L6 (Image Feature Detection)

Edge Detection

Local gradient of luminance(brightness) is used to detect edges.
The gradient is a vector given by:

$$\nabla s(x,y) = \begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} \frac{\delta s(x,y)}{\delta x} \\ \frac{\delta s(x,y)}{\delta y} \end{bmatrix}$$

where s_x is the gradient in horizontal direction and s_y is the gradient in the vertical direction.

- magnitude of the gradient measures how **strong** an edge is:

$$|\nabla s| = \sqrt{s_x^2 + s_y^2}$$

- direction of the edge is given by :

$$\theta = \arctan \frac{s_y}{s_x}$$

  Vertical gradient corresponds to horizontal edges, while horizontal gradient corresponds to vertical edges.
  Discrete Approximation:
- $s_x = s[x,y] - s[x-1,y]$
- $s_y = s[x,y] - s[x,y-1]$

1D Filter-based edge detection (2x2)

Vertical edges (horizontal gradient filter): $G_v = [-1 \quad 1]$
Horizontal edges (vertical gradient filter): $G_h = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Since these gradients are very sensitive to noise, we have to add some averaging in direction **orthogonal to gradient filter**:

- Vertical edges: $A_v = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- Horizontal edges: $A_h = \frac{1}{2}[11]$
  So, these two gradient and averaging filter can be combined to get a simple edge detection filter (separable):

$$H_v = G_v * A_v = \frac{1}{2}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$H_h = G_h * A_h = \frac{1}{2}\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$$

These 2x2 masks are simple but they are **not symmetric** about the center point.

Diagonal edge detection

So H_v and H_h are not symmetric around the center point, so they cannot detect diagonal edges. Roberts cross operator is helpful for detecting diagonal edges:

$$H_{45} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$H_{135} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

2x2 filters (Roberts, H_h , H_v) are not enough

1. Lack of center symmetry: this leads to misalignment issues in gradient estimation
2. Limited directional sensitivity: Roberts only detects diagonal edges(45 and 135) but **misses** horizontal and vertical edges

- H_v and H_h detect vertical and horizontal edges but do not handle diagonal edges well.

3x3 filters

- Larger masks allow better edge localization and more stable gradient
- Symmetric filters (Sobel or Prewitt) ensure better directional accuracy
- Smoothing effect: with more neighbors comes less noise sensitivity
- Detecting edges in multiple directions (not just vertical, horizontal or diagonal)

Prewitt Edge Detector

$$H_v = A_v * G_v = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$H_h = G_h * A_h = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Roberts edge detector is similar but with additional scalar weighting of $\frac{1}{3}$.

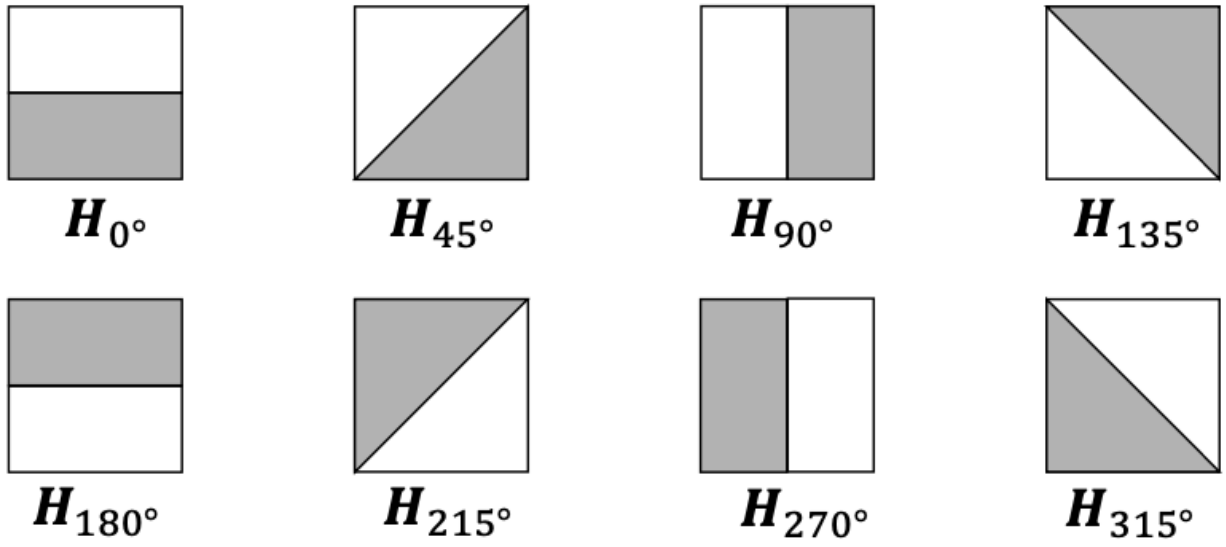
Sobel Edge Detector

It has better smoothing (noise-suppression characteristics) compared to Prewitt edge detector.

$$H_v = A_v * G_v = \begin{bmatrix} 1 \\ \mathbf{2} \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ \mathbf{-2} & 0 & \mathbf{2} \\ -1 & 0 & 1 \end{bmatrix}$$

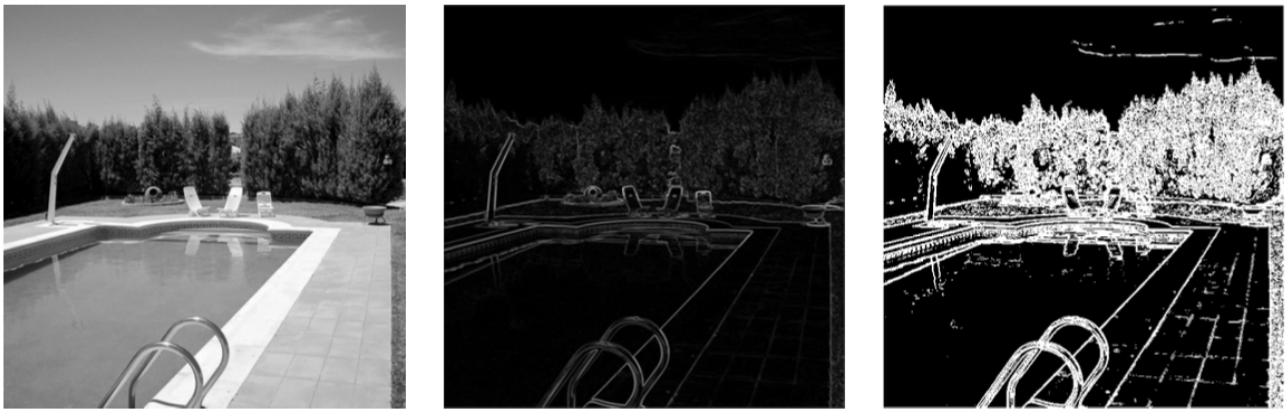
$$H_h = A_h * G_h = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & \mathbf{2} & 1 \end{bmatrix} = \begin{bmatrix} -1 & \mathbf{-2} & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Other filter masks that are not horizontal nor vertical are derived by counter clockwise rotation of H_h



How to actually get edges from gradient map?

Answer: by thresholding



Original image | Sobel gradient | Gradient thresholding (low threshold)

Depending on the chosen threshold, the edges detected might be too noisy or too little.

Note that the sum of the elements in all above operators is 0. The reason is that such setting ensures no response is given to uniform regions

2nd order derivative based Edge Detection

1st order methods are not enough, they have to be combined with 2nd order methods in practice.

- 2nd derivative (Laplacian) finds edges using **zero-crossings**, which explicitly mark edge locations while 1st order methods just provides a gradient magnitude.
- Laplacian is **isotropic**(rotationally invarient): meaning it can detect edges equally well in all directions without needing separate filters
- In practice, Laplacian is combined with smoothing (LoG) to reduce noise effects.
  Zero crossings responds equally to strong and weak edges by suppressing zero-crossings with low magnitude. This means it is also **sensitive** to noise and fine details. So, the image has to be blurred first.
  Laplacian Operator: sum of all 2nd order partial derivatives:

$$\nabla^2 s(x,y) = \frac{\delta^2 s(x,y)}{\delta x^2} + \frac{\delta^2 s(x,y)}{\delta y^2}$$

  Discrete approximation via gradient of gradient:

$$L_v = G_v * G_v = [1 \quad 2 \quad 1]$$

$$L_h = G_h * G_h = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

  2D Laplacian operator is the sum of horizontal and vertical filters:

$$L_4 = L_v + L_h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

  Non-separable 2D Laplacian operator using 8 neighbors:

$$L_8 = L_v + L_h + \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  Separable 2D Laplacian operator

$$L_s = L_v * L_h$$

Laplacian of Gaussian

Remember Gaussian filter(usually 5 x5 kernel) is given by:

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

Blur first with Gaussian filter, then apply Laplacian operator.

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

Parameter σ controls the width of Gaussian kernel (blur strength). As discussed before, applying Gaussian helps to remove noise in **zero-crossings** calculation:

Without Gaussian

$\sigma = 0.5$

$\sigma = 1$

$\sigma = 2$

Canny Edge Detector

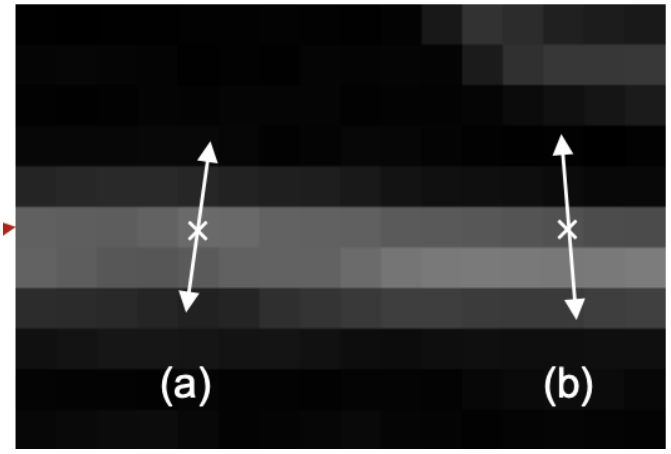
Canny edge detector builds on all of the above by combining gradient magnitude, non-maximum suppression, and hysteresis thresholding for better performance.
Steps:

1. Apply Gaussian smoothing
2. Compute gradient magnitude and angle (Sobel, Prewitt, Robert)
3. Apply non-max suppression
4. Double thresholding for potential edgse
5. Track edges by hysteresis

Non-max Suppression

Why?: the gradient magnitude image is **thick**, but we want **thin edges**
How? Compare each pixel(gradient magnitude) with its two neighbors in the gradient direction:

- if a pixel is not a local maximum, suppress it (set to 0)
- this creates **thin edges**
  For example:
- if $\theta = 0$ (**horizontal edge**)-> compare left & right neighbors
- if $\theta = 90$ (**vertical edge**)-> compare top&bottom neighbors
- if $\theta = 45$ -> compare diagonal neighbors.



(a) is kept
(b) is suppressed

Double thresholding

Some edges are strong, some are weak due to noise or lighting. Let's use double thresholding

- High threshold(T_h)-> strong edges
- Low threshold(T_L)-> weak edges
- $T_H = 2 * T_L$
  If gradient is higher than T_H, it's an edge. If lower than T_L, no edge. If in between, it is a potential edge.

Hysteresis

Weak edges are retained **only if** they are 8-connected to a strong edge (one is enough). This removes isolated noise edges and keeps true edges.

Original

Sobel

$T = 0.15$

Canny

$T_{\text{high}} = 0.15$

$T_{\text{low}} = 0.4T_{\text{high}}$

Comparison of Edge Detectors

Method	Pros	Cons
Sobel	Simple, fast	Thick edges, noisy
Roberts	Small kernel (2×2)	Not rotation invariant
Prewitt	Similar to Sobel	Sensitive to noise
Laplacian of Gaussian (LoG)	Finds zero-crossings	Sensitive to noise
Canny	Best accuracy & noise suppression	Computationally expensive

Hough Transform

Why?
Canny Edge Detection provides a clean and well-defined set of edges but it doesn't detect **shapes.** Given the detected edges from Canny, Hough Transform is used to find **lines**, **circles**, or other parametric shapes. It works by mapping edge points from image space into a transformed parameter space where shapes become easily detectable.
How?
Hough transform transforms shapes into points. Hough transform of an edge map is calculated by **voting mechanism**. Each pixel votes for a line that passes through it. Peaks in this space correspond to dominant lines.

Line Detection

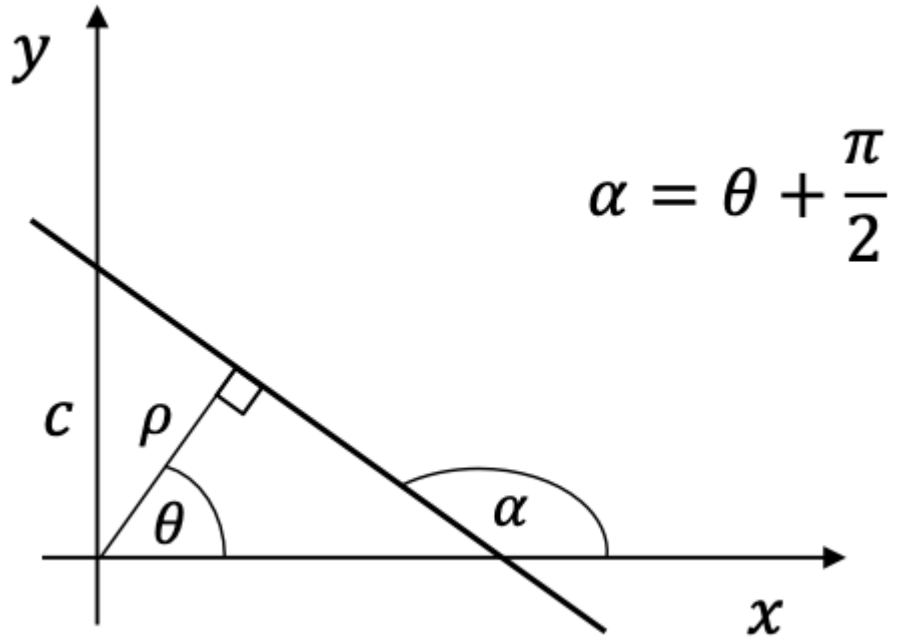
The accumulator here is a discrete (m,c)-plane, voting matrix where all bins initialized by zeros.
A line is better represented by **Hesse normal form**:

$$y = mx + c = \frac{\sin\alpha}{\cos\alpha}x + c$$

or

$$x\cos\theta + y\sin\theta = \rho$$

where ρ is the perpendicular distance from the origin to the line and θ is the angle of the normal.



Circle Detection

Hough transform can be applied to detect any **parametrized** shape:

$$(x - x_0)^2 + (y - y_0)^2 = r_0^2$$

Each pixel votes for possible **circle centers**. A **3D** accumulator is used for x_0, y_0, r_0

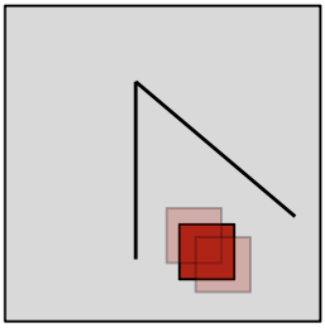
Keypoint Detection

We are interested in finding corners (where two edges meet) and where the gradients change in **multiple directions**. These points are useful for tracking and recognition.
Criteria:

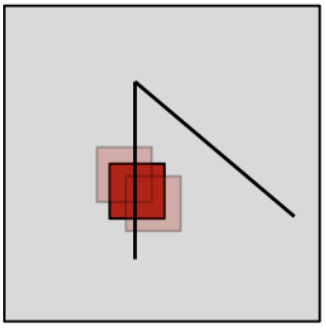
- invariance against shift, rotation, scale, brightness changes
- robust against noise, high repeatability

- accurate localization.

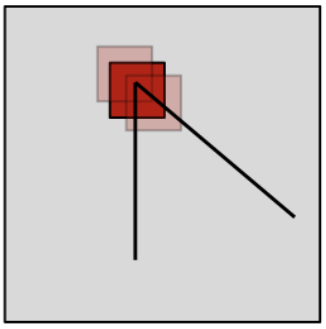
Harris Corner Detector



Flat region: no change in all directions



Edge: no change along edge direction



Corner: significant change in all directions

Harris detectors finds out which case holds above.
Steps:

1. Compute local x and y gradients: s_x and s_y
2. Compute products of local derivatives at every pixel:

$$s_x^2, s_{xy}, s_y^2$$

3. Compute sums of weighted products of derivatives at each pixel
4. At each pixel, define Harris structure matrix $M[x, y]$:

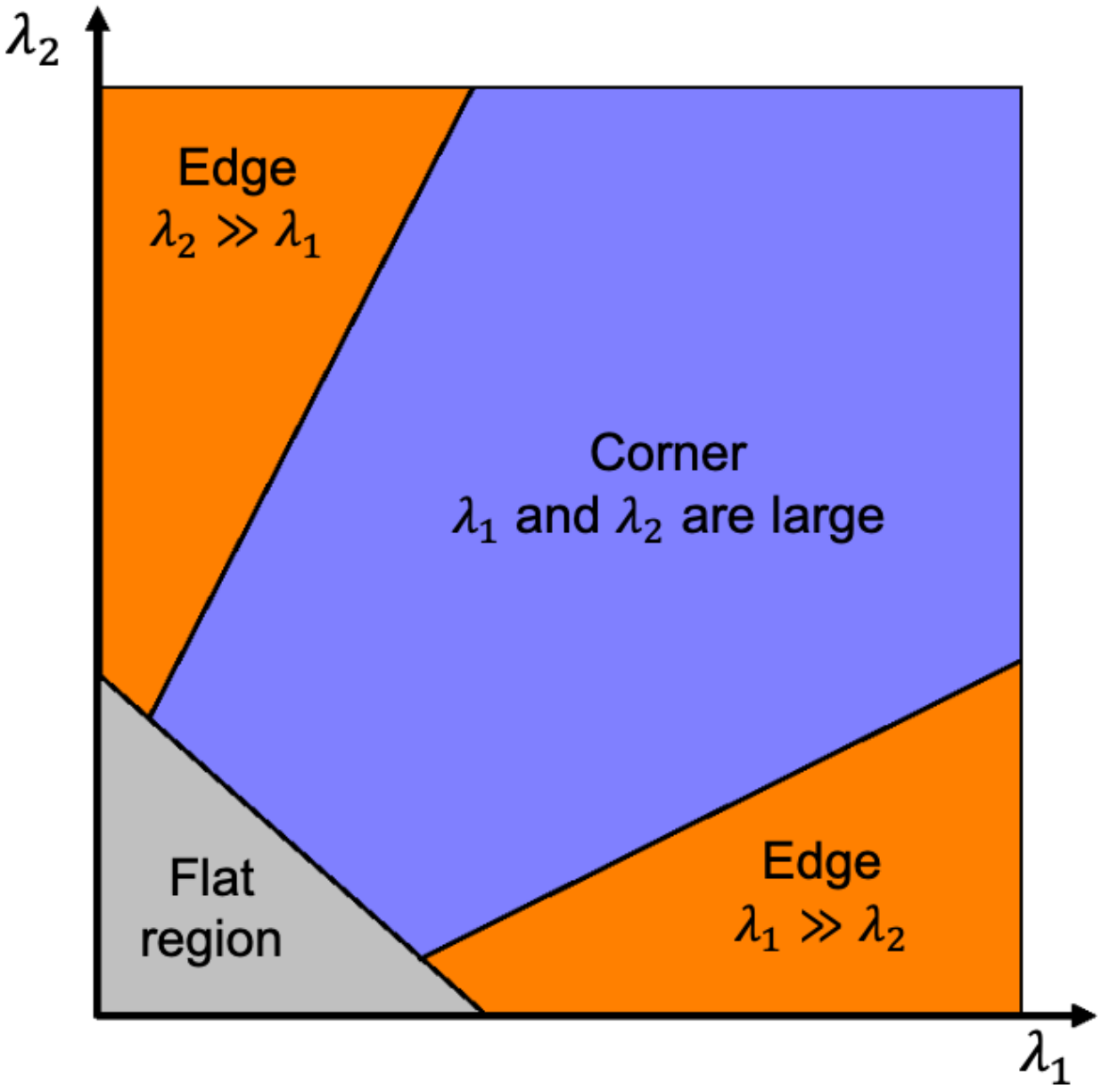
$$M = \begin{bmatrix} \sum_{x,y} w[x,y] s_x^2[x,y] & \sum_{x,y} w_{x,y} s_x[x,y] s_y[x,y] \\ \sum_{x,y} w[x,y] s_x[x,y] s_y[x,y] & \sum_{x,y} w[x,y] s_y^2[x,y] \end{bmatrix}$$

   where the weighted window w is usually a Gaussian that gives more importance to pixels closer to the center of the windows. This helps reduce the impact of noise in the image: $w[x,y] = \exp(-\frac{x^2+y^2}{2\sigma^2})$
5. There is a way to detect corners based on eigenvalues of M structure matrix but it is expensive. Instead, compute "cornerness" value:

$$c[x,y] = det(M) - k(Tr(M))^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$

   here higher value indicates "better" corner.
6. Apply threshold on value of cornerness and apply nonmax suppression.

If both eigenvalues of M are large, intensity changes significantly in two directions. That means it's a corner.



This makes Harris corner detector brightness-shift-rotation-invariant but not **scale-invariant**.

Texture Classification

Regular/periodic textures: nearly periodic structures, analyzed by spectral methods (DFT or power spectrum)
Irregular textures: texture is ensemble of basis texture elements (**texels**) which are different in size and structure. Analyzed by statistical methods.

Co-occurrence matrix is a tool for analyzing texture patterns in images by looking at how pairs of pixel intensities appear together. The matrix C keeps track of how often pairs of pixel intensities (i, j) occur at a specific distance and direction from each other.
The shift (k, l) defines the spatial relationship between pixels. For example $k = 1, l = 0$ would look at horizontally adjacent pixels. $k = 0, l = 1$ would look at vertically adjacent pixels.
$k = 1, l = 1$ for diagonally adjacent pixels.

- Size: $J \times J$, where J is the number of possible gray levels (256 for 8-bit image)
- Asymmetric: $C[i, j] \neq C[j, i]$
- Normalization: divided by MN(total number of pixels) to convert to probabilities.

$$C' = \frac{C}{MN}$$

Co-occurrence matrices become quite large (256×256) for 8 bit resolution. Thus, it's not directly suited as texture feature. In practice, the following features are derived from C' (not C, pay attention!):

- Energy: luminance uniformity of a picture (equal to 1 for a constant image)
- Homogeneity: spatial closeness of C to a diagonal matrix (co-occurrence of the same grey levels)
- Correlation