AARHUS
UNIVERSITET

Computer Vision & Machine Learning

Alexandros Iosifidis
@
Department of Electrical and Computer Engineering
Aarhus University

This week

Edges and lines

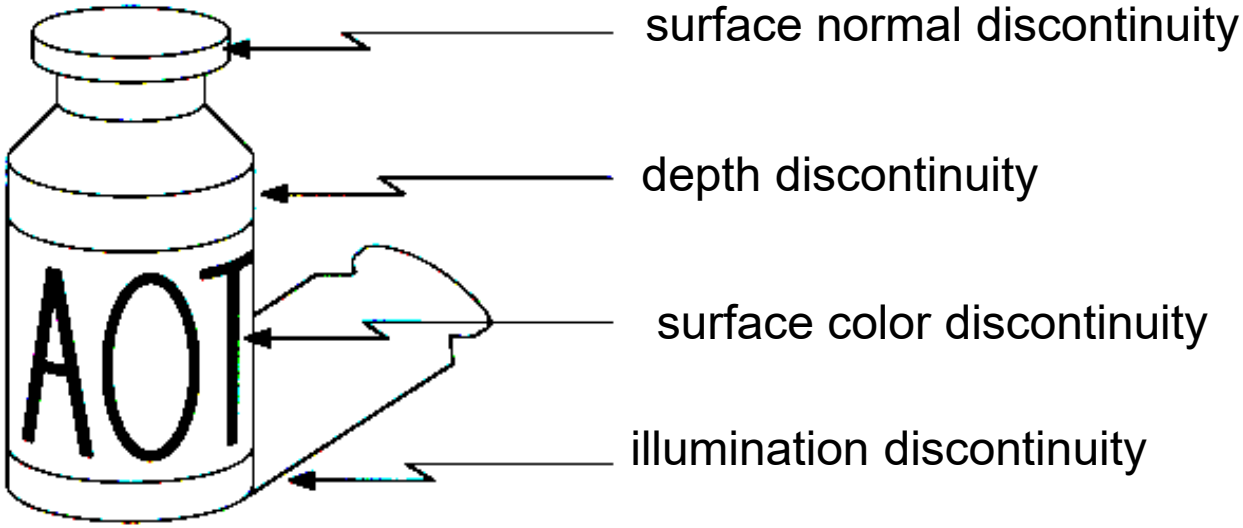
Slides based on:

- Richard Szeliski: CV: Algorithms & Applications

Edges

Edges are caused by a variety of factors:

- Boundaries between regions
- Regions are defined as image locations of different
  - color
  - intensity
  - texture.



Edges

Carry important semantic association

- Boundaries of objects
- Occlusion in 3D
- Shadow boundaries
- Silhouettes

Boundary detection

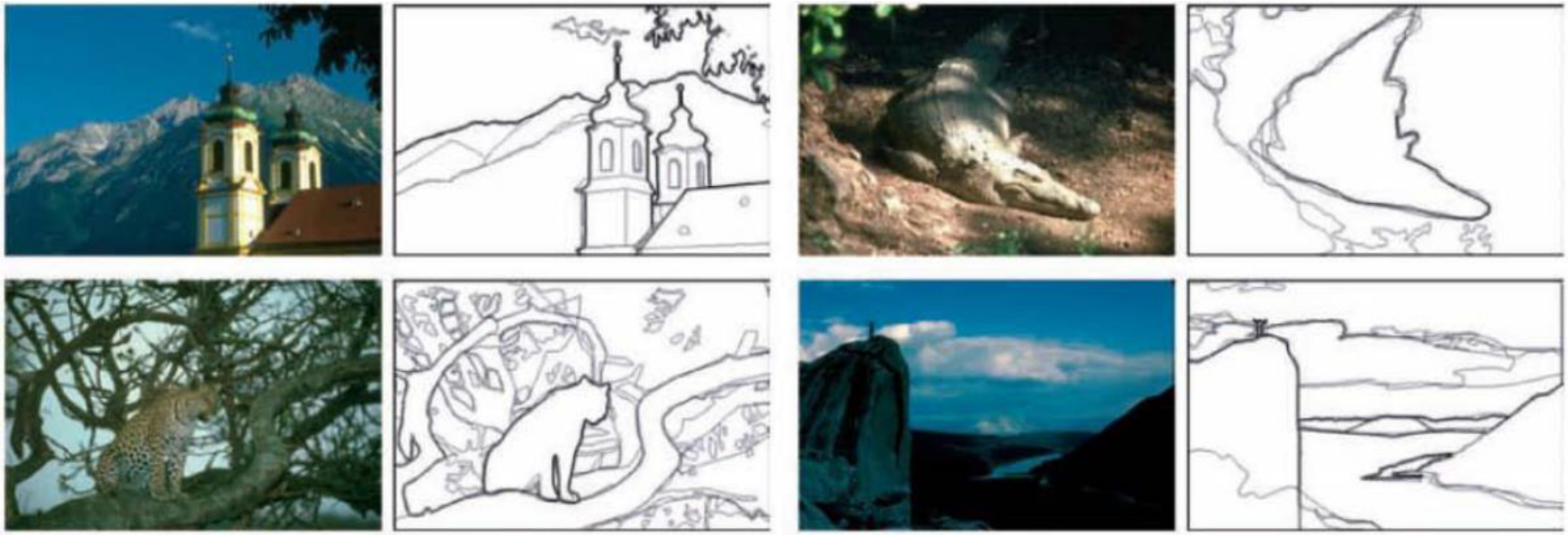
The determination of object boundaries is subjective, when performed by humans:

- We tend to exploit information related to the objects/scene
- Many times we ‘imagine’ what should be the boundary between two objects
- How about occlusions?

Boundary detection

Examples of image segmentation performed by many volunteers

- The black intensity is proportional to how many volunteers outlined a given boundary.



Edge detection

Edges correspond to rapid changes in color/intensity (intensity discontinuity)

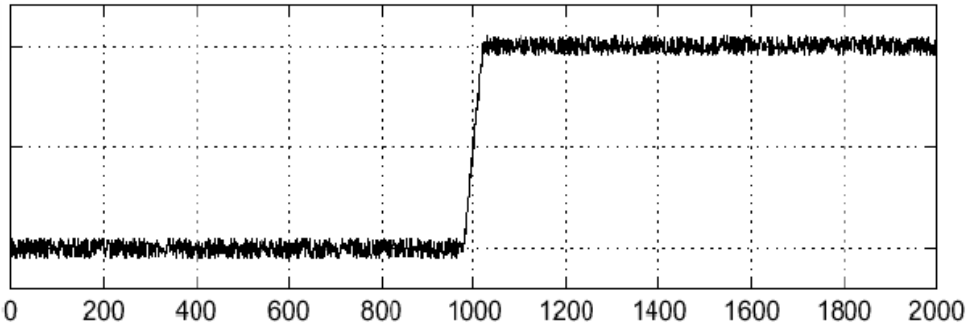
- We can use image gradient operator for edge detection

Edge detection

Edges correspond to rapid changes in color/intensity (intensity discontinuity)
- We can use image gradient operator for edge detection

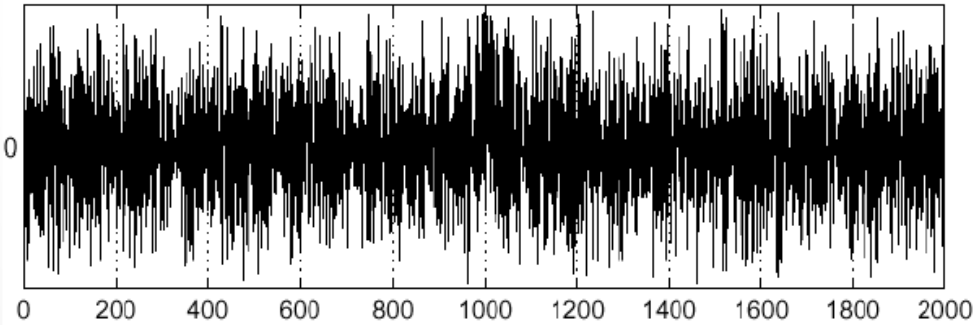
An example: densities along one row of the image.

$f(x)$



Can you find the location of the edge using the second signal?

$\frac{d}{dx}f(x)$



Edge detection

Edges correspond to rapid changes in color/intensity (intensity discontinuity)

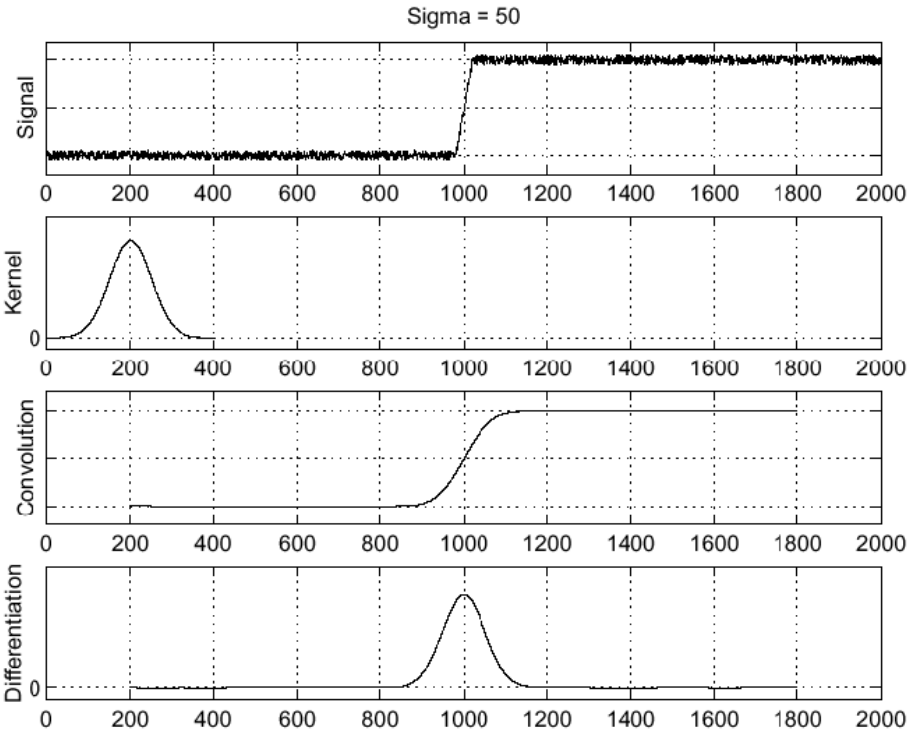
- We can use image gradient operator for edge detection
- We need to smooth the intensity values

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Edge detection

Edges correspond to rapid changes in color/intensity (intensity discontinuity)

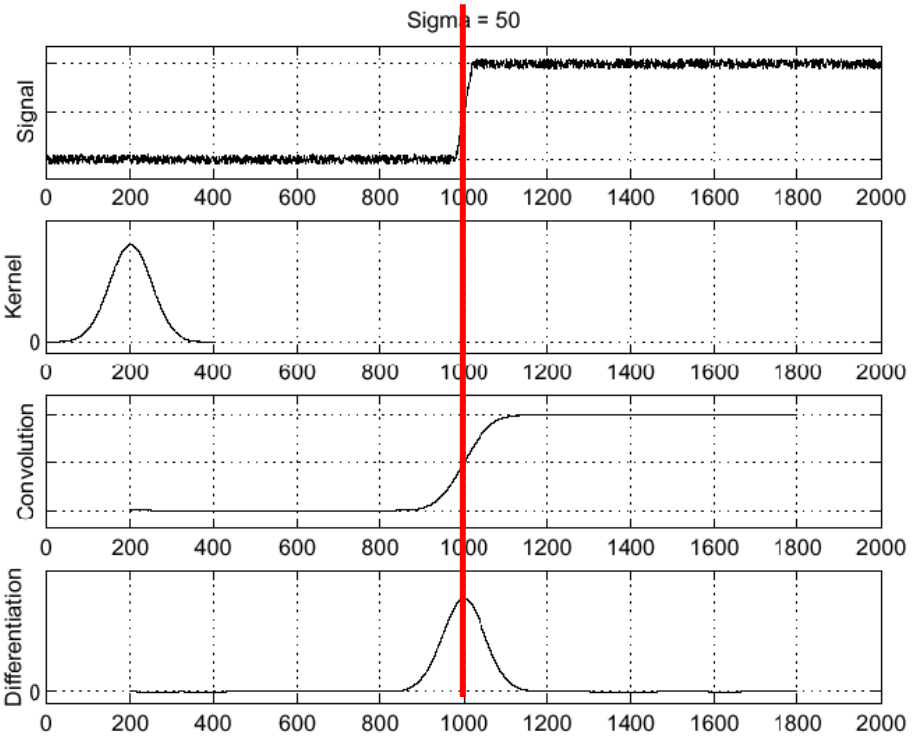
- We can use image gradient operator for edge detection
- We need to smooth the intensity values

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Edge detection

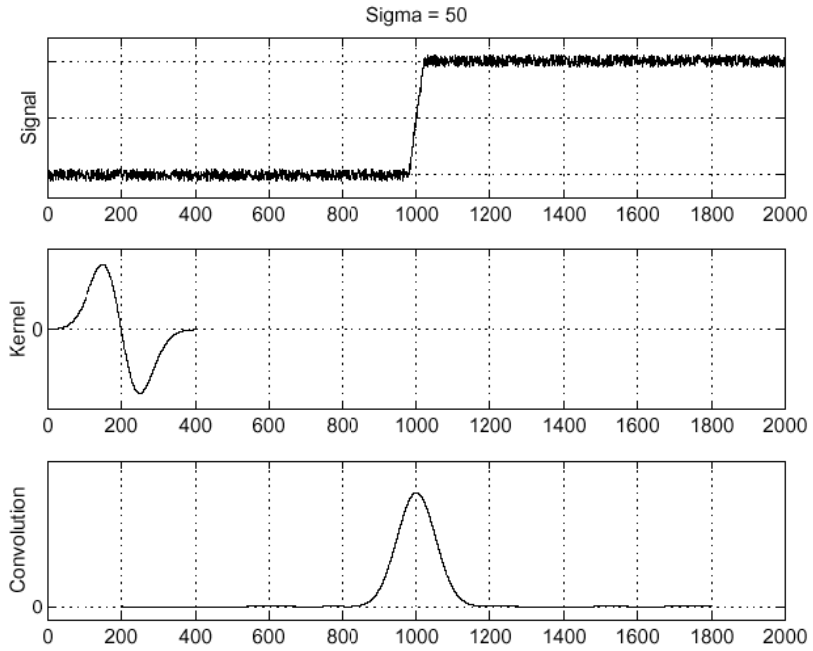
Edges correspond to rapid changes in color/intensity (intensity discontinuity)

- We can use image gradient operator for edge detection
- We need to smooth the intensity values
- To speedup the process, we can use the property $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



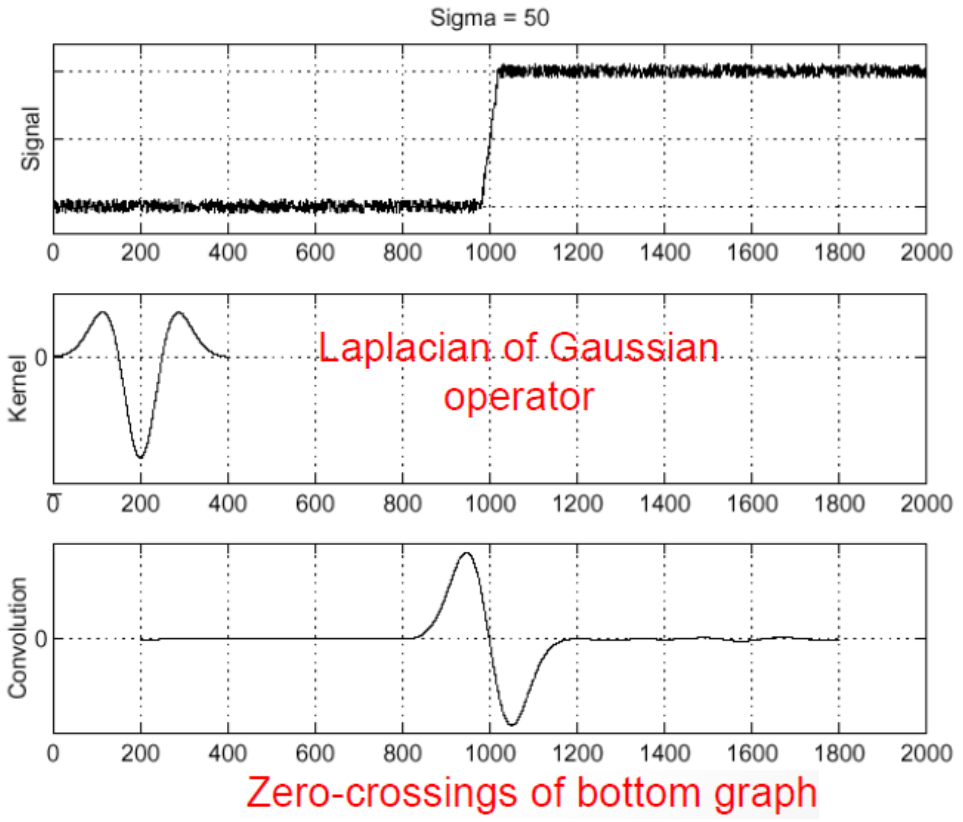
Edge detection

We can also use the Laplacian of Gaussian

f

$\frac{\partial^2}{\partial x^2} h$

$(\frac{\partial^2}{\partial x^2} h) \star f$



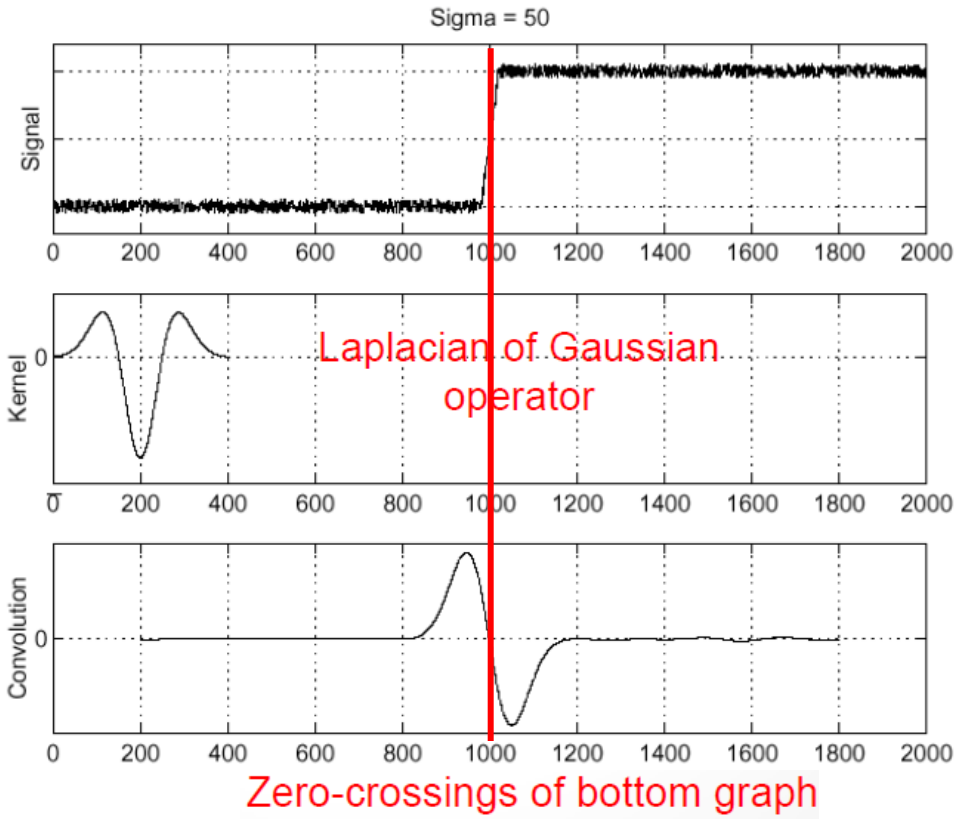
Edge detection

We can also use the Laplacian of Gaussian

f

$\frac{\partial^2}{\partial x^2} h$

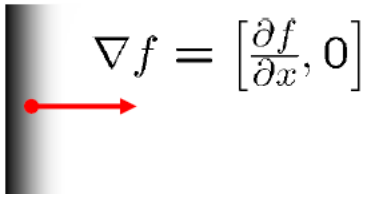
$(\frac{\partial^2}{\partial x^2} h) \star f$



Edge detection

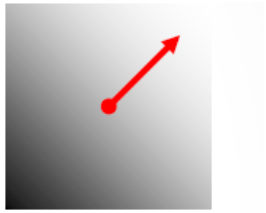
In images, we want to find edges in the 2D image plane:

- Rapid change in the intensity/color values
- It has a magnitude and a direction



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

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



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

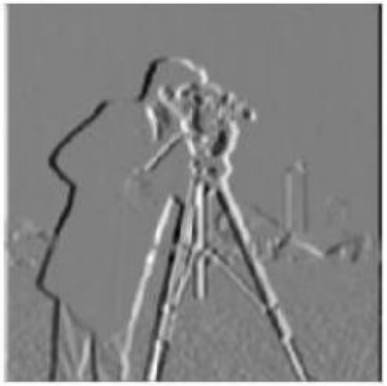
- The direction of the gradient (edge) is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

- The magnitude (or strength) of the gradient (edge) is given by:

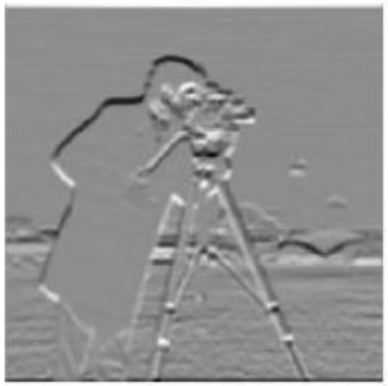
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Edge detection

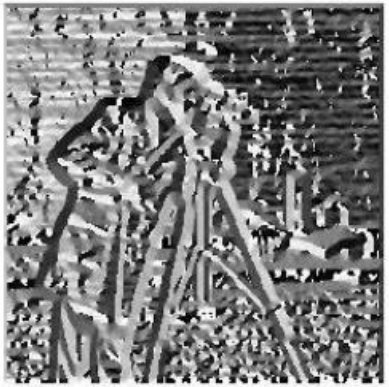
Ix



Iy



Angles

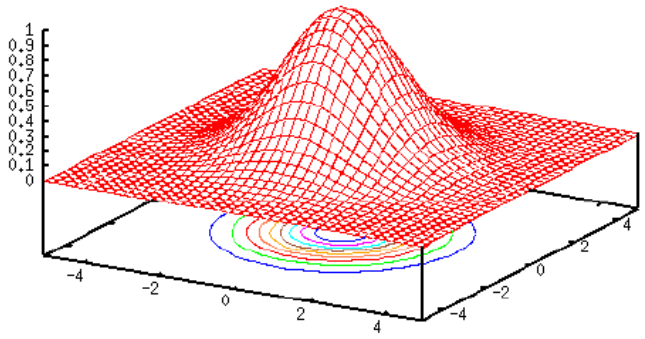


Gradient vectoor field

Edge detection

Rapid intensity variation
- Steep slopes
- Gradient
  - Magnitude: the slope or strength of intensity change
  - Orientation: perpendicular to the local contour
- Image derivatives accentuates high frequency, so amplifies noise
- Preprocessing: Smooth the image using a circularly symmetric filter

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



Derivative of Gaussian

Due to the linearity of the operators we have:

$$\mathrm{J}(\mathbf{x}) = \nabla[G_\sigma(\mathbf{x}) * I(\mathbf{x})] = [\nabla G_\sigma(\mathbf{x})] * I(\mathbf{x})$$

$$\nabla G_\sigma(\mathbf{x}) = \left(\frac{\partial G_\sigma}{\partial x}, \frac{\partial G_\sigma}{\partial y} \right)(\mathbf{x}) = [-x \quad -y] \frac{1}{\sigma^3} \exp\left(- \frac{x^2 + y^2}{2\sigma^2} \right)$$

In order to detect edges in digital images (lattices of discrete pixels), we apply discrete versions of the above operators

Several filter types, like Sobel.

Sobel operator

Sobel filters for vertical edge detection

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Horizontal kernel

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

$$\frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Vertical kernel

Filtered image

I

I_x

Sobel operator

Example

Color image

Grayscale image

Gaussian blurred image

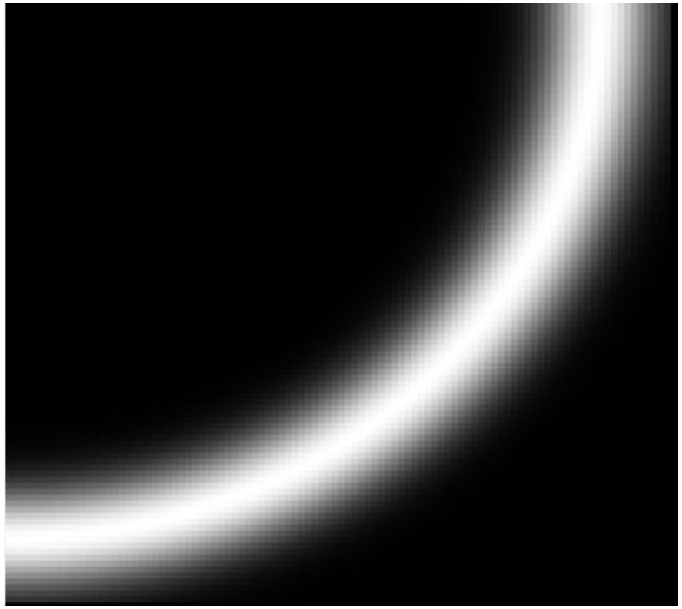
I_x

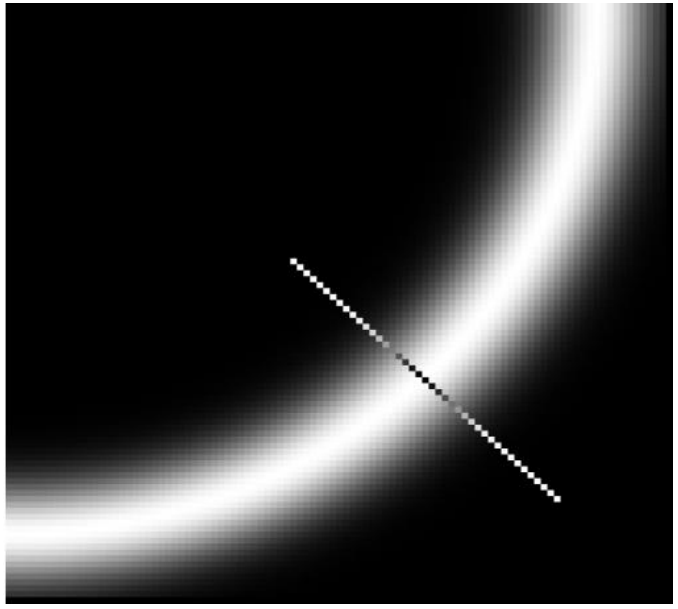
I_y

Edge detection

In order to apply edge thinning:

- Find local maxima in the gradient magnitude.
- Taking a directional derivative of the gradient magnitude strength field in the direction of the gradient and then looking for zero crossings



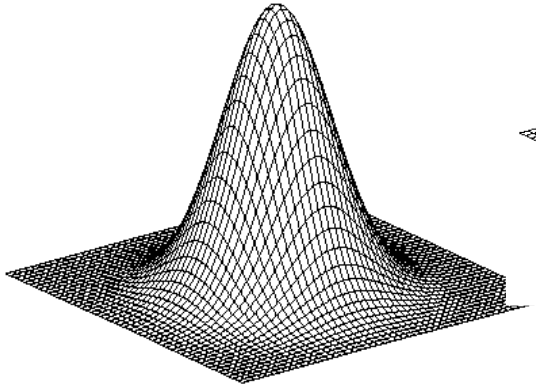


Edge detection filters

Laplacian-of-Gaussian

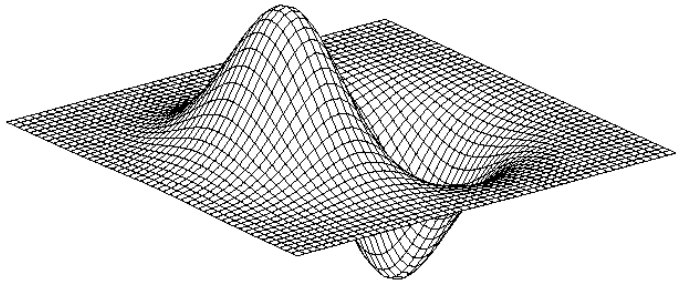
$$\nabla^2 h_\sigma(u,v)$$

Laplacian of Gaussian



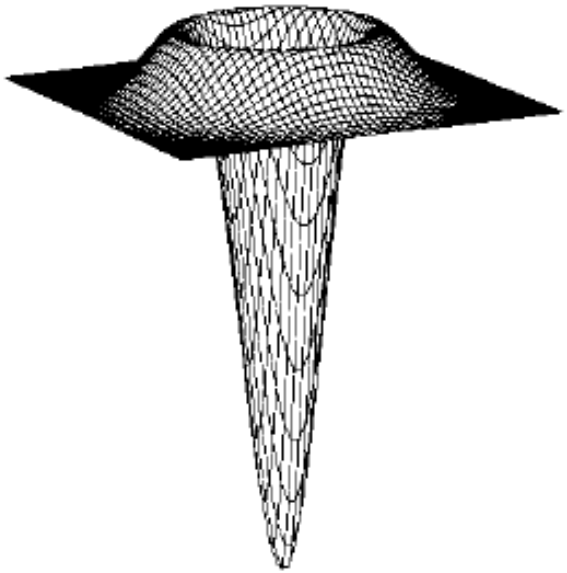
Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



Laplacian operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Edge detection filters

Heat equation: $$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

Difference of Gaussian: $$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

Relation between DoG and LoG:

$$\underbrace{G(x, y, k\sigma) - G(x, y, \sigma)}_{\text{DoG}} \approx (k-1)\sigma \frac{\partial G}{\partial \sigma} = (k-1)\sigma^2 \underbrace{\nabla^2 G}_{\text{LoG}}$$

DoG is more convenient than LoG if a Gaussian pyramid has been already computed

Edge detection filters

Zero crossing:

- We look for adjacent pixels x_i and x_j where the sign of the LoG or DoG changes
- We can define the sub-pixel location of the edge by linearly interpolating the gradient values on the original pixel grid:

$$\mathbf{x}_z = \frac{\mathbf{x}_i S(\mathbf{x}_j) - \mathbf{x}_j S(\mathbf{x}_i)}{S(\mathbf{x}_j) - S(\mathbf{x}_i)}$$

Canny edge detection

The most widely adopted algorithm:

- Low error rate
- Edge points are well localized
- Only one response to a single edge
- Three processing steps:
  - Image smoothing
  - Image derivative calculation
  - Non-maximum suppression for edge thinning

Canny edge detection

Process:

1. Image smoothing to denoise

Gaussian mask

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Canny edge detection

Process:

1. Image smoothing to denoise
2. Application of Sobel operator

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

Gx

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

Gy

Canny edge detection

Process:

1. Image smoothing to denoise
2. Application of Sobel operator
3. Gradient magnitude and orientation calculation

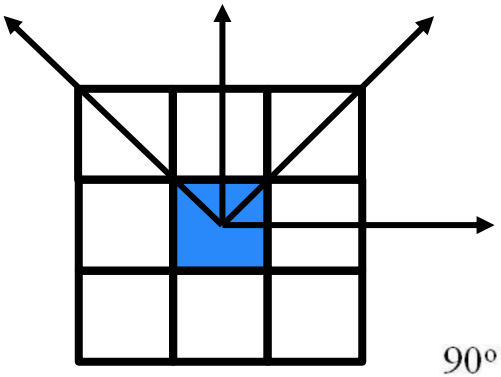
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

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

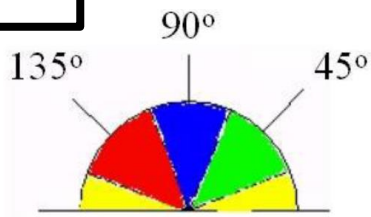
Canny edge detection

Process:

1. Image smoothing to denoise
2. Application of Sobel operator
3. Gradient magnitude and orientation calculation
4. Quantization of gradient direction
   - Because the surrounding of each pixel can be described using only four directions



Yellow: 0 to 22.5 & 157.5 to 180 degrees
Green: 22.5 to 67.5 degrees
Blue: 67.5 to 112.5 degrees
Red: 112.5 to 157.5 degrees



Canny edge detection

Process:

1. Image smoothing to denoise
2. Application of Sobel operator
3. Gradient magnitude and orientation calculation
4. Quantization of gradient direction
   - Because the surrounding of each pixel can be described using only four directions
5. Non-maximum suppression
   - Trace along the edge in the edge direction and suppress any pixel value (set it equal to 0) that is not considered to be an edge
   - This will give a thin line in the output image

Canny edge detection

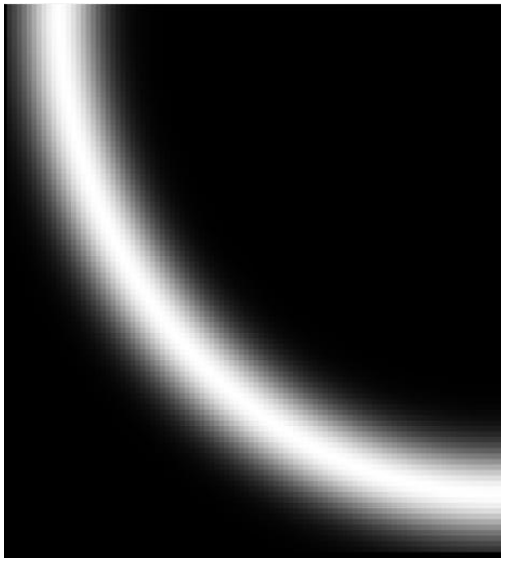
Process:

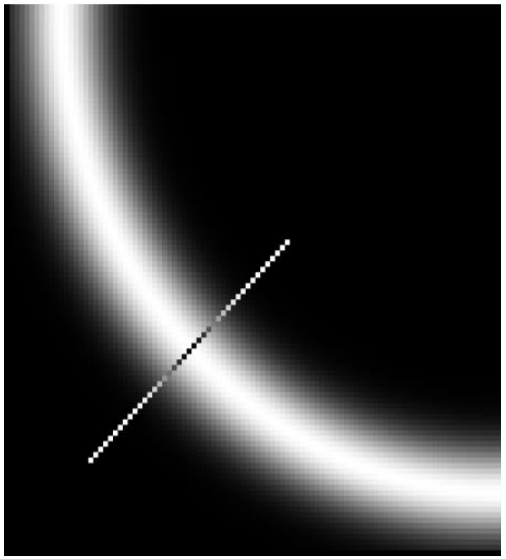
1. Image smoothing to denoise
2. Application of Sobel operator
3. Gradient magnitude and orientation calculation
4. Quantization of gradient direction
   - Because the surrounding of each pixel can be described using only four directions
5. Non-maximum suppression
   - Trace along the edge in the edge direction and suppress any pixel value (set it equal to 0) that is not considered to be an edge
   - This will give a thin line in the output image
6. Hysteresis
   - Eliminate streaking. Use two thresholds T1 and T2
   - Pixel edge g>T1 → strong edge
   - Pixel connected to edge pixel and g>T2 → weak edge

Canny edge detection

Non-maximum suppression:

- Find the points along the curve with the biggest edge magnitude
- We look for a maximum along a slice normal to the curve
- These points should form a curve.





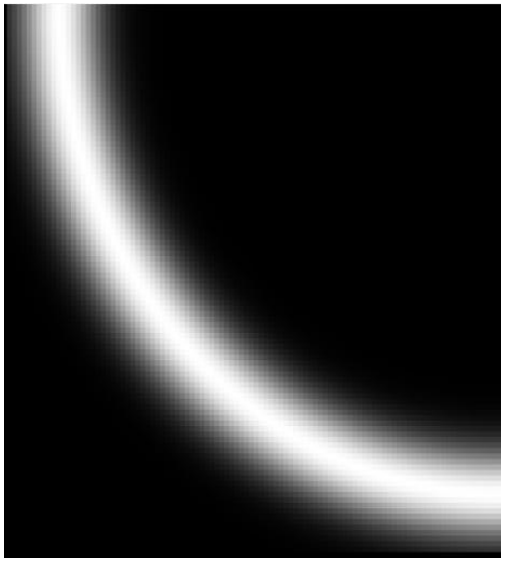
Canny edge detection

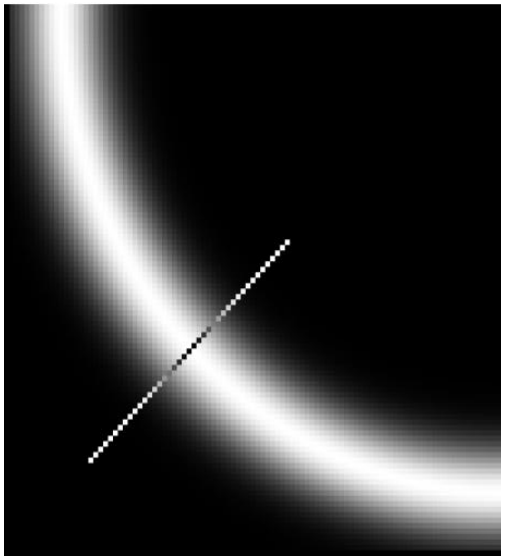
Non-maximum suppression:

- Find the points along the curve with the biggest edge magnitude
- We look for a maximum along a slice normal to the curve
- These points should form a curve.

There are two algorithmic issues:

1. at which point is the maximum
2. where is the next one?



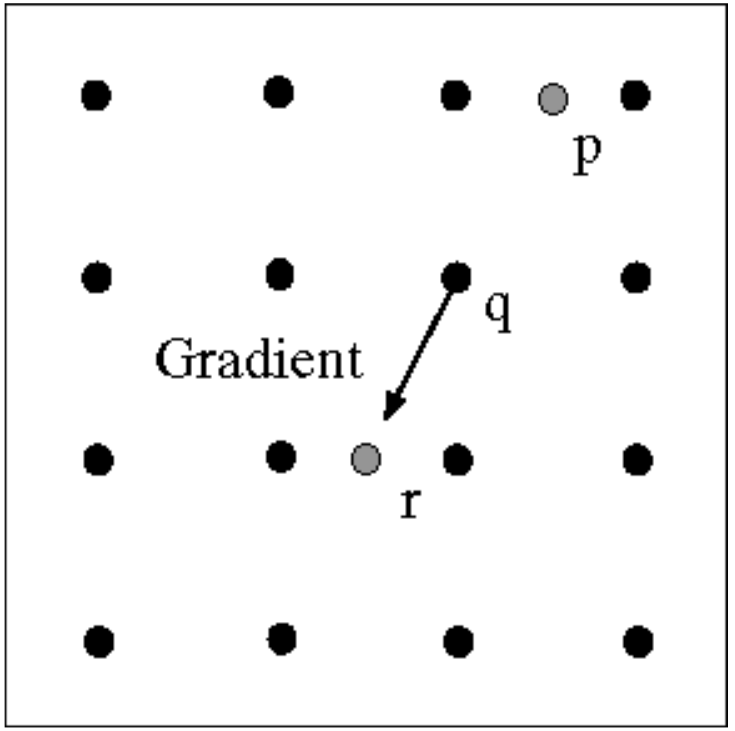


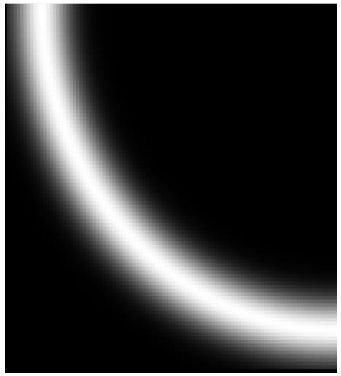
Canny edge detection

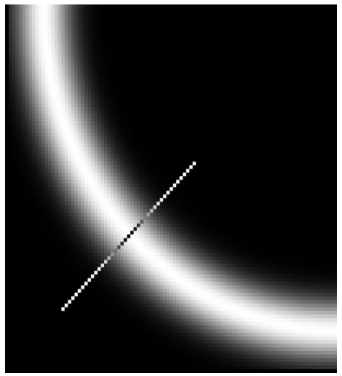
Non-maximum suppression:

1. at which point is the maximum

- Given the edge direction at q, we need to compare its magnitude with the edge magnitudes at the locations r and p.
- In order to find the edge magnitudes in r and p we need to apply interpolation
- If the edge magnitude at q is higher than at p and r, then q corresponds to maximum





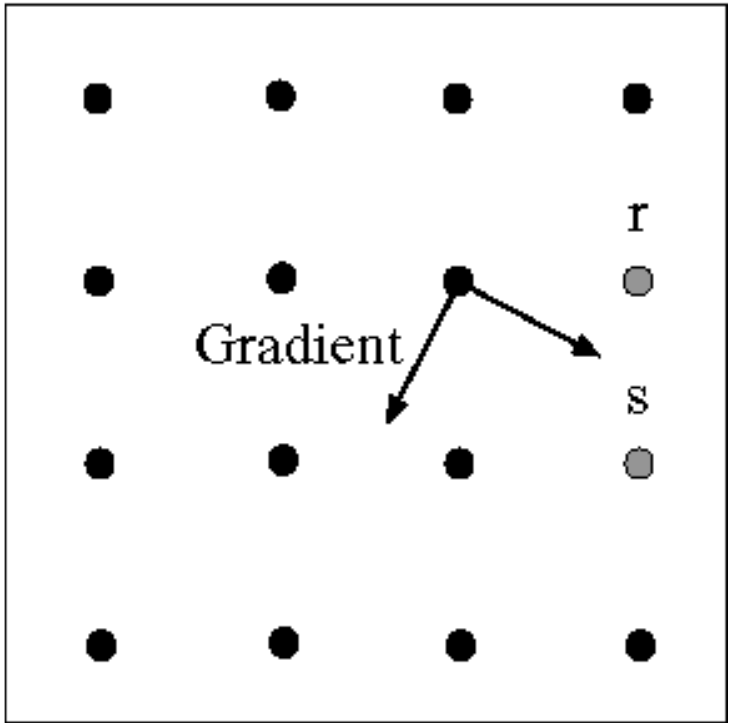


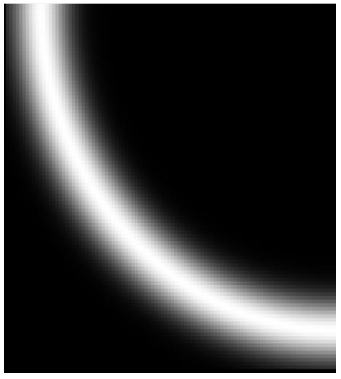
Canny edge detection

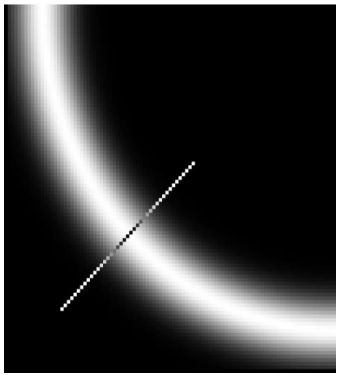
Non-maximum suppression:

1. at which point is the maximum
2. where is the next one?

- If the marked point is an edge, we construct the tangent to the edge (normal/orthogonal to the gradient at that point)
- We use this direction to predict the next points (here either r or s)







Canny edge detection

Example

courtesy of G. Loy

Image

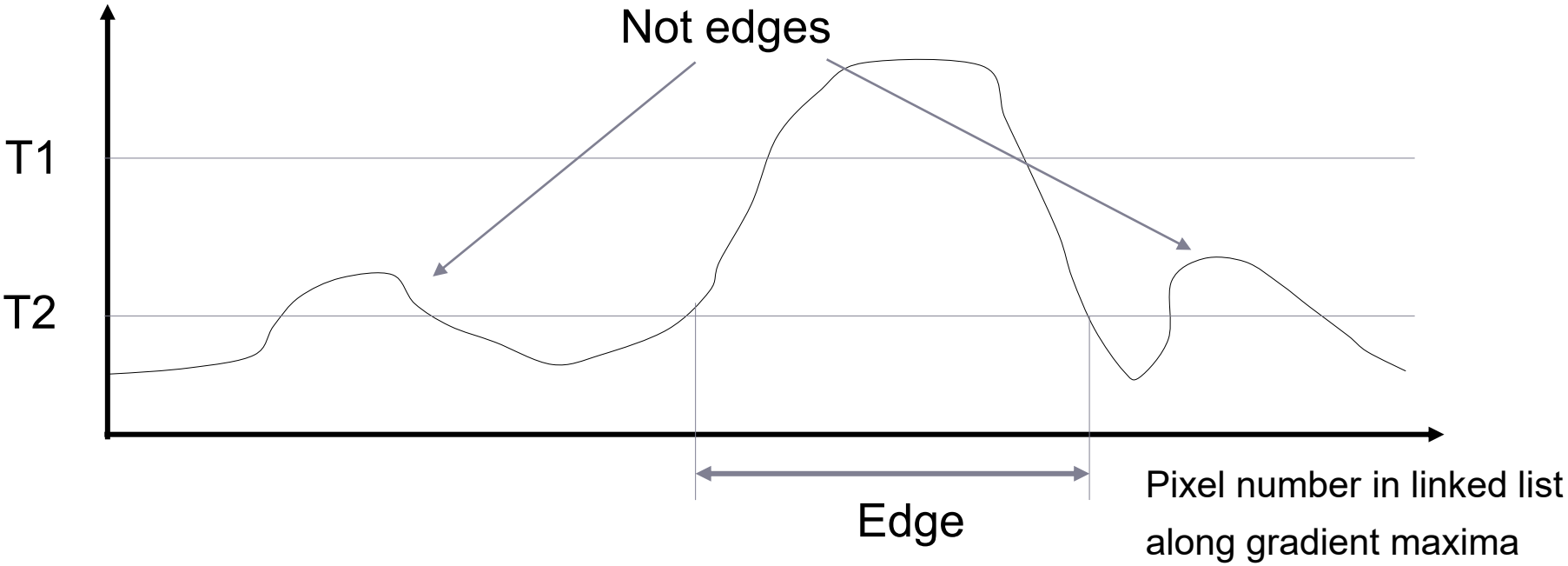
Gradient magnitude image

Non-maxima
suppressed image

Canny edge detection

Closing edge gaps:

- Use the hysteresis idea
  - Create an edge using a high threshold value T1
  - Identify a weak edge if it is connected to an existing edge



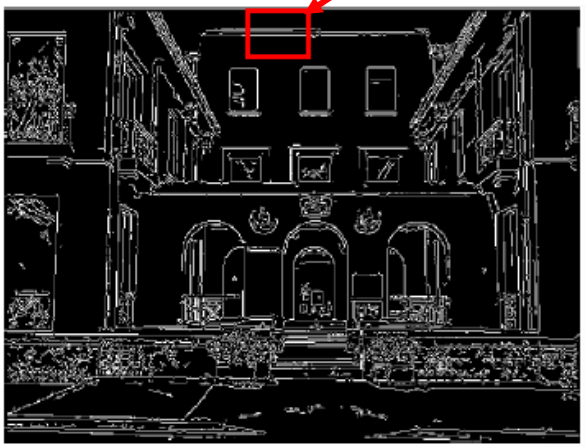
Canny edge detection

Example

Original image

Strong edges only

Gap is gone



Strong and connected week edges

Week edges

courtesy of G. Loy

Combining edge feature cues

Detect edges to match boundary detection obtained by humans

Combine multiple low-level cues such as brightness, color and texture

Martin, Fowlkes, and Malik (2004)

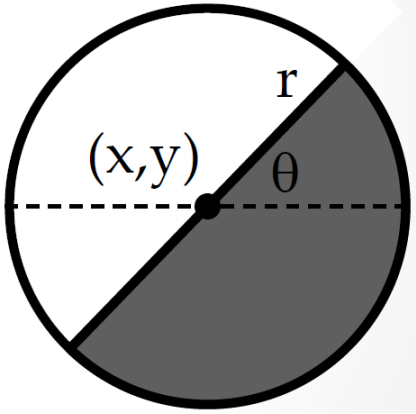
State-of-the-art performance:

- Construct and train separated half-disc detectors for difference in brightness, color and texture
- Sharpen response using a soft non-maximal suppression
- Combine the three detectors using machine learning techniques

Half-disc edge detector

At a location (x,y) in the image:

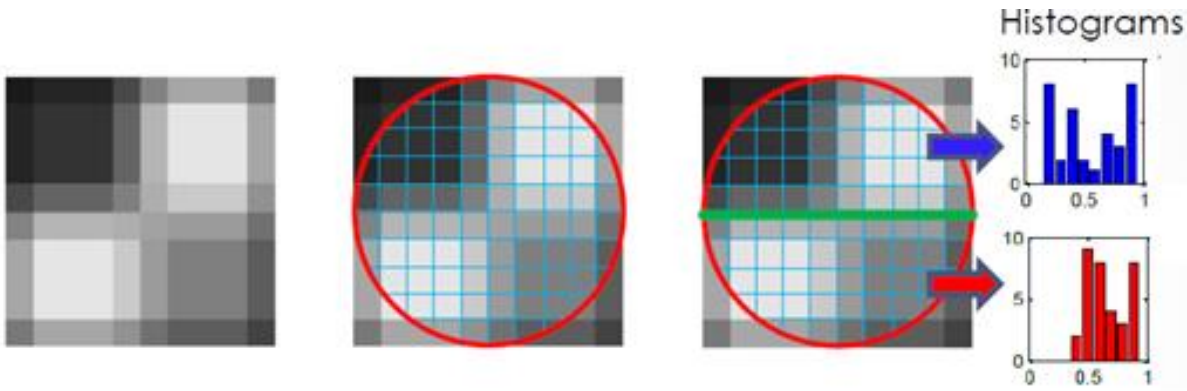
- draw a circle of radius r, and divide it along the diameter at orientation θ
- the gradient function G(x,y,r,θ) compares the contents of the two resulting disc halves.
- a large difference between the disc halves indicates a discontinuity in the image along the disc's diameter.



Half-disc edge detector

At a location (x,y) in the image:

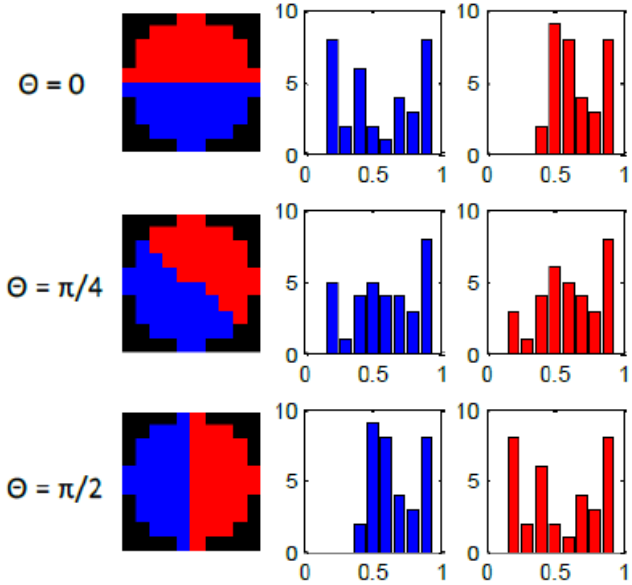
- draw a circle of radius r, and divide it along the diameter at orientation θ
- the gradient function G(x,y,r,θ) compares the contents of the two resulting disc halves.
- a large difference between the disc halves indicates a discontinuity in the image along the disc's diameter.



Half-disc edge detector

At a location (x,y) in the image:

- draw a circle of radius r, and divide it along the diameter at orientation θ
- the gradient function G(x,y,r,θ) compares the contents of the two resulting disc halves.
- a large difference between the disc halves indicates a discontinuity in the image along the disc's diameter.



Half-disc edge detector

Difference between two histograms:

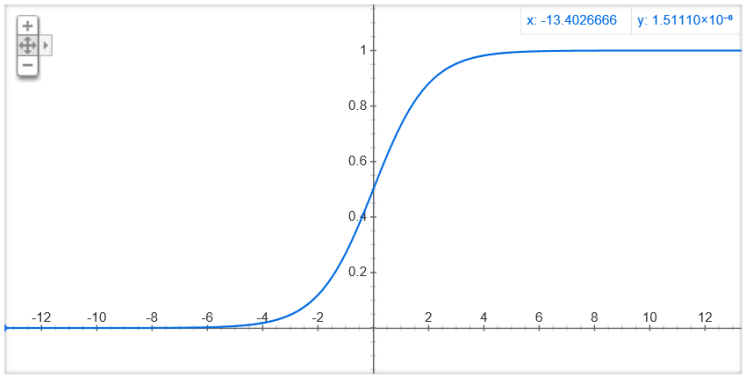
- χ^2 distance

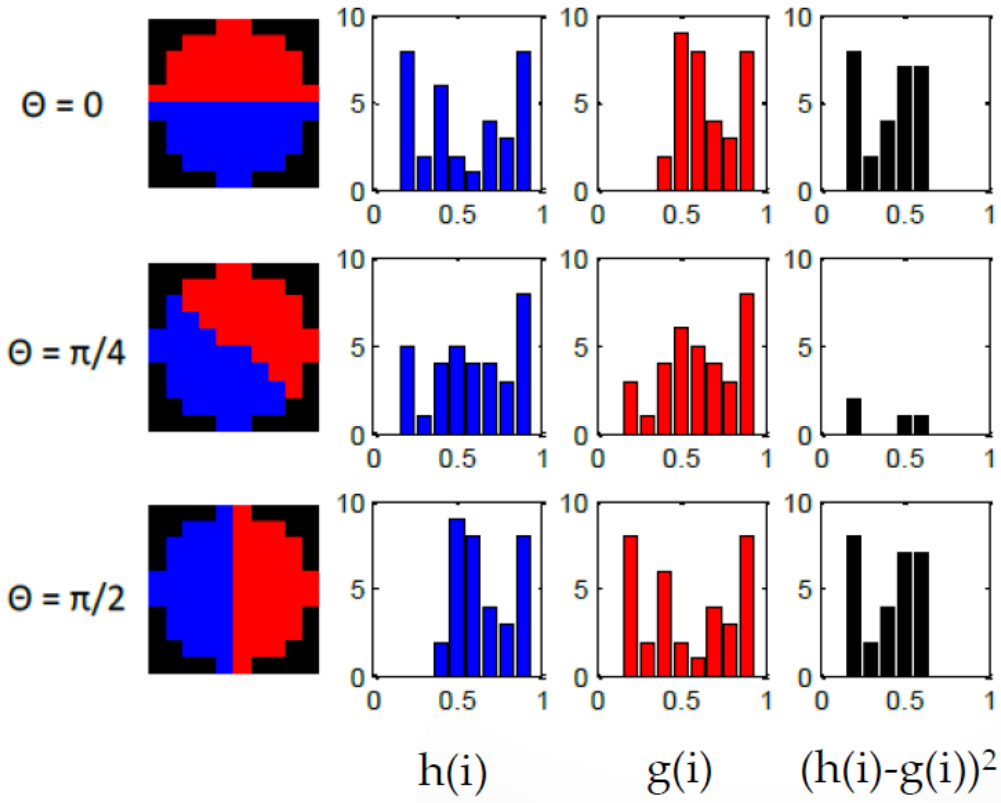
$$\chi^2(g,h) = \frac{1}{2}\sum \frac{(g_i - h_i)^2}{g_i + h_i}$$

- Decide if a pixel corresponds to an edge using the probability-like function (with parameter β_{BG})

$$p_{BG} = \frac{1}{1 + \exp\left(-\chi^2 \cdot \beta_{BG}\right)}$$

1/(1+exp(-x))

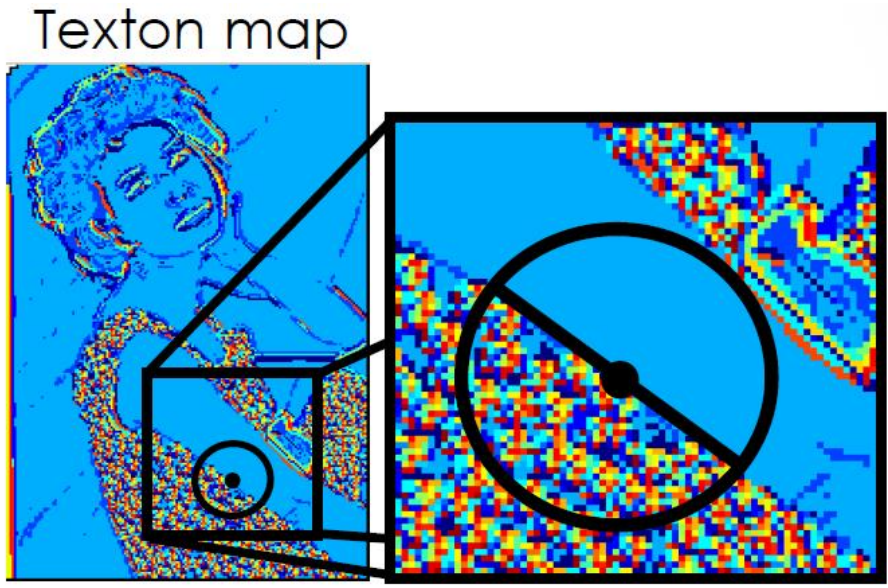




Texture gradients

Below we see a texture (texton) map

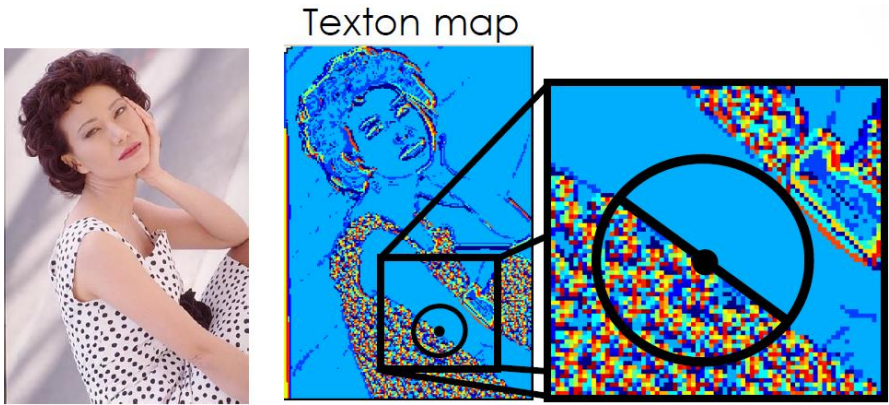
- For now, don't worry about how the texton map intensities are calculated.
- Question: Will the histograms on either side of the half-disc be similar or different?



Texton map

Process:

1. Convolve the input image with a set of 13 oriented filters
2. Each pixel is now described by a 13-dimensional feature vector (one entry for each filter response)
3. Cluster all feature vectors using k-means clustering (say 64 clusters). The mean vector of the i'th cluster is referred to is the i'th texton.
4. Each pixel is assigned the nearest texton, and each texton is assigned a different color.



Texton map

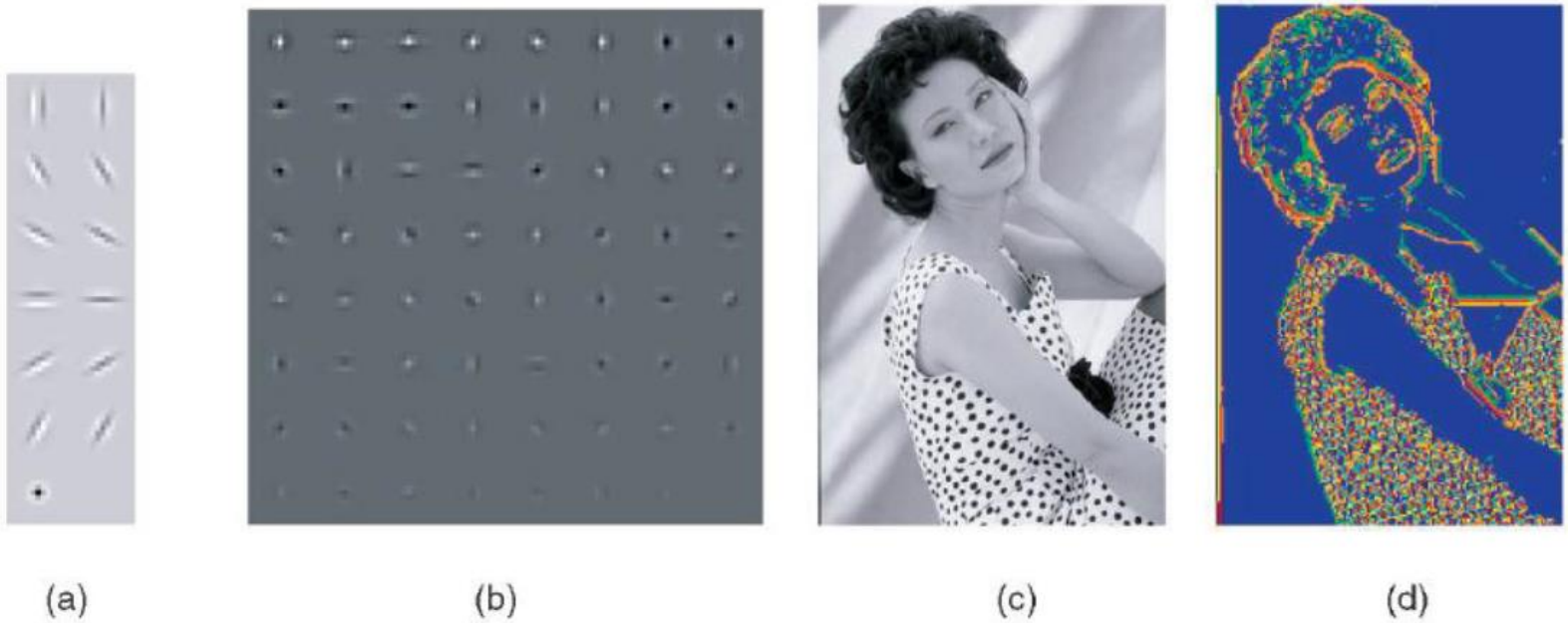
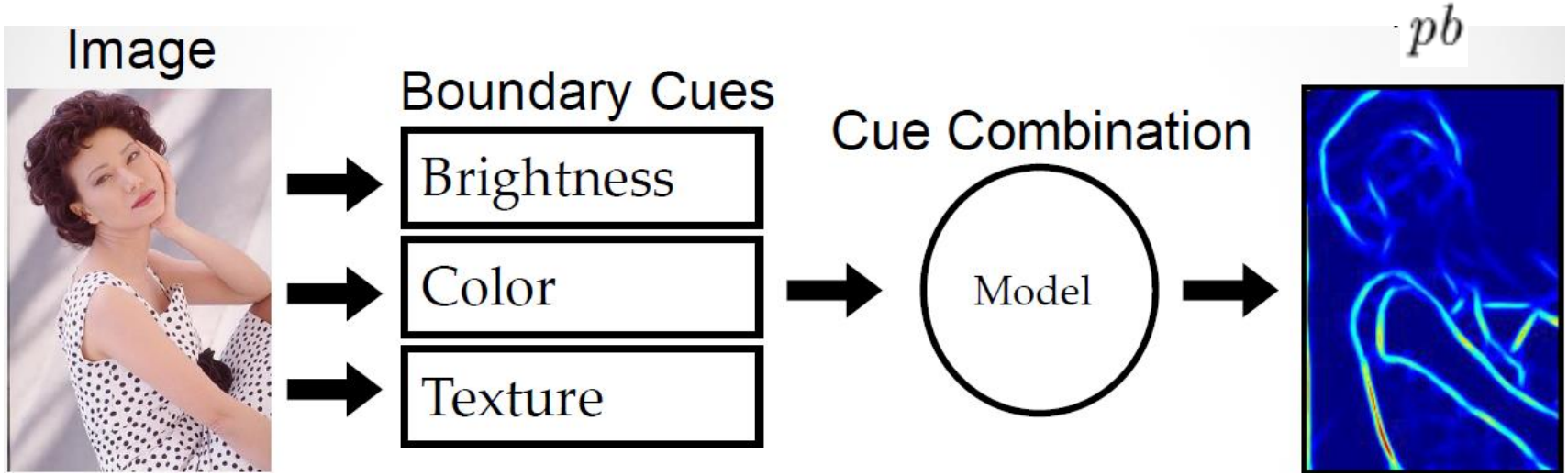


Fig. 4. Computing Textons. (a) Filter Bank: The 13-element filter bank used for computing textons. (b) Universal Textons: Example universal textons computed from the 200 training images, sorted by L1 norm for display purposes. (c) Image and (d) Texton Map: An image and its associated texton map. Texton quality is best with a single scale filter bank containing small filters. Each pixel produces a 13-element response to the filter bank, and these responses are clustered with k-means. In this example, using 200 images with k = 64 yields 64 universal textons. The textons identify basic structures such as steps, bars, and corners at various levels of contrast. If each pixel in the image shown in (c) is assigned to the nearest texton and each texton is assigned a color, we obtain the texton map shown in (d). The elongated filters have 3:1 aspect, and the longer σ was set to 0.7 percent of the image diagonal (about 2 pixels).

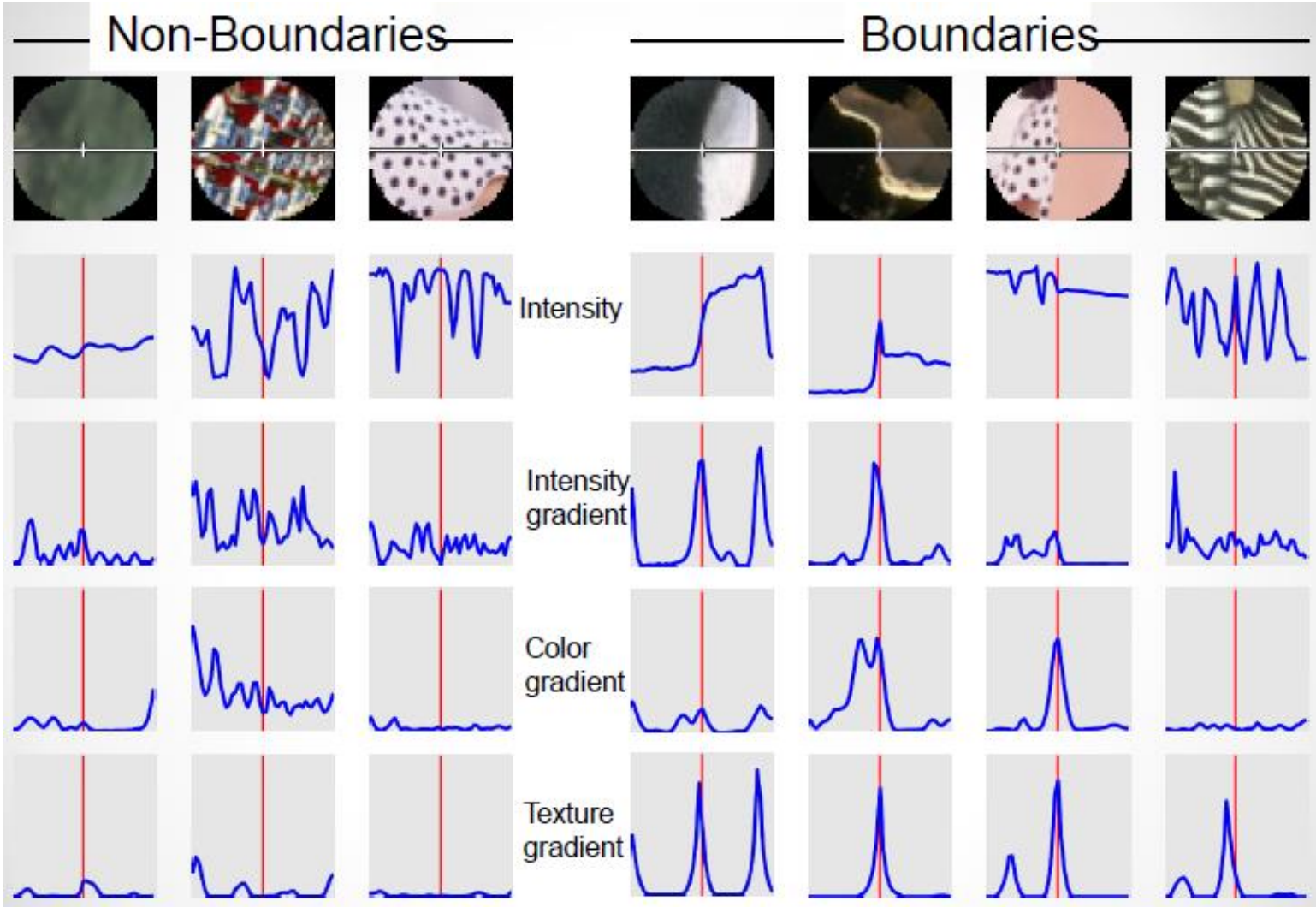
Texton map



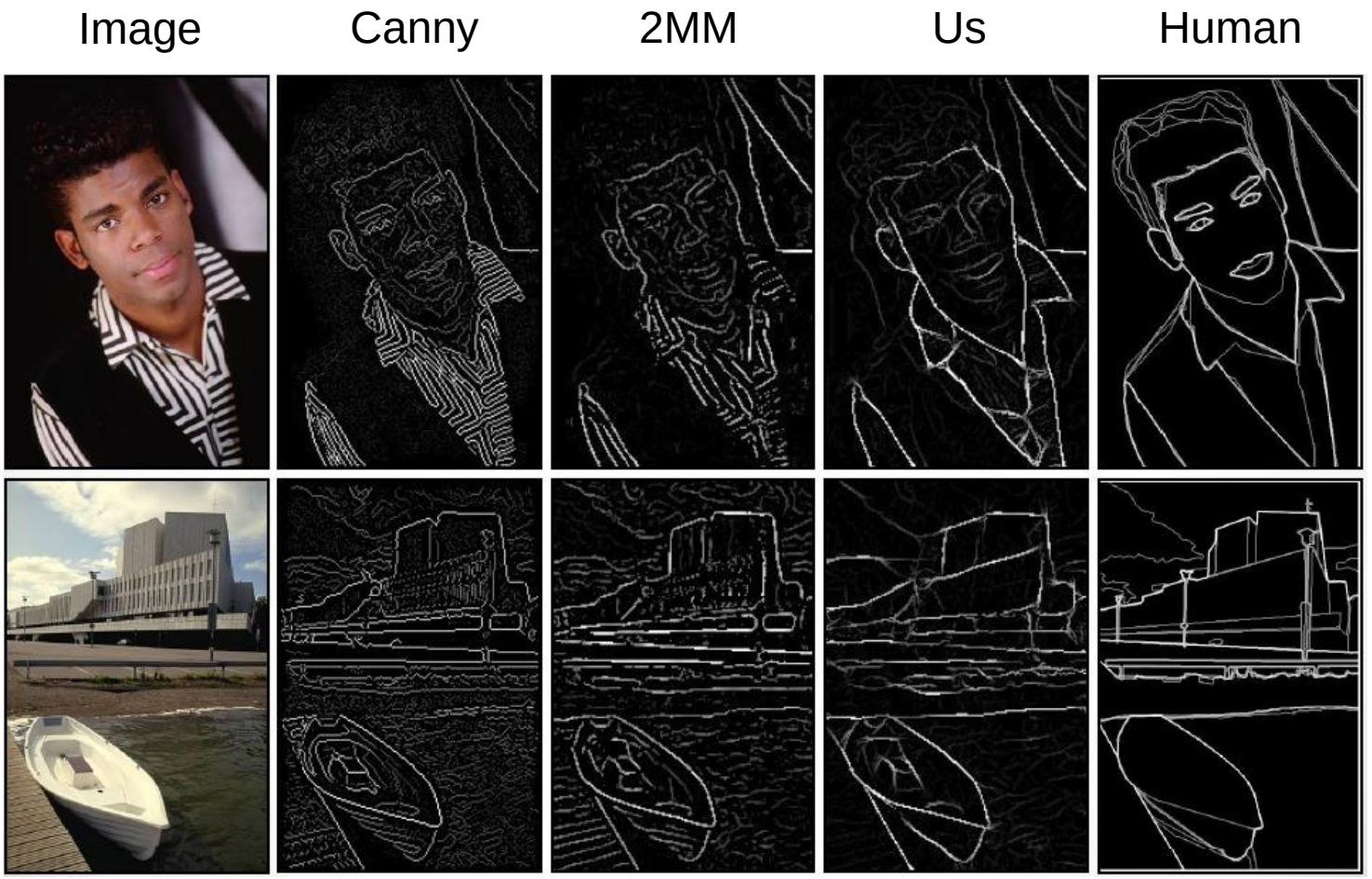
Training: Learn the posterior probability of a boundary $P_b(x,y,\theta)$ based images segmented by humans.

$$pb(x,y) = \frac{1}{1+\exp\left(-\beta_{offset} - BG(x,y)\cdot\beta_{BG} - TG(x,y)\cdot\beta_{TG}\right)}$$

Examples


Non-Boundaries
Boundaries
Intensity
Intensity gradient
Color gradient
Texture gradient

P_b images



Combining edge feature cues

- Hard to implement
- Quite slow
- Matlab source code:
  - Source code:

www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/code/segbench.tar.gz

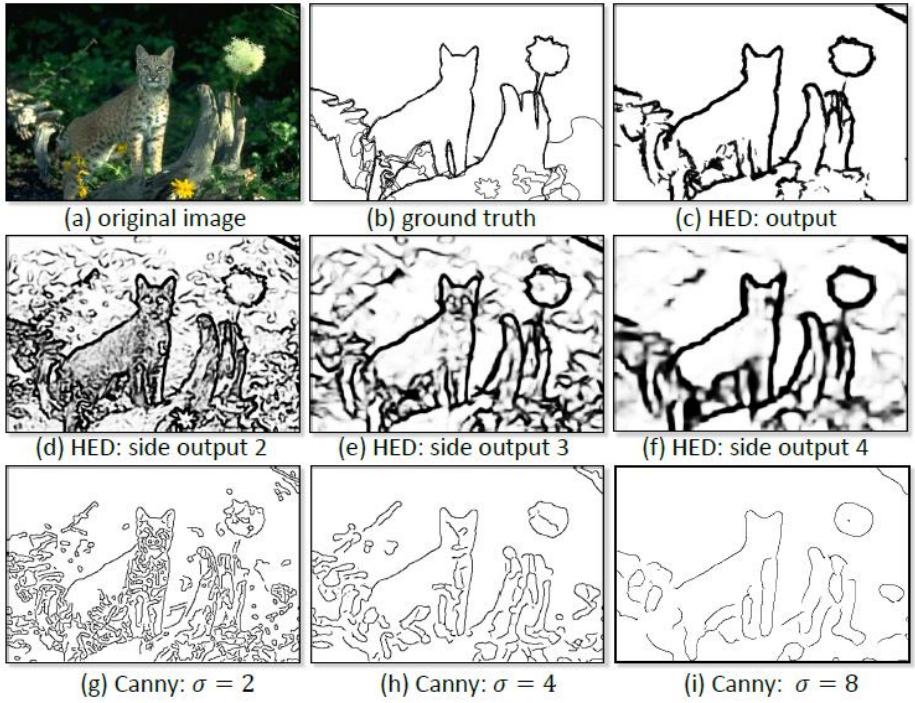
  - The Berkeley Segmentation Dataset and Benchmark:
    http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
  - Homepage of the Berkeley computer vision group:
    http://www.eecs.berkeley.edu/Research/Projects/CS/vision/

Supervised edge detectors

Given a set of annotated images:

- Original images
- Edge maps generated by human annotators

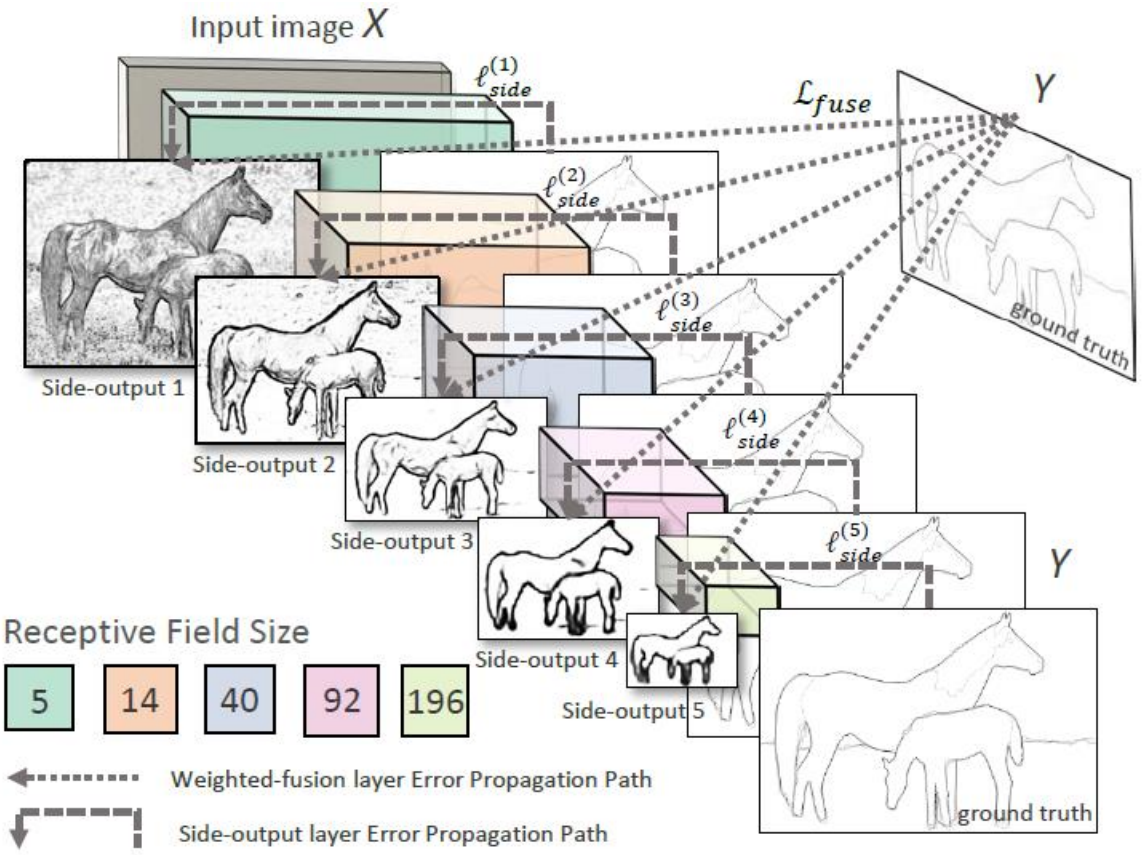
one can train an end-to-end model for edge detection (like HED)



S. Xie and Z. Tu, Hollistically-Nested Edge Detection, arXiv:1504.06375v2

Supervised edge detectors

HED



S. Xie and Z. Tu, Hollistically-Nested Edge Detection, arXiv:1504.06375v2

Edge linking

Link edgels into continuous contours
- Pick up an unlinked edgel
- Follow its neighbors in both directions

Canny
- Using two thresholds to allow a curve being tracked above the higher threshold to dip in strength down to the lower threshold

Curve representation

Contour grouping, boundary completion and junction/crossing detection

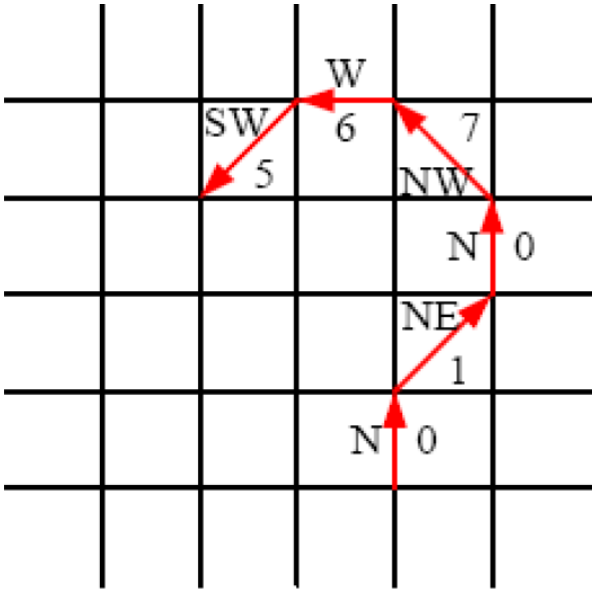
Edge linking

Representations of linked edge lists

- Chain code

  A chain code encodes a list of connected points lying on an N8 grid using a three-bit code corresponding to the eight cardinal directions
  (N, NE, E, SE, S, SW, W, NW)
  between a point and its successor

Compact but not very suitable for further processing



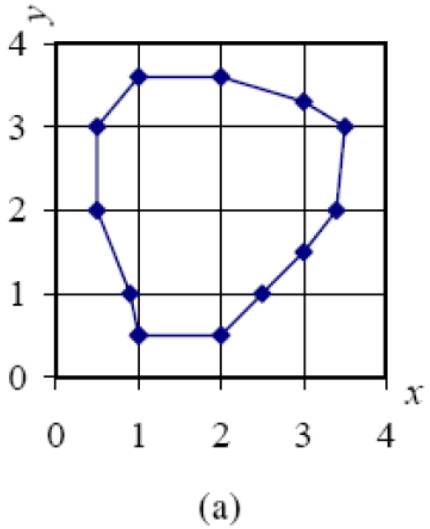
Edge linking

Representations of linked edge lists

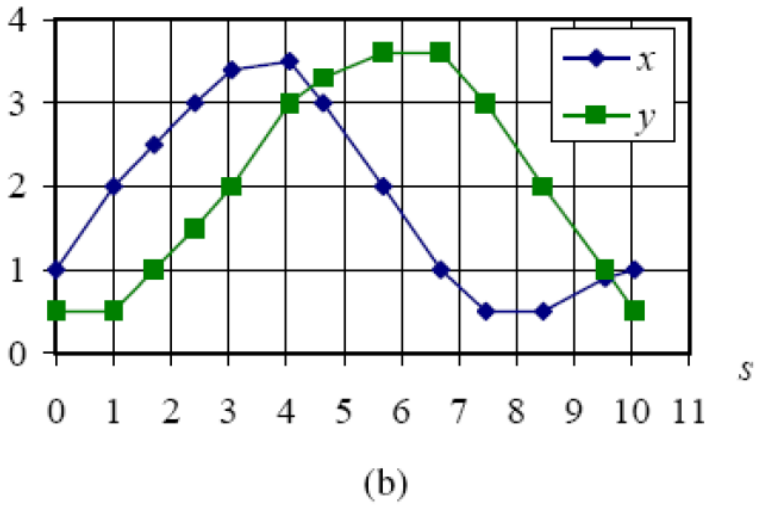
- Arc length parameterization $\mathbf{x}(s)$

- s: the arc length along the curve

- Make the matching and processing operations much easier



(a)



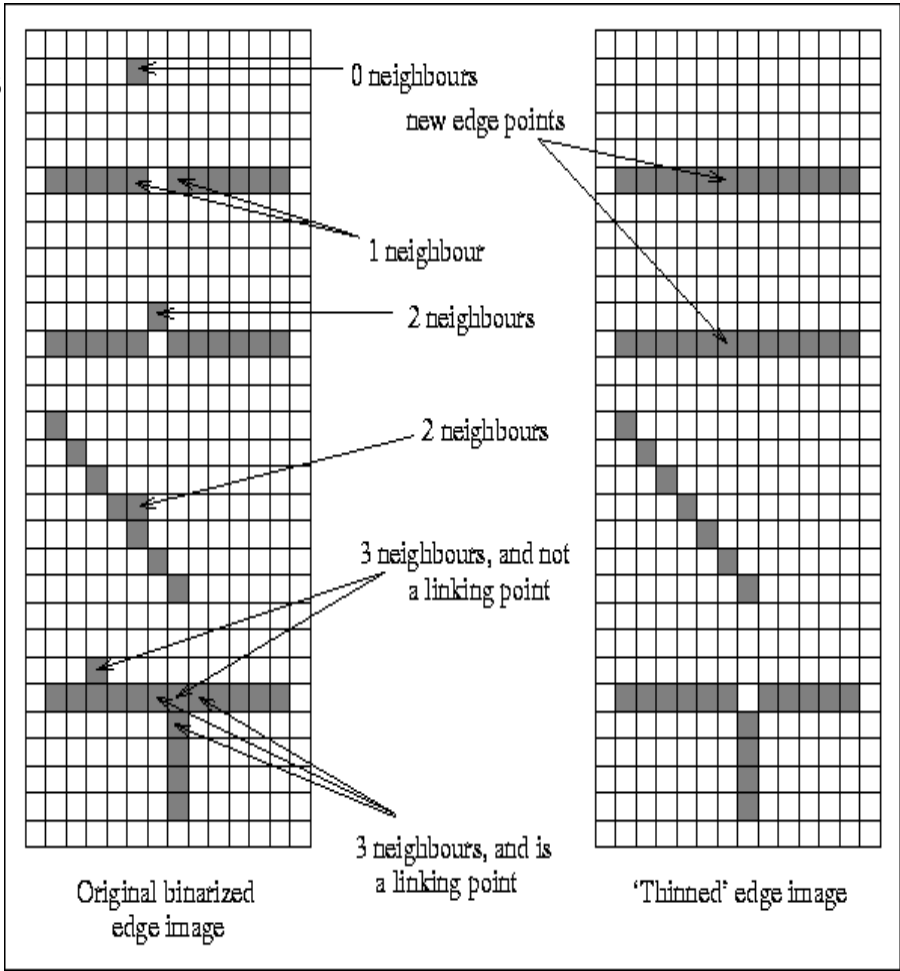
(b)

Edge thinning

Binary thinning:

- remove spurious or unwanted edge points
- add in edge points where they should be reported but have not been.
- the new edge points will only be created if the edge response allows this

local improving rules



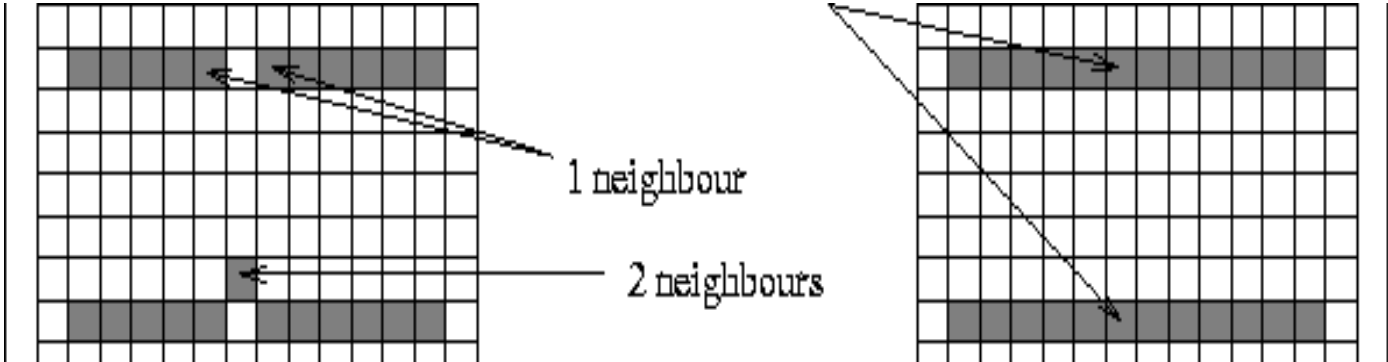
Edge thinning

More info

https://users.fmrib.ox.ac.uk/~steve/susan/

The SUSAN Thinning algorithm: If an edge point has:

- 0 neighbors → Remove the edge point.
- 1 neighbor → Search for the neighbor with the maximum (non-zero) edge response, to continue the edge, and to fill in gaps in edges.
  - The responses used are those found by the initial stage of the edge detector, before non-maximum suppression.
  - They are slightly weighted according to the existing edge orientation so that the edge will prefer to continue in a straight line.
  - An edge can be extended by a maximum of three pixels.



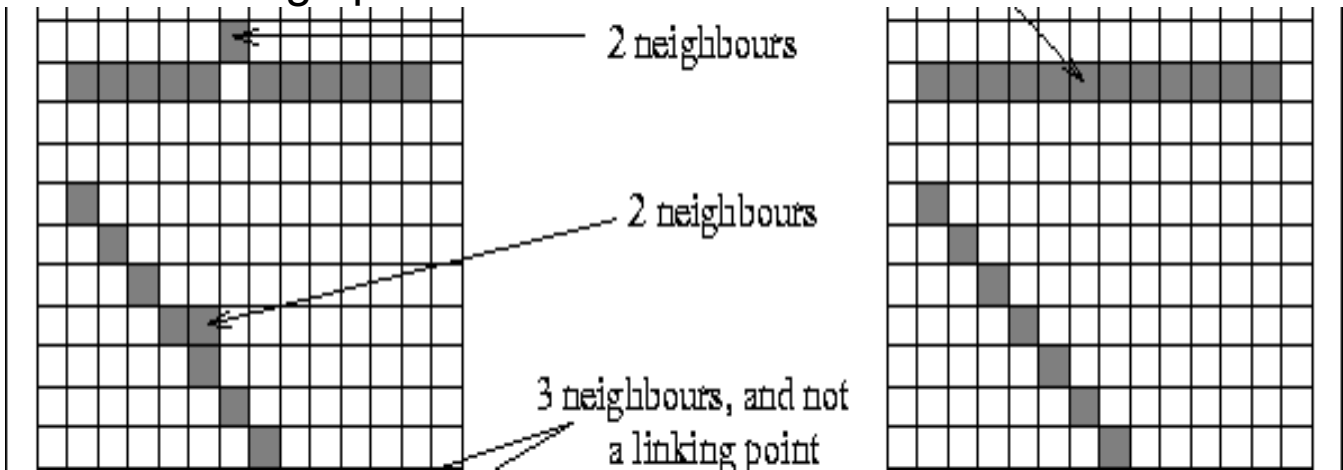
Edge thinning

More info

https://users.fmrib.ox.ac.uk/~steve/susan/

The SUSAN Thinning algorithm: If an edge point has:

- 2 neighbors → Three possible cases:
  - If the point is "sticking out" of an otherwise straight line, then compare its edge response to that of the corresponding point within the line
    - If the potential point within the straight edge has an edge *response greater than 0.7* of the current point's response, move the current point into line with the edge
  - If the point is adjoining a diagonal edge then remove it
  - Otherwise, the point is a valid edge point



Edge thinning

More info

https://users.fmrib.ox.ac.uk/~steve/susan/

The SUSAN Thinning algorithm: If an edge point has:

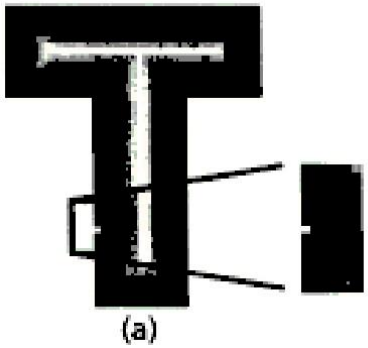
- more than 2 neighbors
  - If the point is not a link between multiple edges then thin the edge.
    - This will involve a choice between the current point and one of its neighbors.
    - If this choice is made in a logical consistent way then a “clean” looking thinned edge will result.

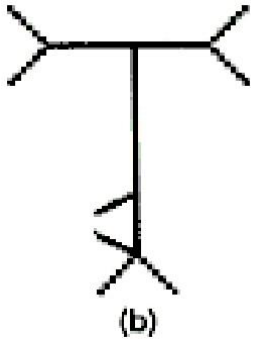
The above rules are applied to all pixels of the image

- starting from top-left pixel
- moving left to right and top to down

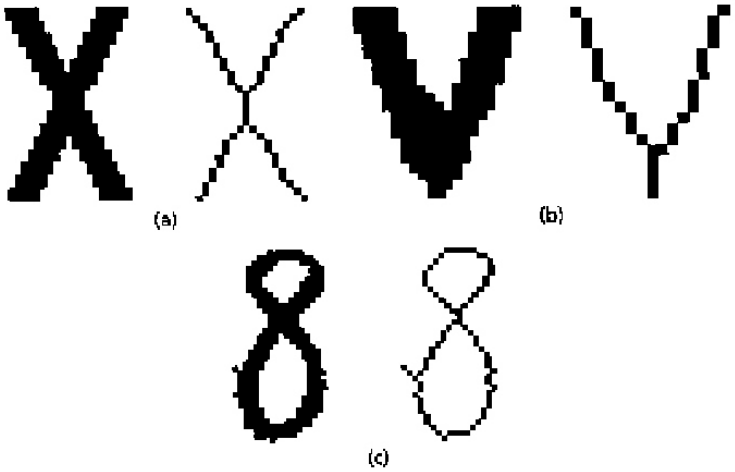
Edge thinning

Examples

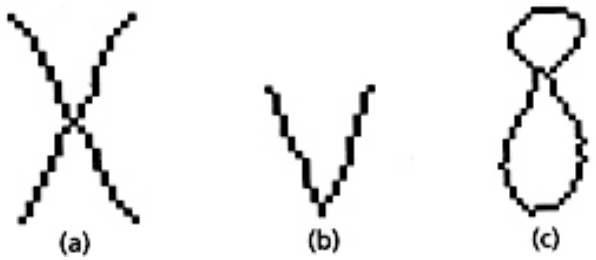




Bad thinning examples



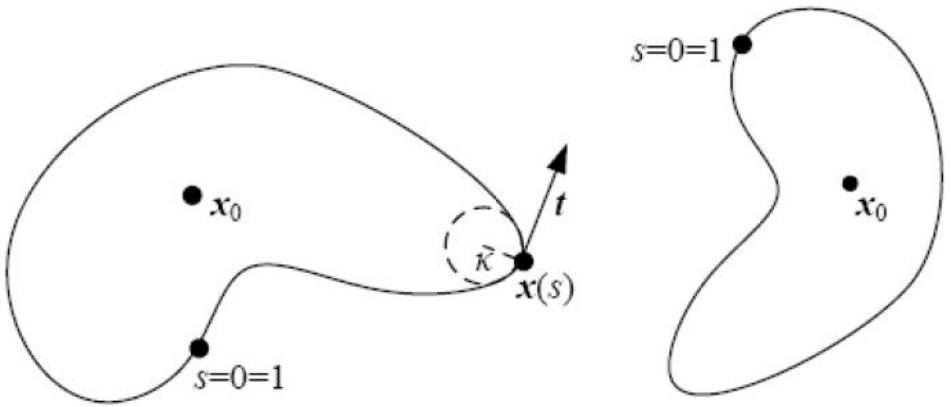
Their correct thinning



Curve matching

Arc-length parameterization for curve matching

- Subtract the average value x_0: centroid
- Rescale each descriptor to normalize the curve to unit length s: [0,1]
- Take Fourier transform of each curve, treating each (x,y) as a complex number

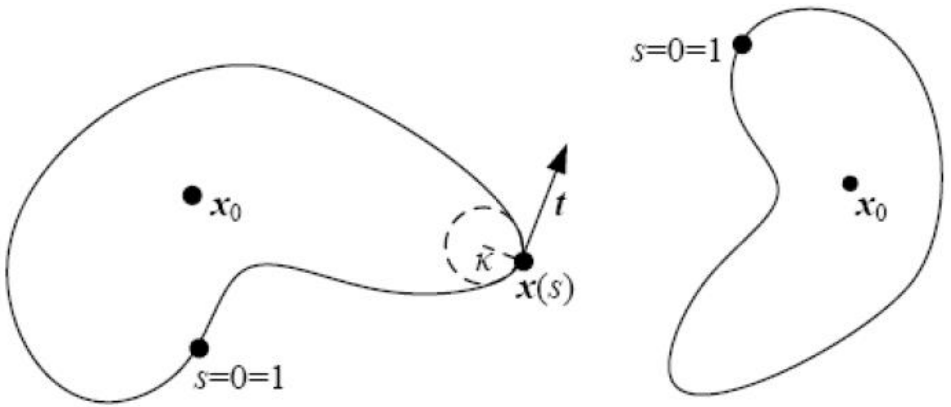


Curve matching

Arc-length parameterization for curve matching

- Subtract the average value x_0: centroid
- Rescale each descriptor to normalize the curve to unit length s: [0,1]
- Take Fourier transform of each curve, treating each (x,y) as a complex number

- The resulting Fourier transforms should differ only by a scale change in magnitude plus a constant complex phase shift, due to rotation, and a linear phase shift in the domain, due to different starting points for s



Lines

Straight lines in man-made world
- Building, furniture

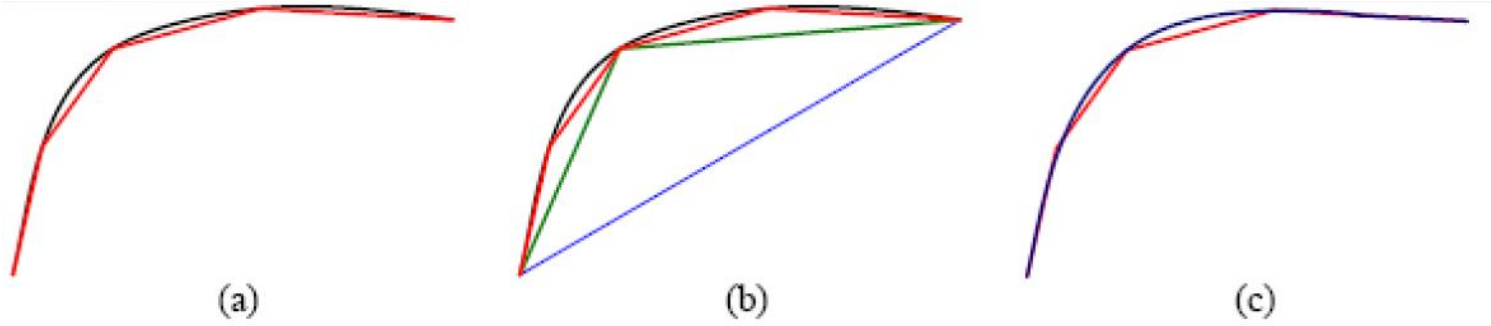
Detect and match lines
- Approximate a curve as a piecewise-linear polyline
- Hough transform
- Vanishing point

Successive approximation

Line simplification

- Recursively subdivide the curve at the point furthest away from the line joining the two endpoints

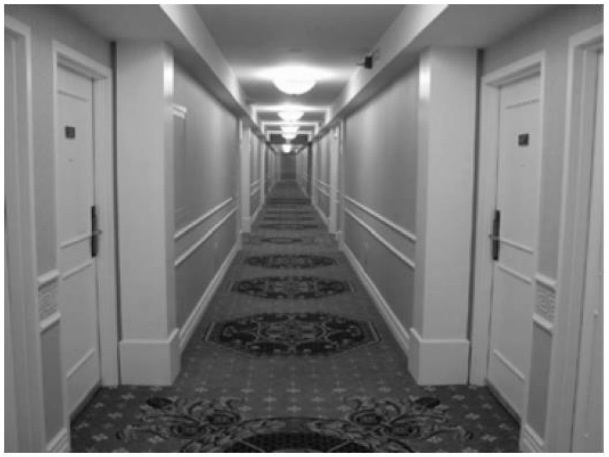
Smooth interpolating splines



Hough Transform

It can be used to:

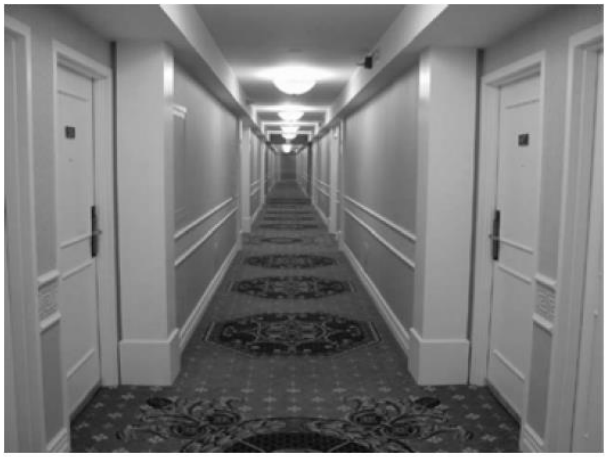
- detect lines, circles or other parametric curves
- give robust detection under noise and partial occlusion



Hough Transform

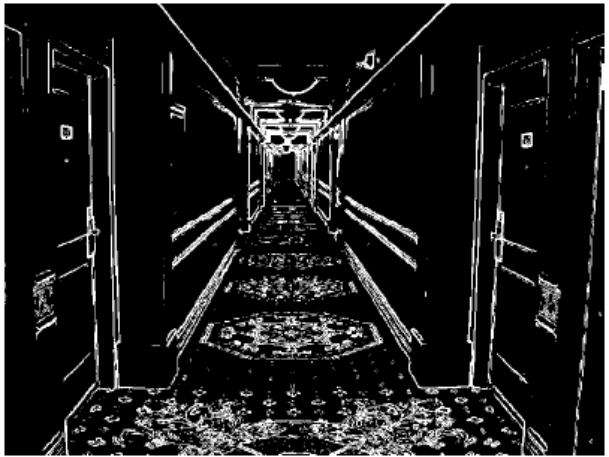
It can be used to:

- detect lines, circles or other parametric curves
- give robust detection under noise and partial occlusion



Edge detection is used as pre-processing step:

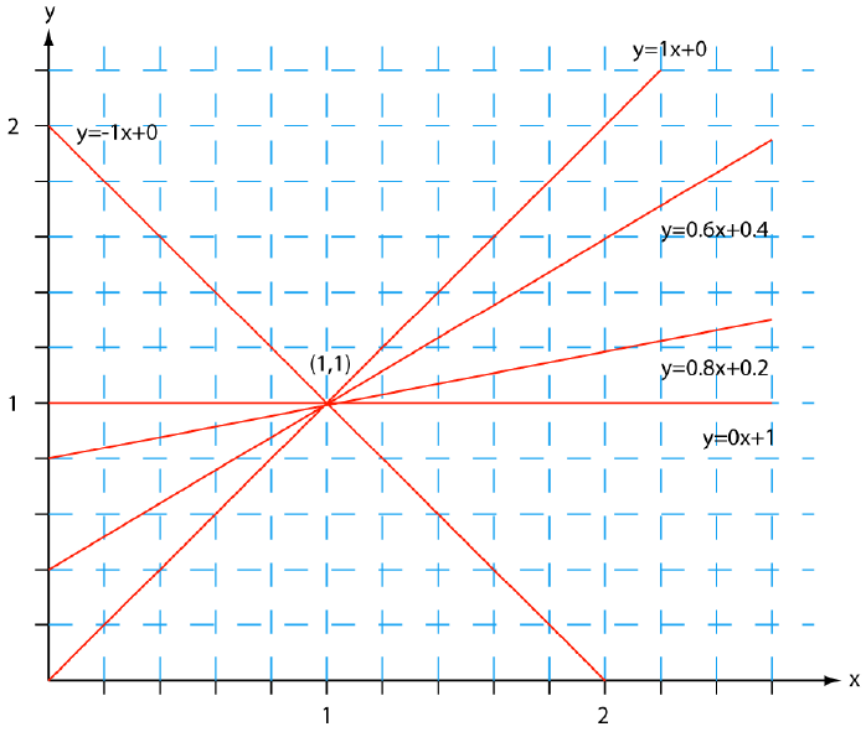
- Compute edge magnitude from input image.
- Threshold the gradient magnitude image to produce a binary edge image



Hough Transform

Basic idea:

- We want to find sets of edge pixels that make a straight line
- For a given point (x_i,y_i) all lines passing through it satisfy $y_i = a\,x_i + b$
- Given different values for a and b, different lines are defined



Hough Transform

Basic idea:

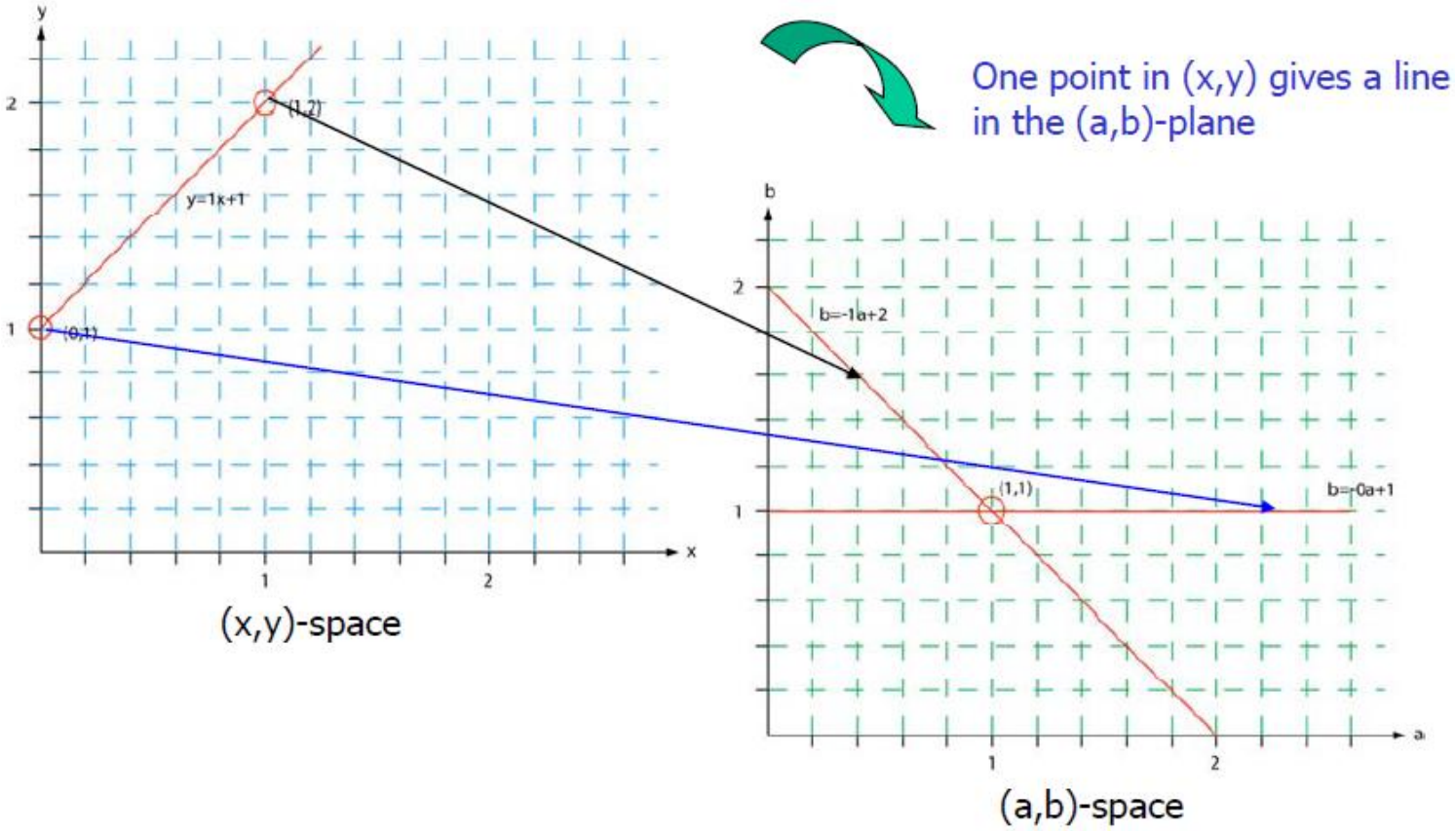
- We want to find sets of edge pixels that make a straight line
- For a given point (x_i,y_i) all lines passing through it satisfy $y_i = a\ x_i + b$
- Given different values for a and b, different lines are defined
- We use the following trick:
  - we re-write the equation of a line as $b = -xa + y$
  - we consider x and y as parameters
  - we consider a and b as variables

Hough Transform

Basic idea:

- We want to find sets of edge pixels that make a straight line
- For a given point (x_i,y_i) all lines passing through it satisfy $y_i = a\ x_i + b$
- Given different values for a and b, different lines are defined
- We use the following trick:
  - we re-write the equation of a line as b = -xa + y
  - we consider x and y as parameters
  - we consider a and b as variables
- Based on the above trick, a line in the (a,b) space corresponds to a point in the (x,y) space

Hough Transform



Hough Transform

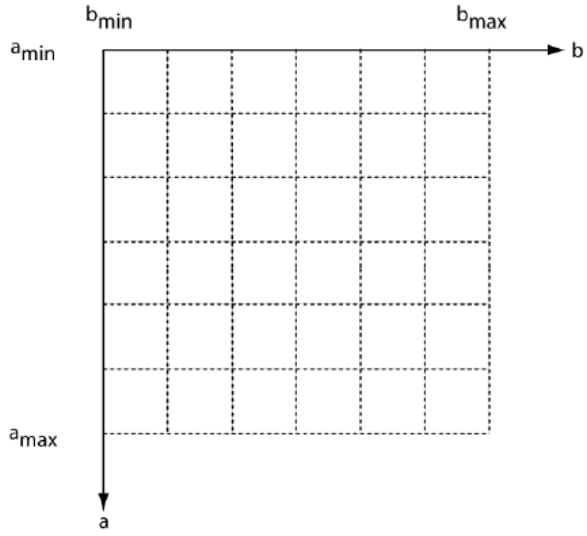
Basic idea:

- We want to find sets of edge pixels that make a straight line
- For a given point (x_i,y_i) all lines passing through it satisfy $y_i = a\,x_i + b$
- Given different values for a and b, different lines are defined
- We re-write the equation of a line as $b = -xa + y$ and consider x and y as parameters and a and b as variables
- Based on the above trick, a line in the (a,b) space corresponds to a point in the (x,y) space
  - Two points (x,y) and (z,k) define a line in the (x,y) plane and two lines in the (a,b) space.
  - In (a,b) space these two lines will intersect in a point (a',b') where a' is the rise and b' the intersect of the line defined by (x,y) and (z,k) in (x,y) space.
  - All points on the line defined by (x,y) and (z,k) in (x,y) space will parameterize lines that intersect in (a',b') in (a,b) space.
  - Points that lie on a line will form a "cluster of crossings" in the (a,b) space.

Hough Transform

Algorithm:

- Quantize the parameter space (a,b) (divide it into cells). The quantized space is often referred to as the accumulator cells.



Hough accumulator cells

Hough Transform

Algorithm:

- Quantize the parameter space (a,b) (divide it into cells). The quantized space is often referred to as the accumulator cells
- Count the number of times a line intersects a given cell
  - For each point (x,y) with value 1 in the binary edge image, find the values of (a,b) in the range $[[a_{min},a_{max}],[b_{min},b_{max}]]$ defining the line corresponding to this point
  - Increase the value of the accumulator bin (square) where the points [a',b'] belong to
  - Then proceed with the next point in the image

Hough Transform

Algorithm:

- Quantize the parameter space (a,b) (divide it into cells). The quantized space is often referred to as the accumulator cells
- Count the number of times a line intersects a given cell
  - For each point (x,y) with value 1 in the binary edge image, find the values of (a,b) in the range $[[a_{min},a_{max}],[b_{min},b_{max}]]$ defining the line corresponding to this point
  - Increase the value of the accumulator bin (square) where the points [a',b'] belong to
  - Then proceed with the next point in the image
- Cells receiving more than a minimum number of "votes" are assumed to correspond to lines in (x,y) space
  - Lines can be found as peaks in the accumulator space

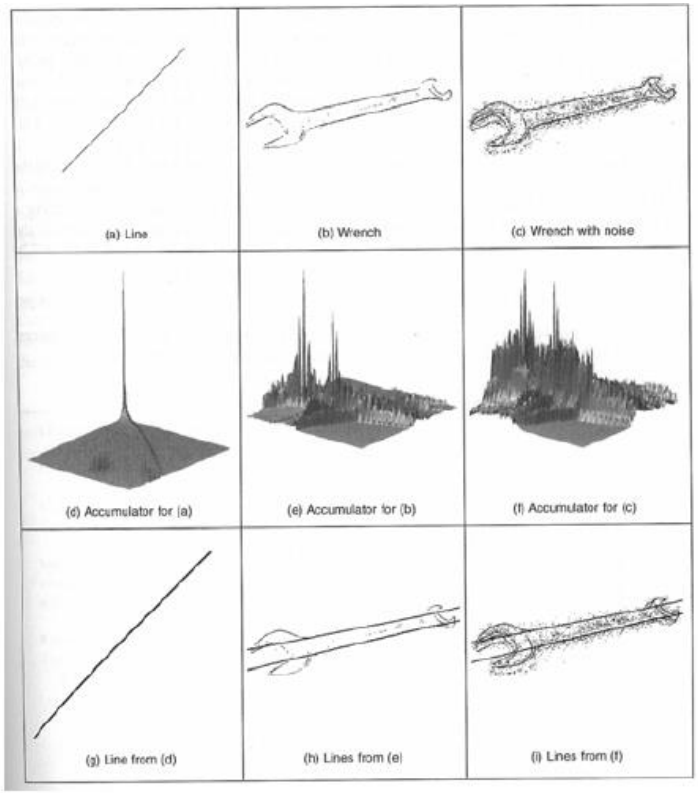
Hough Transform

Examples:

Thresholded edge images

Visualizing the accumulator space
The height of the peak will be defined by the number of pixels in the line.

Thresholding the accumulator space and superimposing this onto the edge image



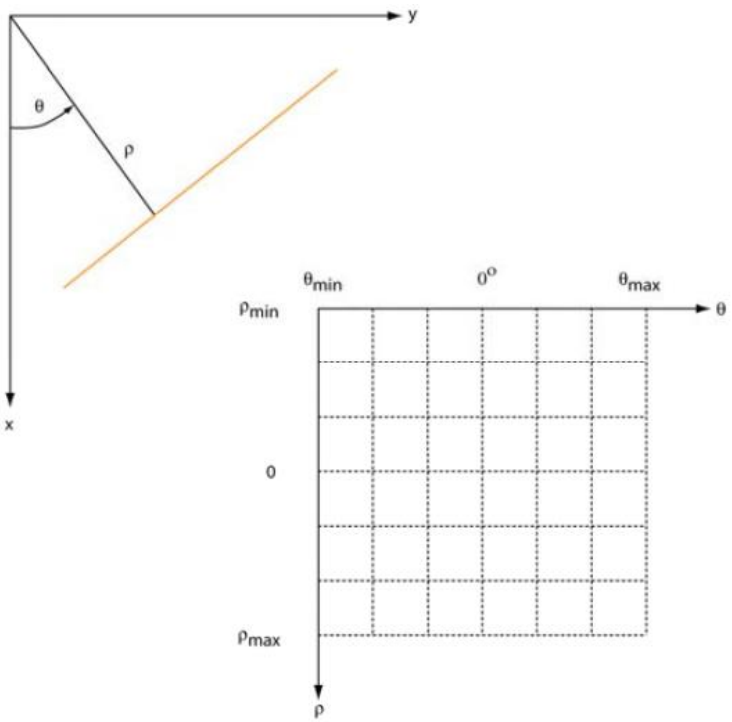
Note how noise effects the accumulator. Still with noise, the largest peaks correspond to the major lines.

Hough Transform

Polar representation of lines:

- In practice we do not use the (x,y) space to represent lines (vertical lines lead to infinite slope, which means that a → ∞)
- We rather use the polar representation of lines

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



Polar representation of lines

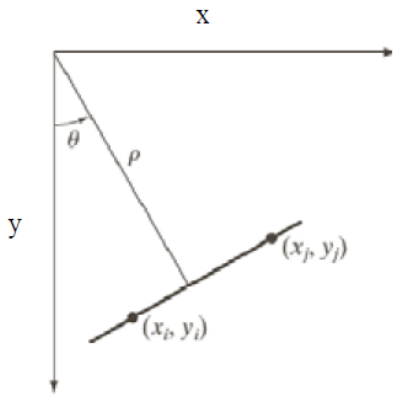
Hough Transform

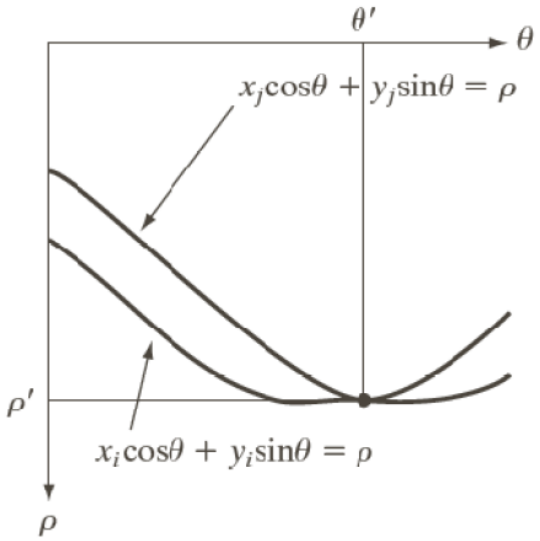
Polar representation of lines:

- In practice we do not use the (x,y) space to represent lines
- We rather use the polar representation of lines

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

- Each point (x_i,y_i) in the (x,y) space gives a sinusoid curve in the (ρ,θ)-space
- M points lying on the same line (i.e. a line with the same ρ and θ parameters) will give M curves that intersect at (ρ_i,θ_i) point in the (ρ,θ) space



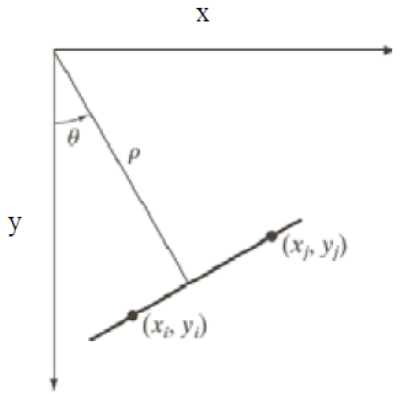


Hough Transform

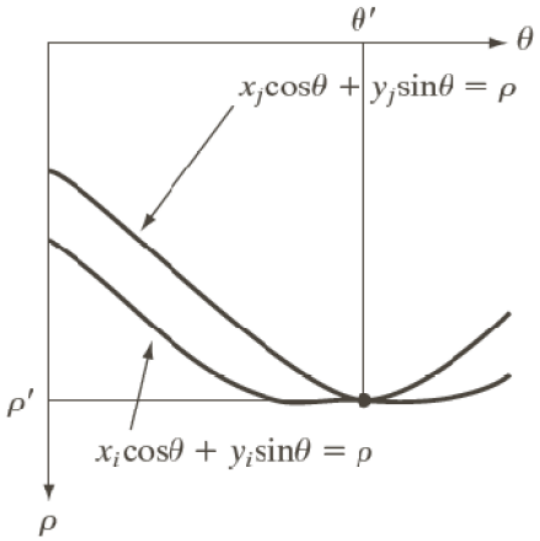
Polar representation of lines:

- In practice we do not use the (x,y) space to represent lines
- We rather use the polar representation of lines

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



- Each point (x_i,y_i) in the (x,y) space gives a sinusoid curve in the (ρ,θ)-space
- M points lying on the same line (i.e. a line with the same ρ and θ parameters) will give M curves that intersect at (ρ_i,θ_i) point in the (ρ,θ) space



- When quantizing the (ρ,θ) space:
  - $\theta \in \{-90^o, 90^o\}$
  - for an image of MxN pixels, $\rho \in \{0, \text{sqrt}(M^2+N^2)\}$

Hough Transform

Algorithm using polar representation:

- Quantize the parameter space (ρ,θ) (divide it into cells)
- The cell (i,j) corresponds to the square associated with parameter values (θ_j, ρ_i)
- Initialize all cells with value 0
- For each foreground point (x_k,y_k) in the thresholded edge image
  - Let θ_j equal all the possible θ-values
    - Solve for ρ using $\rho = x \cos \theta_j + y \sin \theta_j$
    - Round ρ to the closest cell value, ρ_q
    - Increment A(i,q) if the θ_j results in ρ_q
- After this procedure, A(i,j)=P means that P points in the (x,y) space lie on the line $\rho_j = x \cos \theta_j + y \sin \theta_j$
- Find line candidates where A(i,j) is above a suitable threshold value.

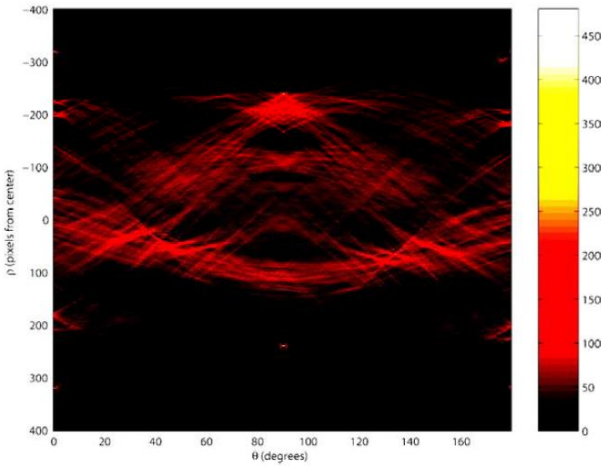
Hough Transform

Image

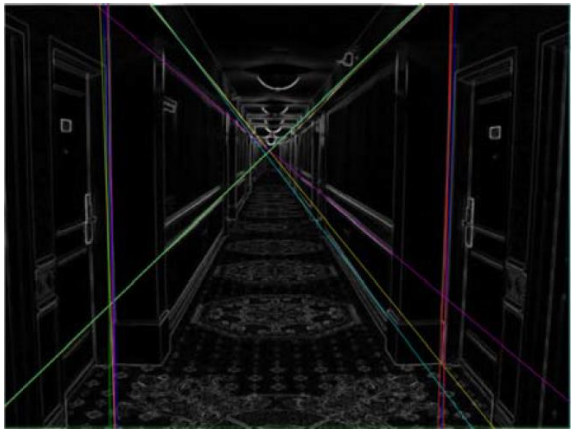
Edge image

Thresholded edge image

accumulator image



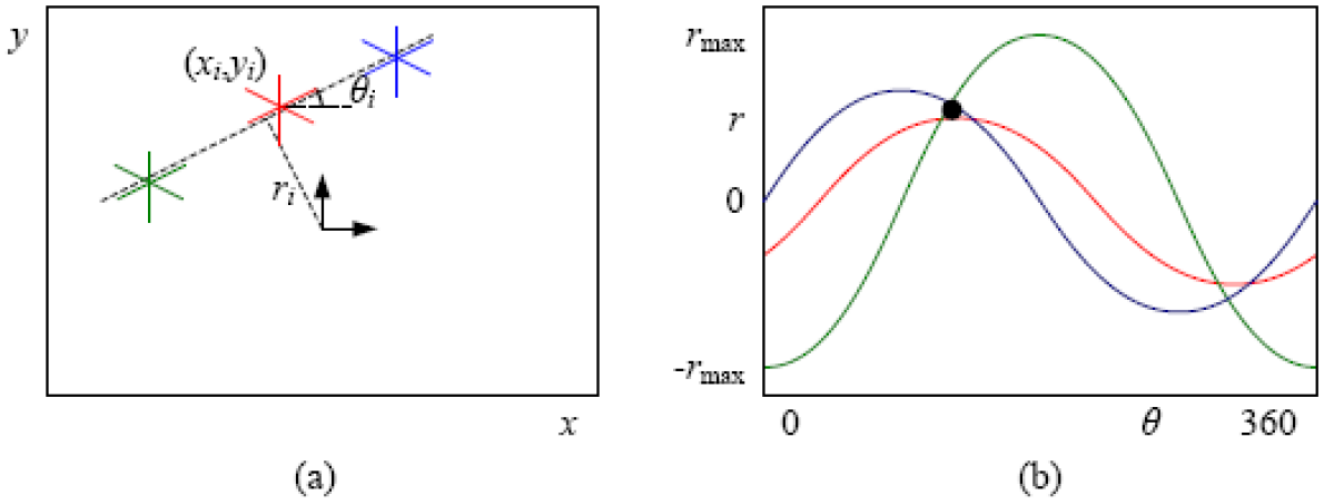
20 most prominent lines



Hough Transform

Each point votes for all possible lines passing through it

Lines corresponding to high accumulator or bin values are examined for potential line fits



In general, there is no clear consensus on which line estimation technique performs best. It is therefore a good idea to think carefully about the problem at hand and to implement several approaches (successive approximation, Hough, etc.) to determine the one that works best for your application.