Edge Detection

S. Vartalami, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr

Version 4.1

Edge Detection Overview

- **Introduction**
- Edge detection
- Edge thresholding
- Hough transform
- Edge following algorithms
- Contour detection
- Active Contours
- Neural Edge detection
- Neural Contour detection.

Introduction

An ***image edge*** can be considered as the border between two homogeneous image regions having different illumination intensities.

Edges are useful for:

- image analysis, object recognition and
- image filtering, image compression.

Introduction

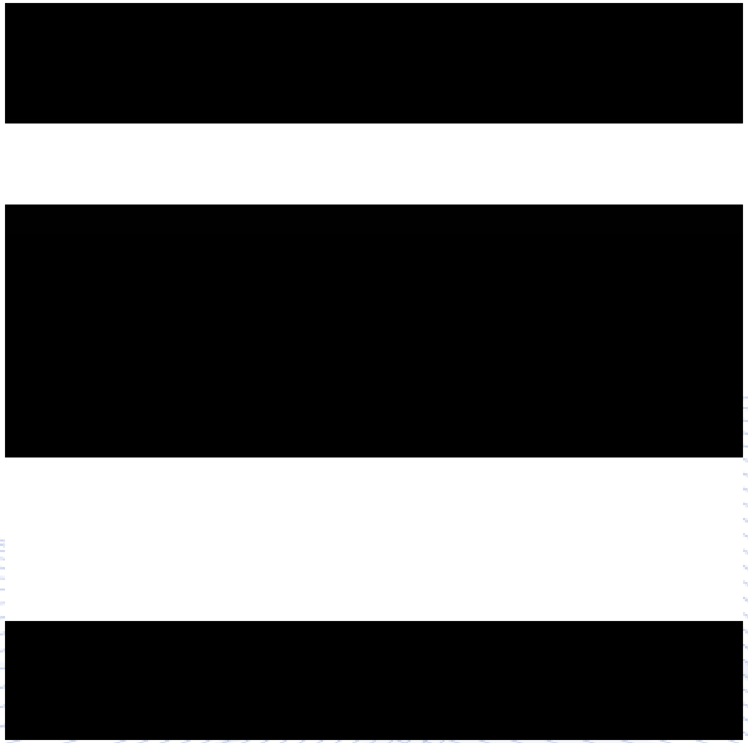
Edge detectors can be grouped into two classes:

- ***Local techniques*** use operators on local image neighborhoods.
- ***Global techniques*** use global information and filtering methods to extract edge information.

Edge Detection Overview

- Introduction
- **Edge detection**
- Edge thresholding
- Hough transform
- Edge following algorithms
- Contour detection
- Active Contours
- Neural Edge detection
- Neural Contour detection.

Edge types



Horizontal image edges

Edge types

Vertical image edges

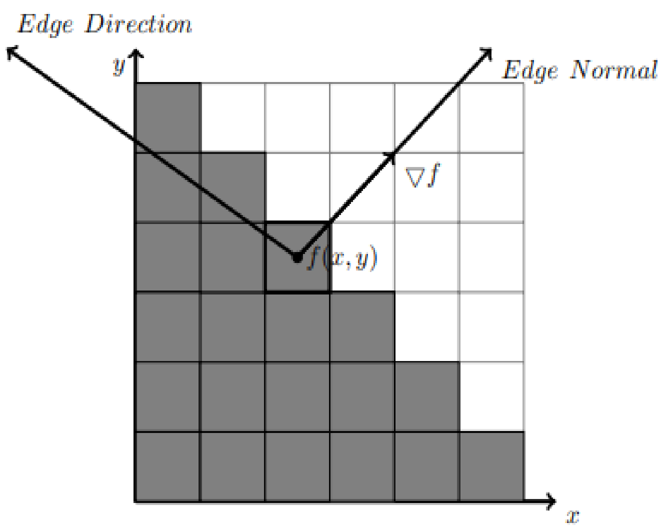
Edge types

Horizontal

a)Step Edge b) Ramp Edge

Edge descriptors

- ***Edge normal*** is a unit vector in the direction of maximum image intensity change (image grad).
- ***Edge direction*** is a unit vector perpendicular to edge normal. It can also be described by ***edge direction angle***.
- ***Edge position or center*** is the image position where the edge is located.
- ***Edge strength*** is related to the local image intensity change along edge normal.



Edge detection steps

- ***Image Smoothing*** suppresses as much noise as possible, without destroying true image edges.
  - Image smoothing is a low-pass image operator.
- ***Image Enhancement*** enhances edge quality, typically by image sharpening.
  - Image sharpening is a high-pass image operator.
- ***Edge Detection*** retains true edge pixels, while discarding edge noise.
  - Usually, edge thresholding is used for true edge pixel detection.

Edge detection steps

- ***Edge Localization*** determines the exact edge location.
  - Sub-pixel edge localization might be required for some applications at a fraction pixel distance, at an e.g., $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ pixel resolution.

- ***Edge thinning*** reduces edge width possibly to 1 pixel.
- ***Edge linking*** connects broken edge segments.

Edge detection

- Edge detection is typically a ***local image differentiation*** of the 2D signal $f(x, y)$ along x, y image directions.

- Local image differentiation techniques can produce edge detector operators.

Edge detection

Image intensity gradient:

$$\nabla f(x,y) \triangleq [\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}]^T \triangleq [f_x \quad f_y]^T.$$

Gradient magnitude can be used as edge detector:

$$e(x,y) = \sqrt{f_x^2(x,y) + f_y^2(x,y)}.$$

Edge detection

- It can be used as a measure of edge strength.

- Alternatives to be used for fast calculation:

$$e(x,y) = |f_x(x,y)| + |f_y(x,y)|.$$

Edge direction angle:

$$\varphi(x,y) = \arctan\left(\frac{f_y}{f_x}\right).$$

Edge detection

Gradient estimates can be obtained by using ***gradient operators*** of the form:

$$\widehat{f_x} = \mathbf{w}_1^T \mathbf{x},$$
$$\widehat{f_y} = \mathbf{w}_2^T \mathbf{x}.$$

- $\mathbf{x}$: local $M \times M$ image neighborhood pixel vector.
  - Typically, 3×3 image neighborhoods are used.
- $\mathbf{w}_1$, $\mathbf{w}_2$: ***gradient masks*** (weight vectors) having $M \times M$ entries.

Edge detection

Gradient masks examples:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt edge detector masks.

Sobel edge detector masks.

- They can be used for horizontal (left) or vertical (right) edge detection.
- No or trivial (by 2) multiplications are involved.

Edge detection

Edge templates are masks that can be used to detect edges along different edge directions.

- Such masks of size 3×3 are:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

Kirsch edge detector masks.

- They detect horizontal (0^0), vertical (90^0), $45^0, 135^0$ image edges.

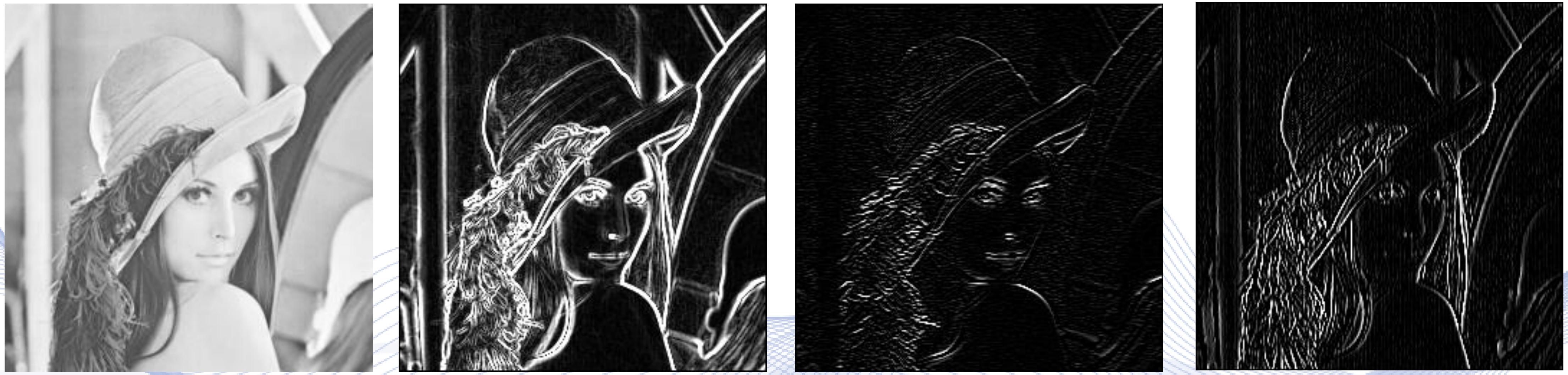
Edge detection

All templates are applied to each image pixel. The template that produces the maximal output is the winner:

$$e(x,y) = \left|\mathbf{w}_i^T\mathbf{x}\right|, \quad \text{if} \left|\mathbf{w}_i^T\mathbf{x}\right| \geq \left|\mathbf{w}_j^T\mathrm{x}\right|, \qquad j = 1,2,\dots,n.$$

- $\mathbf{w}_i, i = 1,\dots n$ is the weight vector associated with each template.
- The corresponding output $|\mathbf{w}_i^T\mathbf{x}|$ is a measure of confidence of the edge detector output (edge strength).

Edge detection



a) Lenna image; b) Sobel edge detector output ; c) horizontal edges; d) vertical edges.

Edge detection

Edge detection using the ***Laplace operator*** :

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

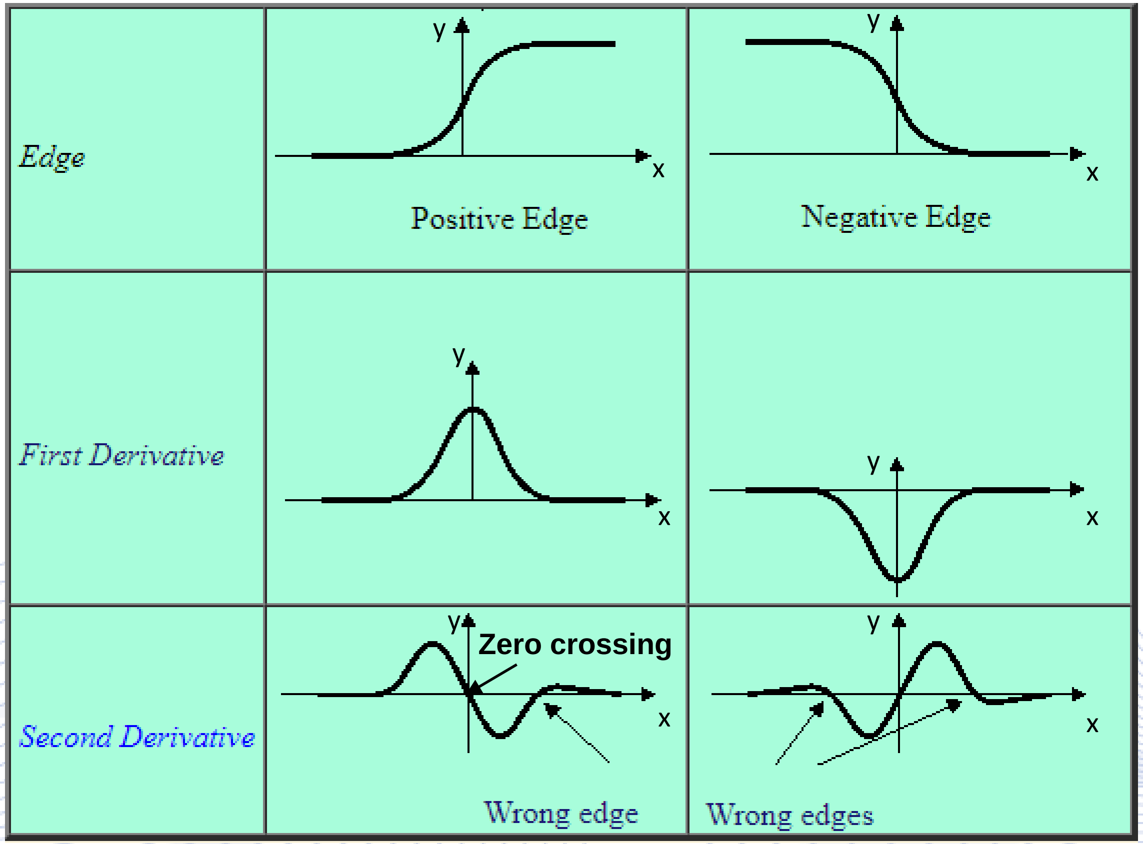
- Edges correspond to large image changes producing:
  - Maxima or minima of first-order image derivatives.
  - Zero-crossings of second-order image derivatives.
- Laplace operator can be numerically approximated:

$$\nabla^2 f(x,y) \cong f(x,y) - \frac{1}{4}[f(x,y+1) + f(x,y-1)\, f(x+1,y) + f(x-1,y)]$$

to find zero-crossing image locations.

Edge detection

$$y = f(x)$$



First and second order differentiation. Zero crossings [DECETI].

Edge detection

- Differentiation is a high-pass operator, enhancing noise.
- Second-order differentiation tends to enhance image noise too much.
- The Laplacian operator creates several false edges, especially in areas where the image variance is small.
- Methods to reduce its noise sensitivity:
  - ***Laplacian-of-Gaussian (LoG)*** performs low-pass Gaussian filtering before differentiation.
  - Consider zero-crossings only in areas, where the local image variance $\sigma^2(i,j)$ is large.

Edge detection

Laplacian-of-Gaussian (***LoG***) HVS model $\nabla^2 G(x,y)$:

- $G(x,y)$ is a low-pass Gaussian function:

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

- Laplacian operator $\nabla^2 f(x,y)$ is a ***2D high-pass filter***.
- LoG operator is given by:

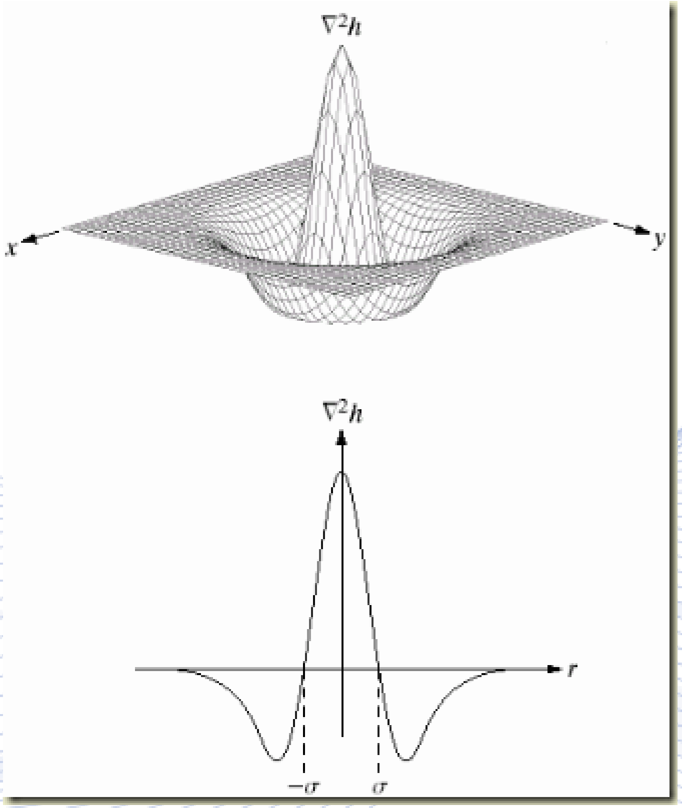
$$\nabla^2 G(x,y) = -\frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)\exp\left\{-\frac{1}{2\sigma^2}(x^2+y^2)\right\}.$$

Edge detection

- LoG has ***band-pass*** frequency characteristics.
- It can smooth noise and perform edge detection.

- 2D LoG has the shape of a ***Mexican sombrero***.
- It models well retina ***ganglion receptive fields***.

Edge detection



Negative LoG function [LOG].

Canny edge detector

- This is probably the most widely used edge detector in computer vision.
- Canny has shown that the first derivative of a Gaussian filter kernel closely approximates the operator that optimizes the product of signal-to-noise ratio and localization.

- This analysis is based on step edges corrupted by additive Gaussian noise.

Canny edge detector

- Image smoothing can be performed by applying a ***Gaussian filter*** $G(x, y)$:

$$f(x, y) = i(x, y) ** G(x, y),$$

where $G(x, y)$ is the Gaussian kernel function:

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

and $**$ denotes 2D convolution.

Artificial Intelligence & Information Analysis Lab

Canny edge detector

- The partial derivatives f_x and f_y are given by:

$$f_x = \frac{\partial(i**G)}{\partial x} = i(x,y)**\frac{\partial G}{\partial x}(x,y) = i(x,y)**G_x(x,y),$$

$$f_y = \frac{\partial(i**G)}{\partial y} = i(x,y)**\frac{\partial G}{\partial y}(x,y) = i(x,y)**G_y(x,y).$$

Differentiation property:

$$\frac{d}{dx}(f ** g) = \frac{df}{dx}**g = f**\frac{dg}{dx}$$

Canny edge detector

- $G_x(x,y), G_y(x,y)$ are the partial derivates of $G(x,y)$ with respect to x, y:

$$G_x(x,y) = \frac{-x}{\sigma^2} G(x,y)$$

$$G_y(x,y) = \frac{-y}{\sigma^2} G(x,y).$$

- Compute the gradient magnitude:

$$e(x,y) = \sqrt{{f_x}^2 + {f_y}^2}.$$

- Apply non-maxima suppression.
- Apply hysteresis thresholding/edge linking.

Edge detection

- Local ***data dispersion*** measures can be used as edge detector.

- ***Local variance*** $\sigma^2(i,j)$ in a $M \times M,\ M = 2\nu + 1$ ***image neighborhood*** (***image window***):

$$\sigma^2(i,j) = \frac{1}{M^2} \sum_{k=i-\nu}^{i+\nu} \sum_{l=j-\nu}^{j+\nu} [f(k,l) - \bar{f}(i,j)]^2,$$

$$\bar{f}(i,j) = \frac{1}{M^2} \sum_{k=i-\nu}^{i+\nu} \sum_{l=j-\nu}^{j+\nu} f(k,l).$$

Edge detection

- ***Local image range***:

$$w(k,l) = \max_{A}\{f(k,l)\} - \min_{A}\{f(k,l)\}$$

- A: Local $M \times M$ image window.

Edge Detection Overview

- Introduction
- Edge detection
- **Edge thresholding**
- Hough transform
- Edge following algorithms
- Contour detection
- Active Contours
- Neural Edge detection
- Neural Contour detection.

Edge thresholding

When the edge detector output is large, a local edge is present. This can be found by ***thresholding***:

$$E(i,j) = \begin{cases} 1, & \text{if } e(i,j) \geq T, \\ 0, & \text{otherwise.} \end{cases}$$

- Threshold T can be chosen using edge detector output histogram, so that it exceeds only a small percentage of edge pixels.
- Thresholding is global.
- Edge detector output thresholding produces a binary image.

Edge thresholding

- Global thresholding may produce thick edges in one region and thin or broken edges in another region. Thus, locally adapted thresholding is desirable.
- A heuristic adaption technique is to calculate the local arithmetic mean of the edge detector output:

$$\bar{e}(i,j) = \frac{1}{M^2}\sum_{k=i-\nu}^{i+\nu}\sum_{l=j-\nu}^{j+\nu} e(k,l)$$

and to use it in the threshold calculation:

$$T(i,j) = \bar{e}(i,j)(1+p).$$

- p is a percentage indicating the level of the thresholding above the local arithmetic mean.

Edge Detection Overview

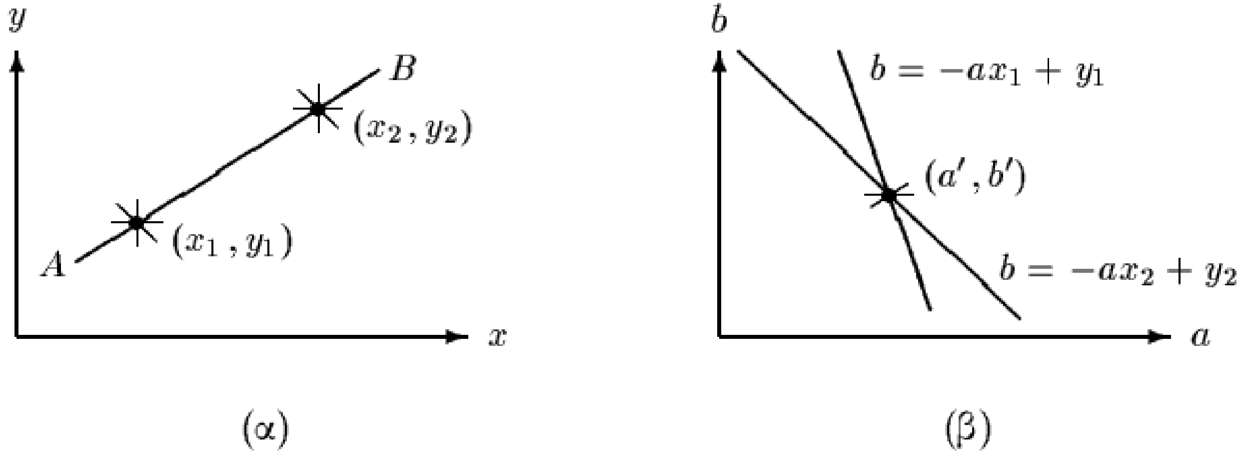
- Introduction
- Edge detection
- Edge thresholding
- **Hough transform**
- Edge following algorithms
- Contour detection
- Active Contours
- Neural Edge detection
- Neural Contour detection.

Hough transform

Hough Transform uses a parametric description of simple geometrical shapes (curves), in order to reduce the computational complexity of the search space.

- The parametric description a straight line is a linear equation:

$$y = ax + b.$$



a) Image plane; b) parameter space.

Hough transform

Hough transform for straight line detection:

- The parameter space is discretized to form a parameter matrix $P(\alpha,b)$, $a_1 \leq \alpha \leq a_k$, $b_1 \leq b \leq b_k$.
- For every pixel $[x_i, y_i]^T$ that possesses value 1 at the binary edge detector output, the equation $b = -\alpha x_i + y_i$ is formed.
- For every parameter value $\alpha, a_1 \leq \alpha \leq a_k$, the corresponding parameter b is calculated and the appropriate parameter matrix element (***bin***) $P(a,b)$ is increased by 1:

$$P(\alpha,b) = P(\alpha,b) + 1.$$

- This process is repeated until the entire binary image is scanned.

Hough transform

- The parametric model has difficulties in representing vertical straight lines, because parameter a must tend to infinity.
- A ***polar representation*** of a straight line can be used instead :

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

- It describes a line having the orientation θ at the distance r from the origin.
- For a binary image of size $N_1 \times N_2$:

$$-\sqrt{{N_1}^2 + {N_2}^2} \leq r \leq \sqrt{{N_1}^2 + {N_2}^2},$$

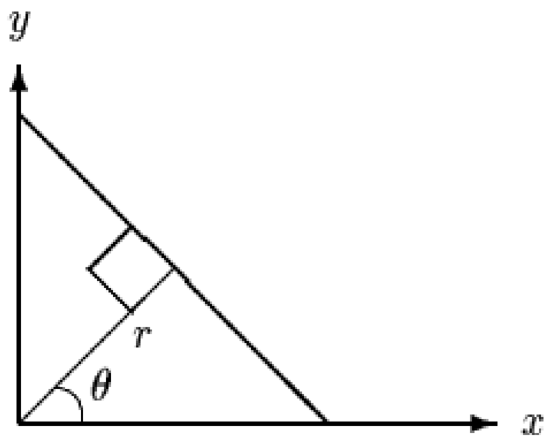
$$-\pi/2 \leq \theta \leq \pi/2.$$

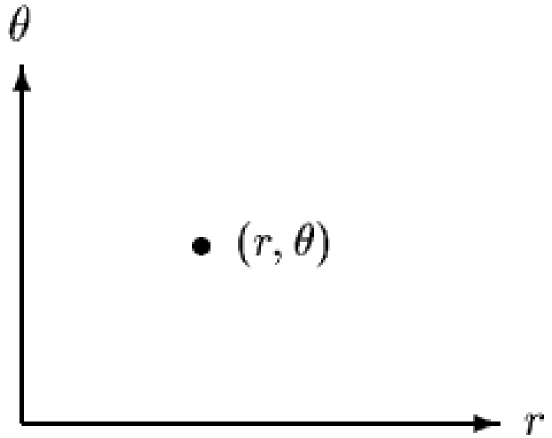
Hough transform

The same Hough transform algorithm can be used by employing the model:

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

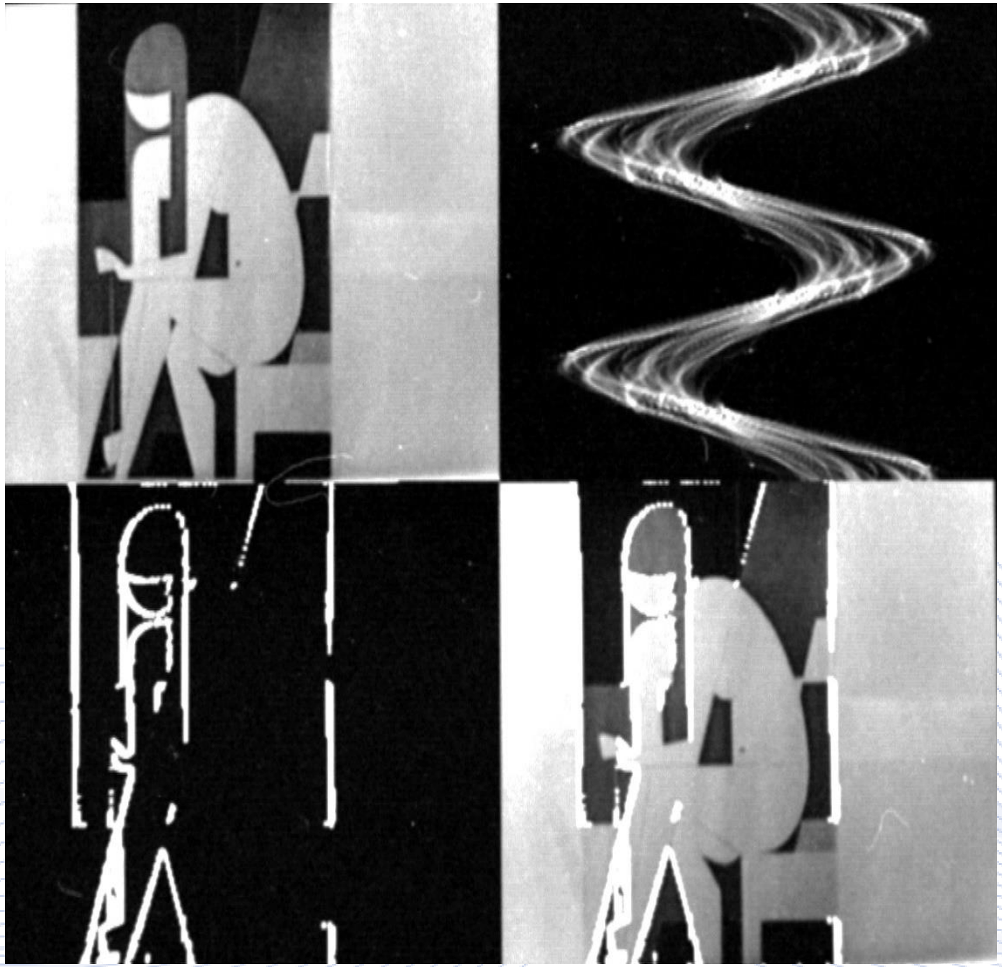
using a parameter matrix $P(r, \theta)$.





a) Polar straight-line representation on the image plane; b) parameter space.

Hough transform



Upper row: a) Original image; b) Hough polar parameter space;
Lower row: c) detected straight lines; d) lines overlaid on original image.

Artificial Intelligence &
Information Analysis Lab

Hough transform

- Local edge direction can be used in the Hough Transform calculation, by reducing a 2D search to a 1D search.
- If both sides of $r = x\cos\theta + y\sin\theta$ are differentiated with respect to x, the following equation gives the line gradient:

$$\frac{dy}{dx} = -\cos\theta = \tan\left(\frac{\pi}{2} + \theta\right),$$

$$\theta = \frac{\pi}{2} - \varphi.$$

- φ: local edge direction.
- The use of the edge gradient reduces the computational complexity of the Hough Transform to the order $O(N)$.

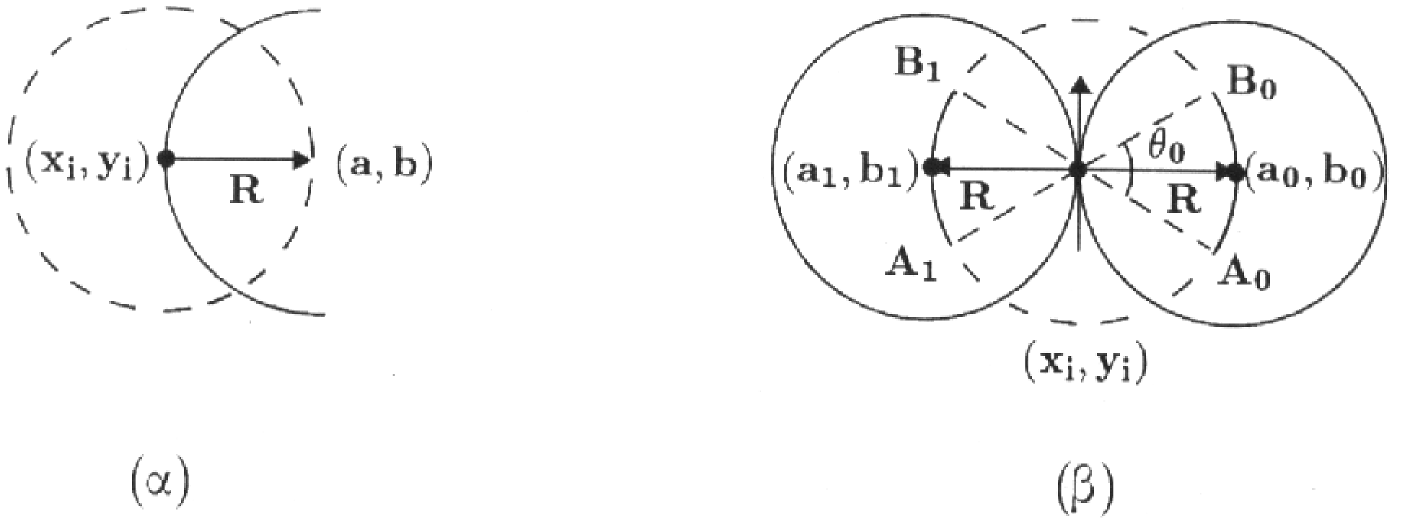
Hough transform

- Hough Transform can be generalized to detect any parametric curves of the form $f(\mathbf{x}, \mathbf{a}) = 0$, where $\mathbf{a}$ is the parameter vector.
- The memory required for the parameter matrix $P(\mathbf{a})$ increases as K^p , where p is the parameter number.
- This method is practical only for curves having a small number of parameters, e.g., for circles:

$$(x-a)^2+(y-b)^2=r^2.$$

- Its parameters are the radius r and the center coordinates (a, b).
- A 3D parameter matrix $P(r, a, b)$ is needed.

Hough transform



a) Locus of circle centers that traverse $[x_i, y_i]^T$; b) Locus of circle centers that traverse $[x_i, y_i]^T$ and are tangent to local edge.

Hough transform

Hough Transform for circles

Let $[x_i, y_i]^T$ be a candidate binary edge image pixel. A circle of radius $r = R$ having center (a, b) and passing through $[x_i, y_i]^T$ is given by:

$$x_i = a + R\cos\theta,$$
$$y_i = b + R\sin\theta.$$

- For any radius r, $0 < r \leq r_{max}$, the coordinates (a, b) are calculated and the corresponding matrix $P(a, b, r)$ elements increase by one.
- These points belong to a cone surface.
- This process is repeated for any eligible pixel of the binary edge detector output.

Hough transform

Hough Transform in Byzantine iconography analysis.

Edge Detection Overview

- Introduction
- Edge detection
- Edge thresholding
- Hough transform
- **Edge following algorithms**
- Contour detection
- Active Contours
- Neural Edge detection
- Neural Contour detection.

Edge-following algorithms

Boundary-following algorithms follow the local edge elements, ensuring local edge continuity.

Edge continuity features:

- $e(\mathbf{x}) = e(x, y)$: edge magnitude at location $\mathbf{x} = [x, y]^T$.
- $\varphi(\mathbf{x}) = \varphi(x, y)$: edge direction.
- $|e(\mathbf{x}_i) - e(\mathbf{x}_j)|$: similarity measure for neighboring edge magnitude.
- $|\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j)|$: direction difference similarity measure.

Edge-following algorithms

Two neighboring edge pixels can be linked (for edge following), if:

$$|e(\mathbf{x}_i) - e(\mathbf{x}_j)| \leq T_1,$$
$$\left|\varphi(\mathbf{x}_i) - \varphi\left(\mathbf{x}_j\right)\right| \mod 2\pi \leq T_2,$$
$$|e(\mathbf{x}_i)| \geq T, \qquad \left|e\left(\mathbf{x}_j\right)\right| \geq T.$$

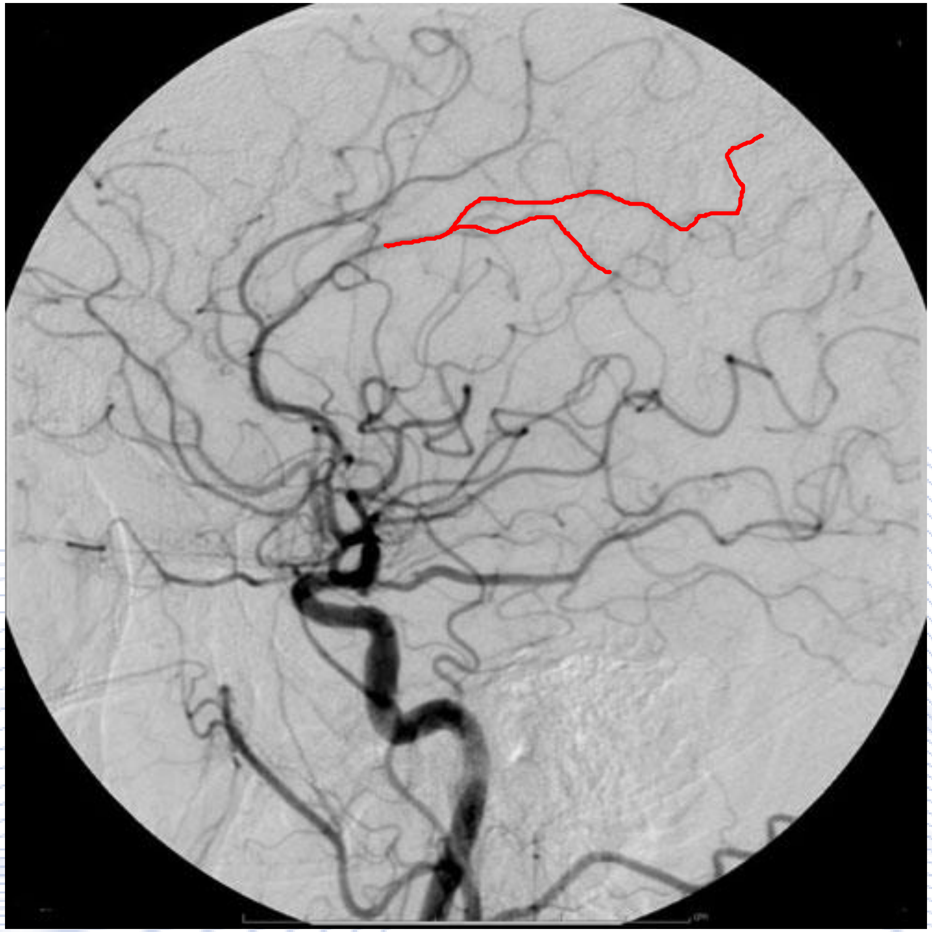
- Edges do not change magnitude and/or direction abruptly.
- Small edge magnitude pixels should not be mistaken as edge elements to be followed.

Edge-following algorithms

Heuristic edge following uses the previous constraints:

- Edge following starts from an edge pixel $\mathbf{x}_A$, satisfying $|e(\mathbf{x}_A)| \geq T$.
- If no neighboring edge pixel satisfies all inequalities, the algorithm stops.
- If more than one neighbor satisfies them, edge pixel $\mathbf{x}_N$ that possesses the minimal differences $|e(\mathbf{x}_N) - e(\mathbf{x}_A)|$, $|\varphi(\mathbf{x}_N) - \varphi(\mathbf{x}_A)|$ is chosen.
- The procedure continues recursively, with the new edge pixel $\mathbf{x}_N$ as a starting element.

Edge-following algorithms

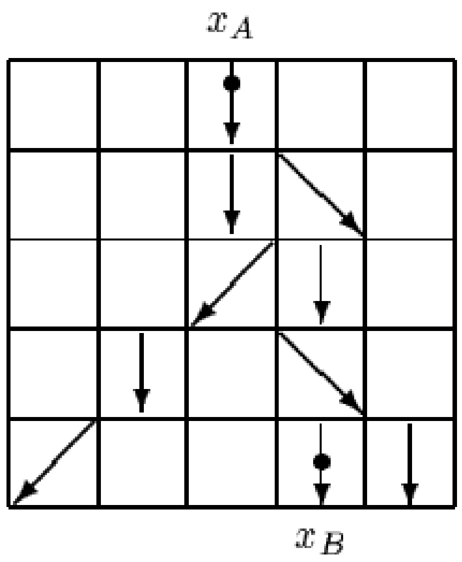


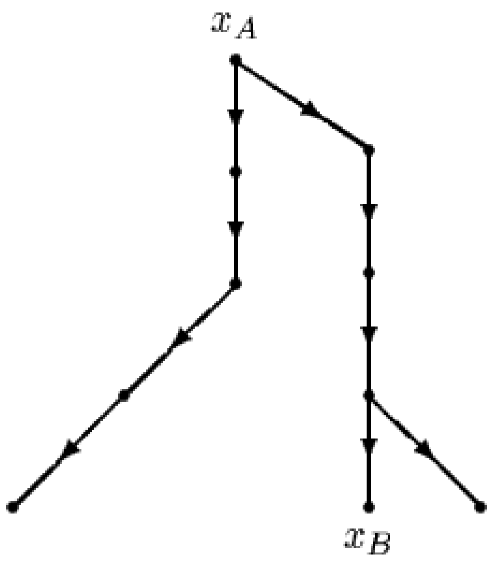
Heuristic contour following in subtractive angiography [Wikipedia].

Edge-following algorithms

Edge following can be based on ***graph search***:

- Edge elements at position $\mathbf{x}_i$ can be considered as graph nodes.
- The nodes are connected to each other, if local edge linking rules are satisfied.





Edge-following algorithms

- Let us suppose we form a ***cost function*** $C(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ for a path connecting nodes $\mathbf{x}_1 = \mathbf{x}_A$ to $\mathbf{x}_N = \mathbf{x}_B$:

$$C(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$$
$$\triangleq -\sum_{k=1}^{N} |e(\mathbf{x}_k)| + a\sum_{k=2}^{N} |\theta(\mathbf{x}_k) - \theta(\mathbf{x}_{k-1})| + b\sum_{k=2}^{N} |e(\mathbf{x}_k) - e(\mathbf{x}_{k-1})|.$$

- The heuristic graph search algorithm tries to produce a minimum cost path from $\mathbf{x}_A$ to $\mathbf{x}_B$.
- The algorithm is based on the cost function and on the choice of the successors of a node $\mathbf{x}_i$, by using edge linking criteria.

Edge-following algorithms

Basic disadvantages of the heuristic graph search algorithm:

- The need to keep track of all current best paths.

- Short paths (close to the origin) may have smaller cost than longer paths that are more likely to be the final winners.

Edge-following algorithms

Edge following based on ***dynamic programming***:

- The optimal path between two nodes $\mathbf{x}_A, \mathbf{x}_B$ of an edge graph consists of optimal subpaths for any node lying on it.
- Thus, the optimal path between two nodes $\mathbf{x}_A, \mathbf{x}_B$ can be split into two optimal subpaths $\mathbf{x}_A\,\mathbf{x}_i$ and $\mathbf{x}_i\,\mathbf{x}_B$ for any $\mathbf{x}_i$ lying on the optimal path $\mathbf{x}_A\,\mathbf{x}_B$.
- Following objective function to be maximized:

$$\mathrm{F}(\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N) = \sum_{k=1}^{N} |e(\mathbf{x}_k)| - a \sum_{k=2}^{N} |\theta(\mathbf{x}_k) - \theta(\mathbf{x}_{k-1})|$$

Edge-following algorithms

- Start and target nodes: $\mathbf{x}_1 = \mathbf{x}_A$ and $\mathbf{x}_N = \mathbf{x}_B$.

- The target function F can be written:

$$F(\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N) = F(\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_{k-1}) + f(\mathbf{x}_{k-1}, \mathbf{x}_k),$$

where:

$$f(\mathbf{x}_{k-1}, \mathbf{x}_k) = |e(\mathbf{x}_k)| - a|\theta(\mathbf{x}_k) - \theta(\mathbf{x}_{k-1})|.$$

Edge-following algorithms

- The optimal path $\hat{\mathbf{x}}_1\hat{\mathbf{x}}_k$ can be divided into two optimal paths $\hat{\mathbf{x}}_1\hat{\mathbf{x}}_{k-1}$ and $\hat{\mathbf{x}}_{k-1}\hat{\mathbf{x}}_k$ that satisfy the following relation:

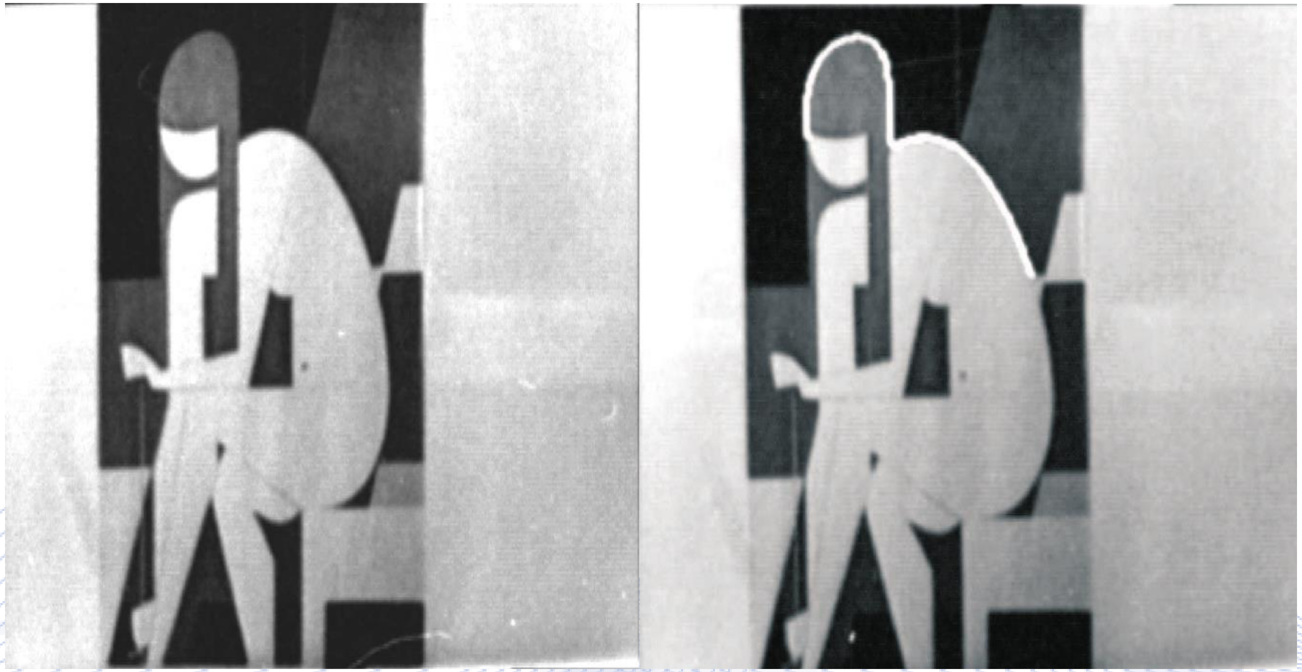
$$\hat{F}(\hat{\mathbf{x}}_1,..,\hat{\mathbf{x}}_k) = \max_{\mathbf{x}_{i,i=1,..,k}} F(\mathbf{x}_1,..,\mathbf{x}_k) =$$

$$\max_{\mathbf{x}_{i,i=1,..,k}} \{F(\mathbf{x}_1,..,\mathbf{x}_{k-1}) + f(\mathbf{x}_{k-1},\mathbf{x}_k)\} =$$

$$\max_{\mathbf{x}_k}\{\hat{F}(\hat{\mathbf{x}}_1,..,\hat{\mathbf{x}}_{k-1}) + f(\hat{\mathbf{x}}_{k-1},\hat{\mathbf{x}}_k)\}.$$

- The initial value of $\hat{F}(\hat{\mathbf{x}}_1)$ is given by: $\hat{F}(\hat{\mathbf{x}}_1)= |e(\mathbf{x}_1)|$.
- N independent optimization steps: In every step, we are looking for nodes $\mathbf{x}_k$ such that the objective function $\hat{F}(\hat{\mathbf{x}}_1,..,\hat{\mathbf{x}}_k)$ to be maximized.

Edge-following algorithms



Edge following based on dynamic programming: a) original image; b) edge following result.

Edge Detection Overview

- Introduction
- Edge detection
- Edge thresholding
- Hough transform
- Edge following algorithms
- **Contour detection**
- Active Contours
- Neural Edge detection
- Neural Contour detection.

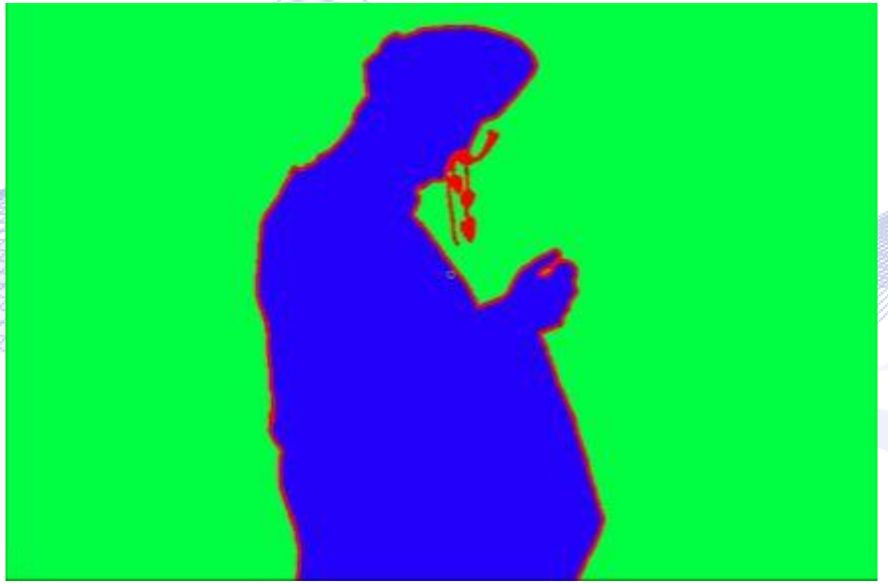
Contour detection

- Edge detection aims at detecting characteristic image changes in image intensity and/or color.

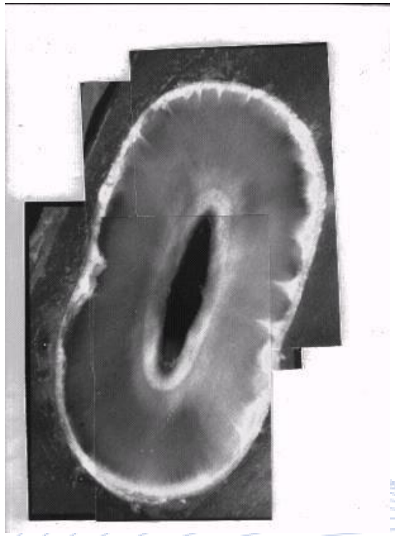
- An ***object contour*** is a typically closed curve enclosing all image object pixels , i.e., image pixels having same color or intensity or texture.
- ***Contour detection*** aims to find pixel label changes at the border between two image objects.
  - Typically, a binary classifier determines whether an image pixel belongs to a contour.

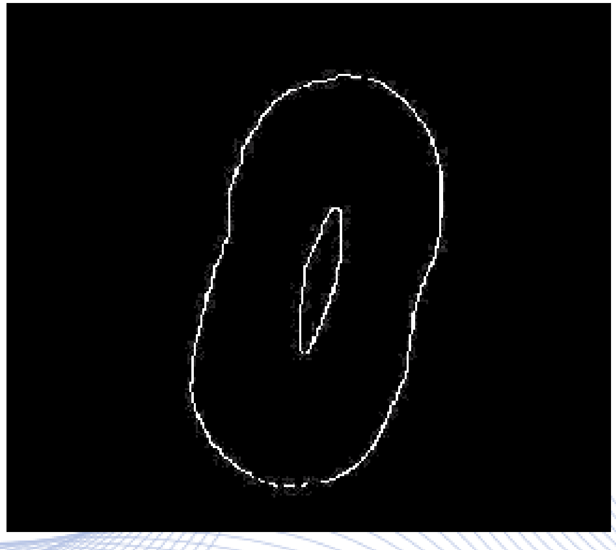
Contour detection

- Contour detection is more difficult than edge detection.
- It is useful for shape analysis and object recognition.
- Simplest contour description: ordered list of contour pixels $[x_i, y_i]^T, i = 1, \dots N$.



Contour following





a) Tooth cross-section mosaic; b) tooth and oral cavity contour following.

Contour following algorithms

- Binary valued digital image $\mathcal{X}$.
- A pixel $\mathbf{x}$ is equaled to one when it belongs to the pattern (black pixel) or zero when it is part of the background (white pixel).
- ***Contour***: list of black pixels that are ***connected*** to each other (forming pixel sequence $\mathcal{B}$).
  - Types of contour pixel: 4-border and 8-border.

Contour following algorithms

Square Tracing Algorithm

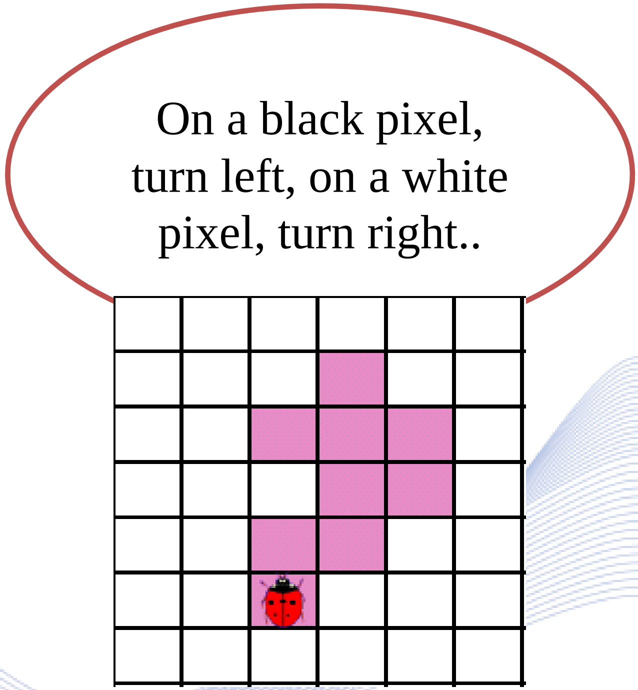
- It is one of the first attempts to extract the contour of a binary pattern.
- Input: A binary image $\mathcal{X}$, containing one object (connected component) $\mathcal{P}$ of black pixels in a background of white pixels.
- Output: A sequence $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_k\}$ of contour pixels.

- At algorithm start, set $\mathcal{B}$ as empty.

Contour following algorithms

Square Tracing Algorithm

- Start at ***starting pixel*** $\mathbf{p}$: Scan each pixel column from bottom to top and left to right, until encounter a black pixel $\mathbf{x}$ belonging to $\mathcal{P}$ and insert $\mathbf{x}$ in $\mathcal{B}$. The starting pixel $\mathbf{p}$ is $\mathbf{x}$ and the current pixel $\mathbf{x}$ is the left to the previous one.
- If you find black pixel $\mathbf{x}$, turn left and if you find a white one, turn right in a square clockwise motion until you find a black pixel $\mathbf{x}$ again.



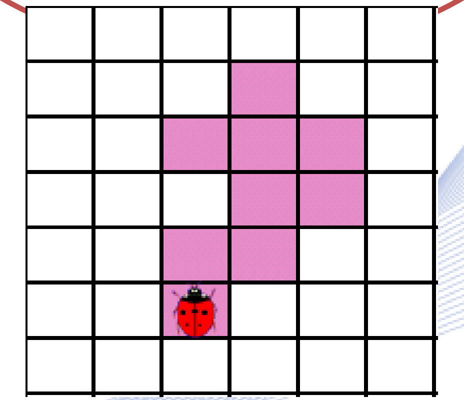
[GHU2000]

Contour following algorithms

- The algorithm stops when you encounter the starting pixel again.

- The black pixels you walked over will be the contour of the pattern.

On a black pixel, turn left, on a white pixel, turn right..

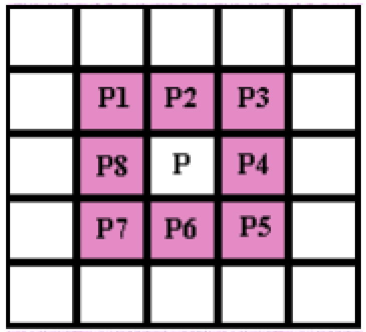


[GHU2000]

Contour following algorithms

Moore-Neighbor Tracing Algorithm

- ***Moore neighborhood*** of a pixel $\mathbf{p}$ is the set of 8 pixels $M = \{\mathbf{p}_1, \ldots, \mathbf{p}_8\}$, which shares a vertex or edge with that pixel.



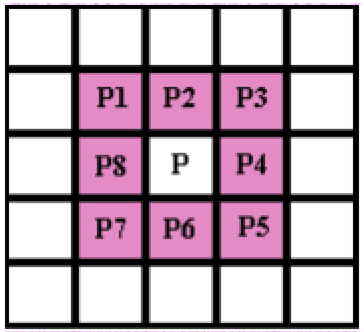
[GHU2000]

- Input: A binary image $\mathcal{X}$, containing one object (connected component) $\mathcal{P}$ of black pixels in a background of white pixels.
- Output: A sequence $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ of contour pixels.

Contour following algorithms

- At algorithm start, set $\mathcal{B}$ as empty.

- Start at ***starting pixel*** $\mathbf{p}$ **:** Scan each pixel of $\mathcal{P}$ from bottom to top and left to right until encounter a black pixel $\mathbf{x}$. The starting pixel $\mathbf{p}$ is $\mathbf{x}$ and the current pixel is the white pixel next to it which belongs to M.
- Insert the starting pixel to $\mathcal{B}$.



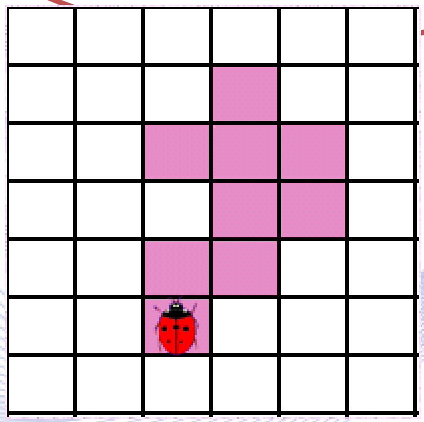
[GHU2000]

Contour following algorithms

- If $\mathbf{x}$ is a black pixel, let this pixel be the starting pixel $\mathbf{p}$ and until we find a black pixel $\mathbf{x}$ again, define as the current pixel the white pixel next to it which belongs to Moore neighborhood M. This continues, until the starting pixel is visited for a second time.

- The walked over black pixels will be the object contour.

Every time you hit a black pixel, backtrack, go around pixel in a clockwise direction until you hit a black pixel.

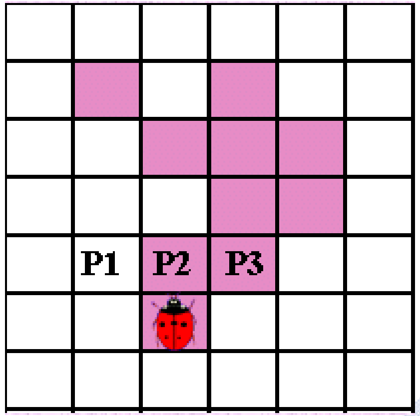


[GHU2000]

Contour following algorithms

Theo Pavlidis' Algorithm

- Let 3 image pixels be denoted by: $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$.
- Input: A binary image $\mathcal{X}$, containing one object (connected component) $\mathcal{P}$ of black pixels in a background of white pixels.
- Output: A sequence $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_k\}$ of contour pixels.
-

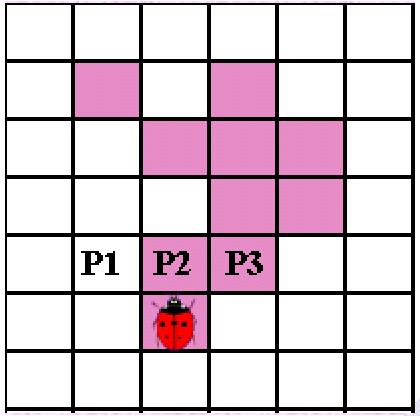


[GHU2000]

Contour following algorithms

- At algorithm start, set $\mathcal{B}$ is empty.

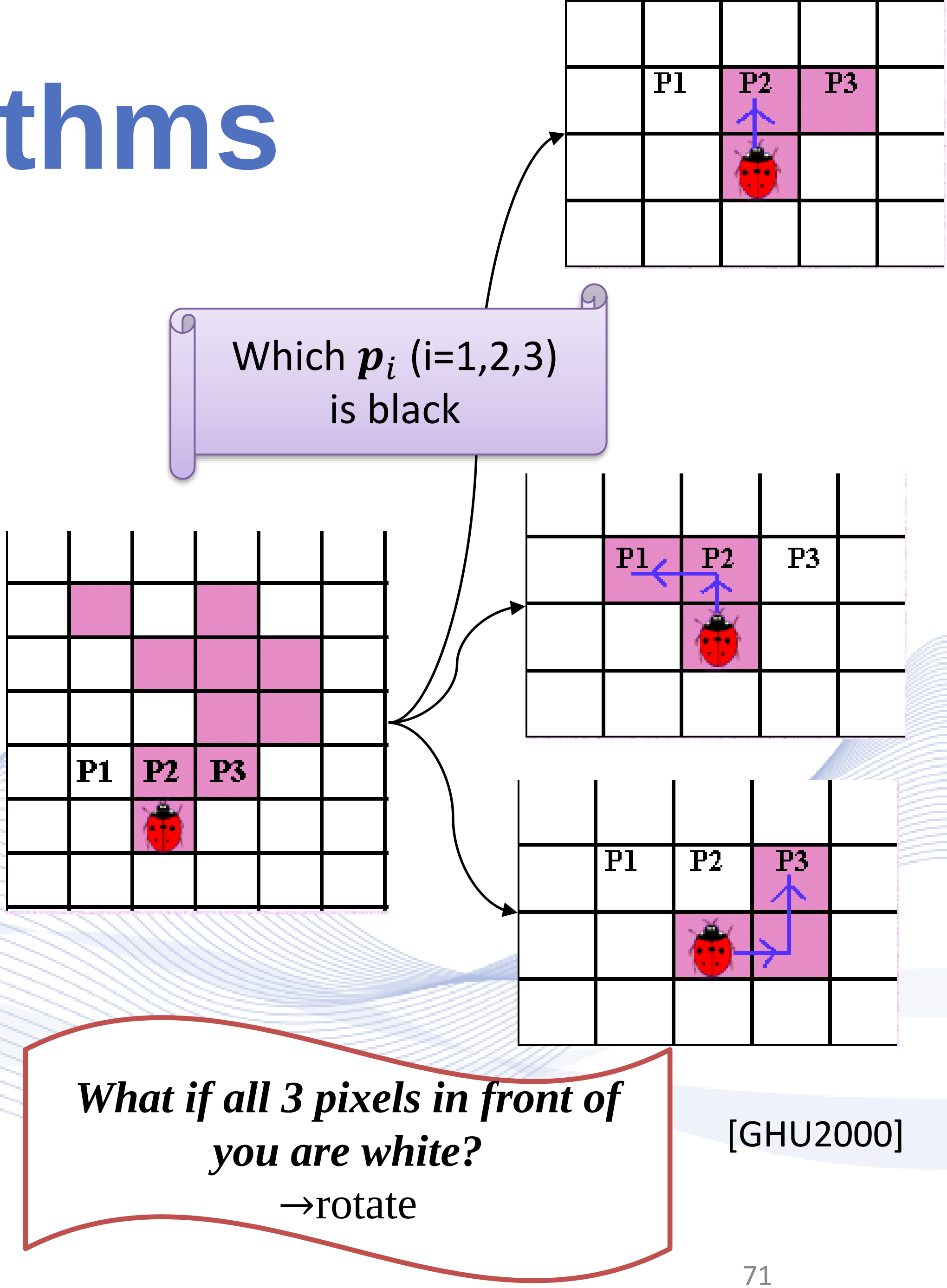
- Start at ***starting pixel*** $\mathbf{p}$: Scan each pixel column of from bottom to top and left to right until encounter a black pixel $\mathbf{x}$, whose left adjacent pixel is white. Now the start pixel $\mathbf{p}$ is $\mathbf{x}$. Insert the start pixel $\mathbf{p}$ in $\mathcal{B}$.
- If $\mathbf{p}_1$ is black, insert it to $\mathcal{B}$ and move one step forward followed by one step to your current left pixel to land on $\mathbf{p}_1$.



[GHU2000]

Contour following algorithms

- If $\mathbf{p}_2$ is black, insert it to $\mathcal{B}$ and move one step forward to land on $\mathbf{p}_2$.
- If $\mathbf{p}_3$ is black, insert it to $\mathcal{B}$ and move one step to your right followed by one step to your current left.
- If they are all white, rotate.
- Stop when you have rotated 3 times, or the start pixel is visited for a second time.
- The contour will be the black pixels in $\mathcal{B}$.



[GHU2000]

Edge Detection Overview

- Introduction
- Edge detection
- Edge thresholding
- Hough transform
- Edge following algorithms
- Contour detection
- **Active Contours**
- Neural Edge detection
- Neural Contour detection.

Active Contours

- ***Active contours*** or ***snakes*** are deformable models of an image contour.
- They describe object boundaries/contours by a parametric curve.
- An ***energy functional*** is always associated with an active contour.
- The desired contour is obtained by defining ***energy functional minimization***.

Active Contours

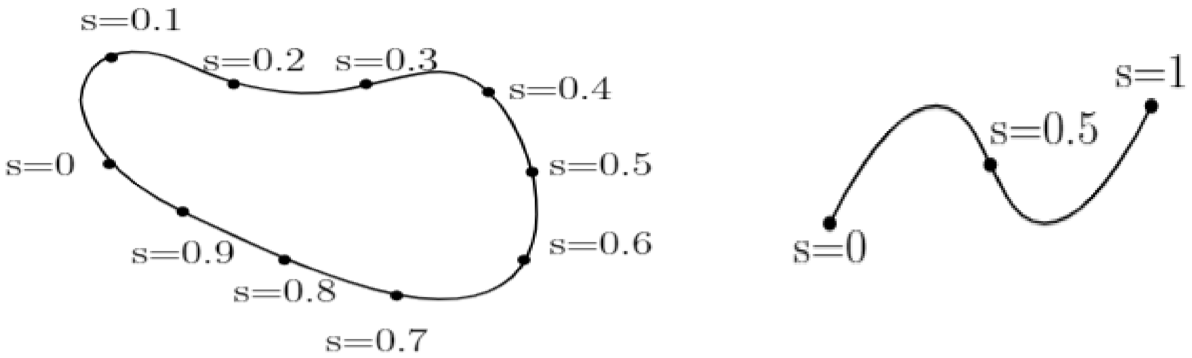
A curve can be represented by a vectorial function.

- In the continuous case:

$$\mathbf{v}(s) = [x(s), y(s)]^T, \ \ 0 \leq s \leq 1.$$

- In the discrete space case, a contour is described by a vertex list:

$$C = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, .., \mathbf{v}_{n-1}\}, \qquad \mathbf{v}_i = [x_i, y_i]^T.$$



a) Closed curve; b) Open curve.

Active Contours

- An active contour possesses energy E(***energy functional***), which is defined as the sum of the three energy terms:

$$E = E_i + E_e + E_c = \int_0^1 \Big(E_i\big(\mathbf{v}(s)\big) + E_e\big(\mathbf{v}(s)\big) + E_c\big(\mathbf{v}(s)\big) \Big) ds.$$

- E_i: ***Internal energy*** due to contour bending. It serves to impose piecewise contour smoothness constraint.

Artificial Intelligence & Information Analysis Lab

Active Contours

- E_e: ***External energy*** that describes how well the contour matches local image data.
  - Numerous forms can be used, attracting the curve toward specific image features, e.g., local image edges.2

- E_c: ***External constraints*** are responsible for putting the snake near the desired local minimum (optional).

Active Contours

Internal Energy

$$E_i\big(\mathbf{v}(s)\big) = \alpha(s)|d\mathbf{v}/ds|^2 + \beta(s)|d^2\mathbf{v}/ds^2|^2.$$

- $d\mathbf{v}/ds$ is the first order derivative, forcing the contour to act like a ***membrane***.
- $d^2\mathbf{v}/ds^2$ is the second order derivative, forcing the contour to act like a ***thin-plate***.
- $\alpha(s)$ and $\beta(s)$ controls the relative importance of membrane and thin-plate terms: ***elastic/stretching*** and ***stiffness/bending***.

Artificial Intelligence & Information Analysis Lab

Active Contours

- ***Smoothness*** of the whole snake:

$$E_i = \int_0^1 E_i\big(\mathbf{v}(s)\big)ds$$

- In the discrete space case, numerical differentiation can be performed:

$$d\mathbf{v}/ds \cong \mathbf{v}_{i+1} - \mathbf{v}_i$$
$$d^2\mathbf{v}/ds^2 \cong \mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}$$

- Internal energy is given by:

$$E_i = \sum_{i=0}^{n-1} \alpha|\mathbf{v}_{i+1} - \mathbf{v}_i|^2 + \beta|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}|^2 .$$

Active Contours

External Energy

- Image edges are described by image gradient $\nabla I(x,y)$.
- External energy at a contour point $\mathbf{v}(s)$ is given by:

$$E_e(\mathbf{v}(s)) = -|\nabla I(x,y)|^2.$$

- and or the whole snake:

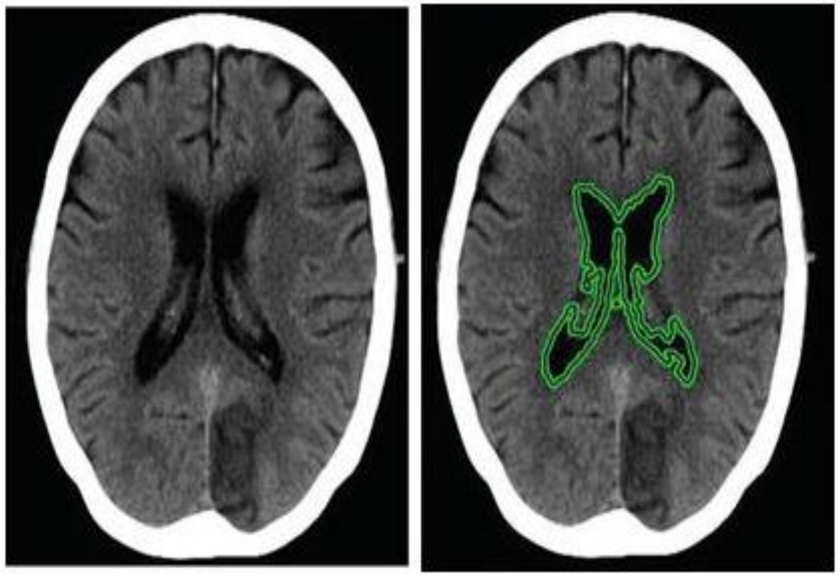
$$E_e = \int_0^1 E_e\big(\mathbf{v}(s)\big)ds \text{ (continuous case)}$$

$$E_e = \sum_{i=0}^{n-1} E_e(\mathbf{v}_i) \text{ (discrete case).}$$

- Simplified version of the total energy:

$$E = \alpha \sum_{i=0}^{n-1} |\mathbf{v}_{i+1} - \mathbf{v}_i|^2 - \sum_{i=0}^{n-1} |\nabla I(\mathbf{v}_i)|^2.$$

Active Contours



[HEM2018]

Active contour on a brain CT image.

Edge Detection Overview

- Introduction
- Edge detection
- Edge thresholding
- Hough transform
- Edge following algorithms
- Contour detection
- Active Contours
- **Neural Edge detection**
- Neural Contour detection.

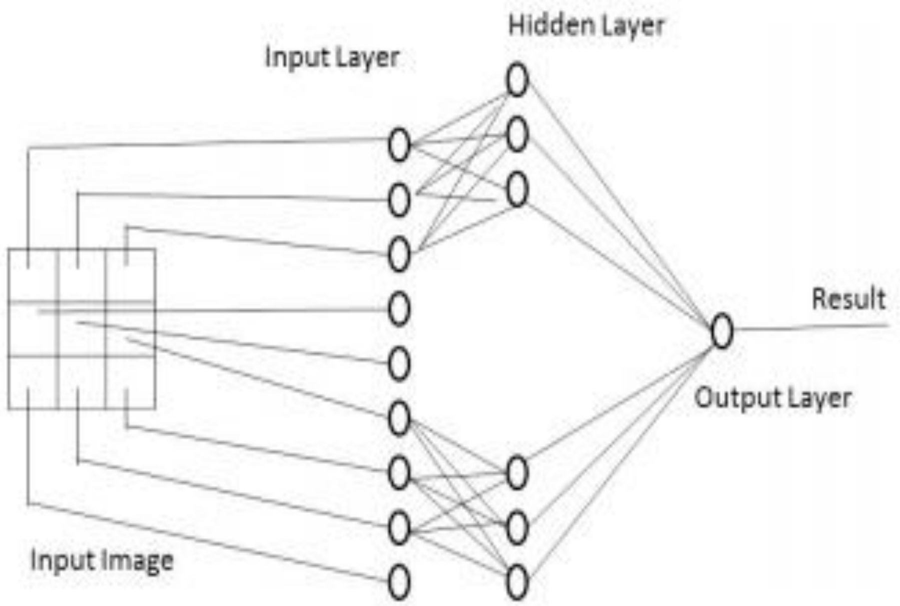
NN Edge detection

- A ***Neural Network*** (NN) edge detector can be considered as a nonlinear filter: it can have a built-in thresholding capability.
- Thus, the filtering, thresholding operation of edge detection is a natural application for neural network processing.

Convolutional Neural networks (***CNN***) have convolutional layers and nonlinear activation functions interspersed with pooling (subsampling) layers.

- Typical CNN convolution kernels perform edge detection (learned only by training).

NN Edge detection

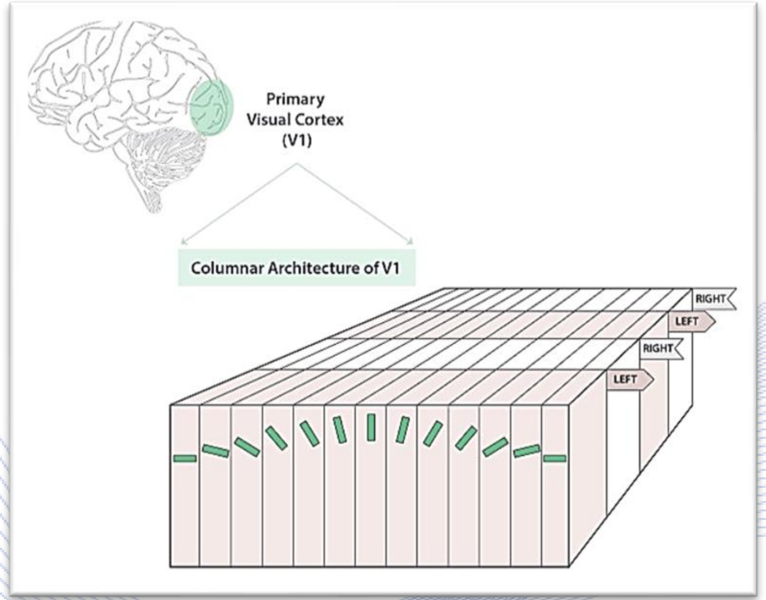


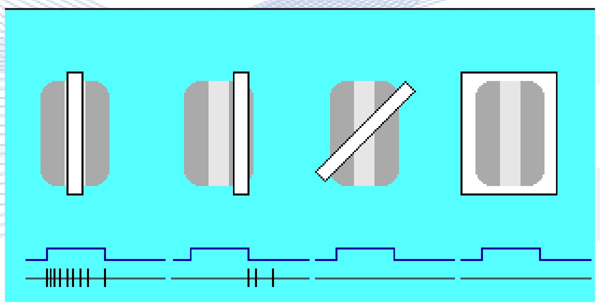
Neural Network Architecture for image edge detection [SEN2012].

NN Edge detection

Biological V1 Hypercolumn:

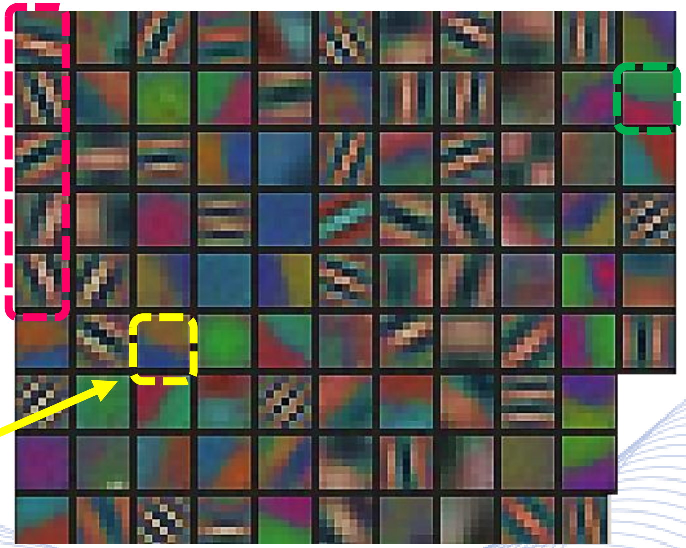
- CNNs were inspired by brain neurons in the mammalian ***primary visual cortex*** (V1).
- V1 cells are mapped to the same local region of the retina, forming ***hypercolumns***.
- Hidden layers are similar to V1 simple cells, detect image lines and are sensitive to orientation.





NN Edge detection

orientation selectivity
found in V1 simple cells



green/red color opponency
observed in retinal neurons
and human visual perception

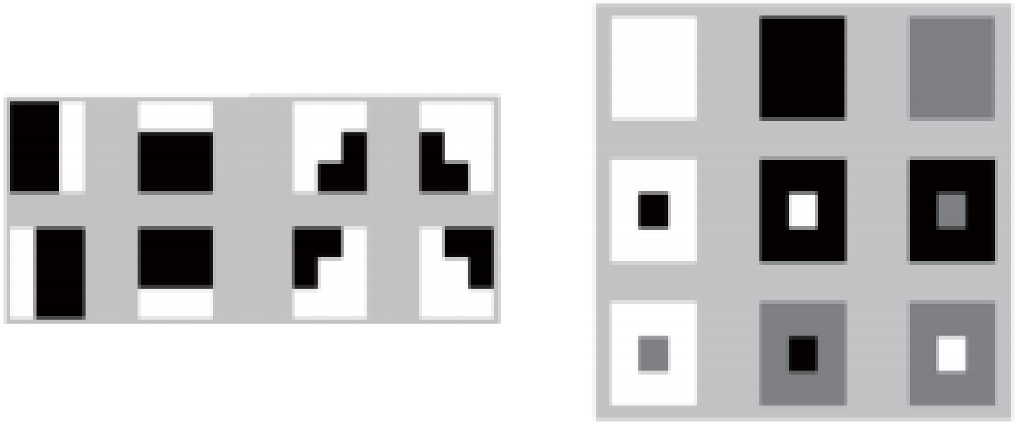
blue/yellow color opponency
observed in retinal neurons
and human visual perception

ZFNet convolution kernels that have been produced by training to perform edge/line detection.

NN Edge detection

CNN training for edge detection:

- 17 spatial image patterns are considered (8 edge and 9 non-edge patterns).
- Edge thresholding implemented through sigmoid activation functions.



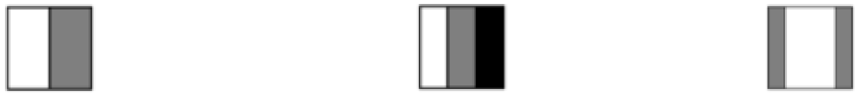
a) Edge Training Patterns; b) Non edge Training Patterns [MOH2013].

NN Edge detection

goal is to learn 9 parameters

By just treating these 9 numbers as parameters the backprop can choose to learn 1,1,1 or −1,−1,−1 or learn the Sobel filter or Scharr filter.

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}_{6 \times 6} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}_{3 \times 3} = \begin{bmatrix} -0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}_{4 \times 4}$$



Edge detection – an original image (left), a filter (in the middle), a result of a convolution (right)

How to choose weights in the filter?

convolution

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix}_{6 \times 6} * \begin{bmatrix} W_1 & W_1 & W_1 \\ W_1 & W_1 & W_1 \\ W_1 & W_1 & W_1 \end{bmatrix}_{3 \times 3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 30 & 10 & -10 & -30 \\ 30 & 10 & -10 & -30 \\ 0 & 0 & 0 & 0 \end{bmatrix}_{4 \times 4}$$

W_i=weights→edge templates

We can make our algorithm to learn parameters of the filter

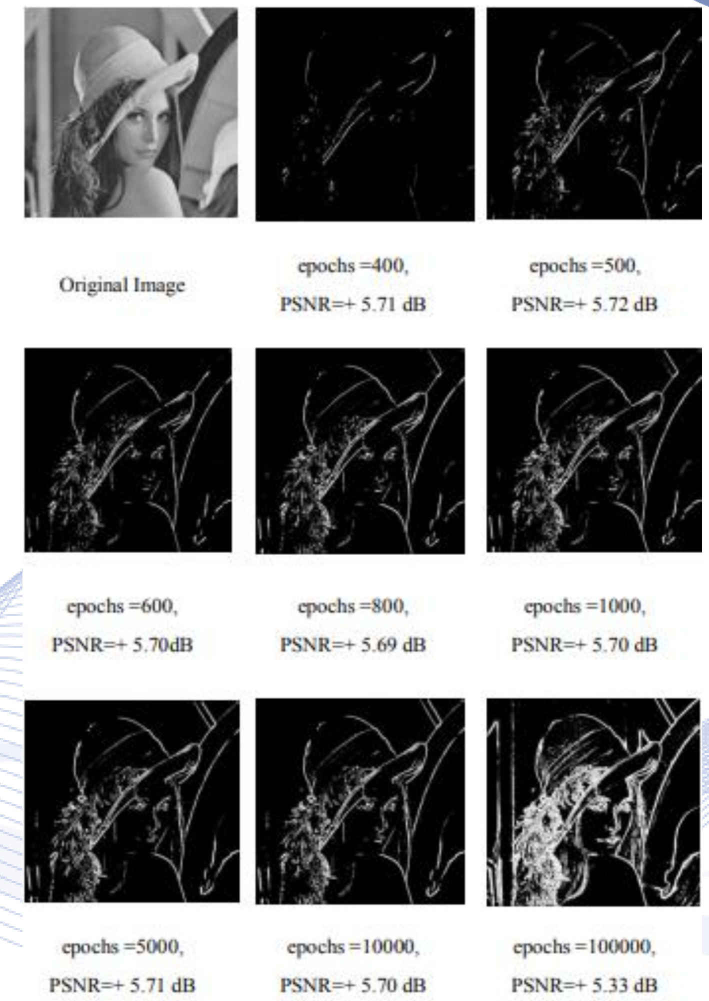
We can also learn to detect edges there at 45° or 70° or 73° or any other orientation it chooses.

Artificial Intelligence & Information Analysis Lab

NN Edge detection

- The best Peak signal-to-noise ratio (PSNR) is obtained when the test image is applied for the maximum epochs trained network.

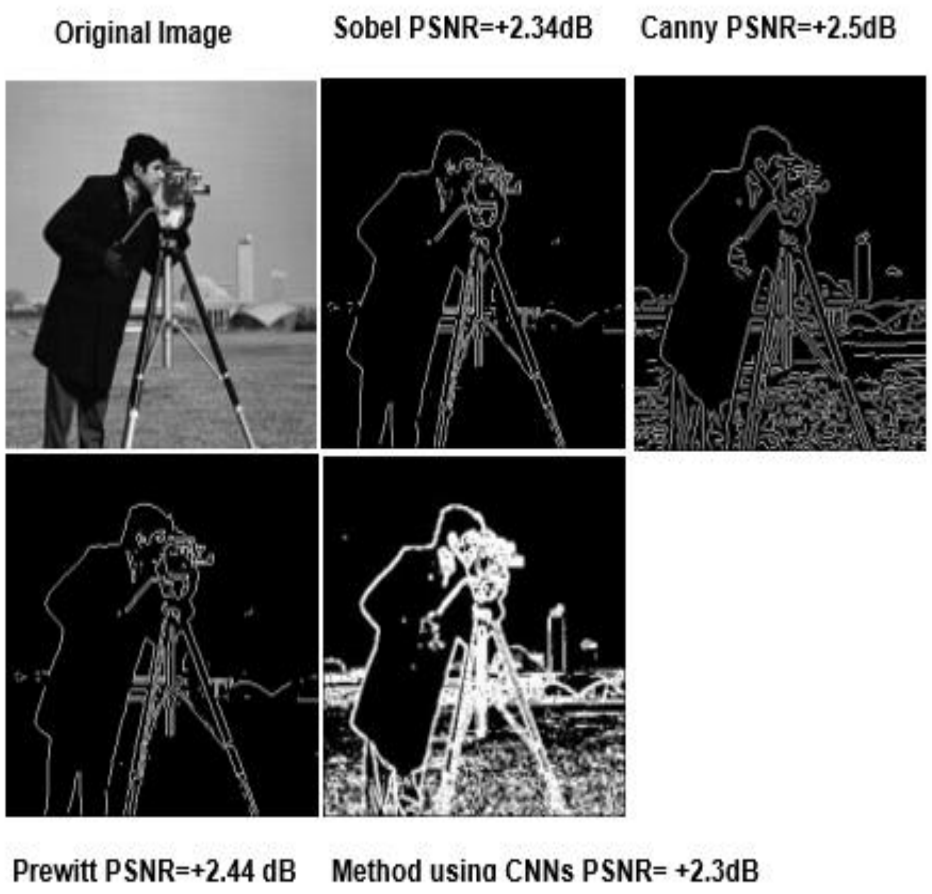
- NN edge detection is better than other edge detection methods.
- It detects more true edge pixels and produces little edge noise.



[MOH2013]

Artificial Intelligence & Information Analysis Lab

NN Edge detection

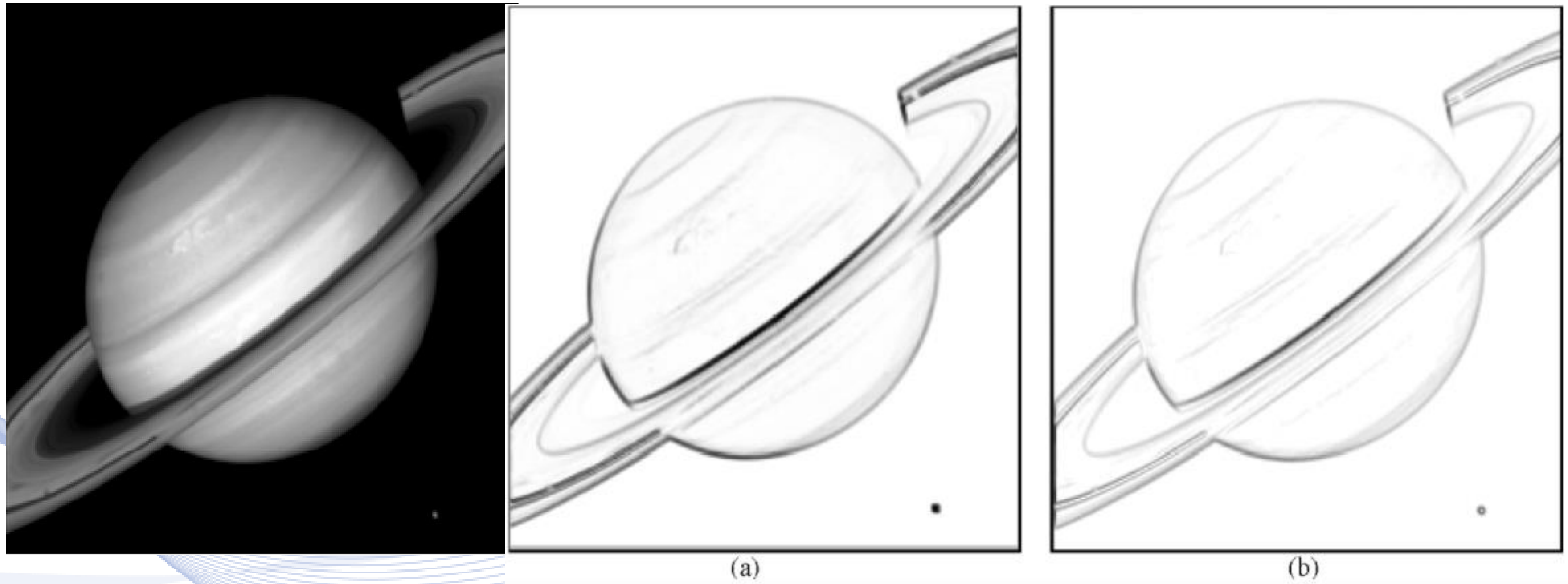


[MOH2013]

Artificial Intelligence &
Information Analysis Lab

SVM Edge Detection

- Binary SVM classification:
  - ‘the pixel is part of an edge’ or not.



a) SVM edge image; b) Sobel edge image [GOHA2000].

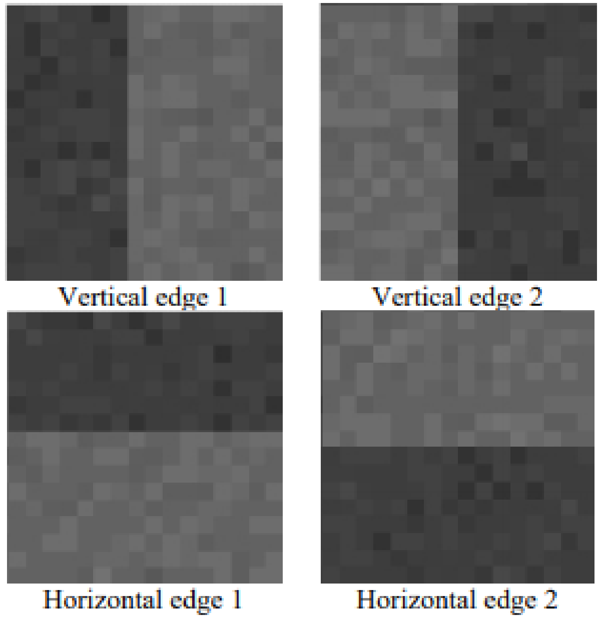
SVM Edge Detection

- Input to the SVM: a vector which is formed for each pixel given the difference between this one and the pixels in its 3×3 neighborhood.

- In Training: horizontal and vertical edges are used.
    - The other edges will be generalized by the SVM.

- The pixels considered as edges are those into each image that are in the border between bright and dark zones.

SVM Edge Detection



Training images [GOH2000].

Edge Detection Overview

- Introduction
- Edge detection
- Edge thresholding
- Hough transform
- Edge following algorithms
- Contour detection
- Active Contours
- Neural Edge detection
- **Neural Contour detection.**

NN Contour detection

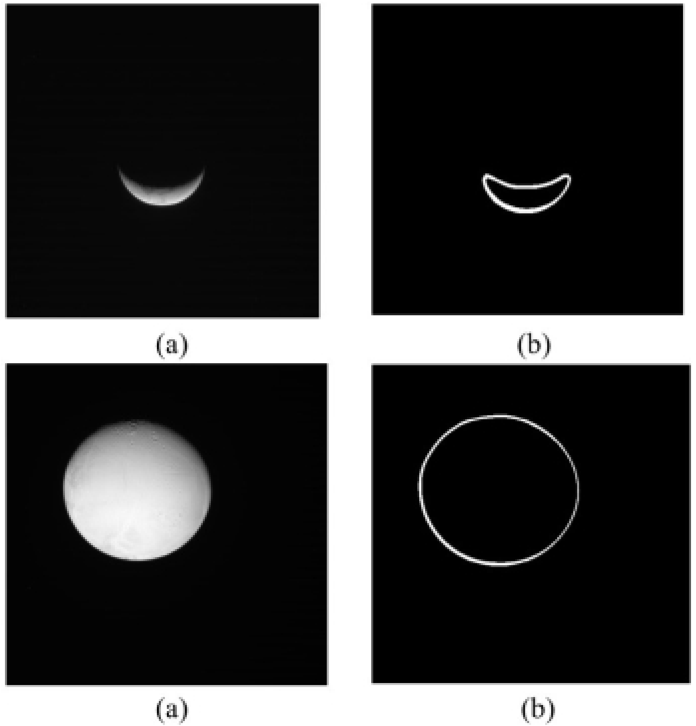
Contour detection can be considered as a classification task:

- Classify a pixel as contour or non-contour one.

- Contour detection can be achieved by sliding-window strategy:
  - CNN image features are extracted in which each image window, to be followed by classification.
  - Pixels as features: number of inputs neurons.

- Any classifier, e.g., random forest classifier, can be used to predict whether the central pixel of this local image window is a contour point or not.

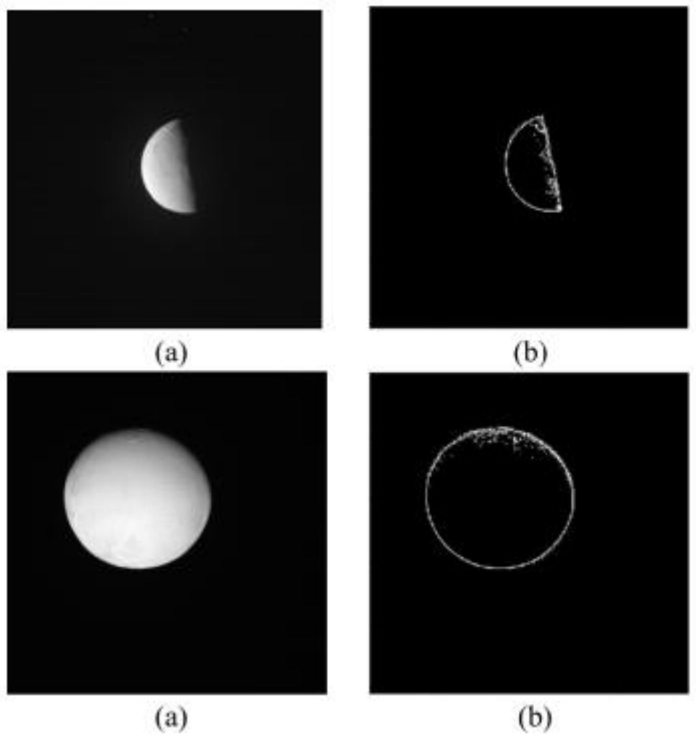
NN Contour detection



a) Original Image; b) CNN contour detection in Cassini ISS images [LI2019].

SVM Contour detection

- We can use SVMs for binary classification.
  - The SVM is connected to neural network with two fully connected layers.



a) Original Image; b) CNN contour detection in Cassini ISS images [LI2019].

Bibliography

[AMM1987] L. Ammerdaal, *Programs and data structures in C*, Wiley,1987

[BAL1982] D.H. Ballard, C.M. Brown, *Computer vision*, Prentice-Hall, 1982

[BEL1962] R.E. Bellman, S. Dreyfus, *Applied dynamic programming*, Princeton University Press, 1962

[CHA1997] V. Chatzis, I. Pitas, “Fuzzy Cell Hough Transform for Curve Detection”, Patern Recognition, Elsevier, vol. 30, pp. 2031-2042, 1997

[DAV1981] H.A. David, *Order statistics*, Wiley, 1981

[GON1987] R.C. Gonzalez, P. Wintz, *Digital image processing*, Addison-Wesley, 1987

[HAR1973] R.M. Haralick, K. Shanmugan, “Computer classification of reservoir sandstones”, IEEE Transactions on Geoscience Electronics. vol. GE-11, no. 4, pp. 171-177, Oct. 1973

[HAR1975] R.M. Haralick, I. Dinstein, “A spatial clustering procedure for multiimage data”, IEEE Transactions on Circuits and Systems, vol. CAS-22, no. 5, pp. 440-450, May 1975

[JAI1989] A.K. Jain, *Fundamentals of digital image processing*, Prentice-Hall, 1989

[LEV1985] M.D. Levine, *Vision in man and machine*, McGraw-Hill, 1985

[MAR1982] A. Martelli, “Edge detection using heuristic search methods”, Computer Graphics and Image Processing, vol. 1, no. 2, pp.169-182, Aug. 1982

Bibliography

[NIK1998] N. Nikolaidis and I. Pitas, “Nonlinear processing and analysis of angular signals”, IEEE Trans. on Signal Processing, vol.46, no. 12, pp. 3181-3194, December 1998

[NIL1980] N.J. Nilson, *Principles of artificial intelligence*, Tioga,1980

[PIT1986] I. Pitas, A.N. Venetsanopoulos, “Edge detectors based on order statistics”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 4, pp. 538-550, July 1986

[PIT1990] I. Pitas, A.N. Venetsanopoulos, *Nonlinear digital filters: Principles and applications*, Kluwer Academic, 1990

[SCH1989] R.J. Schalkof, *Digital image processing and computer vision*, Wiley, 1989

[GOH2000] H. Gómez-Moreno, S. Maldonado-Bascón, P. Martín-Martín , Edge Detection by Using the Support Vector Machines ,2000

[JIA2018] S. Jiao Tong University, Chapter 10 Image Segmentation. Retrieved from https://www.slideshare.net/UlaBac/lec11-active-contour-and-level-set-for-medical-image-segmentation,2018

[CHA2016] S. Chae, C. Cheong, J. Seo, D. Kim , J. Shim, D. Kim, and T. *Han, Fast contour-tracing algorithm based on a pixel-following method for image sensors*, 2016

[FOT2013] S. Fotopoulos, *Edge Detection, Retrieved from*

http://www.hep.upatras.gr/class/download/psi_epe_iko/kef4.pdf,2013

[GHU2000] A. G. Ghuneim ,*Contour Tracing Algorithms*. Retrieved from

http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/alg.html, 2000

Bibliography

[KAS1988] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, International Journal of Computer Vision, 1988

[HEM2018] R.J. Hemalatha, T.R. Thamizhvani, A. Josephin Arockia Dhivya, Josline Elsa Joseph, Bincy Babu and R. Chandrasekaran, *Active Contour Based Segmentation Techniques for Medical Image Analysis*, 2018

[BAG2017] U. Bağcı, *Active Contour and Level Set for Medical Image Segmentation*,2017

[WIK2020] Wikipedia, *Active contour model*, last edited on 10 September 2020

[KAT1988] Z. Kato ,*Snakes: "*Active Contours, International Journal of Computer Vision, International Journal of Computer Vision", Vol. 1, pp 321 Vol. 1, pp 321-331, 1988

[SEN2012] Dr. N. Senthilkumaran, *Edge Detection for Dental X-ray Image Segmentation using Neural Network approach* ,Volume 1, No. 7, September 2012 ISSN – 2278-1080

[MOH2013] Mohamed A. El-Sayed, Yarub A. Estaitia, Mohamed A. Khafagy, *Automated Edge Detection Using Convolutional Neural Network International Journal of Advanced Computer Science and Applications*, Vol. 4, No. 10, 2013

Bibliography

[ZHA2017] Z. Zhang, H. Sua , X. Shia , F. Xinga , L. Yanga , *Recent Advances in the Applications of Convolutional Neural Networks to Medical Image Contour Detection*, University of Florida, 2017

[BAD2017] V.Badrinarayanan, R. Cipolla, A. Kendall, *Computer Science, Medicine*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017

[JOR2018] D. Jordacevic, S. Stefanovic, S. Zivkovic, *CNN More On Edge Detection*, 2018

[AHM2013] G. Ahmadi, K. Inthavong, J. Tu, *Reconstruction of the Human Airways*, In book: Computational Fluid and Particle Dynamics in the Human Respiratory System, September 2013

[FAN2018] T. Fang, H. Huo, N. Liu, L. Wan, Y. Yuan, "A Comparative Study for Contour Detection Using Deep Convolutional Neural Networks",pp 203–208,2018

[LI2019] Z. Li, Q. Zhang, X. Yang, *Contour Detection in Cassini ISS images based on Hierarchical Extreme Learning Machine and Dense Conditional Random Field*, Research in Astron. Astrophys. Vol.0 (20xx) No.0, 000–000,2019

[DECETI]Development of a common educational and training infrastructure," Edge Detection and Zero Crossing"

[LOG] http://weisu.blogspot.com/2009/05/laplacian-of-gaussian-log.html

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr