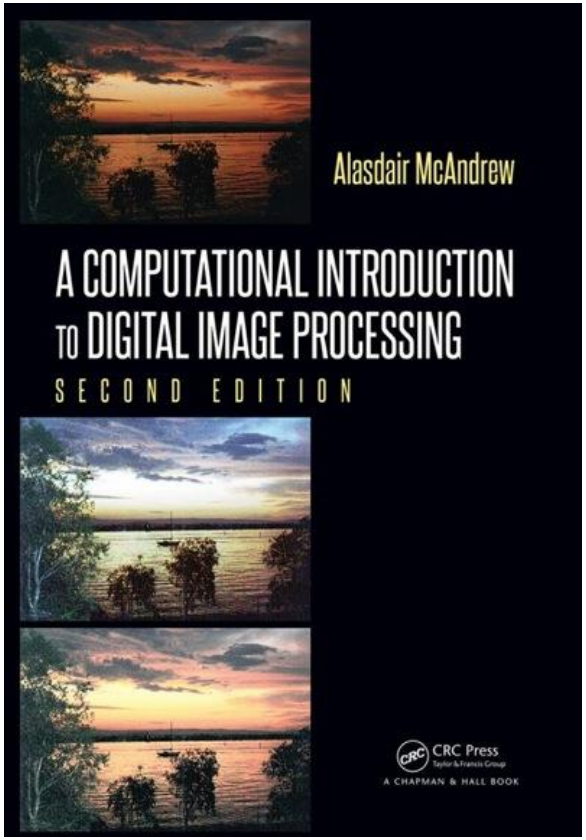
Chapter 9

MIDWESTERN STATE UNIVERSITY

Image Segmentation



Contents

MIDWESTERN STATE UNIVERSITY

- Introduction
- Thresholding
- Applications of Thresholding
- Choosing an Appropriate Threshold Value
- Adaptive Thresholding
- Edge
- Weiner Filtering

Contents

MIDWESTERN STATE UNIVERSITY

Introduction

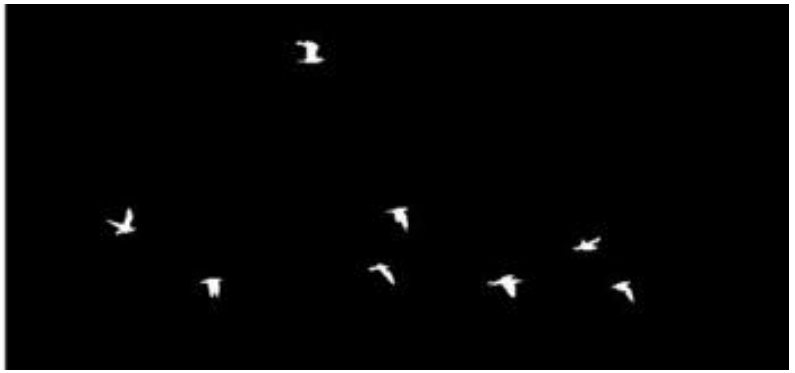
MIDWESTERN STATE UNIVERSITY

Segmentation: the operation of partitioning an image into component parts or into separate objects.

Thresholding: A vital part of segmentation where objects are isolated from the background

- Important component of robot vision

Figure 9.1: Thresholded image of flying birds



Thresholding

Thresholding can be done very simply in any of our systems. Suppose we have an 8-bit image, stored as the variable X. Then the command

X>T or X<T

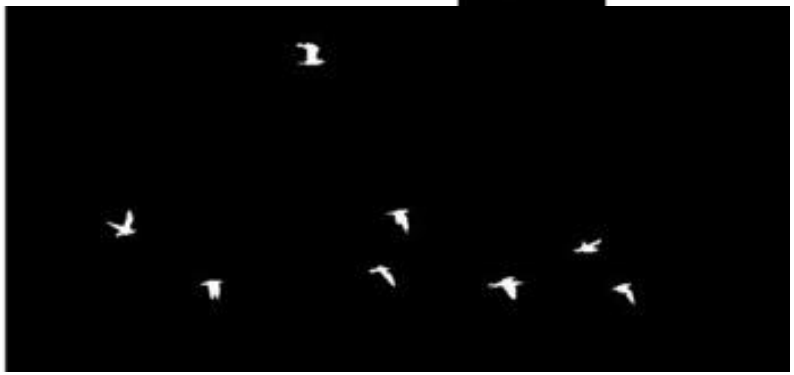
will perform the thresholding. For example, consider an image of some birds flying across a sky; it can be thresholded to show the birds alone in MATLAB or Octave:

```
>> f = imread('flying.png');
>> imshow(f); figure,imshow(f<50)
```

MATLAB/Octave

```
In :  f = io.imread('flying.png')
In :  io.imshow(f)
In :  fig = plt.figure(); fig.show(io.imshow(f<50))
```

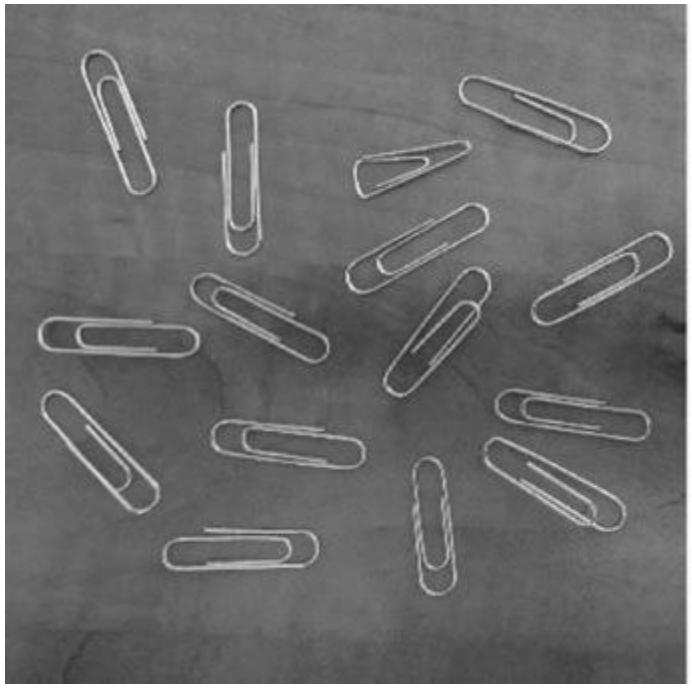
Python

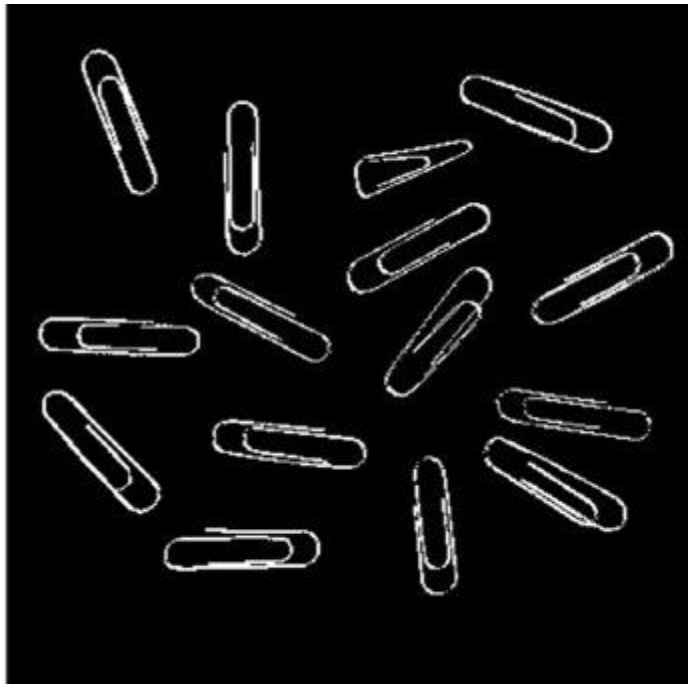


Thresholding

MIDWESTERN STATE UNIVERSITY

Figure 9.2: Thresholded image of paperclips





Thresholding

Thresholding also provides a way of showing hidden aspects of an image

- Ex. Handmade paper (Figure 9.3)

Figure 9.3: The paper image and result after thresholding

Double Thresholding

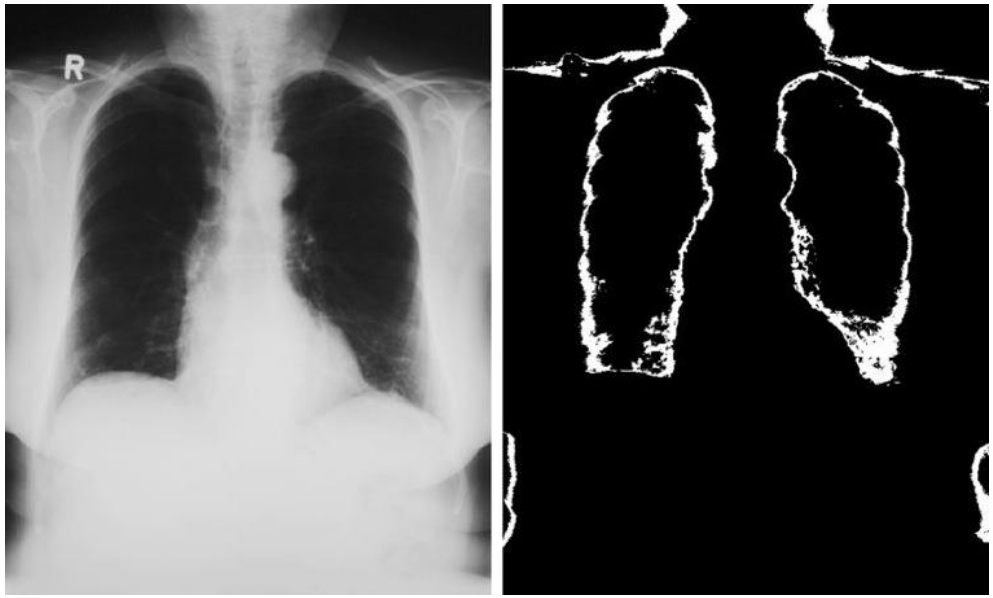
MIDWESTERN STATE UNIVERSITY

Choose two values T_1 and T_2 and apply a thresholding operation

A pixel becomes

- White: if gray level is between T_1 and T_2
- Black: gray level is otherwise

Figure 9.4: The image xray.png and the result after double thresholding



Double Thresholding

MIDWESTERN STATE UNIVERSITY

Try It:

MatLab/Octave:

```
x = imread('xray.png');
imshow(x);
dt = (x > 50 & x < 80);
imshow(dt)
```

Python:

```
import skimage.io as io
x = io.imread('xray.png')
dt = ((x > 50) & (x < 80))
imshow(dt)
```

Applications of Thresholding

MIDWESTERN STATE UNIVERSITY

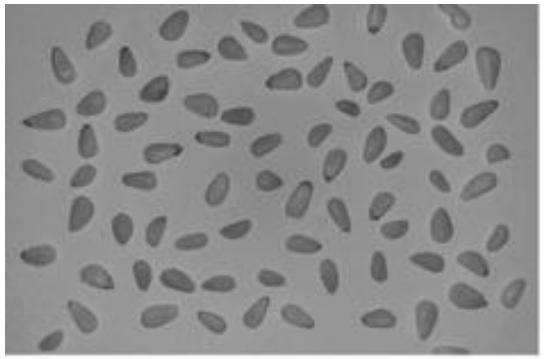
When can Thresholding be useful?

- Remove unnecessary detail
- Bring out hidden detail
- Remove a varying background

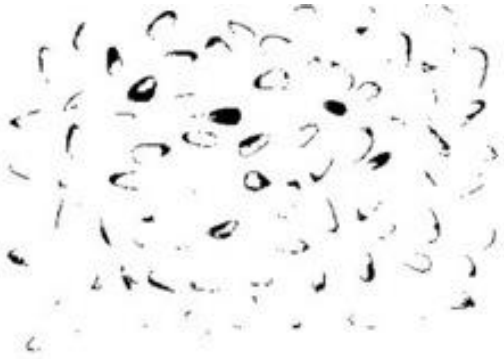
Choosing an Appropriate Threshold Value

MIDWESTERN STATE UNIVERSITY

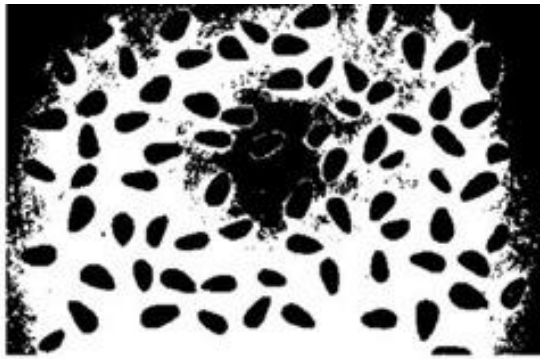
Figure 9.5: Attempts at thresholding



n: Original image



n1: Threshold too low

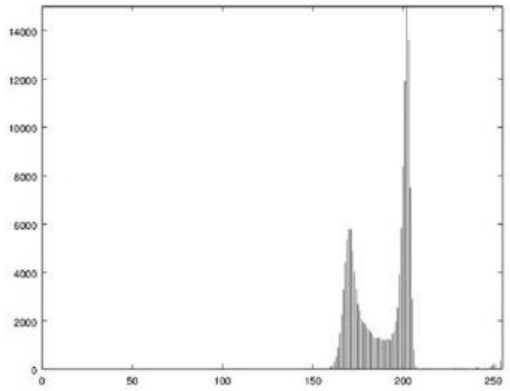


n2: Threshold too high

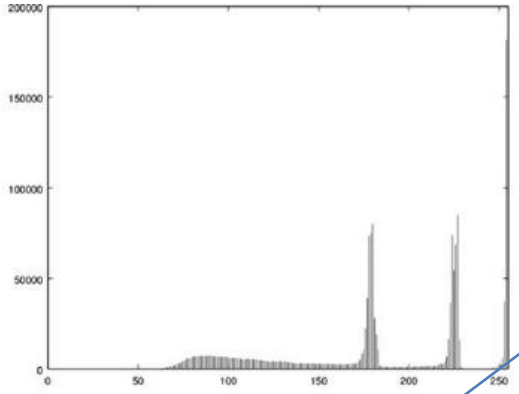
Choosing an Appropriate Threshold Value

MIDWESTERN STATE UNIVERSITY

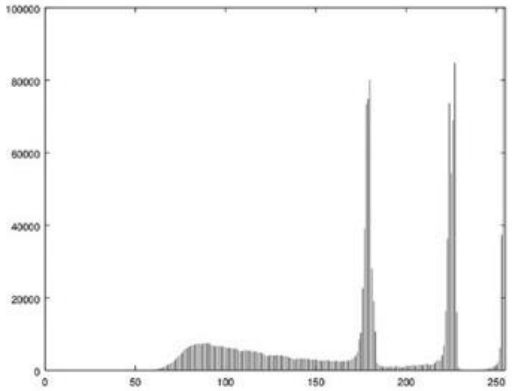
Figure 9.6: Histograms



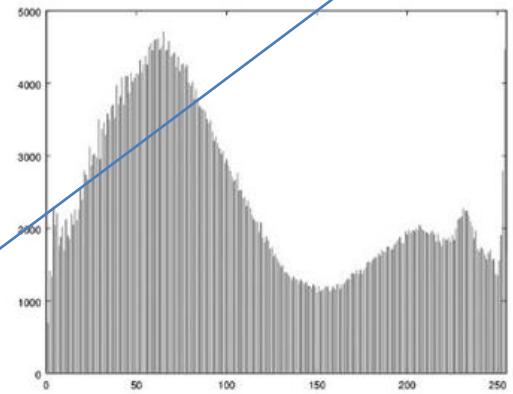
Blood smear image: `blood.png`



Paramecium image: `paramecium1.png`



Pine nuts image: `pinenuts.png`



Daisies image: `daisies.png`

Third peak at far right
Difficult to find appropriate split

Histogram is not the best method for finding a threshold value

Choosing an Appropriate Threshold Value

MIDWESTERN STATE UNIVERSITY

Two methods for choosing the best threshold:

- Otsu's Method
  - First described by Nobuyuki Otsu in 1979
  - Maximizes *inter-class variance*

- The ISODATA Method
  - Iterative Self-Organizing Data Analysis Technique A

Both methods perform statistical analyses on the pixel data to determine the best threshold value.

Choosing an Appropriate Threshold Value

MIDWESTERN STATE UNIVERSITY

MATLAB and Octave graythresh function:

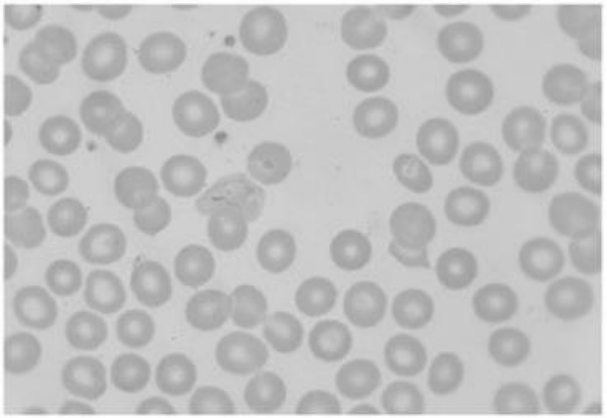
- Parameter ‘otsu’ for Otsu's method
- Parameter ‘intermeans’ for ISODATA method

Python filter module of skimage:

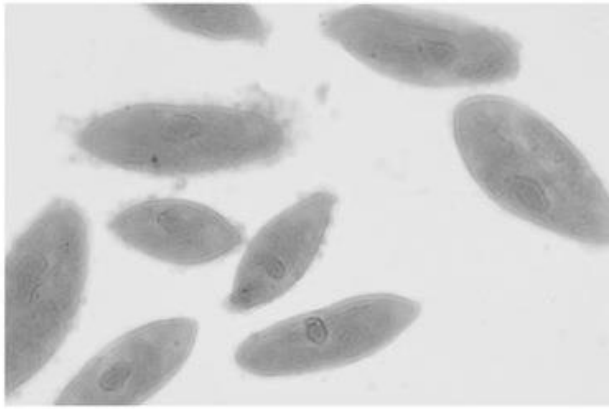
- Method threshold_otsu
- Method threshold_isodata

Otsu's Method

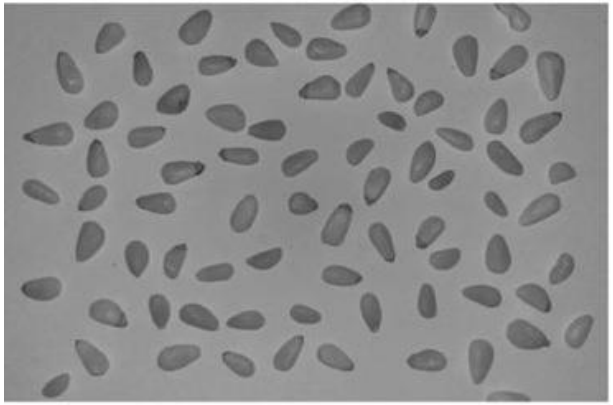
Figure 9.9: Images to be thresholded



blood.png



paramecium1.png

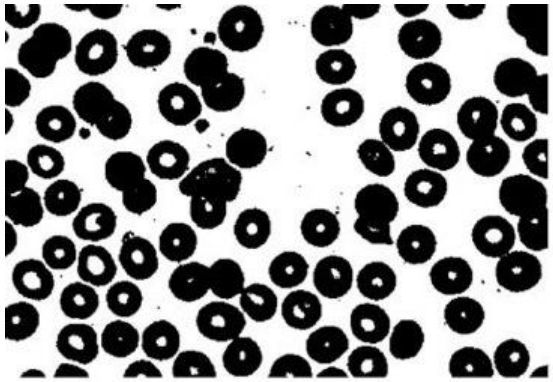


pinenuts.png

daisies.png

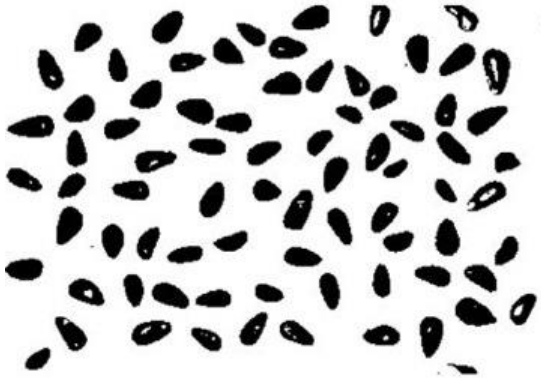
Otsu's Method

Figure 9.10: Thresholding with values obtained with Otsu's method

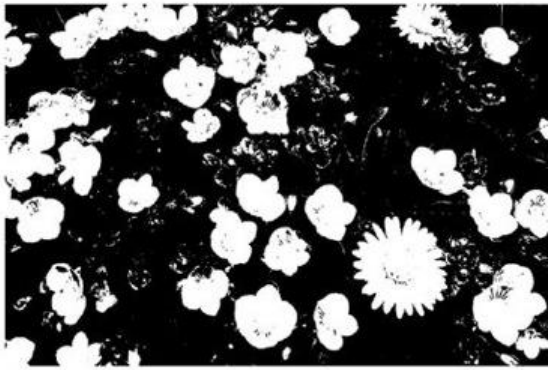


Blood

Paramecia



Pinenuts



Daisies

Otsu's Method – Try It

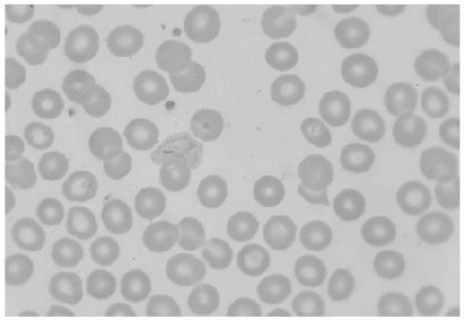
MIDWESTERN STATE UNIVERSITY

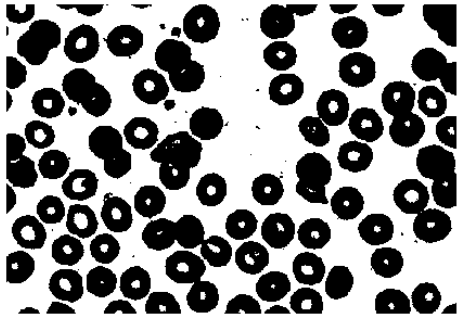
MatLab/Octave:

```
b = imread('blood.png');
imshow(b);
t = graythresh(b, 'otsu')
int_t = t * 255
bt = b > t_int;
imshow(bt);
```

Python:

```
import skimage.io as io
import skimage.filters as fl
b = io.imread('blood.png')
imshow(b)
t = fl.threshold_otsu(b)
bt = (b > t)
imshow(bt)
```



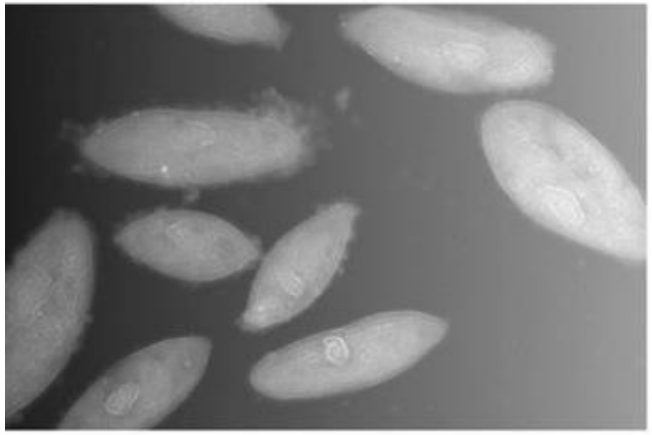


Adaptive Thresholding

Sometimes it is not possible to obtain a single threshold value that will isolate an object completely

- This may happen if both the object and its background vary.

Figure 9.11: An attempt at thresholding

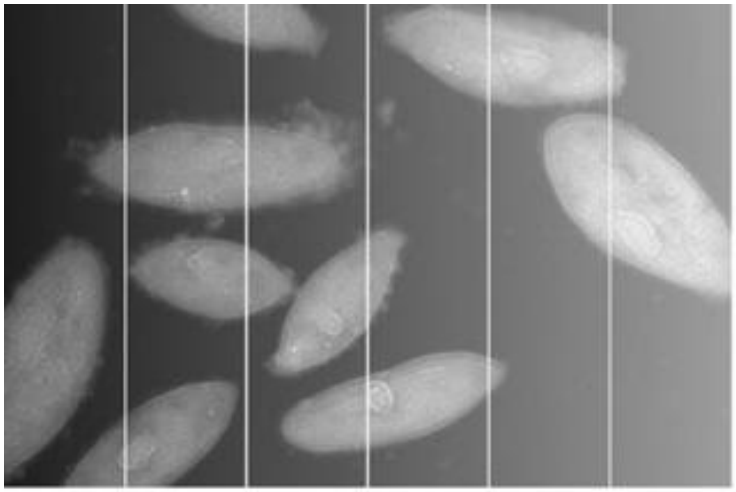


(a) Paramecium image: p2

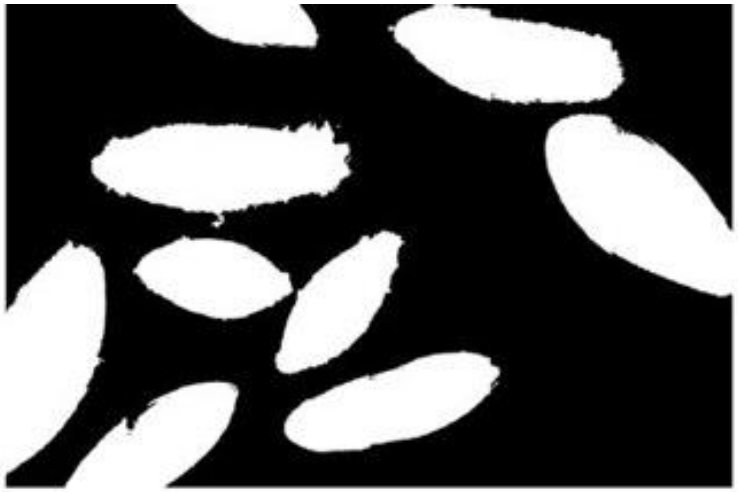
(b) Thresholding attempt

Adaptive Thresholding

Adaptive Thresholding: Cut the image into small pieces, and apply thresholding to each piece individually.



(a) Cutting up the image



(b) Thresholding each part separately

Edge Detection

MIDWESTERN STATE UNIVERSITY

- Edges contain some of the most useful information in an image
- Edges can be used to
  - Measure the size of objects in images
  - Isolate particular objects from their backgrounds
  - Reorganize or classify objects
- **Edge**: a local discontinuity in the pixel values which exceeds a given threshold (an observable difference in pixel values)

Figure 9.14: Blocks of pixels

51	52	53	59
54	52	53	62
50	52	53	68
55	52	53	55

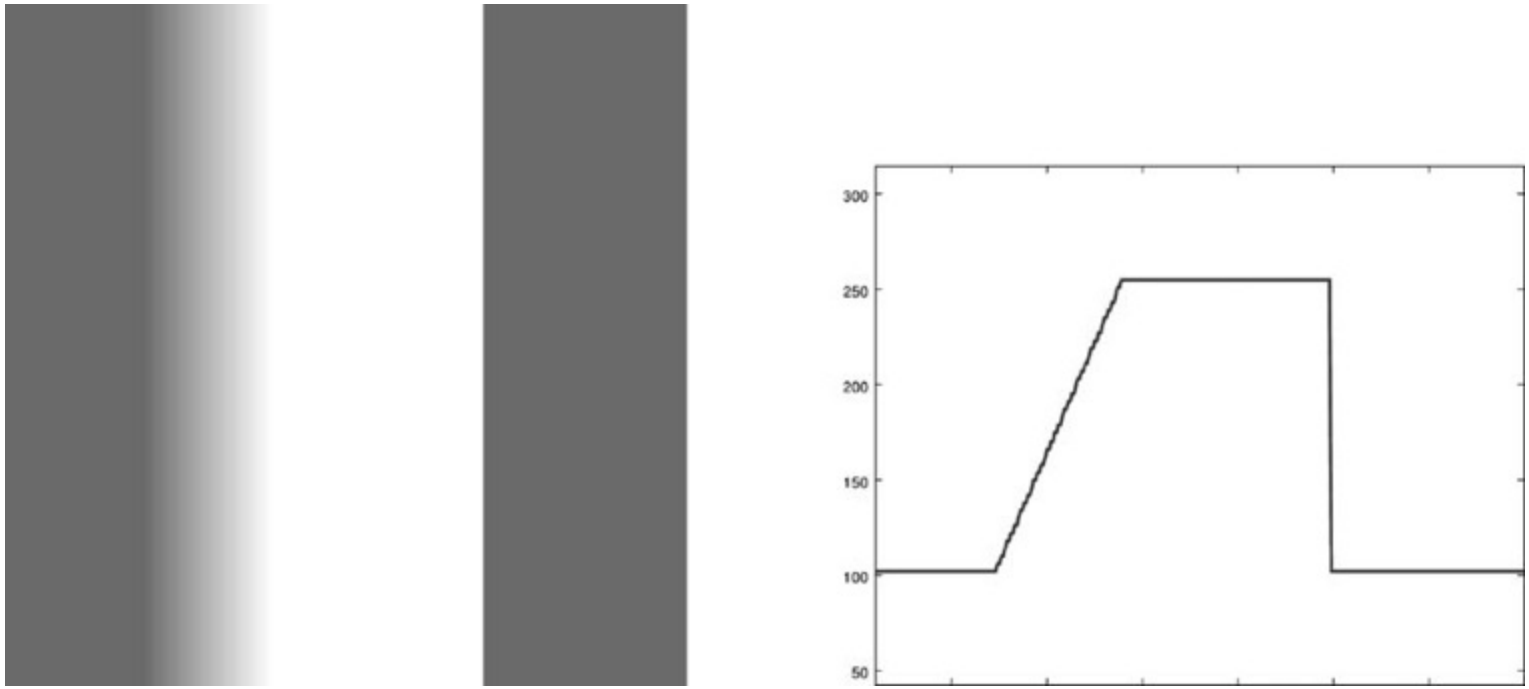
50	53	155	160
51	53	160	170
52	53	167	190
51	53	162	155

Derivatives and Edges

Fundamental Definitions

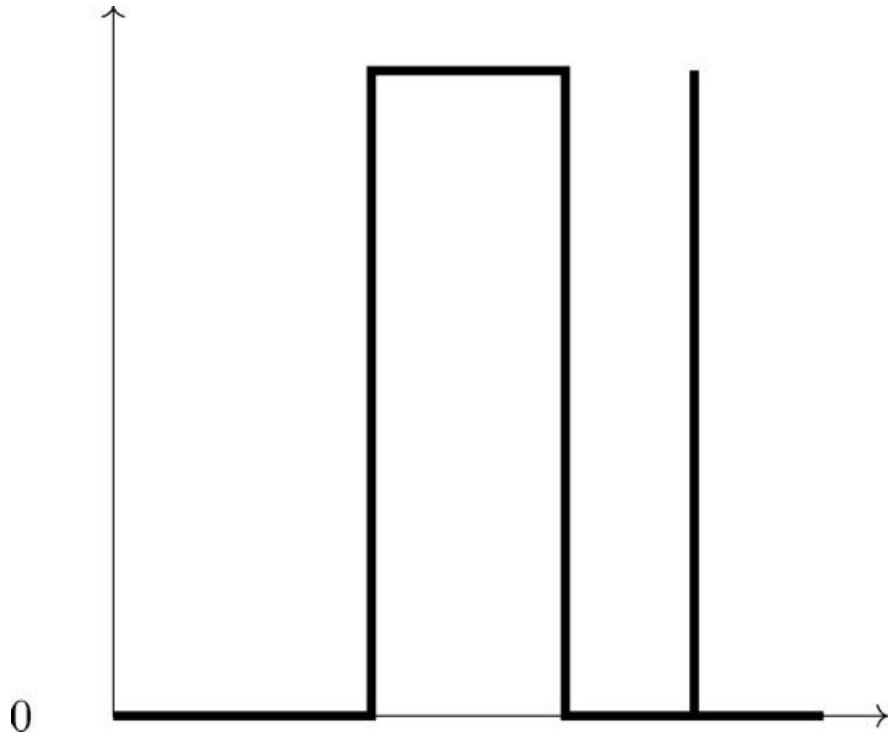
- **Ramp edge:** gray values change slowly
- **Step (Ideal) edge:** gray values change suddenly

Figure 9.15: Edges and their profiles



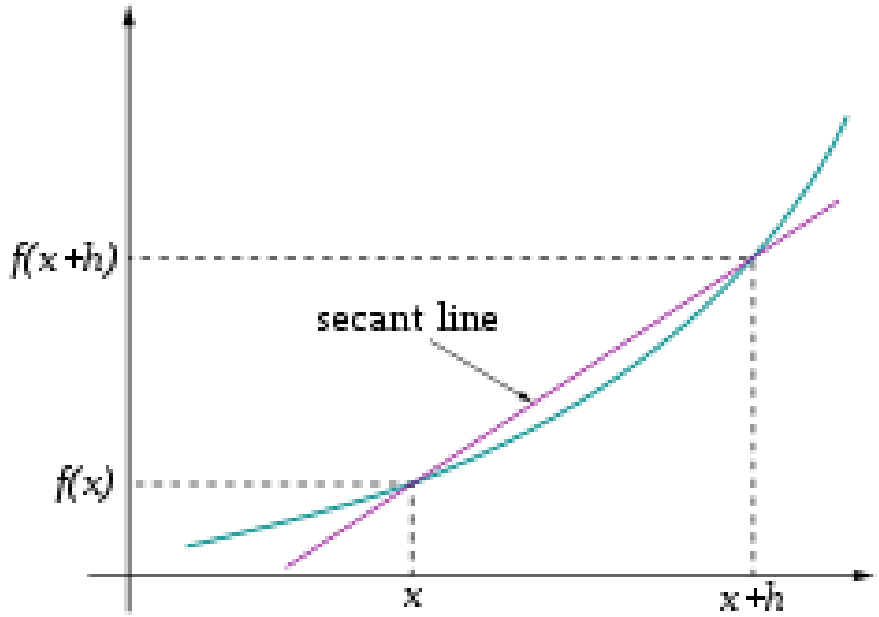
Derivatives and Edges

Figure 9.16: The derivative of the edge profile



The derivative, as expected, returns zero for all constant sections of the profile, and is non-zero (in this example) only in those parts of the image in which differences occur

Return of Calculus!!!!



$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

Return of Calculus!!!!

MIDWESTERN STATE UNIVERSITY

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

In an image, the smallest possible value of h is 1, the difference between the two adjacent pixels. A discrete version of the derivative expression:

$$f(x+1) - f(x).$$

Return of Calculus!!!!

MIDWESTERN STATE UNIVERSITY

Other expressions for the derivative:

$$\lim_{h \to 0} \frac{f(x) - f(x-h)}{h}, \qquad \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

Discrete counterparts:

$$f(x) - f(x-1), \qquad (f(x+1) - f(x-1))/2.$$

For an image, with two dimensions, we use partial derivatives; an important expression is the **gradient**:

$$\left[\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y} \right]$$

Edge Detection Filters

MIDWESTERN STATE UNIVERSITY

Using the expression f(x + 1) − f(x − 1) for the derivative, leaving the scaling factor out, produces horizontal and vertical filters:

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Will find vertical and horizontal edges but produce “jerky” edges; use smoothing in the opposite direction

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Combined into one filter each, known as the **Prewitt Filters**:

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

Vertical edges

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

Horizontal edges

Prewitt Filters

MIDWESTERN STATE UNIVERSITY

Figure 9.17: A set of steps: A test image for edge detection

Prewitt Method – Try It

MIDWESTERN STATE UNIVERSITY

MatLab/Octave:

```
s = imread('stairs.png');
imshow(s);
px = [-1 0 1; -1 0 1; -1 0 1];
py = px';
sx = imfilter(s, px);
imshow(sx);
sy = imfilter(s, py);
imshow(sy);
```

Python: ????

```
import skimage.io as io
import skimage.filters as fl
s = io.imread('stairs.png')
imshow(s)
sx = fl.prewitt_h(s)
io.imshow(sx)
sy = fl.prewitt_v(s)
io.imshow(sy)
```

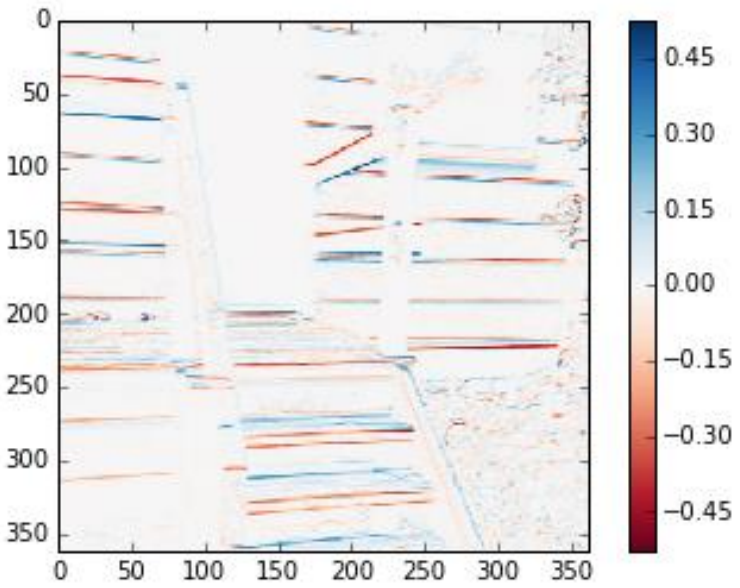
Prewitt Filters

MIDWESTERN STATE UNIVERSITY

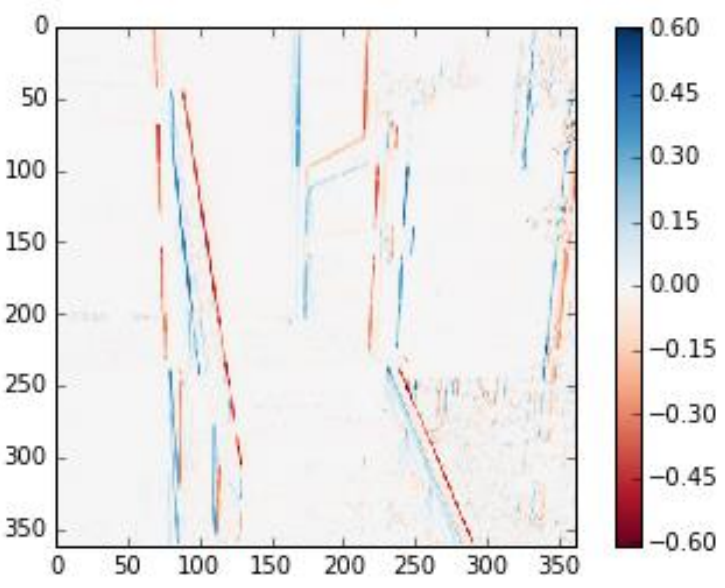
MatLab/Octave:

Prewitt Filters

Python:



sx



sy

Prewitt Filters

MIDWESTERN STATE UNIVERSITY

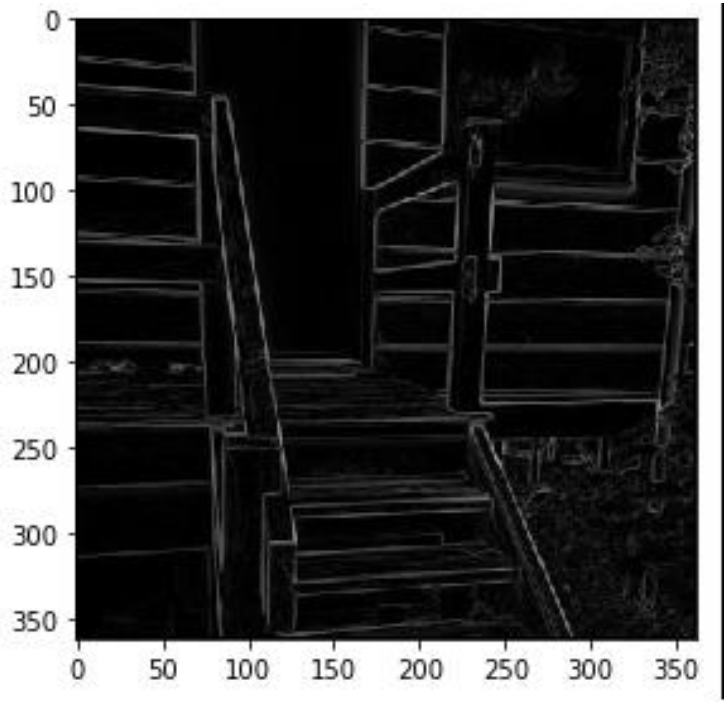
MatLab/Octave All edges:

```
edge_p = edge(s, 'prewitt');
imshow(edge_p);
```

Performs edge detection
and thresholding

Python:

```
import numpy as np
edge_p = np.sqrt(sx**2 + sy**2)
io.imshow(edge_p);
```



Roberts and Sobel Filters

Figure 9.21: Results of the Roberts and Sobel filters

(a) Roberts edge detection

(b) Sobel edge detection

Roberts and Sobel Filters

Figure 9.22: Results of the Roberts and Sobel filters in Python

(a) Roberts edge detection

(b) Sobel edge detection

Laplacian Filter

MIDWESTERN STATE UNIVERSITY

Laplacian

- The sum of second derivatives in both directions
- Isotropic filter: invariant under rotation
- Extremely sensitive to noise

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

Discrete Laplacian Filter

Figure 9.24: Result after filtering with a discrete Laplacian

Laplacian Filter

MIDWESTERN STATE UNIVERSITY

Laplacian

- The sum of second derivatives in both directions
- Isotropic filter: invariant under rotation
- Extremely sensitive to noise

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

Discrete Laplacian Filter

Figure 9.24: Result after filtering with a discrete Laplacian

Laplacian Filter

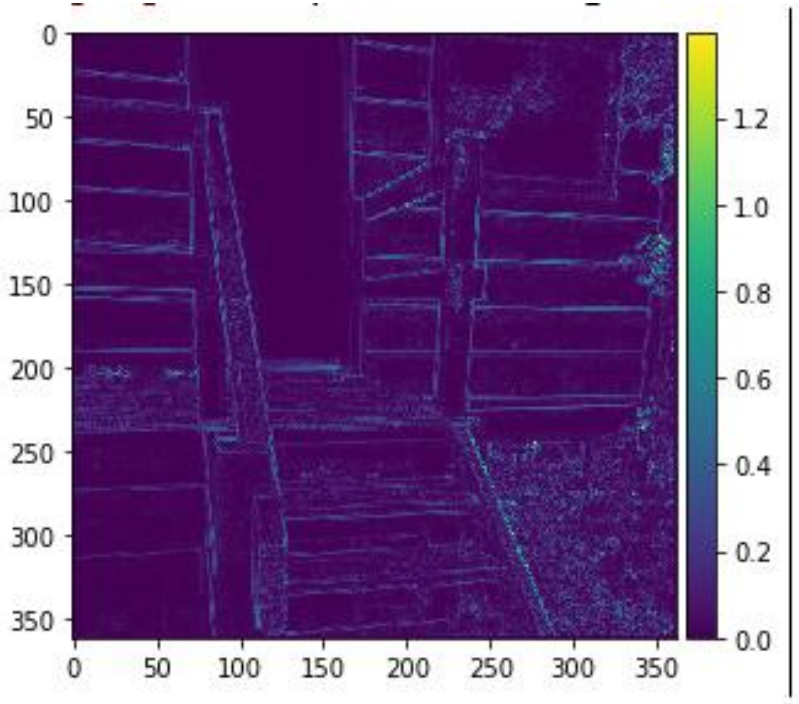
MIDWESTERN STATE UNIVERSITY

MatLab/Octave:

```
lap = fspecial('laplacian', 0)
s_lap = imfilter(s, lap);
imshow(s_lap);
```

Python: ????

```
import skimage.util as ut
s2 = ut.img_as_float(s)imshow(s_lap)
s_lap = abs(fl.laplace(s2))
io.imshow(s_lap)
```



The Canny Edge Detector

MIDWESTERN STATE UNIVERSITY

Canny designed his method to meet 3 criteria for edge detection:

- **Low error rate of detection.** (It should find ALL edges, and only edges)
- **Localization of edges.** (Distance between actual edges in the image and edges found by this algorithm should be minimized)
- **Single response.** (Algorithm should not return multiple edge pixels when only a single edge exists)

The Canny Edge Detector

MIDWESTERN STATE UNIVERSITY

MatLab/Octave:

```
sc = edge(s, 'canny');
imshow(sc)
```

Python:

```
import skimage.feature as ft
sc = ft.canny(s)
io.imshow(sc)
```