Computer Vision

Paulo Dias, António Neves

deti departamento de electrónica telecomunicações e informática universidade de aveiro

Sumary

- Edges
  - Introduction
  - Edge detection
- Lines and corners
  - Line detection operators
  - Hough Transform
  - Harris corner detector
  - Other feature detectors

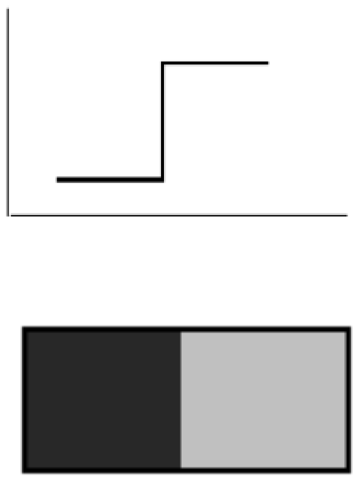
Sumary

- Edges
  - Introduction
  - Edge detection
- Lines and corners
  - Line detection operators
  - Hough Transform
  - Harris corner detector
  - Other feature detectors

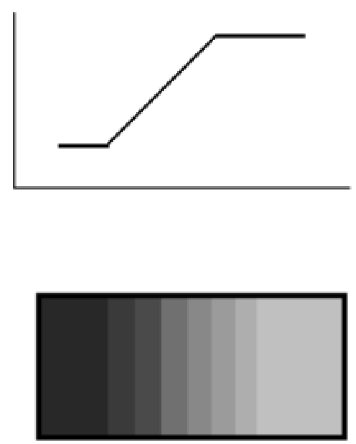
Edge Detection - Introduction

- Edges are useful to capture important events and changes in properties of the images/world

- Edge detection is difficult
  - noise
  - non ideal edges



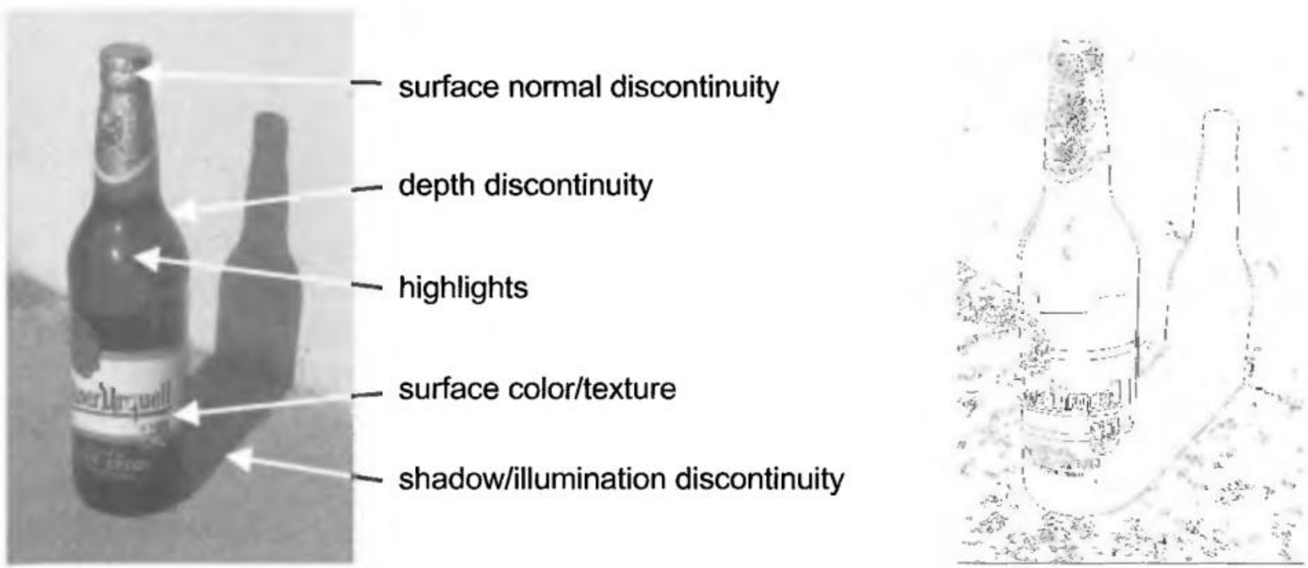
Ideal edge



non ideal edge

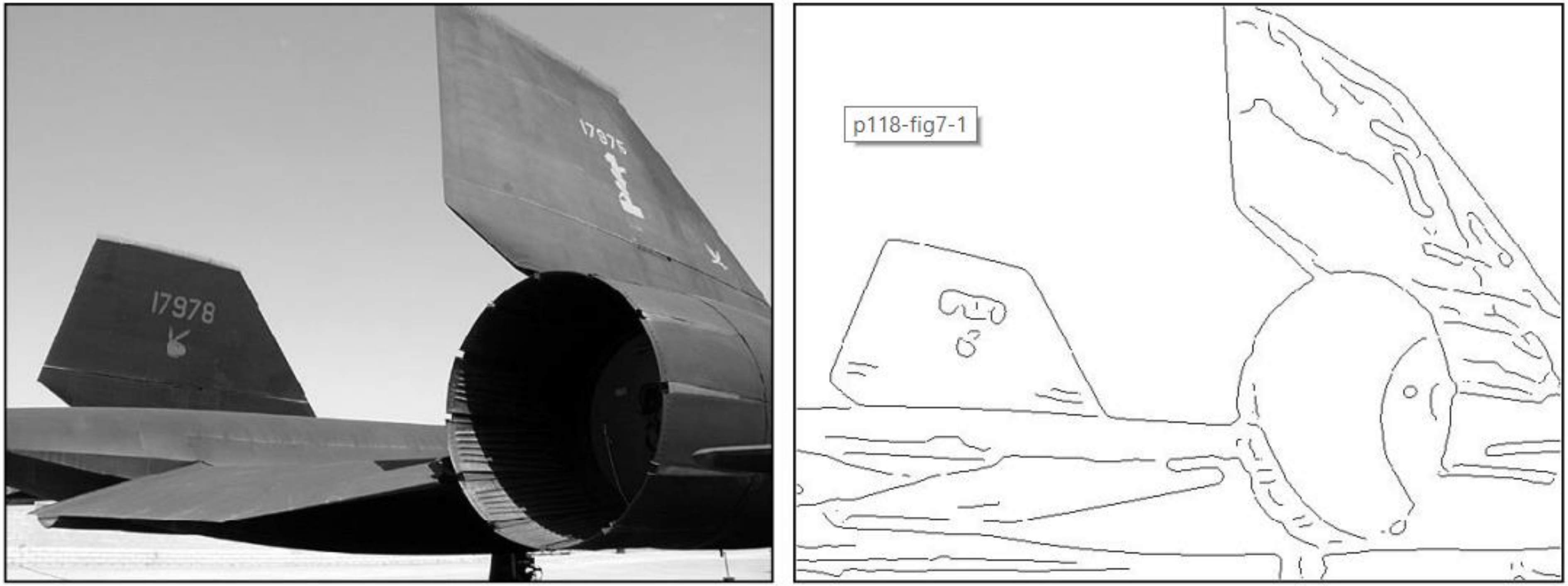
Edge Detection - Introduction

- Edges correspond to
  - discontinuities in depth,
  - discontinuities in surface orientation,
  - changes in material properties,
  - variations in scene illumination.
- How to detect the relevant edges?



Adapted from Image Processing, Analysis and Machine Vision, 3Ed, Sonka et al.

Edge Detection - example



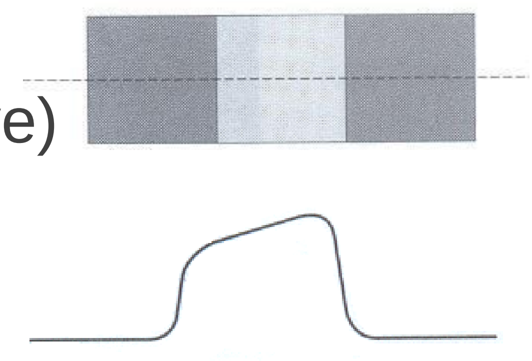
Burger and Burge

Sumary

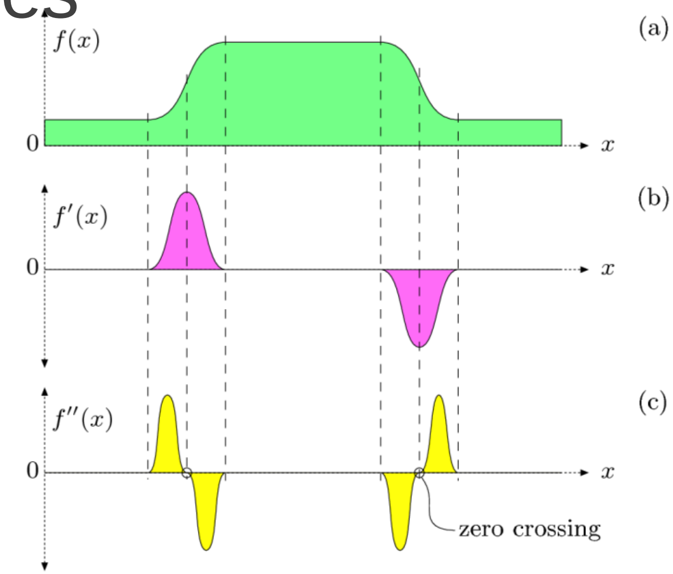
- Edges
  - Introduction
  - Edge detection
- Lines and corners
  - Line detection operators
  - Hough Transform
  - Harris corner detector
  - Other feature detectors

Edge detection

- Typical 2 steps for edge detection:
  - Apply a mask (to approximate a derivative)
  - Aggregate detected pixels (edgels) in edges



- Derivatives are used to detect edges
  - 1st derivative
    - > or < 0 depending on $I(x)$ variation
    - =0 in areas of same intensity
  - 2nd derivative
    - =0 in both positive and negative edges

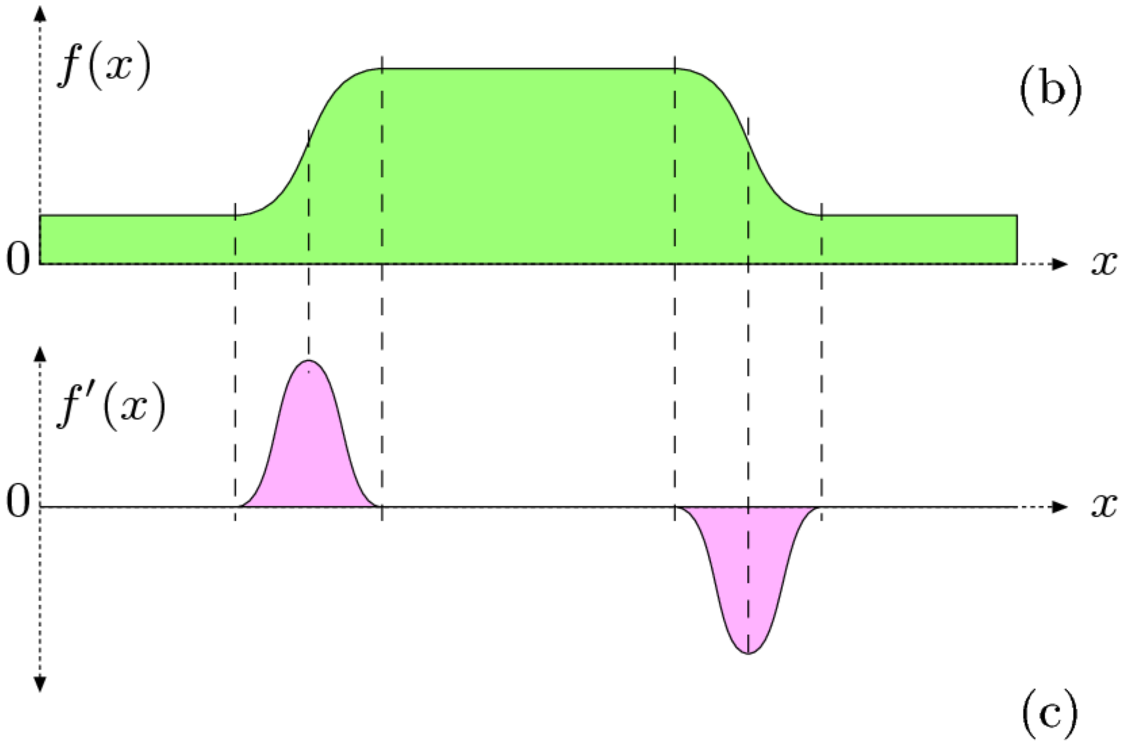


Burger and Burge

Edge detection – 1st derivative

- 1st derivative

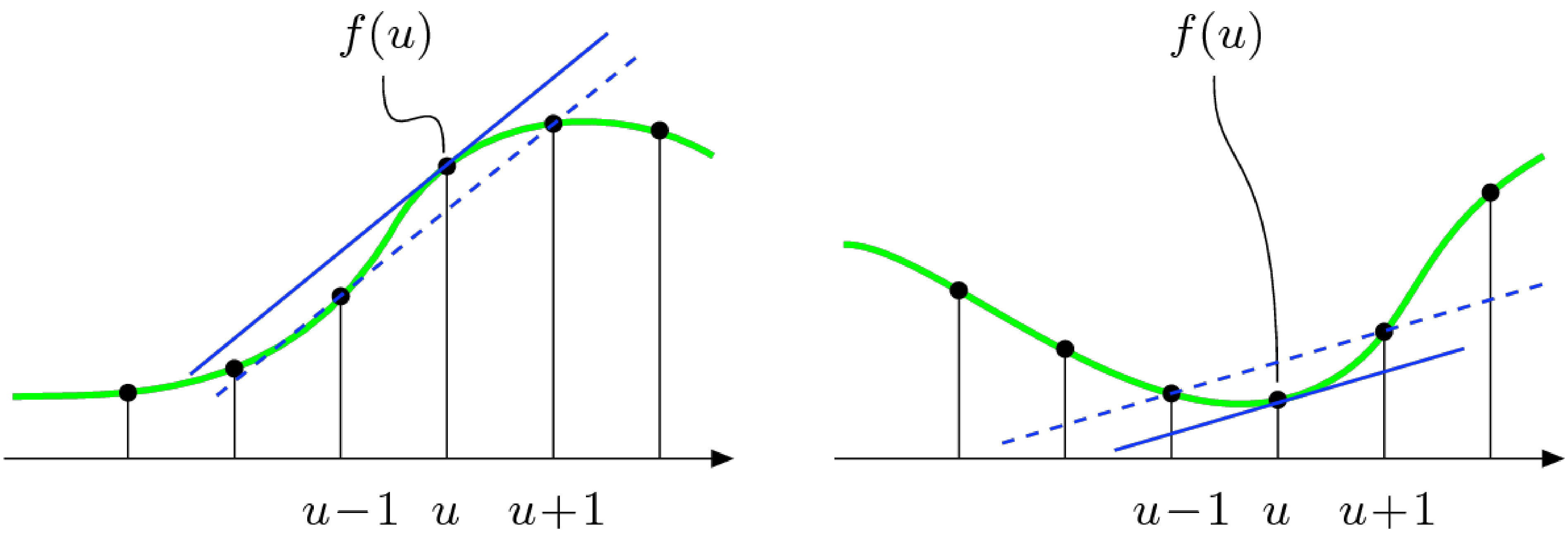
(a)



$$f'(x) = \frac{df}{dx}(x)$$

Burger and Burge

- 1st derivative – simple approximation



$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$

Burger and Burge

Edge detection – 1st derivative

- Partial derivatives and gradient

$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$

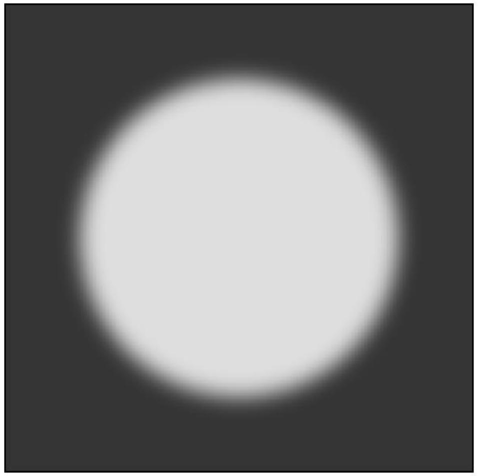
$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

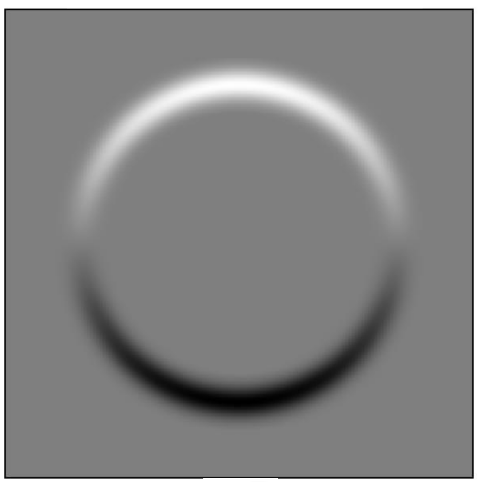
Burger and Burge

- Partial derivatives and gradient



Original image

H_x^D



H_y^D

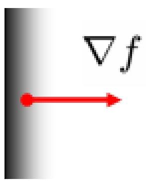
$|\nabla I|(u, v)$

Burger and Burge

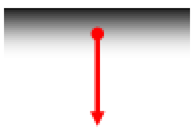
Edge detection - operators

- Derivatives Operators
  - Image gradient points into the direction of larger intensity variation

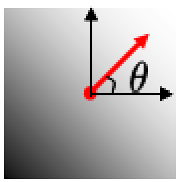
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

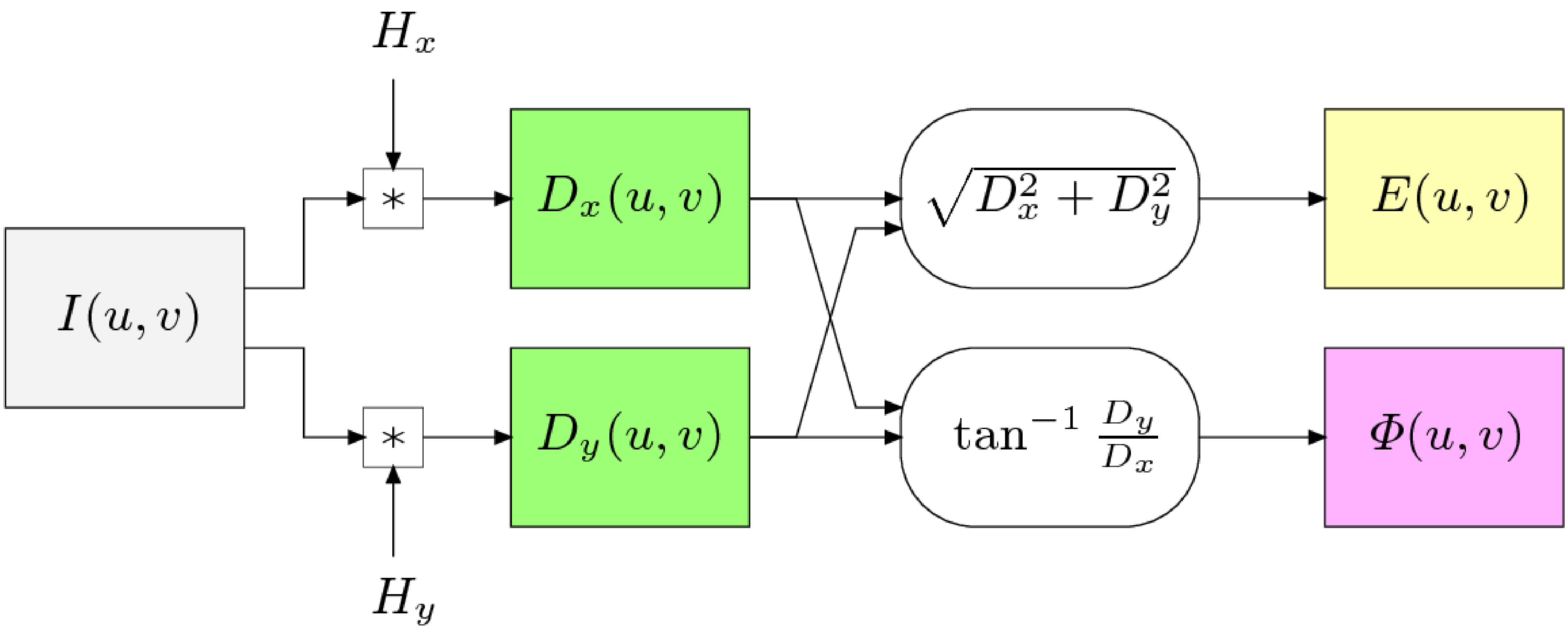
Gradient amplitude

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Gradient direction

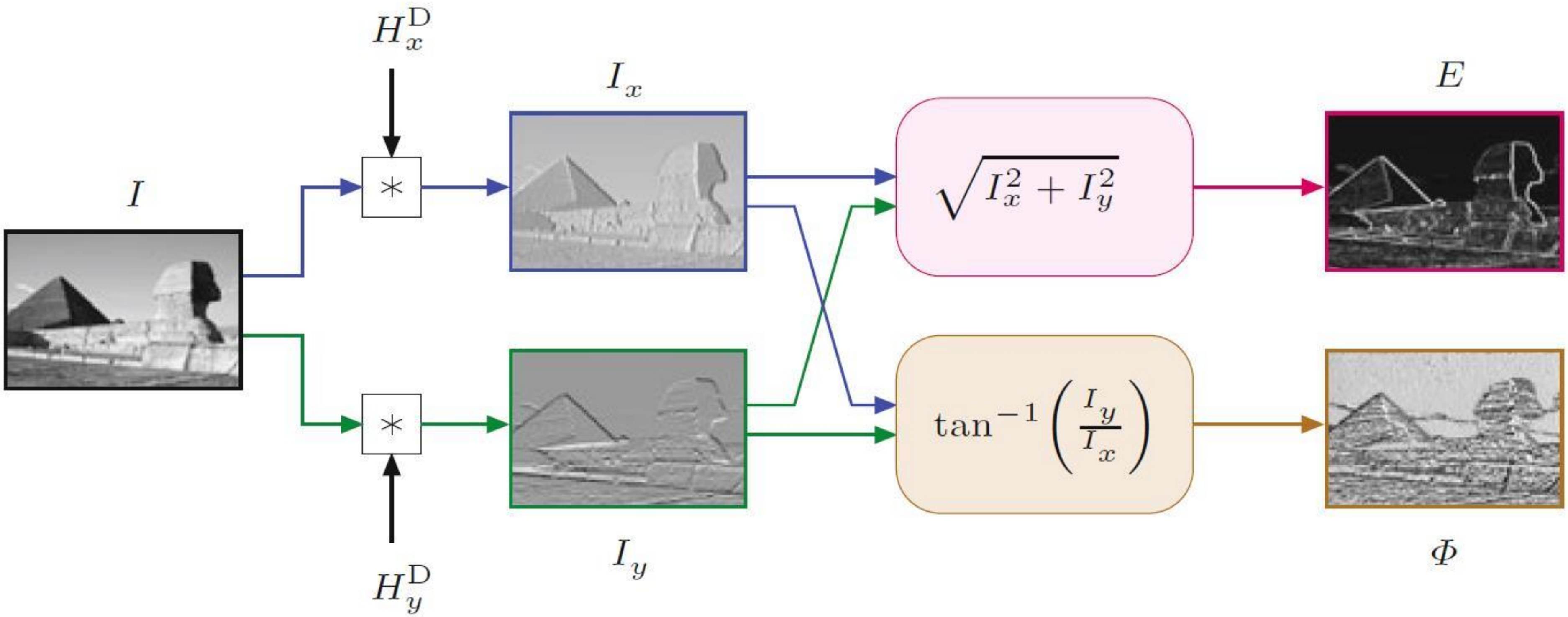
Burger and Burge

- How to use operators?



Burger and Burge

- How to use operators?



Burger and Burge

- Roberts operator
  - Simple, fast but very noise sensitive

$D_1 = I * H_1^R$

$D_2 = I * H_2^R$

$$H_1^R = \begin{bmatrix} 0 & \mathbf{1} \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$

Burger and Burge

Edge detection - operators

- Prewitt operator

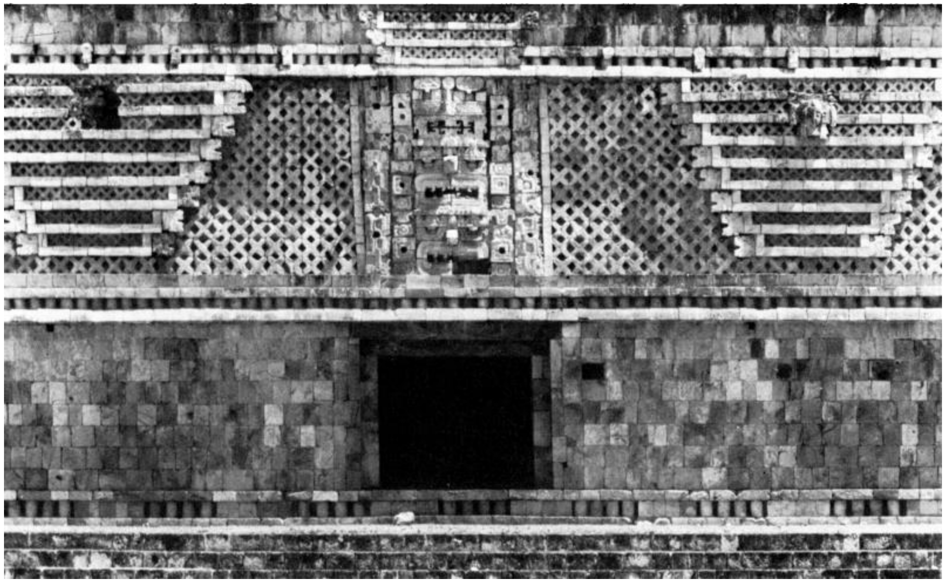
$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

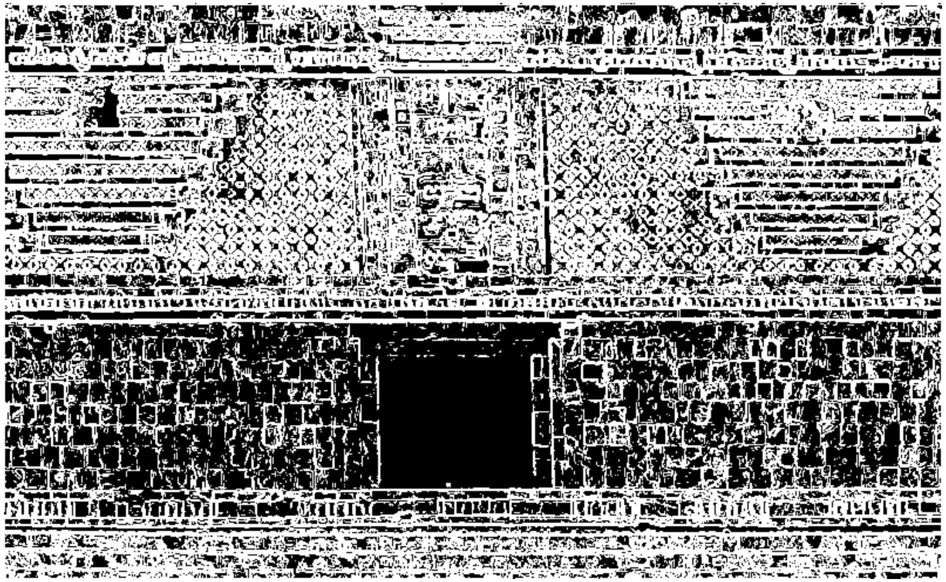
-1	-1	-1
0	0	0
-1	-1	-1

Horizontal

-1	0	-1
-1	0	-1
-1	0	-1

Vertical





Edge detection - operators

- Sobel operator

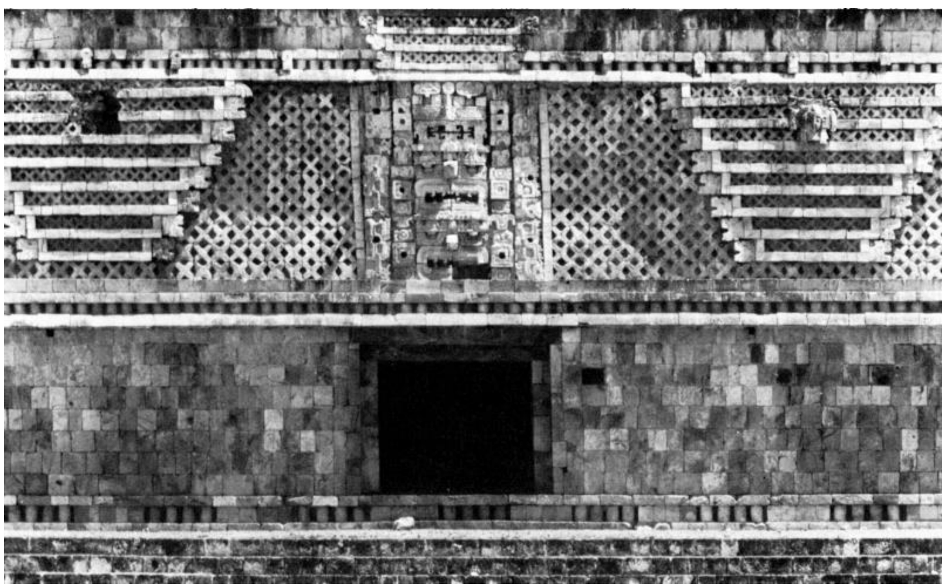
$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

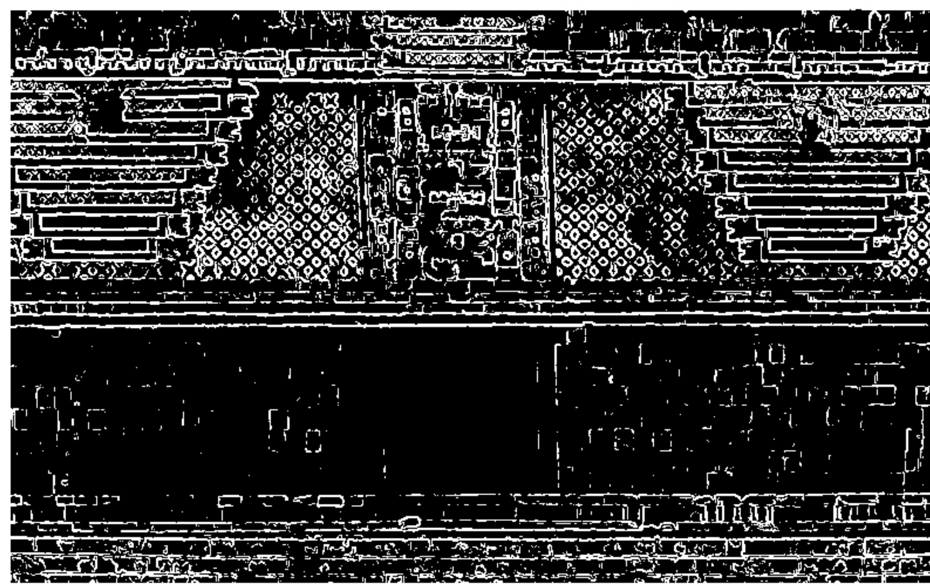
-1	-2	-1
0	0	0
1	2	1

Horizontal

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

Vertical





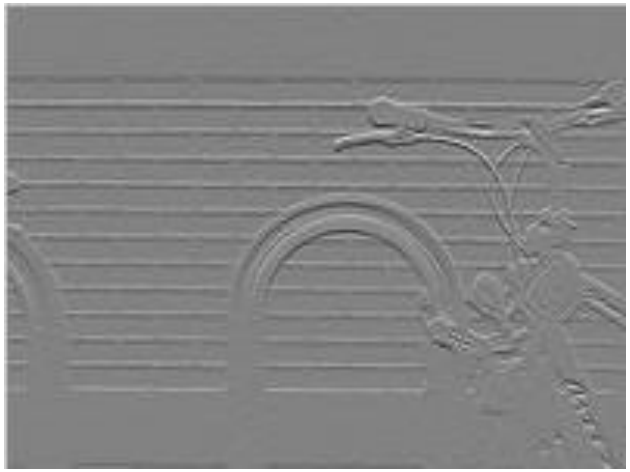
Edge detection - operators

- Sobel operator

Y – Direction Kernel

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

Original Image



X – Direction Kernel

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

Resulting image

Edge detection - operators

- Compass Edge Detection
  - alternative to gradient edge detection (Roberts and Sobel operators).
- Usually outputs two images
  - Gradient magnitude
  - edge orientation
- Gradient is estimated in eight (for a 3 x 3 convolution mask) possible orientation (from 0° [vertical] to 315° in steps of 45°.
- The convolution result of greatest magnitude indicates the gradient direction

- Extended-Sobel Operator

$$H_0^{\mathrm{ES}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_1^{\mathrm{ES}} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix},$$

$$H_2^{\mathrm{ES}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad H_3^{\mathrm{ES}} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix},$$

$$H_4^{\mathrm{ES}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad H_5^{\mathrm{ES}} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix},$$

$$H_6^{\mathrm{ES}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad H_7^{\mathrm{ES}} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}.$$

Edge detection – 2nd derivative

- Laplacian operator
  - Second derivative approximation of ∇^2

4-neighboorhood

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

8-neighboorhood

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

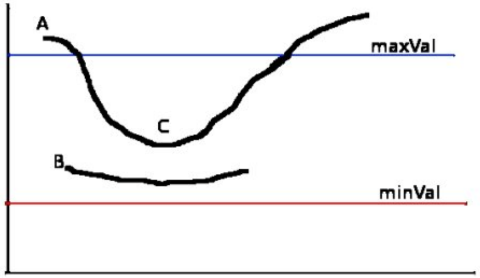
  - Edge detection with first derivative are noise sensitive and object dependent
    - The first derivative of the image function should have an extremum at the position corresponding to the edge
    - It is much easier and more precise to find a zero-crossing position than an extremum.

Edge detection - Canny

- Canny objectives
  - Good location (zero crossing)
  - Minimize weak edges

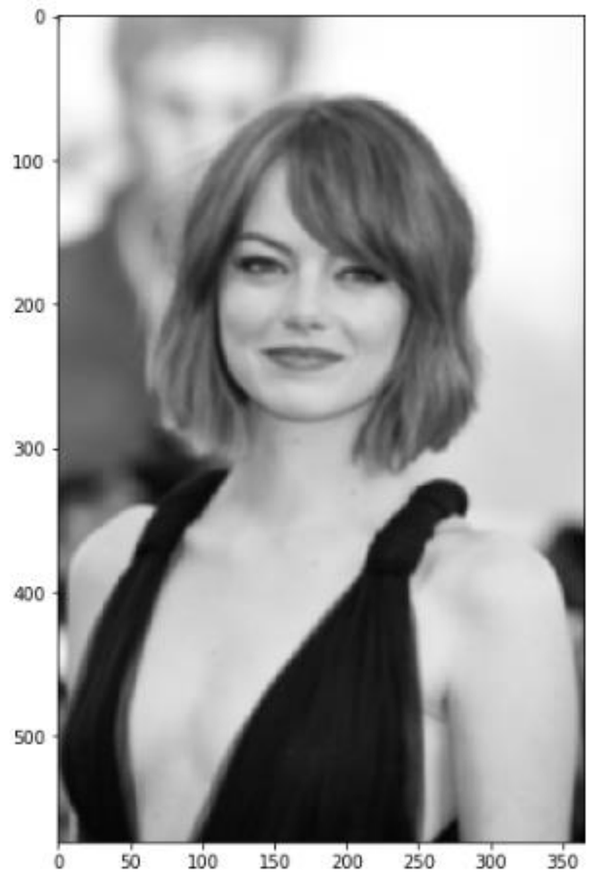
Burger and Burge

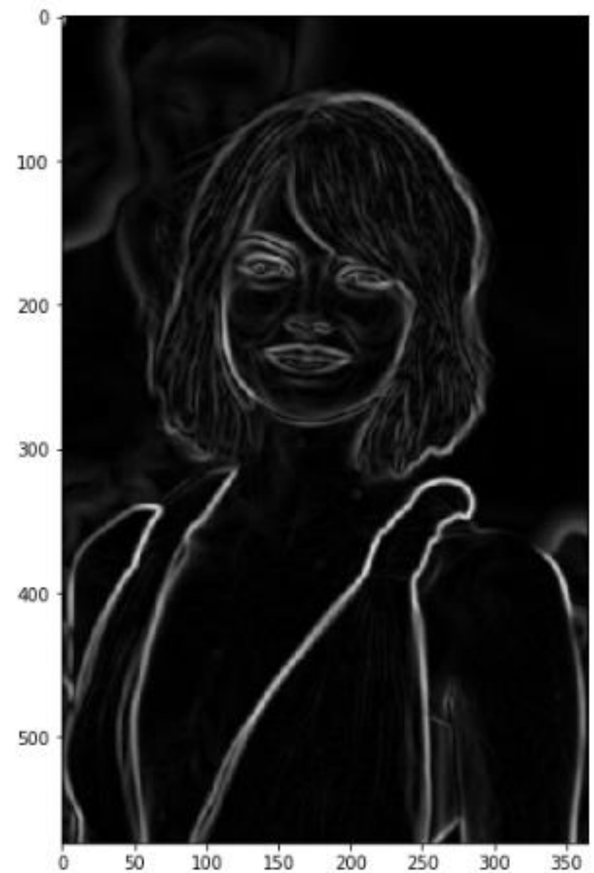
- Canny Edge Detector (1986)
  - Process in five steps:
    1. Gaussian filter to smooth and remove noise
    2. Find intensity gradients of the image (Sobel operator)
    3. Non-maximum supression
    4. Double threshold to determine potential edges
       High threshold for strong pixels
       Low threshold for non-relevant pixels
    5. Edge Tracking by Hysteresis
       Transform weak into strong pixels, if at least one neighboring pixels is processed as strong



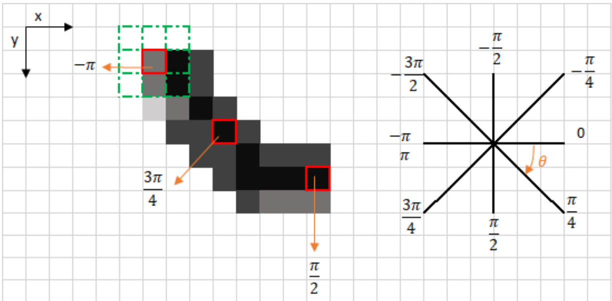
- Canny Edge Detector (1986)
  - Process in five steps:
    1. Gaussian filter to smooth and remove noise

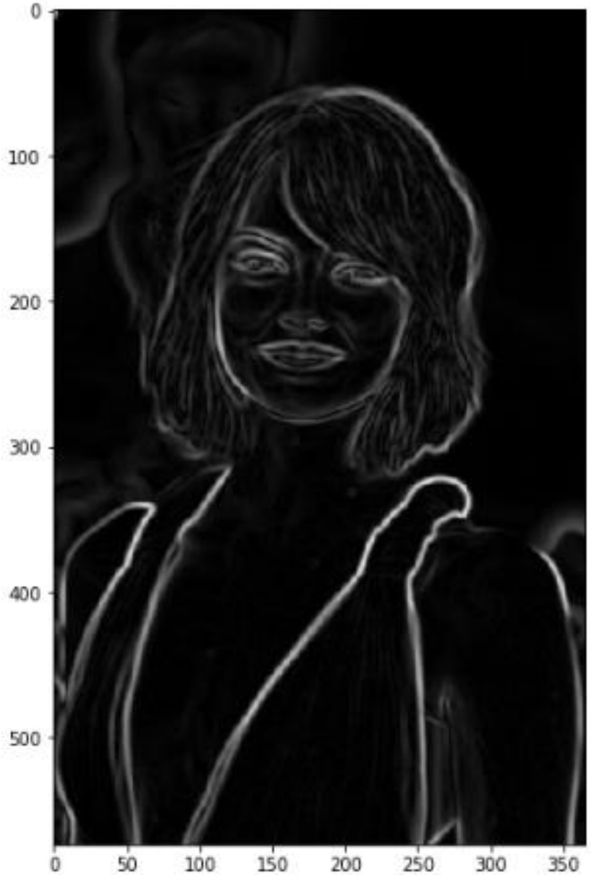
- Canny Edge Detector (1986)
  - Process in five steps:
    2. Find intensity gradients of the image (Sobel operator)

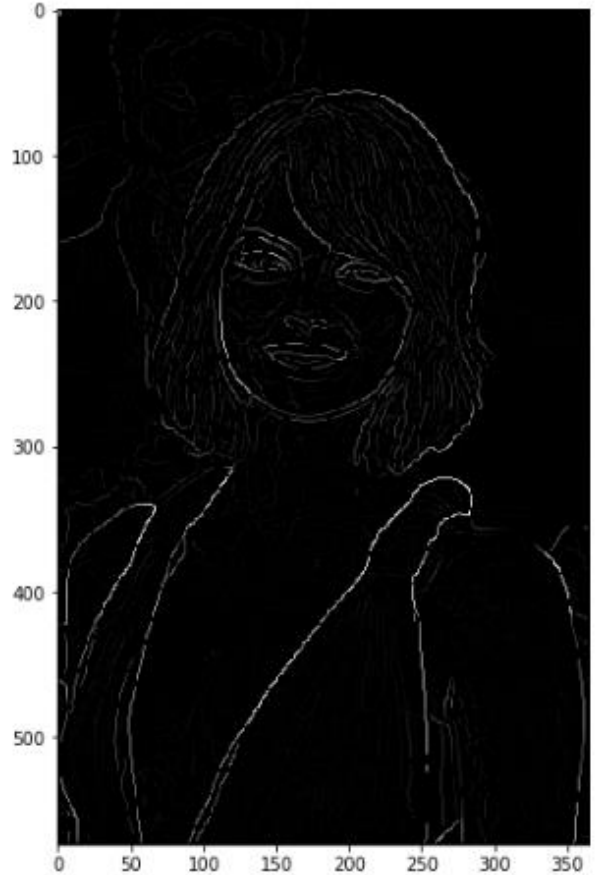




- Canny Edge Detector (1986)
  - Process in five steps:
    3. Non-maximum suppression



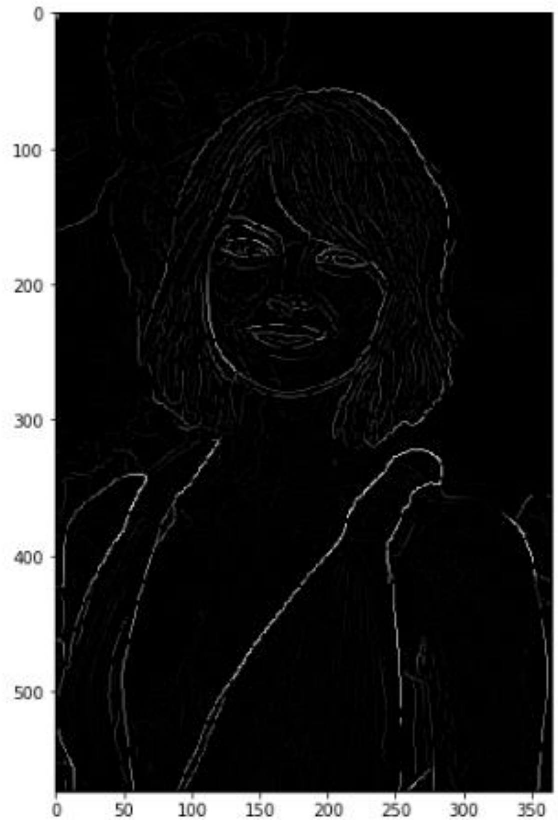


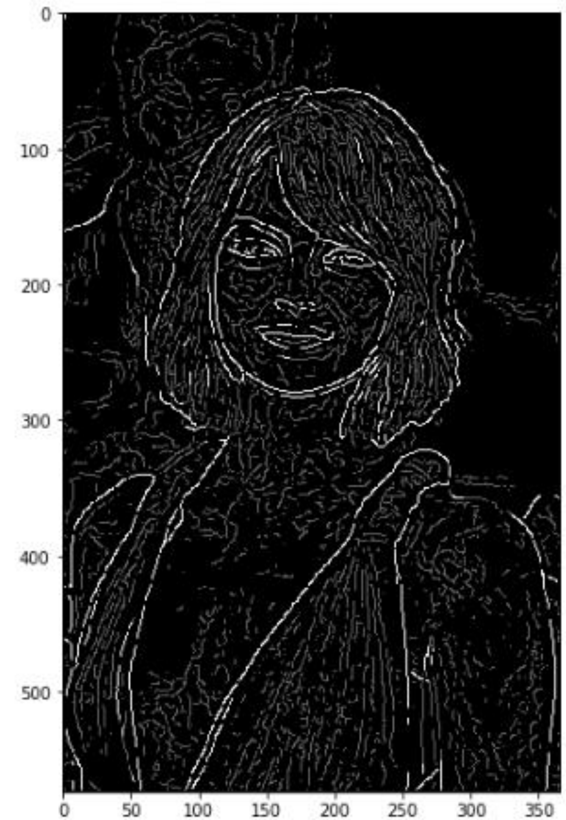


- Canny Edge Detector (1986)
  - Process in five steps:
    4. Double threshold to determine potential edges

High threshold for strong pixels

Low threshold for non-relevant pixels

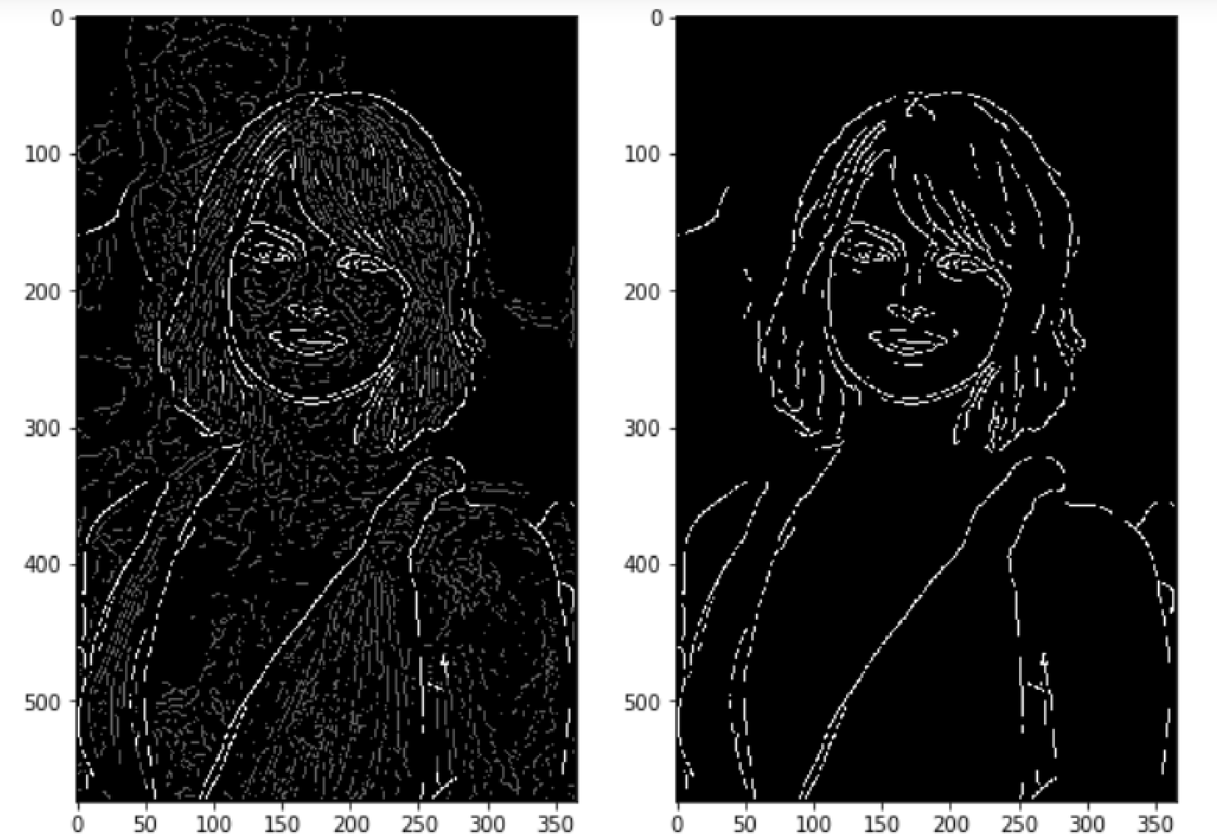




Edge detection - Canny

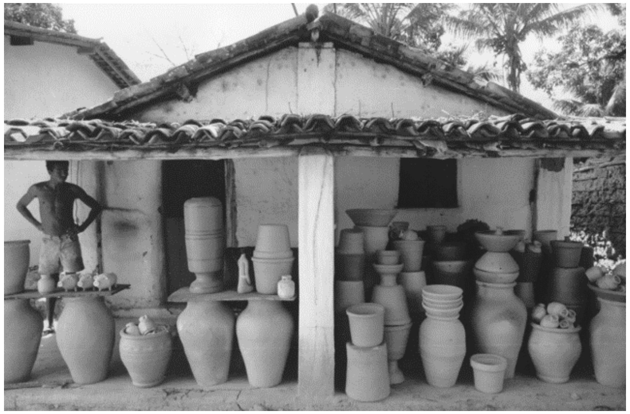
- Canny Edge Detector (1986)
  - Process in five steps:
    5. Edge Tracking by Hysteresis

Transform weak into strong pixels, if at least one neighboring pixels is processed as strong

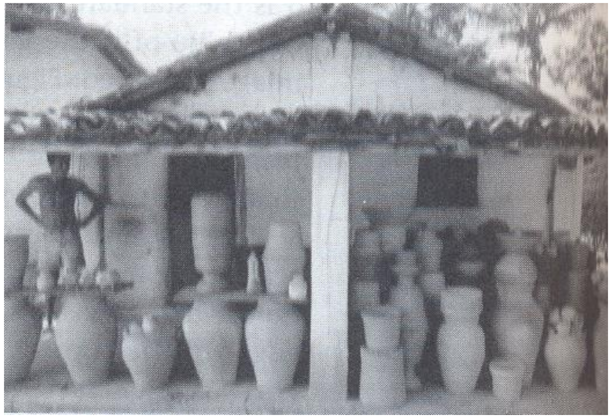


Images from https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Edge detection - Canny

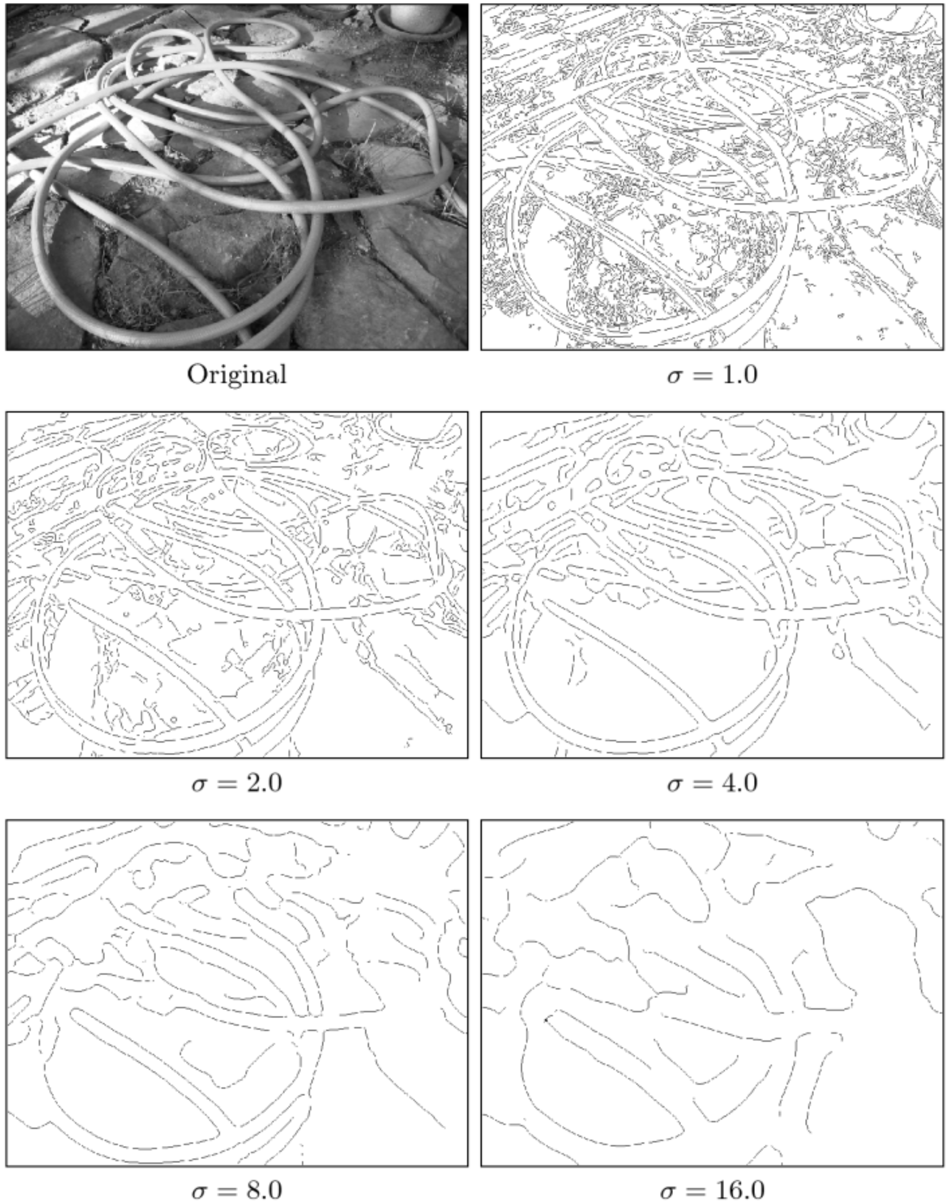


Original Image



1D convolution (x,y) with Gaussian smoothing $\sigma = 6$

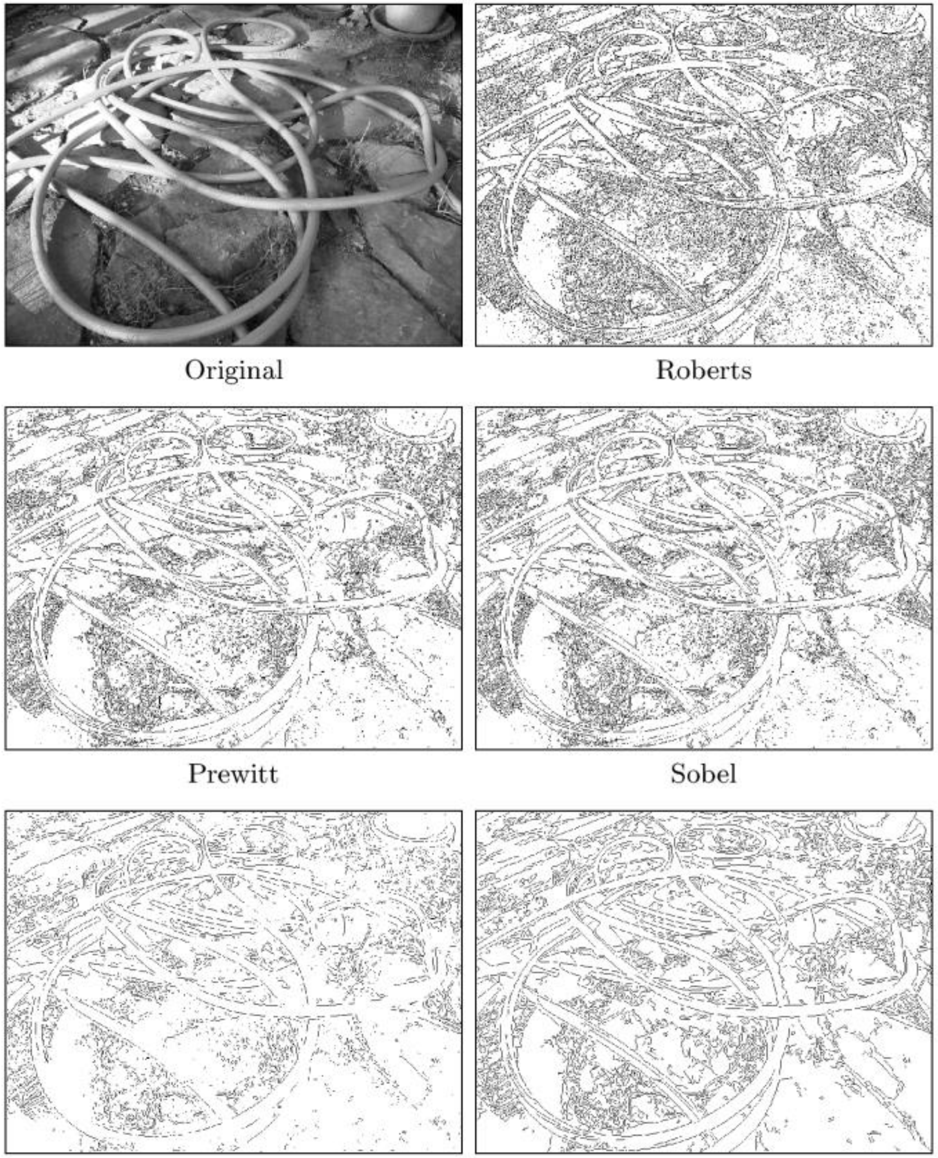
Canny edges



Burger and Burge

Edge detection - comparison

- Possible criteria:
  - Number of weak/false edges
  - Connectivity
  - …



Original Roberts Prewitt Sobel Laplacian of Gaussian Canny ($\sigma = 1.0$)

Burger and Burge

Sumary

- Edges
  - Introduction
  - Edge detection
- Lines and corners
  - Line detection operators
  - Hough Transform
  - Harris corner detector
  - Other feature detectors

Lines detection

- Same rationale of detecting "roof" like profiles along "strategic“ orientations: 0º; 45º; 90º; 135º [see compass]
- Convolution Kernels

$$h_1 = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \quad h_3 = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

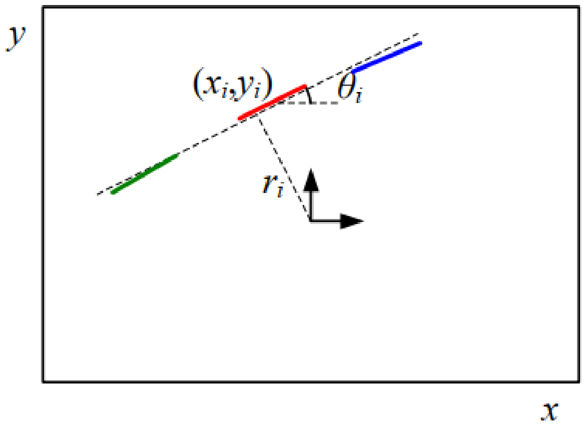
- Lines detected this way are collections of edges. Most of the time non single pixel wide edges.
- Necessary to introduce line thinning algorithms

Sumary

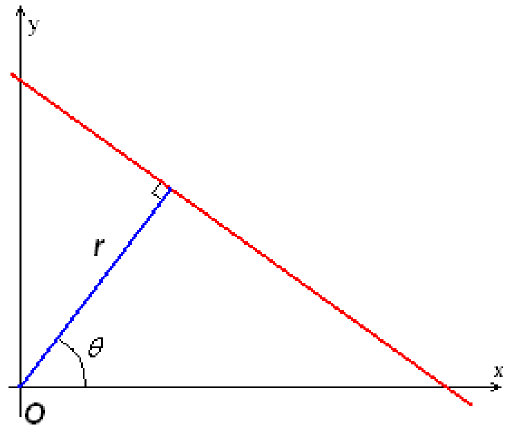
- Edges
  - Introduction
  - Edge detection
- Lines and corners
  - Line detection operators
  - Hough Transform
  - Harris corner detector
  - Other feature detectors

Hough transform (1962)

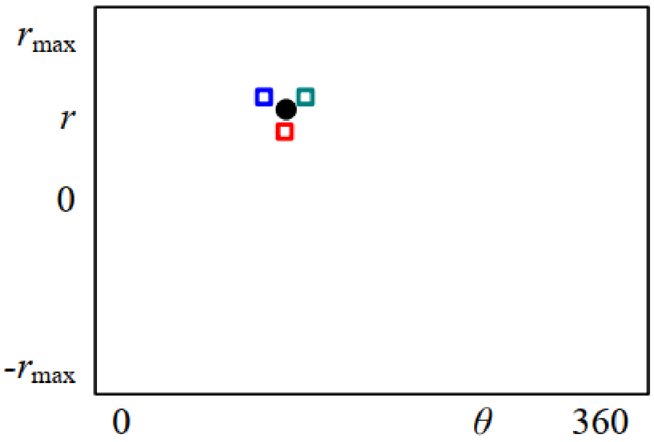
- Technique for having edges "vote" for plausible line locations
- Represent line edges in polar coordinates (r, θ) in the Hough space



Line edge in original image



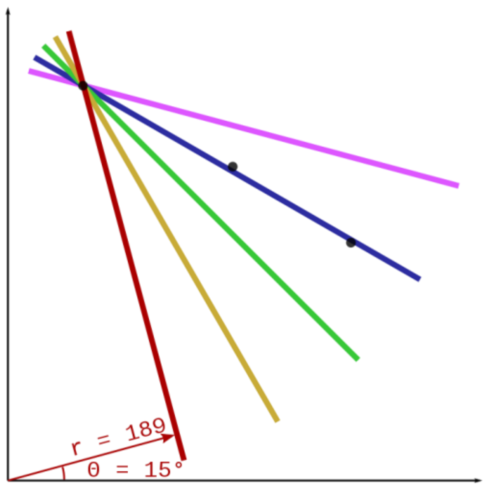
Conversion to (r, θ) representation



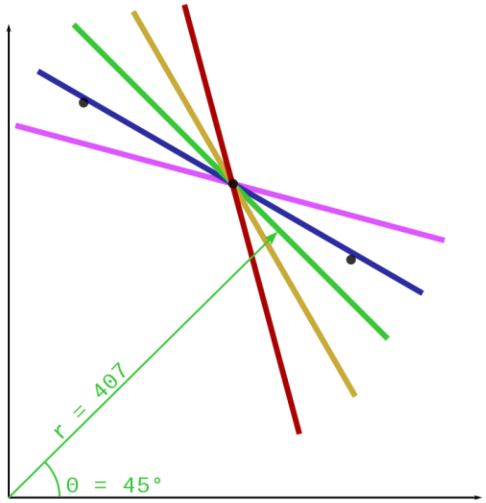
Line representation in Hough space

Hough transform

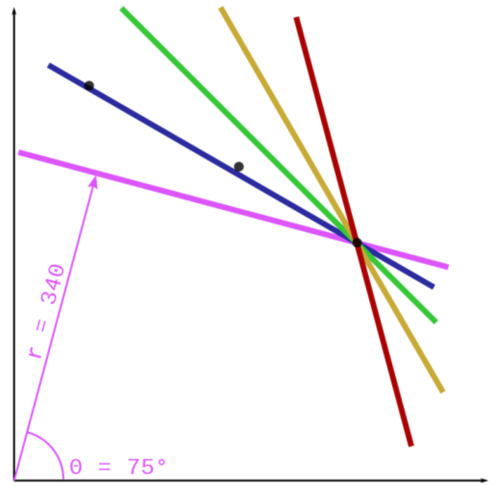
- Line representation in Hough space



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4



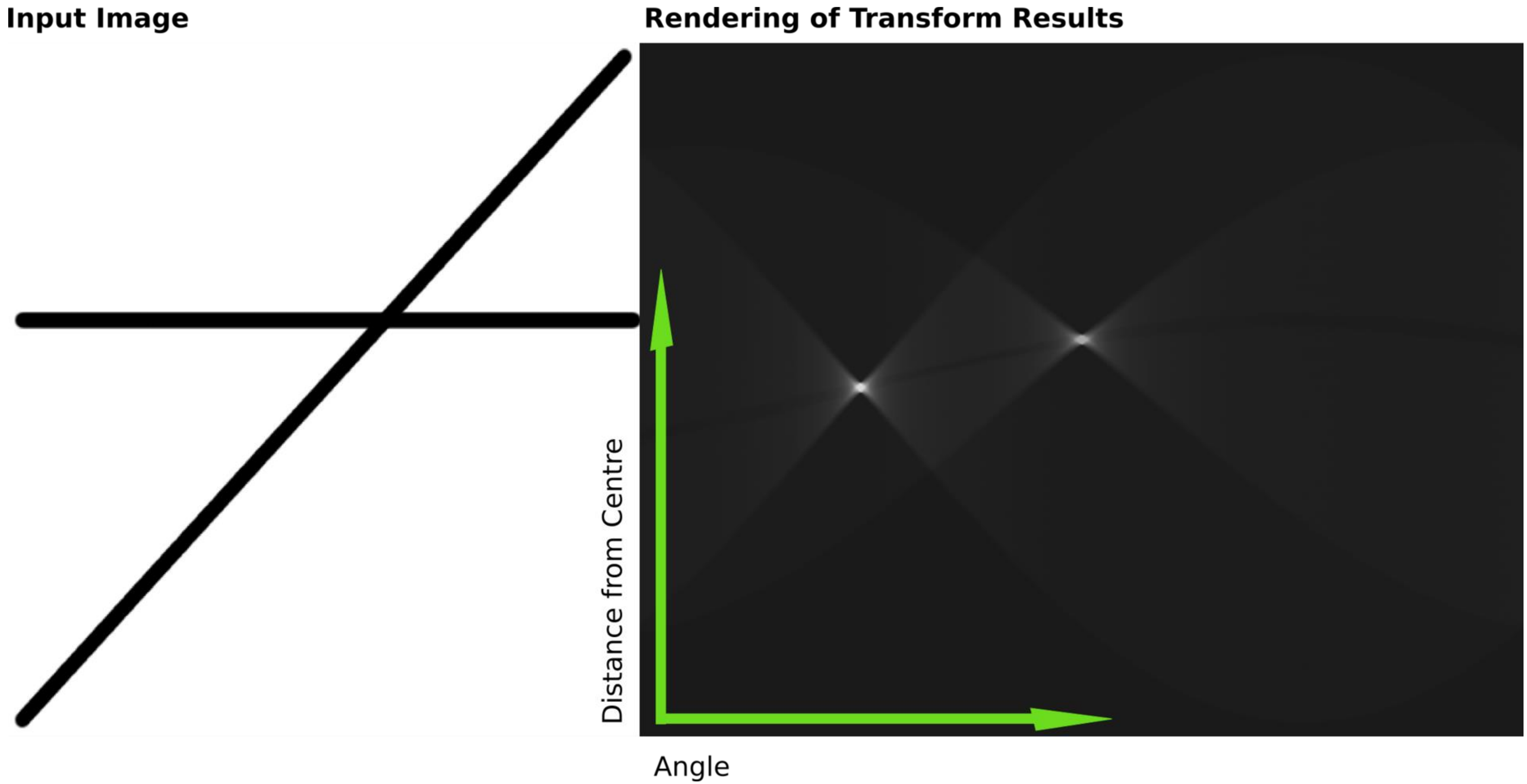
θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3



θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

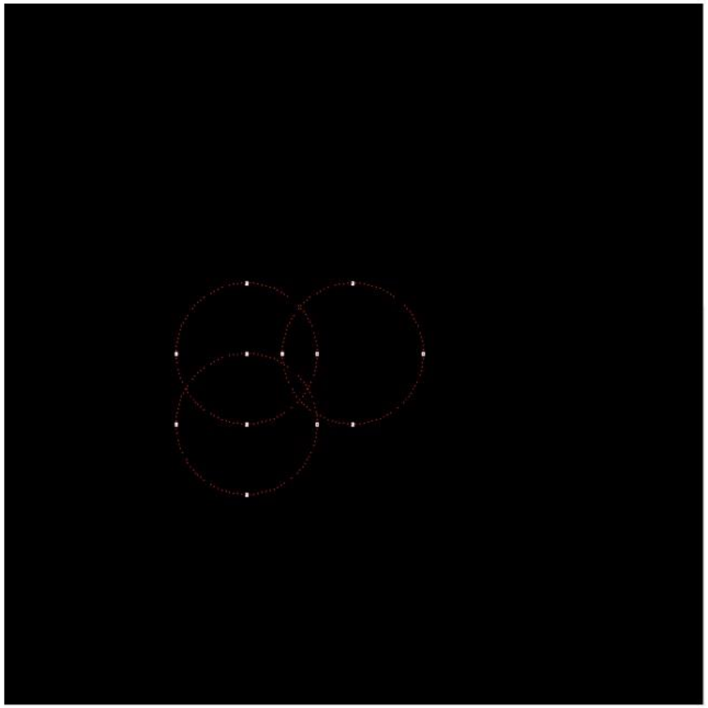
Hough Transform

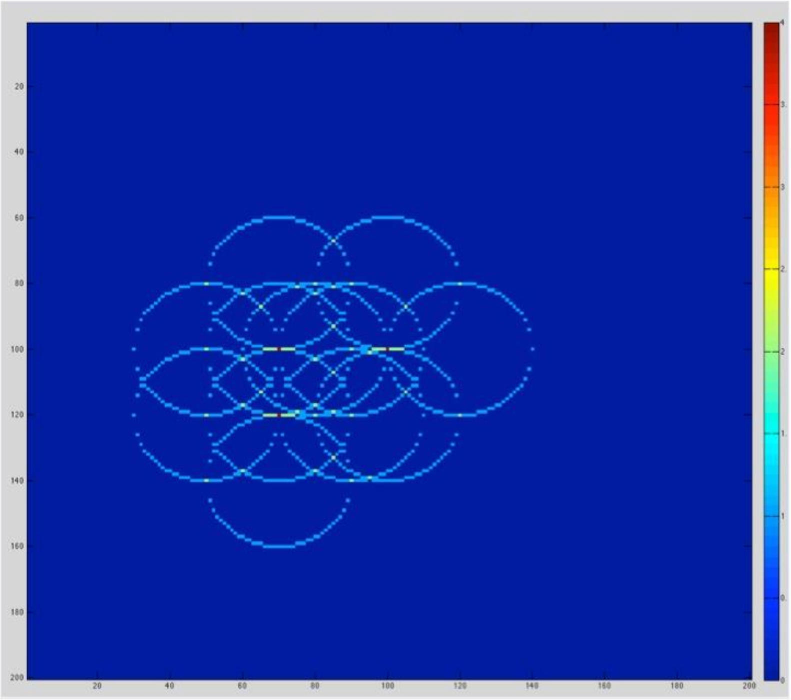
- Higher cell values in Hough accumulator are the Hough parameters of the lines for which angle and distance can be determined



Hough Transform

- Classical Hough transform for line identification
- Extended to identifying positions of other shapes as circles or ellipses.



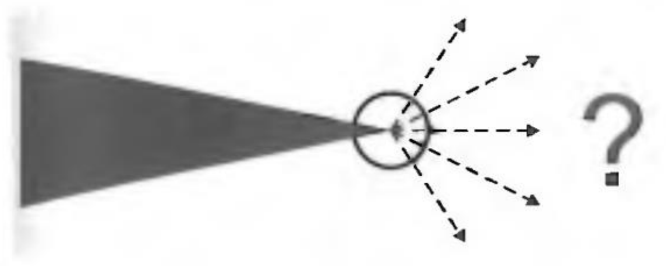


Sumary

- Edges
  - Introduction
  - Edge detection
- Lines and corners
  - Line detection operators
  - Hough Transform
  - Harris corner detector
  - Other feature detectors

Corner detector

- Corners in images can be located using local detectors;
  - Input to the corner detector is the Gray-level image
  - Output is the image in which values are proportional to the likelihood that the pixel is a corner.
  - Interest points are obtained by thresholding the result of the corner detector.
- Edge detectors themselves are not stable at corners.
  - Gradient at the tip is ambiguous



Moravec corner detector

- One of the earliest corner detection - 1979
- Corner: point with low self-similarity
  - Tests pixels as corners considering similarity between nearby, largely overlapping patches.
  - Similarity is measured by taking the sum of squared differences (SSD) between the corresponding pixels of two patches

Harris corner detector

- Auto-correlation based
- Improvement upon Moravec's corner detector
- Use a sliding window W patch and estimate the sum of square differences of the discriminant function:

$$N = \begin{bmatrix} \sum\limits_{window} f_r^2(r,c) & \sum\limits_{window} f_r(r,c)\cdot f_c(r,c) \\ \sum\limits_{window} f_r(r,c)\cdot f_c(r,c) & \sum\limits_{window} f_c^2(r,c) \end{bmatrix}$$

$f_r(r,c)$: horizontal gradient

$f_c(r,c)$: vertical gradient

- Compute smallest eigenvalue of the structure tensor:

$$\lambda_{\min} \approx \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)} = \frac{\det(M)}{\operatorname{tr}(M)}$$

with the trace $\operatorname{tr}(M) = m_{11} + m_{22}$.

Corner detector

- Other corner detector:
  - Kitchen - 82
  - Harris - 88
  - Deriche - 90
  - Mehrotra - 90
  - Schmid - 98
  - Smith – 98
  - …

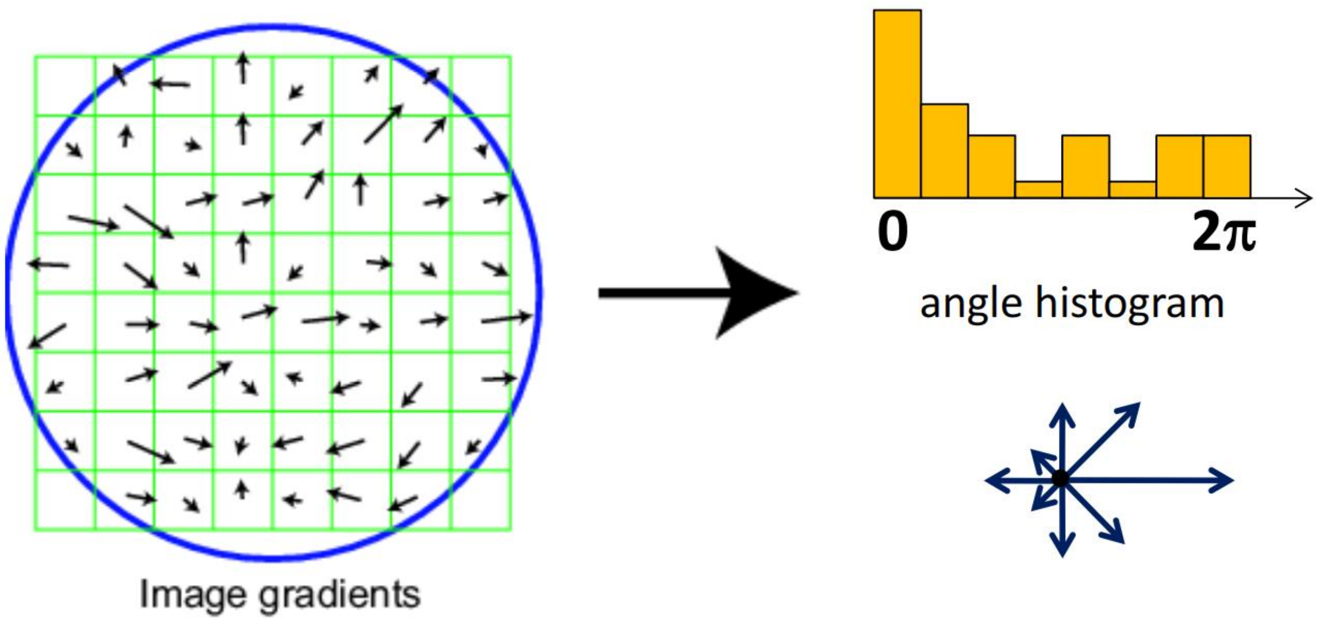
Sumary

- Edges
  - Introduction
  - Edge detection
- Lines and corners
  - Line detection operators
  - Hough Transform
  - Harris corner detector
  - Other feature detectors

Other feature detectors

- Several feature descriptors more or less invariant to scale, rotation, affine transformations:
  - Histogram based (use histogram of oriented gradient)
    - SIFT - Scale invariant feature transform
    - SURF - Speeded-Up Robust Features
    - GLOH - Gradient Location and Orientation Histogram
    - HOG - Histogram of Oriented Gradients
  - Compact descriptors (use binary strings comparing pairs of intensity images)
    - BRIEF - Binary Robust Independent Elementary Features
    - FAST - Features from accelerated segment test
    - ORB - Oriented FAST and Rotated BRIEF
    - BRISK - Binary Robust invariant scalable keypoints

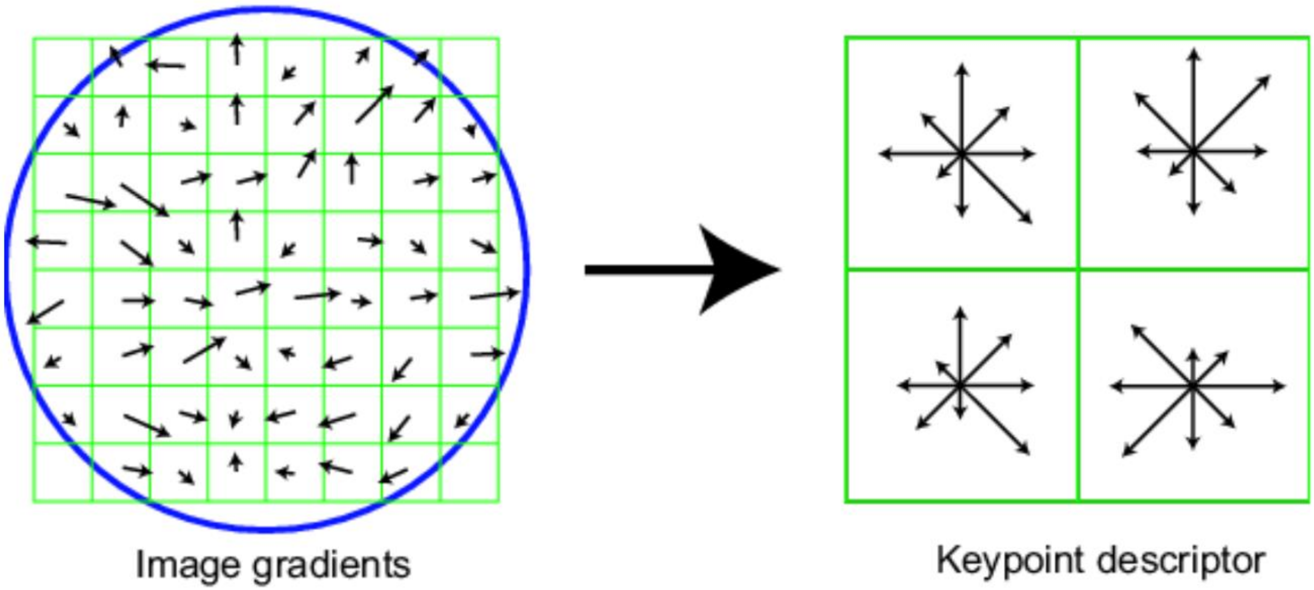
- SIFT Basic idea:
  - Take 16x16 square window around detected feature
  - Compute edge orientation (angle of the gradient) for each pixel
  - Throw out weak edges (threshold gradient magnitude)
  - Create histogram of surviving edge Segment Test



Distinctive image features from scale-invariant keypoints. David G. Lowe. IJCV 60 (2), pp. 91-110, 2004

Other feature detectors - SIFT

- SIFT Full version:
  - Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
  - Compute an orientation histogram for each cell
  - 16 cells * 8 orientations = 128 dimensional descriptor



Distinctive image features from scale-invariant keypoints. David G. Lowe. IJCV 60 (2), pp. 91-110, 2004