Assignment 7
Signal & Image Processing

Casper B. Hansen
University of Copenhagen
Department of Computer Science
fvx507@alumni.ku.dk

October 23, 2014

Abstract

In this assignment we discuss the principles behind mathematical morphology, which is closely related to set theory as its operations can be described by means of this branch of mathematics. Some of these will be described as set operations.

In our discussion we will perform a few trials and experiments, and show some techniques like background normalization, recognition and extraction.

Contents

1 Erosion and Dilation as Set Theory

The dilation ($\oplus$) can be described in set theory as the union of all points $b \in B$ over each point in A, whilst erosion can be described by the intersection of all points $b \in B$ over each point in the complement of A, or $\bar{A}$.

$$A \oplus B = \bigcup_{b \in B} A_b \qquad A \ominus B = \bigcap_{b \in B} \bar{A}_b \tag{1}$$

2 Sketch Outcomes

By applying the *open* ($\circ$) and *close* ($\bullet$) operations, given in equation (2) below, to the provided image data, we get two smaller objects for the *opened* image, and a larger one for the *closed* one.

$$A \circ B = (A \ominus B) \oplus B \qquad A \bullet B = (A \oplus B) \ominus B \tag{2}$$

What is happening is that the *open* operation is first eroding away outer details, followed by a dilation which closes the gaps left behind. For the *closing* operation we first dilate the image, causing an expansion of the object, followed by an erosion which then strips away excess outer detail.

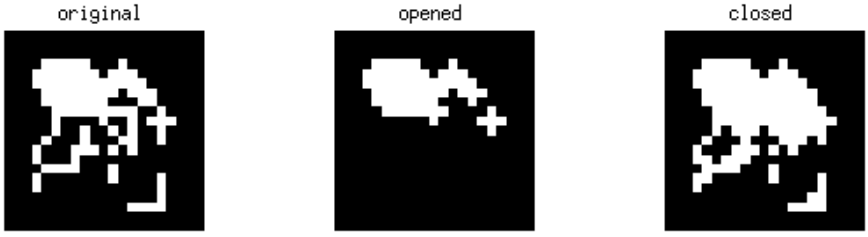

Figure 1: Original (left), opened (middle) and closed (right)

Had we used a diamond structural element of radius 2 we would get a single object in both cases. This is due to the fact that the diamond would be able to reach across the inner gaps of the original objects and that the little blob in the lower right area is too far away for it to reach across.

3 Experimentation

In each application of the three operations I've used the same three structural elements, the parameters of which are given in the table.

Name	Shape	Parameters
line	`'line'`	(50,1)
disk	`'disk'`	3
square	`'square'`	6

Figure 2: Structural elements used

$$\text{TopHat}_B(A) = A - ((A \ominus B) \oplus B) \tag{3}$$
$$\text{BotHat}_B(A) = ((A \oplus B) \ominus B) - A \tag{4}$$

3.1 Hit-and-Miss

On the left of figure 3 below, we see that the line does indeed find the lines as expected, in the middle we get anything which fits the disc shape of radius 3, and lastly so is the case for the larger square, which finds less objects, but still some.

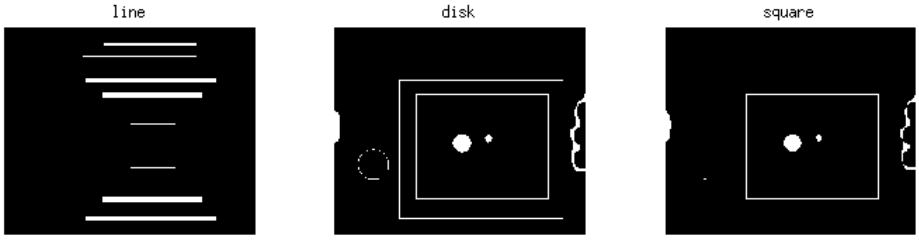

Figure 3: Hit-and-miss experiments

3.2 Top hat

The top hat operation (3) subtracts the detected objects of the opening from the original image, the effects of which can be seen in figure 4 below.

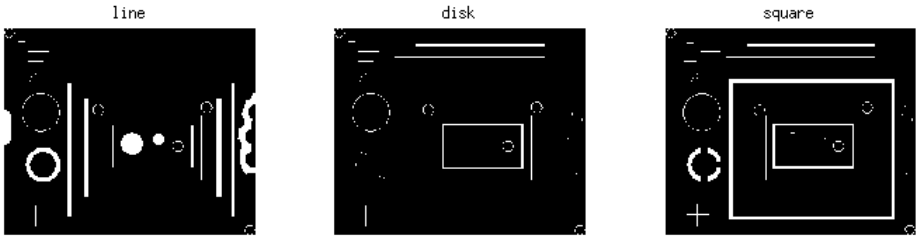

Figure 4: Top hat experiments

3.3 Bottom hat

The bottom hat operation (4) subtracts the original image from the closed, the effects of which can be seen in figure 5 below.

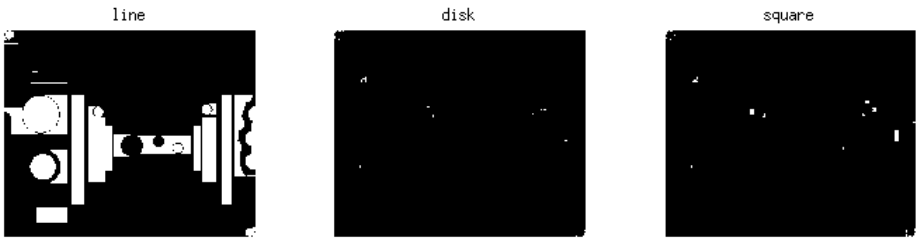

Figure 5: Bottom hat experiments

4 Background Normalization

After some experimentation I found that in and around the radii interval 7–9 the opened image begins to settle into a stabile image. As we can see in figure 6, when the radius is set to 7 we stil have a slight artifact on the left, but with radii 8 and 9 there is virtually no immediate visible change in the produced image.

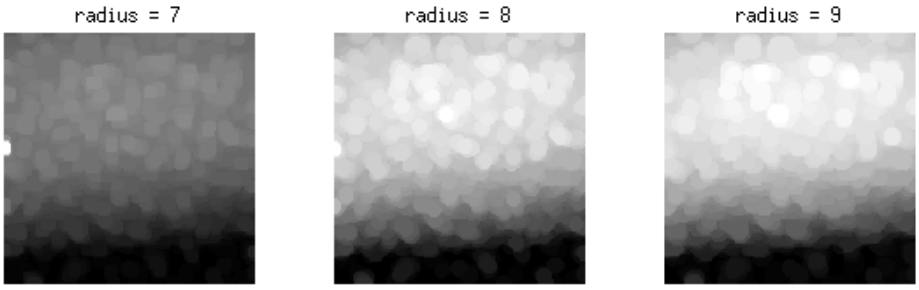

Figure 6: Background extractions of `rice.png` in the interval 7–9.

We choose a radius of 8 because it is the first in the interval to produce an even background result and the last from which notable changes can be seen, and if we were to continue increasing the radius we would end up with the original, or much like it — which would defeat the purpose.

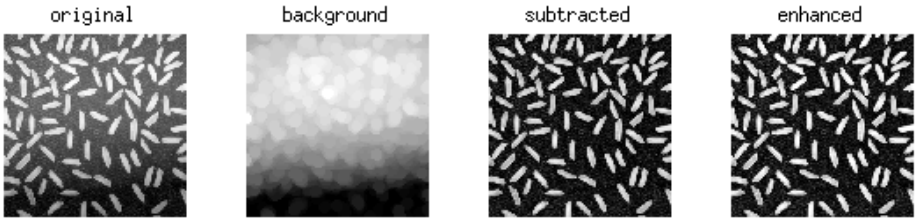

Figure 7: Optimal background normalization of `rice.png`

As apparent from figure 7 the background appears to have vanished from the image when we subtract it from the original image.

5 Digit Recognition

The strategy I followed to relax the method was to produce a very thin, minimal base shape of each digit to project hits from and a very forgiving, bold shape of each digit to project misses from. Each of these are shown in figure 8 below.

The first (thin) shape was produced using the skeletonization technique after a slight erosion which makes for simpler skeletons, and the latter by simply eroding the inverse.

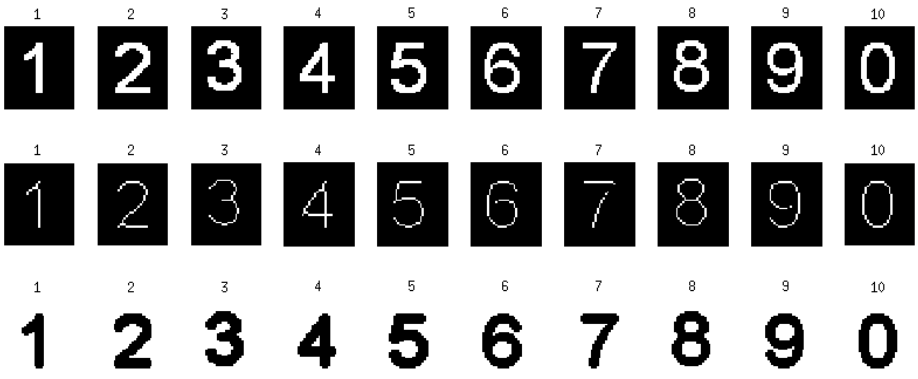

Figure 8: Original digits (top), skeletonized hit structural elements (middle) and expanded (or eroded) miss structural elements

These were then used as the structural elements for detecting the digit shapes in the source image, where B_1 is designated as being the skeletonized shape and B_2 as the bold shape.

Figure 9: Points of recognition, based on hit-or-miss calculation

As evident of figure 9 above, we see that we do indeed detect all three digits in the source image. Review the code that was used to produce these results in appendix A.4.

6 Feature Extraction

After a bit of twisting and tweaking of the knobs involved in the pipeline I ended up using, I arrived at a pipeline that; produces a gray-scale image of the image, detects the edges of the gray-scale version by subtracting a dilated from an eroded version of it, and then closed.

Each step of the pipeline is shown in figure 10 below.

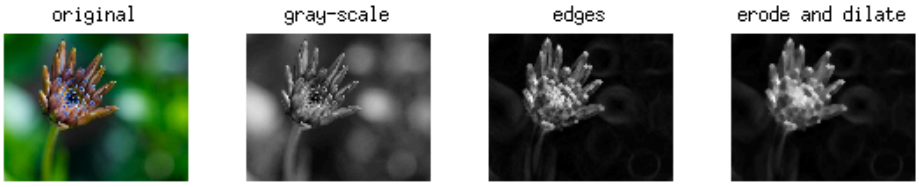

Figure 10: Pipeline

I did try to perform thickening, but I couldn't get it to work properly, and produced far worse results. It does, however, produce a good result, which is shown in figure 11 below.

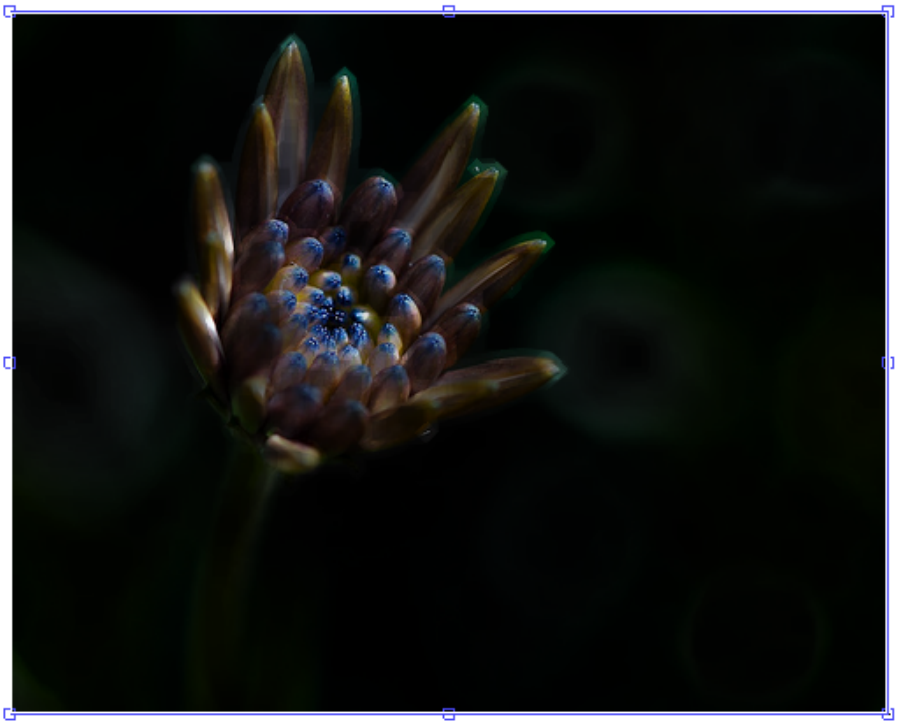

Figure 11: Result on a black background

Further adjustments could be made to increase the alpha-channel mask, but the overall result is somewhat satisfactory as we have managed to capture and mostly isolated the shape of the flower.

A Assignments

A.1 Opening and Closing

```
8  %% assignment 2
9
10 ones(10)*ones(10); % bug circumvention
11 clc, clear all;
12
13 I = load('images/TestImageQn2.mat');
14 SE = strel('diamond', 1); % R=2 closes entirely
15
16 I1 = imopen(I.TestImageQn2, SE);
17 I2 = imclose(I.TestImageQn2, SE);
18
19 figure (1), colormap(gray);
20 subplot(1,3,1), imagesc(I.TestImageQn2), axis image, axis off, title('original
       ');
21 subplot(1,3,2), imagesc(I1), axis image, axis off, title('opened');
22 subplot(1,3,3), imagesc(I2), axis image, axis off, title('closed');
```

Figure 12: Solution code for assignment 2 (../assignment.m)

A.2 Experimentation

```
25 %% assignment 3
26
27 clc, clear all;
28
29 I = imread('images/blobs.png');
30 N = 3;
31
32 SE = [strel('line', 50, 1), strel('disk', 3), strel('square', 6)];
33 t = cellstr(['line  '; 'disk  '; 'square']);
34
35 % hit-or-miss
36 figure (2), colormap(gray);
37 for i = 1:N
38     R = bwhitmiss(I, SE(i), strel(not(SE(i).getnhood())));
39     subplot(1,N,i), imagesc(R), axis image, axis off, title(t(i));
40 end
41
42 % top hat
43 figure (3), colormap(gray);
44 for i = 1:N
45     R = imtophat(I, SE(i));
46     subplot(1,N,i), imagesc(R), axis image, axis off, title(t(i));
47 end
48
49 % bottom hat
50 figure (4), colormap(gray);
51 for i = 1:N
52     R = imbothat(I, SE(i));
53     subplot(1,N,i), imagesc(R), axis image, axis off, title(t(i));
54 end
```

Figure 13: Solution code for assignment 3 (../assignment.m)

A.3 Background Normalization

```
57 %% assignment 4
58
59 clc, clear all;
60
61 I = imread('images/rice.png');
62 B7 = imopen(I, strel('disk',7));
63 B8 = imopen(I, strel('disk',8)); % optimal radius
64 B9 = imopen(I, strel('disk',9));
65 S = imsubtract(I,B8);
66 R = imadjust(S);
67
68 figure (5), colormap(gray);
69 subplot(1,3,1), imagesc(B7), axis image, axis off, title('radius = 7');
70 subplot(1,3,2), imagesc(B8), axis image, axis off, title('radius = 8');
71 subplot(1,3,3), imagesc(B9), axis image, axis off, title('radius = 9');
72
73 figure (6), colormap(gray);
74 subplot(1,4,1), imagesc(I), axis image, axis off, title('original');
75 subplot(1,4,2), imagesc(B8), axis image, axis off, title('background');
76 subplot(1,4,3), imagesc(S), axis image, axis off, title('subtracted');
77 subplot(1,4,4), imagesc(R), axis image, axis off, title('enhanced');
```

Figure 14: Solution code for assignment 4 (../assignment.m)

A.4 Digit Recognition

```
80  %% assignment 5
81
82  clc, clear all;
83  DS = {}; SE = {}; I = imread('images/test_digits.bmp');
84  FS = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine',
        'zero'};
85
86  % load digit files
87  for d = 1:size(FS,2)
88      DS{d} = imread(strcat('images/', FS{d}, '.bmp'));
89
90      % construct structural elements
91      E1 = imerode(DS{d}, strel('diamond', 1));
92      E2 = imerode(~DS{d}, strel('diamond', 1));
93      SE1 = strel(bwmorph(E1, 'skel', Inf));
94      SE2 = strel(E2);
95      SE{d} = [SE1,SE2];
96  end
97
98  % display digits images
99  figure (7), colormap(gray);
100 l = size(DS,2);
101 for d = 1:l
102     subplot(3,10,d), imagesc(DS{d}), axis image, axis off, title(int2str(d));
103     subplot(3,10,(d+l)), imagesc(SE{d}(1).getnhood), axis image, axis off,
            title(int2str(d));
104     subplot(3,10,(d+(2*l))), imagesc(SE{d}(2).getnhood), axis image, axis off,
            title(int2str(d));
105 end
106
107 % perform hit or miss and display detections
108 figure (9), colormap(gray);
109 N = [10,3,5];
110 R = cat(1, zeros(size(I)));
111 size(R)
112 for i = 1:size(N,2)
113     P = bwhitmiss(I, SE{N(i)}(1).getnhood, SE{N(i)}(2).getnhood);
114     P = imdilate(P, strel('disk',15));
115     R = R + P;
116 end
117 subplot(1,2,1), imagesc(I), axis image, axis off, title('original');
118 subplot(1,2,2), imagesc(R-I), axis image, axis off, title('detections');
```

Figure 15: Solution code for assignment 5 (../assignment.m)

A.5 Feature Extraction

```
120 %% assignment 6
121
122 clc, clear all;
123
124 N = 5; sigma = 2; R = 5;
125
126 Disk = strel('diamond', R);
127 Plain = strel(ones(8));
128
129 I = imread('images/flowers.jpg');
130 G = rgb2grey(I);
131 H = fspecial('gaussian', [N N], sigma);
132 A = G; % imfilter(G, H);
133 Amin = imerode(A, Disk);
134 Amax = imdilate(A, Disk);
135 E = Amax - Amin;
136 DE = imerode(imdilate(E, Plain), Plain);
137
138 figure (8), colormap(gray);
139 subplot(1,4,1), imagesc(I), axis image, axis off, title('original');
140 subplot(1,4,2), imagesc(G), axis image, axis off, title('gray-scale');
141 subplot(1,4,3), imagesc(E), axis image, axis off, title('edges');
142 subplot(1,4,4), imagesc(DE), axis image, axis off, title('erode and dilate');
143
144 figure(9), colormap(gray);
145 imshow(cat(1, zeros(size(I))));
146 hold on;
147 h = imshow(I); set(h, 'AlphaData', DE);
148 hold off;
```

Figure 16: Solution code for assignment 6 (../assignment.m)

B Functions

B.1 RGB to Gray-scale

```
1 function [ out ] = rgb2grey( I )
2     out = I(:,:,1) * 0.2989 + I(:,:,2) * 0.5870 + I(:,:,3) * 0.1140;
3 end
```

Figure 17: RGB to Gray-scale converter (../rgb2grey.m)

References

[1] Solomon & Breckon, Fundamentals of Digital Image Processing - A Practical Approach with Examples in Matlab, 2011