Edge Detection and Active Contours

CS 585, February 22, 2024
Lecture by Margrit Betke

The Causes of Brightness Changes in an Image

The Causes of Brightness Changes in an Image

1. Occluding boundary
2. Changes in surface orientation
3. Changes in surface reflectance
4. Illumination discontinuity

Edge Detection

Definition:

Edges = brightness changes (discontinuities) in an image

due to

- occlusion
- changes in surface orientation
- changes in surface reflectance (material)
- illumination discontinuity

1D Case: Image E(x) = grey value = brightness

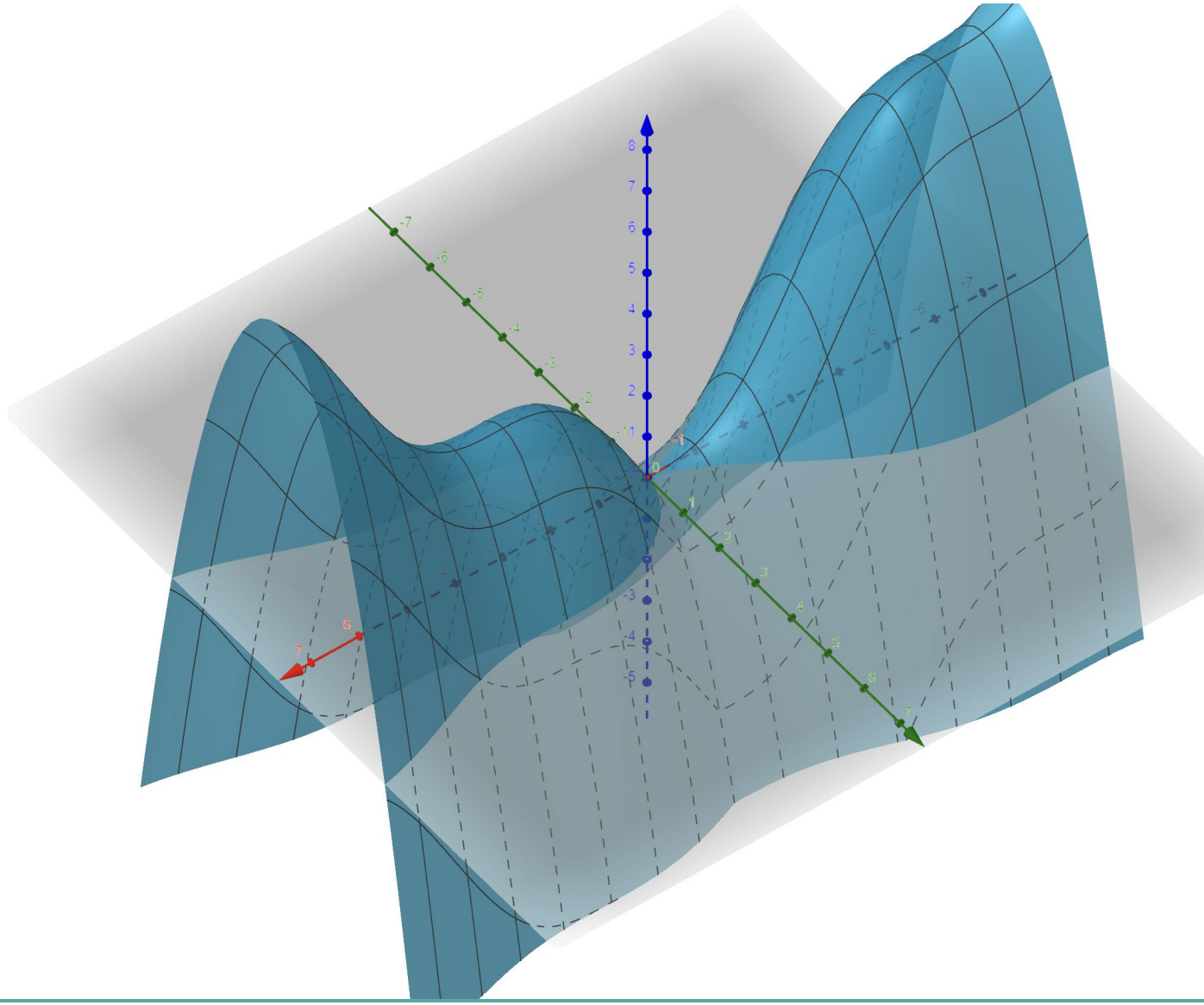
$E(x)$

$dE(x)/dx$

$d^2E(x)/dx^2$

2D Case: E(x,y)

Direction and magnitude of brightness change vector: Gradient of brightness



Edge is perpendicular to gradient

Discrete Approximation of Derivatives = Finite Differences

E(r,s+1)	
E(r,s)	E(r+1,s)

$$E(r+1,s) - E(r,s)$$

$$E(r,s+1) - E(r,s)$$

Discrete Approximation of Derivatives = Finite Differences

E(r,s+1)	
E(r,s)	E(r+1,s)

$$E(r,s+1) - E(r,s)$$

1
-1

$$E(r+1,s) - E(r,s)$$

-1	1

Which is $\partial E / \partial x$? Which $\partial E / \partial y$?

Better Approximation of Brightness Derivatives

E(r,s+1)	E(r+1, s+1)
E(r,s)	E(r+1,s)

$$E(r+1, s+1) - E(r,s+1) + E(r+1,s) - E(r,s)$$

$$E(r+1, s+1) - E(r+1,s) + E(r,s+1) - E(r,s)$$

Better Approximation of Brightness Derivatives

E(r,s+1)	E(r+1, s+1)
E(r,s)	E(r+1,s)

Measuring Horizontal edges $E(r+1, s+1) - E(r+1,s) + E(r,s+1) - E(r,s)$

1	1
-1	-1

Vertical edges $E(r+1, s+1) - E(r,s+1) + E(r+1,s) - E(r,s)$

-1	1
-1	1

Simple Edge Detection Algorithm

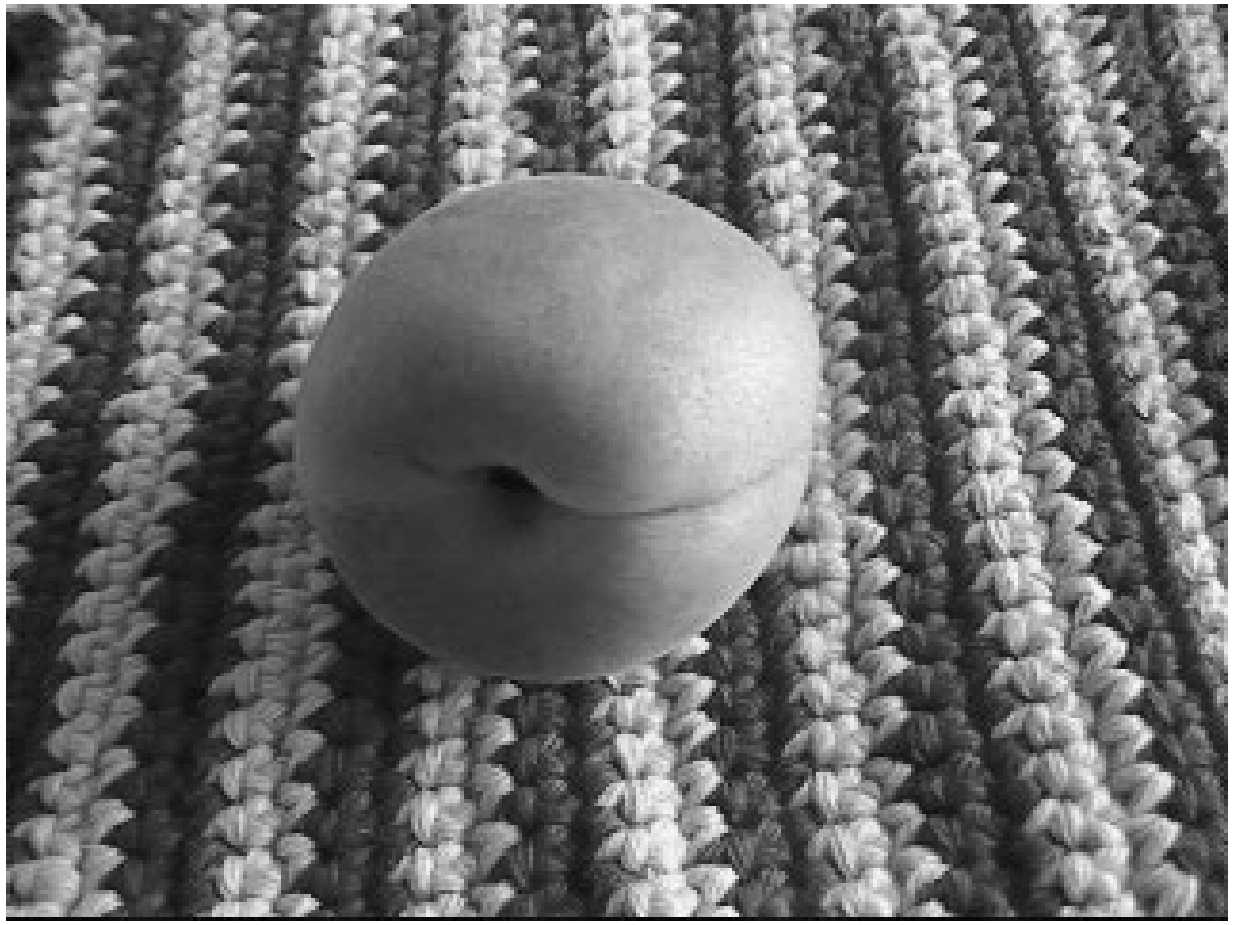
1. Compute $\partial E/\partial x$ and $\partial E/\partial y$ to determine brightness gradient direction.

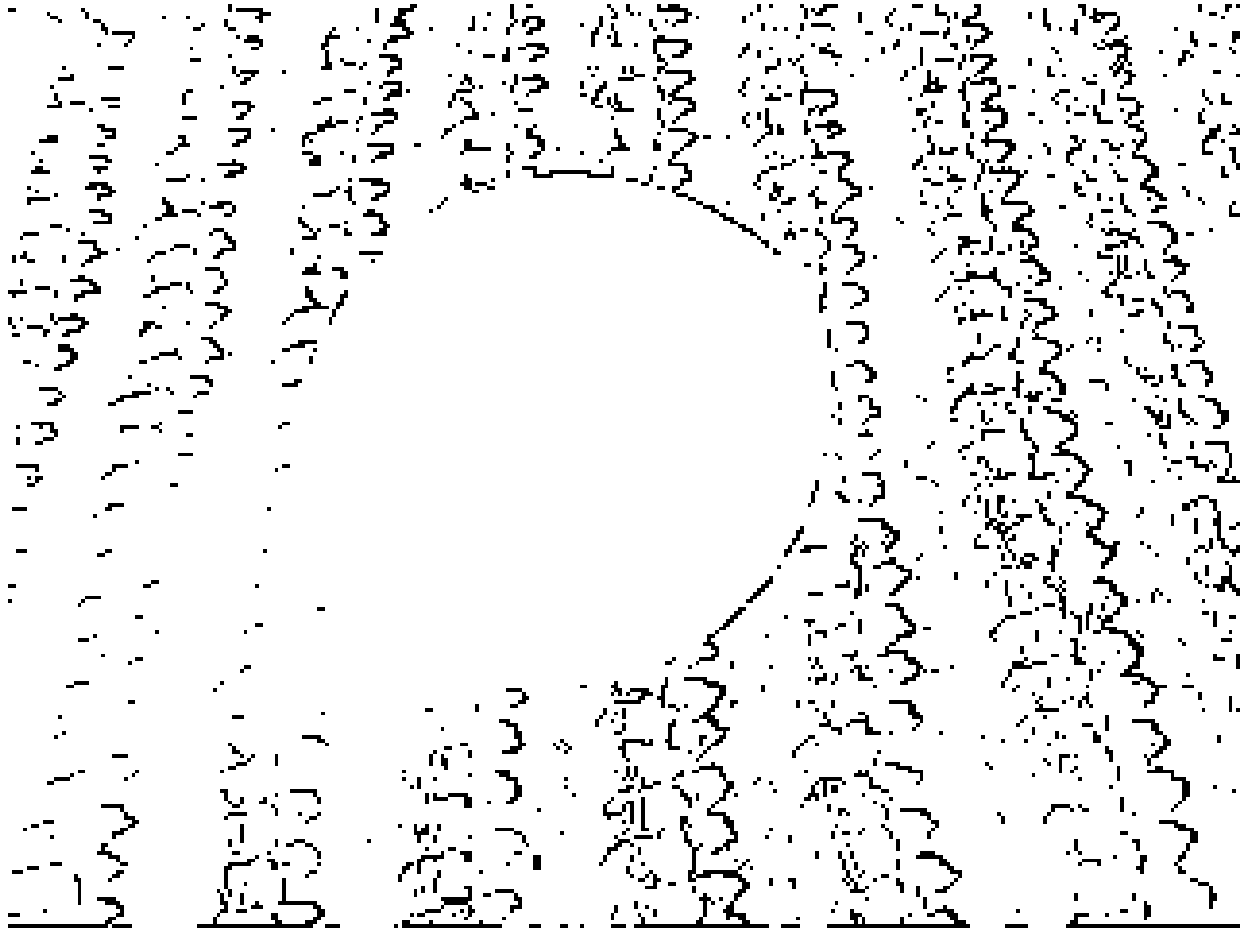
2. To compute "edge strength" M, use
   - $|\partial E/\partial x| + |\partial E/\partial y|$ or
   - $(\partial E/\partial x)^2 + (\partial E/\partial y)^2$ or
   - magnitude $\text{sqrt}\{(\partial E/\partial x)^2 + (\partial E/\partial y)^2\}$

3. Compute binary edge map:

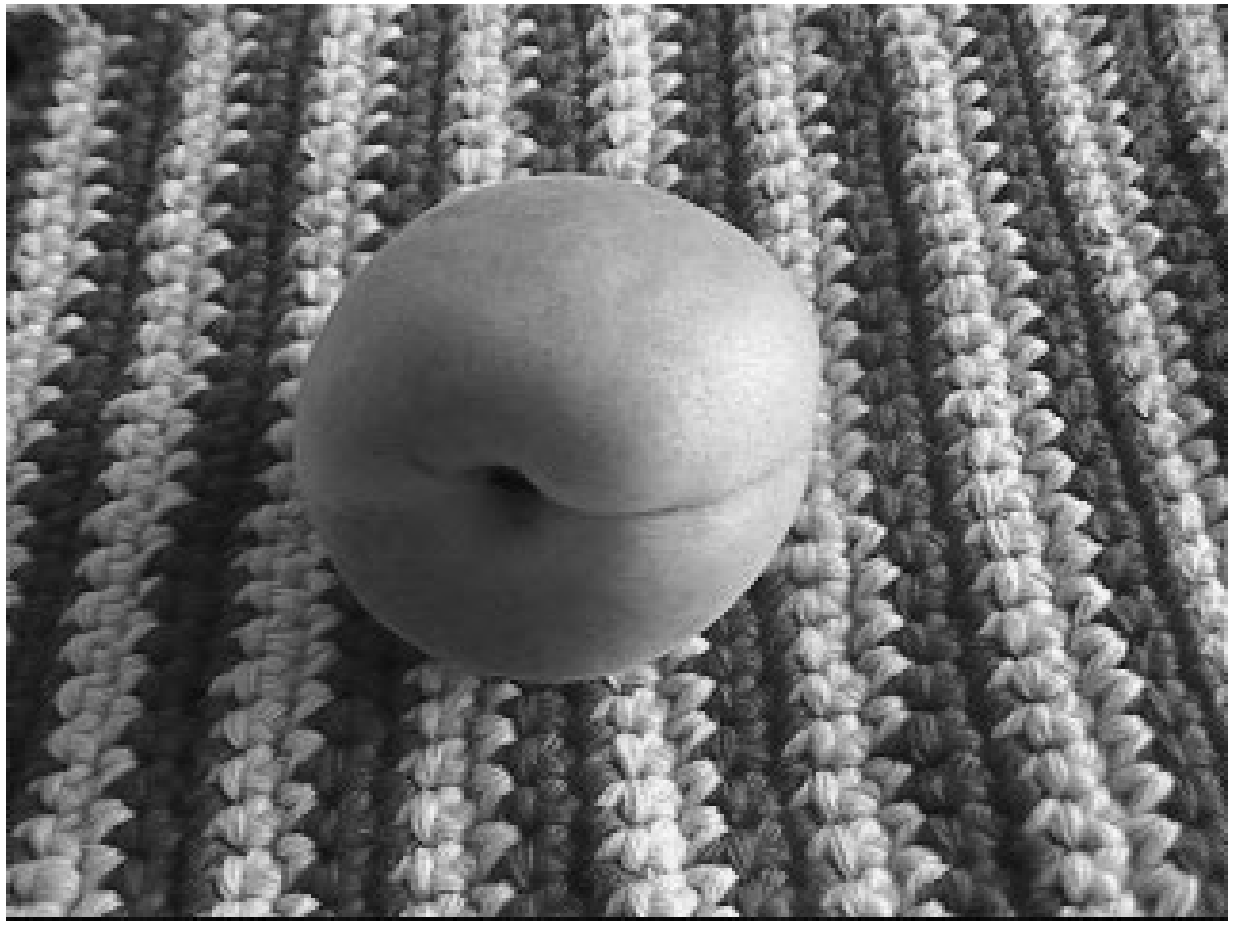
IF $M > \text{threshold}$,
    $\text{EdgeMap}(x,y) = 1;$
ELSE
    $\text{EdgeMap}(x,y) = 0;$

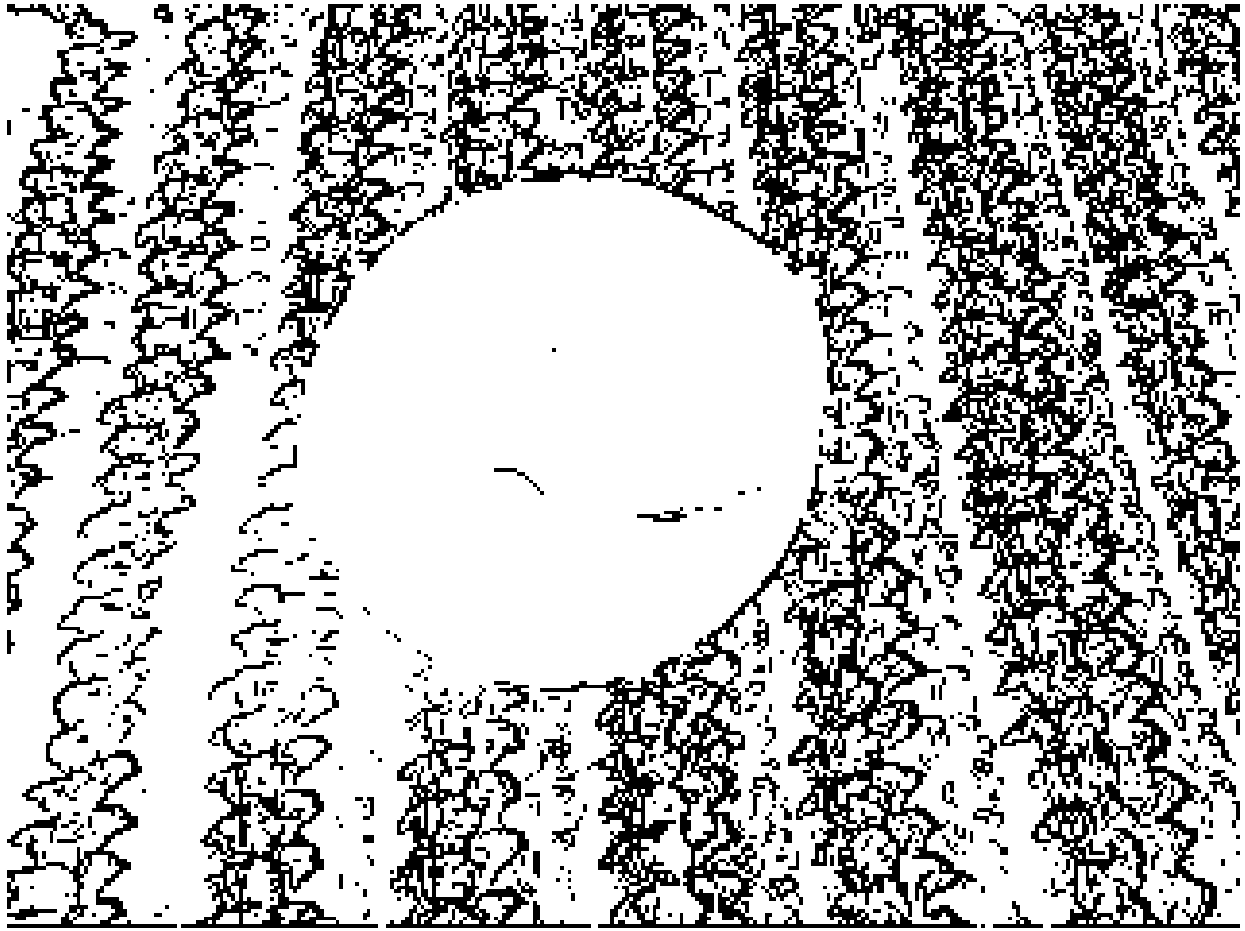
High Threshold on Magnitude of Brightness Change



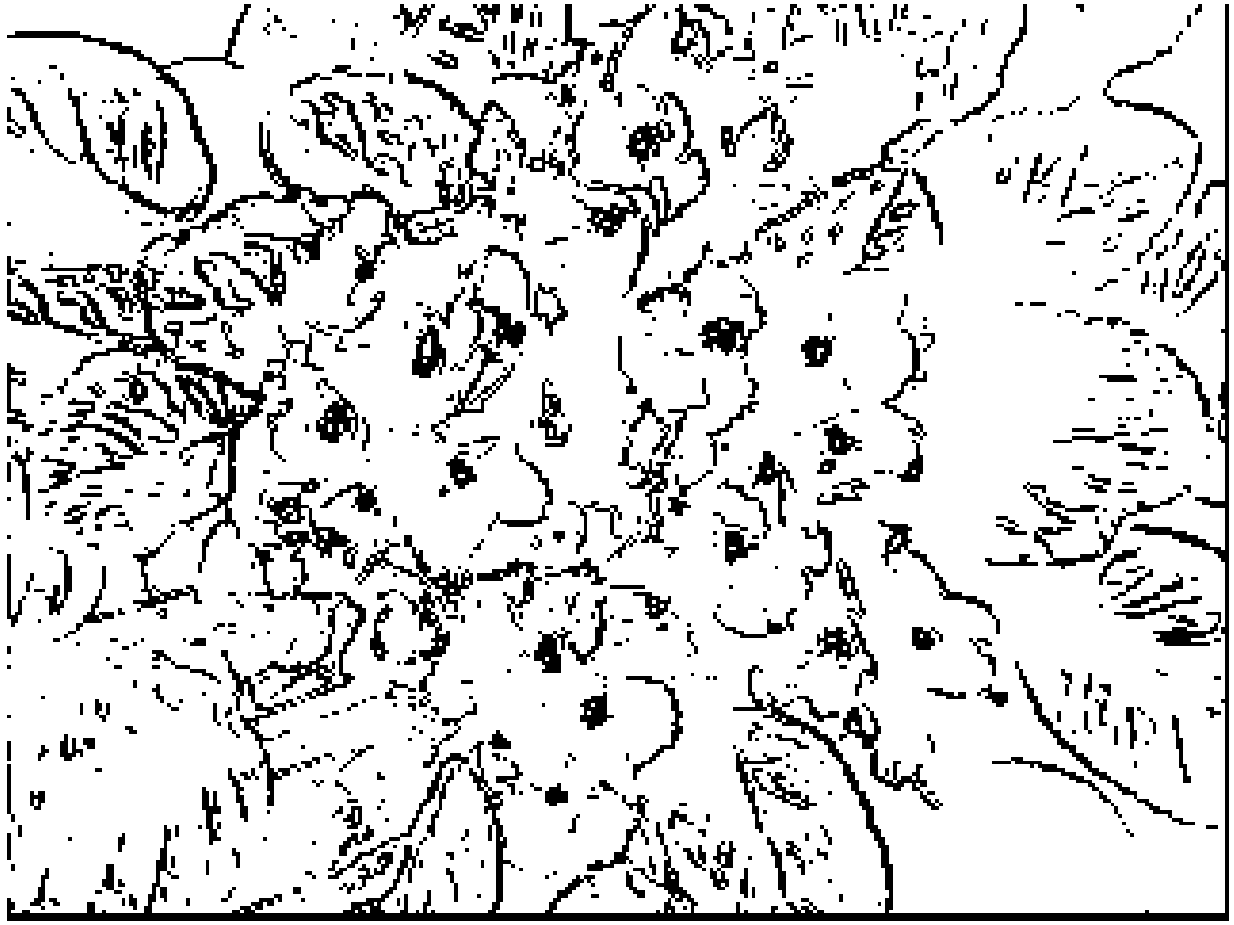


Low Threshold on Magnitude of Brightness Change

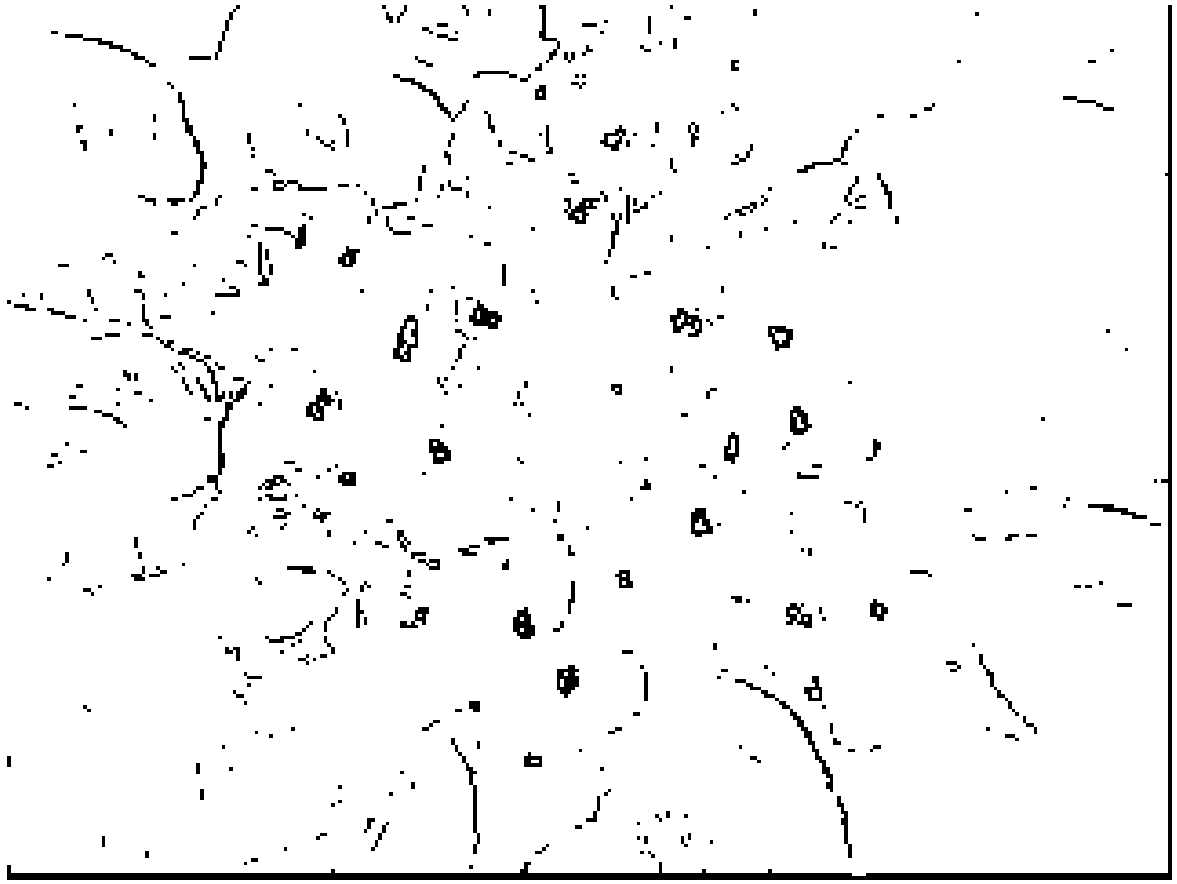




Low Threshold on Magnitude of Brightness Change



High Threshold on Magnitude of Brightness Change



Other Commonly Used Edge Masks to Approximate the Brightness Gradient

Approximating	Roberts	Prewitt	Sobel	for Measuring:
$\partial E/\partial x$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	vertical edges
$\partial E/\partial y$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	horizontal edges

Approximating 2nd Derivatives of Brightness

Difference of differences:

$$\partial^2 E/\partial x^2 = 1/\varepsilon^2 \; \{ [E(r-1,s) - E(r,s)] - [E(r,s) - E(r+1,s)] \} =$$

$$1/\varepsilon^2 \; \{ E(r-1,s) - 2\,E(r,s) + E(r+1,s) \}$$

$$\partial^2 E/\partial y^2 = 1/\varepsilon^2 \; \{ E(r,s-1) - 2\,E(r,s) + E(r,s+1) \}$$

where

E(r-1,s+1)	E(r,s+1)	E(r+1,s+1)
E(r-1,s)	E(r,s)	E(r+1,s)
E(r-1,s-1)	E(r,s-1)	E(r+1,s-1)

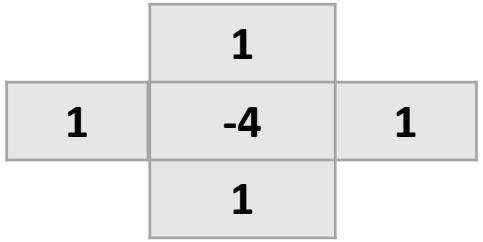
Edge Detection via Laplacian at Center Cell

The Laplacian of E(x,y) is defined as $\partial^2 E/\partial x^2 + \partial^2 E/\partial y^2$.

Approximation of the Laplacian at center cell of 9-pixel window:

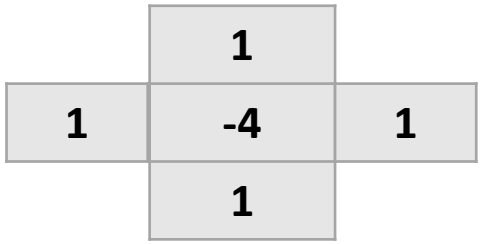
$4/\varepsilon^2$ { ¼ [E(r-1,s) + E(r, s-1) + E(r+1,s) + E(r,s+1)] - E(r,s) }

Mask:

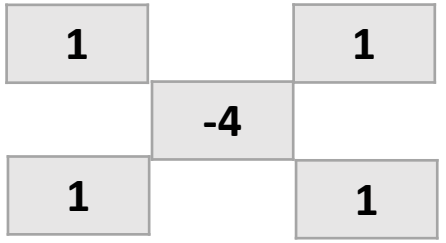


Edge Detection via Laplacian at Center Cell

Mask:



Mask rotated by 45 degrees:



Accurate approximation of the Laplacian (linear combination of above):

1	4	1
4	-20	4
1	4	1

Edge Detection Algorithm:

1) Approximate Laplacian
2) Find zero crossings

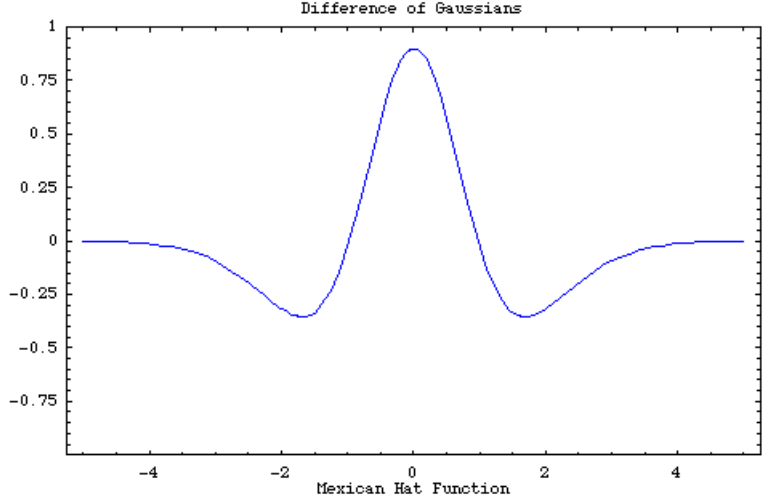
1	**4**	**1**
4	**-20**	**4**
1	**4**	**1**

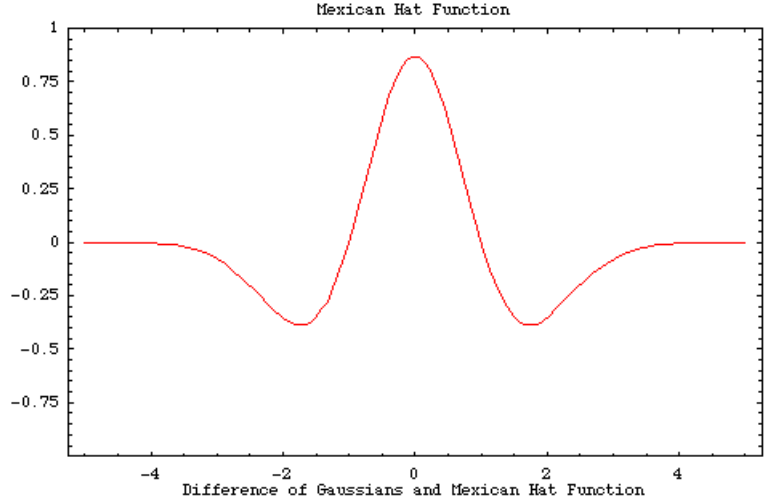
Approximation by

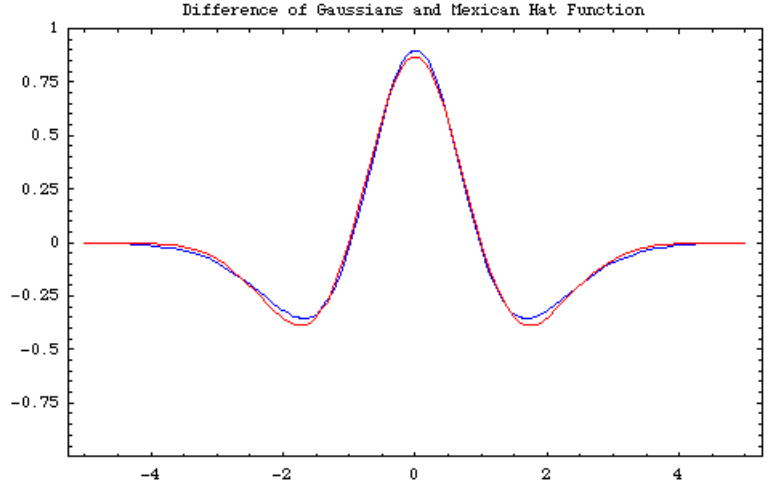
Difference of Gaussians

Or

“Mexican Hat Function”







Source of plot: Wikipedia, Justin Wick

Image Smoothing

Use Image Masks multiple times:

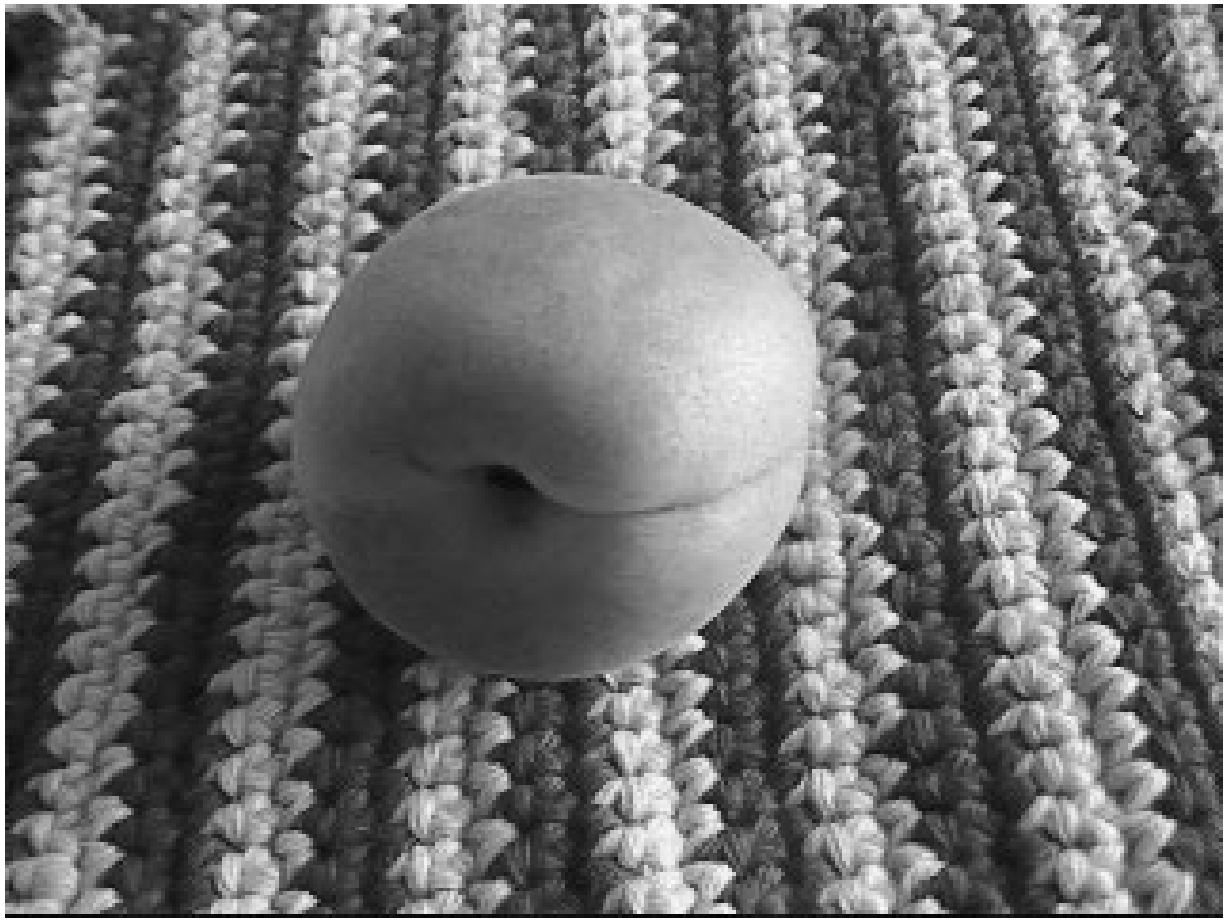
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \quad \text{or} \quad \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix} \quad \text{or} \quad \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

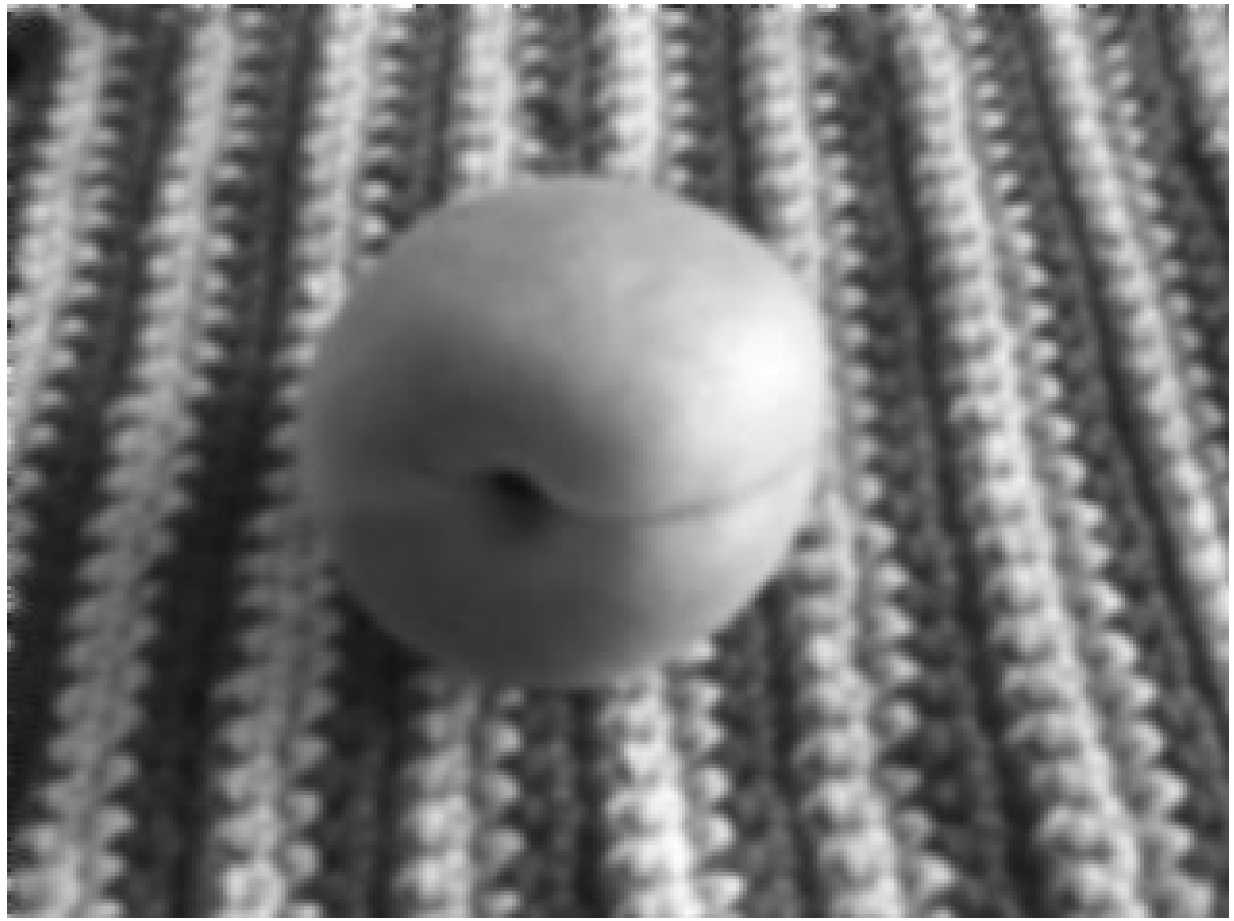
Or use a "Gaussian Mask," for example:

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

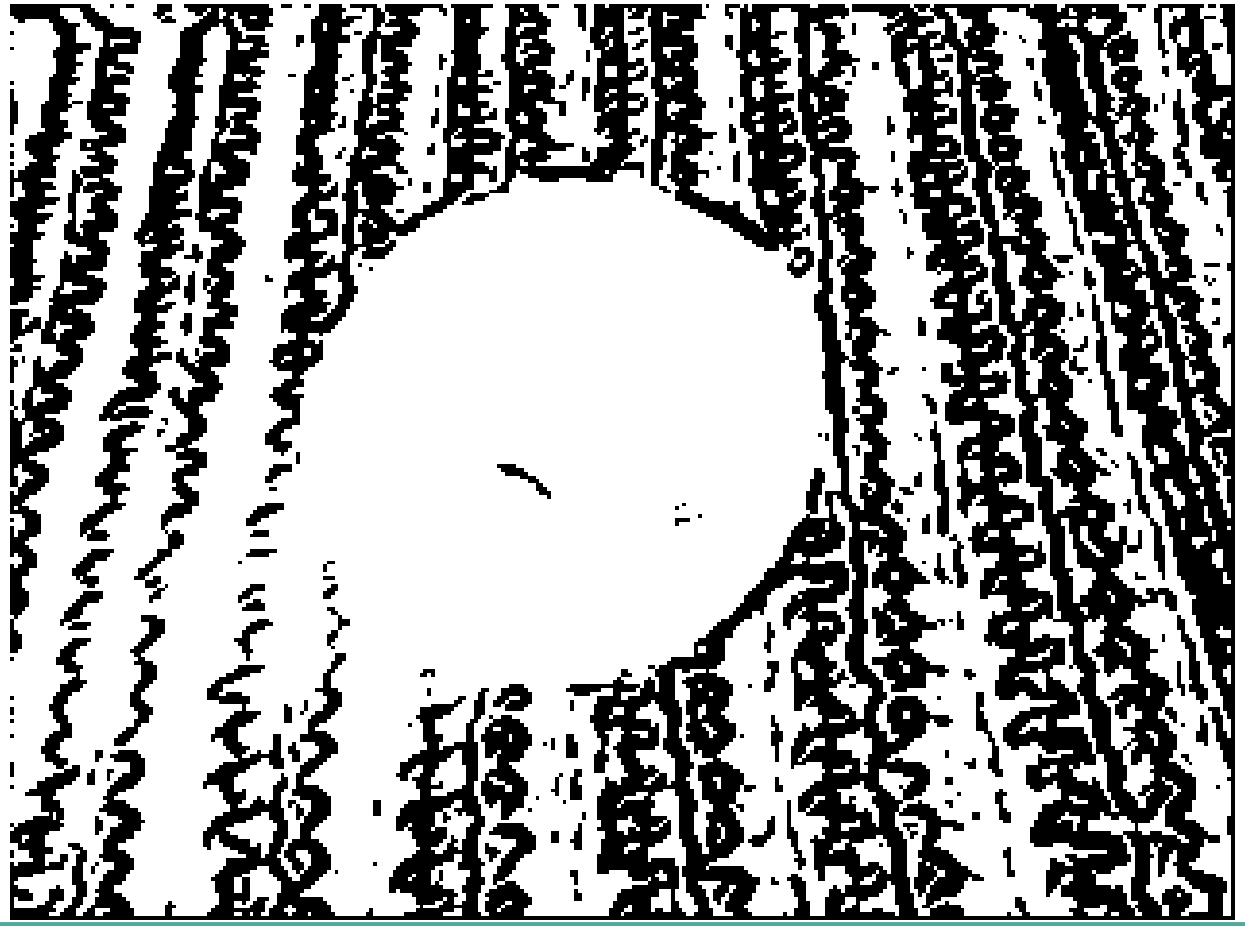
Smoothing Applied to our Example

Smoothing Applied to our Example

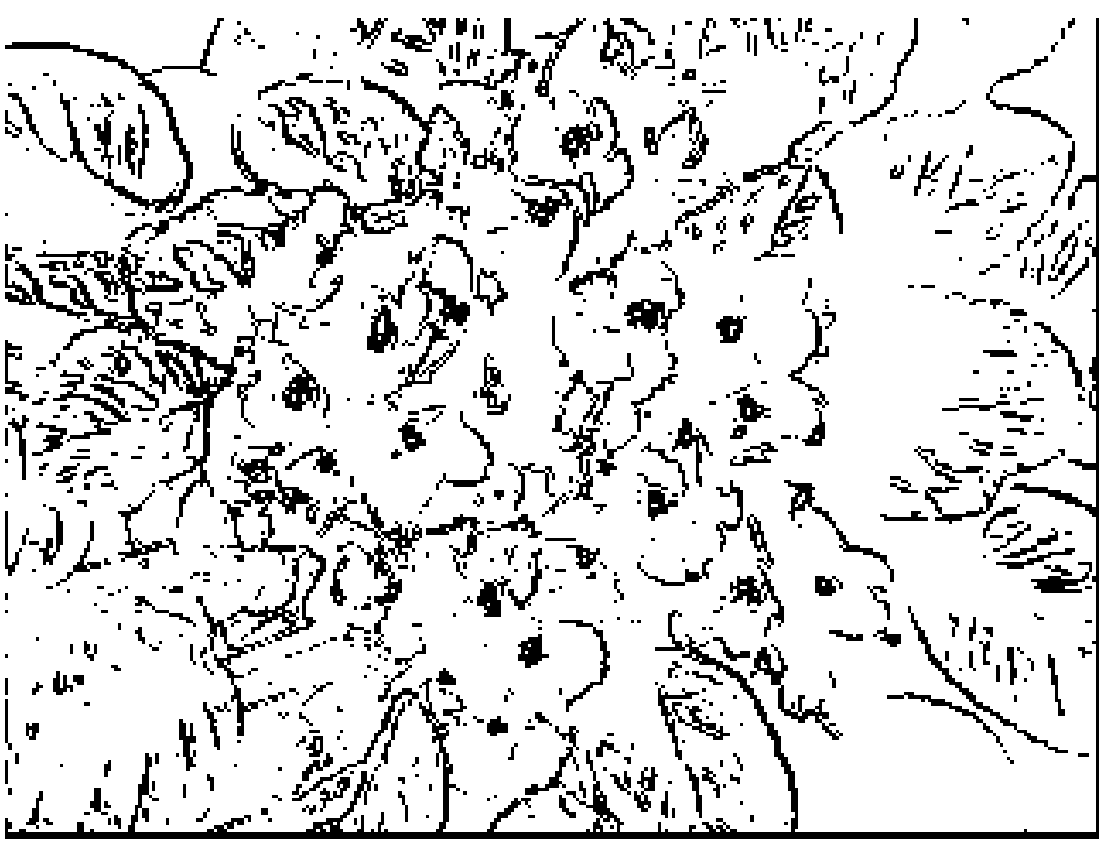




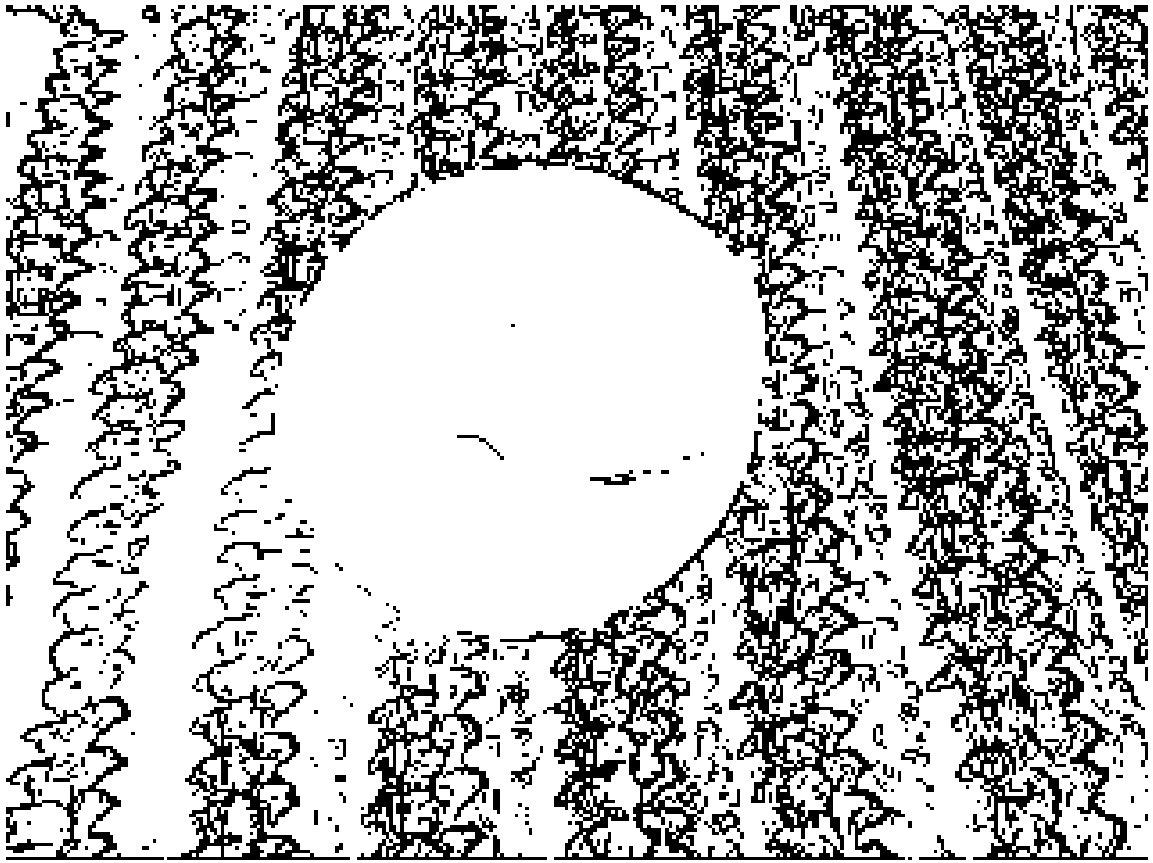
Edge Detection on Smoothed Images

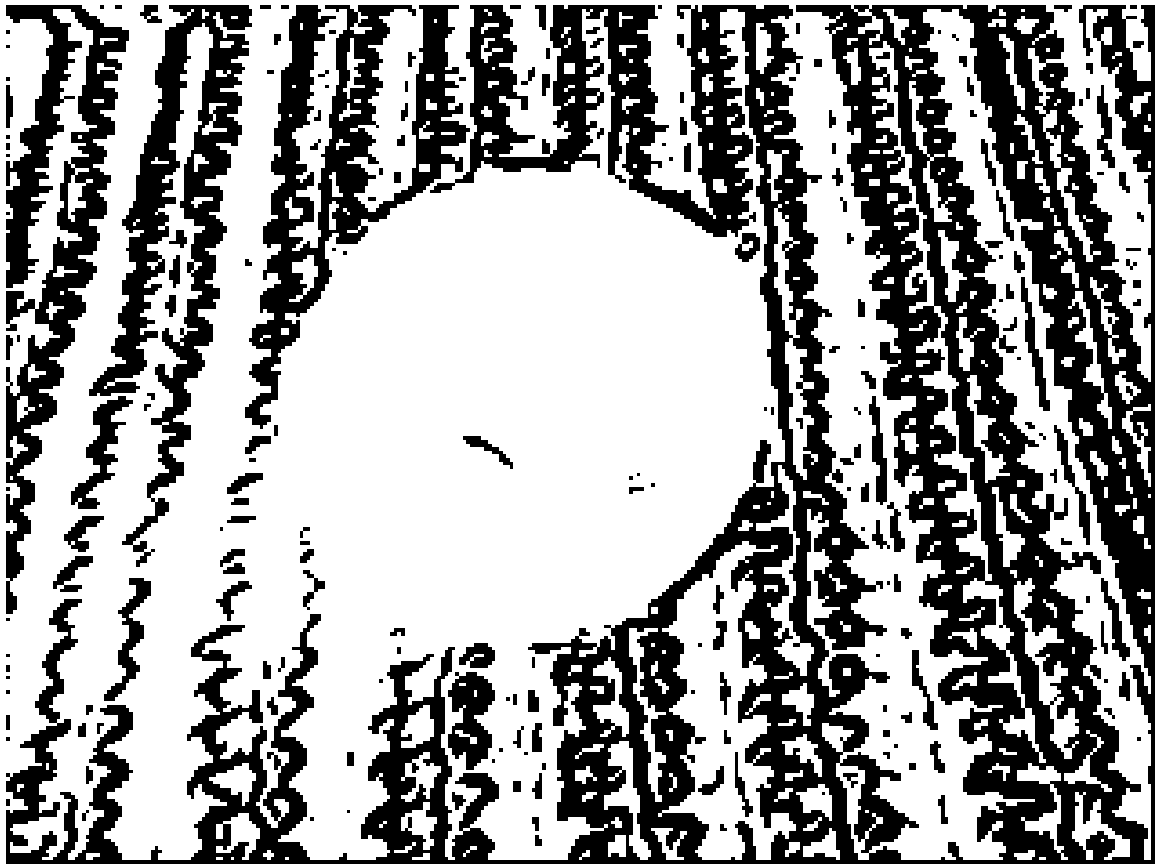


Edge Detection on Original and Smoothed



Edge Detection on Original and Smoothed





Smoothed Image yields “thick” edges

Solution:

We need a “non-maximum suppression algorithm:”

For vertically pointing brightness gradient:

10
5
3

10
5
3

10
5
14

10
0
14

Canny Edge Detection

1. Smooth image with Gaussian filter
2. Compute gradient magnitude map $M(x,y) = \text{sqrt}((dI/dx)^2+(dI/dy)^2)$
   & gradient direction map $\theta(x,y) = \arctan(dI/dx, dI/dy)$
3. Apply "Nonmaximum Suppression" to M:
   a. Reduce number of angles into 4 sectors: $\theta(x,y)$ -> $\alpha(x,y)$
   b. Scan through M(x,y) with a 3x3 mask & check 3 pixels (A,B,C) along the line defined by $\alpha(x,y)$: If $M(B) \le M(A)$ and $M(B) \le M(C)$, then set M(B) to zero.
4. Apply "Double Thresholding:"
   a. Initialize: Choose $\tau_2 \approx 2\ \tau_1$; Copy M(x,y) into $T_1(x,y)$ & $T_2(x,y) = M(x,y)$
   b. If $M(x,y) < \tau_1$ then $T_1(x,y) = 0$. If $M(x,y) < \tau_2$ then $T_2(x,y) = 0$.
   c. Link gaps in T_2 by gathering edges from T_1 (compare N8 neighbors)

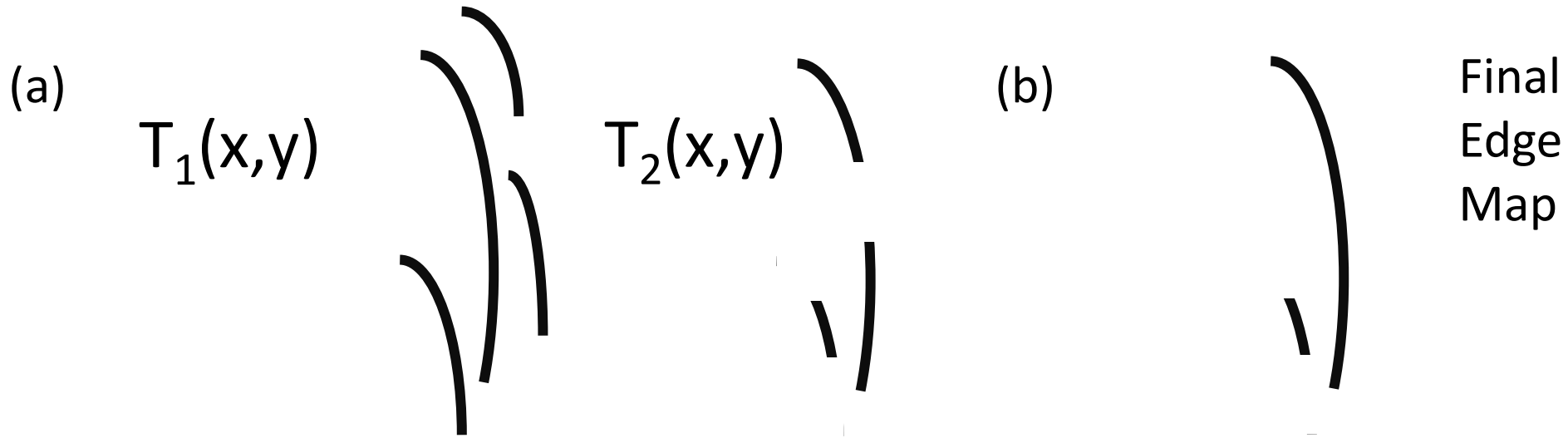
Canny Edge Detection Result

Long edges along hair strands
Closed edge along chin & clothes

Image source: Wikipedia, JonMcLoone

How does Canny Edge Detection work?

1. Smoothing removes speckle noise but creates thicker edges
3. This step thins edges
4. This step performs noise reduction and edge linking:

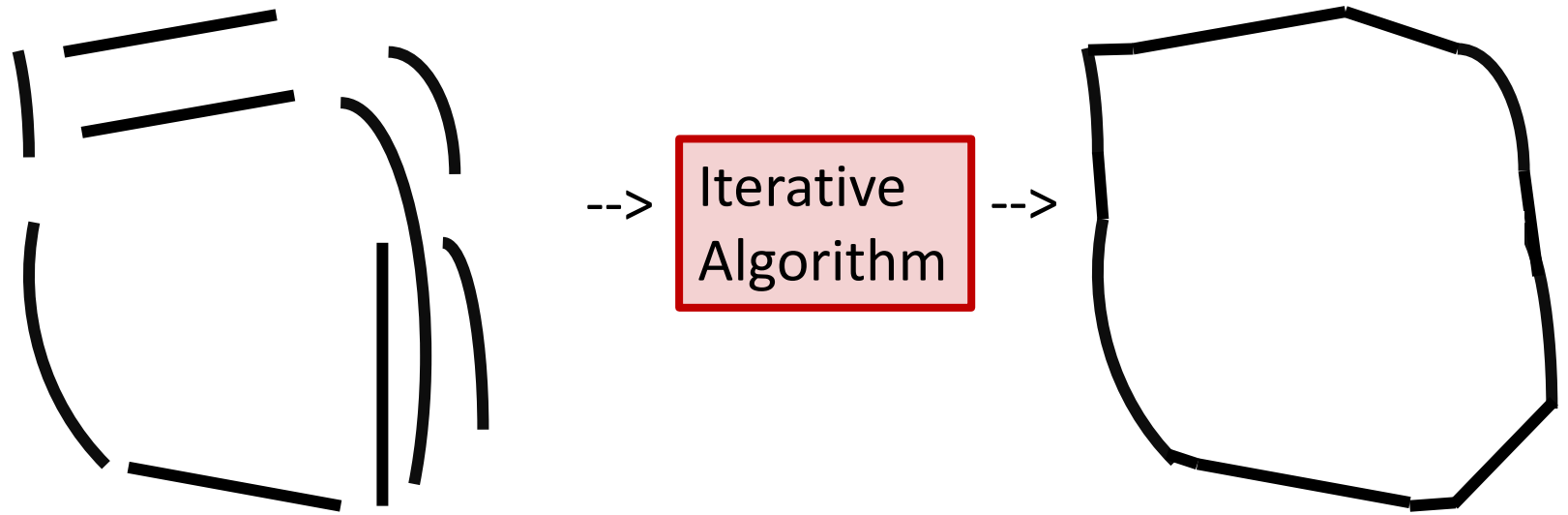


Active Contours (also called "Snakes")

Goal: Given an edge image of an object,

Find the outline of the object = Find its contour points

Idea: Combine disconnected edges (gaps), allow corners



The Active Contour Algorithm is an Iterative Optimization Algorithm

Input: Greyscale image E(x,y); Output: Binary contour map

1. Initial Solution for Contour (e.g., large box, not necessarily bounding)
2. Evaluate cost function (or "energy" function) for
   - Fit of contour with edge image E_{image}
   - Curvature properties of contour $E_{curvature}$
   - Distances of contour points to each other $E_{continuity}$
3. Move contour point
4. Repeat 2.

Possible termination conditions:

Upper bound on # iterations or on points moved OR
Lower bound on cost function or change in cost

"Energy" Function of Active Contour Algorithm

$E_j = \alpha_i \; E_{continuity,\, j} + \beta_i \; E_{curvature,\, j} + \gamma_i \, E_{image,\, j}$

where

α_i , β_i, γ_i control the relative influence of each term

e.g., $\beta_i = 0$ at corner point

Explanation of each energy follows:

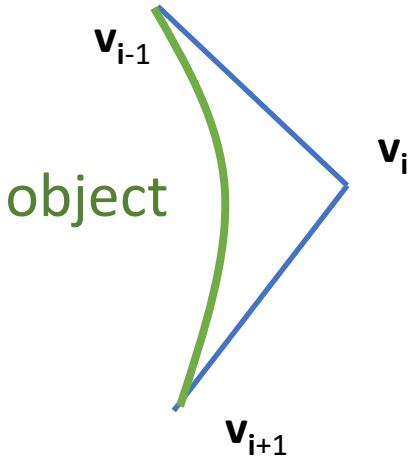
$E_{curvature,\, j}$: Use one of the 4 definitions from our previous lecture

Read the paper by Willams and Shah on active contours

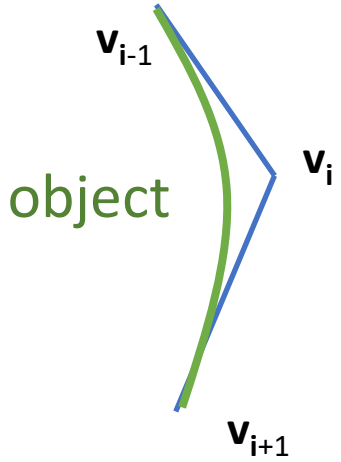
Energy Function of Active Contour Algorithm

$$E_j = \alpha_i \, E_{continuity,\, j} + \beta_i \, E_{curvature,\, j} + \gamma_i \, E_{image,\, j}$$

$E_{image,\, j}$ is the edge component of the energy function, e.g., use the Sobel mask to approximate the squared brightness gradient

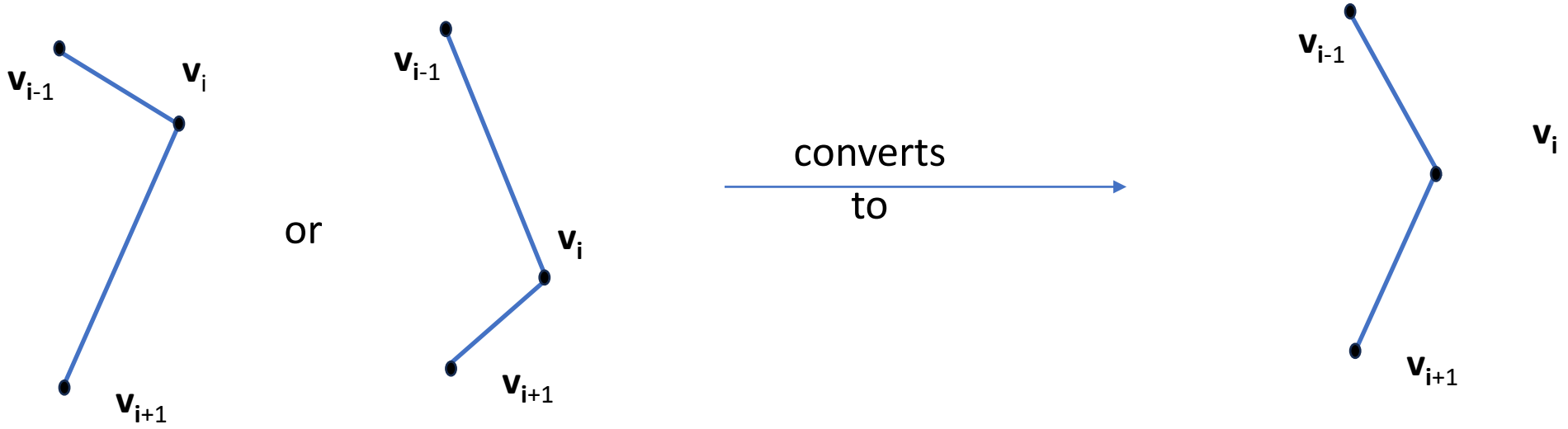


We want v_i closer to object boundary. Among 8 neighbors of v_i, Find neighbor with higher E_{image}



Energy Function of Active Contour Algorithm

The continuity term $E_{continuity,\ j}$ ensures even spacing of contour points:

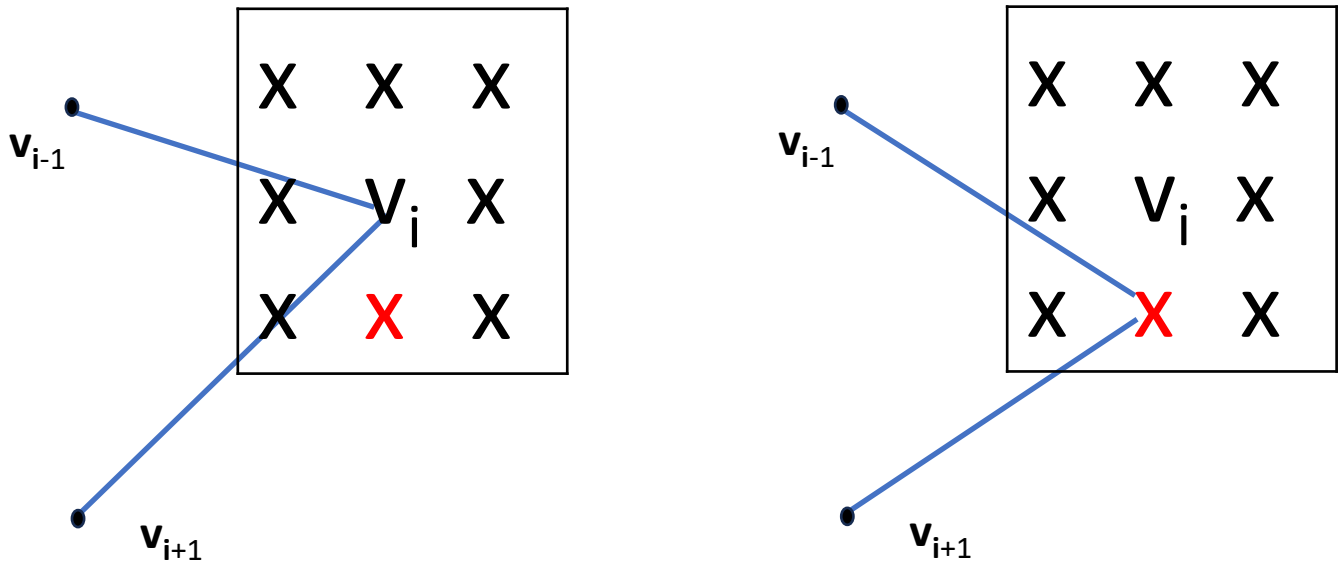


d_j = average distance between all points on curve in jth iteration

Check $d_j - |v_i - v_{i-1}|$ at each of the 8 neighbors of curve point v_i

Energy Function of Active Contour Algorithm

The continuity term $E_{continuity,\ j}$ ensures even spacing of contour points by checking 8 neighbors of contour point v_i and reassigning it if necessary:



At position x,
E_j is minimum:

Reassign v_i

Active Contour Algorithm

Input: Greyscale image E(x,y); Output: Binary contour around object in image

Initialize n-point contour around object (e.g., large box, not necessarily bounding): v_1 ,…, v_n

Initialize α, β, γ to 1 for all points v_i # index arithmetic is modulo n

While termination condition # e.g., bound on number of moved points

For i = 0 to n: # loop to move contour points to new locations

E_{min} = LARGE

For j=0 to 7 # 8 neighborhood of point v_i

$E_j = \alpha_i E_{continuity, j} + \beta_i E_{curvature, j} + \gamma_i E_{image, j}$ # Evaluate energy at each neighborhood point

If $E_j < E_{min}$ Then

$E_{min} = E_j$

jmin = j

Move contour point v_i to location jmin

If jmin not current location, pointsmoved ++ # count how many points moved

Examples of Active Contour Algorithm

Iterations of Algorithm:

Image Credit of shamrock animation:
https://medpure.blogspot.com/2012/03/active-contour-in-c.html

Segmentation of grey and white Matter in MR scan of the brain:

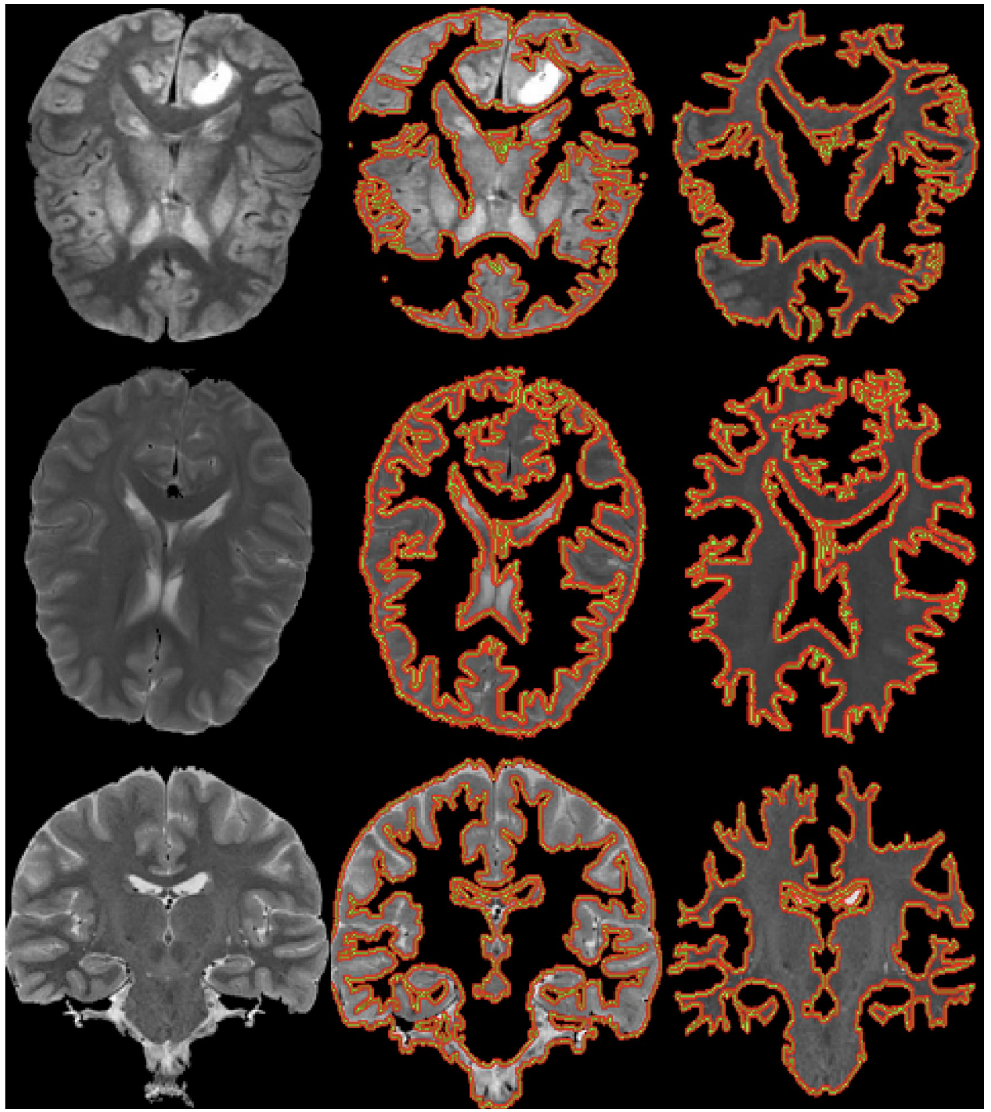


Image credit: Varma et al., 2011

Learning Outcomes

Explain the four causes of brightness changes in an image

Explain why the brightness gradient is perpendicular to an edge

Know masks used to compute edges

Evaluate edges in an image using approximations of 1^{st} or 2^{nd} derivatives, as well as the difference of Gaussians

Explain why smoothing is used in edge detection and provide a smoothing mask

Explain the Canny Edge Detection & Active Contour Algorithms

Define the energy function used in computing an active contour (3 terms involving edges, curvature, continuity)