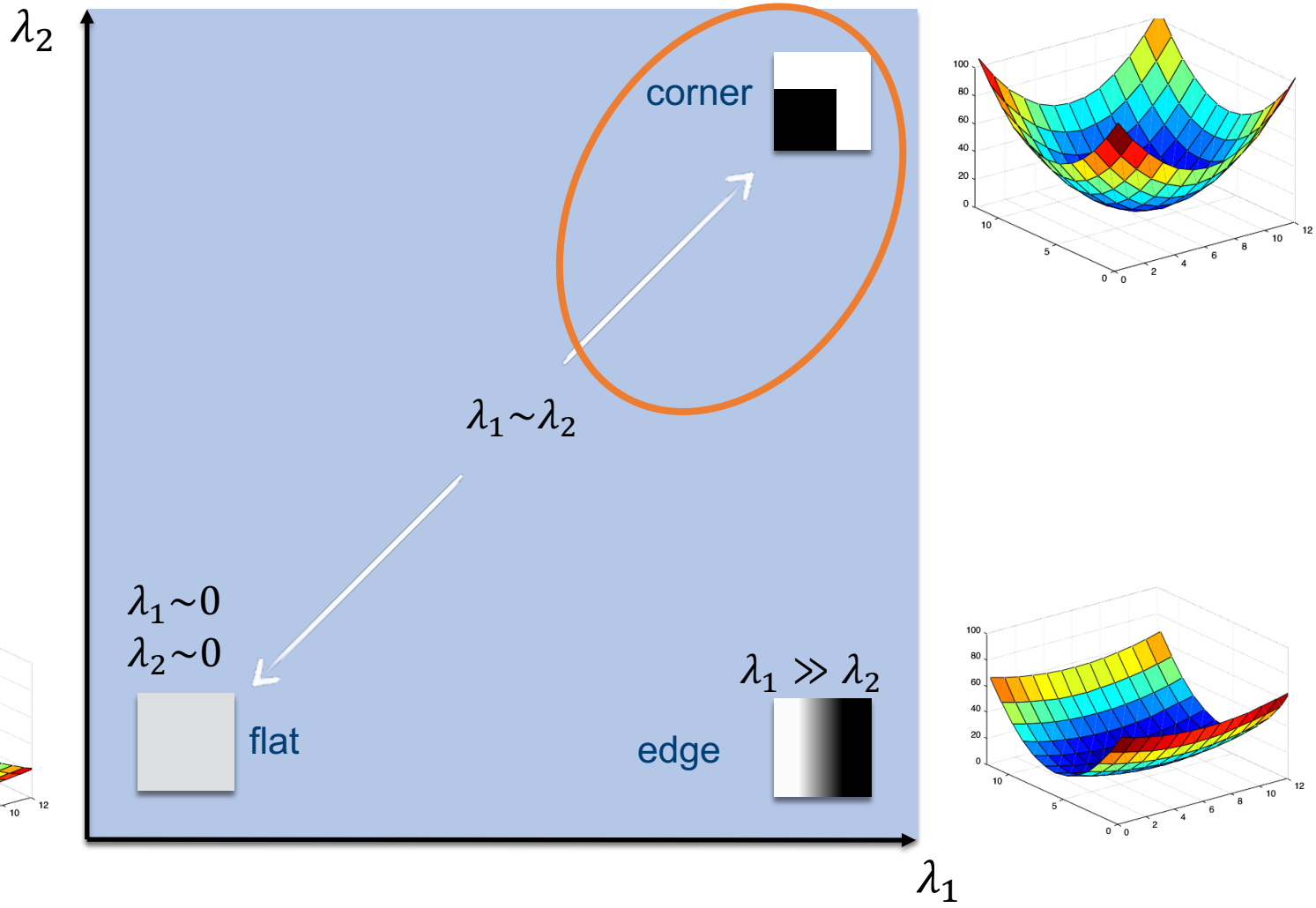
corner

- If $\Lambda = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$, there will be a large change whichever way we shift the window. We are on a **corner**

Interpretation of Λ



Interpretation of Λ



Cornerness definition

We need to define a **measure of cornerness**:

1. Harris and Stephens (1988)

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

where k is an empirically determined small number ($k \in [0.04, 0.06]$)

Note: if we only want the values of $\lambda_1\lambda_2$ and $\lambda_1 + \lambda_2$, we do not need to perform eigen decomposition: it can be demonstrated that

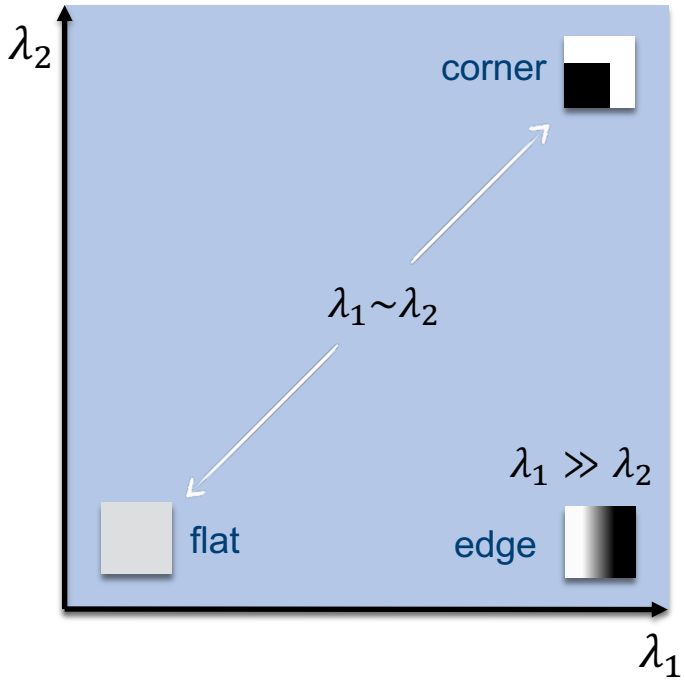
$$R = \det(M) - k(\text{trace}(M))^2$$

2. Kanade and Tomasi (1994)

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

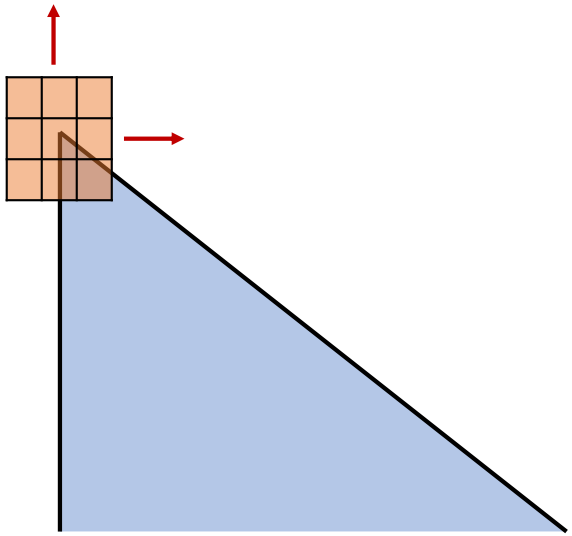
3. Noble (1998)

$$R = \frac{\lambda_1\lambda_2}{\lambda_1 + \lambda_2 + \epsilon}$$

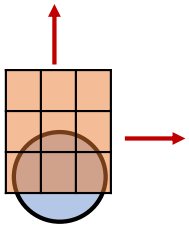


Harris corner detector

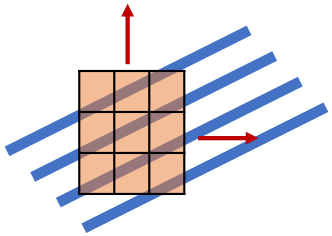
- If we use the cornerness definition of Harris and Stephens, we obtain the "Harris corner detector"
- In reality, this detector finds strong responses not only at corners, but also at **blobs and textures**
- As a consequence, it is often referred to as Harris detector



corner
change along both directions



blob
change along one direction



texture
change along both directions

Harris corner detector

The complete list of steps for the Harris corner detector are the following:

1. Compute x and y derivatives of the image I

$$I_x = G_x * I$$
$$I_y = G_y * I$$

E.g.: G_x, G_y can be Sobel filters

2. Compute **at each pixel** the matrix M,

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

3. Compute at each pixel the **detector response** $R = \det(M) - k(\mathrm{trace}(M))^2$
4. Clamp the response R (everything below a user-defined threshold is set to 0)
5. Find local maxima through non-maximum suppression (NMS)

Built-in Matlab function: `detectHarrisFeatures`

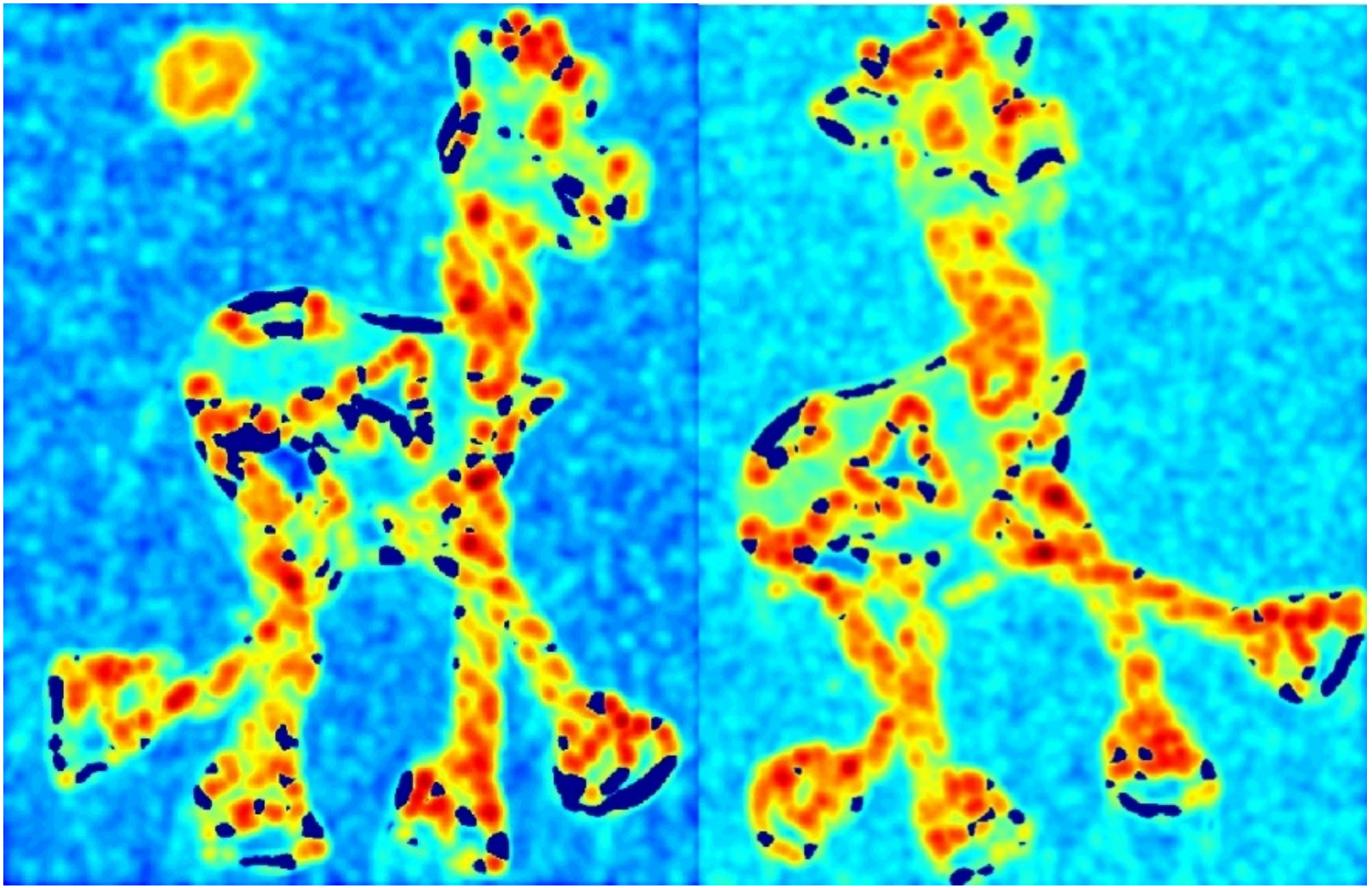
C. Harris and M. Stephens. A combined corner and edge detector, AVC, 1988

Harris corner detector

Input images

Credits: Saad J Bedros

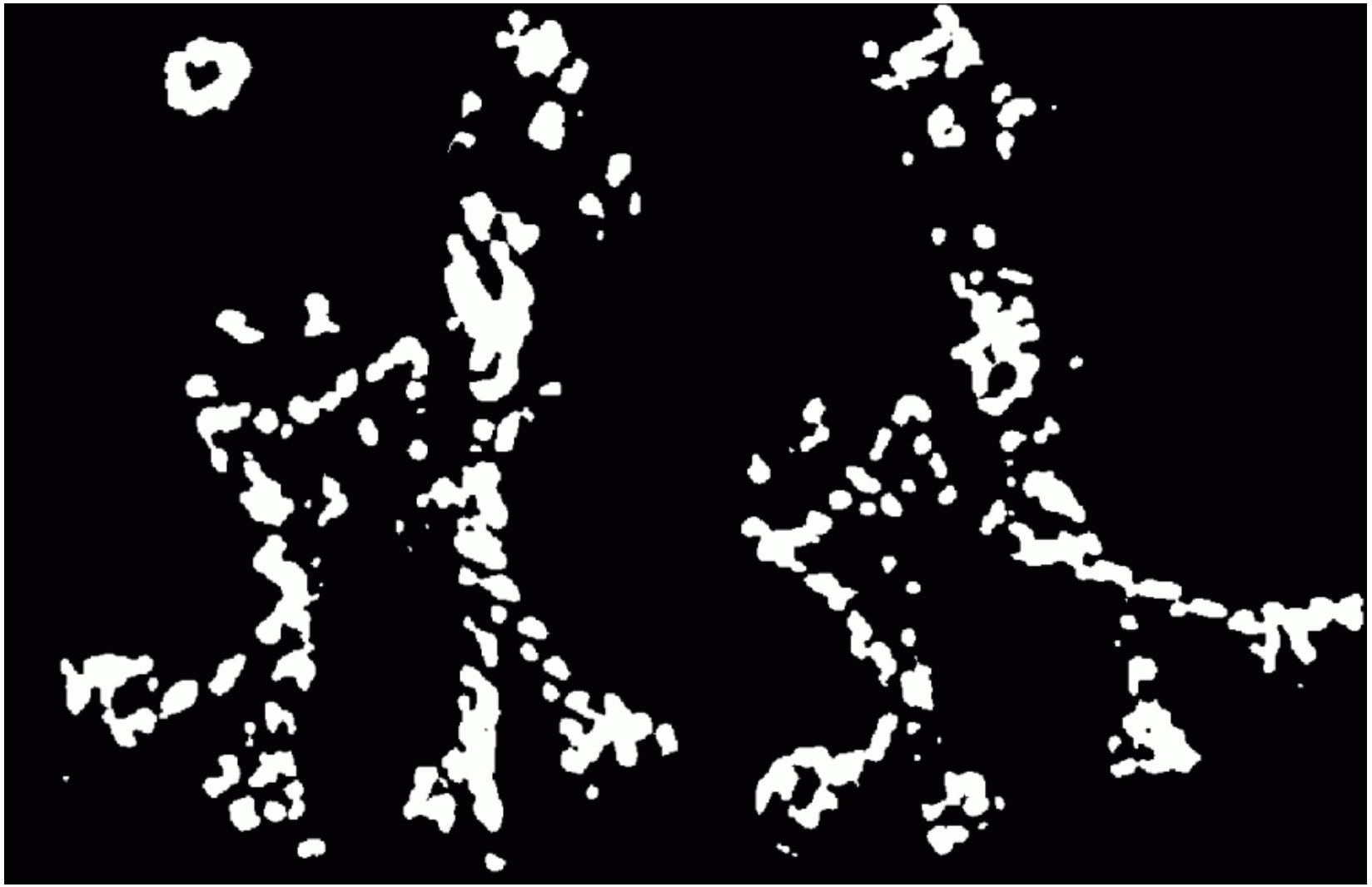
Harris corner detector



Harris corner detector response

Credits: Saad J Bedros

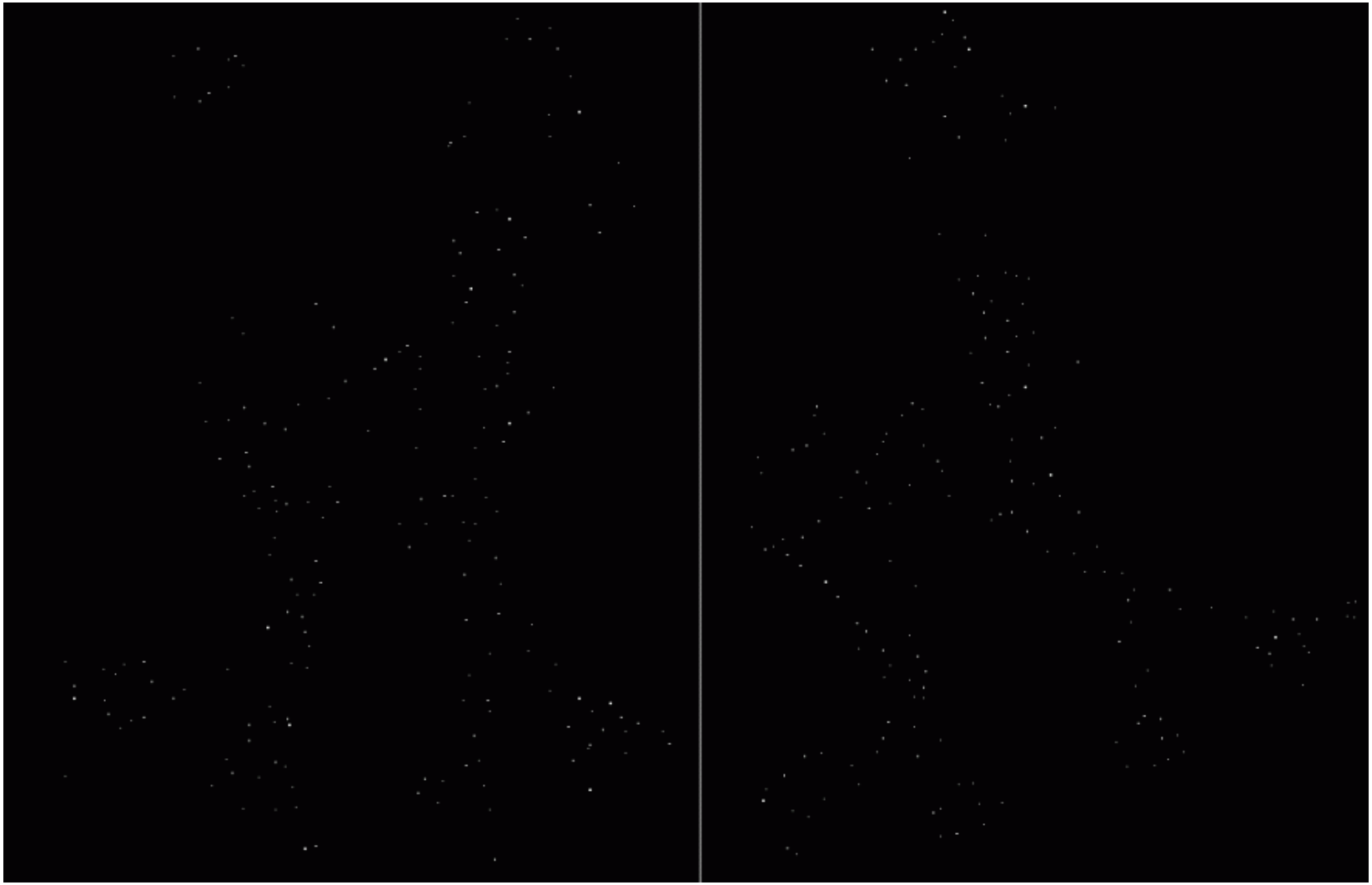
Harris corner detector



Thresholding

Credits: Saad J Bedros

Harris corner detector



Non-maximum suppression

Credits: Saad J Bedros

Harris corner detector

Detection results

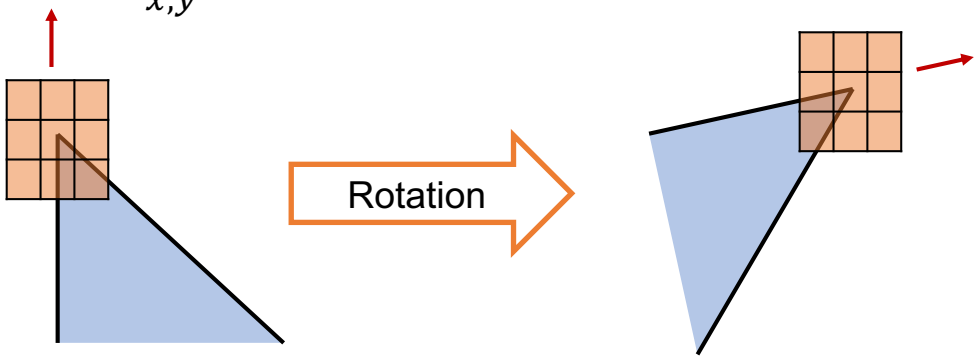
Credits: Saad J Bedros

Rotation and scaling invariances

The Harris corner detector is **invariant to rotation**:

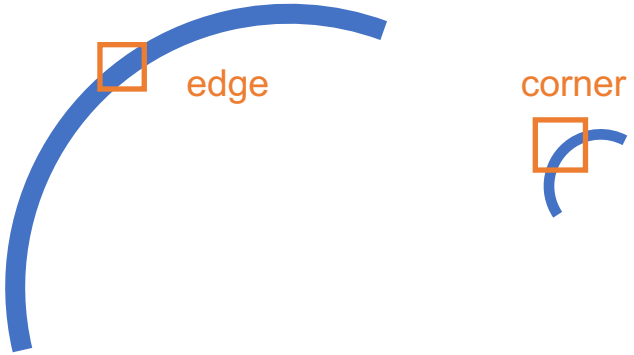
- If the corner has rotated, the same change of intensities will be measured shifting the window, but for a different direction
- The eigenvalues λ_1 and λ_2 of M do not change!

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = P \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} P^T$$



The Harris corner detector is **not invariant to scaling**:

- However, we can apply the detector at multiple scales



Scale-space theory

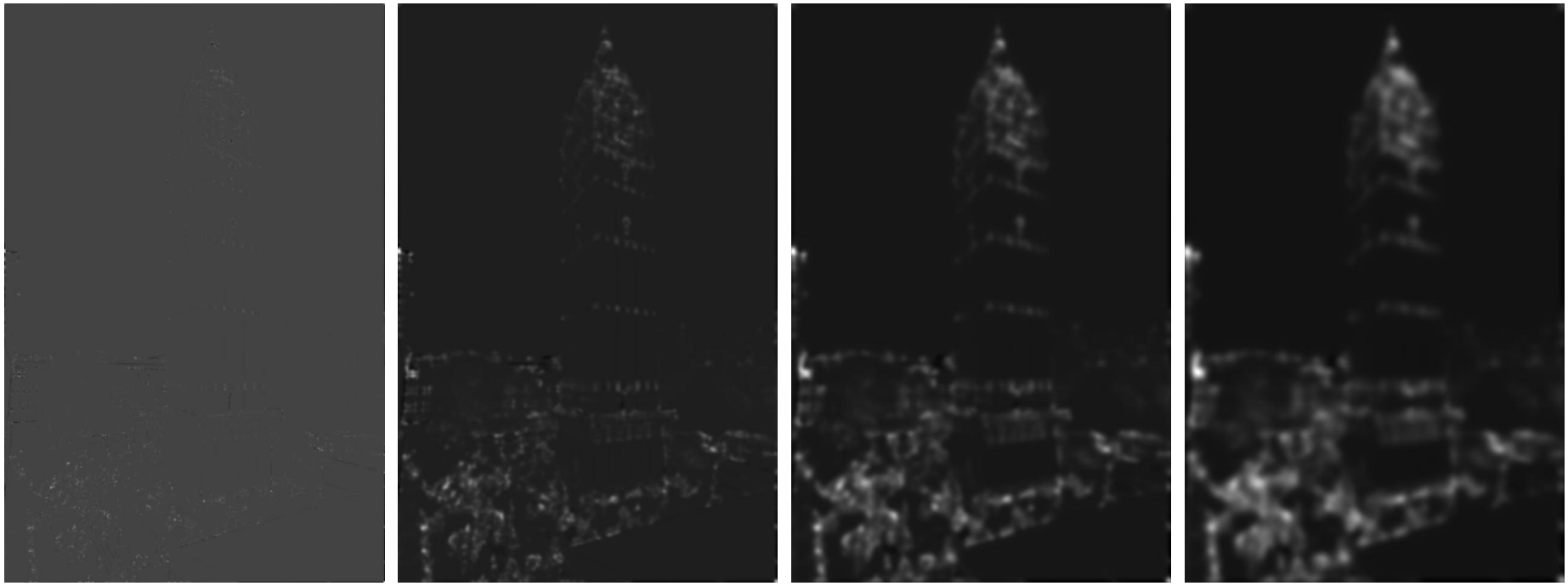
- Real-world objects are composed of different structures at different scales: the closer we are to an object, the more the details we perceive
- A **scale-space representation** is created by
  - generating smoothed versions of a given image using a Gaussian kernel with progressively bigger σ (the “scale parameter”)
  - creating smaller versions through progressive image resizing

$\sigma = 1$ $\sigma = 3$ $\sigma = 5$ $\sigma = 7$

Response at different scales

- With a scale-space representation, we can apply the detector at multiple scales. However, if we keep all the points we obtain at different scales, we will have **many duplicates for the same structures at different scales**
- What is the most suitable scale for a selected pixel? Intuitively, the one at which we get the **highest response** from the Harris detector

Harris detector response for images after smoothing with different Gaussian kernels



$\sigma = 1$ $\sigma = 3$ $\sigma = 5$ $\sigma = 7$

Response at different scales

There is no need to explicitly create the scale-space representation: we can simply substitute the image derivatives I_x and I_y with Derivatives of Gaussian:

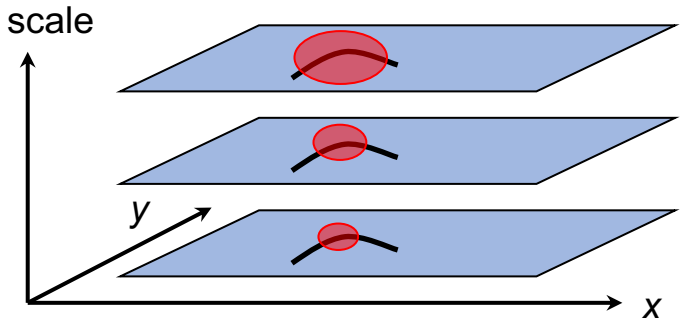
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2(\sigma) & I_x(\sigma)I_y(\sigma) \\ I_x(\sigma)I_y(\sigma) & I_y^2(\sigma) \end{bmatrix}$$

with

$$I_x(\sigma) = G_x(\sigma) * I$$
$$I_y(\sigma) = G_y(\sigma) * I$$

where $G_x(\sigma)$ can be the concatenation between a Gaussian smoothing filter (with standard deviation σ) and a Sobel filter. Same for y direction

- The response R is determined by the eigenvalues of M, which are in turn affected by the derivatives $I_x(\sigma)$ and $I_y(\sigma)$
- We can **compute the response** at each pixel and **at different scales**, and then **compare** the results



Harris detector response at each scale

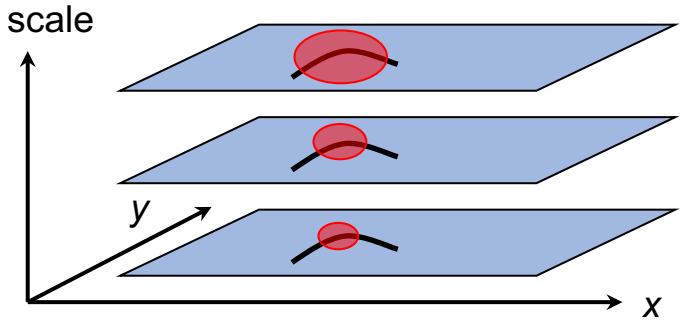
Scale-adapted Harris detector

- Is a direct comparison of the Harris detector response between scales fair?
- It can be shown that the magnitude of the derivatives $I_x(\sigma)$ and $I_y(\sigma)$ is inversely correlated with the scale σ: the larger the scale, the stronger the smoothing, the smaller the magnitude
- To **make the derivative magnitude comparable across scales**, we need to take this into account multiplying by σ^2:

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

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

- This is the **scale-adapted Harris detector**:
  1. Apply it at different scales
  2. Keep the points that give the highest response over scale



Harris detector response at each scale

Harris-Laplace detector

- Studies show that the (normalized) Laplacian of Gaussian (LoG) is a very robust blob detector, and can be used for finding the appropriate scale [1]

- The LoG response at scale σ is

Gaussian filter

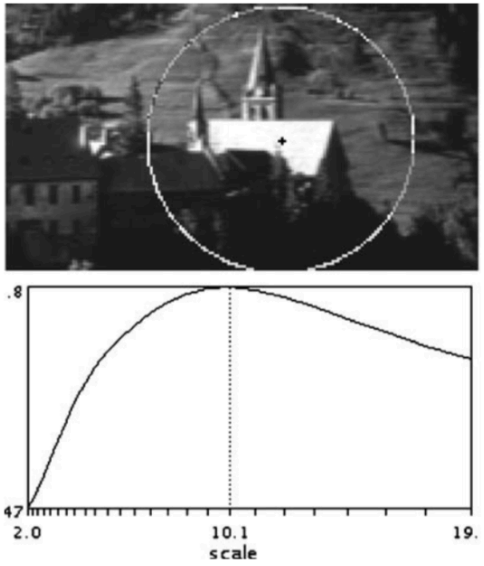
Gaussian filter followed by second derivative

$$LoG(x, y, \sigma) = \Delta G(\sigma) * I = I_{xx}(x, y, \sigma) + I_{yy}(x, y, \sigma)$$

- The normalised LoG response at scale σ is

$$LoG_{norm}(x, y, \sigma) = \sigma^2(\Delta G(\sigma) * I) = \sigma^2(I_{xx}(x, y, \sigma) + I_{yy}(x, y, \sigma))$$

- This is the **Harris-Laplace detector**:
  1. Apply the scale-adapted Harris at different scales
  2. Keep only the points that are a maximum in the $|LoG_{norm}|$ response over scale



[1] K. Mikolajczyk et al. Scale & affine invariant interest point detectors, IJCV 2004

Scale-invariant feature transform (SIFT)

- SIFT is an algorithm that performs interest point detection, description and matching: let's focus on the first aspect here

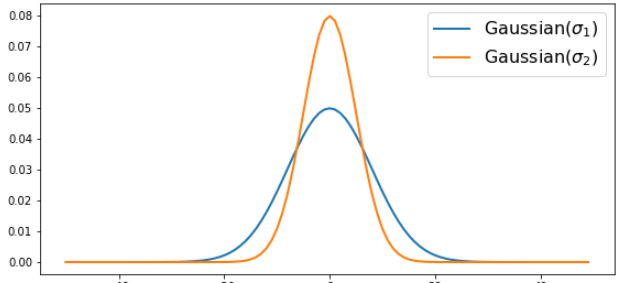
- It uses the **Difference of Gaussians (DoG) filter**, which is an approximation to the normalised Laplacian of Gaussian (LoG):

Gaussian filters for different scales

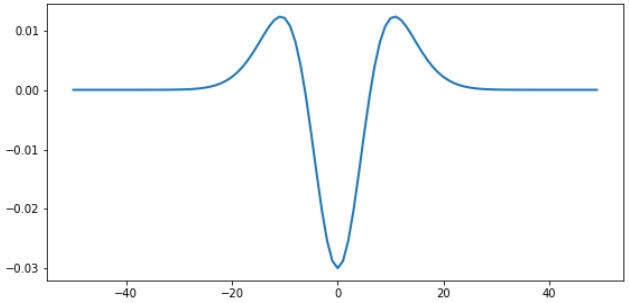
$$DoG(x, y, \sigma) = G(k\sigma) * I - G(\sigma) * I$$
$$= \big(G(k\sigma) - G(\sigma)\big) * I$$

- For small k, we have

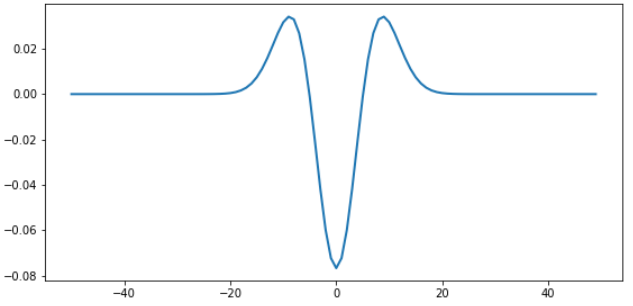
$$DoG(x, y, \sigma) \approx (k-1) LoG_{norm}(x, y, \sigma)$$



Two Gaussian kernels



DoG kernel

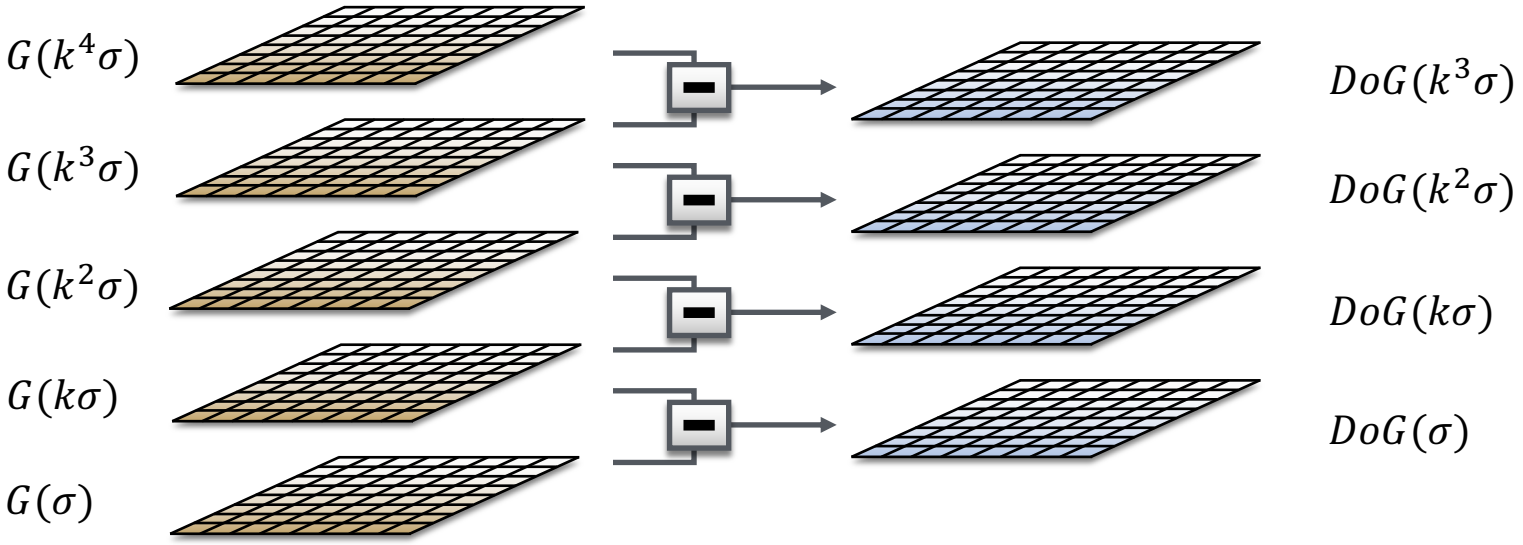


LoG kernel

D. Lowe. Distinctive image features from scale-invariant keypoints, IJCV 2004

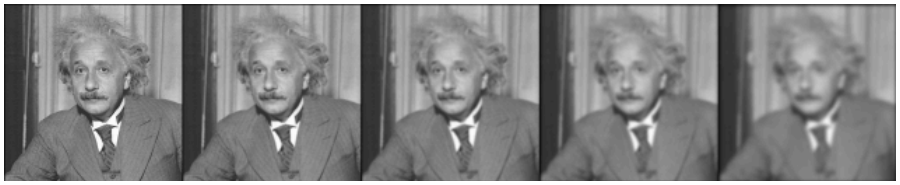
Scale-space representation in SIFT

In SIFT, the canonical scale-space representation is created and used to generate a series of DoG images:



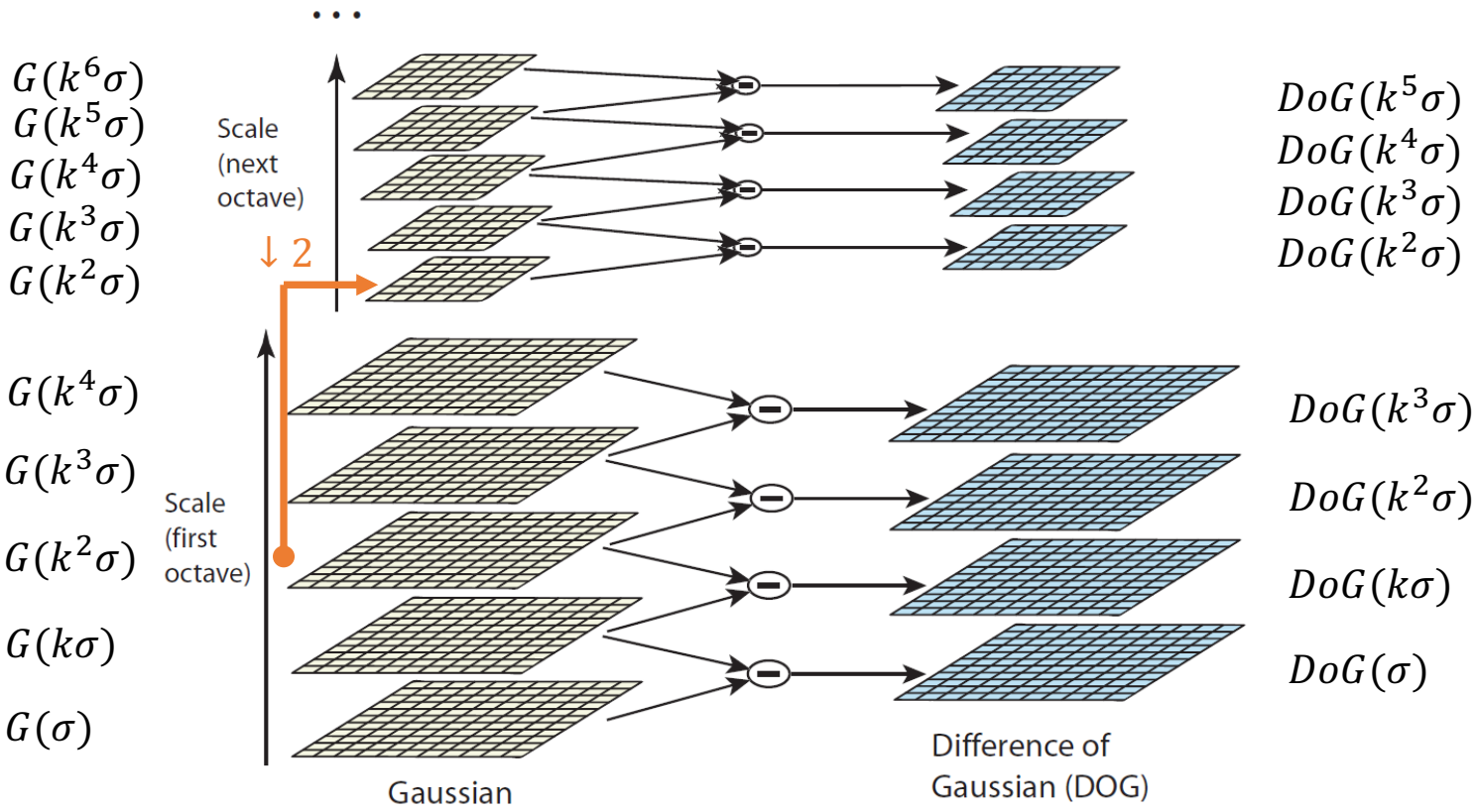
Gaussian

Difference of Gaussians (DoG)



Scale-space representation in SIFT

When a new octave is reached (i.e. σ has doubled), the image is also resized by a factor of 2

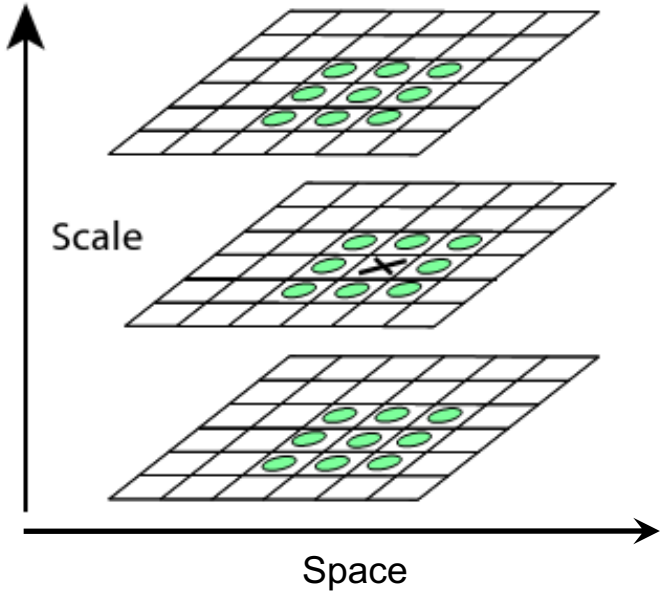


Both k and σ have to be defined beforehand

D. Lowe. Distinctive image features from scale-invariant keypoints, IJCV 2004

SIFT: detection of scale-space extrema

- Interest points are identified as local extrema (maxima or minima) of the DoG images across scales
- This is done by comparing each pixel in the DoG images to its local neighbours at the same scale and in each of the two neighbouring scales
- If the DoG pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate interest point



D. Lowe. Distinctive image features from scale-invariant keypoints, IJCV 2004

SIFT: accurate keypoint localization

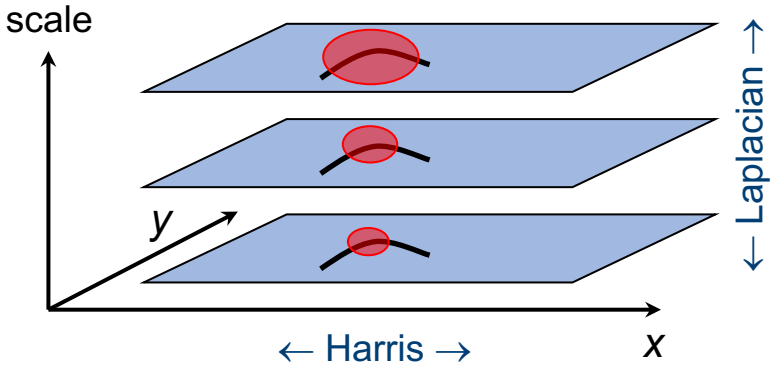
Candidates are then further selected and refined in position:

- Candidates with low contrast (i.e. they are barely extrema) are considered noise and discarded
- The position of the remaining candidates is then refined through subpixel interpolation of neighbouring pixel values. The interpolation is done using the quadratic Taylor expansion of the DoG images
- The candidates that lie on edges are then discarded: these aren't good candidates because they are invariant to translations parallel to the edge direction. Edge extrema are identified by comparing the principal curvatures of the DoG images (a method inspired by Harris detector, but uses second order derivatives)

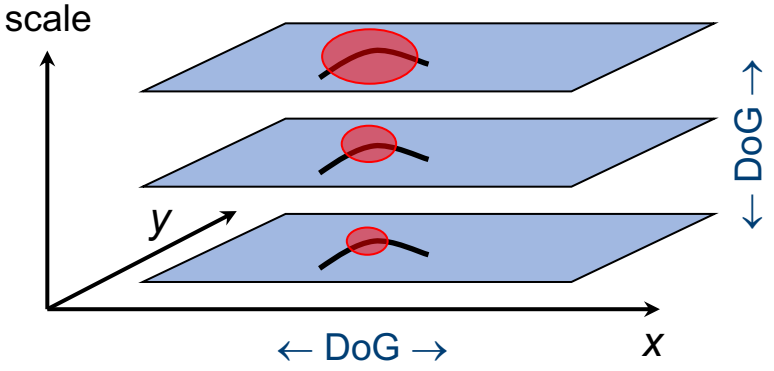
D. Lowe. Distinctive image features from scale-invariant keypoints, IJCV 2004

Scale invariant detectors

- Harris-Laplace detector [1]:
  - Find the local maxima in space using Harris detector response at different scales
  - Find the optimal scale σ using the Laplacian response for each candidate



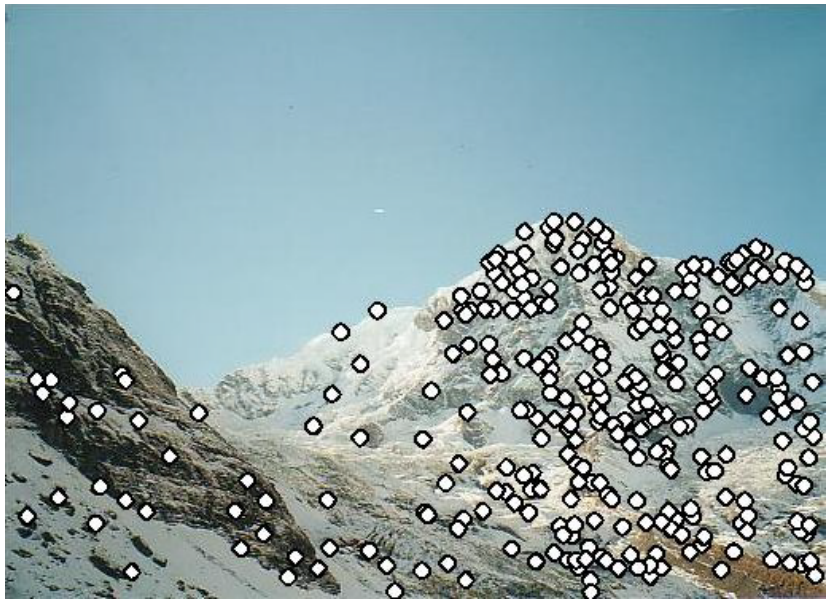
- SIFT detector [2]:
  - Find the local extrema (maxima/minima) in space using DoG response
  - Find the optimal scale σ using DoG response

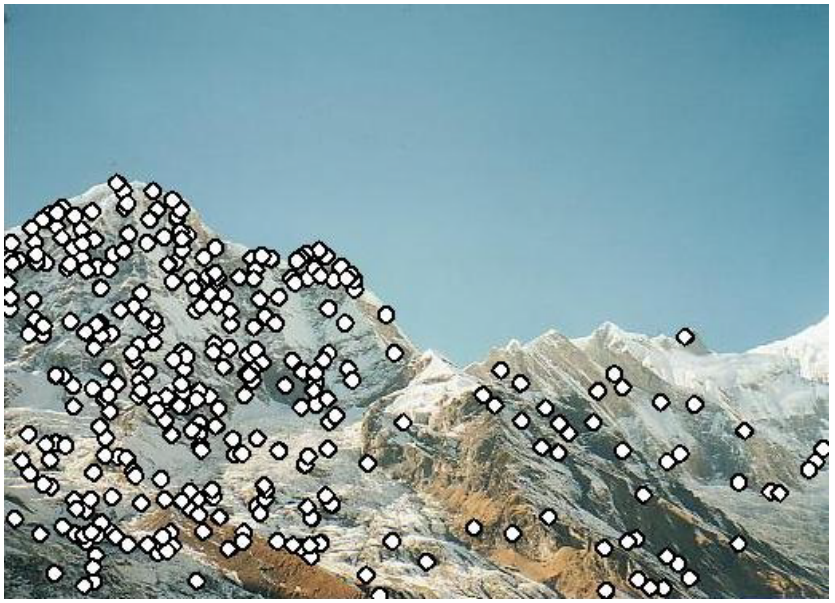


[1] K. Mikolajczyk et al. Scale & affine invariant interest point detectors, IJCV 2004
[2] D. Lowe. Distinctive image features from scale-invariant keypoints, IJCV 2004

Matching by interest points

- Now we know how to detect interest points on different images
- How do we know which point corresponds to which?
- We need to **describe each point** so that similarity can be evaluated





M. Brown. Automatic panoramic image stitching using invariant features, IJCV 2007

Feature descriptors

- This is done by associating each point with a **feature descriptor**, i.e. a list of numbers which should
  - describe the single point (local information)
  - be independent of image perturbations (camera angle, lighting, etc.)
- Some ideas:
  - Intensity/colour of a single pixel
  - Intensity/colour of a patch
  - Gradient
  - Histogram
  - …

Pixel intensity

Simply use the intensity of a single pixel.

- Limitations:
  - Sensitive to absolute intensity value
  - Not very discriminative:
    - The grayscale intensity ranges from 0 to 255
    - There are thousands of pixels with exactly the same intensity value
    - A single pixel does not represent any local pattern (edge, blob, peak of the mountain etc.)

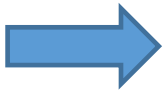
199	192	158	111	110	123	130	130
189	149	108	111	113	120	126	125
130	100	98	108	113	113	114	120
85	100	96	**104**	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	70	87	84
77	80	72	71	60	52	59	64
68	67	63	58	53	51	54	52

Pixel intensity of a patch

Use the intensity values in a local patch.

- Examples:
  - PatchMatch [1]
  - Patch-based segmentation [2]
- Advantages:
  + Represent the local pattern
  + Perform well if the images are of similar intensities and roughly aligned
- Limitations:
  - Sensitive to absolute intensity values
  - Not rotation-invariant

1	2	3
4	5	6
7	8	9



Patch representation

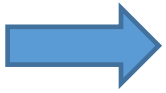
[1, 2, 3, 4, 5, 6, 7, 8, 9]

2	4	6
8	10	12
14	16	18

If intensity changes (*2)

[2, 4, 6, 8, 10, 12, 14, 16, 18]

7	4	1
8	5	2
9	6	3



If being rotated

[7, 4, 1, 8, 5, 2, 9, 6, 3]

[1] C. Barnes et al. PatchMatch: A randomized correspondence algorithm for structural image editing, SIGGRAPH 2009
[2] P. Coupe et al. Patch-based segmentation using expert priors. NeuroImage, 2011

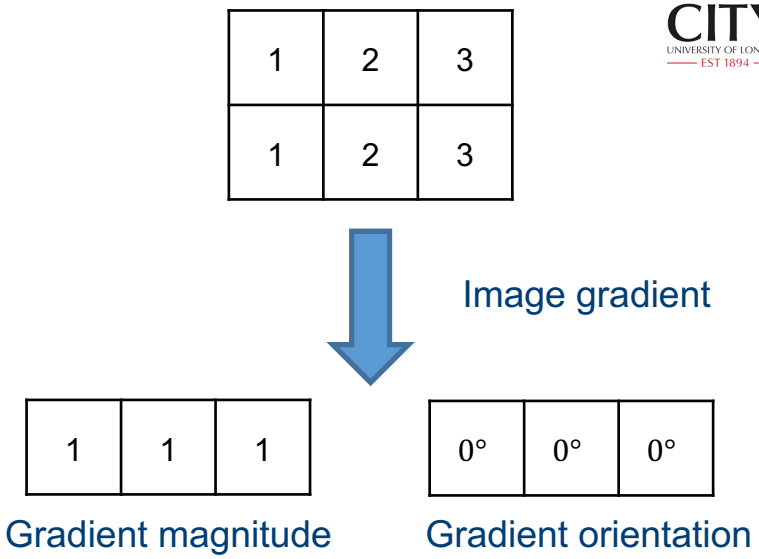
Image gradient

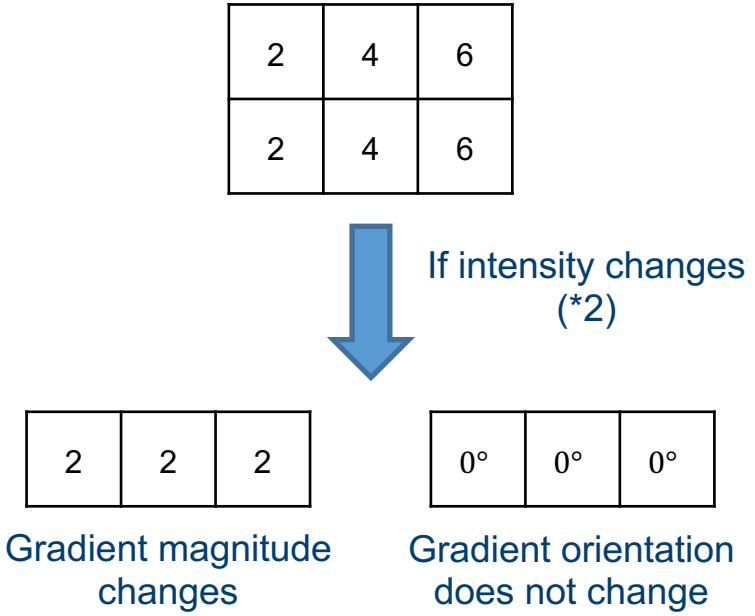
Use the magnitude of the gradients in an image patch.

- Limitations:
  - Sensitive to absolute intensity values

Use the orientation of the gradients in an image patch.

- Advantages:
  - + More robust to intensity changes
- Limitations:
  - Not rotation-invariant





Histogram

Use the intensity histogram in an image patch.

- Limitations:
  - Sensitive to absolute intensity values
- Advantages:
  + Robust to rotation
  + Somewhat robust to scaling

1	2	3
4	5	6
7	8	9

Histogram representation

7	4	1
8	5	2
9	6	3

If being rotated

Histogram representation

1	1	2	2	3	3
1	1	2	2	3	3
4	4	5	5	6	6
4	4	5	5	6	6
7	7	8	8	9	9
7	7	8	8	9	9

If being scaled

Histogram representation

Combining different properties

- Gradient orientation
  - Robust to change of absolute intensity values
- Histogram
  - Robust to rotation and scaling

- Can we combine the advantages of both?
  - Scale-invariant feature transform (SIFT) descriptor

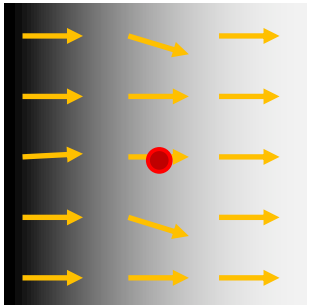
- Note: the SIFT algorithm consists of the several steps:
  1. Detection of scale-space extrema (seen previously)
  2. Accurate keypoint (interest point) localization (seen previously)
  3. Orientation assignment
  4. Local image descriptor
  5. Image matching

D. Lowe. Distinctive image features from scale-invariant keypoints, IJCV 2004

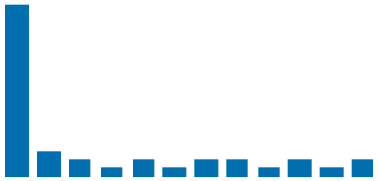
SIFT: orientation assignment

Each previously identified keypoint is assigned an orientation. Main steps:

- A small window is positioned around each keypoint at the relative scale
- The gradient magnitude and orientation of the image pixels (at that scale) are calculated in this window
- An orientation histogram with 36 bins (each covering 10°) is created: each pixel votes for an orientation bin, with a weight equal to the gradient magnitude
- The peak of the histogram determines the **local dominant orientation** θ, which is assigned to the keypoint



The pixel gradient orientations in a patch

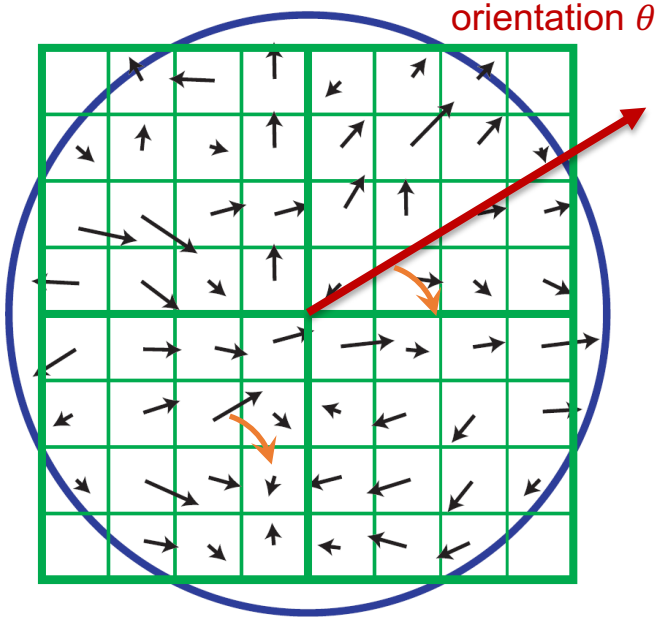


Orientation histogram (36 bins). The pixels in the example patch vote for orientation 0° (first bin)

SIFT: local image descriptor

Now we can calculate a feature descriptor for each keypoint. Main steps:

- A small window is positioned around each keypoint at the relative scale
- The gradient magnitude and orientation of the image pixels (at that scale) are calculated in this window
- The **gradient orientations are rotated** relative to the **keypoint dominant orientation** θ
- The window is divided in subregions
- For each subregion, an orientation histogram with 8 bins (each covering 45°) is created: each pixel votes for an orientation bin, with a weight equal to the gradient magnitude

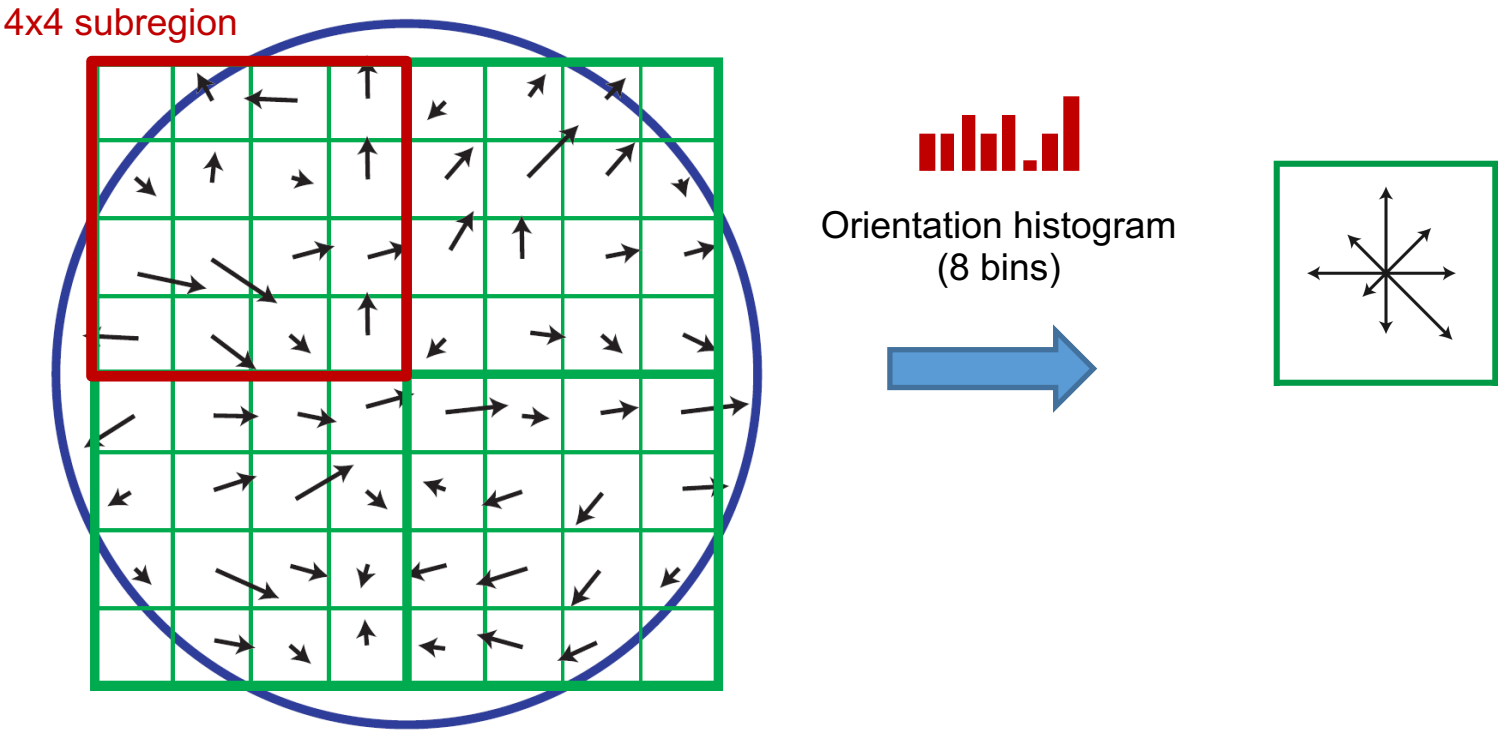


gradient orientation is rotated by θ for each pixel

After the rotation, the pixelwise gradient orientations in these patches will be similar

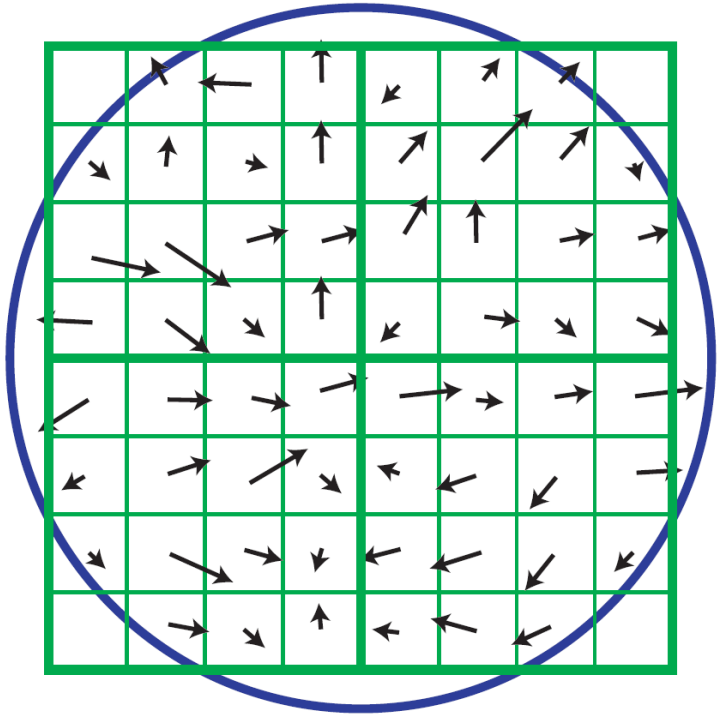
SIFT: local image descriptor

- For each subregion, an orientation histogram with 8 bins (each covering 45°) is created: each pixel votes for an orientation bin, with a weight equal to the gradient magnitude

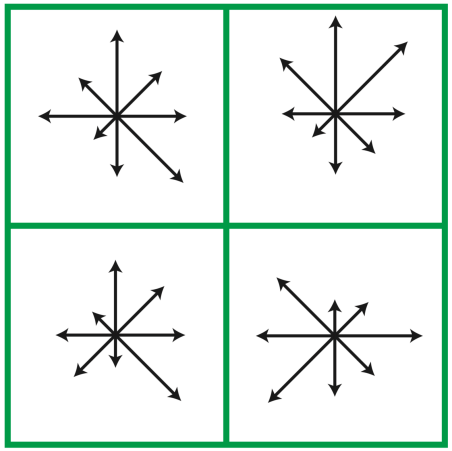


SIFT: local image descriptor

- For each subregion, an orientation histogram with 8 bins (each covering 45°) is created: each pixel votes for an orientation bin, with a weight equal to the gradient magnitude

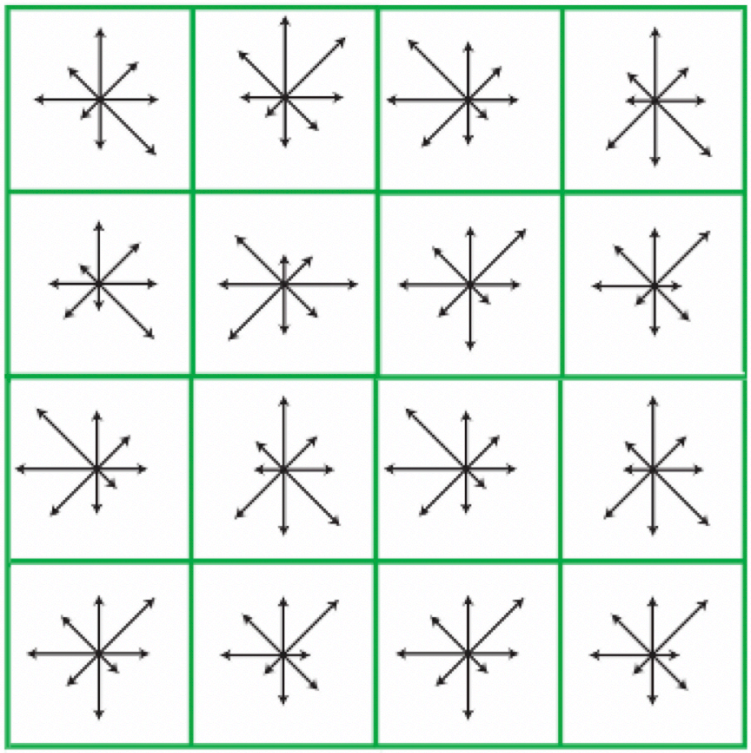


Orientation histogram
for each subregion



SIFT: local image descriptor

- Each subregion has a gradient orientation histogram that describes the local features
- In the original paper, 16 subregions were used
- As a consequence, the number of elemens of the SIFT descriptor was 16 subregions x 8 bins = 128
- We use a feature vector of 128 elements to describe each keypoint



SIFT descriptor

Interactive tutorial on SIFT: http://weitz.de/sift/index.html?size=large

SIFT Matlab Toolbox: http://www.vlfeat.org/overview/sift.html

Speeded-Up Robust Features (SURF)

- SIFT is relatively fast, but still has to:
  - Create the DoG pyramid
  - Compute the gradients at each scale
  - Generate (and later manage) a 128-dimensional feature vector

- SURF was proposed to speed up the whole process by using effective approximations for each of these steps:
  - Using Haar wavelet approximations for keypoints detection (not presented here)
  - Using Haar wavelet approximations for feature descriptor
  - Using a 64-dimensional feature vector
- Bay et al. [1] introduced two versions for the SURF feature descriptor:
  - A more complete one, with an orientation assignment step similar to SIFT (not presented here)
  - A faster one (called U-SURF) that does not assign one. It still works well but clearly not rotationally-invariant

[1] H. Bay et al. SURF: Speeded Up Robust Features. ECCV, 2006

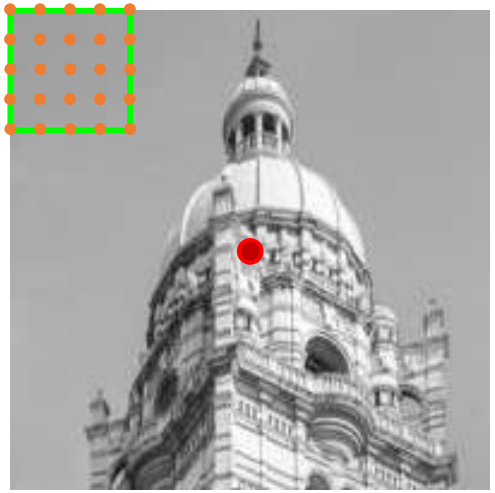
SURF: local image descriptor

Let's calculate a feature descriptor for each keypoint.
Main steps:

- A small window is positioned around each keypoint
- The window is divided in subregions
- For each subregion, 25 sample points (in a 5x5 grid) are selected
- For each sample point, two Haar wavelets are applied, with a size depending on the scale associated with each keypoint
- Haar wavelets sum the pixel intensities underneath the white portion and subtract the ones underneath the black one
- The wavelet responses are summed up (both with and without sign) within each subregion:

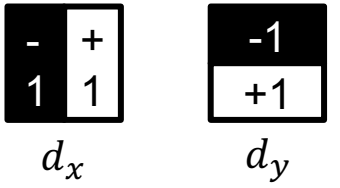
$$(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$$

- The descriptor for this subregion is defined by these 4 numbers



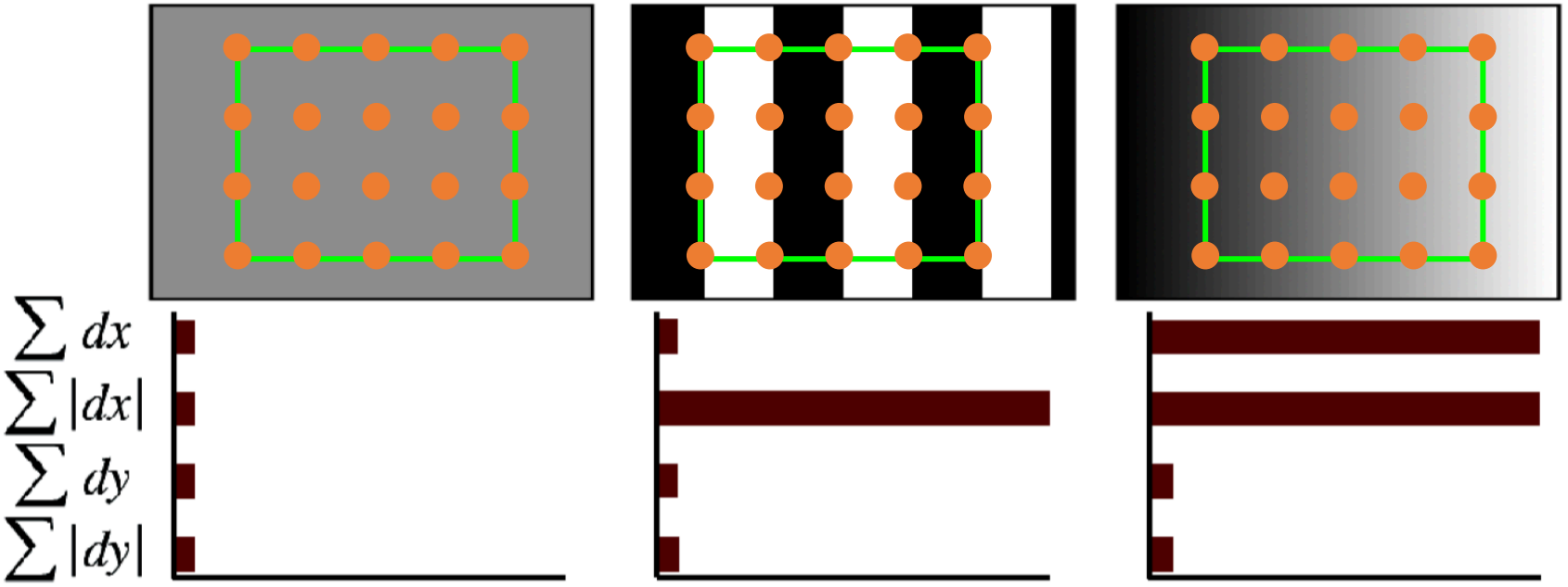
Subregion and sample points

Haar wavelets



SURF: local image descriptor

This seemingly simple descriptor can represent the nature of the image pattern for a given subregion:

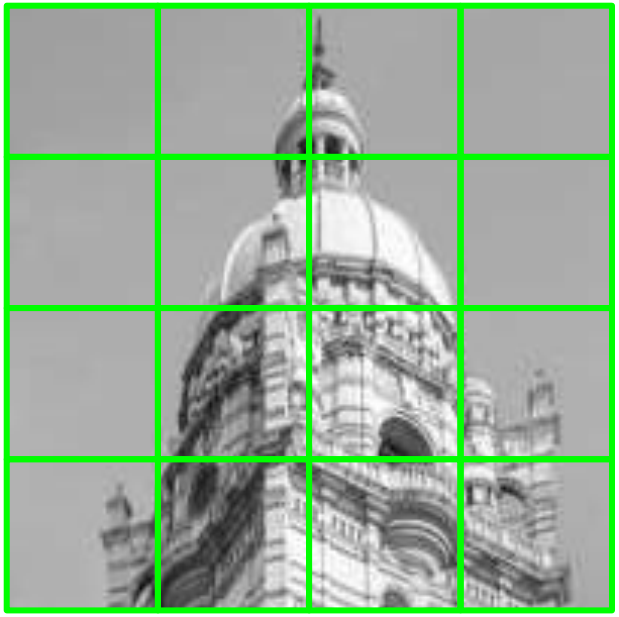


H. Bay et al. SURF: Speeded Up Robust Features. ECCV, 2006

SURF: local image descriptor

- In the original paper, 16 subregions were used in a 4x4 grid
- As a consequence, the number of elements of the SURF descriptor was
  16 subregions x 4 numbers = 64
- We use a feature vector of 64 elements to describe each keypoint

- SURF is much faster to compute than SIFT, and does not sacrifice performance too much

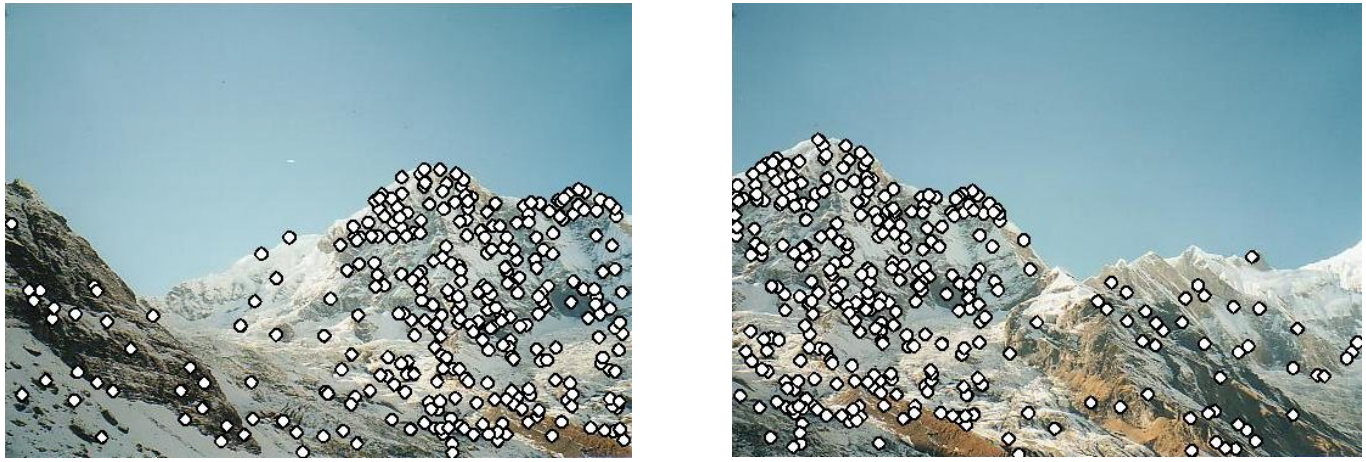


Built-in Matlab function: `detectSURFFeatures`, `extractFeatures`

H. Bay et al. SURF: Speeded Up Robust Features. ECCV, 2006

Image matching

- Now we have a feature descriptor for each interest point in the two images. How do we know which point corresponds to which?



- Matching is performed by **identifying the nearest neighbours** (NN):
  - Compute the (Euclidean) distance between each keypoint in image A and each keypoint in image B **in feature space** (e.g. 64D if using SURF)
  - For each keypoint in B, the closest one in A is its matching one
- Exact nearest neighbours is computationally expensive: use fast algorithms for approximate nearest neighbours (e.g. FLANN)
- Built-in Matlab function: `matchFeatures`

Image matching

After the keypoints have been matched, we can leverage the correspondences using their (spatial!) coordinates (exactly as in Lecture 2):

- Suppose that a keypoint (x, y) in image B corresponds to another keypoint (x', y') in image A and that we look for an affine transformation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y + a_{13} \\ a_{21}x + a_{22}y + a_{23} \\ 1 \end{bmatrix}$$

- For many keypoints, we can write

NN keypoint coord. in A

$$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ & & \vdots & & & \end{bmatrix} \cdot \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix}$$

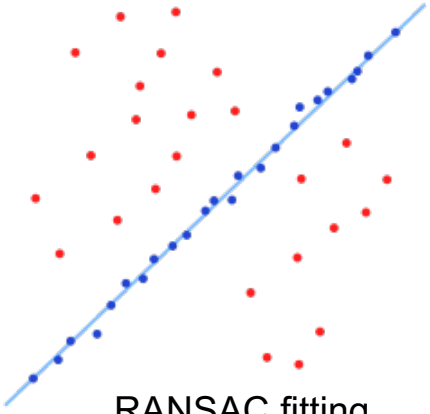
Keypoint coord. in B

- It can be written as a linear system $q = Ma$, which can be solved using least-squares (i.e. minimizing the squared difference error $\|Ma - q\|^2$)
- The solution is given by $a = (M^T M)^{-1} M^T q$
- Once the affine parameters are solved, we know the estimated spatial transformation between the two images and the matching is done

Image matching

- The squared difference $\|Ma - q\|^2$ is sensitive to outliers
- Some matching algorithms, such as Random Sample Consensus (RANSAC), take into account outliers during model fitting
- RANSAC iterative procedure:
  1. Select a random group of points
  2. Fit a line model (i.e. estimate a)
  3. Based on the line model, check how many points are inliers (i.e. reasonably close to their predicted versions using a)
  4. Go back to 1 until a certain number of iterations is reached or a model with enough inliers has been found

Data points



RANSAC fitting
(outliers; inliers)

https://en.wikipedia.org/wiki/Random_sample_consensus

Image stitiching & panorama

By matching the two images by interest points, we can stitch them to create a panorama

Image stitching / panorama

M. Brown. Automatic panoramic image stitching using invariant features, IJCV 2007

Image stitiching & panorama

By matching the two images by interest points, we can stitch them to create a panorama

Multiple images

Image stitching

Image blending (weighted average of overlapping regions)

M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. IJCV, 2007

Additional resources

- Matlab's tutorial for interest point detection, feature descriptors and image matching: https://uk.mathworks.com/help/vision/ug/local-feature-detection-and-extraction.html

Overview of next week's lecture

Image classification:

- Basics of machine learning for classification
- Learning paradigms (supervised, unsupervised, semi-supervised)
- Linear classifiers
- Support vector machines (SVMs)
  - Basics
  - Hard/soft margin
  - Hinge loss
  - Multi-class SVMs
- Ensembles
  - Committees
  - Boosting
  - Cascading classifiers

Object detection:

- Viola-Jones object detector