Feature Extraction and Matching

…@liris.cnrs.fr - http://liris.cnrs.fr/…

Laboratoire d'InfoRmatique en Image et Systèmes d'information

LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon
Université Claude Bernard Lyon 1, bâtiment Nautibus
43, boulevard du 11 novembre 1918 — F-69622 Villeurbanne cedex
http://liris.cnrs.fr

Object Recognition

- **Widely used in the industry for**
  - Inspection
  - Registration
  - Manipulation
  - Robot localization and mapping
- **Current commercial systems**
  - Correlation-based template matching
  - Computationally infeasible when object rotation, scale, illumination and 3D pose vary
  - Even more infeasible with partial occlusion
- **Alternative: Local Image Features**

Local Image Features

- Local features are robust to
  - Nearby clutter
  - Partial occlusion
- Invariant to
  - Illumination
  - 3D projective transforms
  - Common object variations
- Distinctiveness
  - Can differentiate a large database of objects
- Quantity
  - Hundreds/thousands in a single image
- Efficiency
  - Real-time performance

Related work

- Line segments, edges and regions grouping
  - Detection not good enough
- Peaks detection in local image variations
  - Example: Harris corner detector
  - Drawback: only a single scale
  - Key locations varies with the image scale changes
- Eigenspace matching, color and receptive field histograms
  - Successful on isolated objects
  - Unextendable to cluttered and partially occluded images

Contents

- Feature point matching
  - What is it?
  - What is it for?
- Feature point detection
  - Moravec feature point detector
  - Harris corner detector
  - Scale space detection
- Feature point extraction & matching
  - Matching using templates
    - Cross-correlation

Feature point matching

- **Feature Point:**
  - **Useful for image processing (by human or computer)**
  - **Local properties in the neighbourhood of the point**

- **A key issue for feature point is matching process, which has in general has 3 steps:**
  - **Feature point detection**
  - **Feature point extraction**
  - **Feature point matching**

Feature point matching

- Useful for
  - Motion detection
  - Object tracking
  - Object recognition
  - Multi-view reconstruction
    - Stereo+ vision
    - Structure from motion
  - Image stitching
  - Localisation
  - Simultaneous Localisation And Mapping

Feature point matching

Object tracking

- Useful for 3D localisation
- Resistant to occlusion (local features)
- Resistant to clutter (local features)

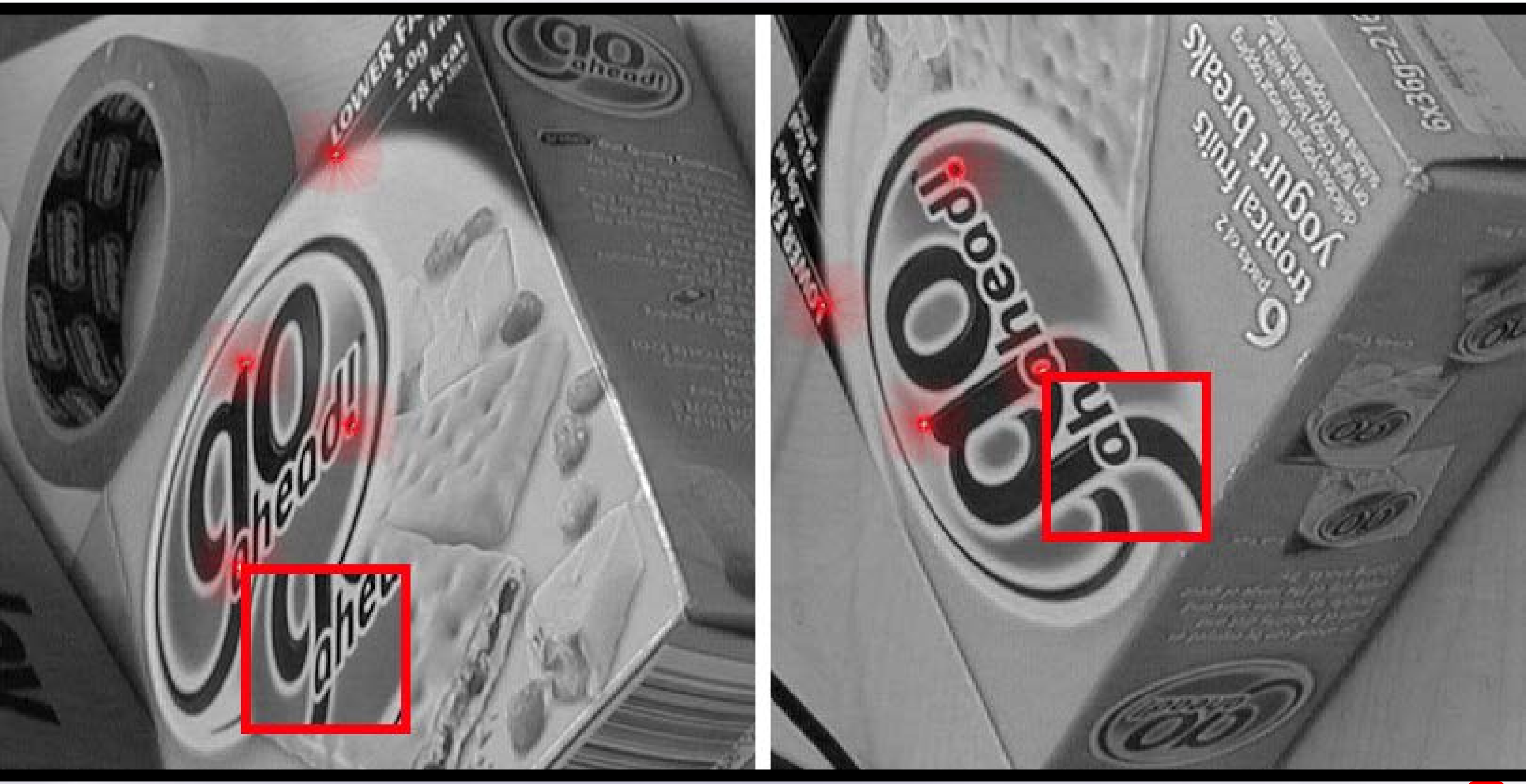
Motion detection

- Defeats aperture problem (distinctive features)

Feature point matching - phases

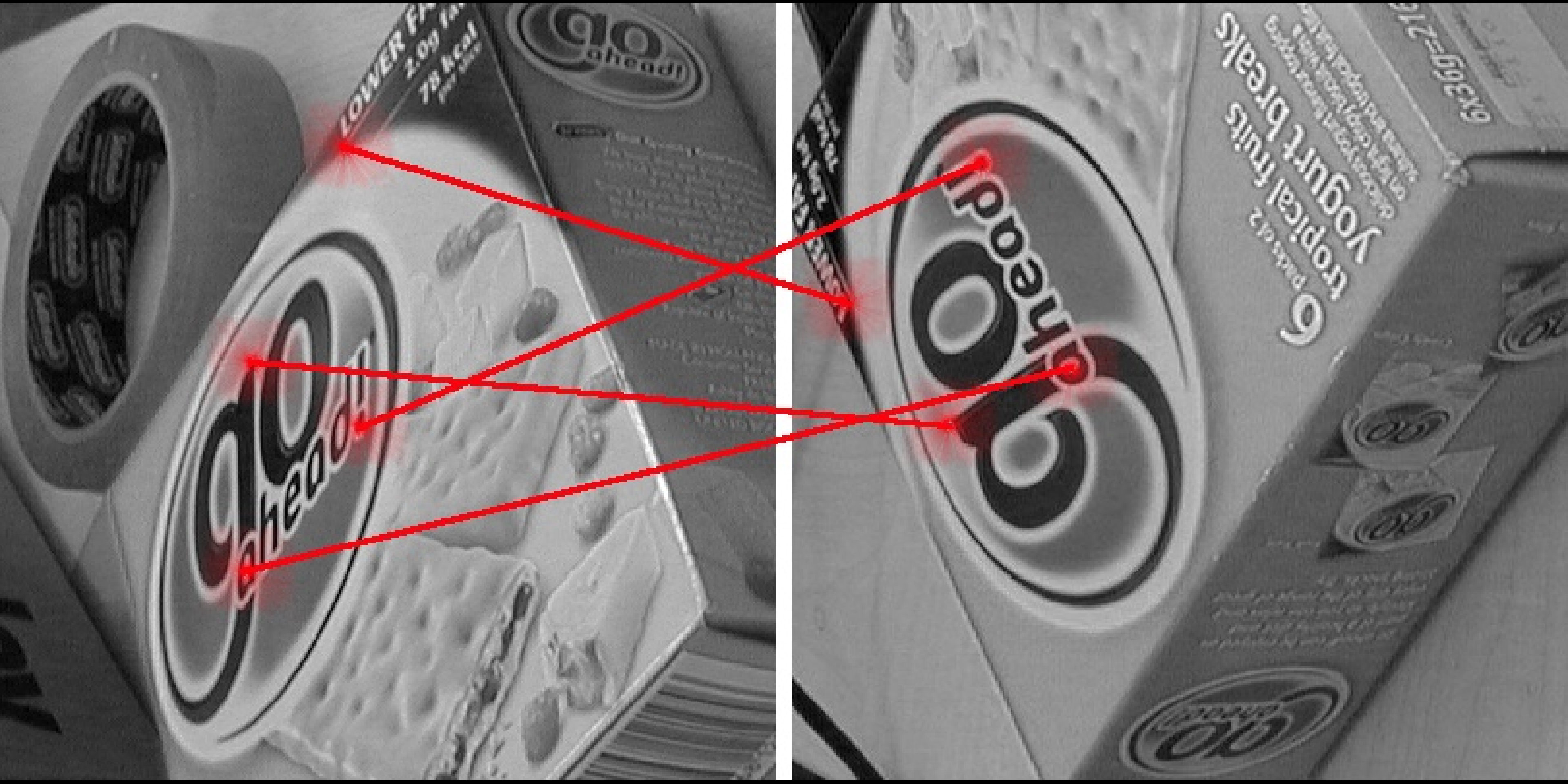
- **Feature Point Detection:**
  - The process of finding such useful points in an image in the first place. One common example is the Harris "Corner Detector"
- **Feature Point Extraction:**
  - The process of finding a description of the local properties of an image (its features) around a feature point (this might just be a patch extracted from the image)
- **Feature Point Matching:**
  - The process of using the local properties of images around feature points to identify the points across images that refer to the same points in the world

Matching between images



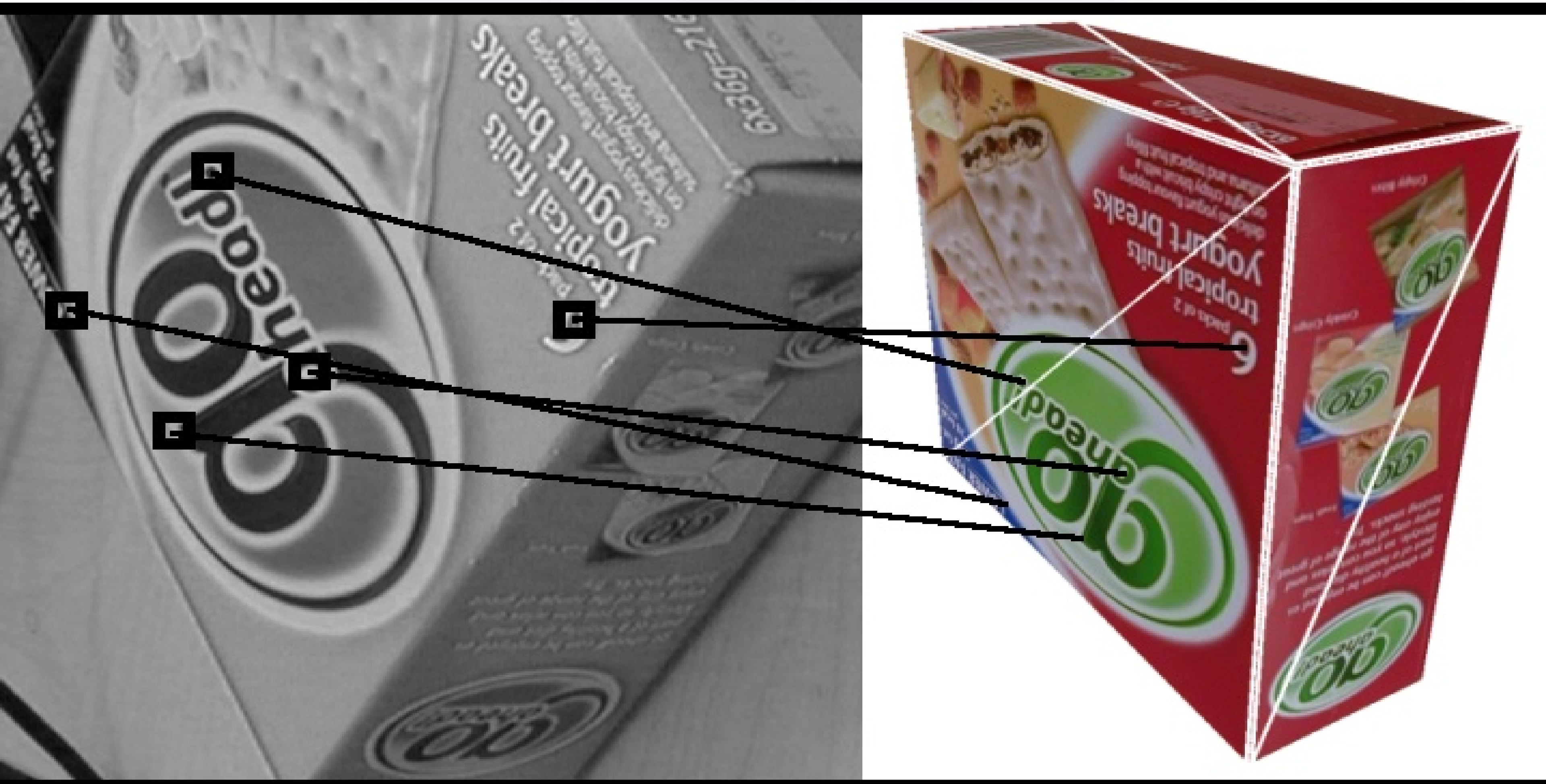
Extraction

Matching between images



Matching to a known textured object

Matching to a known textured object



Feature point detection

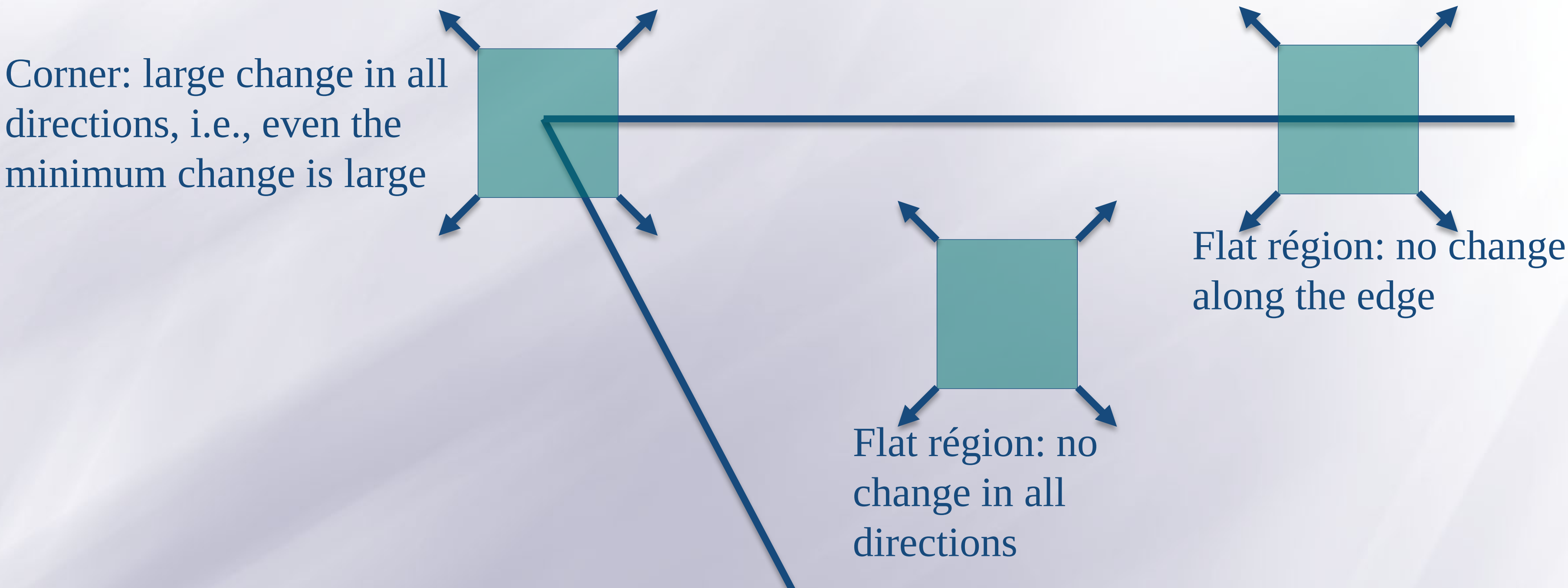
- Probably because it is not so well defined, feature point detection is an ill-posed problem
  - Change the matching method, and the criteria for a good feature point changes
- Many detection methods exist, such as
  - Moravec (1977)
  - Harris (1988)
  - Scale space extrema (1999)
- Often conceptualised as a search for a generalised “corner”, but other definitions work!

Feature point detection

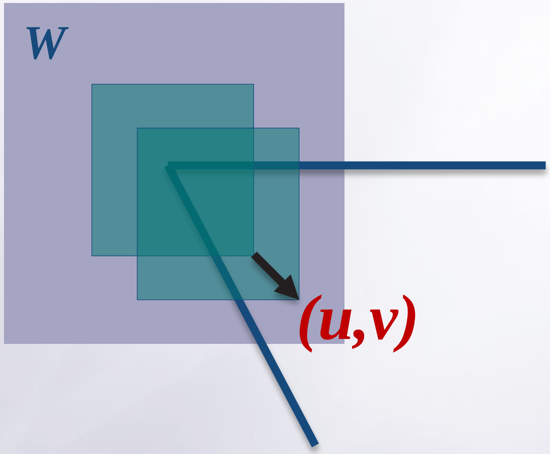
- General Desiderata
  - Points repeatably detectable under transforms
  - Computationally efficient
  - Easily localisable
  - High detection rate
  - Robust to noise
- Useful for
  - Finding candidates for point matching
  - Creating salience maps (see previous lecture on attention)

Some theory…

How does the window change when it is shifted?



Find locations that imply the minimum change by shifting the window in any direction is large



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

- how do the pixels in ***W*** change?
- compare each pixel before and after using the Sum of Squared Differences (SSD)
- this defines an SSD "error" *E(u,v)*:

$$E(u,v) = \sum_{(x,y)\epsilon W} [I(x+u,y+v) - I(x,y)]^2 \qquad \textbf{(1)}$$

Taylor Series expansion of *I*:

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

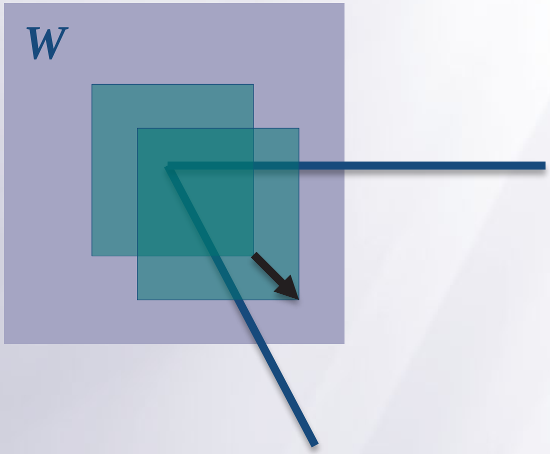
$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}; \;\; I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y} \qquad \textbf{(2)}$$

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

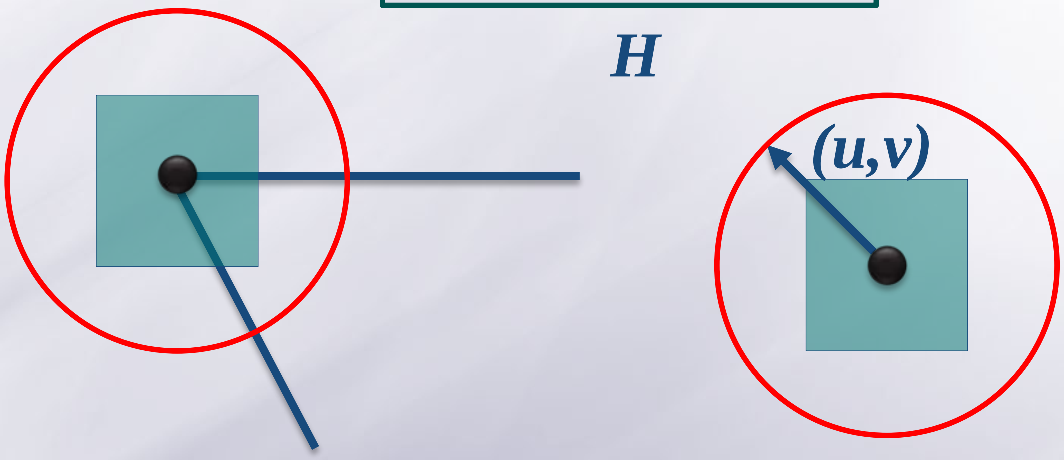
$$\approx \sum_{(x,y)\epsilon W} \left[I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y) \right]^2$$

$$\approx \sum_{(x,y)\epsilon W} \left[\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$



$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\boxed{\left(\sum_{(x,y)\in W} \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix} \right)}}_{\boldsymbol{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u,v) \approx [u \quad v] \underbrace{\boxed{\left(\sum_{(x,y)\in W} \begin{bmatrix} I_x{}^2 & I_x I_y \\ I_x I_y & I_y{}^2 \end{bmatrix} \right)}}_{\boldsymbol{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$



For the example above:

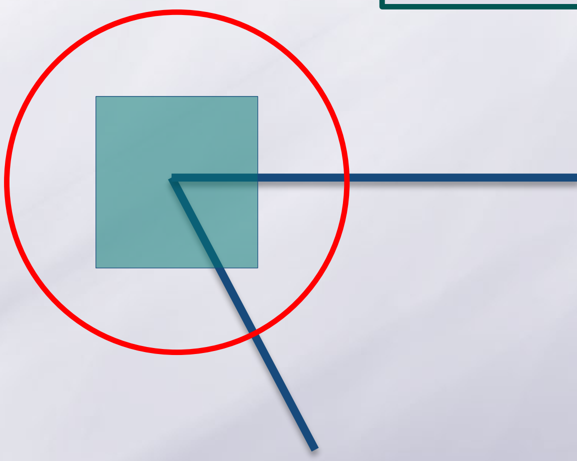
- You can move the center of the blue window to anywhere on the red unit circle
- How do we find directions that will result in the largest and smallest *E* values?
- Find these directions by looking at the eigenvectors of ***H***

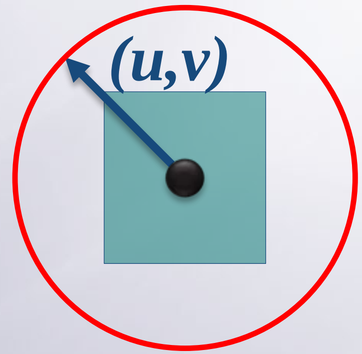
The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy: $Ax = \lambda x$

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

- The eigenvalues are found by solving: $det(A - \lambda I) = 0$
- In our case, ***A***= ***H*** is a 2x2 matrix, thus: $det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$
- The solution: $\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4\, h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$
- Once you know λ, you find ***x*** by solving: $\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$

$$E(u,v) \approx [u \quad v] \underbrace{\boxed{\left(\sum_{(x,y)\in W} \begin{bmatrix} I_x{}^2 & I_xI_y \\ I_xI_y & I_y{}^2 \end{bmatrix} \right)}}_{\boldsymbol{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$



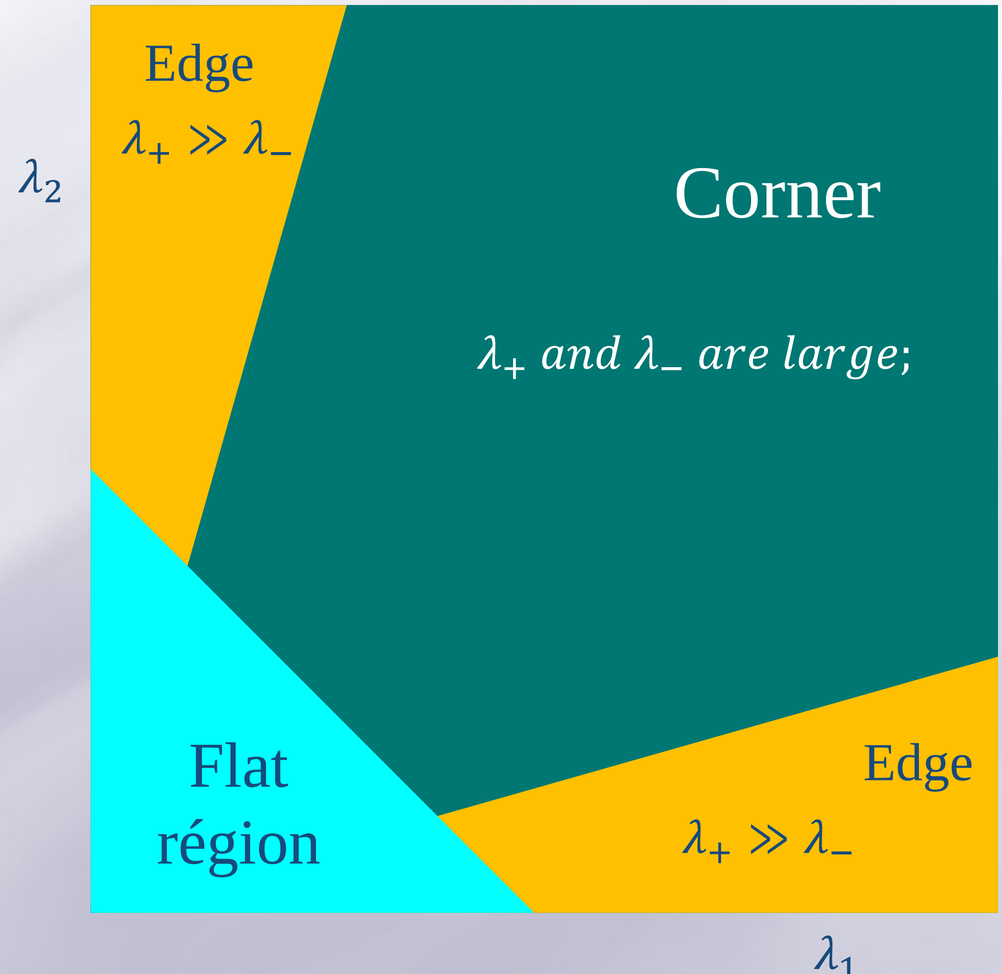


Eigenvalues and eigenvectors of H:

- Capture shifts with the smallest and largest change (E value)
- x_+= direction of **largest** increase in E
- λ_+= amount of increase in direction x_+
- x_-= direction of **smallest** increase in E
- λ_-= amount of increase in direction x_-

$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

How are λ_+, $\boldsymbol{x}_+$, λ_-, and $\boldsymbol{x}_-$ relevant for feature detection?



Want *E(u,v)* to be ***large*** in ***all*** directions

- the *minimum* of *E(u,v)* should be large over all unit vectors [*u v*]
- this minimum is given by the smaller eigenvalue λ_- of ***H***
- Look for large values of λ_-

Algorithm for interest point detection

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues
- Find points with large λ_- (i.e., $\lambda_- >$ threshold)
- Choose points where λ_- is a local maximum as interest points

Invariant to **rotation** of the image by some angle

- Will you still pick up the same feature points? Yes (since eigenvalues remain the same)

What about the change of the **brightness**?

- Will you still pick up the same feature points? Mostly yes (uses gradients which involve pixel differences)

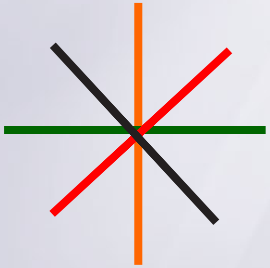
Scale?

- No!

Lets practice that…

Feature point detection - Moravec

- Recall Moravec point detector
  - Used on the Stanford Cart in the 70s
  - At each pixel an interest score is calculated and local maxima in this score are used (i.e. where the score at a pixel is higher than in all other pixels in the neighbourhood)
  - This score at a point is the change in intensity around the point in the direction that the intensity changes slowest – BUT you only check 4 key directions around the point



The Stanford Cart

The Stanford Cart was a long-term research project undertaken at Stanford University between 1960 and 1980. In 1979, it successfully crossed a room on its own while navigating around a chair placed as an obstacle. Hans Moravec rebuilt the Stanford Cart in 1977, equipping it with stereo vision. A television camera, mounted on a rail on the top of the cart, took pictures from several different angles and relayed them to a computer.

http://www.computerhistory.org/timeline/1979/

Feature point detection - Moravec

Recall Moravec point detector

- **Here is how to calculate the interest score:**

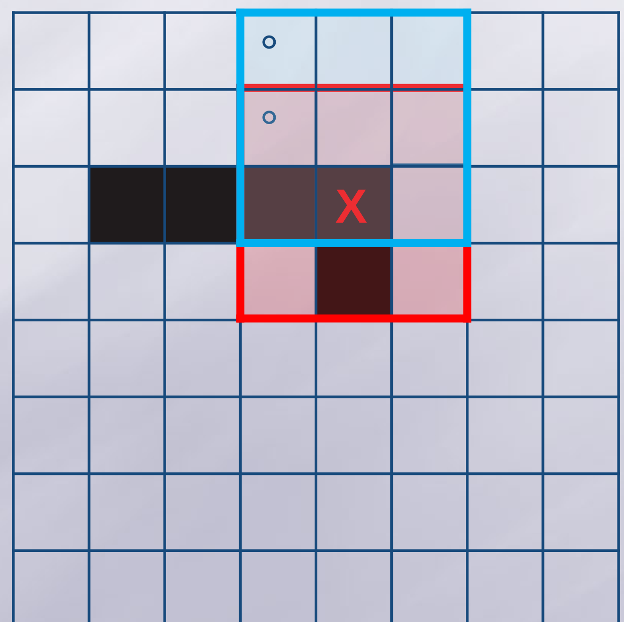
$$Score(x,y) = \min_{u,v \in P}\{ \sum_{x,y \in W} [I(u+x, v+y) - I(x,y)]^2 \}$$

- **Where P is the set of perturbations**
  - **{ (-1,1),(0,1),(1,0),(1,1)}**
- **W is the local window. I image intensity. x, y, u, v pixel indices.**
- **Example to follow**

Feature point detection - Moravec

Moravec point detector

Sum(
$(1-1)^2=0$
$(1-1)^2=0$
$(1-1)^2=0$
$(1-1)^2=0$
$(1-1)^2=0$
$(0-0)^2=0$
$(1-0)^2=1$
$(0-1)^2=1$
$(0-1)^2=1$
)=3



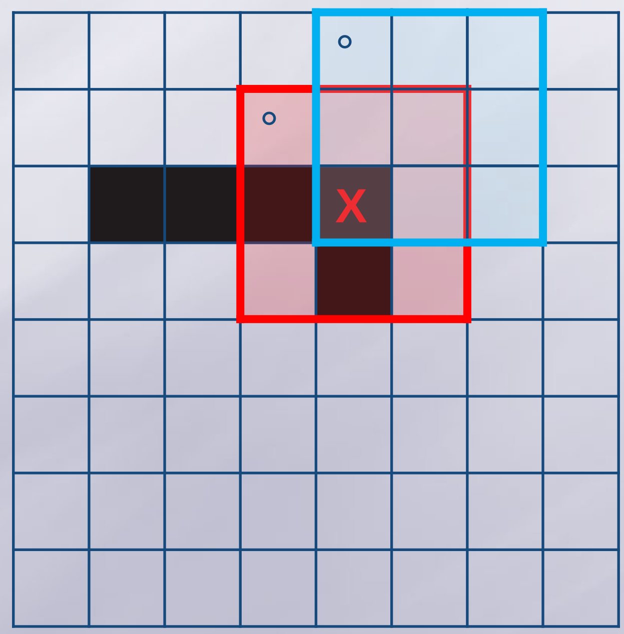
Local Window
W(x,y)

Shifted Window
W(x,y+1)

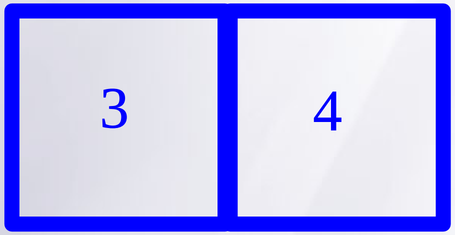
$$Score(x,y) = \min_{u,v \in P}\{ \sum_{x,y \in W} [I(u+x, v+y) - I(x,y)]^2 \}$$

Feature point detection - Moravec

Moravec point detector



Local Window
W(x,y)



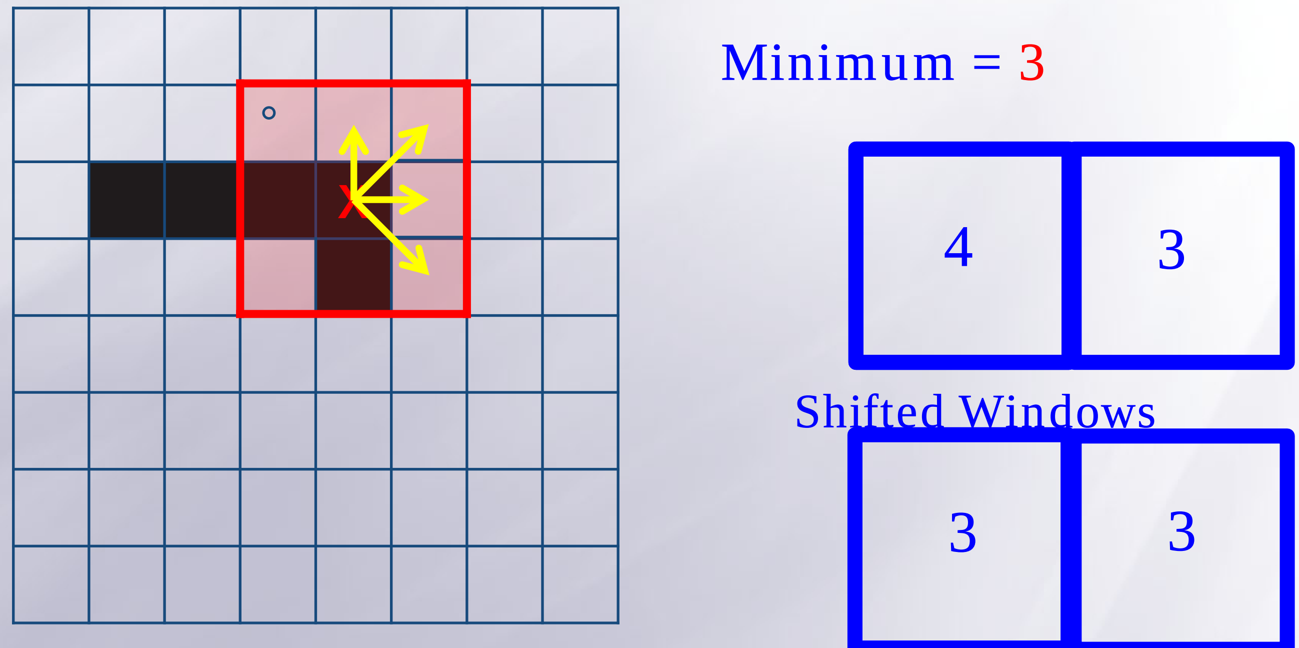
Shifted Window
W(x+1,y+1)

Sum(
$(1-1)^2=0$
$(1-1)^2=0$
$(1-1)^2=0$
$(1-1)^2=0$
$(1-1)^2=0$
$(0-1)^2=1$
$(1-0)^2=1$
$(0-1)^2=1$
$(0-1)^2=1$
)=4

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

Feature point detection - Moravec

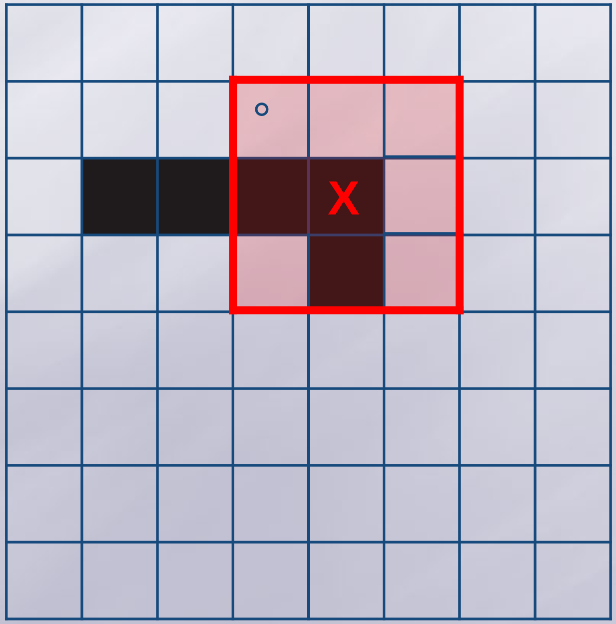
Moravec point detector



$$Score(x,y) = \min_{u,v \in P}\left\{ \sum_{x,y \in W} \left[I(u+x, v+y) - I(x,y)\right]^2 \right\}$$

Feature point detection - Moravec

Moravec point detector



Minimum = 3

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

Feature point detection - Moravec

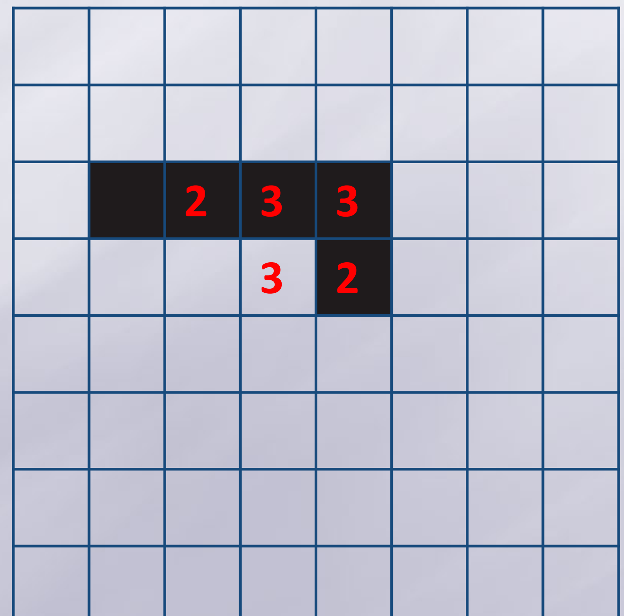
Moravec point detector

0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0
1	1	2	3	3	1	0	0
1	1	1	3	2	1	0	0
0	0	0	1	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

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

Feature point detection - Moravec

Moravec point detector

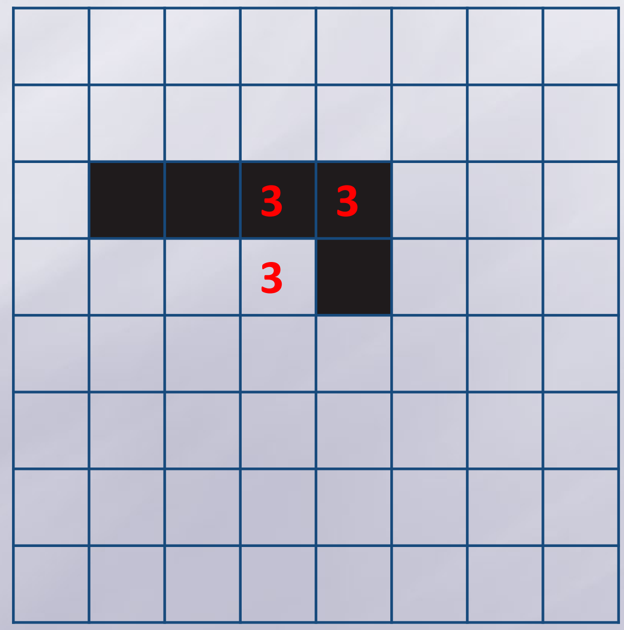


Thresholding...

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

Feature point detection - Moravec

Moravec point detector

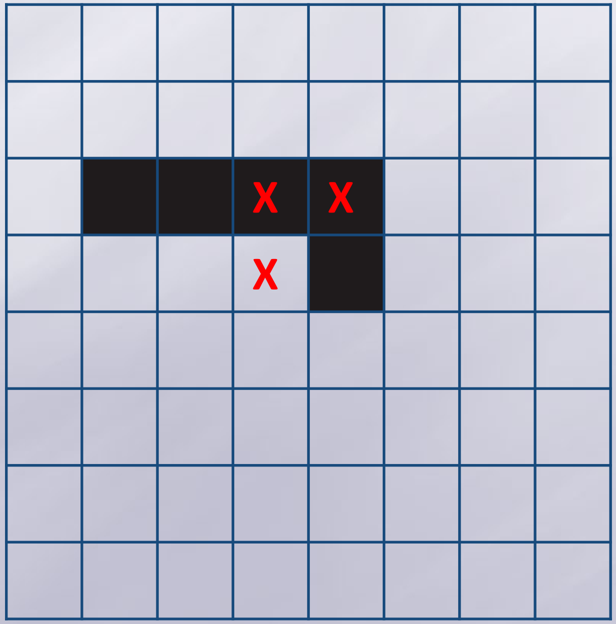


Performing non-maximal suppression...

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

Feature point detection - Moravec

Moravec point detector



Corner points!

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

Feature point detection

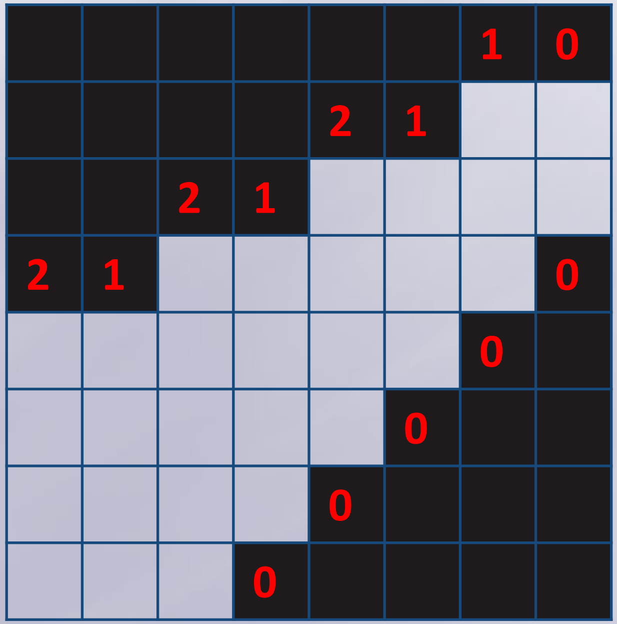
- Moravec point detector
  - Fast
  - Terrible with angles away from 45 deg
  - Has problems with noise
  - Not invariant to scale
  - Not invariant to rotation
  - In other words, obsolete.
    - More robust detectors exist.

Feature point detection - Moravec

Moravec point detector

- **Terrible with angles away from 45 deg**

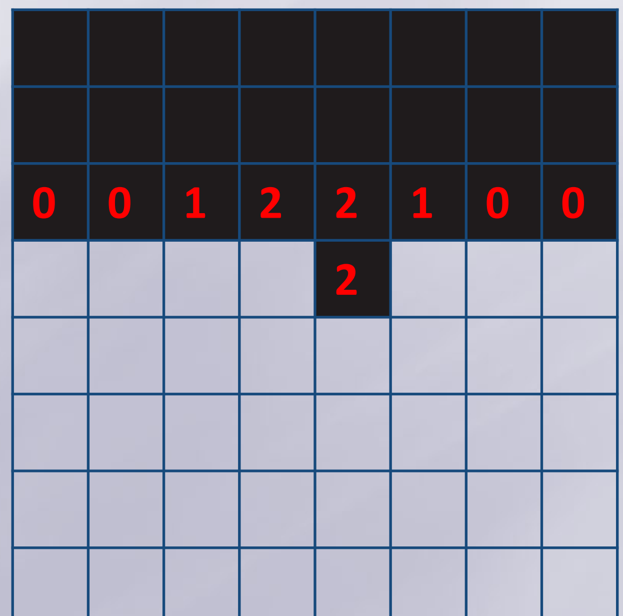
- Lots of false matches away from the diagonal.



Feature point detection - Moravec

Moravec point detector

- Has problems with noise



Feature point detection - Moravec

- Moravec point detector
  - Implicit assumption is that points of interest are points where the image brightness changes fast in all directions
  - More recent approaches have attempted to create better detectors, incorporating the same basic assumption
    - But without the limitation of 4 cardinal directions

Feature point detection – Harris

- The Harris detector generalises the Moravec detector:
  - Attempting to calculate the strength of image change in a window in the direction of largest change and of smallest change (rather than in 4 principal directions, which allows for better rotational invariance)
  - Using a circular gaussian window rather than a square one (less noise)
  - Taking into account both the direction of minimum change in brightness and maximum change in brightness (to reduce the probability of picking up edges)

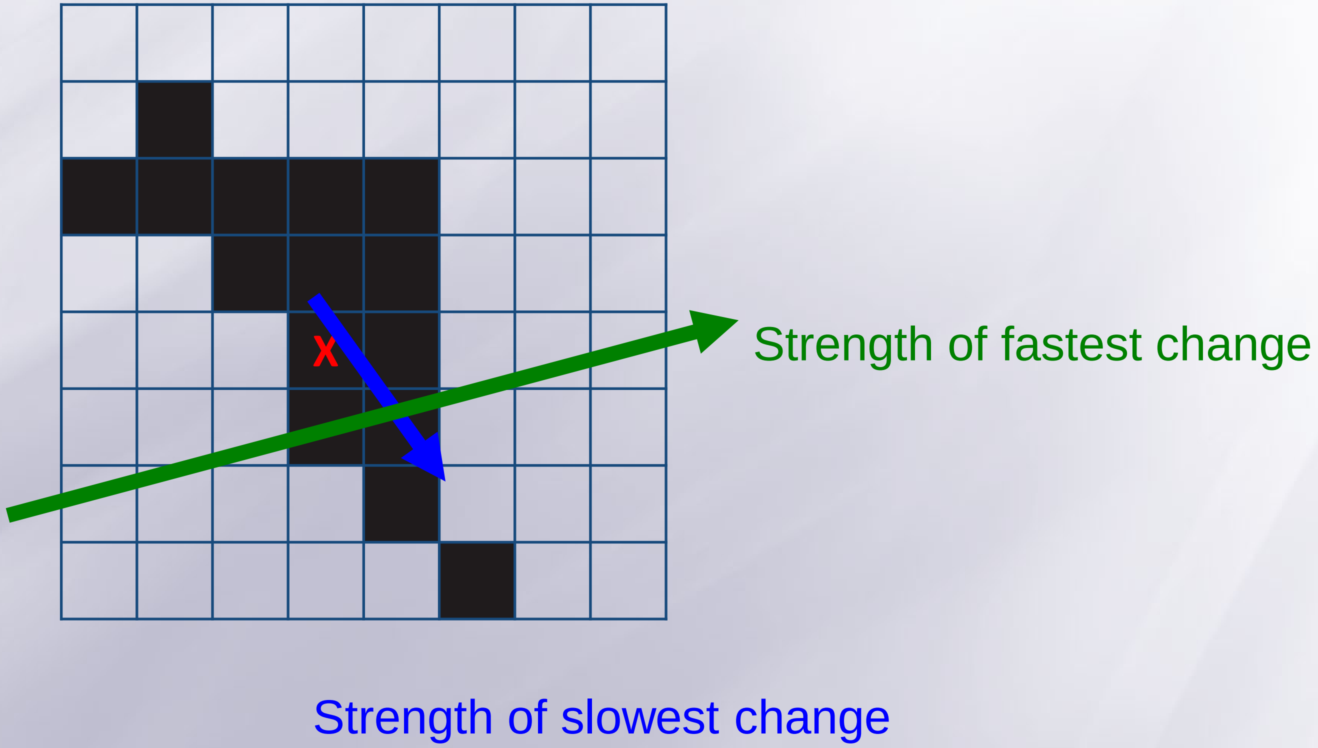
Feature point detection – Harris

- **The Harris detector generalises the Moravec detector:**

Strength in direction of slowest change is <u>low</u>

Strength in direction of fastest change is <u>high</u>...

...so this must be an edge pixel!



Feature point detection – Harris

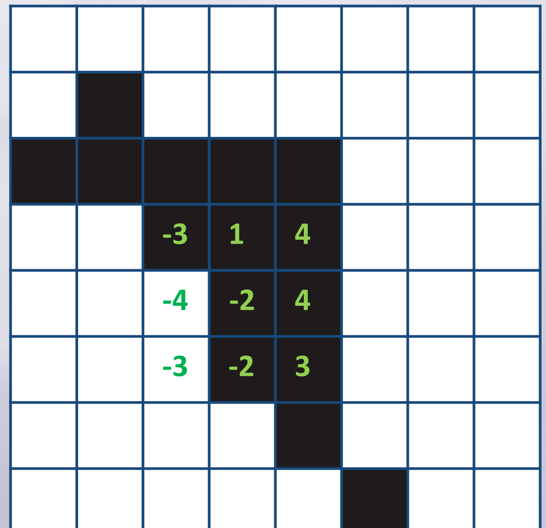
How to calculate these strengths?

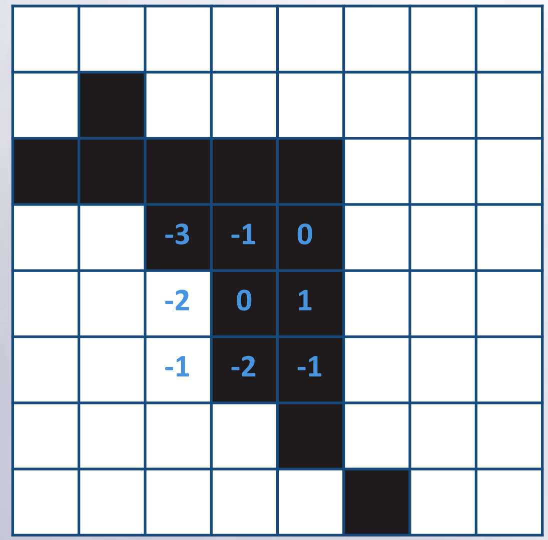
Estimate the

X and Y gradients

for all pixels in the region of your point of interest.

(this can be done efficiently by convolution with a small mask.
e.g a sobel mask:)





-1	0	1
-2	0	2
-1	0	1

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

1	2	1
-0	0	0
-1	-2	-1

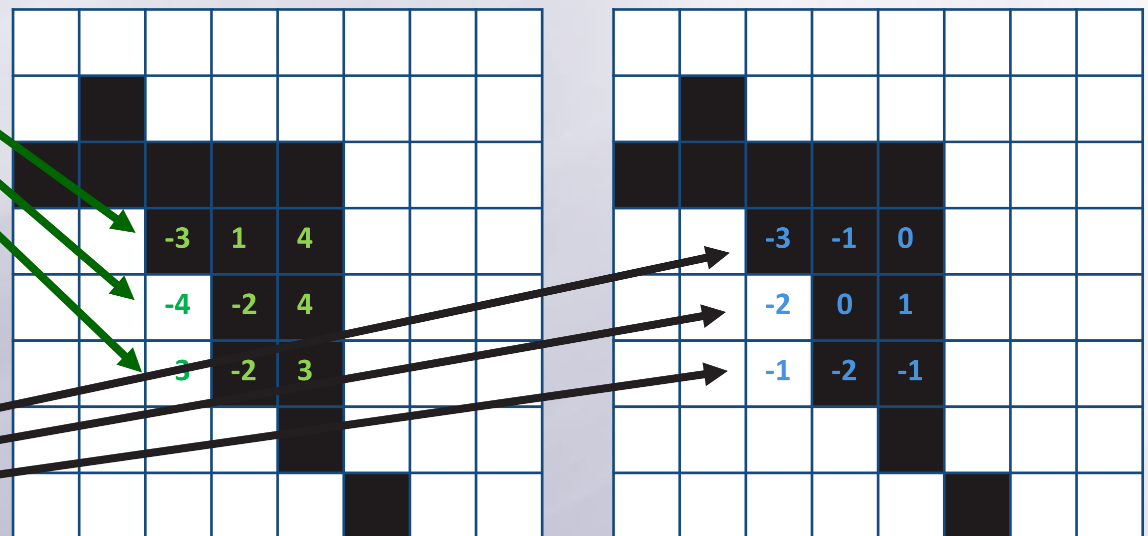
$$\frac{\partial I}{\partial y}$$

Feature point detection – Harris

How to calculate these strengths?

The sum of squares of these values will give us an idea of the strength of change in the x direction.

The sum of squares of these values will give us an idea of the strength of change in the y direction.

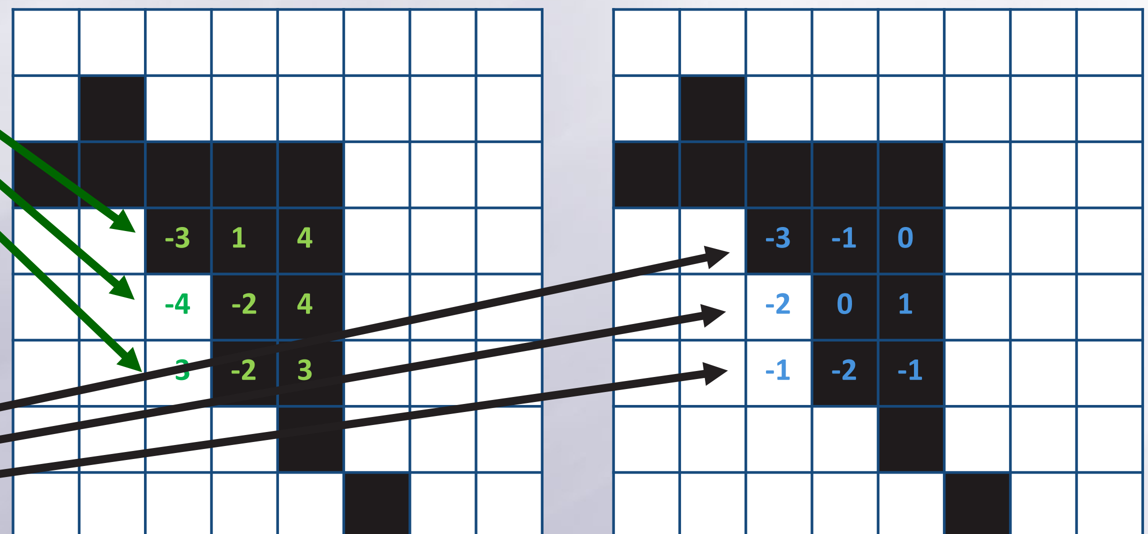


$$\frac{\partial I}{\partial x} \qquad \frac{\partial I}{\partial y}$$

Feature point detection – Harris

How to calculate these strengths?

The sum of squares of these values will give us an idea of the strength of change in the x direction.



The sum of squares of these values will give us an idea of the strength of change in the y direction.

Very similar to the Moravec operator… which wasn't good enough for us!

We need a way to calculate the strength of change in the <u>directions of maximum & minimum change.</u>

Feature point detection – Harris

How to calculate these strengths?

$$str_W(\Delta x, \Delta y) = \sum_W \left(\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y\right)^2$$

With:

$$\sqrt{(\Delta x)^2 + (\Delta y)^2} = 1$$

Check the following special cases:

$$\{\Delta x = 1, \Delta y = 0\},\ \{\Delta x = 0, \Delta y = 1\},\ \left\{\Delta x = \frac{1}{\sqrt{2}}, \Delta y = \frac{1}{\sqrt{2}}\right\},\ \left\{\Delta x = \frac{1}{\sqrt{2}}, \Delta y = -\frac{1}{\sqrt{2}}\right\}$$

it's the same as Moravec's!

Feature point detection – Harris

- How to calculate these strengths?

$$str_W(\Delta x, \Delta y) = [\Delta x \;\; \Delta y] \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

- Now we can analyse this matrix:

$$M_W = \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

Feature point detection – Harris

- How to calculate these strengths?

$$str_W(\Delta x, \Delta y) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

- It turns out that if we assume that $\sqrt{(\Delta x)^2 + (\Delta y)^2} = 1$

then the maxima and minima of this function are the eigenvalues of matrix M_W, λ_1, λ_2

Feature point detection – Harris

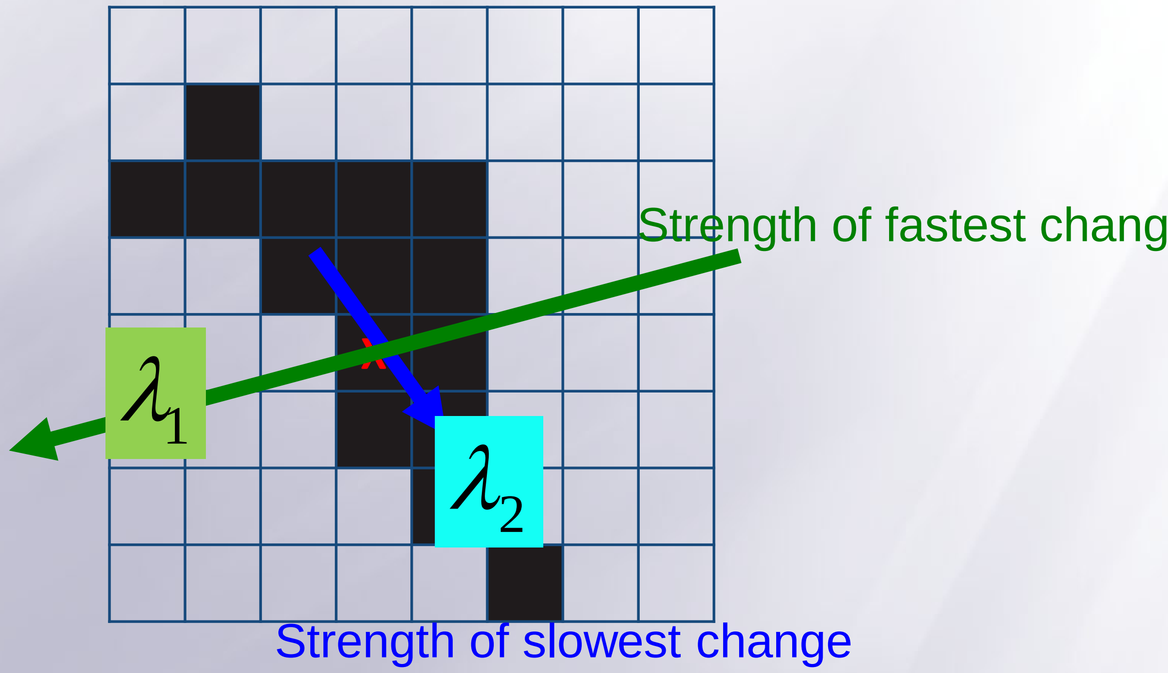
How to calculate these strengths?

Now let's try to calculate M_W from our data:

$$\lambda_1 = 101.2$$

$$\lambda_2 = 13.8$$

$$\begin{bmatrix} 84 & 24 \\ 24 & 21 \end{bmatrix} = M_W$$

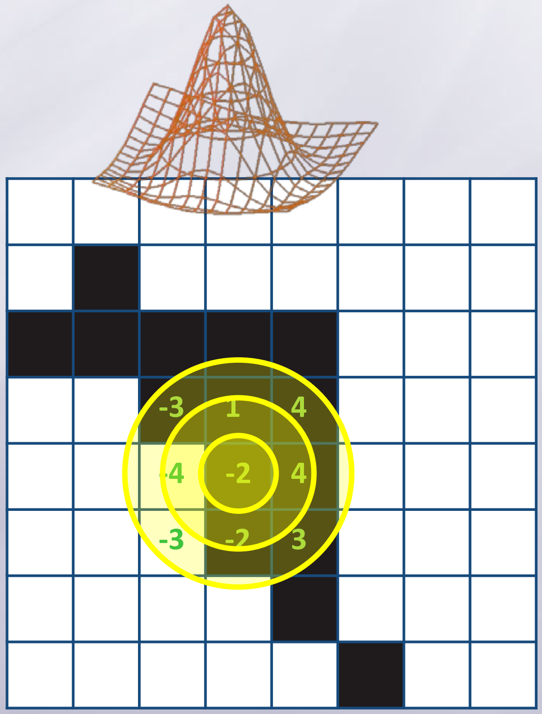


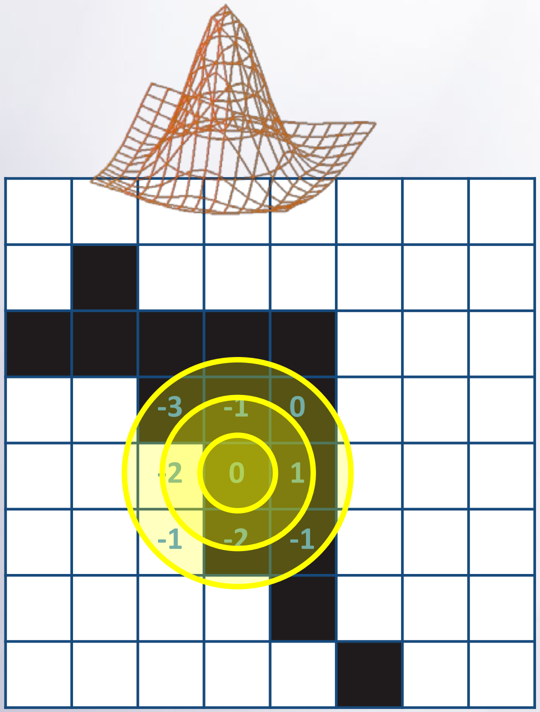
Feature point detection – Harris

How to calculate these strengths?

Usually we use a circular gaussian window when summing to create our matrix M_W.

$$\begin{bmatrix} \sum_W g_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W g_W \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right) \\ \sum_W g_W \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right) & \sum_W g_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$





We sum up the gradients for all nearby pixels, weighted by distance.

Feature point detection – Harris

- How to calculate these directions?

- Actually, because it is only the relationship between the eigenvalues that is of interest, and not the values themselves, the eigenvalues of M are usually not computed directly. Instead the interest score is computed:

$$Score(x,y) = \det(M_W) - k \cdot \mathrm{trace}(M_W)^2$$

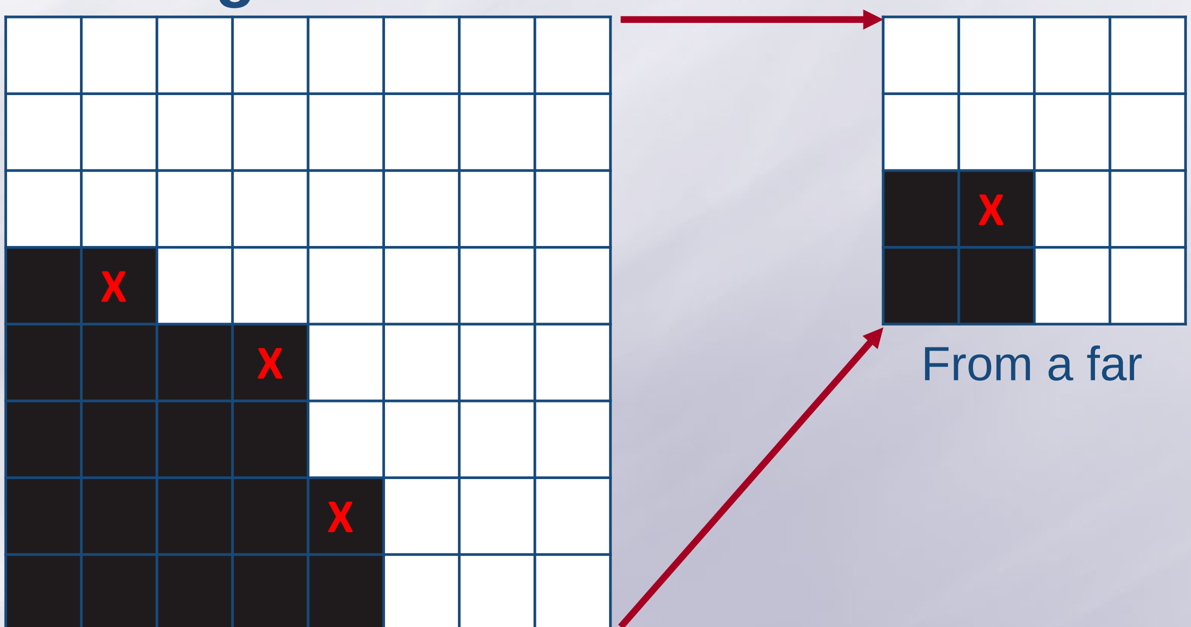
- Where k is some constant ~ 0.04

Feature point detection – Harris

- The Harris detector generalises the Moravec detector:
  - Moravec did not generalise to non-cardinal directions but the Harris operator does
  - Generalisations of the Harris detector exist too, for larger stability with scaling (generally using pyramidal images)

Feature point detection – Harris

- **The Harris detector has problems with scale changes**

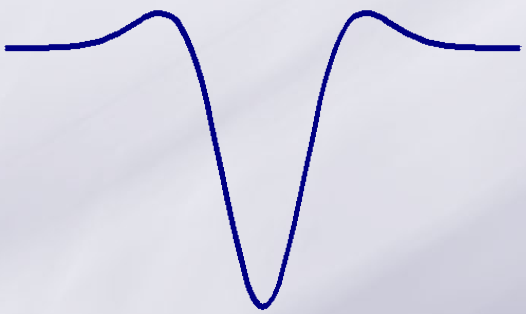


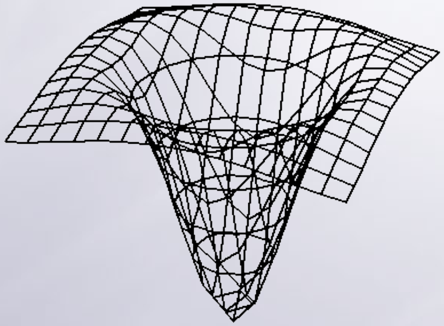
Up close

- **Where is the interest point really?**

Feature point detection – scalespace

- Before getting into scale-space detection, see your use of Difference of Gaussian (DoG) filters in salience maps - also often used is the very similar Laplace of Gaussian (LoG)

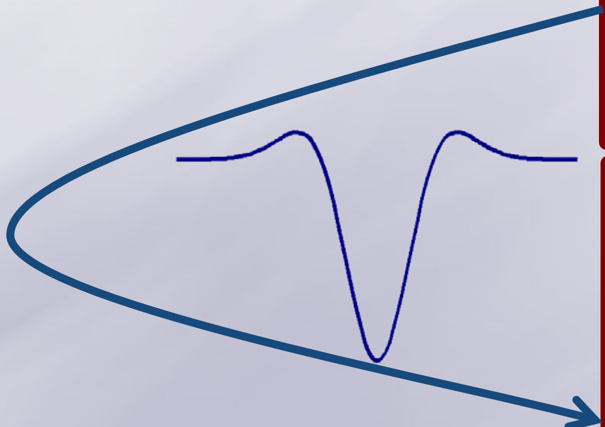


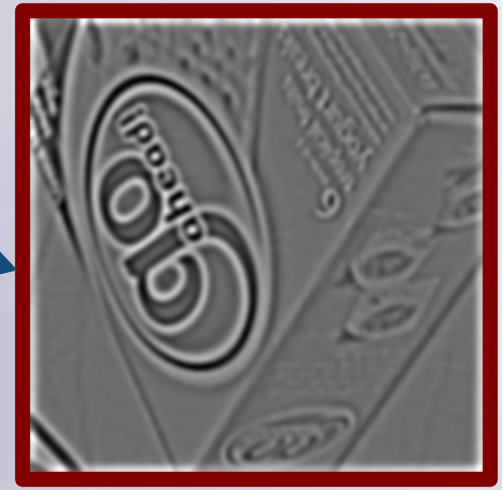


- It turned out that these were useful filters for finding the salience of each point in an image
- Zero crossings in images filtered with DoG/LoG can also be used for finding edges
- LoG/DoG are also useful for finding interest points

Feature point detection – scalespace

- Here is an image





- Here is the DoG of that image

Feature point detection – scalespace

- Local maxima and minima in the DoG image correspond to interesting points in our original image…
- This is a new definition of interest point: DoG extrema.
- …i.e. points that have a larger or lower DoG response than all their neighbours.

Feature point detection – scalespace

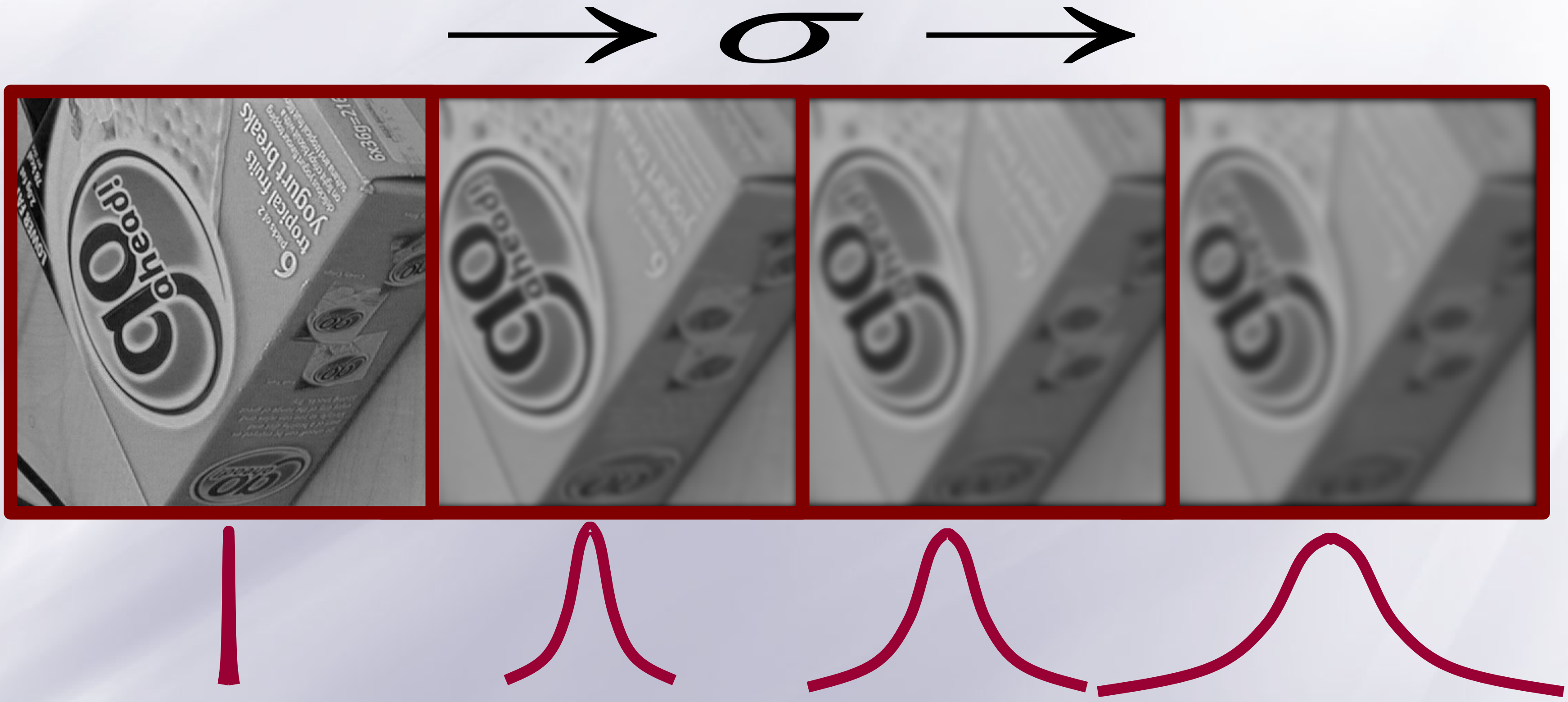
- **DoG extrema are about as good as Harris points:**
  - They are resistant to rotation (since the filter is rotationally invariant)
  - They are relatively resistant to noise (because they are the response of a smooth filter)
- **BUT:**
  - Like Harris points they are not robust to changes in scale

Feature point detection – scalespace

- **Problems with scaling motivate another concept – that of scale space**
- **The concept of scale space motivates another tentative definition for feature points – DoG scale space extrema**

Feature point detection – scalespace

- Let us examine this notion of scale space



Feature point detection – scalespace

- **Let us examine this notion of scale space**

$$I(x,y)$$

Image Function

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+v^2)/2\sigma^2}$$

Gaussian Function

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

Gaussian Scale Space

* the star here is convolution

Feature point detection – scalespace

- **Let us examine this notion of scale space**

$$\longrightarrow \sigma \longrightarrow$$

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

Gaussian Scale Space

* the star here is convolution

Feature point detection – scalespace

- **Let us examine this notion of scale space extrema**

The scale space extrema of interest to us
are maxima and minima
in the DoG scale space function.

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

Difference of Gaussian Scale Space Function

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$$

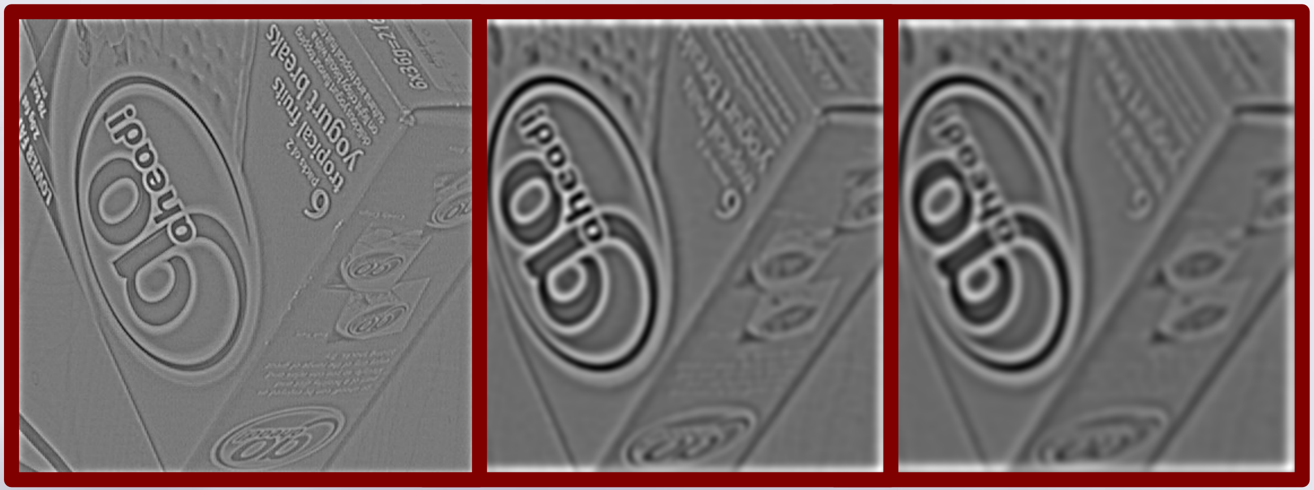
Equivalent Form – Difference of Gaussian Scale Space Function
(sometimes faster to compute)

* the star here is convolution

Feature point detection – scalespace

- **Here is the scale space function for the DoG function...**

$$\longrightarrow \quad \sigma \quad \longrightarrow$$

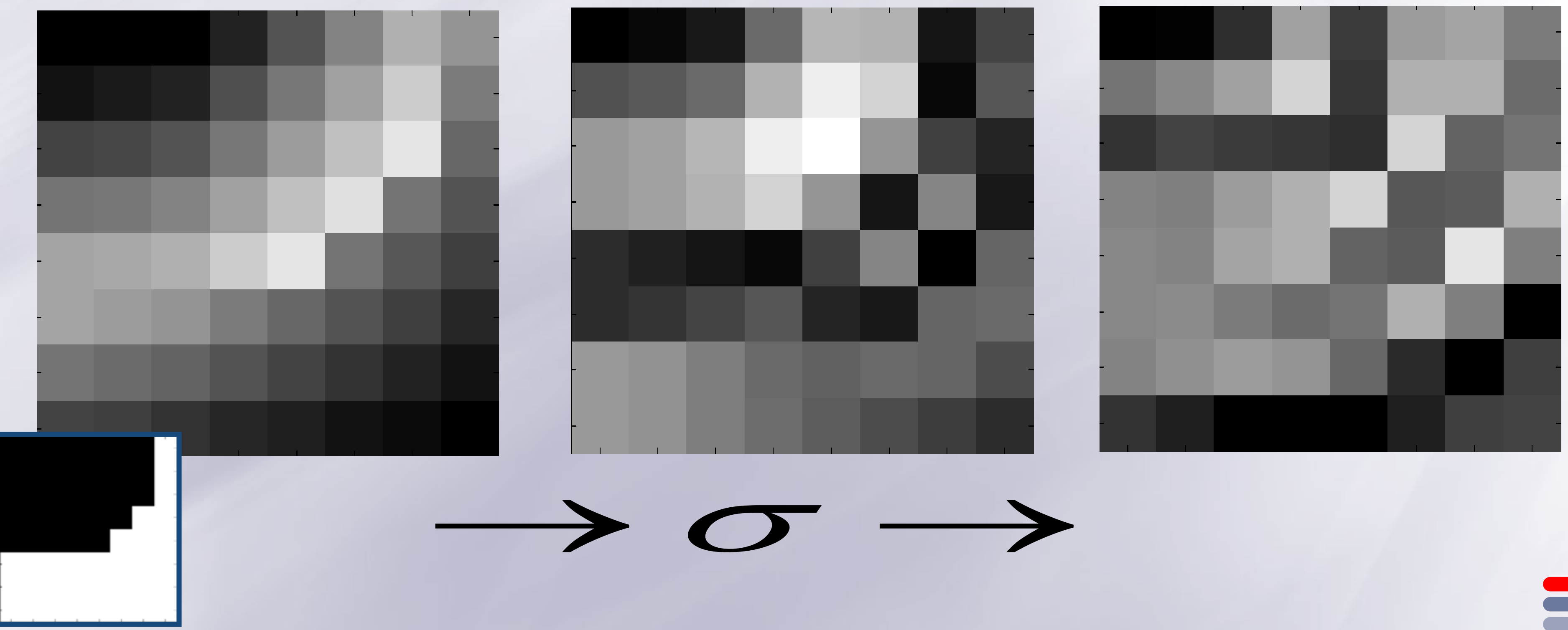


$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

Difference of Gaussian Scale Space Function

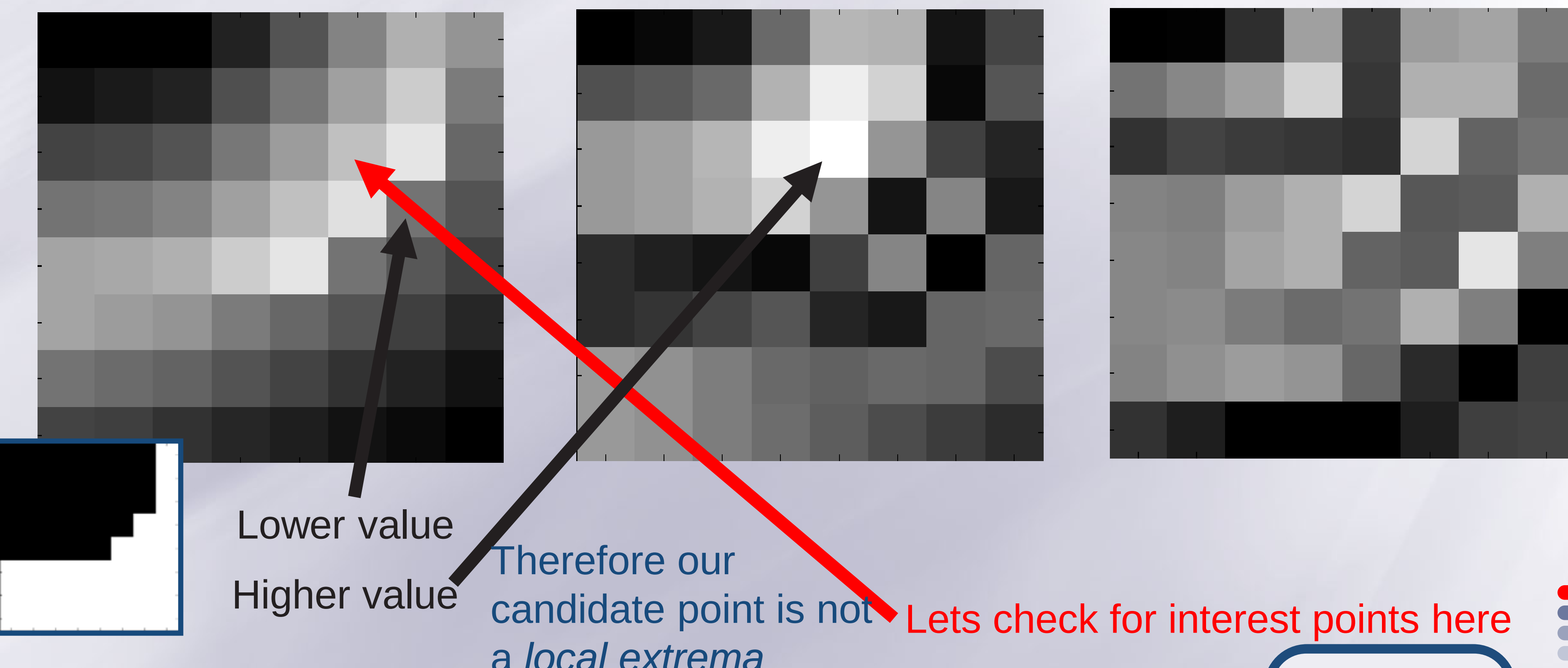
Feature point detection – scalespace

- **Local extrema can be found by checking all pixels in the neighbourhood of each candidate pixel**



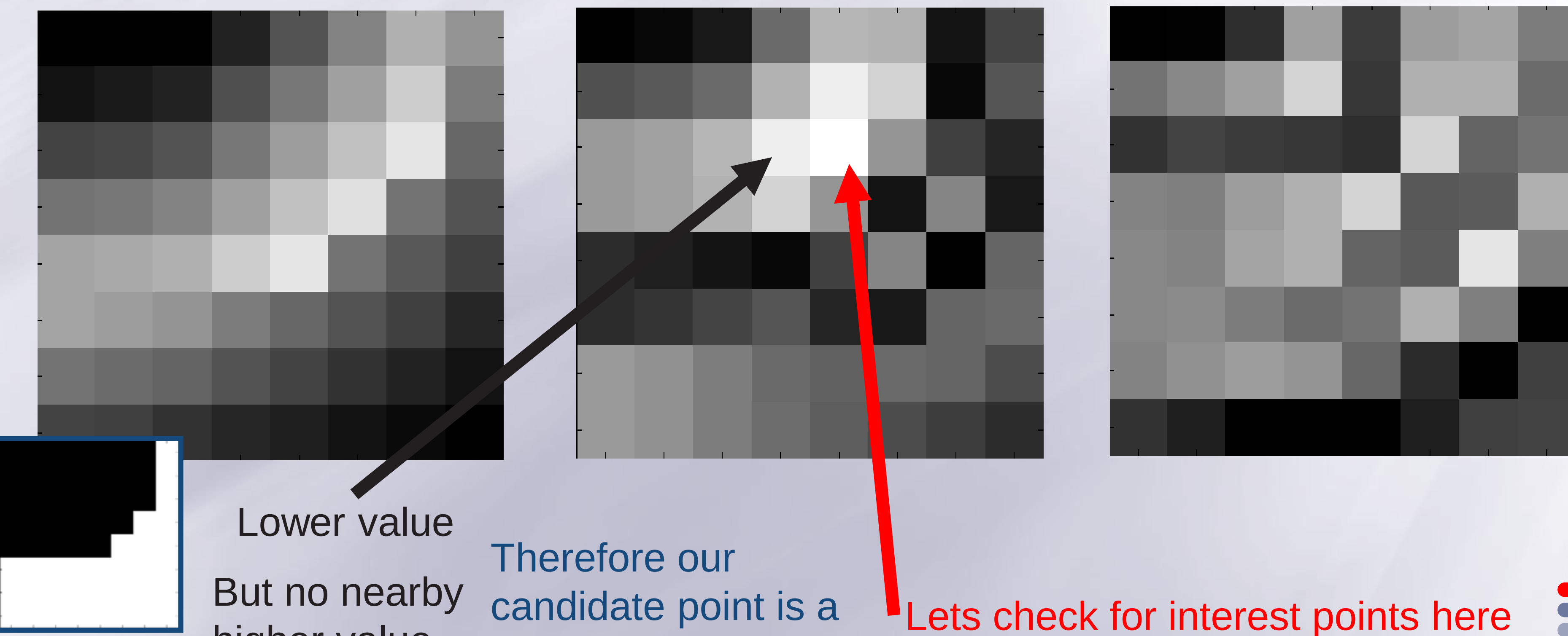
Feature point detection – scalespace

- **Local extrema can be found by checking all pixels in the neighbourhood of each candidate pixel**



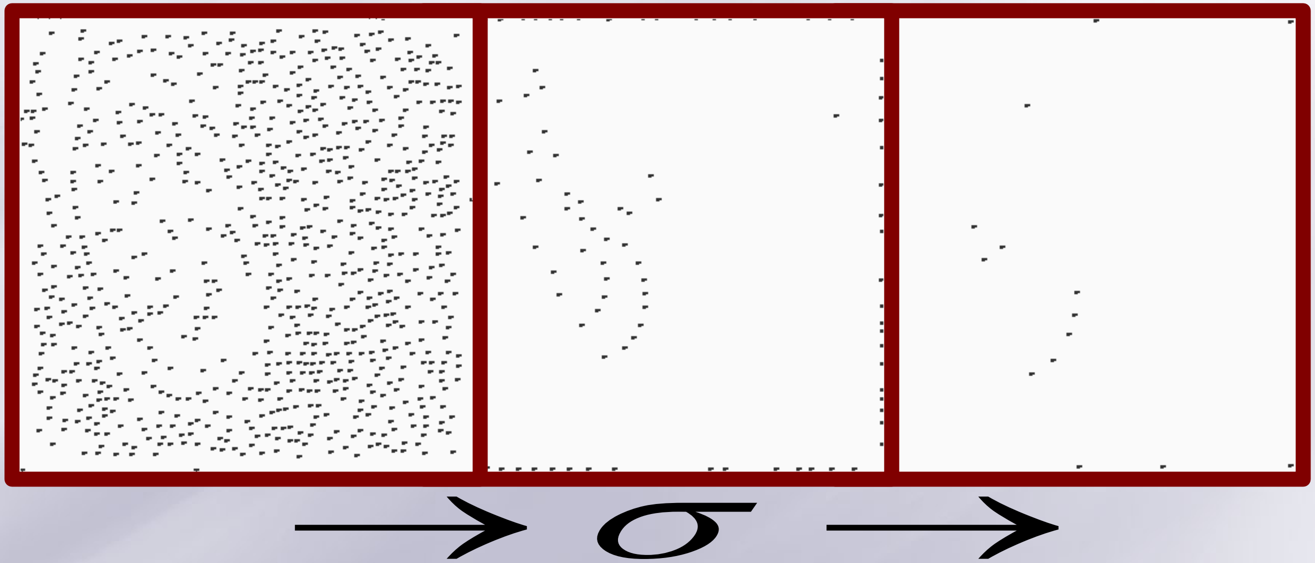
Feature point detection – scalespace

- **Local extrema can be found by checking all pixels in the neighbourhood of each candidate pixel**



Feature point detection – scalespace

- **Local extrema can be found by checking all pixels in the neighbourhood of each candidate pixel**



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

Local Extrema from Difference of Gaussian Scale Space Function

Feature point extraction

- Once a feature-point has been detected, but before we attempt to match it we may decide to process it (this is dubbed feature point **extraction**) to enable more efficient and more general matching. For example:
  - Extract a colour histogram from the image in the neighbourhood of the feature-point
  - Extract a histogram of local gradients from the image in the neighbourhood of the feature-point
  - Rotationally pre-align the image in the neighbourhood of the feature-point according to some criteria
  - And so forth

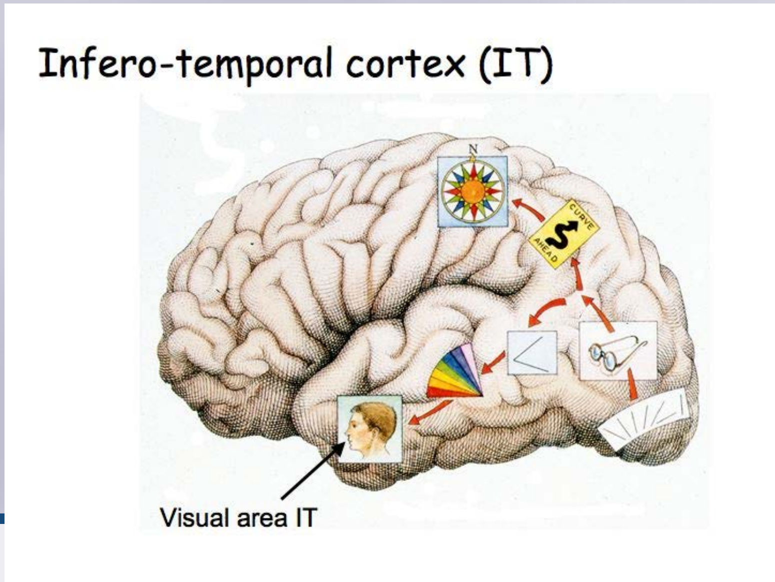
"Local Patch"

SIFT Method

- Scale Invariant Feature Transform (SIFT)
- Staged filtering approach
  - Identifies stable points (image “keys”)
- Computation time less than 2 secs

SIFT Method (2)

- **Local features:**
  - **Invariant to image translation, scaling, rotation**
  - **Partially invariant to illumination changes and 3D projection (up to 20° of rotation)**
  - **Minimally affected by noise**
  - **Similar properties with neurons in Inferior Temporal cortex used for object recognition in primate vision**



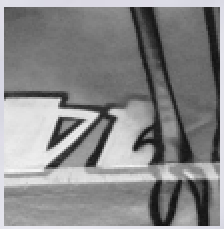
Step 1: feature extraction

Scale-invariant image regions + SIFT

- **Robust description of the extracted image regions**

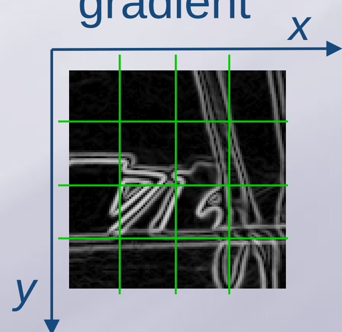
3D histogram

image patch

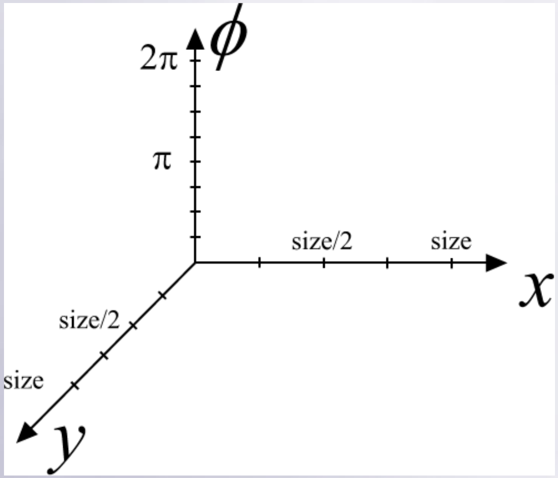


→

gradient



→



– 8 orientations of the gradient
– 4x4 spatial grid

First stage

- Input: original image (512 x 512 pixel)
- Goal: key localization and image description
- Output: SIFT keys
  - Feature vector describing the local image region sampled relative to its scale-space coordinate frame

First stage (2)

Description:

- Represents blurred image gradient locations in multiple orientations planes and at multiple scales
- Approach based on a model of cells in the celebral cortex of mammalian vision
- Less than 1 sec of computation time

Build a pyramid of images

- Images are difference-of-Gaussian (DOG) functions
- Resampling between each level

Key localization

- **Algorithm:**
  - **Expand original image by a factor of 2 using bilinear interpolation**
  - **For each pyramid level:**
    - **Smooth input image through a convolution with the 1D Gaussian function (horizontal direction):**

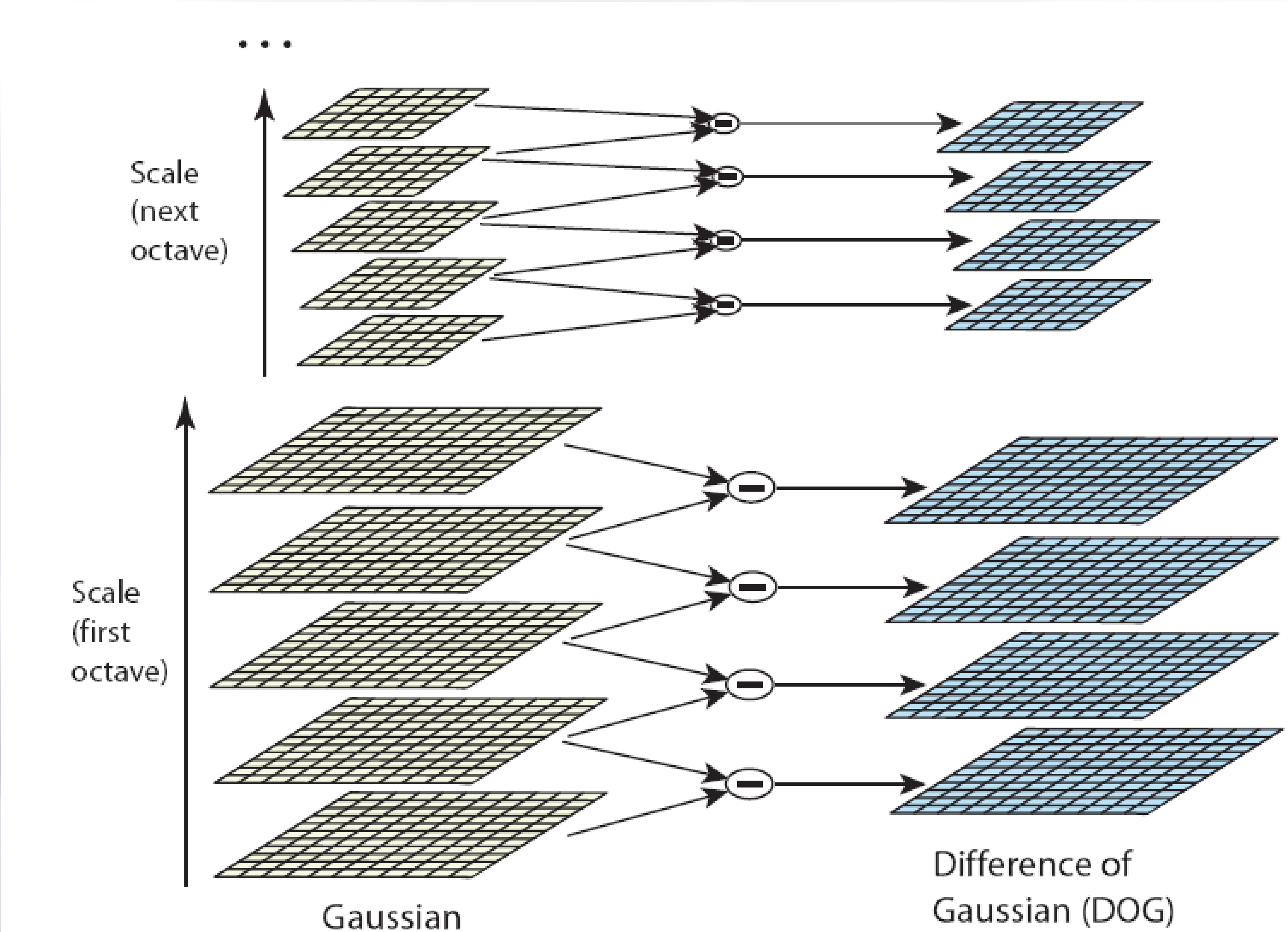
$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

with $\sigma = \sqrt{2}$ obtaining Image A

Key localization (2)

- Smooth Image A through a further convolution with the 1D Gaussian function (vertical direction) obtaining Image B
- The DOG image of this level is B-A
- Resample Image B using bilinear interpolation with pixel spacing 1.5 in each direction and use the result as Input Image of the new pyramid level
  - Each new sample is a constant linear combination of 4 adjacent pixels

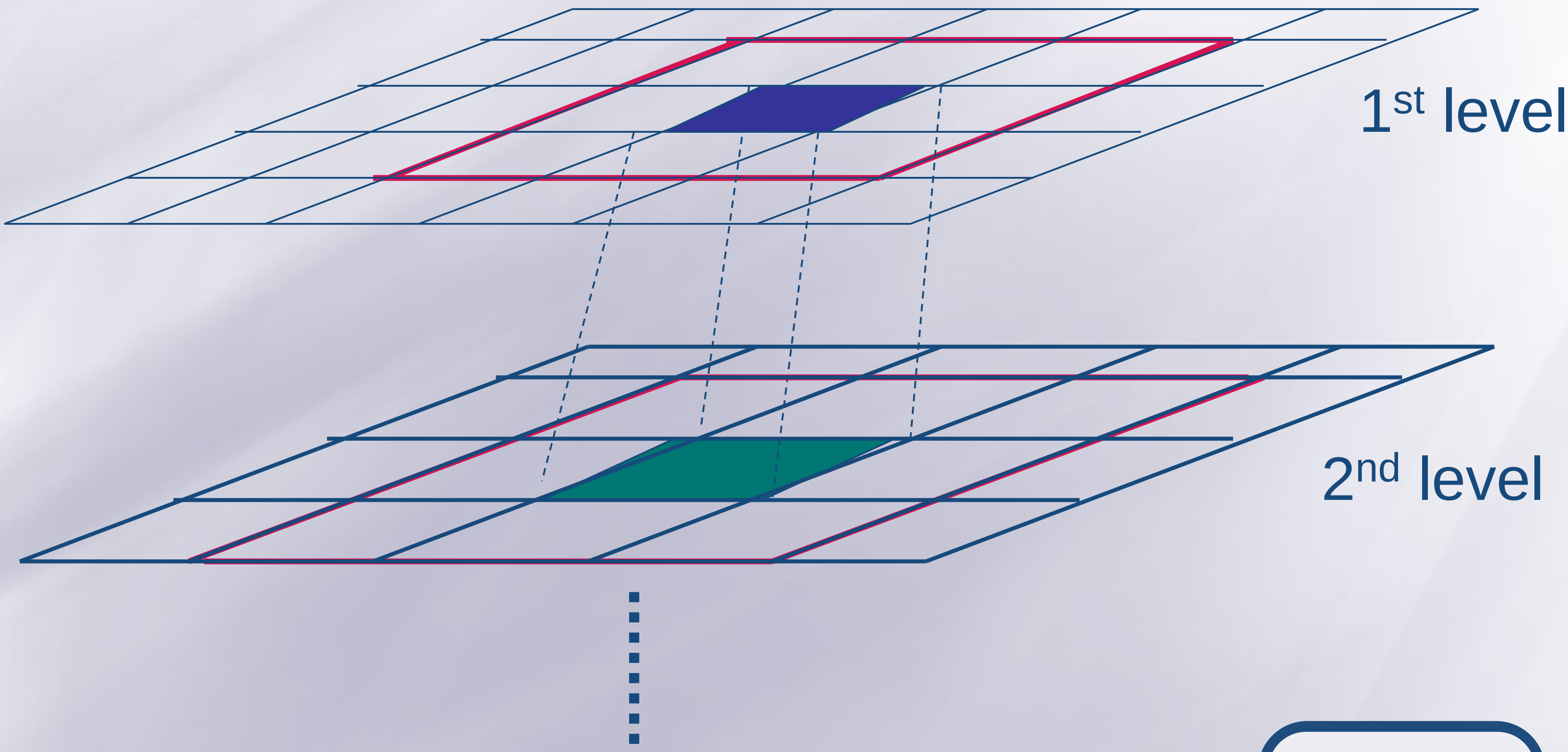
Key localization (2)



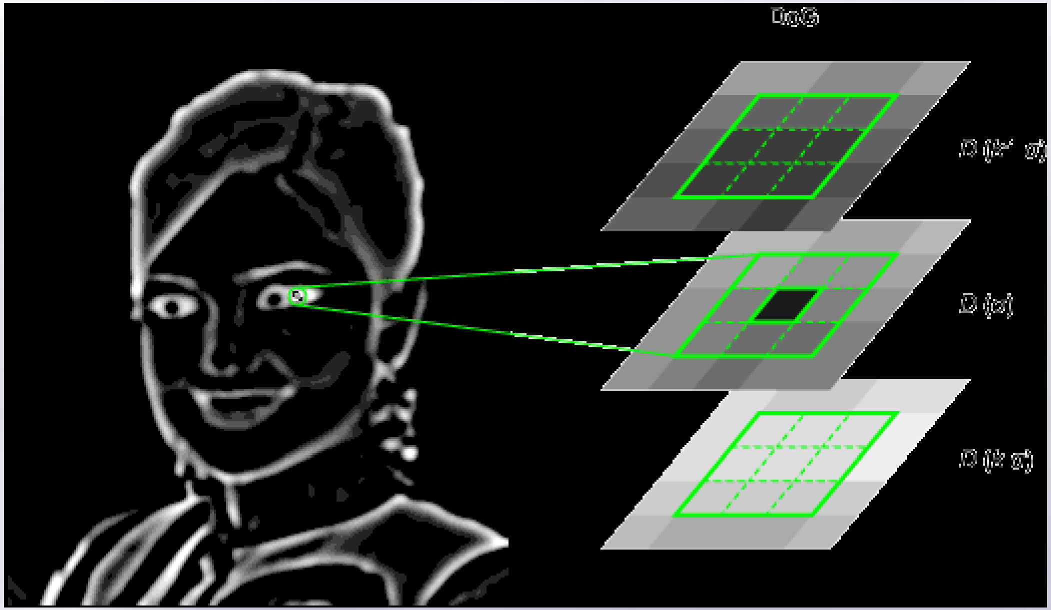
Key localization (2)

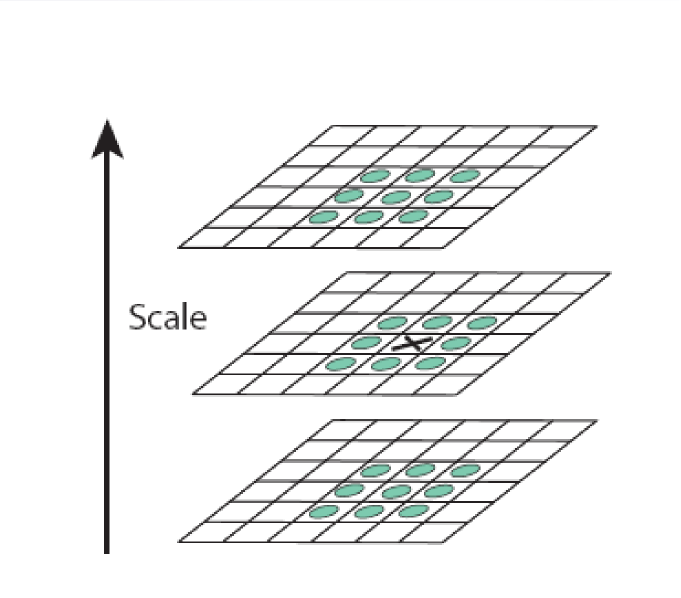
Key localization (3)

- **Find maxima and minima of the DOG images:**



Key localization (3)





Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales (marked with circles).

Key orientation

- **Extract image gradients and orientation at each pyramid level. For each pixel Aij compute**

Image Gradient Magnitude $$M_{ij} = \sqrt{(A_{ij} - A_{i+1,j})^2 + (A_{ij} - A_{i,j+1})^2}$$

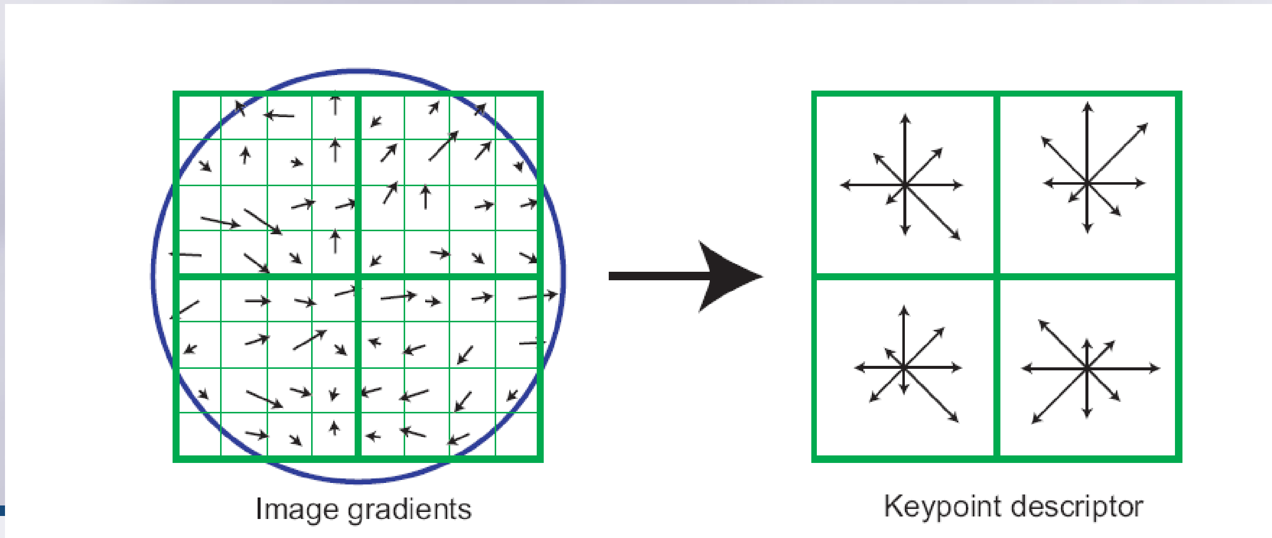
Image Gradient Orientation $$R_{ij} = \arctan 2(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij})$$

$$\forall (A_{ij}) \in \mathrm{neighb}(A_{00})$$

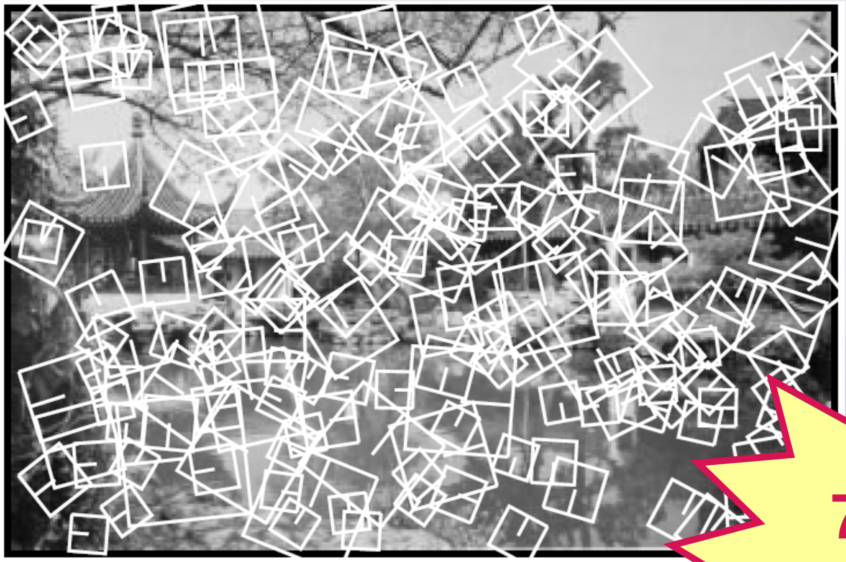
- **M_{ij} thresholded at a value of 0.1 times the maximum possible gradient value**
  - **Provides robustness to illumination**

Key orientation (2)

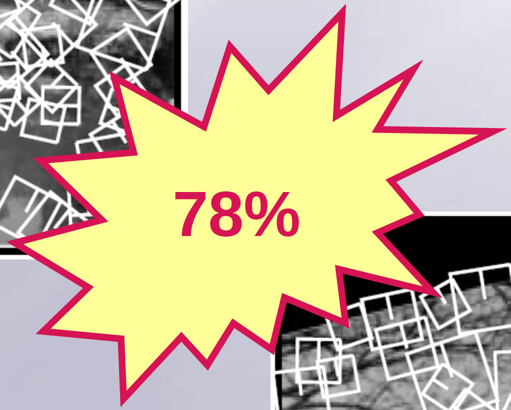
- **Create an orientation histogram using a circular Gaussian-weighted window with σ=3 times the current smoothing scale**
  - The weights are multiplied by Mij
  - The histogram is smoothed prior to peak selection
  - The orientation is determined by the peak in the histogram

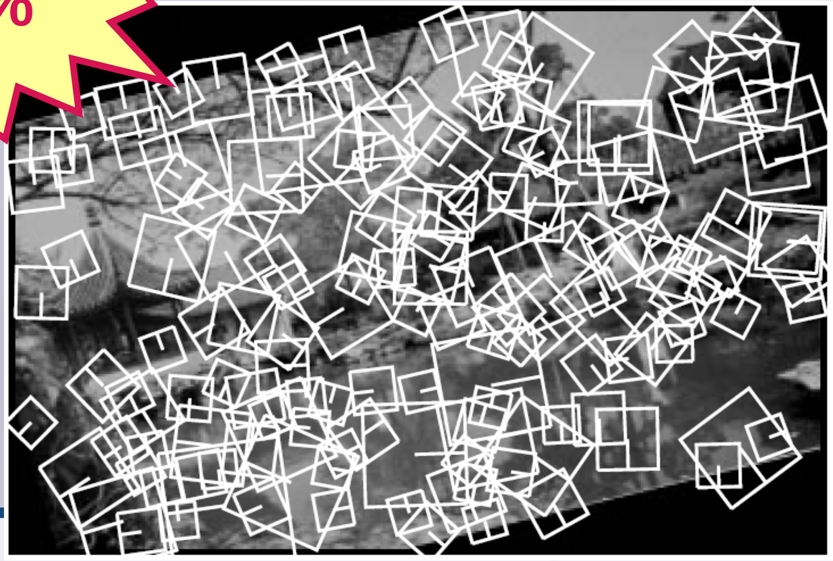


Experimental results



Original image





Keys on image after rotation (15°), scaling (90%), horizontal streching (110%), change of brightness (-10%) and contrast (90%), and addition of pixel noise

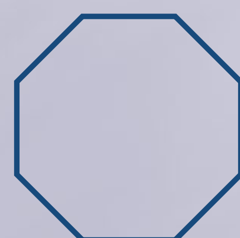
Experimental results

Image transformation	Location and scale match	Orientation match
Decrease constrast by 1.2	89.0 %	86.6 %
Decrease intensity by 0.2	88.5 %	85.9 %
Rotate by 20°	85.4 %	81.0 %
Scale by 0.7	85.1 %	80.3 %
Stretch by 1.2	83.5 %	76.1 %
Stretch by 1.5	77.7 %	65.0 %
Add 10% pixel noise	90.3 %	88.4 %
All previous	78.6 %	71.8 %

20 different images, around 15,000 keys

Image description

- **Approach suggested by the response properties of complex neurons in the visual cortex**
  - **A feature position is allowed to vary over a small region, while orientation and spatial frequency are maintained**
- **Image descripted through 8 orientation planes**
  - **Keys inserted according to their orientations**

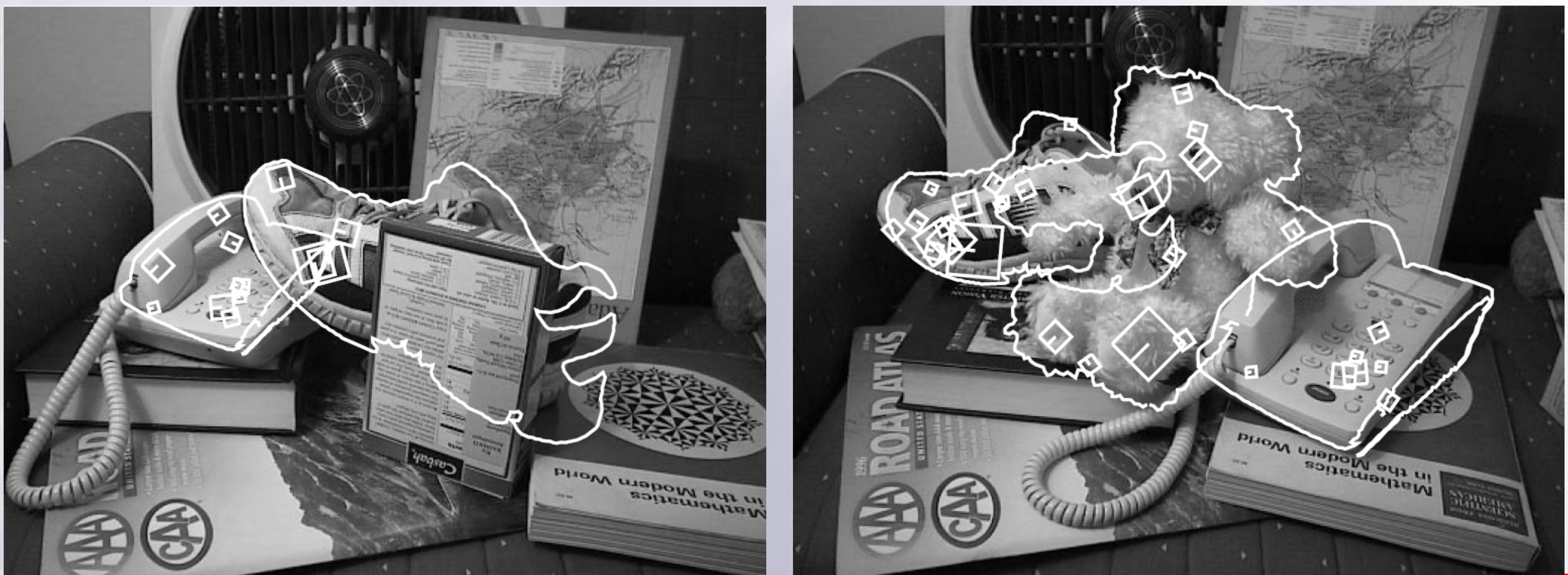


Second stage

- Goal: identify candidate object matches
  - The best candidate match is the nearest neighbour (i.e., minimum Euclidean distance between decriptor vectors)
  - The exact solution for high dimensional vectors is known to have high complexity

Perspective projection

Partial occlusion



Computation time: 1.5 secs on Sun Sparc 10
(0.9 secs first stage)

Connections to human vision

- **Performance of human vision is obviously far superior than current computer vision...**

- **The brain uses a highly computational-intensive parallel process instead of a staged filtering approach**

Connections to human vision

- However... the results are much the same
- Recent research in neuroscience showed that the neurons of Inferior Temporal cortex
  - Recognize shape features
  - The complexity of the features is roughly the same as for SIFT
  - They also recognize color and texture properties in addition to shape
- Further research:
  - 3D structure of objects
  - Additional feature types for color and texture

References

- David G. Lowe, "Object recognition from local scale-invariant features" International Conference on Computer Vision, Corfu, Greece (September 1999), pp. 1150-1157
- Stephen Se, David G. Lowe and Jim Little, "Vision-based mobile robot localization and mapping using scale-invariant features" Proceedings of IEEE International Conference on Robotics and Automation, Seoul, Korea (May 2001), pp. 2051-58
- Iryna Gordon and David G. Lowe, "Scene modelling, recognition and tracking with invariant image features" International Symposium on Mixed and Augmented Reality (ISMAR), Arlington, VA (Nov. 2004), pp. 110-119