AI61201: Visual Computing With AI/ML

Module 6: Feature Design for Image Analysis

Dr. Somdyuti Paul

Low, Mid and High Level Image Features

- An image feature is a distinctive attribute extracted from an image that can be used to represent, analyze, or understand its content.

- Image features could be broadly categorized into three types:
  - Low-level features: simple attributes directly extracted from raw pixel values that represent the lowest level of abstraction, such as color histograms, edges, corners.
  - Mid-level features: provide a higher level of abstraction of the image such as shape, texture and keypoints.
  - High level features: highly abstract and complex representations of an image that capture semantic and contextual information, such as class labels, bounding boxes, segmentation masks.

- The low, mid and high level features share a hierarchical relationship, i.e. features at higher levels of abstraction can be constructed using those extracted at lower levels of abstraction.

- Feature selection and/or extraction is often a preprocessing step in ML pipelines for image understanding and analysis.

Canny Edge Detector

- The Canny operator (Canny, 1986) is still one of the most widely used edge detection operators in image processing and computer vision tasks.

- Edge detection using the Canny operator involves the following steps:

1. *Noise Reduction* : given an image, Gaussian smoothing is performed to reduce the impact of noise on edge detection.

$$f'(x,y) = f(x,y) * g(x,y), \text{ where } g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2. Gradient Computation: gradients are then computed to identify areas of significant changes in intensity (i.e. edges) using gradient operators such as the Sobel kernels.

$$G_x(x,y) = f'(x,y) * h_x(x,y),\ G_y(x,y) = f'(x,y) * h_y(x,y)$$

$$M(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2},\ \Theta(x,y) = \tan^{-1}\frac{G_y(x,y)}{G_x(x,y)}$$

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

$h_x(x,y)$

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

$h_y(x,y)$

Sobel operators along horizontal and vertical directions

Canny Edge Detector

3. *Non-maximum suppression*: if a pixel is not the local maximum in the direction of the gradient, θ, then it is set to 0. This step ensures that the edges are thin and only the strongest edges are preserved:

$$M'(x,y) = \begin{cases} M(x,y) \text{ if } M(x,y) = m \text{ where } m = \max W_\theta(x,y) \\ 0 \text{ otherwise} \end{cases}$$

, where $W_\theta(x,y)$ is the neighborhood in the gradient direction θ.

4. *Double Thresholding*: all pixels are classified into strong, weak and non-edges based on two thresholds:

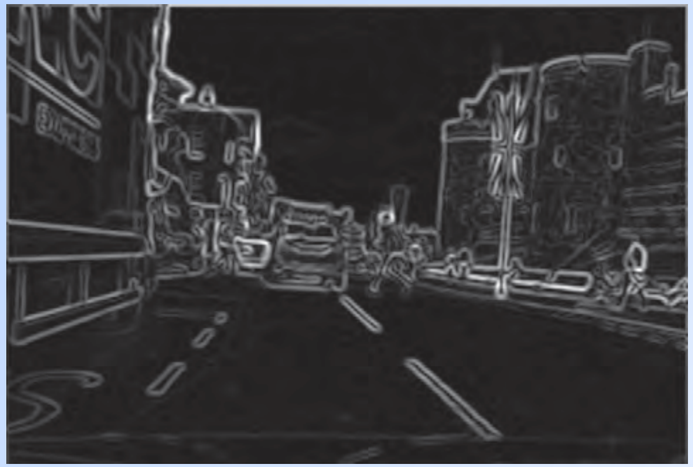
$$E(x,y) = \begin{cases} 1 & \text{if } M'(x,y) \geq \tau_H \text{ (Strong edges)} \\ 2 & \text{if } \tau_L \leq M(x,y) < \tau_H \text{ (Weak edges)} \\ 0 & \text{if } M'(x,y) < \tau_L \text{ (Non-edges)} \end{cases}$$

5. *Edge tracking by Hysteresis*: pixels classified as strong edges are retained in the final edge map, and the weak edges are retained if and only if they are connected to strong edges, by considering the 8-connected neighborhood around the pixel classified as an weak edge.

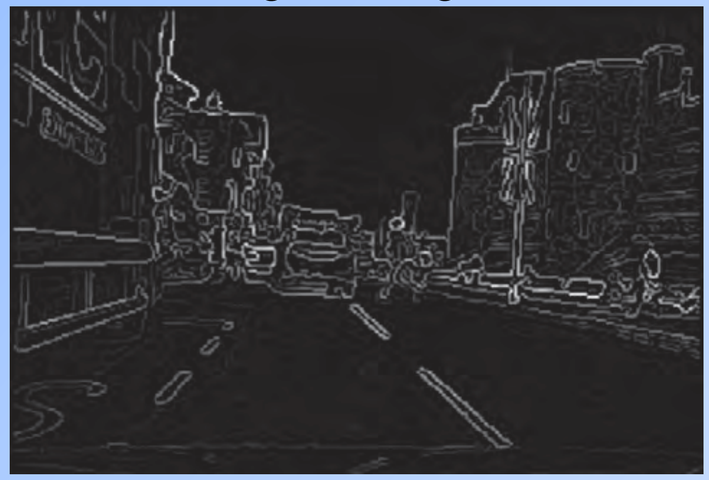
Canny Edge Detector

Original Image

Smoothed Image



Gradient Magnitude



Non-maximum Suppression Result

Thresholding and Edge Tracking Result

Harris Corner Detector

- Corners are points in an image where the intensity changes significantly in multiple directions, making them useful for various computer vision tasks like feature matching for object recognition, tracking, registration etc.
- Harris corner (Harris and Stephens, 1988) is a popular algorithm for detecting corner features based on first order derivatives of intensity.
- Corner detection with Harris detector involves the following steps:

1. *Gradient Calculation:* The intensity gradients G_x and G_y are computed in the horizontal and vertical directions (for example, using the Sobel operator).
2. *Structure Tensor*: the second moments of the gradients are then computed as the structure tensor matrix:

$$A(x,y) = \begin{bmatrix} \sum_{u,v} w(u,v) G_x^2(u+x, v+y) & \sum_{u,v} w(u,v) G_x(u+x, v+y) G_y(u+x, v+y) \\ \sum_{u,v} w(u,v) G_x(u+x, v+y) G_y(u+x, v+y) & \sum_{u,v} w(u,v) G_y^2(u+x, v+y) \end{bmatrix}$$

, where $w(x,y)$ is an window function (such as Gaussian), that weighs the pixels in the neighborhood.

Harris Corner Detector

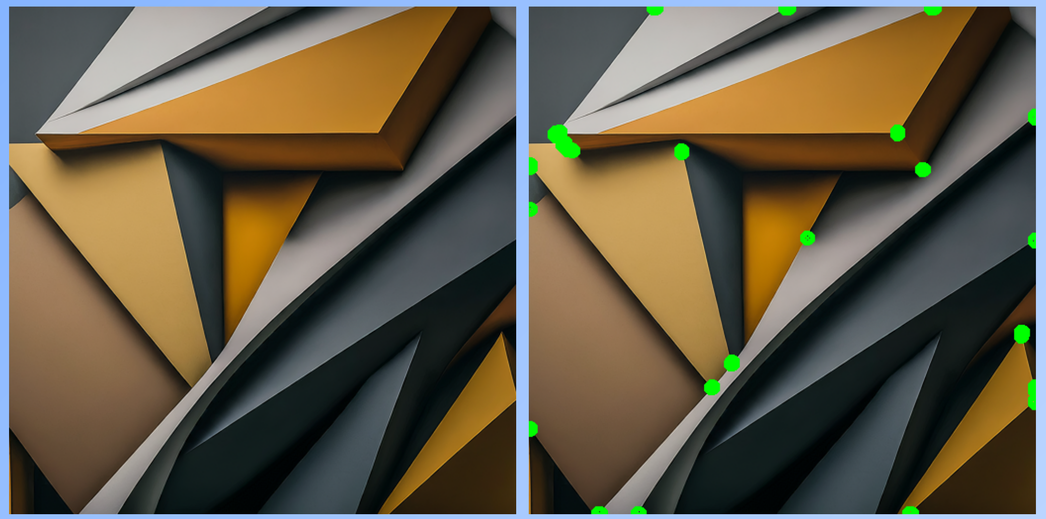
3. *Corner Response function:* The "cornerness" of each pixel is then quantified as follows:
   $R(x,y) = \det(A(x,y)) - \alpha \cdot (\text{Trace}(A(x,y)))^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$
   where λ_1 and λ_2 are the eigenvalues of the structure tensor matrix $A(x,y)$, and α is an empirically determined constant that is typically set to a value in the range $[0.04, 0.06]$

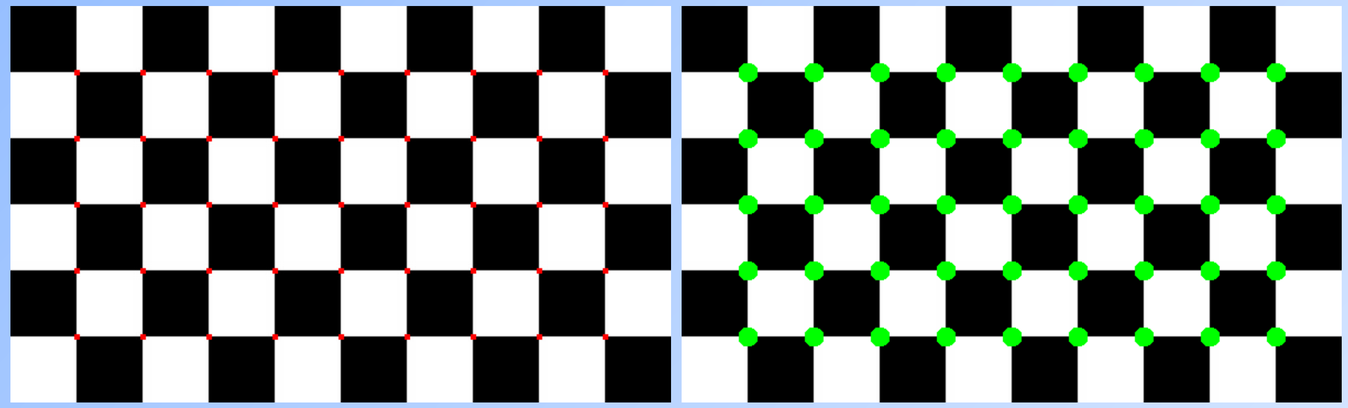
4. Thresholding and Non-maximum Suppression: the corner response is thresholded to identify potential corner points. NMS over a local window is often applied to keep the local maxima.

The values of corner response R can be interpreted as follows:

- Flat regions: λ_1 and λ_2 are small, R is positive and small.
- Edges: $\lambda_1 >> \lambda_2$ or vice versa, $R < 0$
- Corners: λ_1 and λ_2 are large, and $\lambda_1 \sim \lambda_2$, R is large.

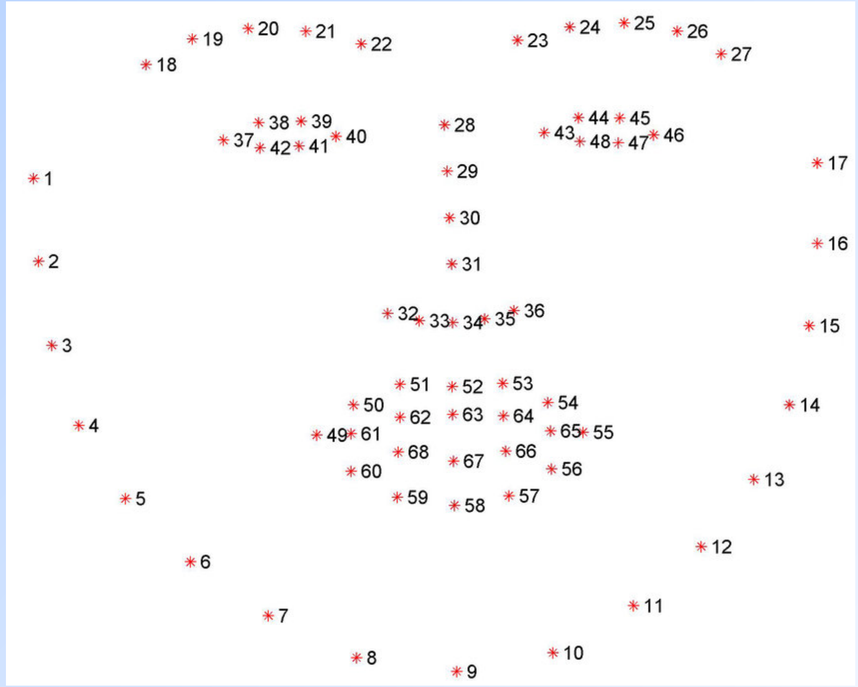
Harris Corner Detector





Keypoint Detection

- Keypoints are distinct and repeatable points of interest in an image that can be used to identify and match features between different images or between different views of the same scene.
- Keypoints are located at positions in the image where the surrounding features are unique or distinctive.
- Keypoints are typically chosen to be robust to changes in scaling, rotation and illumination.
- Keypoints play a key role in providing features for visual search and object matching.



Example of Keypoints: the 68 point facial landmarks

Scale Invariant Feature Transform (SIFT)

- Scale Invariant Feature Transform (SIFT) (Lowe, 2004) is a powerful local feature (keypoint) detection operator that is based on scale-space representation of an image using difference of Gaussians (DoGs).

- The steps involved in detecting key points using SIFT are as follows:

1. *Scale Space Extrema Detection*

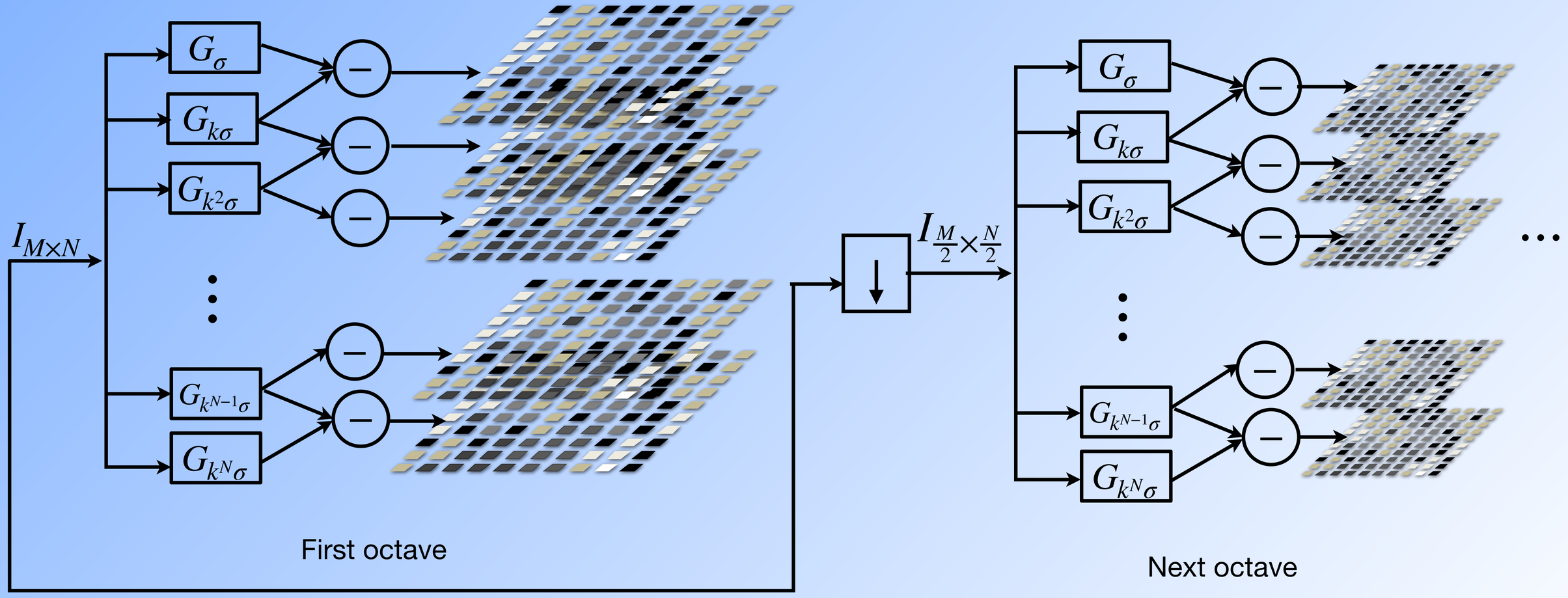
   a. The image is Gaussian blurred at different scales (σ values) and resolutions.

   $L_\sigma(x,y) = G_\sigma(x,y) * I(x,y)$, where $G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2+y^2}{2\sigma^2})$

   b. Difference of Gaussians (DoG): the difference of Gaussians at successive values of σ is computed as follows:

   $D_\sigma(x,y) = L_\sigma(x,y) - L_{k\sigma}(x,y)$, where $k = \sqrt{2}$ (typically).

Scale Invariant Feature Transform (SIFT)



Scale Invariant Feature Transform (SIFT)

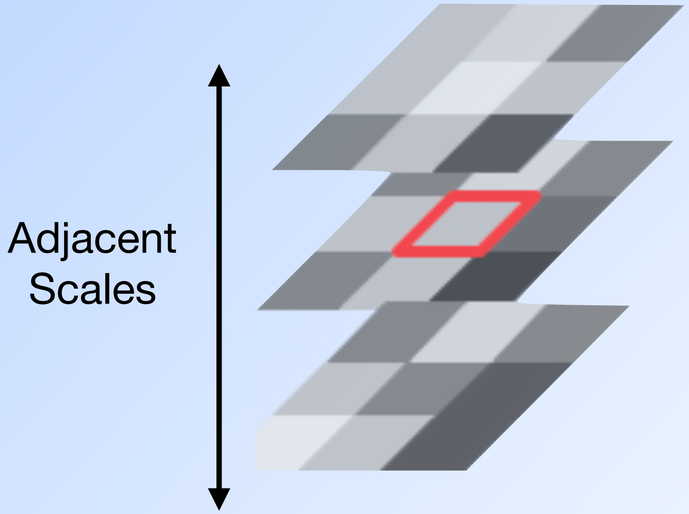
2. *Extrema Detection and Refinement*

a. Extrema Detection: the local maxima and minima of the DoG images in both spatial and scale dimensions are identified by comparing each pixel of the DoG responses with its 26 neighbors in scale space.

b. Accurate Keypoint Localization: the keypoint localization is improved by fitting a 3D quadratic function around the detected extremum.

c. Extrema Refinement: the detected DoG extremes are filtered to eliminate unreliable candidates for keypoints by checking the following:
   - Whether the extrema has high contrast
   - Whether the extrema is not located on or near an edge.

   The extrema that do not meet these conditions are rejected.



Extrema detection in scale-space.

Scale Invariant Feature Transform (SIFT)

3. *Orientation Histogram*

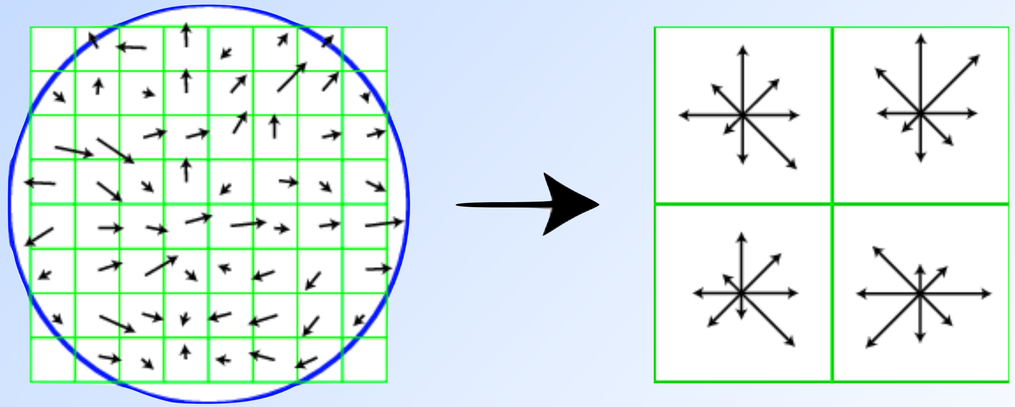
   a. The gradient of the 16×16 local patch around the keypoint is computed at the location and scale of each keypoint:

$$m(x,y) = \sqrt{\left(\frac{\partial I_G(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I_G(x,y)}{\partial y}\right)^2}, \quad \theta(x,y) = \tan^{-1}\left(\frac{\partial I_G(x,y)}{\partial y} \Big/ \frac{\partial I_G(x,y)}{\partial x}\right)$$

   b. To achieve rotation invariance, the orientations of the local patch are computed relative to the keypoint's orientation as $\theta'(x,y) = (\theta(x,y) - \theta(x_0,y_0)) \bmod 2\pi$

   c. Using the orientations $\theta'(x,y)$, a 8-bin gradient orientation histogram is constructed for each 4×4 quadrant.

   d. The contribution of each gradient orientation to its corresponding bin is the Gaussian weighted magnitude of the gradient.



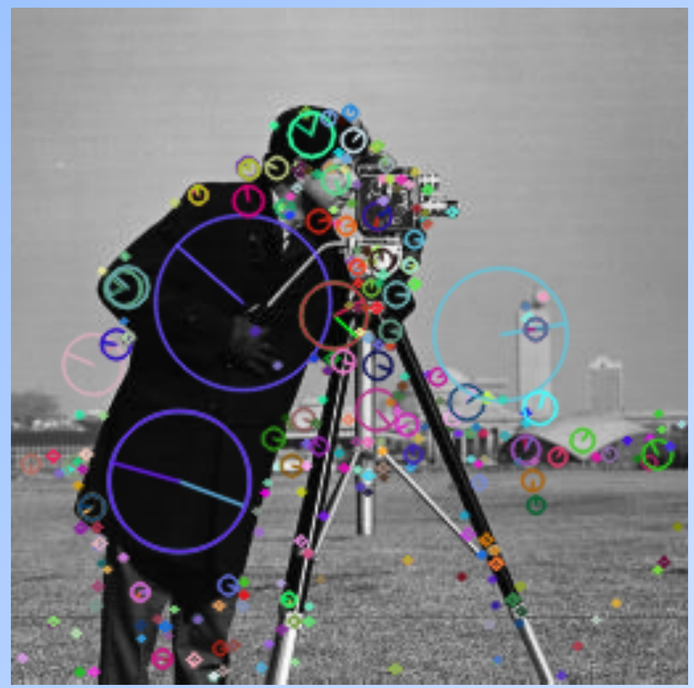
Local patch

4×4 quadrants

Scale Invariant Feature Transform (SIFT)

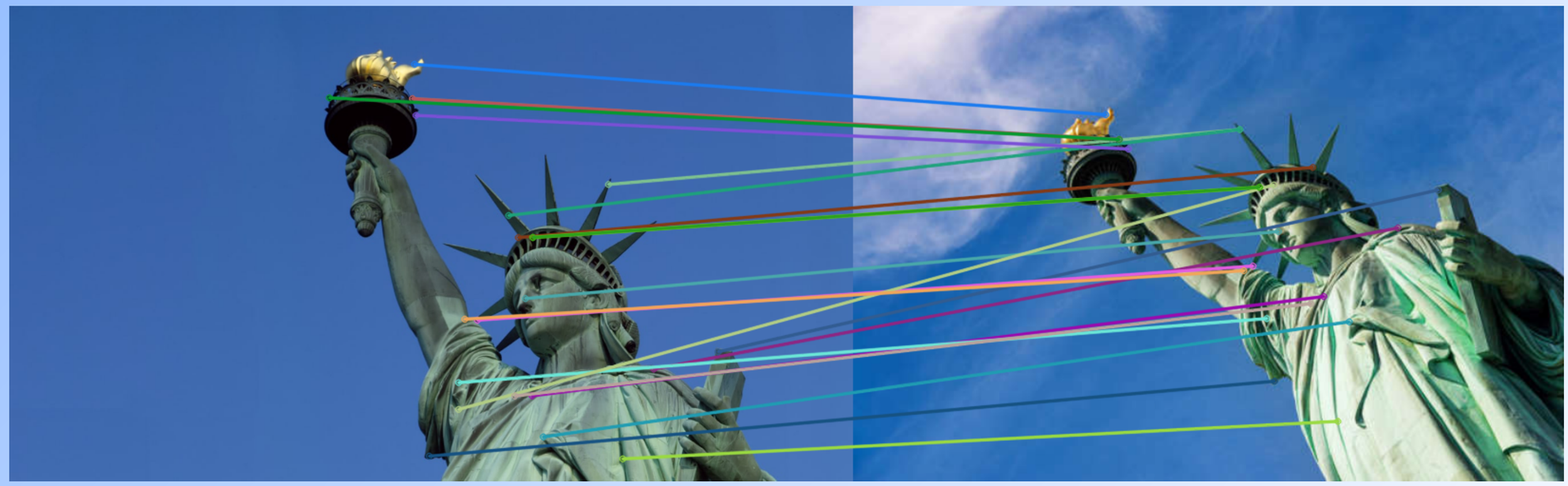
4. *SIFT Descriptor*

    a. The histograms from all quadrants are concatenated to form a 128 dimensional feature descriptor for each keypoint.
    b. The SIFT feature descriptor is normalized to a unit-length vector to achieve illumination invariance.



Scale Invariant Feature Transform (SIFT)

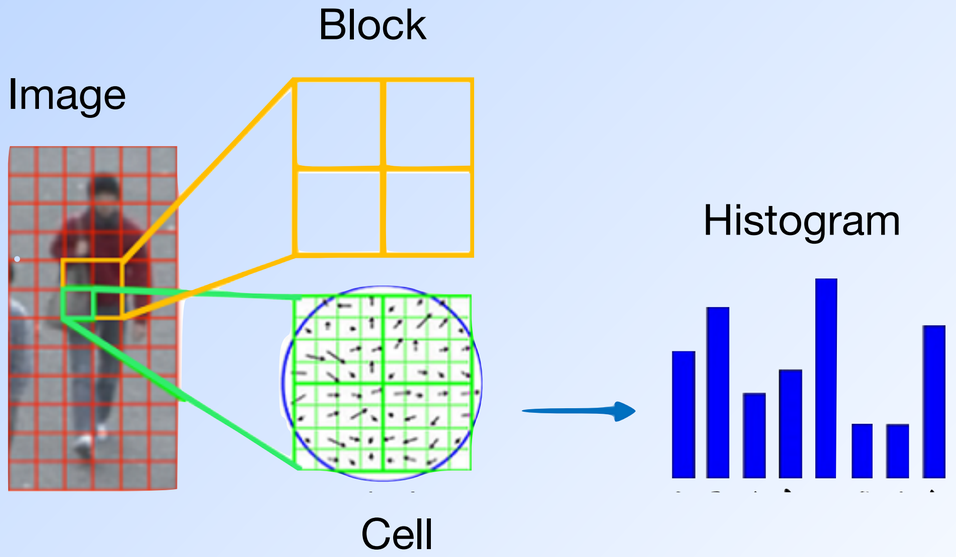
- SIFT features can be used to establish correspondence between images that have different scales, rotations and other transforms.
- The concept of matching keypoints is employed in image stitching, image registration, object recognition and object tracking, 3D reconstruction etc.



Histogram of Oriented Gradients

- The Histogram of Oriented Gradients (HoG) is a widely used feature descriptor for object detection.
- It captures the distribution of gradient orientations in localized portions of an image, providing a robust representation of the object's shape and appearance.
- The steps to compute the HoG descriptors are outlined as follows:

1. Gradient Computation: the horizontal and vertical gradients are computed and magnitude and orientation of the gradient vectors are initially computed.
2. Cell histogram Computation: the pixels are divided into non-overlapping cells, and the gradient orientation histograms are computed for each cell. The vote of each pixel towards the gradient histogram is weighted by its gradient magnitude.
3. Histogram Normalization: the cells are grouped into overlapping blocks, and the histograms from all cells of a block are concatenated and normalized to achieve illumination invariance. The final HoG descriptor are the normalized histograms from all blocks in an image.

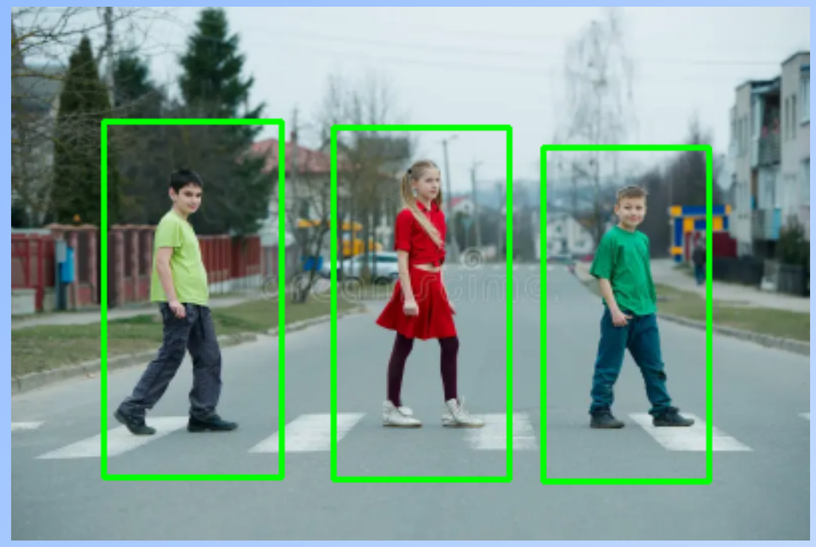


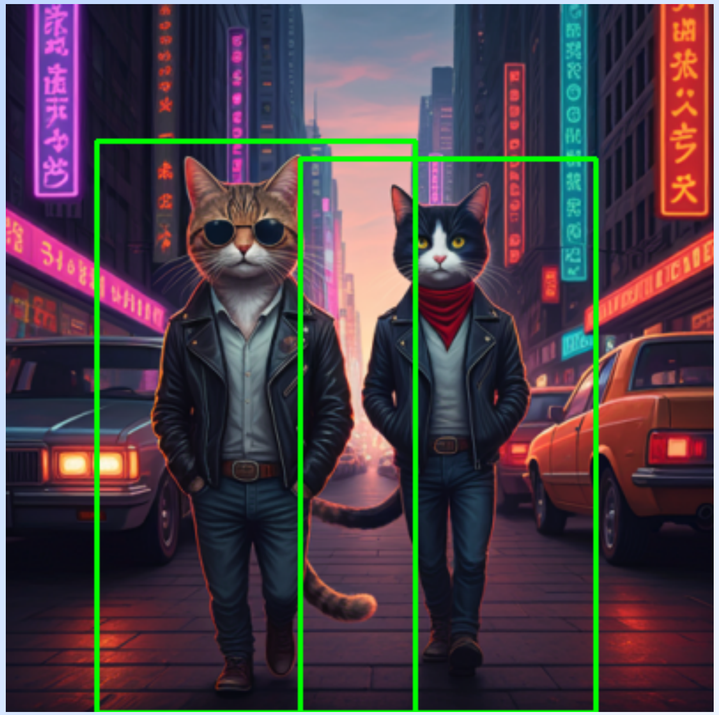
Histogram of Oriented Gradients

- The final HoG descriptor provides a compact representation of the gradient information in the image.
- Cell histograms capture local gradient details, which are important for distinguishing fine textures and shapes.
- Normalization over blocks ensure that the feature descriptors are invariant to changes in illumination and contrast.
- The HoG descriptor effectively balances capturing details and computational efficiency.

Pedestrian Detection Using HoG

- HoG descriptors were shown to be particularly effective for pedestrian detection (Dalal and Triggs, 2005).
- The HoG descriptors from overlapping blocks were used to train a linear support vector machine classifier for pedestrian detection.





Hough Transform

- The Hough transform (Hough, 1962) is an image analysis tool used to detect curves of specific shapes, such as lines, circles, ellipses etc. from an image.
- The Hough transform operates on an edge map and relies on letting edges vote for plausible locations of specific shapes.
- The Hough transform is robust to the presence of noise, and can connect “broken” edges in the edge map to detect the desired shape.
- The Hough transform can be generalized to any geometric shape.
- The Hough transform is computation and memory intensive.

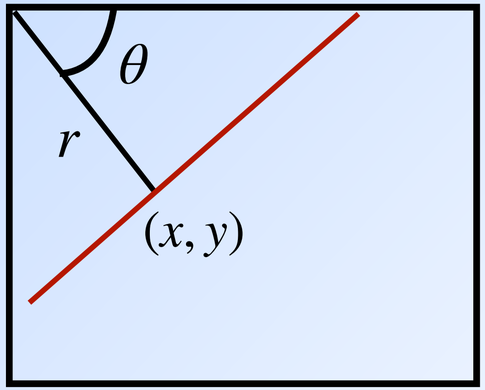
Hough Transform

Hough Transform for Line Detection

1. Line parameterization: A line is parameterized as follows:
   $x\cos(\theta) + y\sin(\theta) - r = 0$, where r is the distance of the origin from the line, and θ is the angle of the line normal with the x-axis.

   Thus, each (r, θ) pair represents an unique line in the image space.

2. Voting or Accumulation:
   - A discrete 2D array called the accumulator is constructed by quantizing the parameter space into discrete bins.
   - For each point (x, y) in the image space, r is computed for the range of values of θ.
   - For each pair (r, θ) thus computed, the corresponding entry in the accumulator array is incremented.

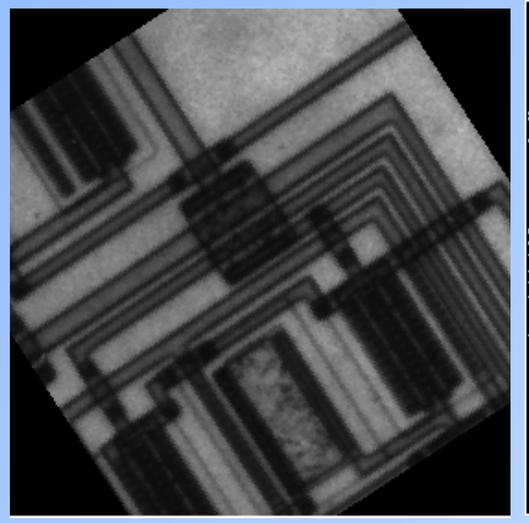


Line representation

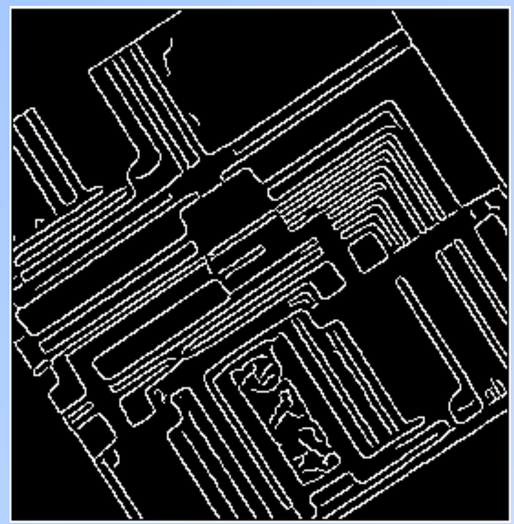
Hough Transform

Hough Transform for Line Detection

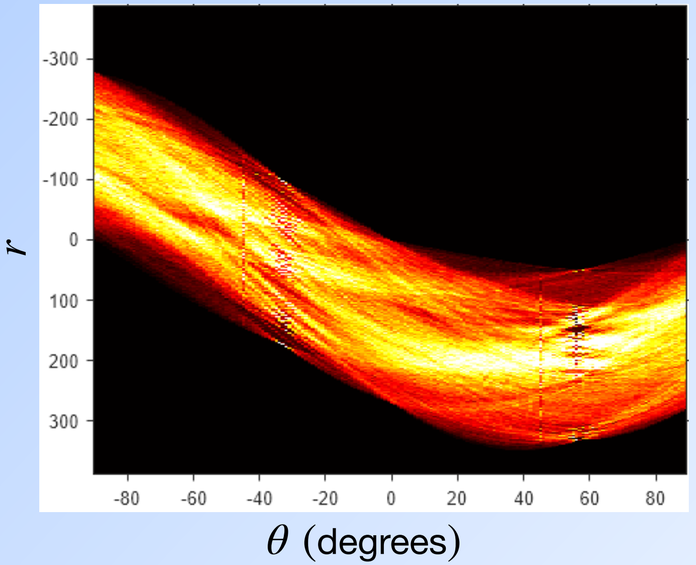
3. Line Detection: the accumulator array is appropriately thresholded to find peaks which correspond to the detected lines, i.e. if a peak occurs at (r, θ), there is a line at a distance of r from the origin, whose normal makes an angle θ with the x-axis.



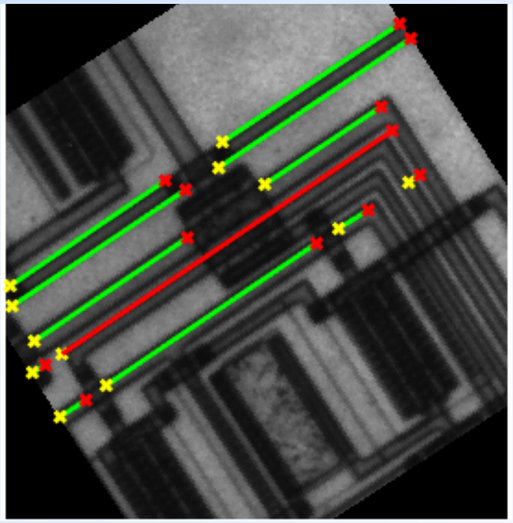
Input image



Edge map



Accumulator Visualization



Detected Lines

Hough Transform

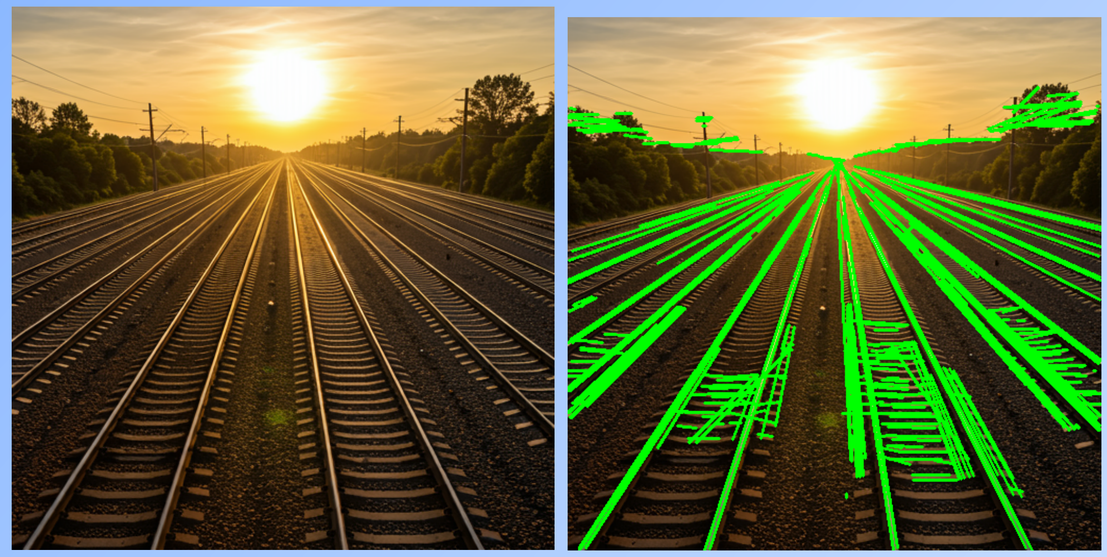
Hough Transform for Circle Detection

1. Circle parameterization: A circle is represented by 3 parameters, the centre coordinates (x_0, y_0), and its radius r.
   $(x - x_0)^2 + (y - y_0)^2 = r^2$

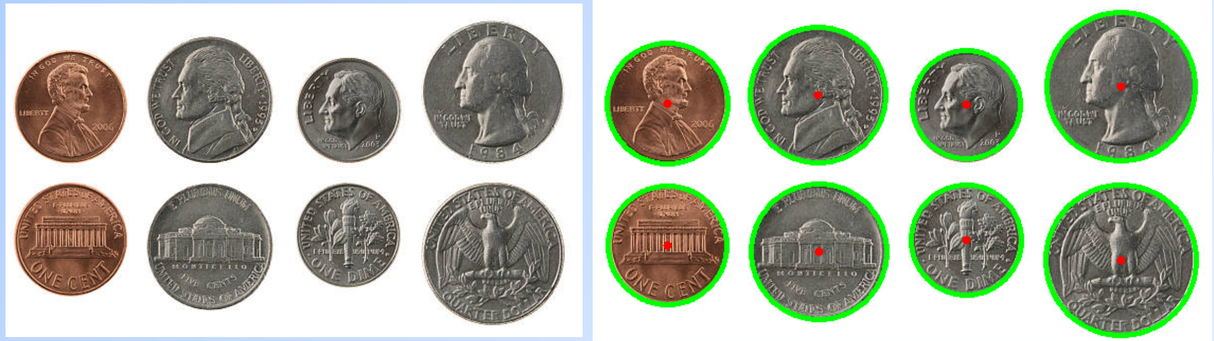
2. Voting or Accumulation:
   - In the case of circle, the accumulator array is 3-dimensional, owing to the three parameters.
   - For any edge point (x, y) and each possible radius r, and centre coordinates (x_0,y_0) are computed as $x_0 = x - r\cos(\theta)$ and $y_0 = y - r\sin(\theta)$, by varying the angle θ between the edge point and the circle centre between 0 to 2π.

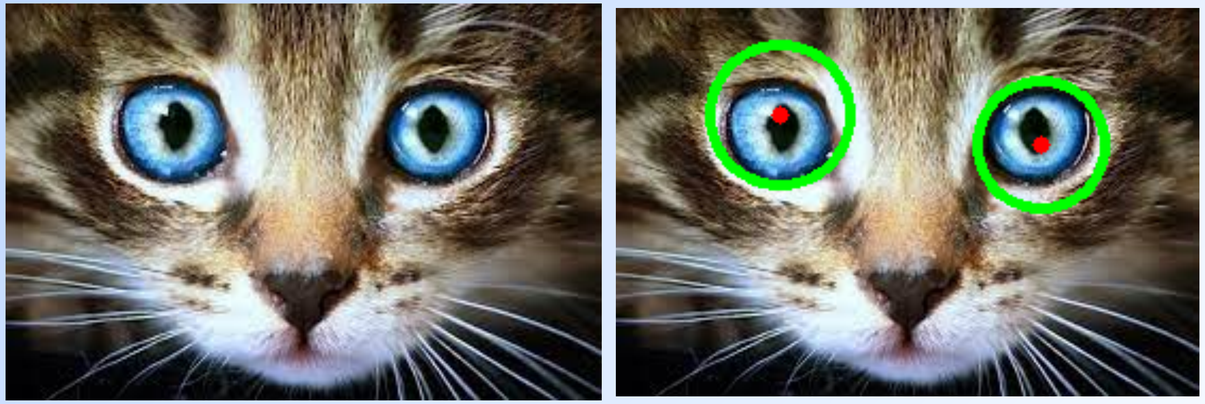
3. The peaks in the accumulator array correspond to the detected circles.

Hough Transform Results



Line Detection

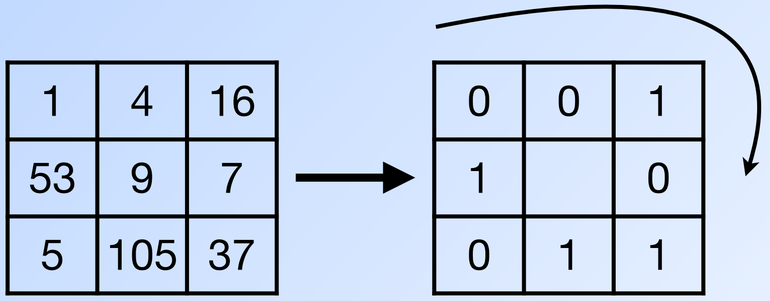




Circle detection

Local Binary Pattern

- Local Binary Patterns (LBP) is a feature extraction technique that is designed to capture local texture information by comparing each pixel to its neighbors.
- Each pixel is assigned a binary code by comparing its intensity to its neighboring pixels.
- The computation of LBP involves the following steps:



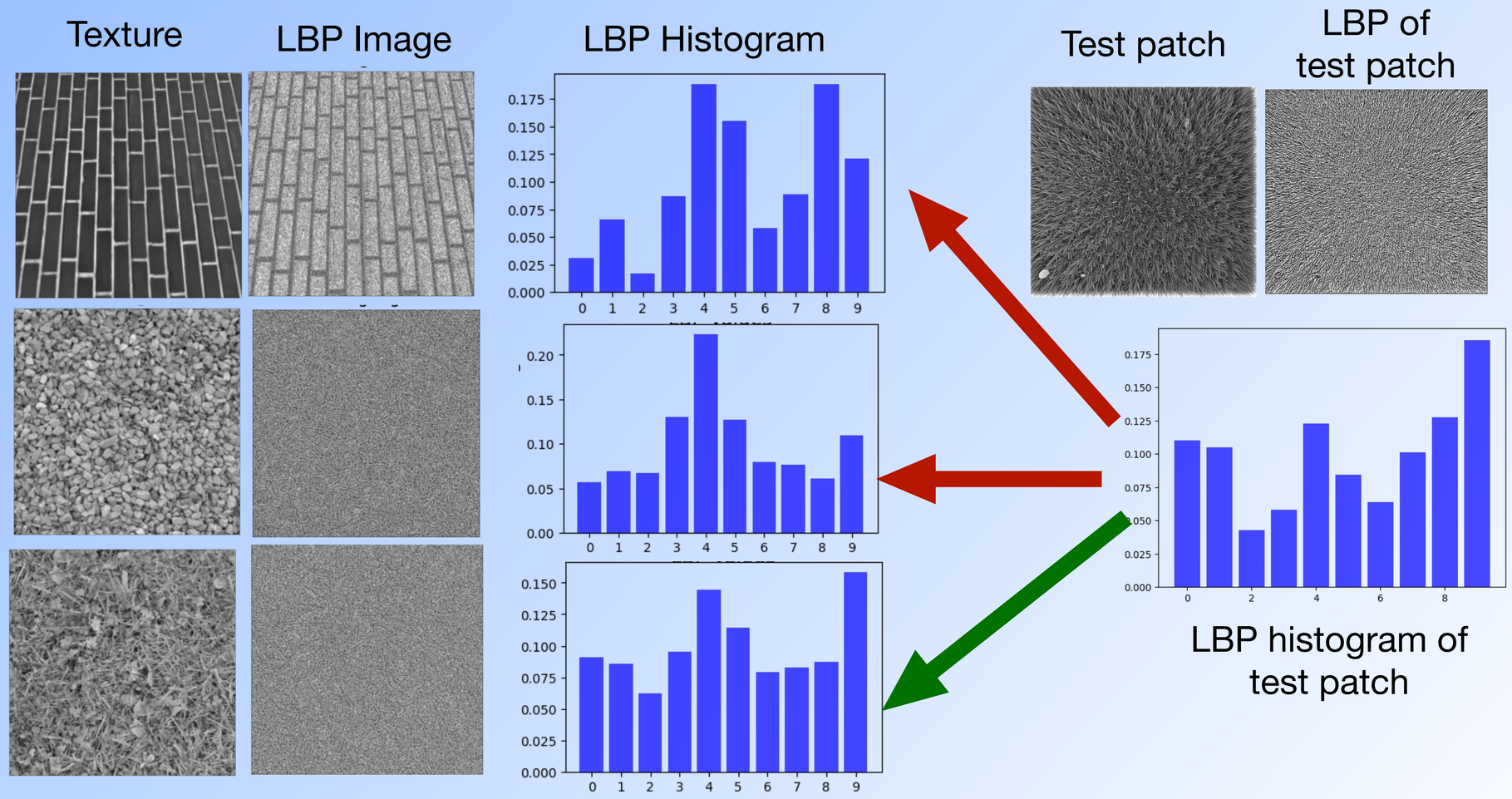
8 bit LBP code: 00101101
Decimal code: 45

LBP computation on an example neighborhood

1. Defining the neighborhood: the neighborhood around each pixel in the image is defined. For example the 8 neighbors in a 3×3 window.
2. Pixel comparisons: each pixel is compared with its neighborhood pixels. If its intensity is greater than the neighborhood pixel, the corresponding comparison outcome is assigned a value of 1, else 0.
3. LBP code: the binary LBP code for each pixel is obtained by grouping the above comparison outcomes in the local neighborhood in a particular order, usually clockwise.
4. For a pixel having an LBP code $(b_7, b_6, \cdots b_0)$, where $b_i \in \{0,1\}$, the LBP value is $\sum_{i=0}^{7} b_i \cdot 2^i$

Local Binary Pattern

- The histogram computed from the LBP could computed for the purpose of texture matching.
- A new texture could be matched by comparing its LBP histogram with the histogram of the textures present in the database (for example, using K-Nearest Neighbors).



Gray Level Co-occurence Matrix

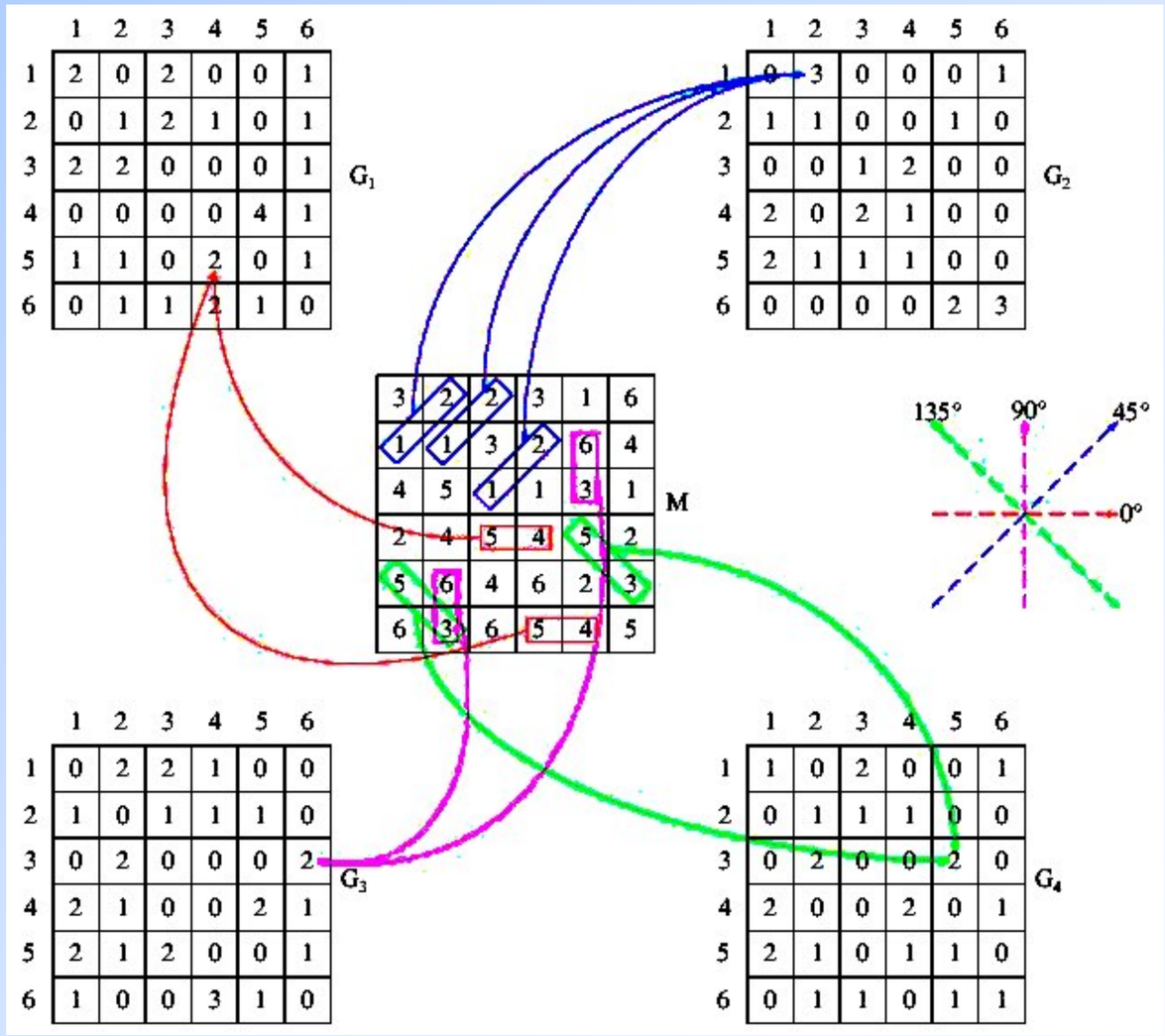
- The Gray Level Co-occurence Matrix (GLCM) offers a powerful technique for characterizing the spatial relationships between pixel intensities.
- GLCM represents how often two pixel intensities occur together in a pre-defined spatial relationship.
- The spatial arrangement of pixel pairs is defined by the following:
  - Pixel offset: distance between two pixels
  - Direction: the direction in which the pixel pairs are considered
- For each such spatial arrangement of the pixel pairs of an image having N gray levels, the GLCM is a $N \times N$ matrix.
- The element $P(i, j)$ of the GLCM matrix P represents the number of times gray level i occurs in the specified spatial relationship with gray level j.
- The GLCM can be normalized to sum to 1 to get the likelihood distribution of pixel cooccurrences.

Gray Level Co-occurence Matrix

- The following features are commonly extracted from each GLCM capture texture information:
  - Contrast: $\sum_{i,j}(i-j)^2 \cdot P(i,j)$
  - Energy: $\sum_{i,j} P(i,j)^2$
  - Homogeneity: $\sum_{i,j} \frac{P(i,j)}{1+|i-j|}$
  - Correlation: $\frac{\sum_{i,j}(i-\mu_x)(j-\mu_y)\cdot P(i,j)}{\sigma_x\sigma_y}$, where

    $\mu_x = \sum_i i \cdot \sum_j P(i,j), \mu_y = \sum_j j \cdot \sum_i P(i,j)$ and

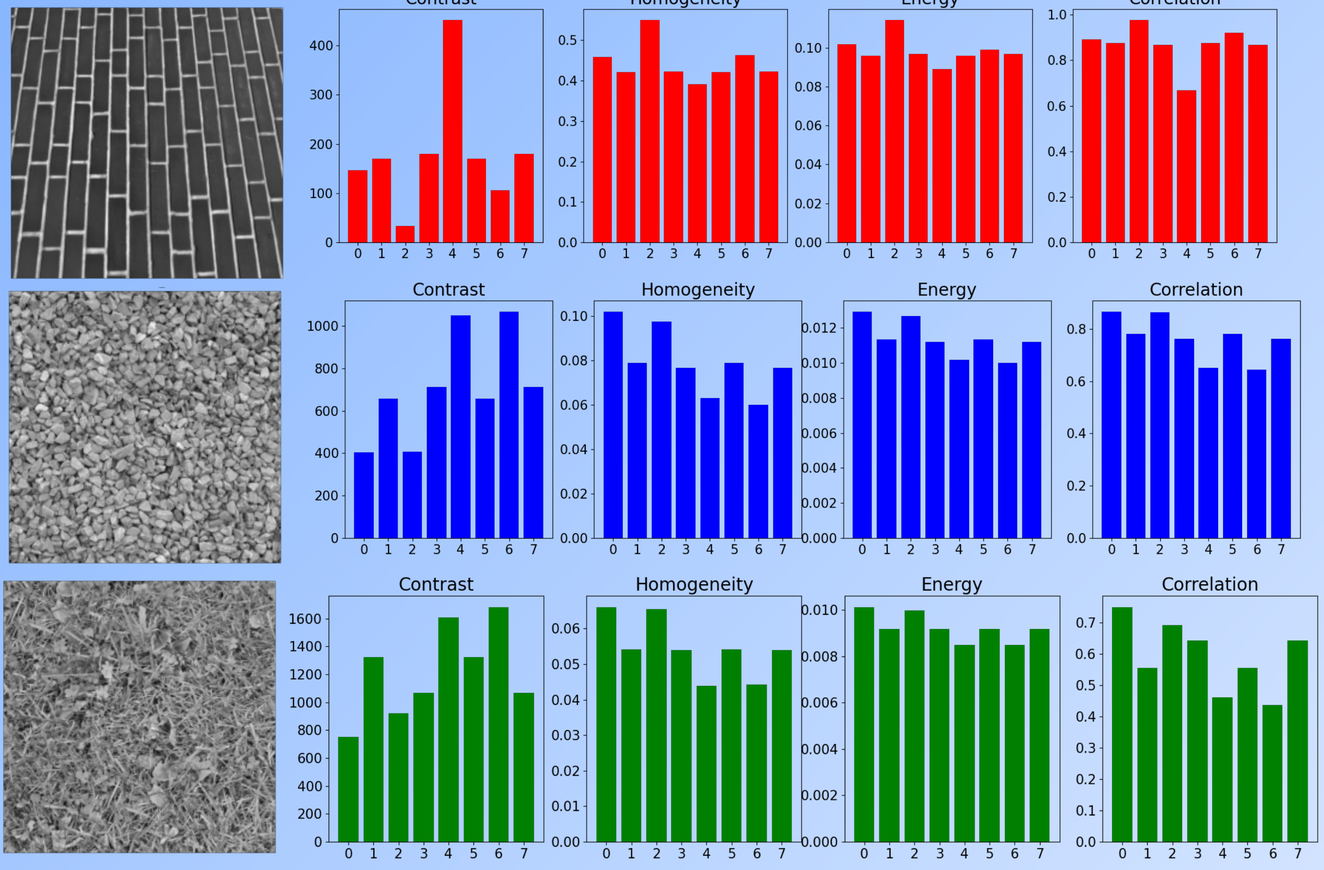
    $\sigma_x^2 = \sum_i (i-\mu_x)^2 \cdot \sum_j P(i,j), \quad \sigma_y^2 = \sum_j (j-\mu_y)^2 \cdot \sum_i P(i,j)$
  - Entropy: $-\sum_{i,j} P(i,j)\log(P(i,j))$



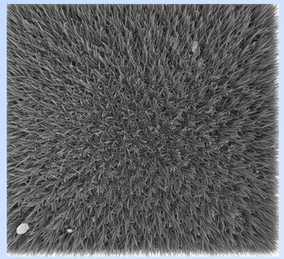
Gray Level Co-occurence Matrix

Texture

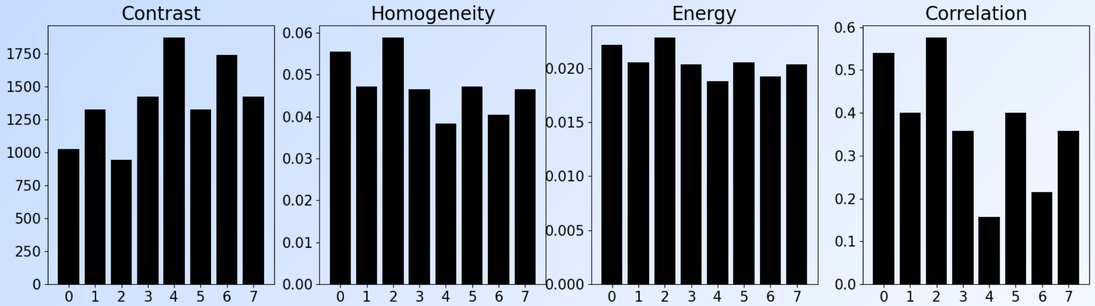
GLCM feature



Test patch



GLCM feature



GLCM features computed using 8 spatial relationships (2 pixel offsets and 4 directions)