Edge Detection

Definition:

Edges = brightness changes (discontinuities) in an image

due to

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

The Causes of Brightness Changes in an Image

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

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

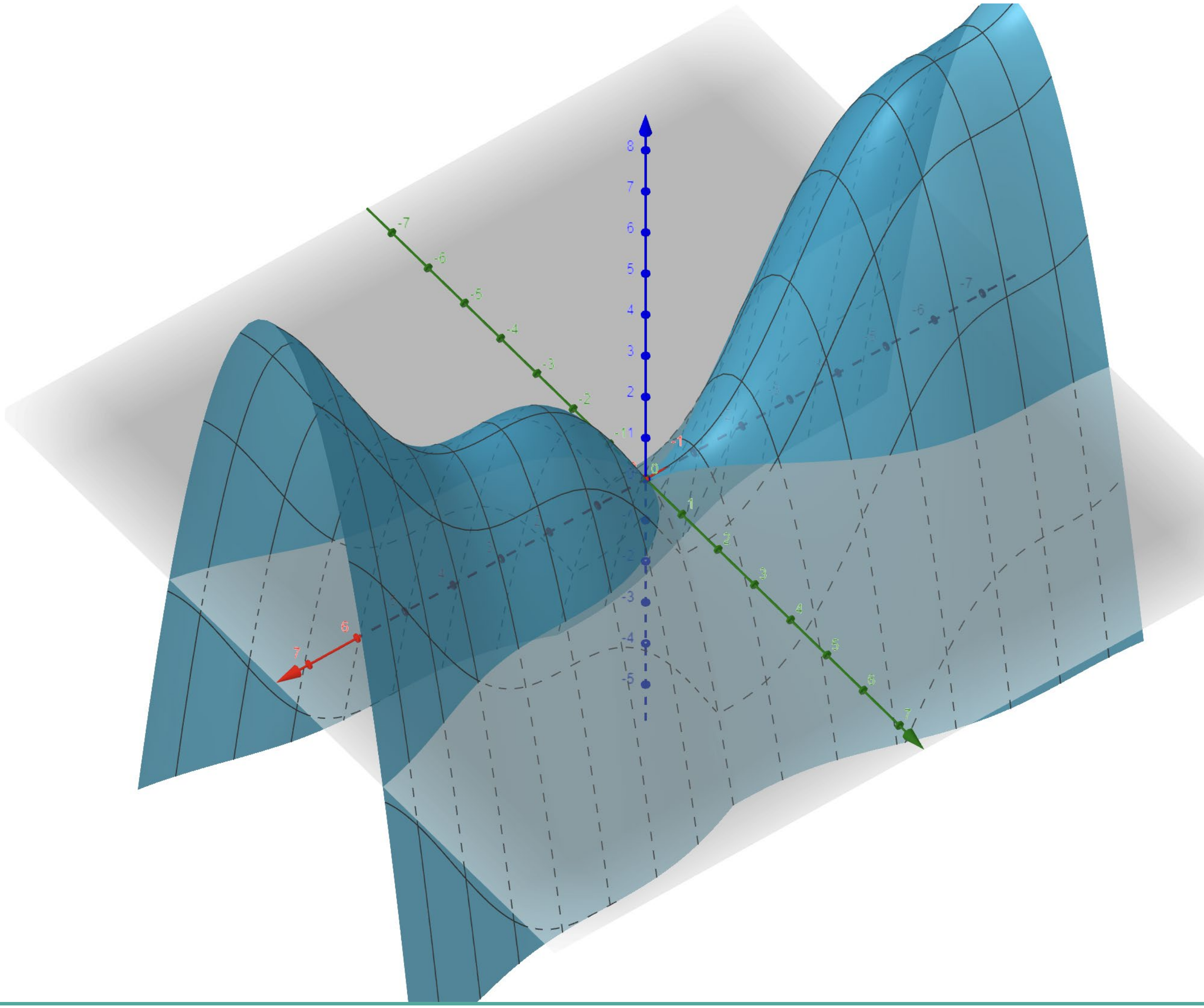
$E(x)$

$dE(x)/dx$

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

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+1,s) – E(r,s)

1
-1

E(r,s+1) – 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)

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

1	1
-1	-1

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

-1	1
-1	1

Simple Edge Detection Algorithm

- Compute $\partial E/\partial x$ and $\partial E/\partial y$ to determine brightness gradient direction.
- Use $|\partial E/\partial x| + |\partial E/\partial y|$ or $(\partial E/\partial x)^2 + (\partial E/\partial y)^2$ or magnitude sqrt$\{(\partial E/\partial x)^2 + (\partial E/\partial y)^2\}$ to compute "edge strength" M

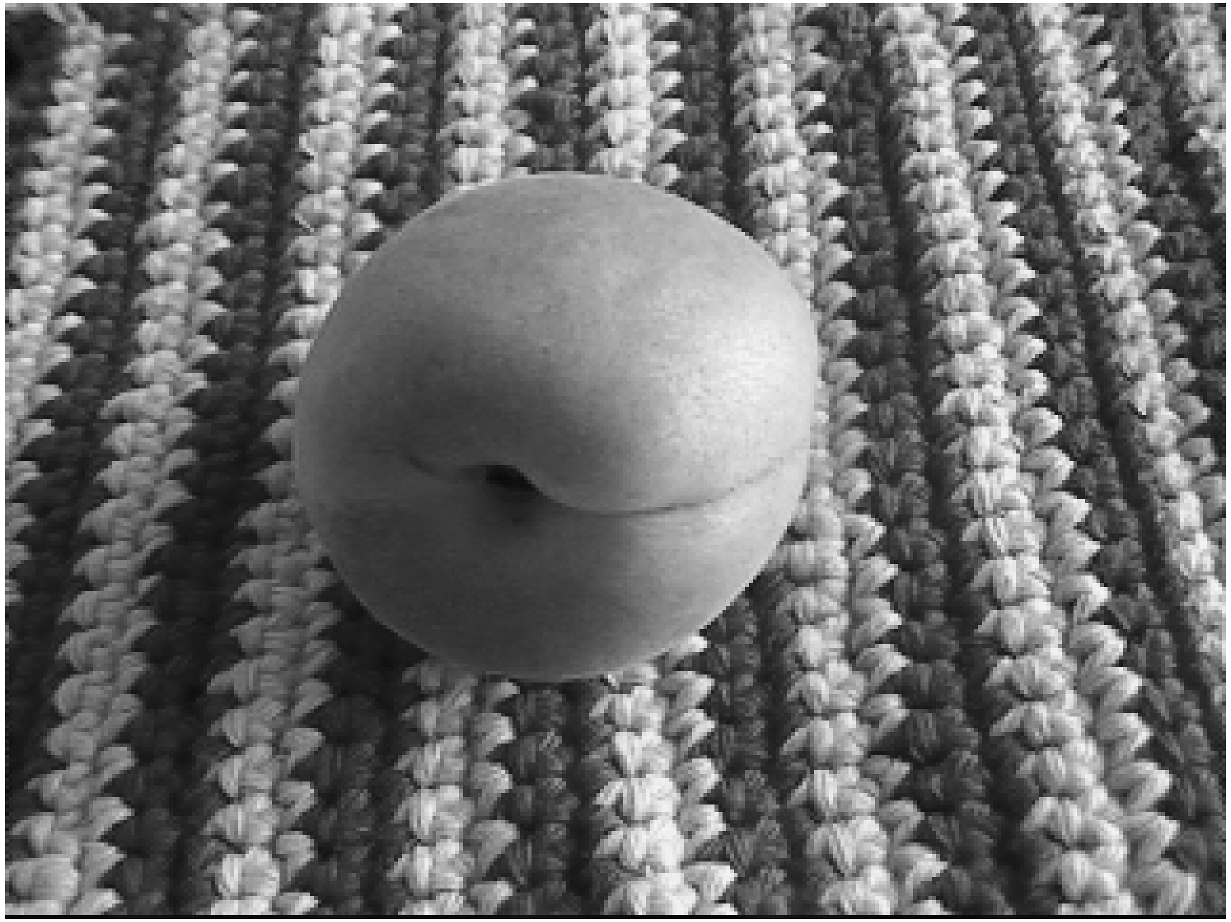
IF M > threshold,

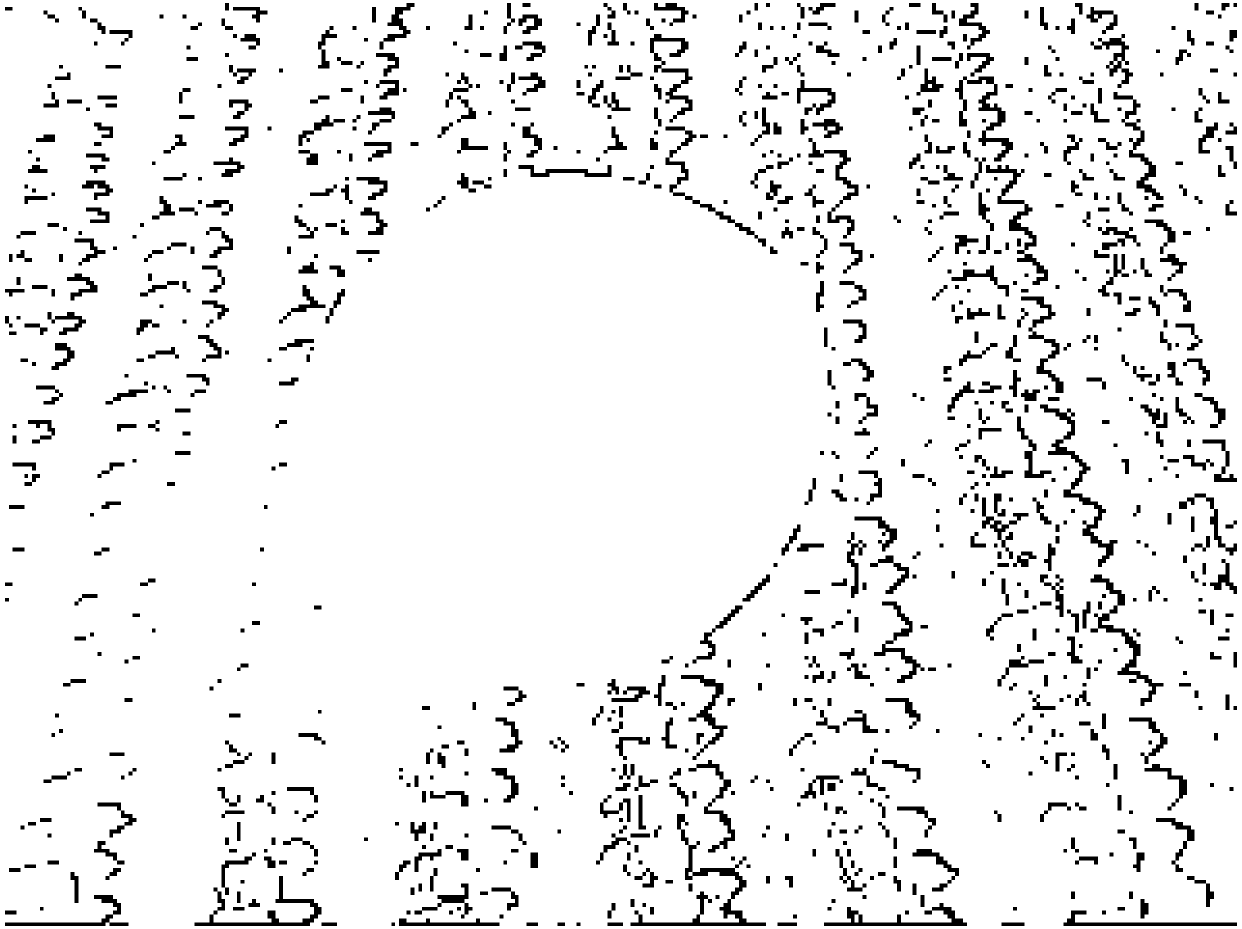
EdgeMap(x,y) = 1;

ELSE

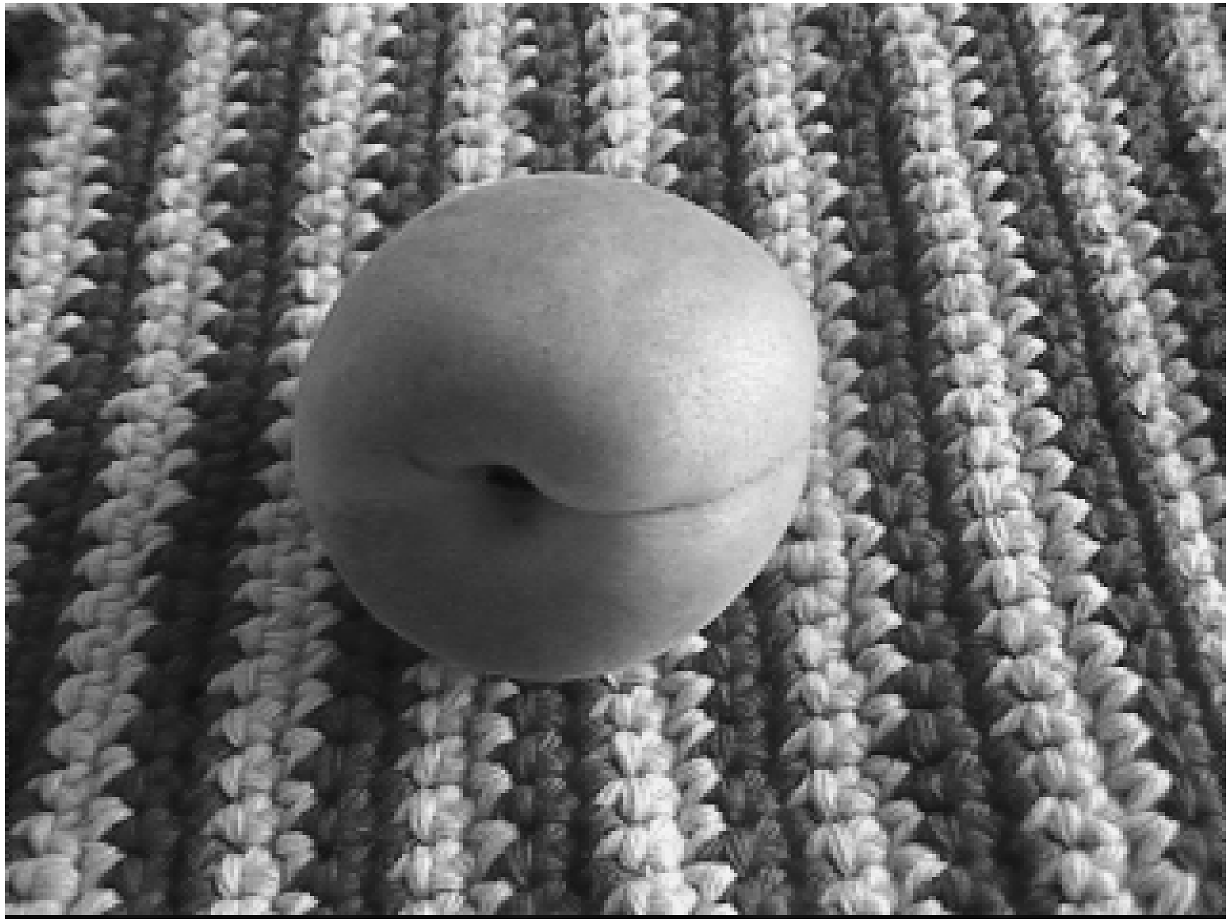
EdgeMap(x,y) = 0;

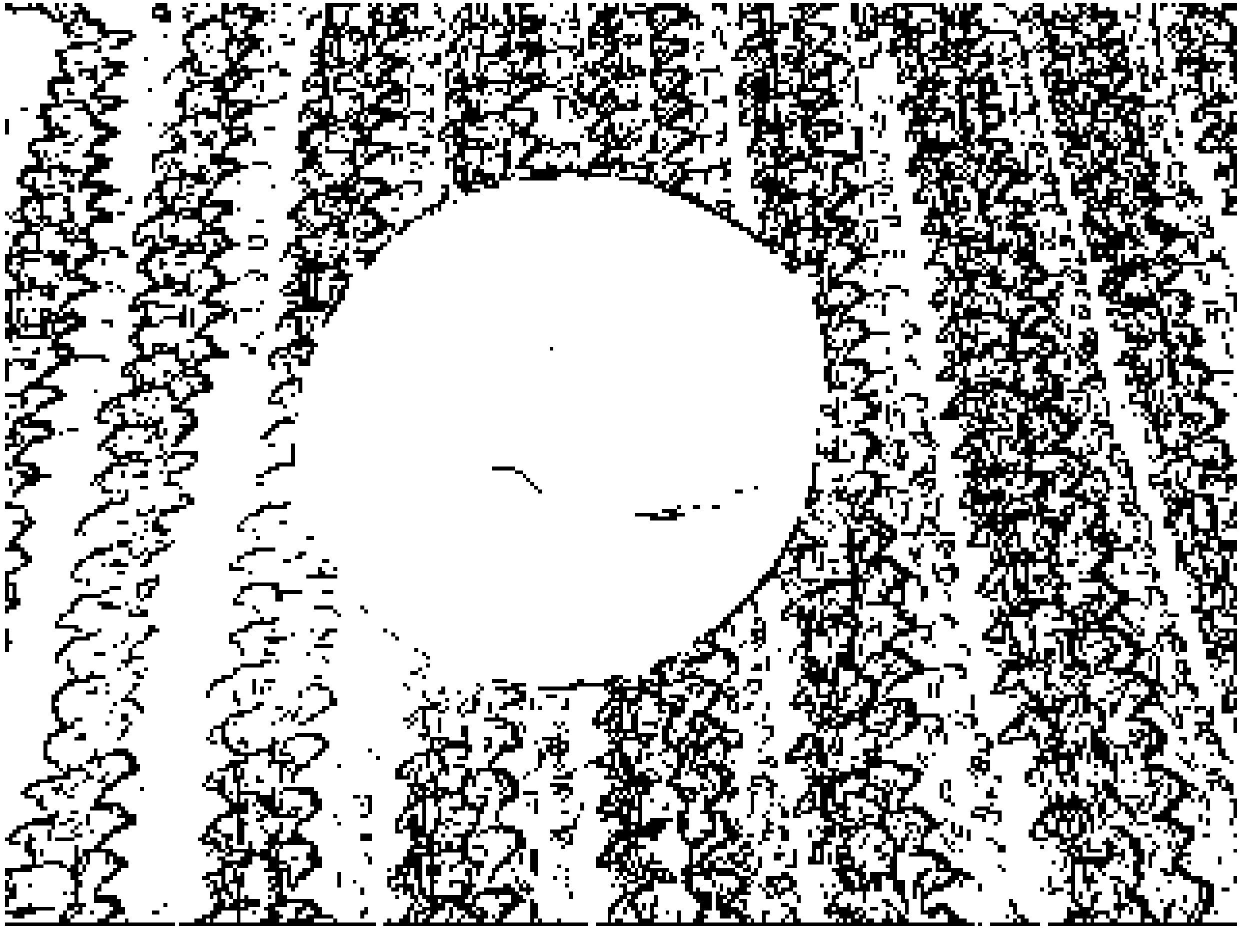
High Threshold on Magnitude of Brightness Change



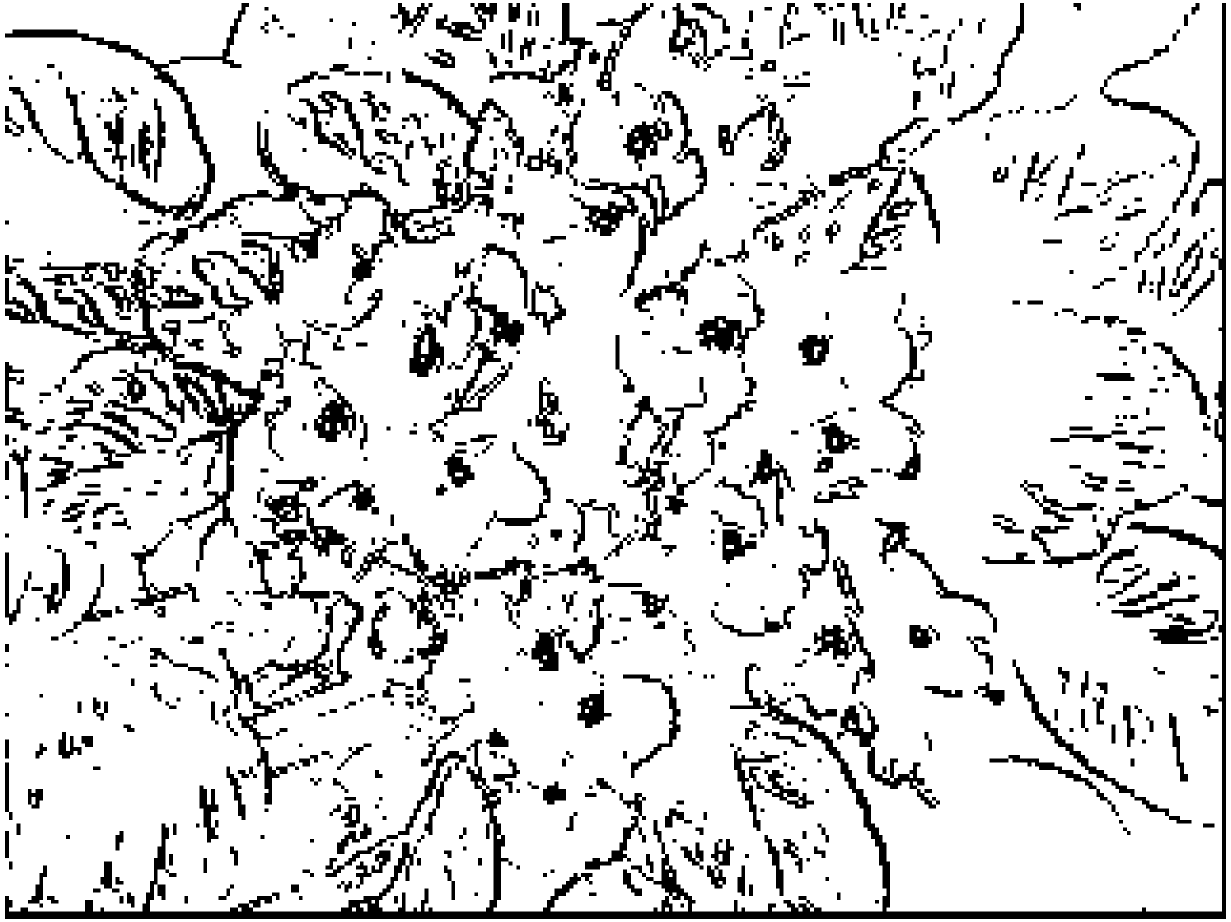


Low Threshold on Magnitude of Brightness Change

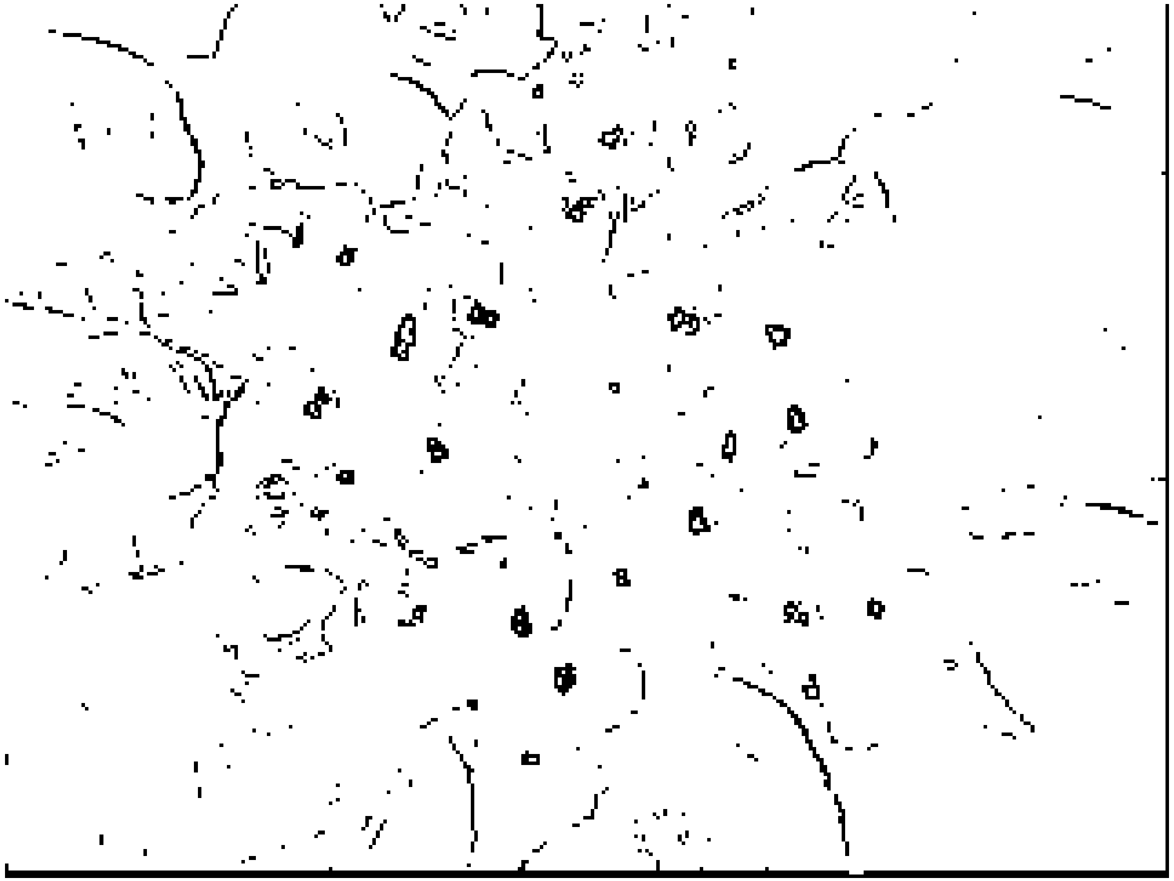




Low Threshold on Magnitude of Brightness Change



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:

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

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

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

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:

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

Mask:

	1	
1	-4	1
	1	

Edge Detection via Laplacian at Center Cell

Mask:

	1	
1	-4	1
	1	

Mask rotated by 45 degrees:

1		1
	-4	
1		1

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

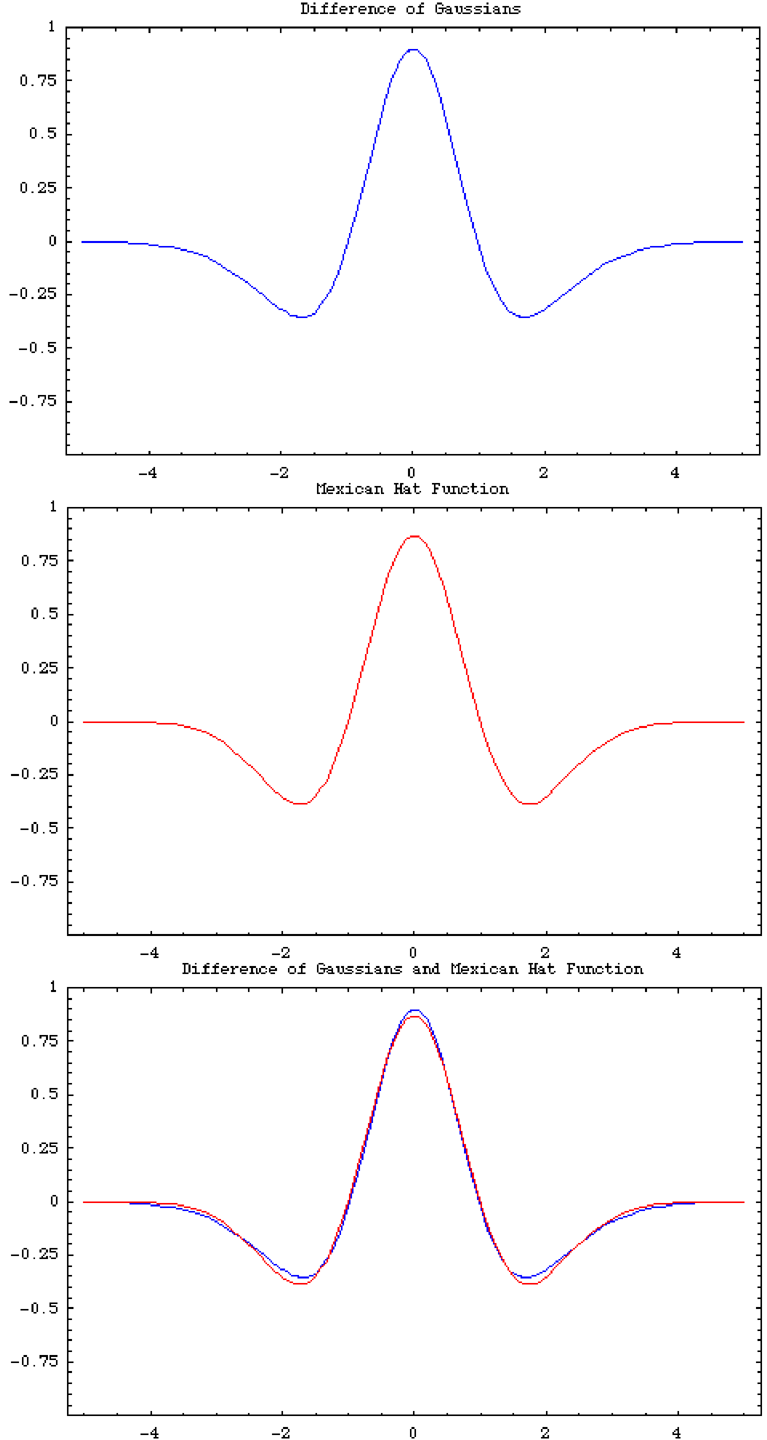
1	4	1
4	-20	4
1	4	1

Approximation by

Difference of Gaussians

Or

“Mexican Hat Function”



BOSTON UNIVERSITY

Source: Wikipedia

Image Smoothing

Use Image Masks multiple times:

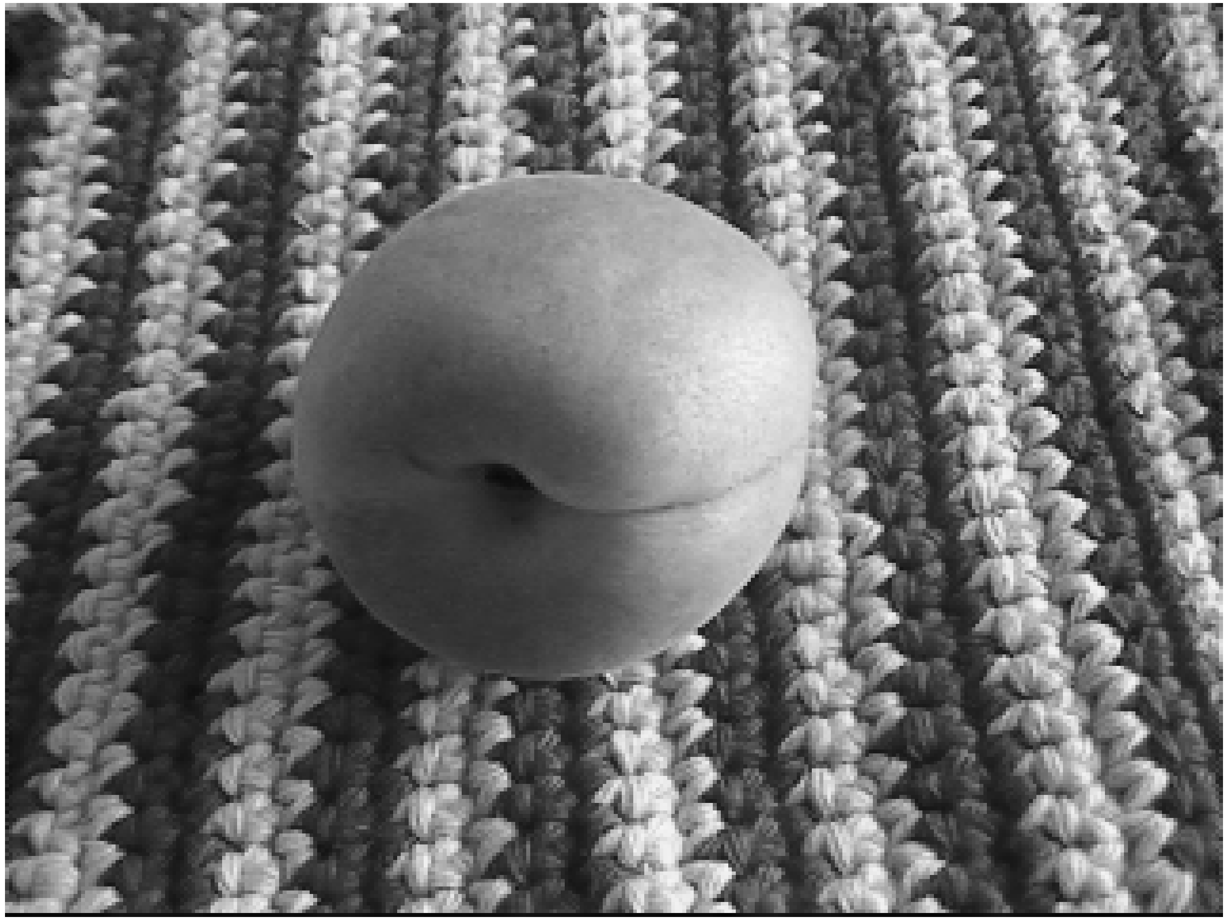
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \quad \text{or} \quad \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix} \quad \text{or} \quad \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

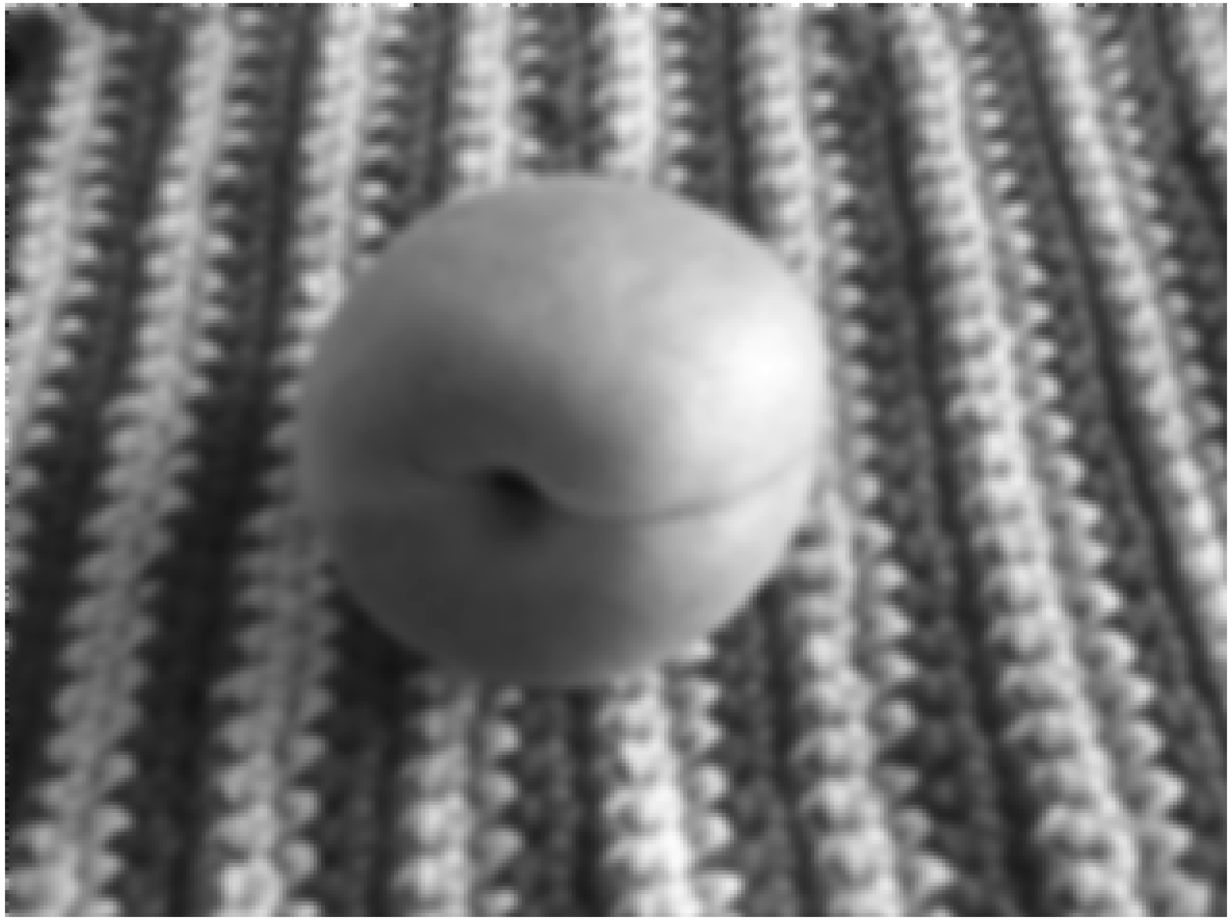
Or use a "Gaussian Mask," for example:

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

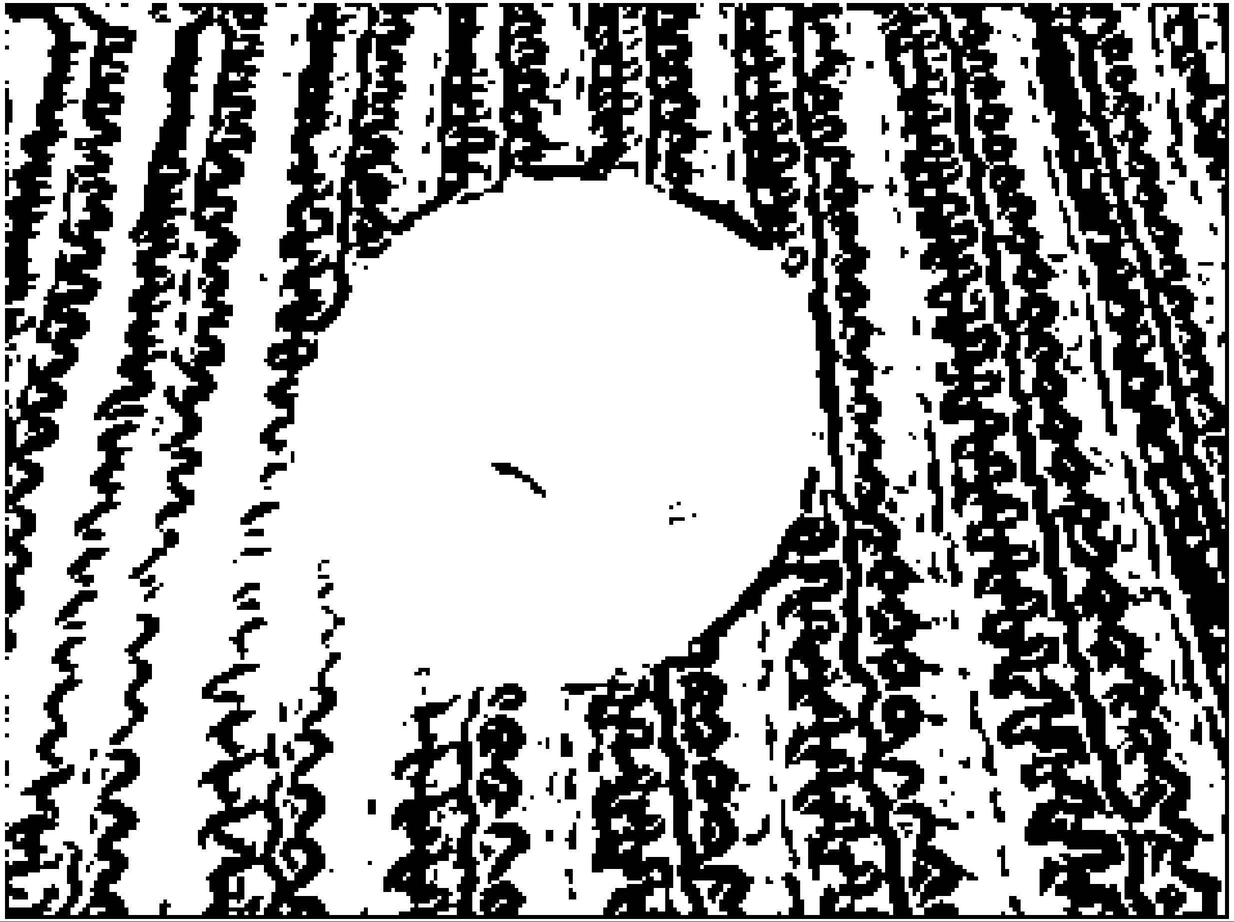
Smoothing Applied to our Example

Smoothing Applied to our Example

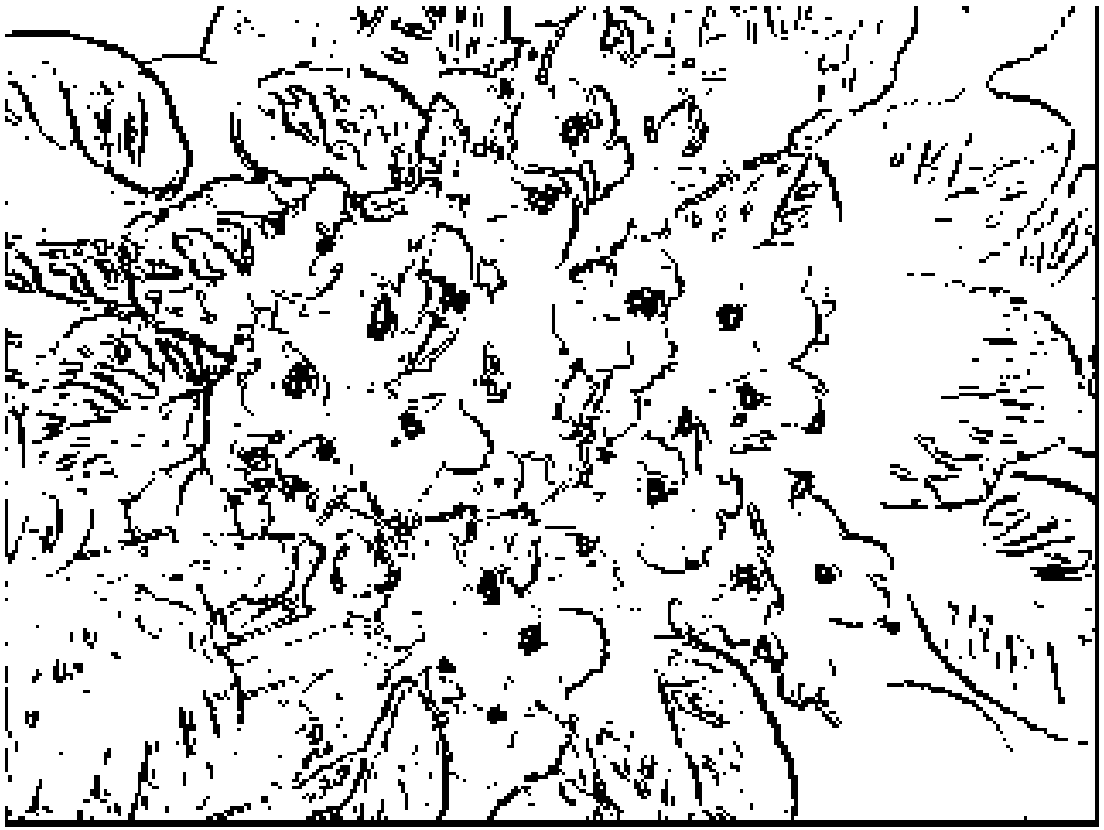




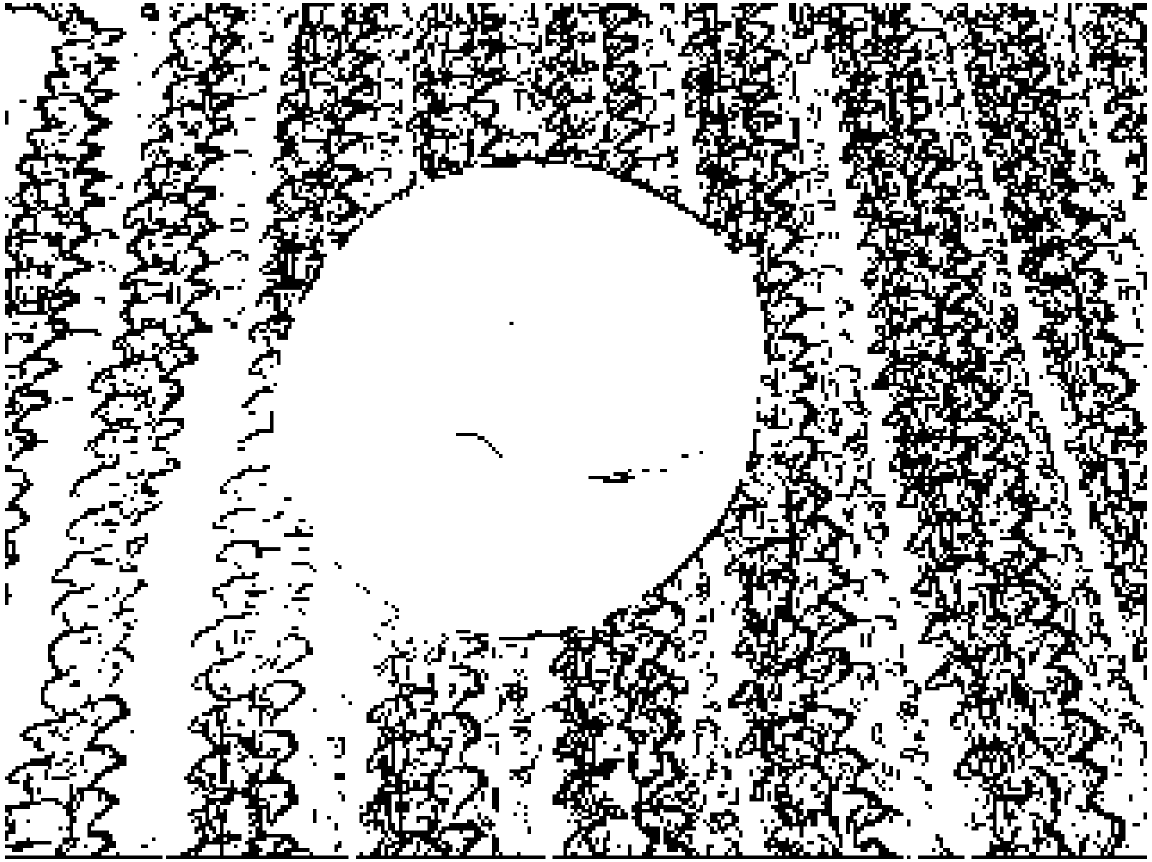
Edge Detection on Smoothed Images

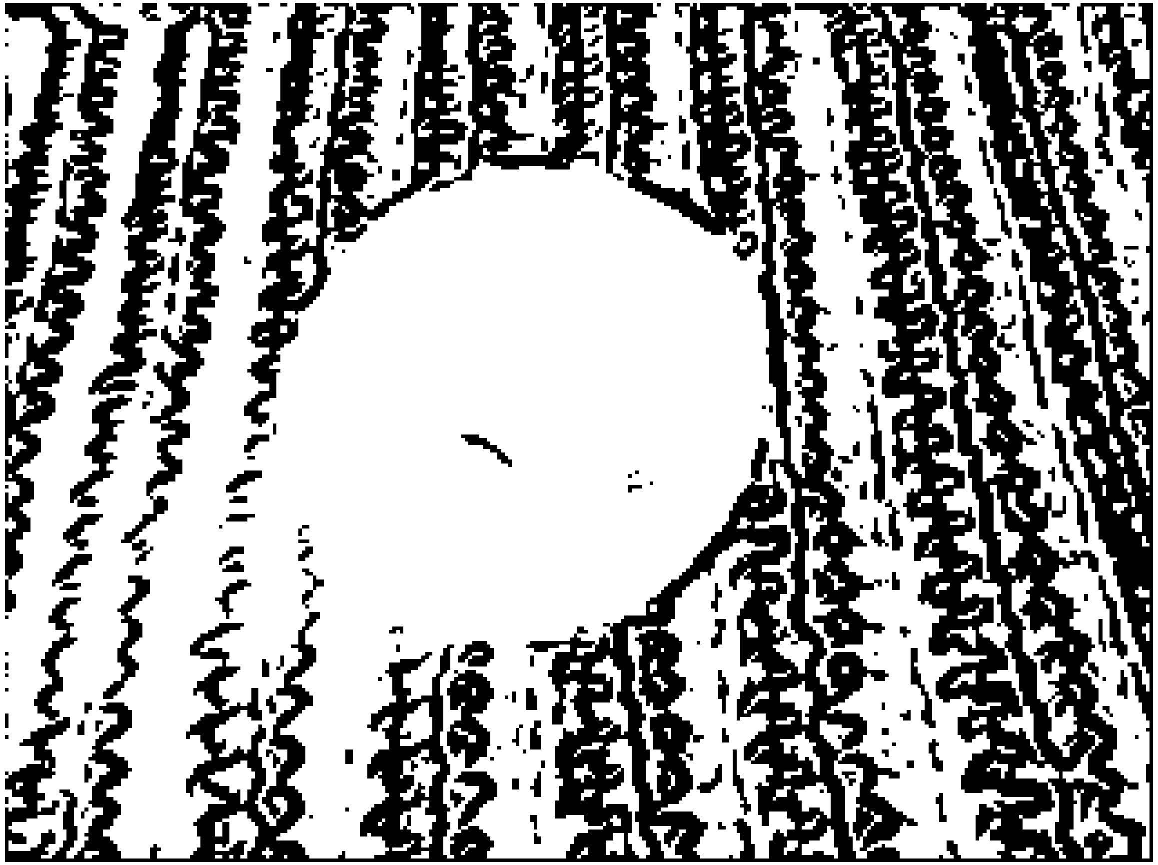


Edge Detection on Original and Smoothed



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

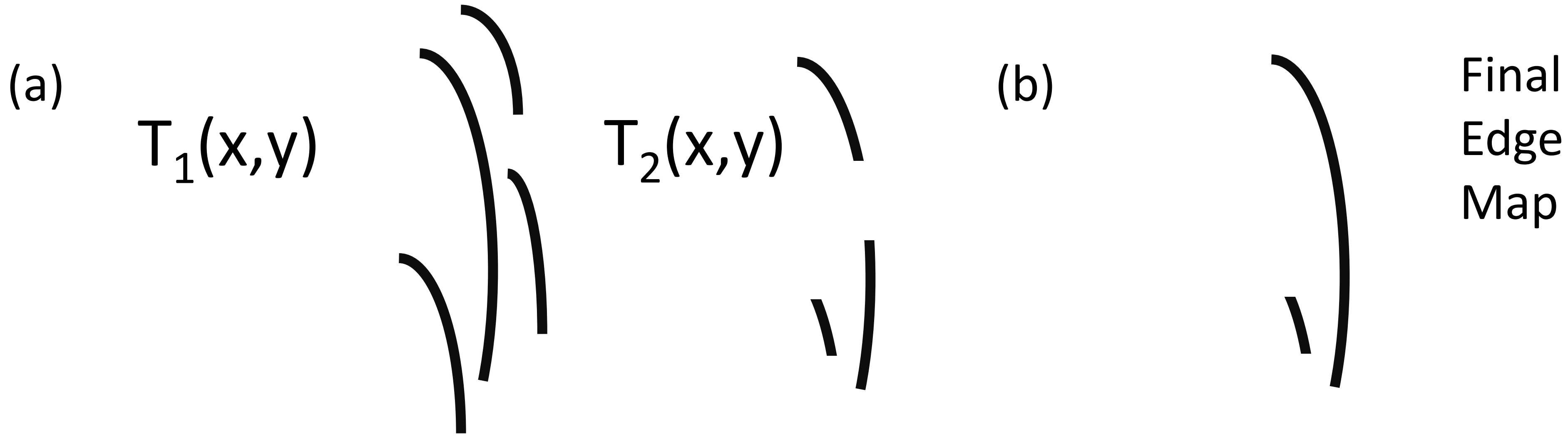
Canny Edge Detection

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 $M(B) \leq M(A)$ and $M(B) \leq M(C)$, 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)

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:

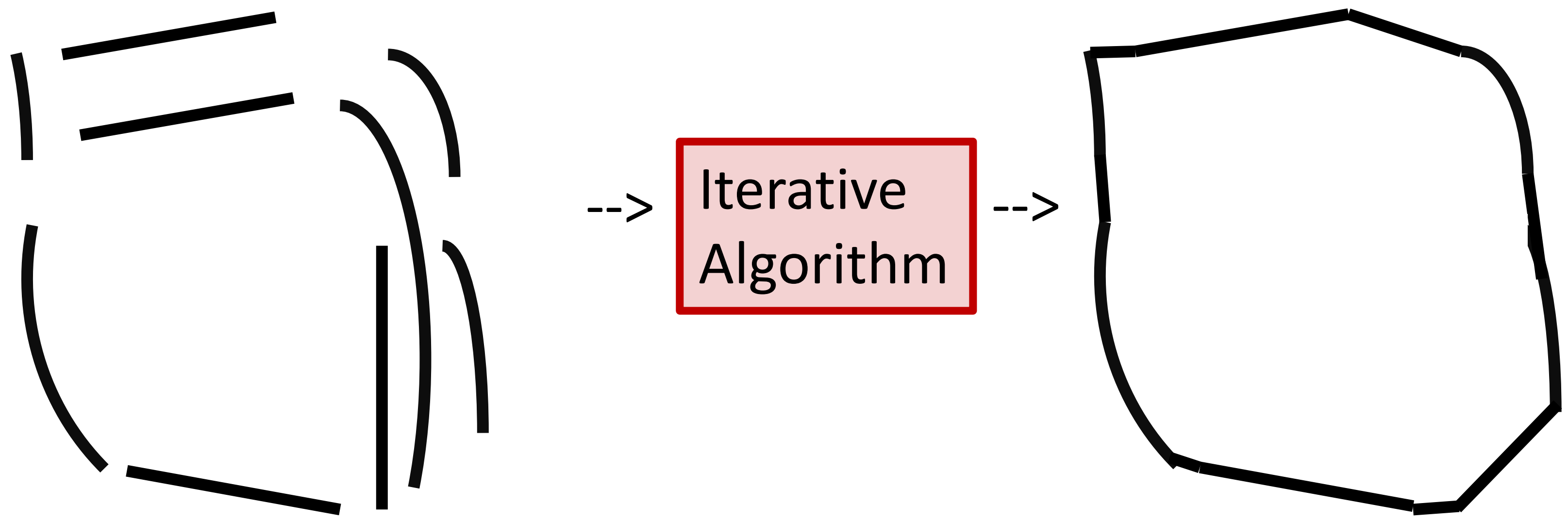


Active Contours (also called “Snakes”)

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



Iterative Optimization Algorithm

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

1. Initial Solution for Contour (e.g., hand drawn or large bounding box)
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

Continuous version on board

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

where

α_i , $\beta_{i,}$ γ_i control the relative influence of each term

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

Explanation of each energy follows

Read the paper by Williams and Shah on active contours