Spring 2015

CSE 617: Digital Image Processing

Local Feature Extraction

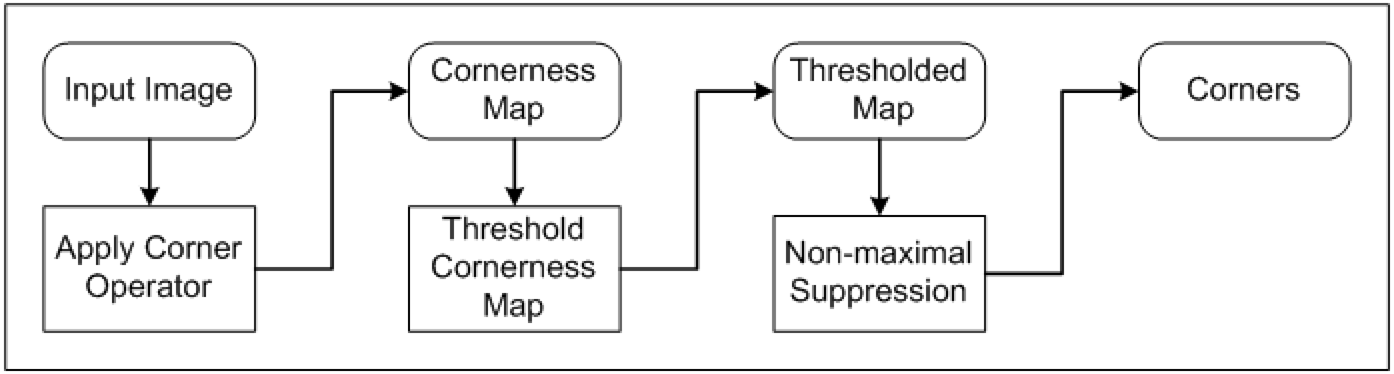
Seif Eldawlatly

Corner Detectors

- It is always useful to find pairs of corresponding points in two similar images
- This could be used in the analysis of moving images
- This could be done by comparing all possible pairs of pixels in the two images. However, it is computationally intensive
- This process might be simplified by comparing interest points only such as corners
- A corner can be defined as a pixel in its small neighborhood where two dominant and different edges meet

Moravec Operator

- General structure of corner detectors



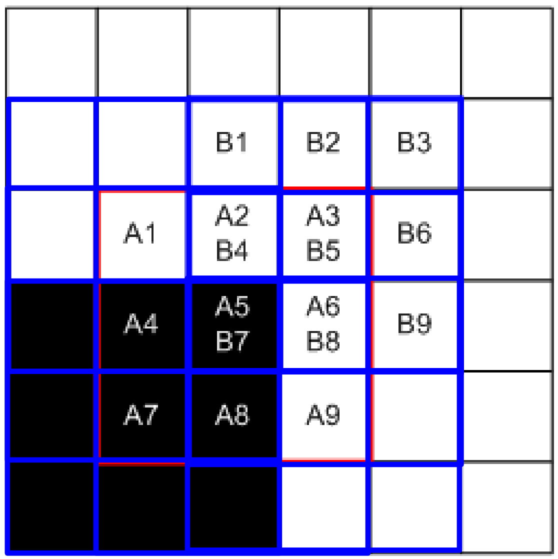
http://kiwi.cs.dal.ca/~dparks/CornerDetection/algorithms.htm

- Moravec operator estimates the *cornerness* of a point by computing a measure of intensity variation in a given neighborhood w with a shift of u and v

$$V(x,y)_{w(u,v)} = \sum_{i=-1}^{1}\sum_{j=-1}^{1}\left(f(x+i,y+j)-f(x+u+i,y+v+j)\right)^2$$

Moravec Operator

- It measures the cornerness by shifting a window around the considered pixel by 1 pixel in each of the 8 principal directions and calculating the corresponding V



$$V = \sum_{i=1}^{9} (A_i - B_i)^2 = 3 * 255^2$$

http://kiwi.cs.dal.ca/~dparks/CornerDetection/algorithms.htm

$$V = \begin{array}{c} \\ \left[\begin{array}{c} 3*255^2 \\ 2*255^2 \\ 3*255^2 \\ 2*255^2 \\ 5*255^2 \\ 2*255^2 \\ 3*255^2 \\ 2*255^2 \end{array}\right] \end{array} \quad \begin{array}{c} \textbf{Window } \boldsymbol{B} \textbf{ Centered at} \\ \text{A3} \\ \text{A2} \\ \text{A1} \\ \text{A4} \\ \text{A7} \\ \text{A8} \\ \text{A9} \\ \text{A6} \end{array}$$

- The cornerness at a pixel (x, y)

$$C(x, y) = \min_{u,v} V(x, y)_{w(u,v)}$$

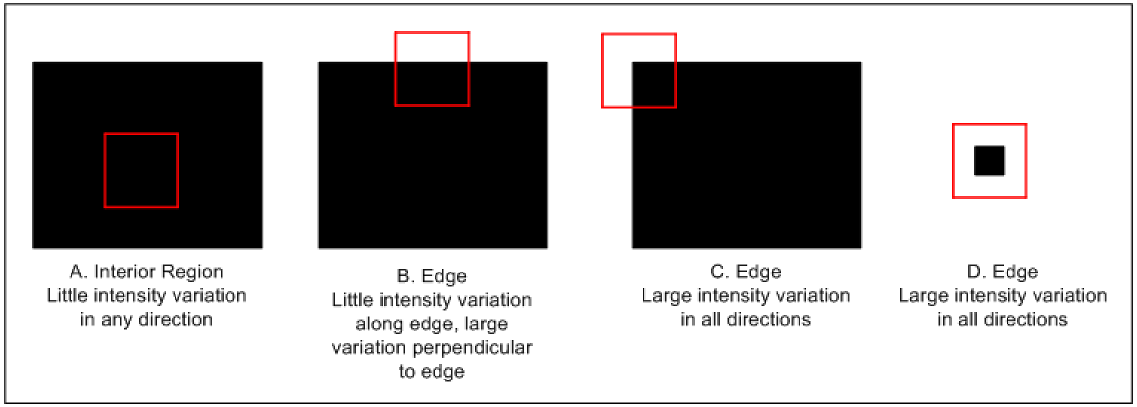
Moravec Operator

- Example (using a 3 x 3 window)

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	0	0	0	0	0	0	0	0	0	1	1	1	X	X
X	X	0	0	0	0	0	1	1	0	0	1	2	1	X	X
X	X	0	0	0	0	0	2	1	0	0	1	1	1	X	X
X	X	0	0	0	0	0	0	0	0	0	0	0	0	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

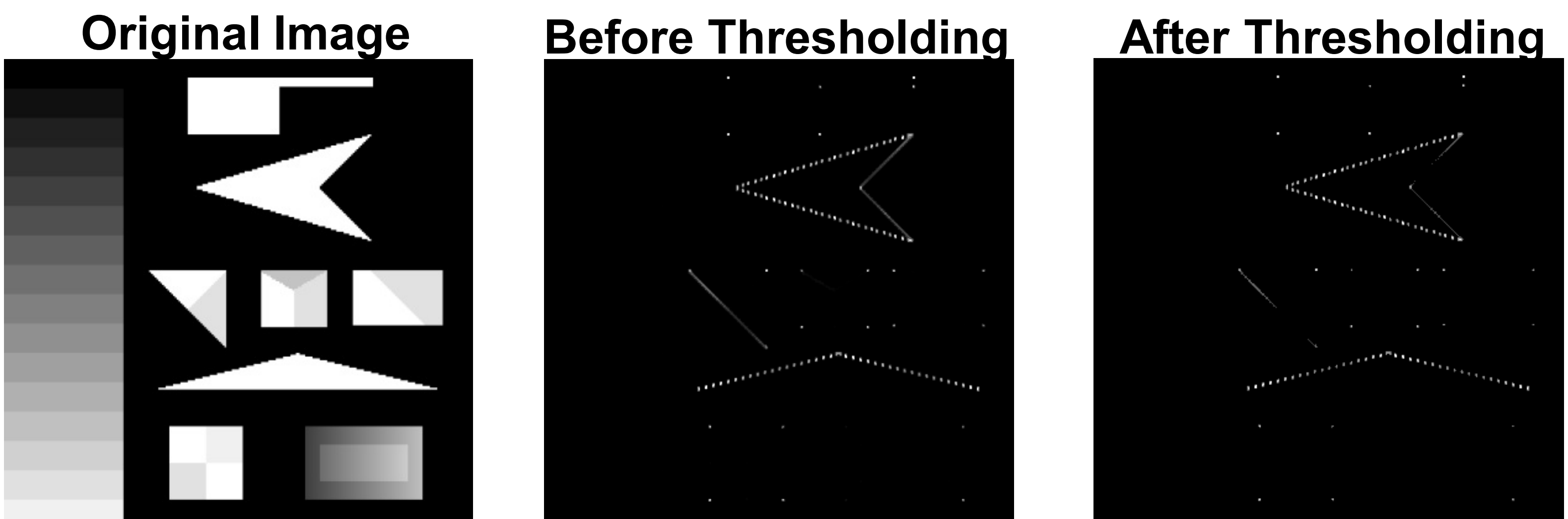
Moravec Operator

- Why would Moravec operator work?



http://kiwi.cs.dal.ca/~dparks/CornerDetection/algorithms.htm

- By setting all points with cornerness below a threshold T to 0, corner points can be detected

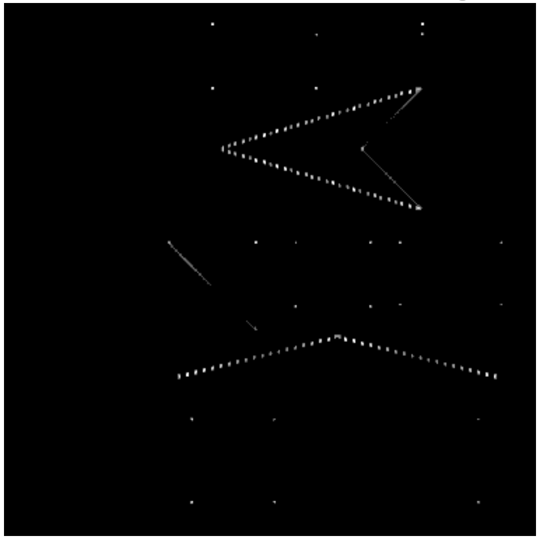


Moravec Operator

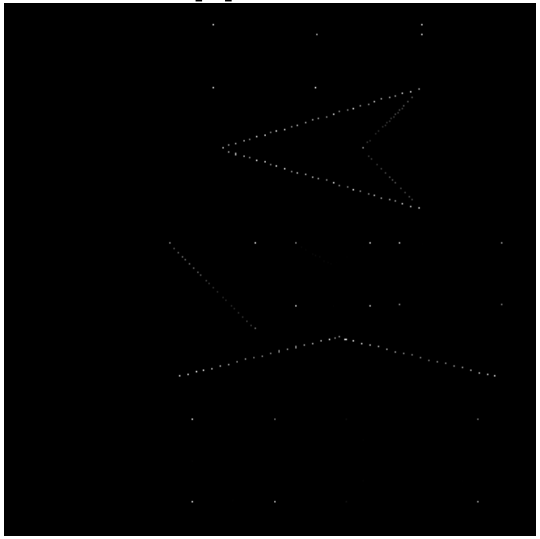
- Finally, non-maximal suppression can be applied

Original Image

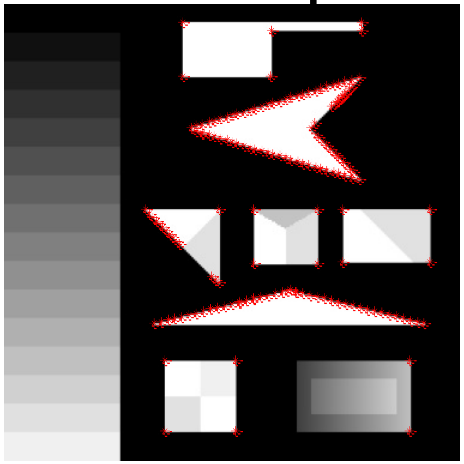
After Thresholding



After Non-maximal Suppression



Final Output



Cornerness of any pixel is set to 0 if it is not larger than the cornerness of all 8-neighbors

Harris Corner Detector

- Improved upon Moravec operator

$$S_W(\Delta x, \Delta y) = \sum_{x_i \in W} \sum_{y_i \in W} \left(f(x_i, y_i) - f(x_i - \Delta x, y_i - \Delta y) \right)^2$$

- Similar to the goal of Moravec operator, we try to find the minimum value of S_W

- This could be found analytically if the shifted image patch is approximated by the first-order Taylor expansion

$$f(x_i - \Delta x, y_i - \Delta y) \approx f(x_i, y_i) + \left[\frac{\partial f(x_i, y_i)}{\partial x}, \frac{\partial f(x_i, y_i)}{\partial y} \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

- Substituting in the expression for S_W

$$S_W(\Delta x, \Delta y) = \sum_{x_i \in W} \sum_{y_i \in W} \left(f(x_i, y_i) - f(x_i, y_i) - \left[\frac{\partial f(x_i, y_i)}{\partial x}, \frac{\partial f(x_i, y_i)}{\partial y} \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2$$

Harris Corner Detector

$$
\begin{aligned}
S_W(\Delta x, \Delta y) &= \sum_{x_i \in W} \sum_{y_i \in W} \left(f(x_i, y_i) - f(x_i, y_i) - \left[\frac{\partial f(x_i, y_i)}{\partial x}, \frac{\partial f(x_i, y_i)}{\partial y} \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 \\
&= \sum_{x_i \in W} \sum_{y_i \in W} \left(- \left[\frac{\partial f(x_i, y_i)}{\partial x}, \frac{\partial f(x_i, y_i)}{\partial y} \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 \\
&= \sum_{x_i \in W} \sum_{y_i \in W} \left(\left[\frac{\partial f(x_i, y_i)}{\partial x}, \frac{\partial f(x_i, y_i)}{\partial y} \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 \\
&= \sum_{x_i \in W} \sum_{y_i \in W} [\Delta x, \Delta y] \left(\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \right) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
&= [\Delta x, \Delta y] \left(\sum_{x_i \in W} \sum_{y_i \in W} \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \right) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
&= [\Delta x, \Delta y] \, A_W(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} ,
\end{aligned}
$$

Harris Corner Detector

- The goal now is to minimize

$$S_W(\Delta x, \Delta y) = [\Delta x, \Delta y]\, A_W(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

- This is equivalent to finding the eigenvector of A_w corresponding to the minimum eigenvalue
- What are the eigenvectors and eigenvalues?

  For any matrix L, the eigenvectors and eigenvalues are defined as

$$Lf = \lambda f \rightarrow \quad f^T L f = \lambda$$

f is an eigenvector of L

λ is the eigenvalue corresponding to f

- For A_w, the eigenvector is $\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$ and the eigenvalue is the corresponding S_w

Harris Corner Detector

- Therefore, finding the eigenvalues of A_w would be sufficient to solve the minimization problem where

$$A(x,y) = \begin{bmatrix} \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial^2 f(x_i,y_i)}{\partial x^2} & \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial f(x_i,y_i)}{\partial x} \frac{\partial f(x_i,y_i)}{\partial y} \\ \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial f(x_i,y_i)}{\partial x} \frac{\partial f(x_i,y_i)}{\partial y} & \sum_{x_i \in W} \sum_{y_i \in W} \frac{\partial^2 f(x_i,y_i)}{\partial y^2} \end{bmatrix}$$

- Instead of computing the eigenvalues, Harris suggested using the following approximation

$$R(A) = \det(A) - \kappa\, trace^2(A)$$

where $\det(A)$ is the determinant of the local structure matrix A

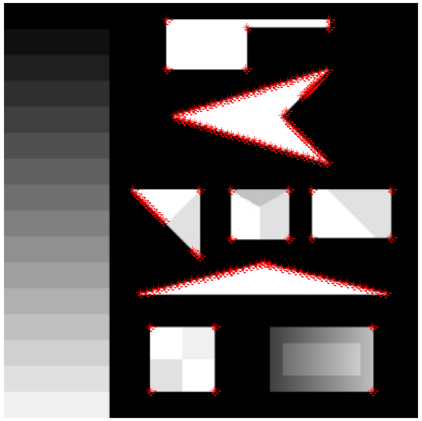
$trace(A)$ is the trace of matrix A (sum of elements on the diagonal)

κ is a tunable parameter

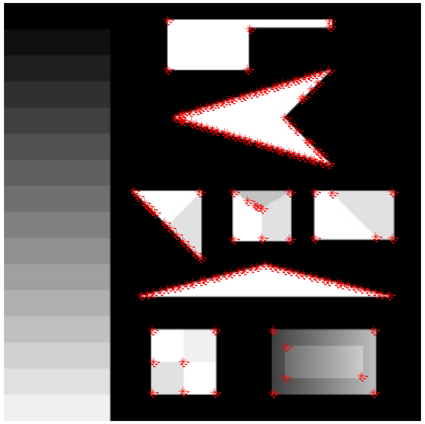
Harris Corner Detector

- Algorithm
  1. Filter the image with a Gaussian filter
  2. Estimate intensity gradient in 2 perpendicular directions for each pixel
     $\frac{\partial f(x,y)}{\partial x}$ and $\frac{\partial f(x,y)}{\partial y}$
  3. For each pixel and a given neighborhood window
     - Calculate the local structure matrix A
     - Evaluate the response function $R(A)$
  4. Set all pixels with response less than a threshold T to 0 and perform non-maximal suppression

Moravec Operator



Harris Detector



Scale Invariant Feature Transform (SIFT)

- References:

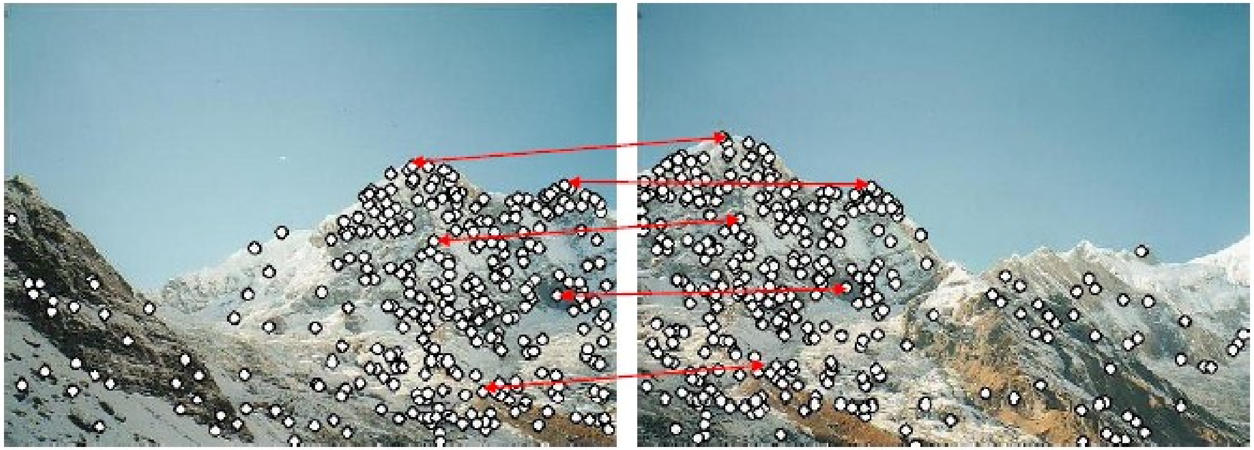
    D.G. Lowe, “Distinctive Image features from scale-invariant keypoints,” International Journal on Computer Vision 60 (2), 91–110, 2004

Local Feature Extraction

- Image matching is a fundamental aspect of many problems in computer vision and image processing
- Applications include object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking
- Example: Consider the problem of stitching multiple images to create a panoramic view

Local Feature Extraction

- Stitching two images can be easily achieved once pairs of matched points across the two images are found



- By mapping the correspondence, a panorama can be easily created

Local Feature Extraction

- This process can be achieved through two steps:

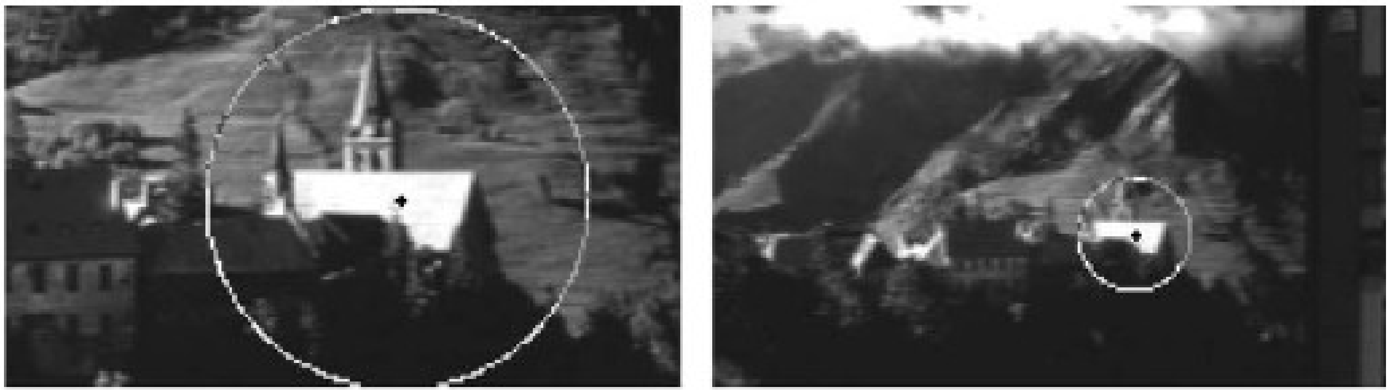
  - Step 1: Identify keypoints in both images

  - Step 2: Match keypoints to identify their correspondence

- Step 1 is needed to reduce the search space in Step 2

Local Feature Extraction

- Another application that can make use of keypoint detection is recognizing similar objects in different images



- In this application, objects (keypoints) should be identified independent of their scales

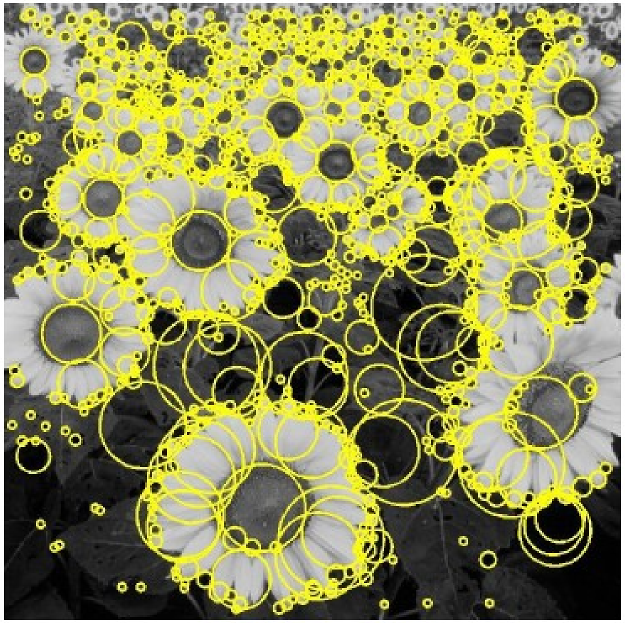
- Moravec operator or Harris corner detectors can be used for this application

Scale Invariant Detection

- An important approach that could be used for both images stitching or different scales recognition is scale invariant detection

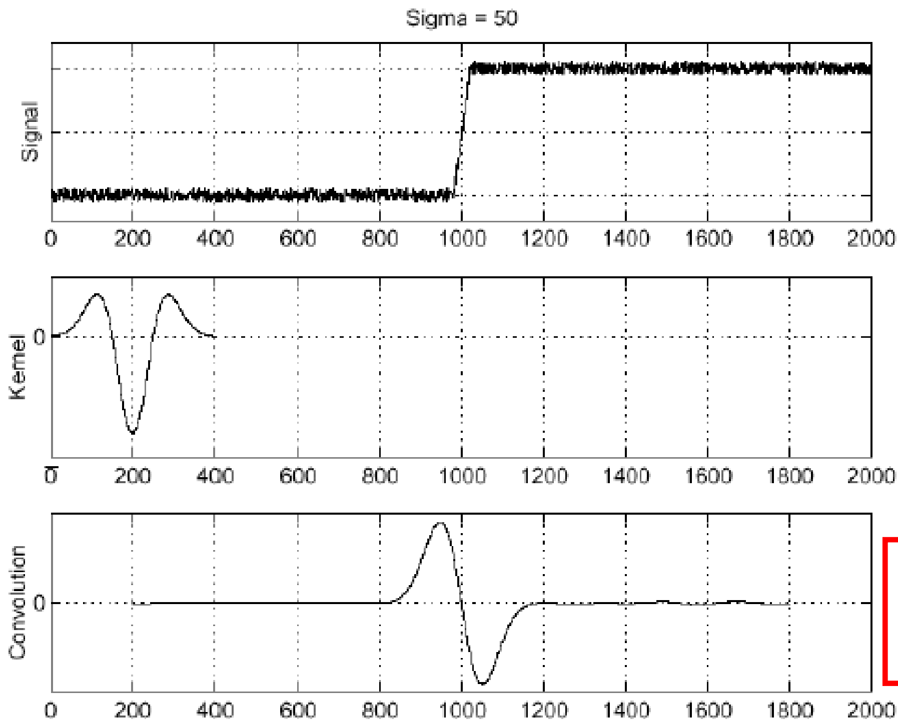
- The goal of such method is to identify a keypoint region as the region whose brightness is different from the surrounding

- A very common method is to identify Blobs in which the circle of smallest radius that encloses a keypoint region is identified



Scale Invariant Detection

- Identifying blobs is based on using the Laplacian of Gaussian (LoG) method we previously encountered in edge detection



f

Edge

$\frac{d^2}{dx^2} g$

Second derivative of Gaussian (Laplacian)

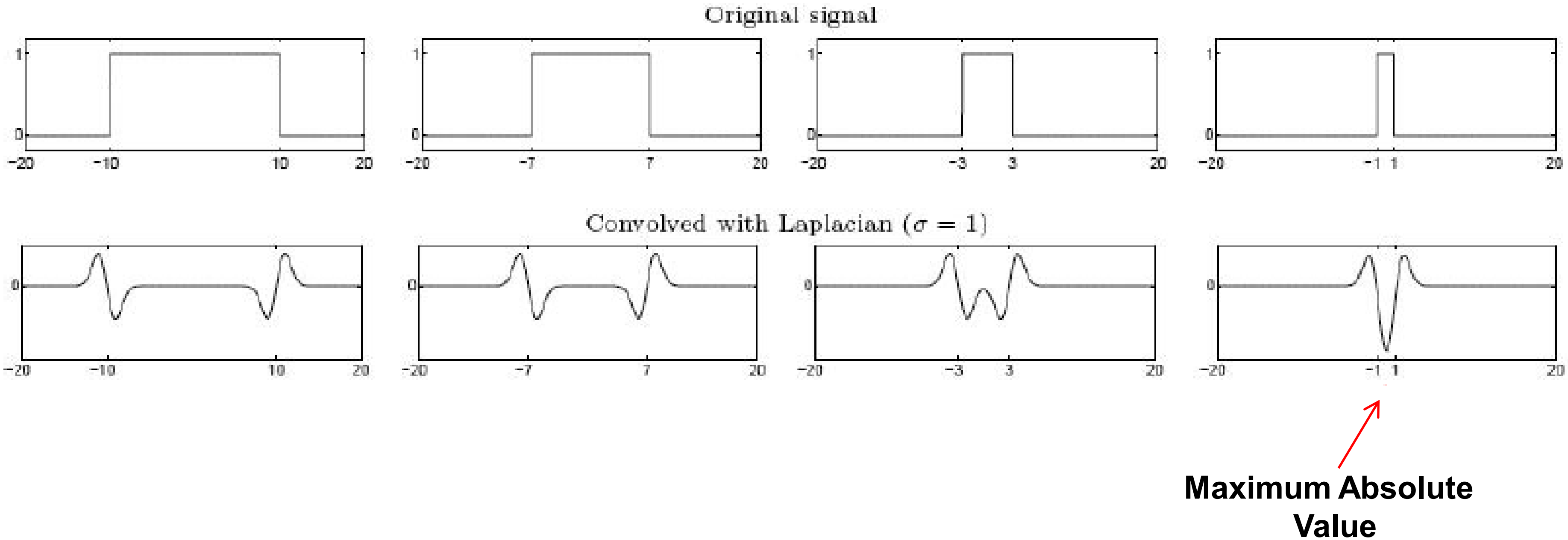
$f * \frac{d^2}{dx^2} g$

Edge = zero crossing of second derivative

Source: S. Seitz

Scale Invariant Detection

- Applying the LoG mask to a blob gives the following response as the size of the blob increases



- If the scale of the LoG is matched to the blob size, the response of the LoG will be maximum at the center of the blob

Scale Invariant Detection

- Based on the previous observation, identifying a blob can be done by searching for the maximum absolute LoG response in:

  - Space: Searching in local neighborhood
  - Scale: Searching across different scales

Scale Invariant Feature Transform (SIFT)

- The SIFT algorithm can identify blobs and match them across different images

- SIFT Algorithm Steps:

  - Scale-space extrema detection

  - Orientation assignment

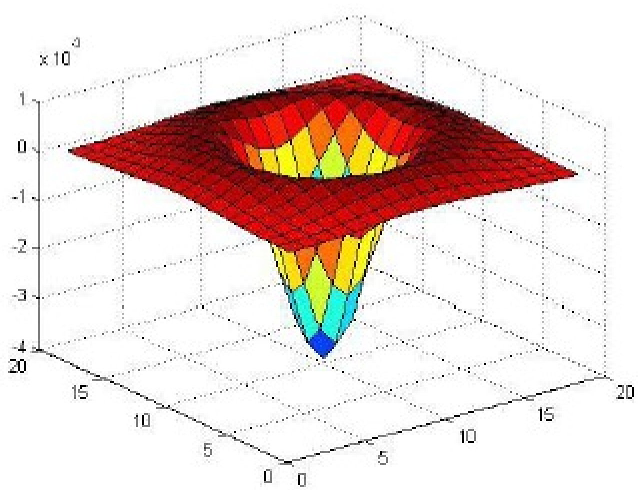
  - Generation of keypoint descriptors

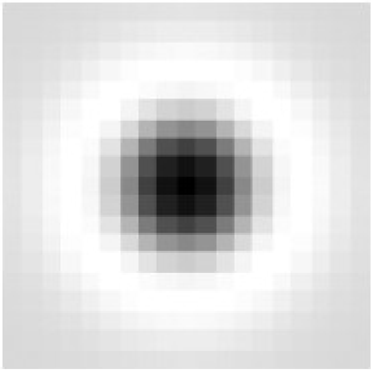
SIFT: Scale-space Extrema Detection

- Keypoints in SIFT correspond to local extrema after applying the LoG mask across different scales

- The LoG mask represents a circularly symmetric operator for blob detection

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





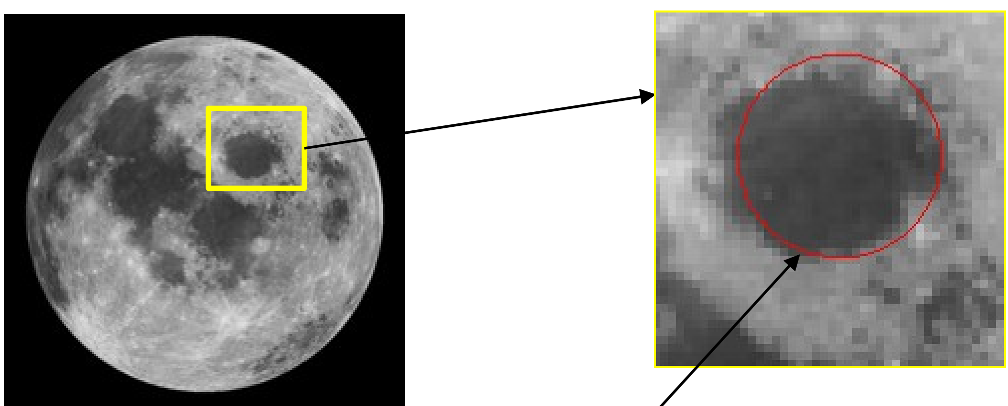
SIFT: Scale-space Extrema Detection

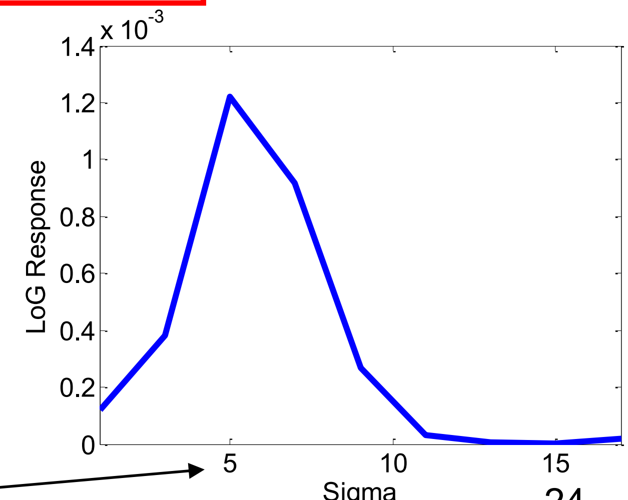
- To normalize the values, the LoG is multiplied by σ^2

$$\sigma^2 LoG(x,y) = \frac{-1}{\pi\sigma^2}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- For a given keypoint, the corresponding best scale σ_{best} is determined as the one that maximizes the squared LoG response

- The corresponding radius is approximately $\sqrt{2}\sigma_{best}$





Identified Blob

Scale of Maximum Squared LoG Response

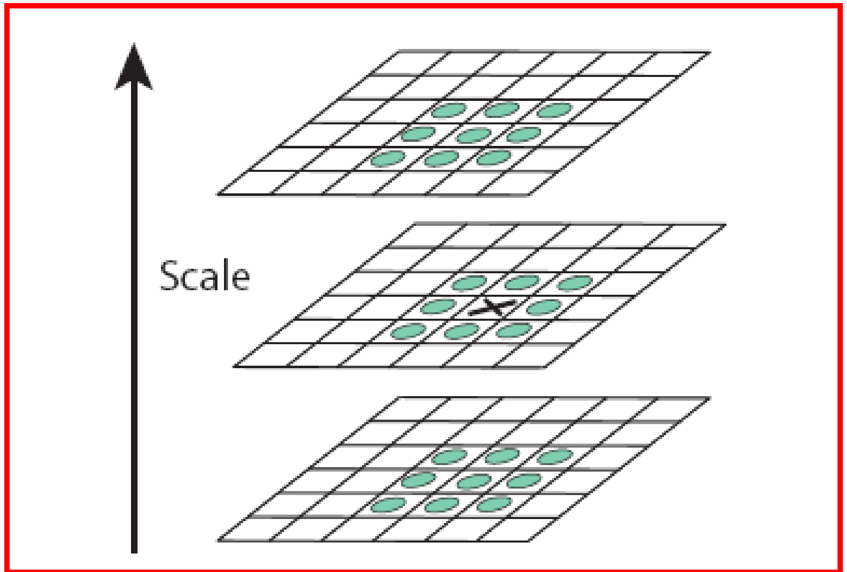
SIFT: Scale-space Extrema Detection

- Steps of the Scale-space Extrema Detection:

  1- Convolve the image with the normalized LoG for different values of σ

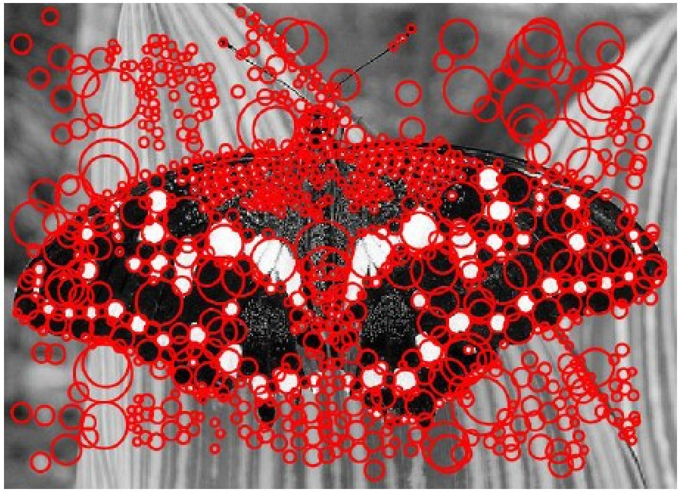
  2- Find the maxima of the squared LoG response in scale-space

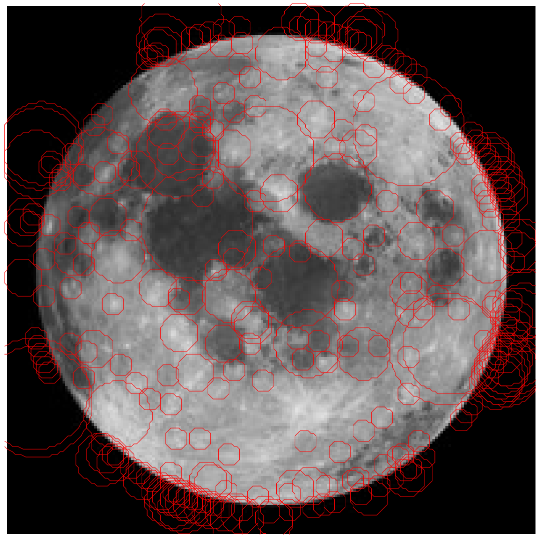
- Maxima of the squared LoG images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales (marked with circles)



SIFT: Scale-space Extrema Detection

- Examples:

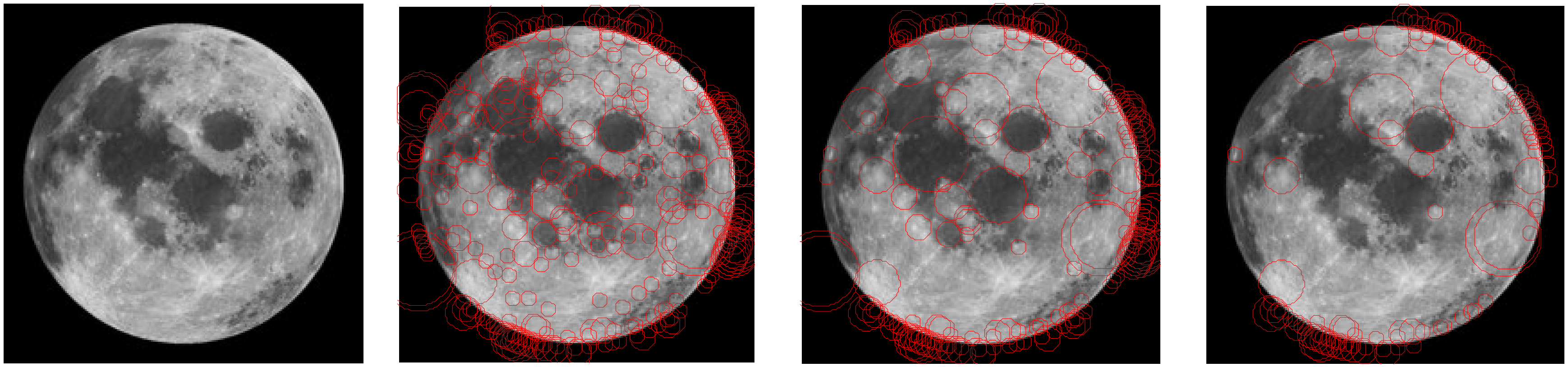




SIFT: Scale-space Extrema Detection

- Obtained extrema less than a predefined threshold t are usually ignored

No Threshold
t = 0   **t = 0.01**   **t = 0.02**



SIFT: Orientation Assignment

- In order to stitch two images, blobs in both images are first identified and then compared

Image 1

Image 2

Blobs of Image 1

Blobs of Image 2

SIFT: Orientation Assignment

- Blobs corresponding to each other across the two images are then identified

- This is done in SIFT by first assigning a consistent orientation to each keypoint based on local image properties

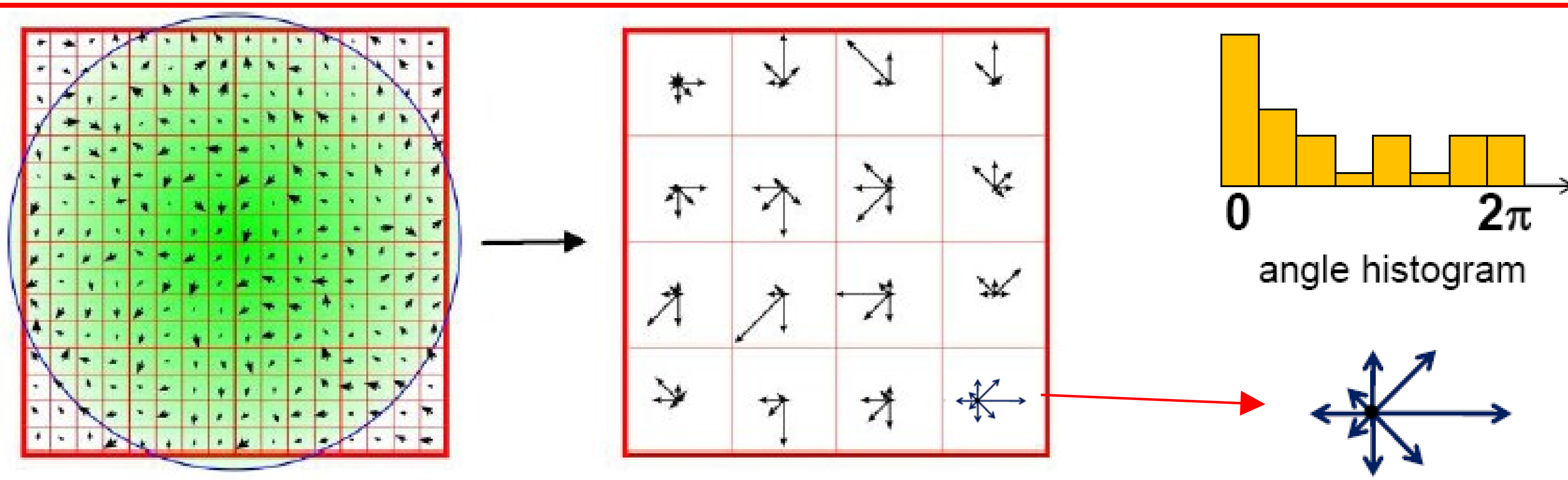
- This is done as follows:
  - For each keypoint, convolve the image with a Gaussian filter with standard deviation of σ_{best}

  - Apply a first-derivative operator (for example Prewitt operator) to find the corresponding gradient image

  - Find the gradient orientation at each of the pixels in the 16 x 16 neighborhood around the keypoint

SIFT: Descriptors Generation

- For each blob, a descriptor is then constructed
- This is done by creating an 8-bin orientation histogram in each 4 x 4 neighborhood of the 16 x 16 neighborhood around the keypoint



- The obtained 16 histograms are concatenated together to form a single descriptor vector of length 4 x 4 x 8 = 128 values
- This descriptor is invariant to scale changes and brightness affine transformations

SIFT: Descriptors Matching

- The final step is to find for each descriptor in one image the corresponding similar descriptor in the other image

- This is done by comparing descriptors across the two images using Euclidean distance

- For each descriptor in one image, the corresponding descriptor in the other image is the one that gives the least Euclidean distance (preferably 0 distance)

Compare the descriptor of this blob with the descriptors of all other blobs in Image 2 and find the one that gives zero (or minimum) distance

From Image 1

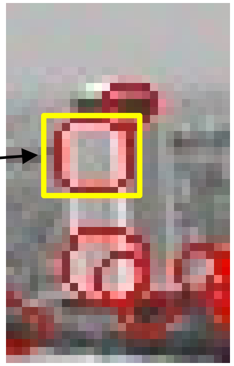


Image 2

SIFT: Descriptors Matching

- Finally, we find the average shift between the corresponding blobs across the two images
- This shift is used to adjust how the two images will be stitched

Image 1

Image 2

Stitched Image