# 6 Image Feature Detection

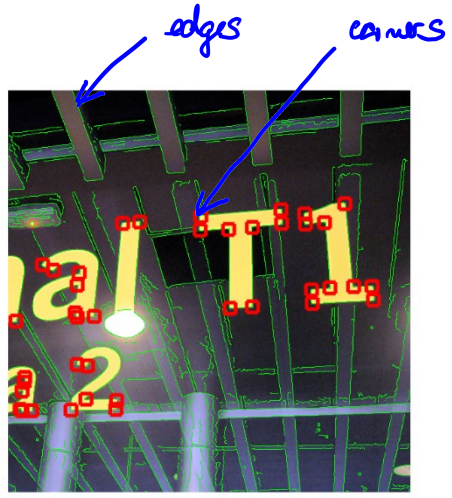
# 6.1 Image Features

Image features are interesting/important local patterns

Detecting features can be an important step in localizing, recognizing or tracking objects

Examples

- Edges
- Lines, curves
- Corners
- …

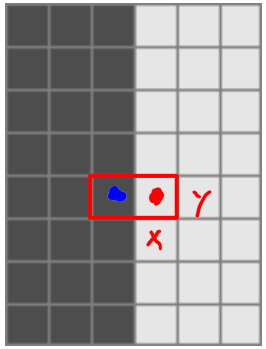
# 6.2 Edge Detection

Edges indicate object outlines, play important role in visual perception

**Idea:** detect local gradient of luminance

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

- Magnitude: $|\nabla s| = \sqrt{s_x^2 + s_y^2}$
- Direction : $\theta = \operatorname{atan}(s_y / s_x)$
- Vertical gradient ↔ horizontal edges
- Horizontal gradient ↔ vertical edges




**Discrete approximation:**

$$s_x = s[x,y] - s[x-1,y]$$
$$s_y = s[x,y] - s[x,y-1]$$

# Edge Detection Using 1D Filters

**Vertical edges:** horizontal gradient filter $\boldsymbol{G}_{\mathrm{v}} = [-1 \quad 1]$

**Horizontal edges:** vertical gradient filter $\boldsymbol{G}_{\mathrm{h}} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Filter matrices $\boldsymbol{G}_{\mathrm{v}}, \boldsymbol{G}_{\mathrm{h}}$ represent impulse responses of FIR filters that are horizontally and vertically flipped

Simple gradient filters are very sensitive to noise

- Pre-filter for local averaging in direction orthogonal to gradient filter

**Averaging filters:** vertical edges $\boldsymbol{A}_{\mathrm{v}} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

horizontal edges $\boldsymbol{A}_{\mathrm{h}} = \frac{1}{2}[1 \quad 1]$

# Edge Detection Using 1D Filters

Simple edge detection using separable filters:

$$\boldsymbol{H}_{\mathrm{v}} = \boldsymbol{G}_{\mathrm{v}} * \boldsymbol{A}_{\mathrm{v}} = \frac{1}{2}\begin{bmatrix} -1 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix} \qquad \boldsymbol{H}_{\mathrm{h}} = \boldsymbol{G}_{\mathrm{h}} * \boldsymbol{A}_{\mathrm{h}} = \frac{1}{2}\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$$

convolution

Detection 2×2 masks compute luminance difference

- Conceptually simple but they are **not** symmetric about the center point
- Diagonal edge detection possible using Roberts cross operator

$$\boldsymbol{H}_{45^\circ} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{H}_{135^\circ} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

45°

**Idea:** Compute central difference using 3×3 masks

- Smallest detection masks that are symmetric about the center point

# Prewitt Edge Detector

Uses simple averaging of 3 neighboring samples (in each direction)

- Introduced in 1970 by Judith M.S. Prewitt

noise reduction, gradient filter } both odd in size

$$\boldsymbol{H}_{\mathrm{v}} = \boldsymbol{A}_{\mathrm{v}} * \boldsymbol{G}_{\mathrm{v}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

3x3 mask

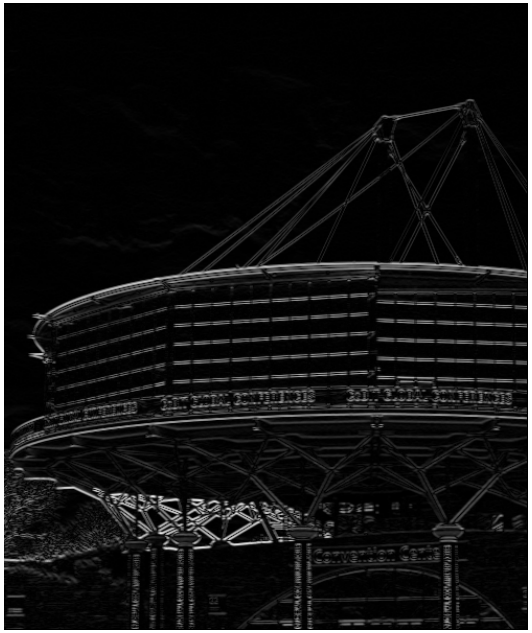
$$\boldsymbol{H}_{\mathrm{h}} = \boldsymbol{G}_{\mathrm{h}} * \boldsymbol{A}_{\mathrm{h}} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Similar approach: **Roberts edge detector**

- As Prewitt edge detector, but with additional scalar weighting of 1/3

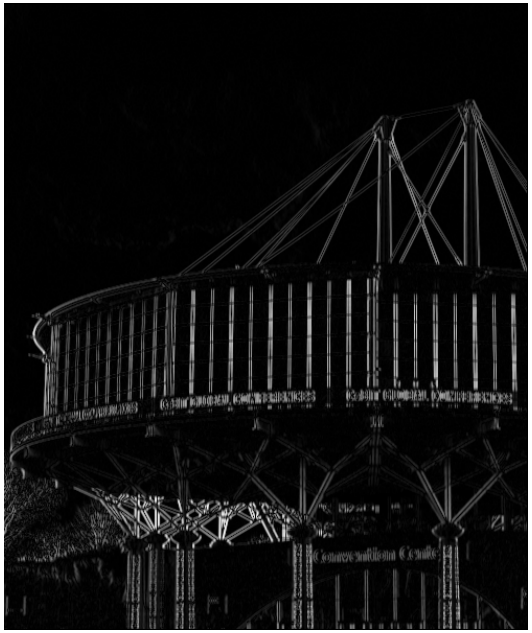
# Prewitt Detector Example


Original Image



Prewitt (vertical magnitude)

horizontal edge / vertical gradient

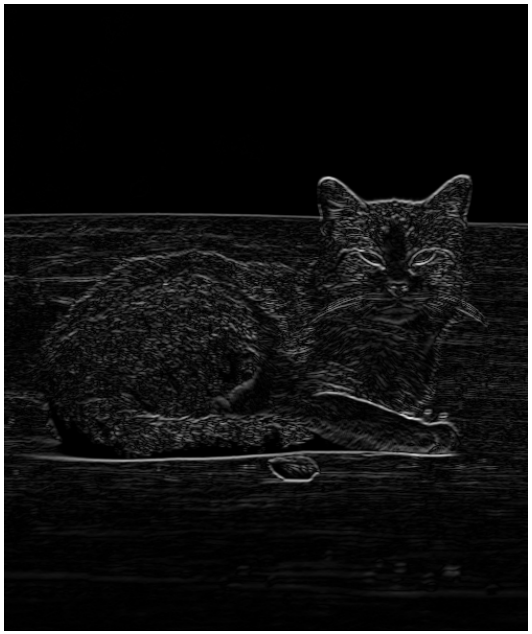


Prewitt (horizontal magnitude)

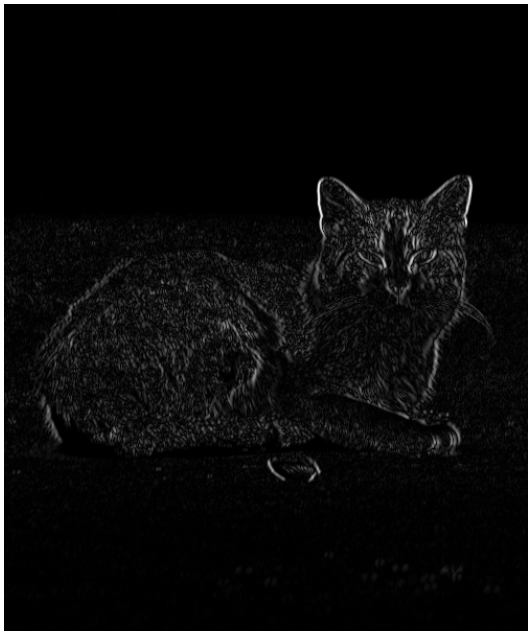
vertical edge / horizontal gradient

# Prewitt Detector Example

Original Image



Prewitt (vertical magnitude)



Prewitt (horizontal magnitude)

# Sobel Edge Detector

Different averaging filter

- Better noise-suppression characteristics (smoothing)
- Introduced in 1970 by R. Sobel

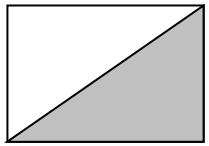
noise fill. gradient

$$\boldsymbol{H}_{\mathrm{v}} = \boldsymbol{A}_{\mathrm{v}} * \boldsymbol{G}_{\mathrm{v}} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [-1 \quad 0 \quad 1] = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{H}_{\mathrm{h}} = \boldsymbol{G}_{\mathrm{h}} * \boldsymbol{A}_{\mathrm{h}} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} [1 \quad 2 \quad 1] = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

# Sobel Edge Detector

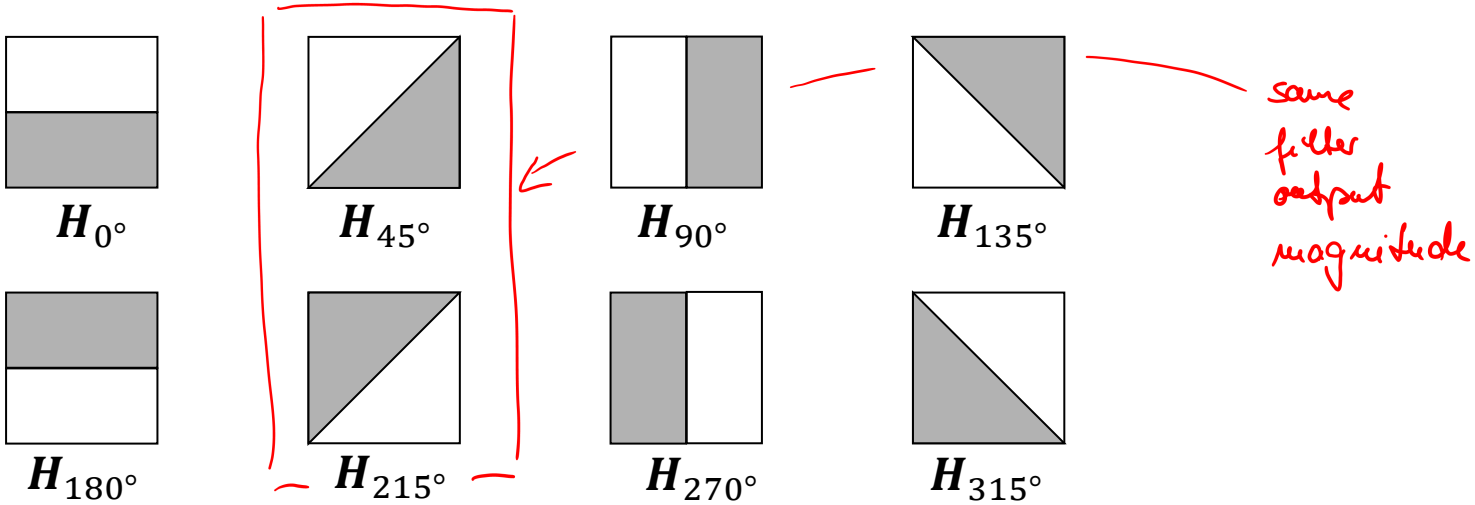
Other filter masks are derived by counter clockwise rotation of $\boldsymbol{H}_h$, e.g.



$$\boldsymbol{H}_{45°} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

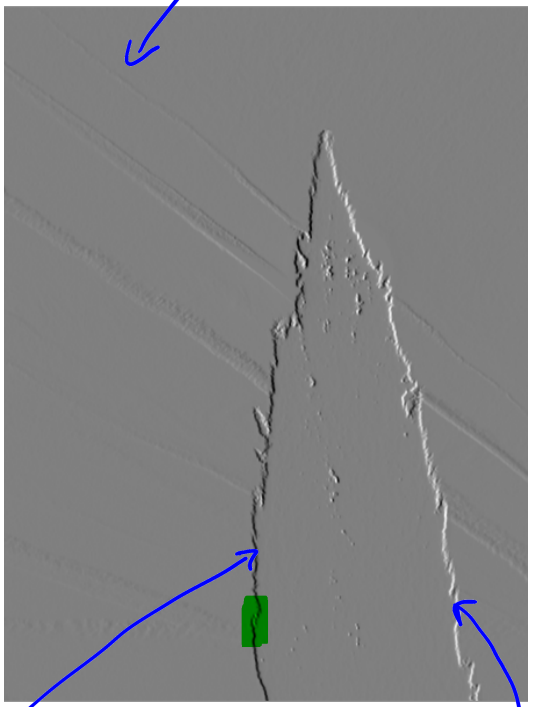
also possible for Prewitt edge detector

Filters $\boldsymbol{H}_{180°}$, $\boldsymbol{H}_{225°}$, $\boldsymbol{H}_{270°}$, $\boldsymbol{H}_{315°}$ allow to distinguish between positive and negative luminance gradient (dark to bright vs. bright to dark)

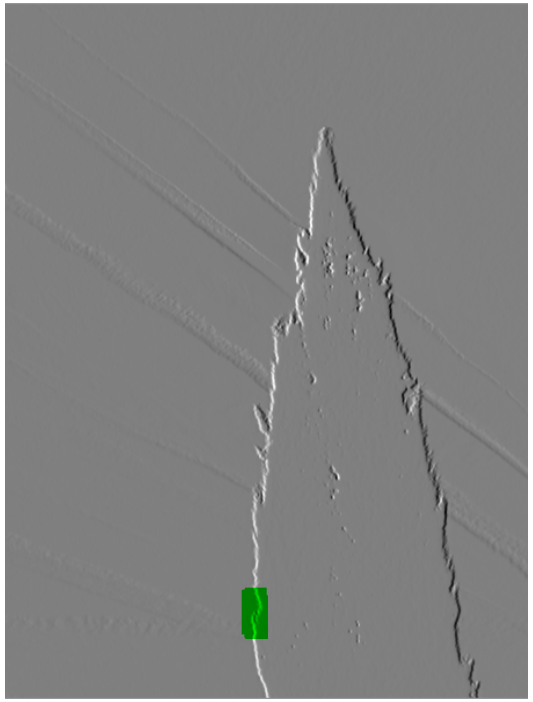


# Sobel Detector Example

Original Image



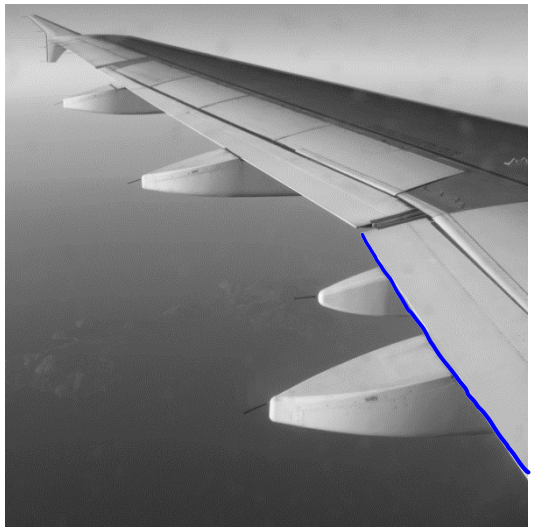

Sobel gradient for $H_{90°}$



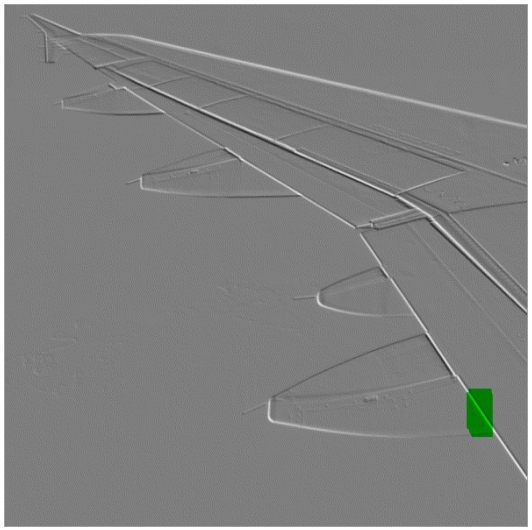

Sobel gradient for $H_{270°}$

# Sobel Detector Example

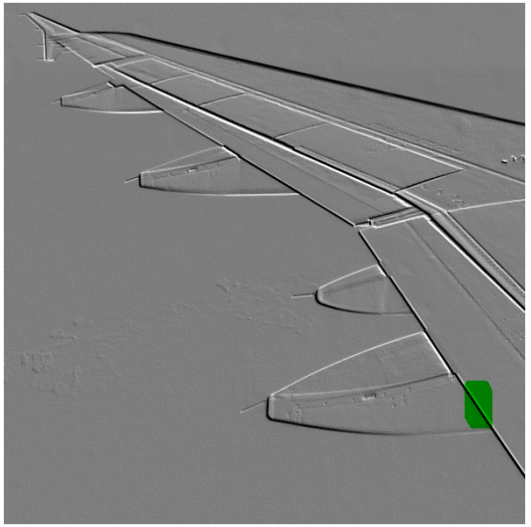
for diagonal edges:



Original Image



Sobel gradient for $H_{315°}$



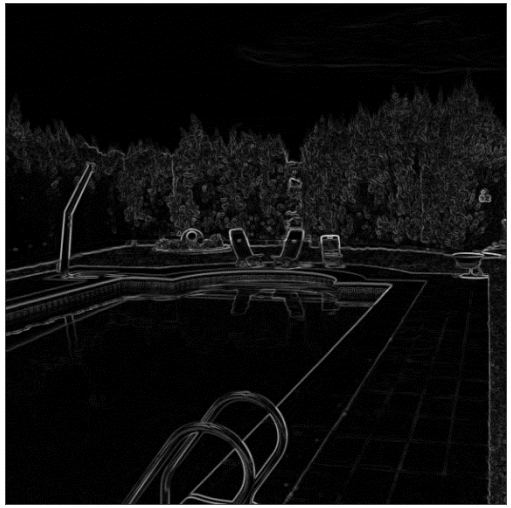
Sobel gradient for $H_{135°}$

# Edge Detection

Given gradient map (Prewitt, Sobel, Roberts, etc.) edges are obtained by **thresholding**

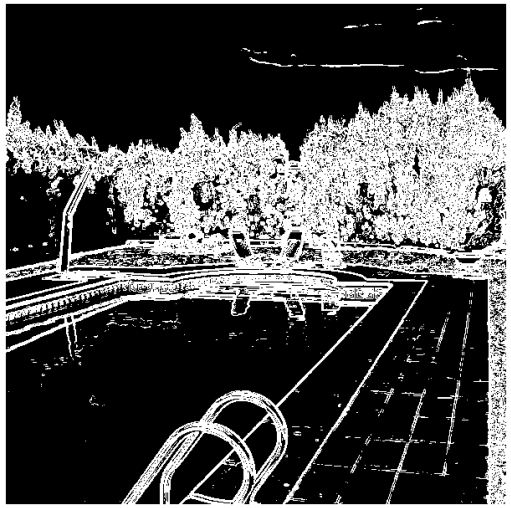
Example:

Original image



Sobel gradient

$|S * H_v|$



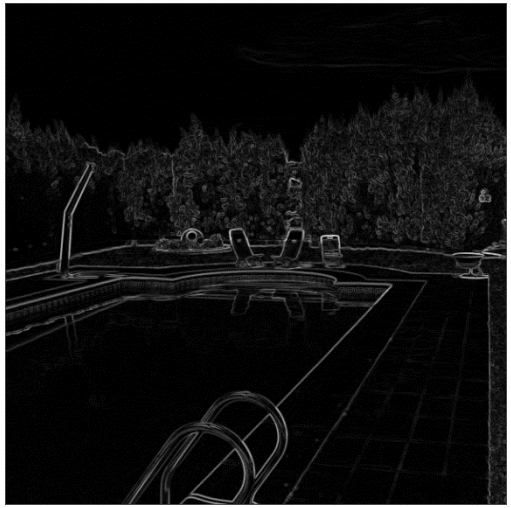
Gradient thresholding (low threshold)

# Edge Detection

Given gradient map (Prewitt, Sobel, Roberts, etc.) edges are obtained by **thresholding**

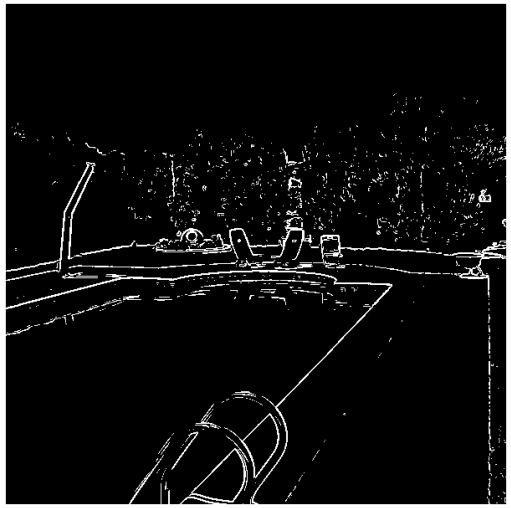
Example:

Original image



Sobel gradient

$|S * H_v|$



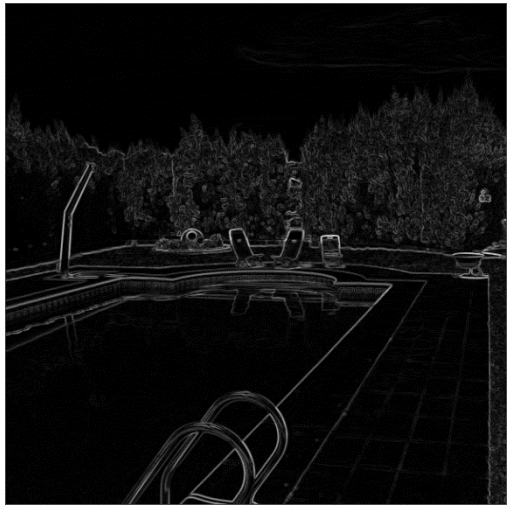
Gradient thresholding (medium threshold)

# Edge Detection

Given gradient map (Prewitt, Sobel, Roberts, etc.) edges are obtained by **thresholding**

Example:

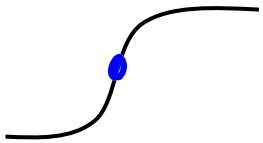
Original image



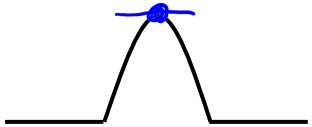
Sobel gradient

Gradient thresholding (high threshold)

# Edge Detection by 2nd Order Derivative

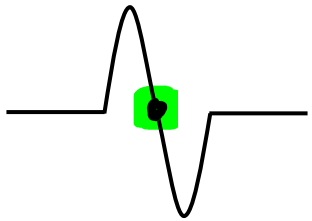


Edge profile $s(x)$



$s'(x)$

1st order



$s''(x)$

2nd order

**Laplacian operator:** sum of all (unmixed) 2nd order partial derivative

- Divergence of gradient

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

Isotropic (rotationally invariant) operator

**Zero-crossings** mark edge location

Pierre-Simon
Laplace (1749-1827)

# Laplacian Operator

**Discrete approximation:** gradient of gradient

$$\boldsymbol{L}_{\mathrm{v}} = \boldsymbol{G}_{\mathrm{v}} * \boldsymbol{G}_{\mathrm{v}} = [1 \quad -2 \quad 1]$$

← 2nd order derivative

$$\boldsymbol{L}_{\mathrm{h}} = \boldsymbol{G}_{\mathrm{h}} * \boldsymbol{G}_{\mathrm{h}} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

**2D Laplacian operator** can be derived by adding results of horizontal and vertical filters, i.e.

$$\boldsymbol{L}_4 = \boldsymbol{L}_{\mathrm{v}} + \boldsymbol{L}_{\mathrm{h}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

# Laplacian Operator

**Non-separable** 2D Laplacian operator using 8 neighbors

$$\boldsymbol{L}_8 = \boldsymbol{L}_\mathrm{v} + \boldsymbol{L}_\mathrm{h} + \begin{bmatrix} 1 & & 1 \\ & -4 & \\ 1 & & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

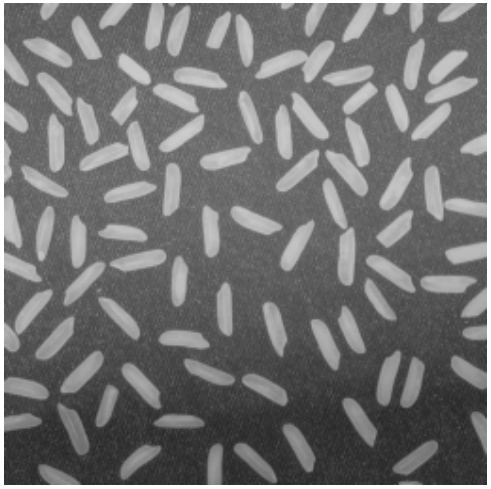
$$\begin{bmatrix} 1 & & 1 \\ & -2 & \\ 1 & & 1 \end{bmatrix} + \begin{bmatrix} & -2 & \\ 1 & & 1 \\ & -2 & \end{bmatrix}$$

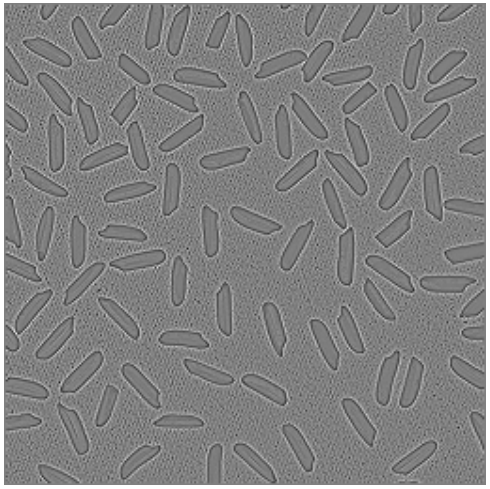
**Separable** 2D Laplacian operator

$$\boldsymbol{L}_\mathrm{s} = \boldsymbol{L}_\mathrm{v} * \boldsymbol{L}_\mathrm{h} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

# Laplacian Operator Example

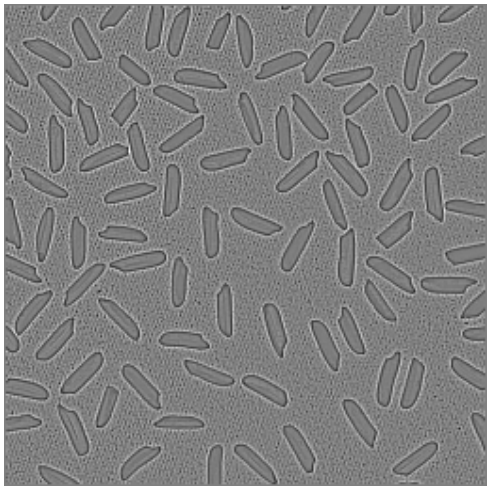
Original

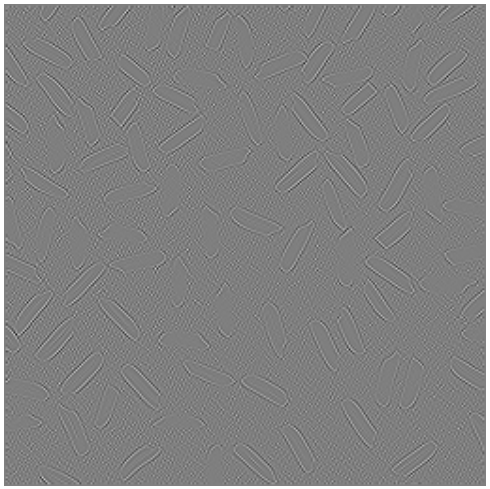




2D Laplacian

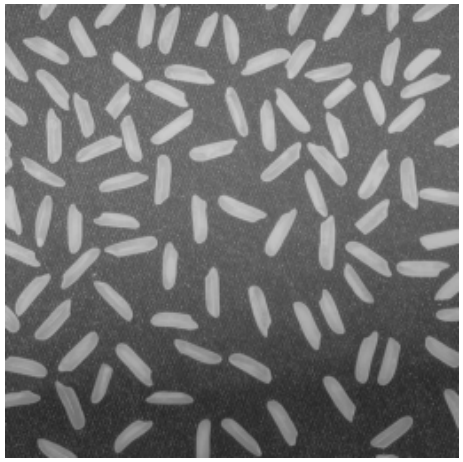
Non-separable
2D Laplacian



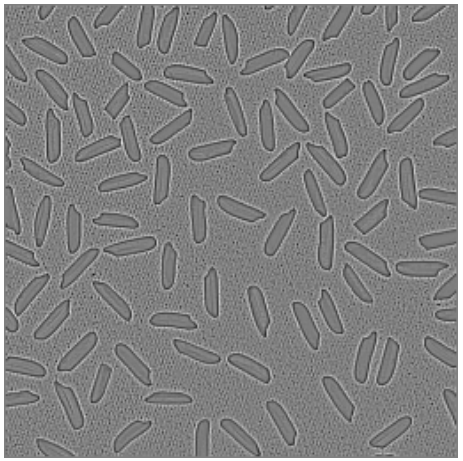


Separable 2D
Laplacian

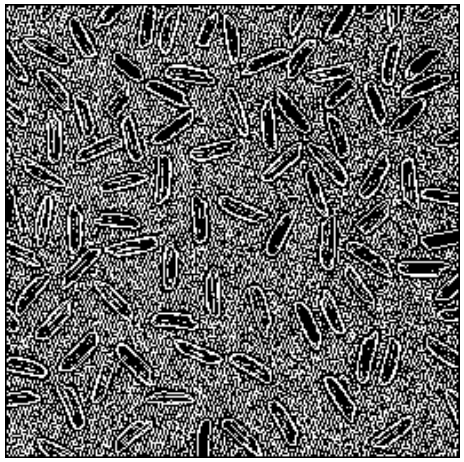
# Laplacian Operator Example



Original



2D non-separable Laplacian

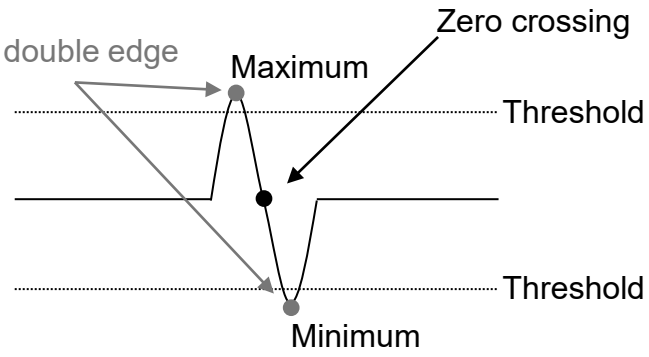


Zero-crossings

# Laplacian Operator

Zero-crossing responds equally to strong and weak edges

- Suppress zero-crossings with low magnitude




- Sensitive to very fine details and noise → blur image first

# Laplacian of Gaussian

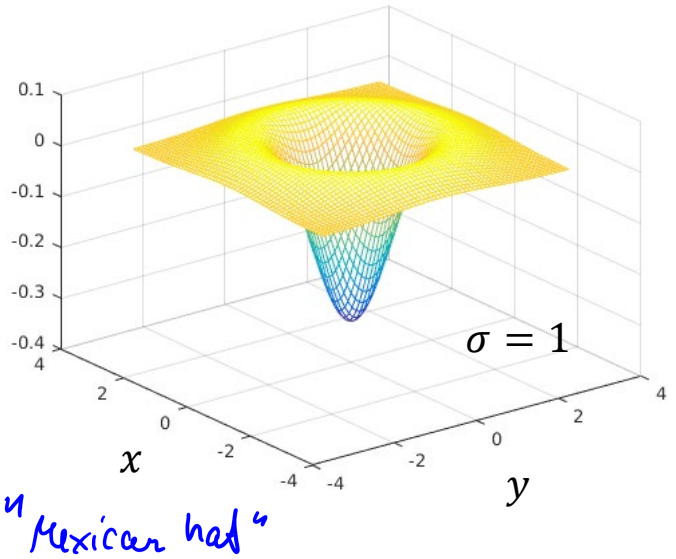
Blur first with Gaussian filter, then apply Laplacian operator:

$$\nabla^2(h_G[x,y] * s[x,y]) = \left(\nabla^2 h_G[x,y]\right) * s[x,y] = \mathrm{LoG}[x,y] * s[x,y]$$

Laplacian of Gaussian

## Laplacian of Gaussian (LoG)

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

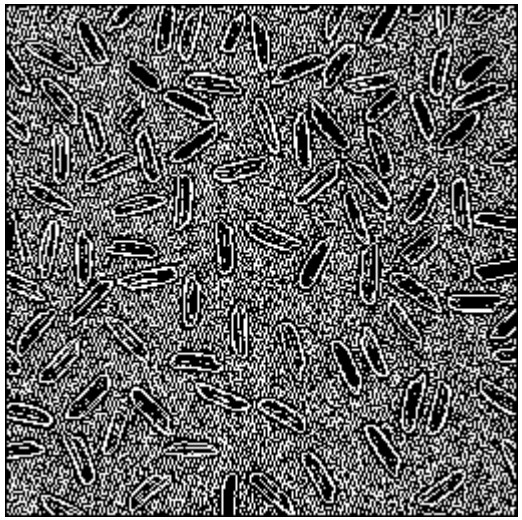


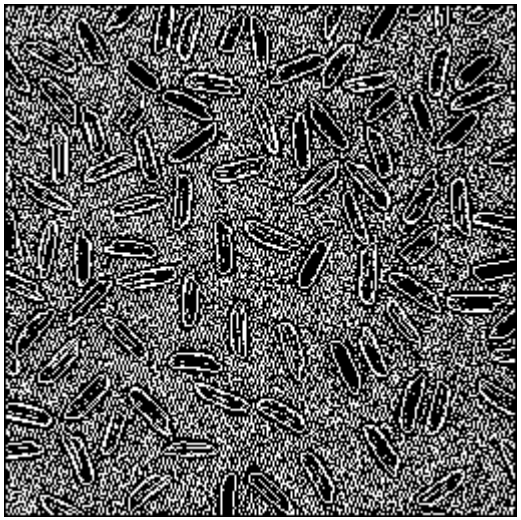

"Mexican hat"

Parameter $\sigma$ controls width of the Gaussian kernel (blur strength)

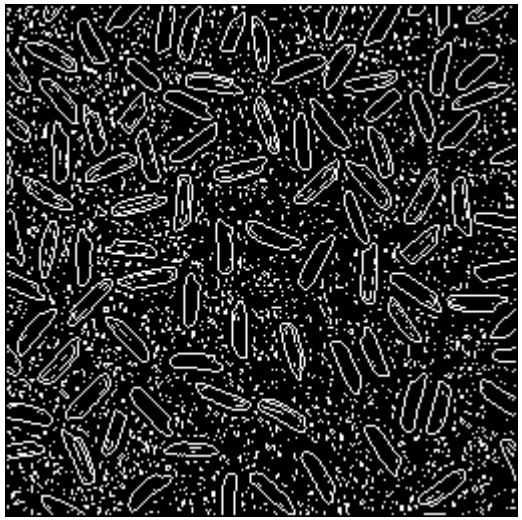
# Zero Crossings of LoG

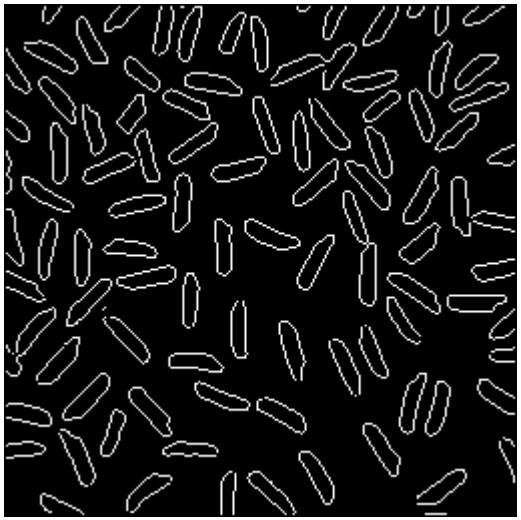
Without Gaussian





$\sigma = 0.5$





noise removal

$\sigma = 1$

$\sigma = 2$

# Zero Crossings of LoG

Without Gaussian

$\sigma = 0.5$

$\sigma = 1$

$\sigma = 2$

# Canny Edge Detector

**Objectives:**

- Insensitive to noise (no false edges)
- Accurate localization

**Steps:**

1. Smooth image with Gaussian filter ($\sigma$)
2. Compute gradient magnitude and angle (Sobel, Prewitt, …)
3. Apply non-maximum suppression
4. Double thresholding for potential edges
5. Track edges by hysteresis

$g = |\nabla f| = \sqrt{g_x^2 + g_y^2}$

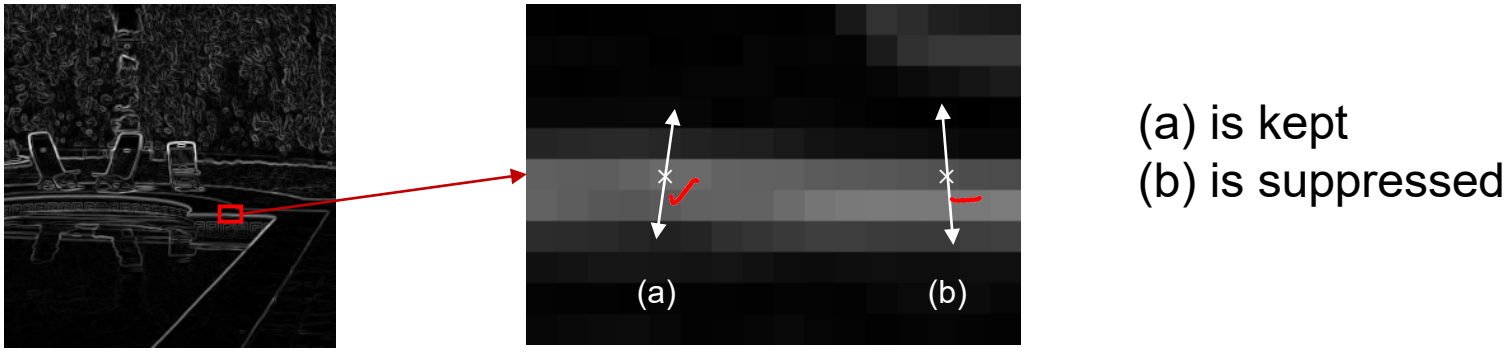
$\theta = \arctan \frac{g_y}{g_x}$

# "3." Non-maximum Suppression

Edge thinning technique

For each gradient point (pixel)

- Check the two neighbors along edge normal $(\theta, \theta + 180°)$
- Keep it if larger than either of its neighbors, otherwise suppress




(a) is kept
(b) is suppressed

In practice, edge normals are quantized to four directions: horizontal $(90°)$, vertical $(0°)$ and the two diagonals $(\pm 45°)$

# "4." Double Thresholding & Hysteresis "5."

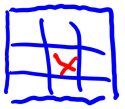
Using a single threshold can leave too much noise or filter out weak but important edges

**Double thresholding:** two thresholds $T_{\text{high}} > T_{\text{low}}$

- If $g > T_{\text{high}}$ $\rightarrow$ edge — significant gradient
- If $g < T_{\text{low}}$ $\rightarrow$ no edge — noise
- If $T_{\text{low}} < g < T_{\text{high}}$ $\rightarrow$ potential edge — => hysteresis step "5."
- Typical setting: $T_{\text{high}}/T_{\text{low}} = 2 \ldots 3$

**Hysteresis**

- Potential edge is regarded as edge if it is 8-connected to a strong edge
- Filters out isolated edges (due to noise)
- Avoids broken edges



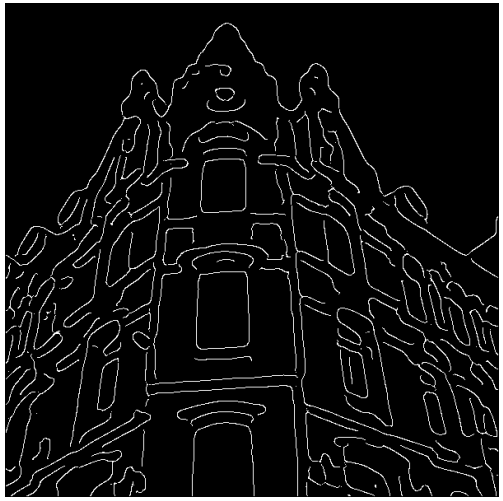
# Canny Detector Example

Original

$\sigma = 1$

of Gaussian filter

$\sigma = 3$



$\sigma = 5$

# Canny Detector Example

Original

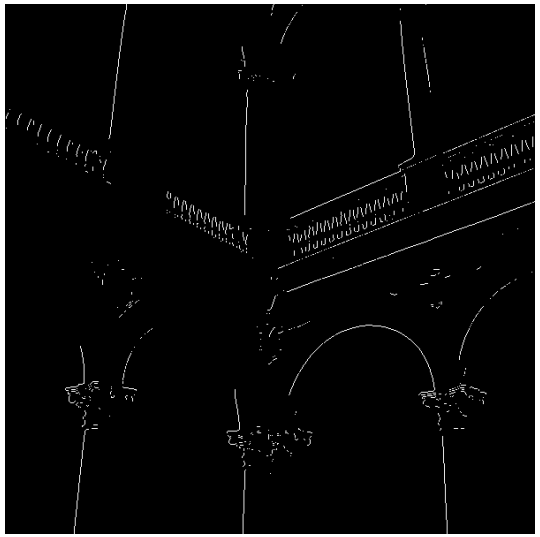
$\sigma = 1$

$\sigma = 3$

$\sigma = 5$

# Canny Detector Example

Original



Sobel

$T = 0.25$

on magnitude g

Canny

$T_{\text{high}} = 0.25$

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

# Canny Detector Example

Original

Sobel

$T = 0.15$

Canny

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

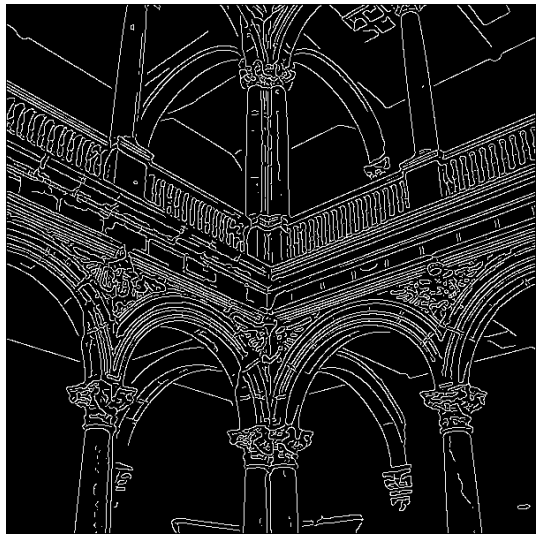
$T_{\text{low}} = 0.4T_{\text{high}}$

# Canny Detector Example

Original

Sobel

$T = 0.05$



Canny

$T_{\text{high}} = 0.05$

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

# 6.3 Hough Transform

Feature extraction technique

- Generalized template matching
- Transforms shapes into points, specifically lines (also circles etc.)

Paul V.C. Hough
(patented in 1962)

Consider detection of straight lines $y = mx + c$

slope, offset

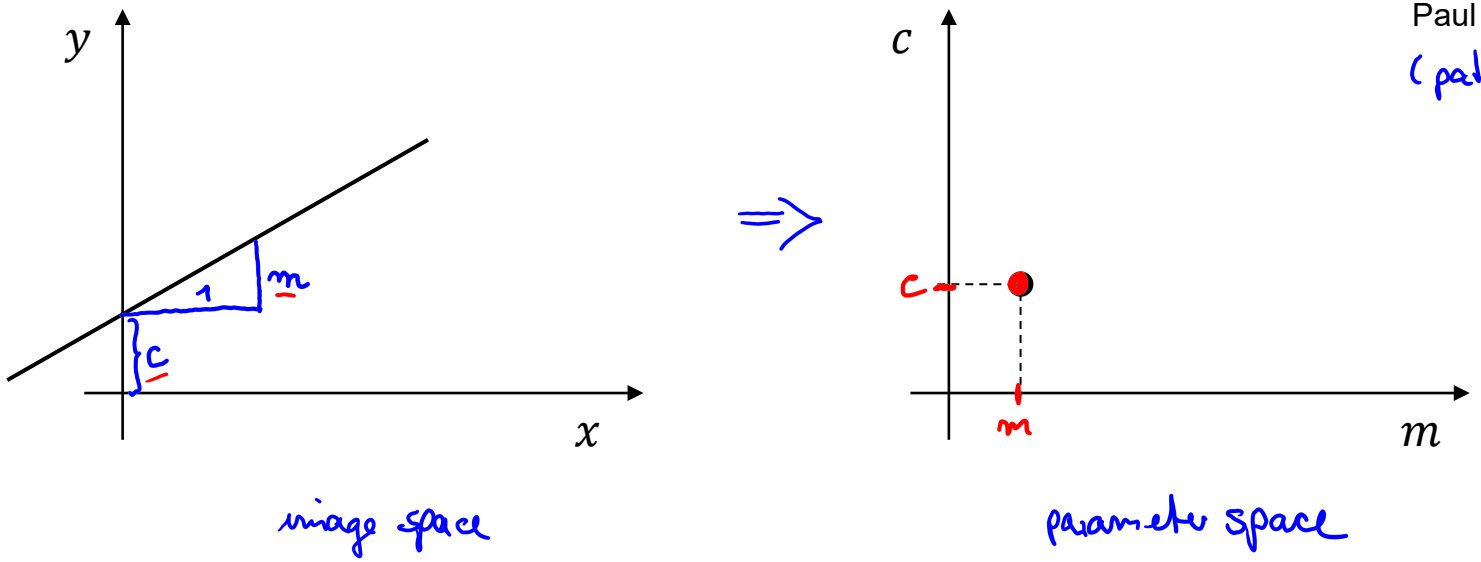


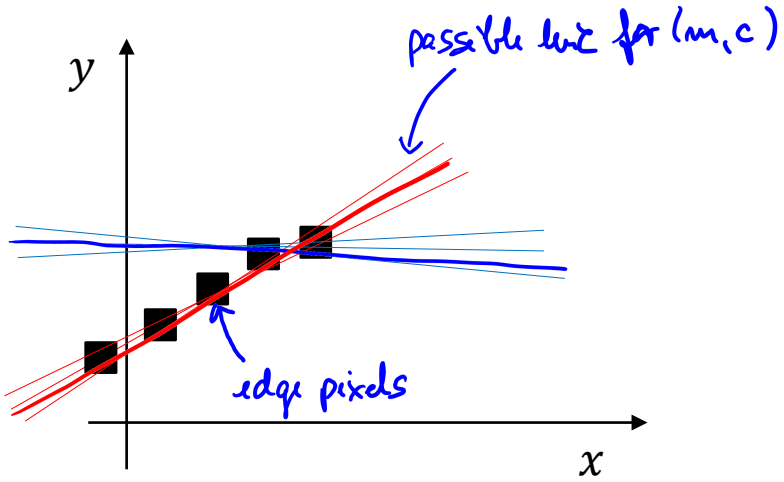

image space

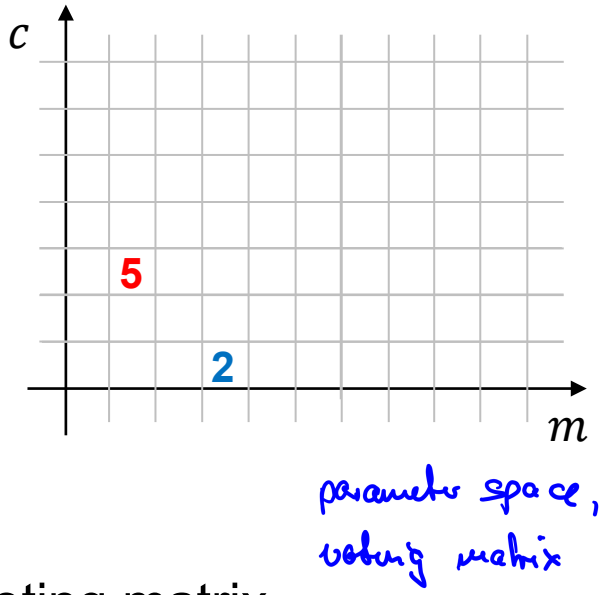
parameter space

# Hough Transform

Hough transform of an edge map is calculated by **voting** mechanism

- Each edge pixel votes for a line that passes through it
- Detect peak(s)




**Accumulator:** discrete $(m,c)$-plane, voting matrix

- All bins initialized by zeros

# Hough Transform

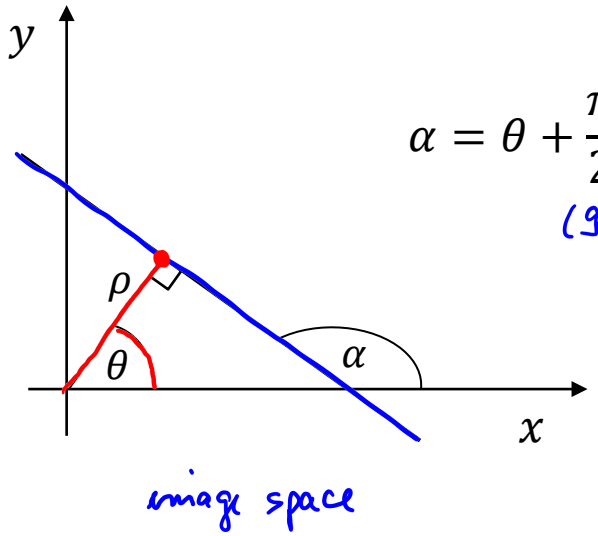
**Problem:** vertical edges have infinite slope ($m \to \infty$)

**Solution:** Hesse normal form

$$y = mx + c = \frac{\sin\alpha}{\cos\alpha}x + c \qquad (\tan\alpha)$$

$$\alpha = \theta + \frac{\pi}{2} \quad (90°)$$

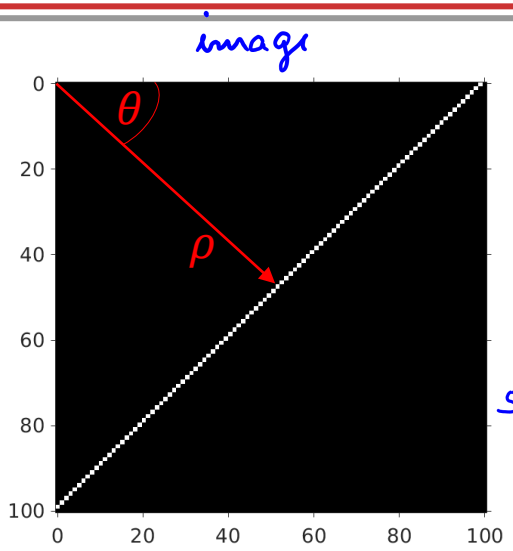
$$x\cos\theta + y\sin\theta = \rho$$

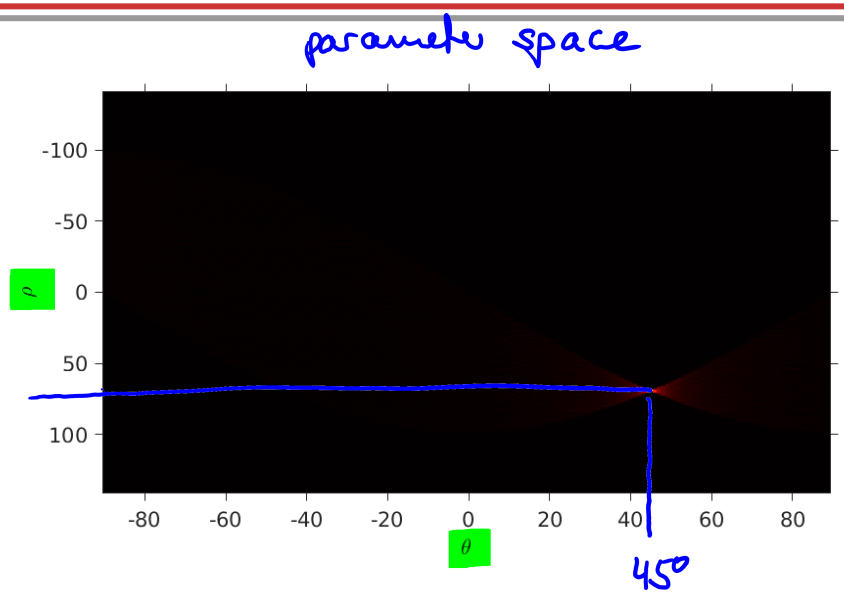



Ludwig Otto Hesse (1811-1874)

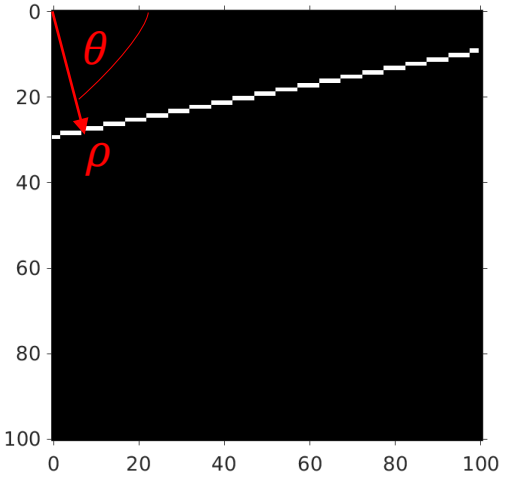
Hough transform is computed for $(\rho,\theta)$-plane

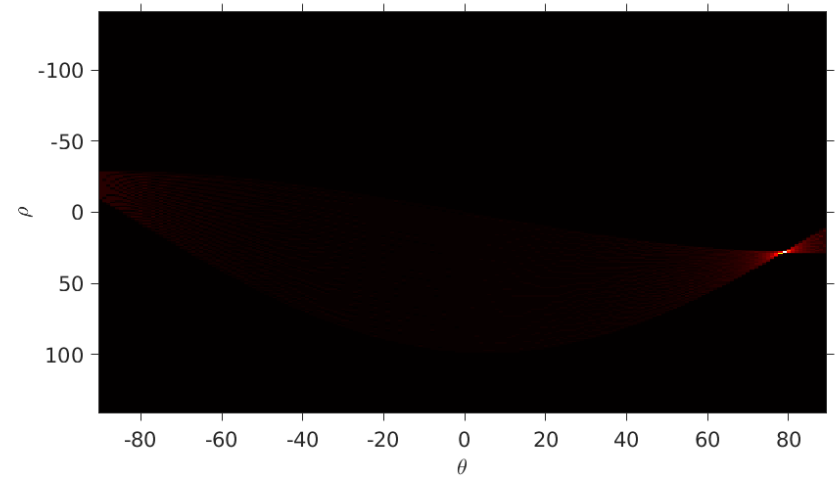
# Hough Transform Example



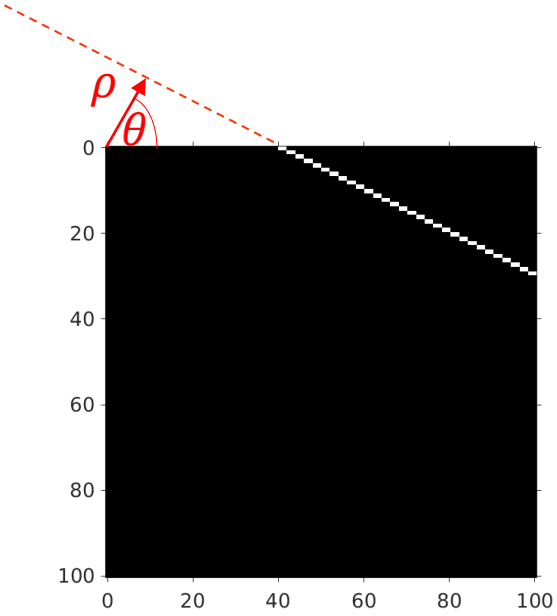



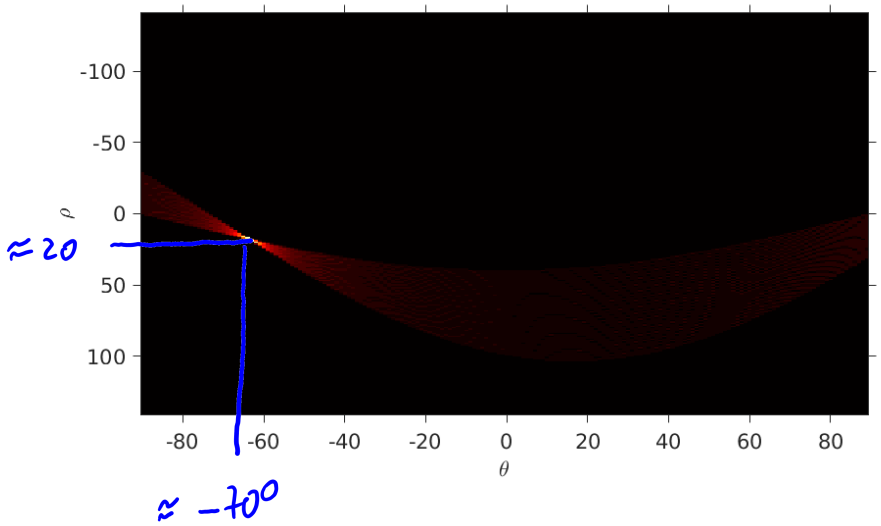

50·T₂ ≈ 70



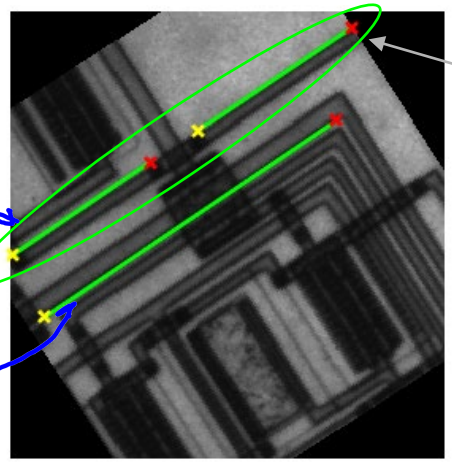



# Hough Transform Example



# Hough Transform Example

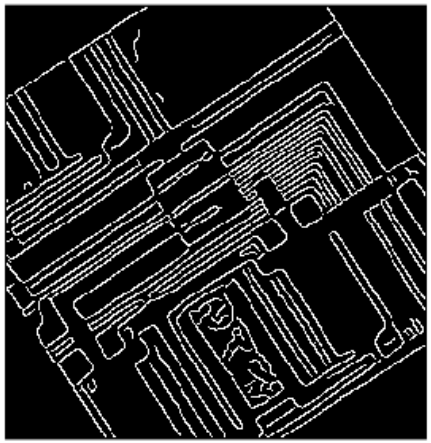



Original image

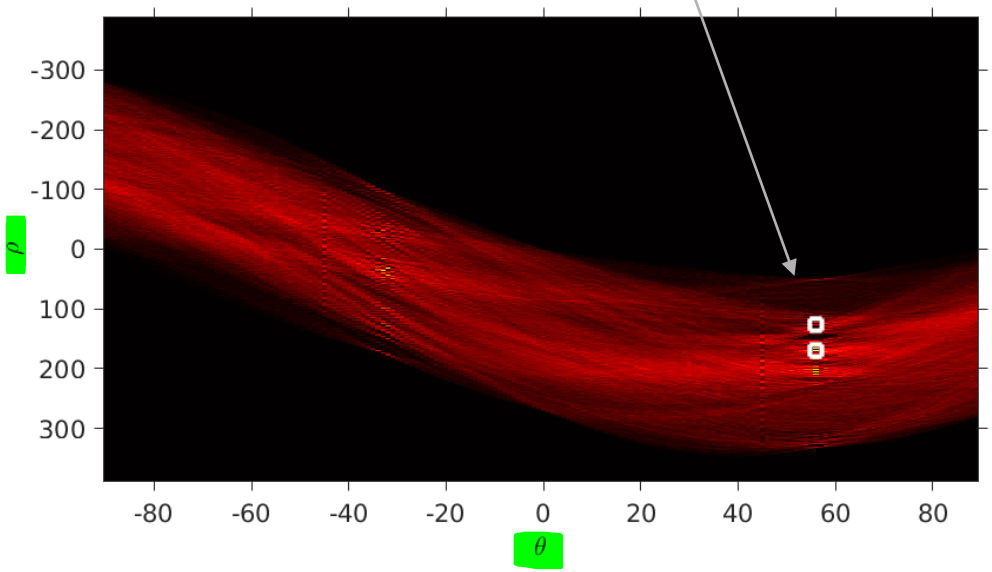
These two edges vote for the same line

Two dominant line structures detected

two lines (a) and (b)



Edge map



# Generalized Hough Transform

Can be applied to detect any shape that can be **parametrized**

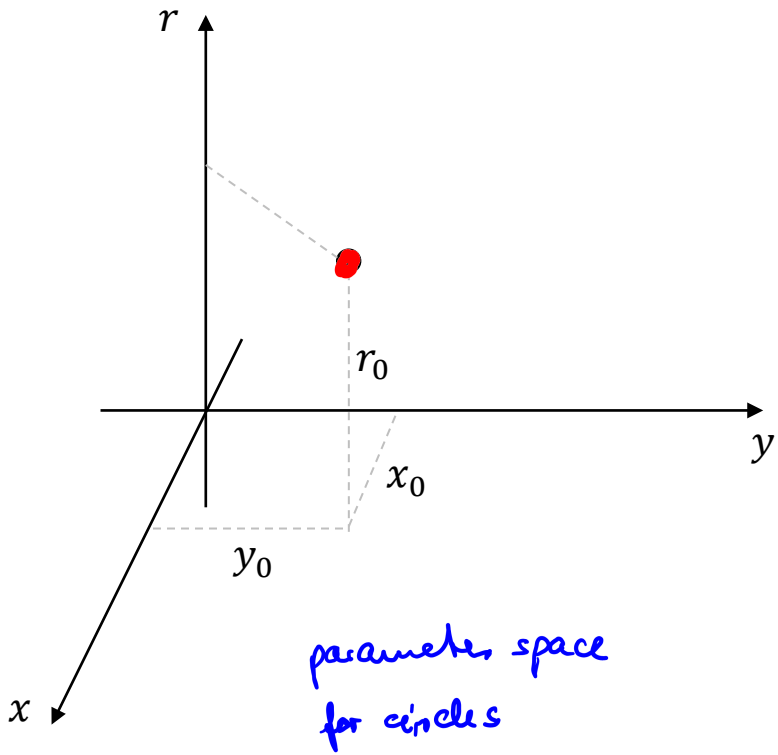
**Example:** circles

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

parameterized description of circle

3D Hough space (3 parameters)

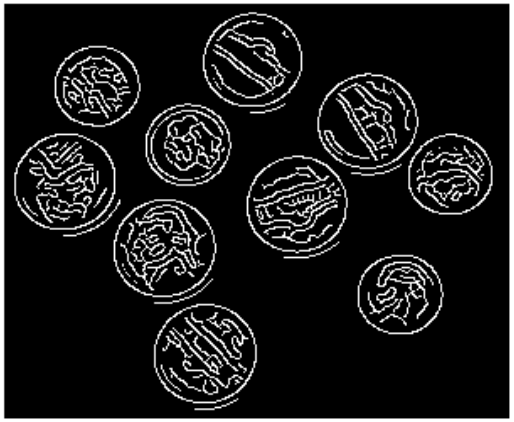
- Center coordinates $(x_0, y_0)$
- Radius $r$



# Circle Detection by Hough Transform

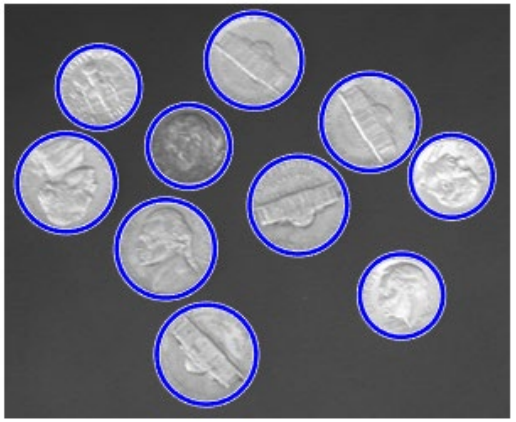
Original image

Detected edges



after Canny edge detection

Detected circles



after maximum detection in Hough parameter space

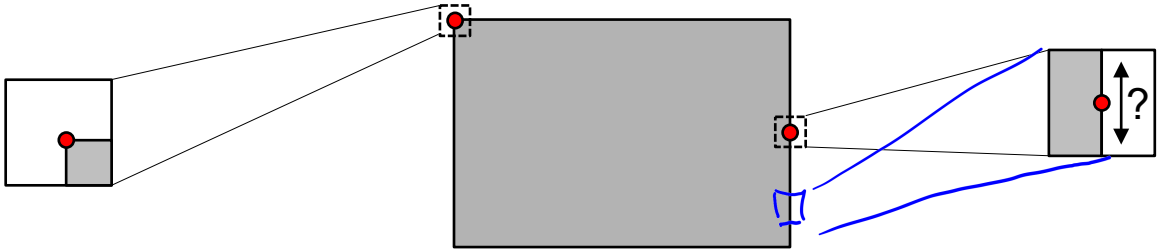
=> also for ellipse (e.g. rice grains)

# 6.4 Keypoint Detection

Many applications benefit from features localized at $(x, y)$

- Image alignment, image stitching, object recognition

Edges well localized only in one direction → detect **corners**?



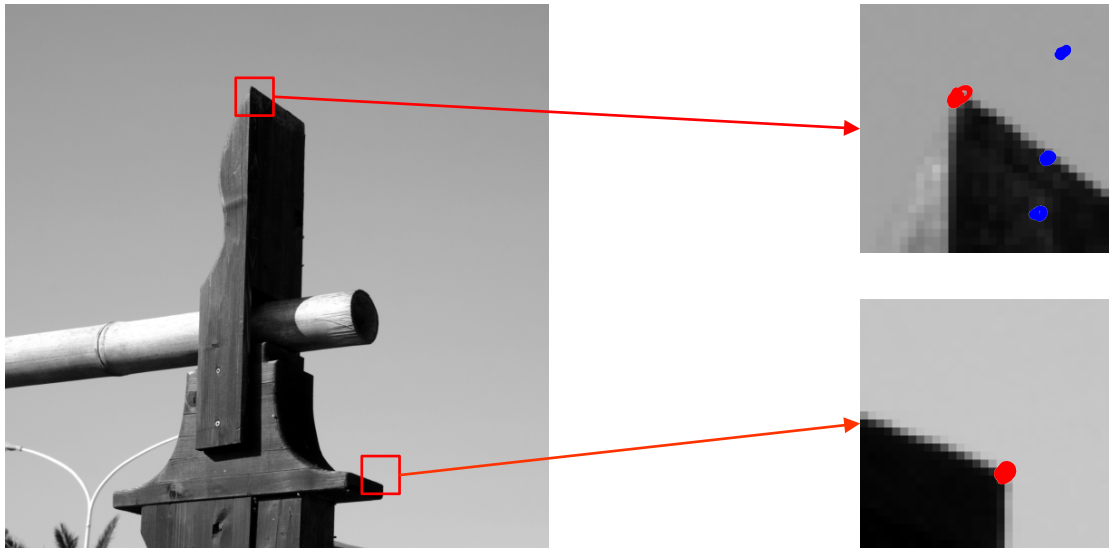

**Desirable properties** of keypoint detector

- Accurate localization
- Invariance against shift, rotation, scale, brightness changes
- Robust against noise, high repeatability

# What Are Corners?

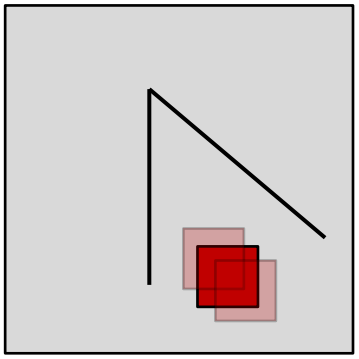
Good (stable) keypoints

- Junction of edges (at least two or more)
- Exhibit large image variations in all directions

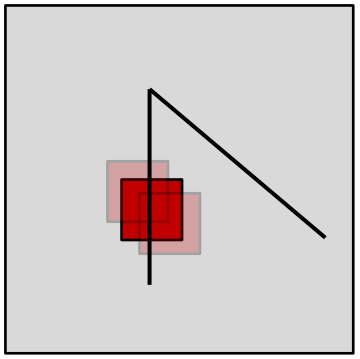


# Harris Corner Detector

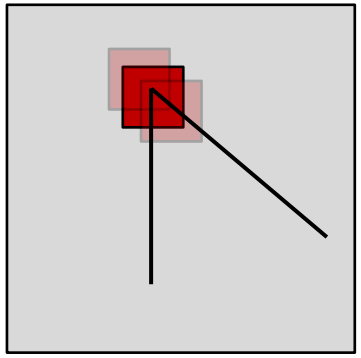
Intuitive approach (Chris Harris, 1998):



Flat region: no change in all directions



Edge: no change along edge direction



Corner: significant change in all directions

⇨ Choose patch, move into various directions, measure changes

Harris detector is a mathematical tool to determine which case holds.

# Harris Detector

Sum of squared differences for shift $(u, v)$:

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

Weighting (local) — Shifted mage — Image

First order Taylor approximation:

shift at $(x,y)$

$$s[x+u, y+v] \approx s[x,y] + us_x[x,y] + vs_y[x,y] \quad (*)$$

Local derivatives at $(x,y)$

Therefore:

$$g[u,v] \overset{(*)}{\approx} \sum_{x,y} w[x,y]\left(us_x[x,y] + vs_y[x,y]\right)^2$$

# Harris Detector

In matricial notation: $g[u,v] \approx [u \quad v]\boldsymbol{M}\begin{bmatrix} u \\ v \end{bmatrix}$

**Harris matrix $\boldsymbol{M}$** is a 2×2 structure matrix (second-moment matrix)

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

Weighted window $w$, usually Gaussian or binary

- Isotropic response

# Harris Detector

Detection based on eigenvalues of $\boldsymbol{M}$

- $\lambda_1, \lambda_2$

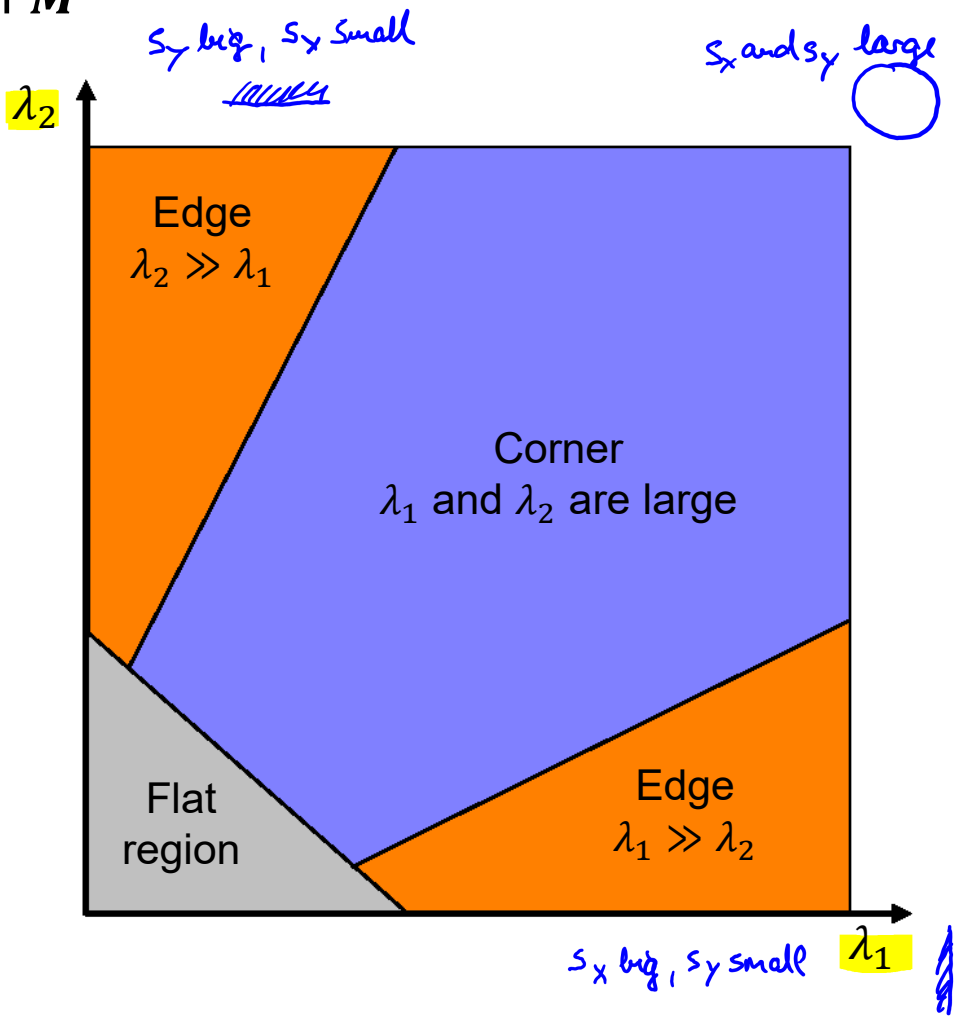
$\hat{=}$ ellipses representing range of $s_x$ and $s_y$

Computation of eigenvalues

- Expensive

Measure of "cornerness" instead

$$c[x,y] = \det(\boldsymbol{M}) - \kappa\left(\mathrm{Tr}\,(\boldsymbol{M})\right)^2$$
$$= \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

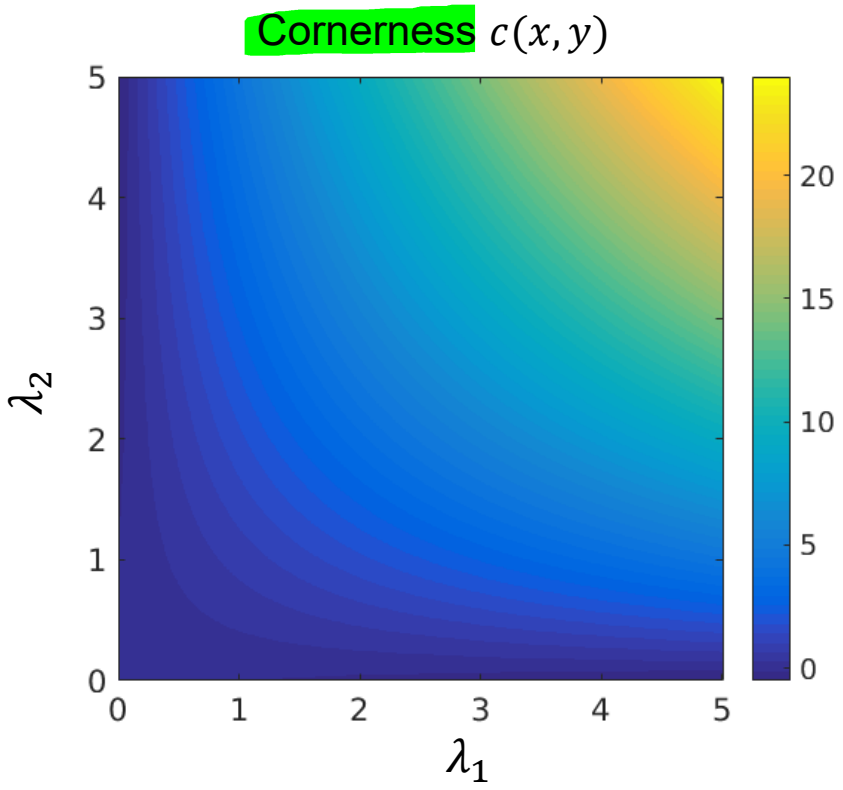


# Harris Cornerness

Higher value indicates “better” corner

Parameter $\kappa$ is set empirically

- Typically around 0.1



# Harris Corner Detection Algorithm

1. Compute local $x$ and $y$ derivatives of image, e.g. by

   $$s_x = s[x,y] - s[x-1,y] \text{ and } s_y = s[x,y] - s[x,y-1]$$

2. Compute products of local derivatives at every pixel

   $$s_x^2 = s_x s_x \quad s_{xy} = s_x s_y \quad s_y^2 = s_y s_y$$

3. Compute sums of weighted products of derivatives at each pixel

   $$\sum_{x,y} w[x,y] s_x^2[x,y] \quad \sum_{x,y} w[x,y] s_{xy}[x,y] \quad \sum_{x,y} w[x,y] s_y^2[x,y]$$

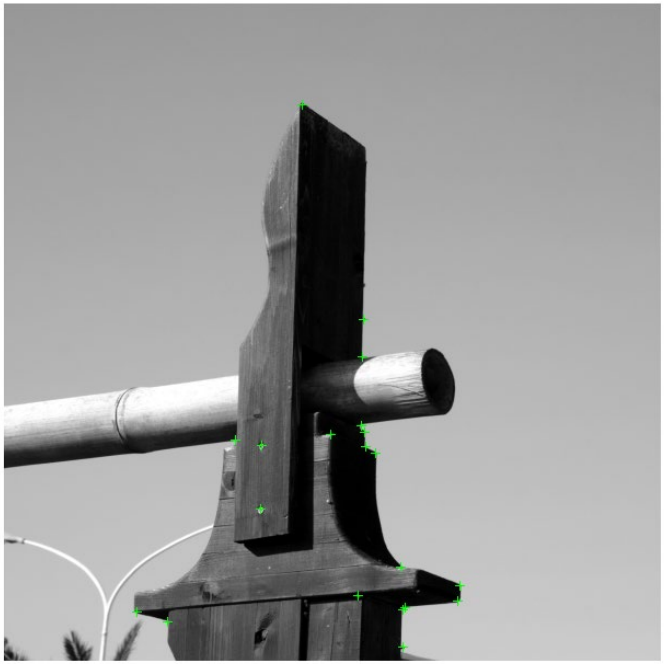
4. Define at each pixel $(x,y)$ Harris structure matrix $\boldsymbol{M}[x,y]$

5. Compute cornerness $c[x,y]$ at each pixel $(x,y)$

6. Threshold on value of cornerness and apply nonmax suppression

# Harris Detector Example

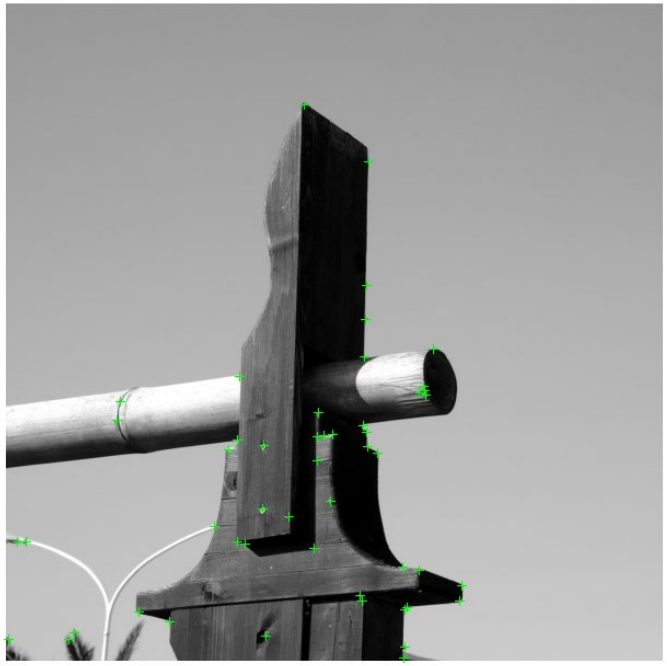
Top 20 strongest corners



Top 20 strongest corners

# Harris Detector Example

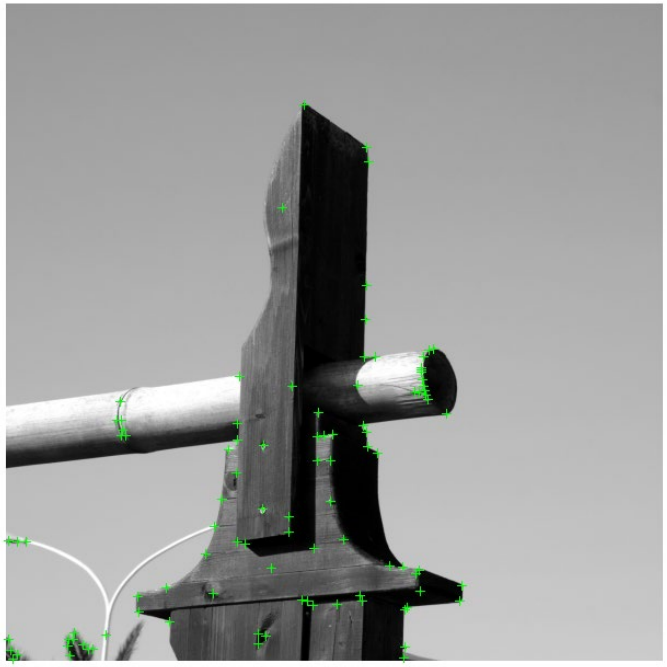
Top 50 strongest corners



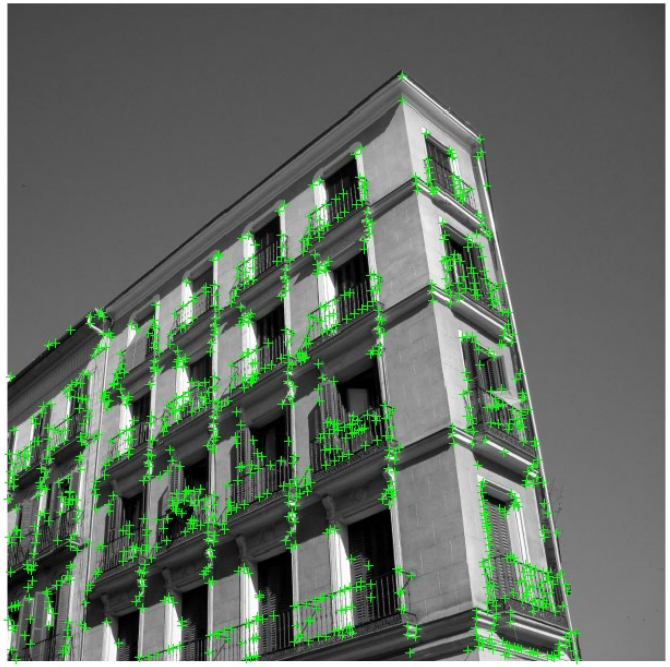
Top 50 strongest corners

# Harris Detector Example

All detected corners

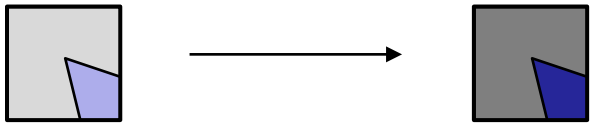


All detected corners

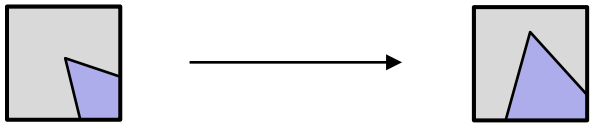


# Robustness of Harris Detector

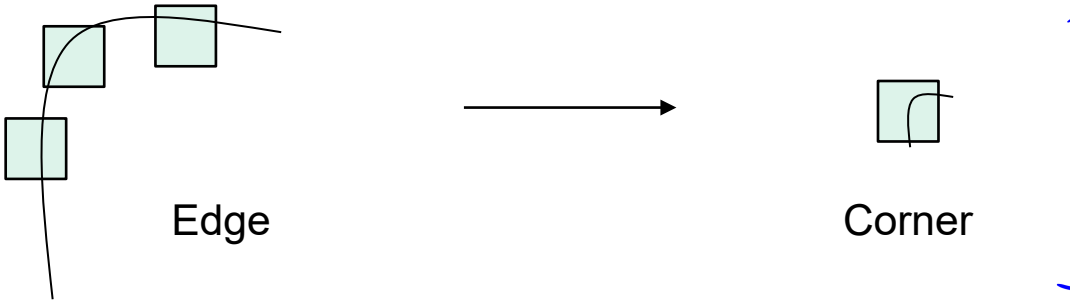
Invariant to brightness offset



Invariant to shift and rotation



Not invariant to scaling

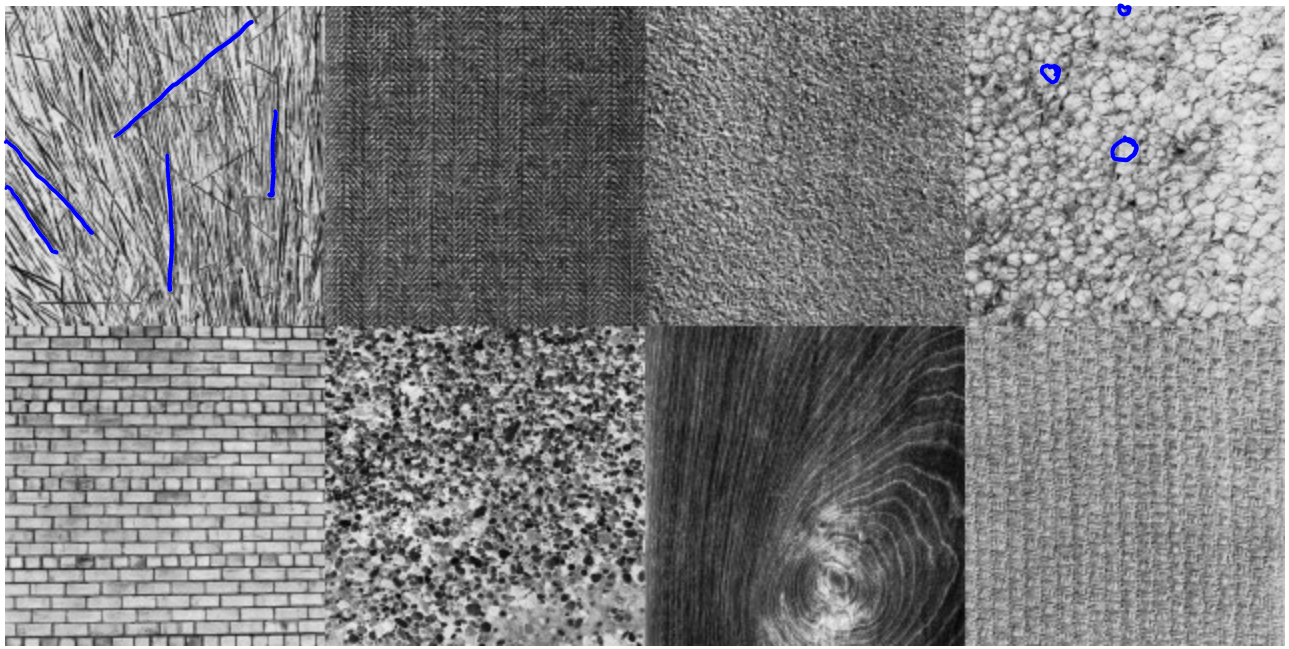



remedy: scale space features
=> chapt. 7

# 6.5 Texture

Structural patterns of surfaces of objects, visible as variation of basically luminance intensity in an image.

**Example:** wood, grain, sand, grass, cloth (taken from Brodatz's texture album)



# Classification of Textures

1. **Regular / periodic** textures:
   - sometimes also called structural textures
   - nearly periodic structures
   - analysis by spectral methods, use e.g. 2D DFT (or DCT) from Chap. 4

2. **Irregular** textures:
   - texture is ensemble of basis texture elements (**texels**) which are different in size and structure
   - change in size and structure may be random or deterministic
   - analysis by statistical methods, use e.g. AKF, PDS, or Co-occurrence C

# Co-occurrence Matrix

Texture descriptors

- Estimate the pairwise statistics of pixel intensity

Each element $(i, j)$ of co-occurrence matrix $\boldsymbol{C}$ represents an estimate of the probability that two pixels with a specified separation (distance) have grey levels $i$ and $j$

- Separation is specified by horizontal and vertical shift $(k, l)$

**Properties** of $\boldsymbol{C}$:

- It is a square $J \times J$ matrix ( $J$ is number of grey levels) => typically $J = 256$
- Not symmetric (doesn't have to be)
- Usually normalized by number of pixels to obtain probabilities rather than counts

$$\boldsymbol{C}' = \frac{1}{MN}\boldsymbol{C}$$

$\underline{C}'$ ~ probability ; $\underline{C}$ ~ count

# Co-occurrence Matrix

**Example:** $J = 3$ (0, 1, 2 are amplitudes)

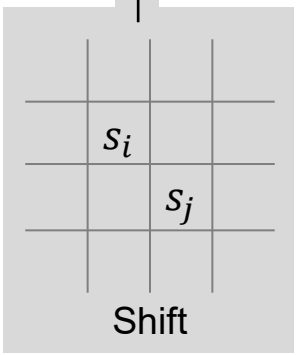
$$s[n_1, n_2] = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Amplitude $s_i$

$$C_{1,1} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 2 \\ 0 & 2 & 0 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \end{matrix}$$

Amplitude $s_j$

↑ total # occurrences $= 2$




shift $(k, \ell) = (1, 1)$

The situation when a pixel ($s_i$) has a value of 1 and its bottom-right neighbor ($s_j$) has a value of 2 occur **twice** within the image.

# Features Derived from Co-occurrence Matrix

Co-occurrence matrices usually become quite large

- 256×256 for 8 bit resolution
- Not directly suited as texture feature

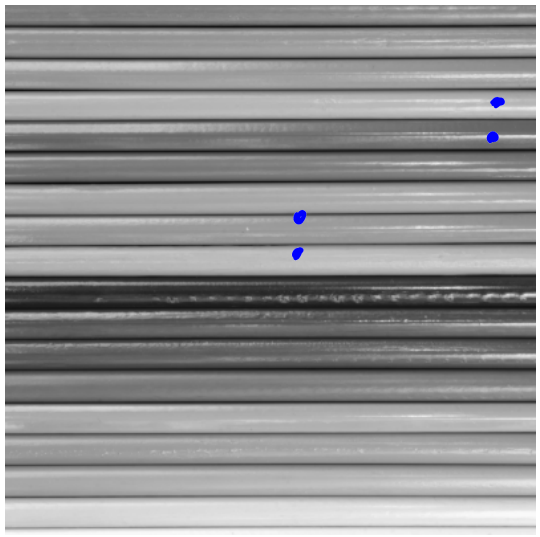
In practice features derived from $\boldsymbol{C}'$ are used, e.g.

**Energy:**
$$e_C[k,l] = \sum_{i,j} \left(C'_{k,l}[i,j]\right)^2$$
← all entries in matrix C

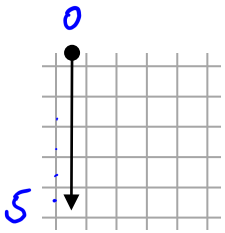
**Homogeneity:**
$$h_C[k,l] = \sum_{i,j} \frac{C'_{k,l}[i,j]}{1+|i-j|}$$
⇒ favors diagonal entries with i=j ⇒ same ampl.

**Correlation:**
$$\varphi_{ss}[k,l] = \sum_{i,j} s_i s_j C'_{k,l}[i,j]$$
↳ shift

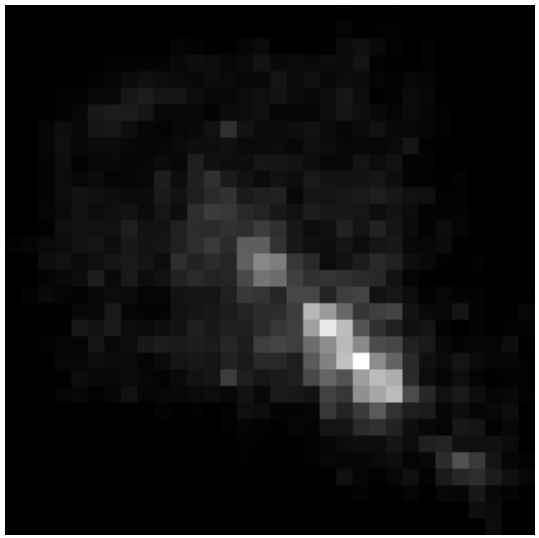
# Co-occurrence Matrix Example



Original image



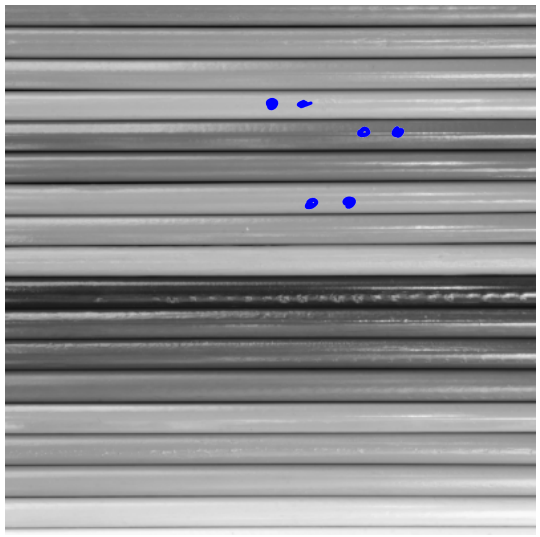

Shift



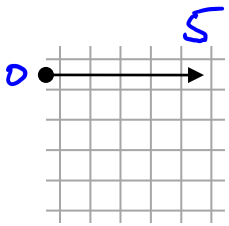
$C_{0,5}$

Homogeneity: 0.3594
Energy: 0.0044

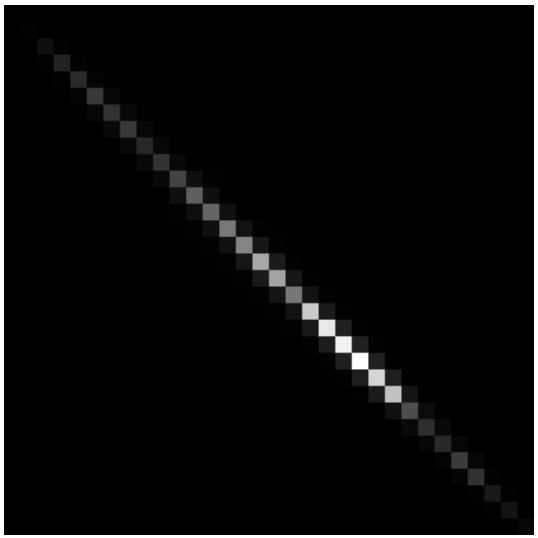
# Co-occurrence Matrix Example



Original image




Shift



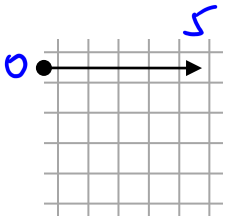
$C_{5,0}$

Homogeneity measures **spatial closeness** of $C$ to a diagonal matrix (co-occurrence of the same grey levels)

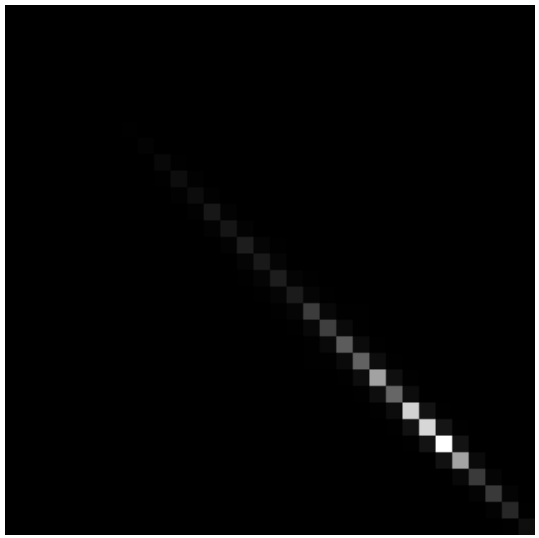
| | |
|---|---|
| Homogeneity: | 0.8781 |
| Energy: | 0.0316 |

# Co-occurrence Matrix Example

Original image




Shift



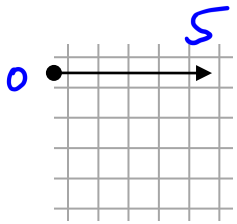
$C_{5,0}$

| | |
|---|---|
| Homogeneity: | 0.9097 |
| Energy: | 0.0562 |

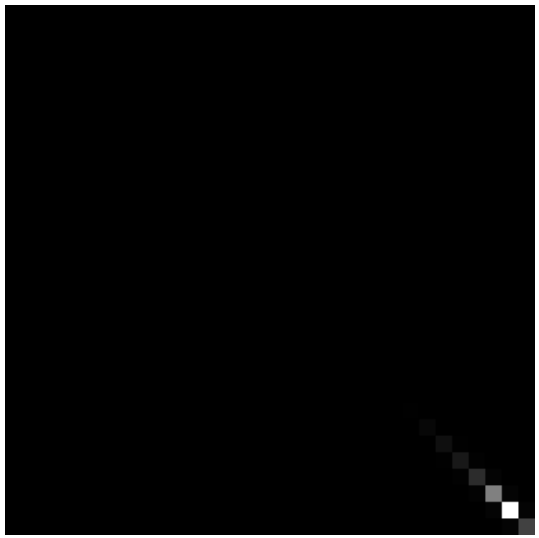
# Co-occurrence Matrix Example

Original image

very bright image




Shift



$C_{5,0}$

Energy measures **luminance uniformity** of a picture (equal to 1 for a constant image)

| | |
|---|---|
| Homogeneity: | 0.9730 |
| Energy: | 0.2727 |