School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

APPM4058A Assignment 2 – Report

Wesley Earl Stander - 1056114 - Coms Hons

May 13, 2020

1 Question 1

1.1 Question 1a

			1			1			1			
					1							
	1											

1.2 Question 1b

The following 2 structuring elements would be used in a 4-pass union of hit-or-miss transform to obtain the image in question. The image would go over 4 passes of hit-or-miss transform and at each pass the structuring elements would be rotated by 90 degrees and the respective image saved. After the 4 passes of hit-or-miss transform have been applied the union of the 4 images would provide the resultant image in question.

	0	
0	1	0

	1	
1	0	1

2 Question 2

2.1 Question 2a

The image must be thresholded and made a binary image for the following method. The process requires the searching through the image for the first foreground pixel by iterating through the pixels. Whence this pixel is found, the component can be extracted through the continuous dilation of the pixel image and intersection of the original image until there is no difference between consecutive dilations hence attaining the object. The object image is then stored and subtracted from the original image to attain the image for the next step. Repeat the steps from the search until now using the image attained in the last step. Stop when the search does not return a foreground pixel because all objects have been removed. Now that all the objects have been extracted into separate images, iterate through them adding them to one image and multiplying their pixel values by the current iteration counter. Whence this step is complete a new image whereby all the connected components are labelled with integer values increasing from 1 through to the number of connected components will be the result.

2.2 Question 2b

The above technique was implemented in python. The image is converted into a binary image through otsu-thresholding. The colouring is done by a colour array generated and then put into a colour map for the image saving. A find pixel function is implemented that returns an image where the first pixel found is the only foreground pixel. The image then does a connected extraction on that pixel through the continuous dilation and logical and with the original image until convergence. The extracted component is then removed from the image and saved in an array so that the next can be extracted. Whence all the components are extracted, they are labelled and then the colours map to the pixel values. Each object has the same pixel value in the range 1 to the amount of connected components. The result of DIP.tif in figure 1 is the required output and detects 5 connected components. The result of balls-with-reflections.tif in figure 2 is the required output and detects 16 connected components.

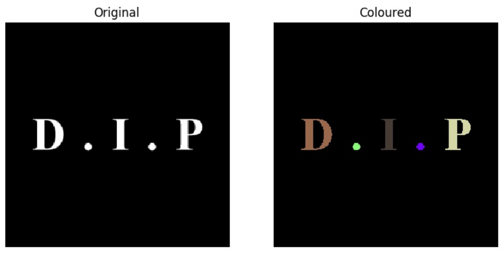

Figure 1: Connected component result of DIP.tif

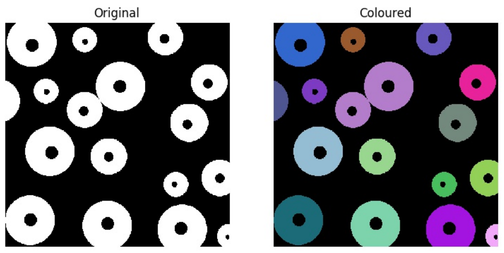

Figure 2: Connected component result of balls-with-reflections.tif

2.3 Question 2c

The same colouring system is used, which requires the same input to be coloured. Both images have each connected component labelled in incremented order and their system of labelling images seems to work from top to bottom and left to right just like my implementation.

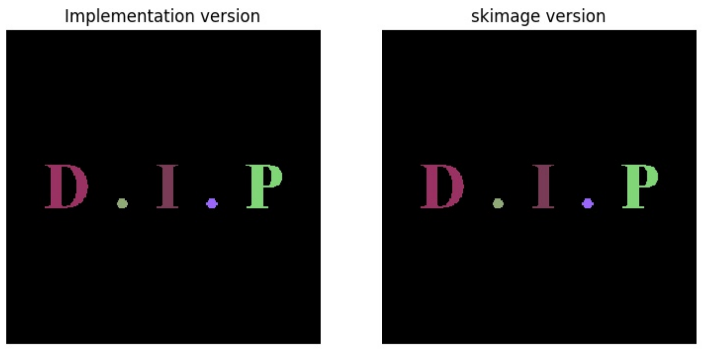

Figure 3: Comparison between implementation and skimage.measure.label of DIP.tif

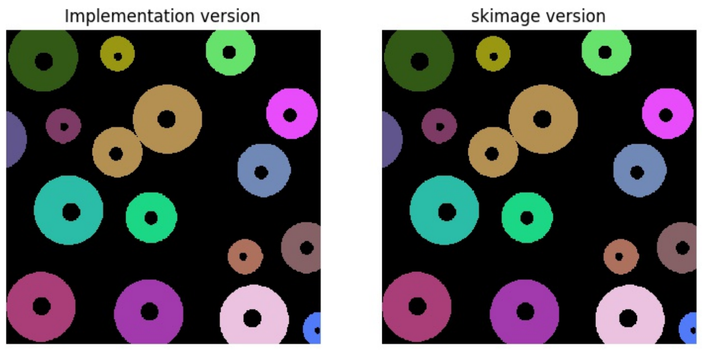

Figure 4: Comparison between implementation and skimage.measure.label of balls-with-reflections.tif

3 Question 3

A closing is performed on the original image using a disk of radius 28. An opening is then applied on the result using a disk of radius 50. The image is then otsu-thresholded and a boundary extraction is applied. This boundary is then added to the original image to superimpose the separation line and produce the result on the right. The thickness of the disk in the erosion to attain the boundary extraction will determine the thickness of the superimposed boundary line. The result is the required output.

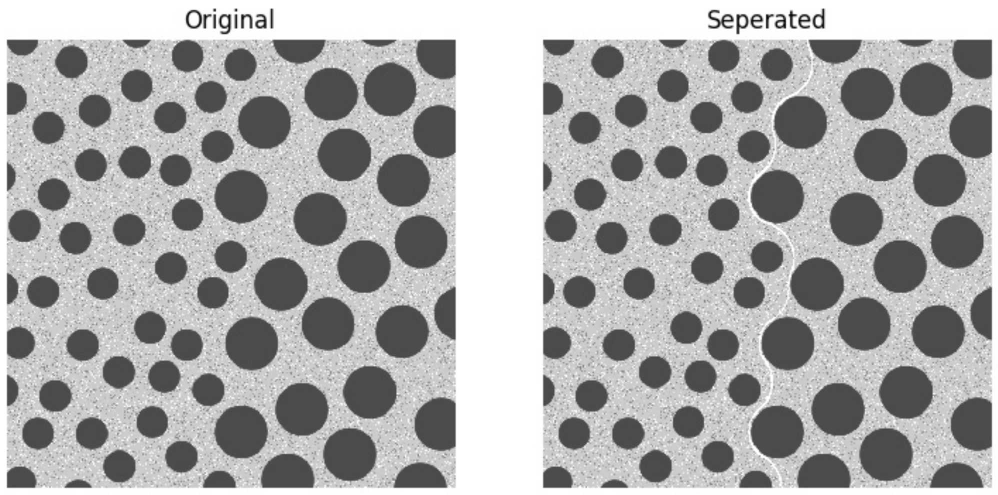

Figure 5: Separated blobs

4 Question 4

The image undergoes a dilation with a disk structuring element of radius 25. The image is then thresholded with an otsu-threshold. The image is inverted as the disks are darker. The components are then extracted and labelled with the technique used in question 2. The total amount of coins is then outputted alongside the colour mapped image using the colour mapping technique discussed in question 2. The result is the required output.

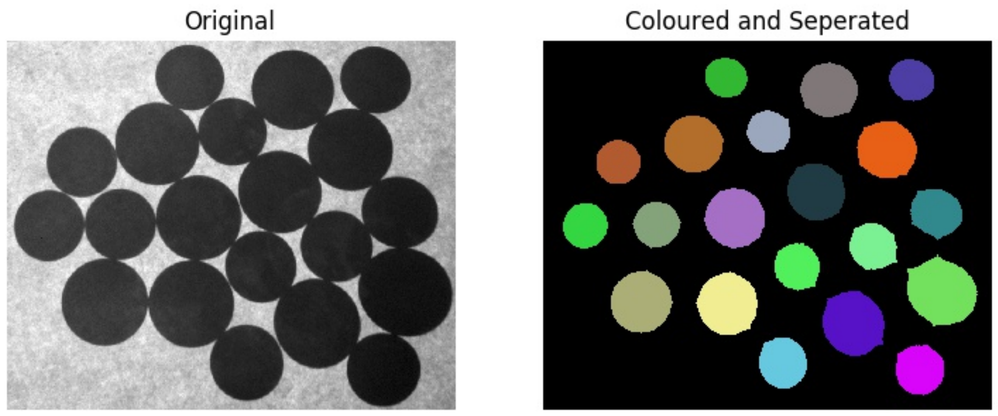

Figure 6: Coloured and separated coins

5 Source Code list

5.1 Libraries

1. numpy
2. matplotlib
3. skimage.color.rgb2gray
4. skimage.morphology
5. skimage.filters.threshold_otsu
6. random.random

5.2 Question 2

```python
import matplotlib.pyplot as plt
import numpy as np
from skimage.filters import threshold_otsu
from skimage.color import label2rgb
from skimage.color import rgb2gray
import skimage.morphology as m
from skimage import measure
from random import random
import matplotlib

def FindPixel(img): #find foreground pixel image
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j] > 0:
                img2 = np.zeros((img.shape[0], img.shape[1]))
                img2[i][j] = img[i][j]
                return img2
    return np.zeros(1)

def connectedExtraction (img, imgE):
    SE = m.square(3)
    img0 = np.logical_and(m.dilation(imgE, SE), img)
    imgTemp = np.ones(1)
    while(not (img0==imgTemp).all()): #check for change in subsequent iterations
        imgTemp = np.logical_and(m.dilation(img0, SE), img)
        img0 = np.logical_and(m.dilation(imgTemp, SE), img)
    return img0

def ExtractAndLabelComponents(img):
    components = []
    imgP = FindPixel(img)
    while imgP.shape[0] != 1: #extract each component
        components.append(connectedExtraction(img, imgP))
        img = np.bitwise_xor(img,components[len(components)-1])
        imgP = FindPixel(img)
    for i in range(len(components)): #label
        components[i] = np.multiply(components[i], i+1)
    return components

colors = [(0,0,0)] #generate colour mapping
for i in range(255):
    if i%25==0 and i >= 50:
        colors.append((i/255,random(),random()))
for i in range(255):
    if i%25==0 and i >= 50:
        colors.append((random(),i/255,random()))
for i in range(255):
    if i%25==0 and i >= 50:
        colors.append((random(),random(),i/255))
new_map = matplotlib.colors.LinearSegmentedColormap.from_list('new_map', colors, N=27)

img2 = rgb2gray(plt.imread('imgs/DIP.tif'))
thresh = threshold_otsu(img2)
binary = rgb2gray(img2 > thresh)
c = ExtractAndLabelComponents(binary)
l = c[0]
for i in range(len(c)-1):
    l += c[i+1]
print("Number of connected components in DIP: ", len(c)) #number of connected components
comparison = measure.label(binary)
plt.gray()
plt.imsave("output/2b.jpg",l, cmap=new_map)

img22 = rgb2gray(plt.imread('imgs/balls-with-reflections.tif'))
thresh2 = threshold_otsu(img22)
binary2 = rgb2gray(img22 > thresh2)
c2 = ExtractAndLabelComponents(binary2)
l2 = c2[0]
for i in range(len(c2)-1):
    l2 += c2[i+1]
print("Number of connected components in balls-with-reflections: ", len(c2)) #number of connected
     components
comparison2 = measure.label(binary2)
plt.gray()
plt.imsave("output/2b-2.jpg",l2, cmap=new_map)

fig, axs = plt.subplots(1,2,figsize=(9,5))
[axi.set_axis_off() for axi in axs.ravel()]
axs[0] = fig.add_subplot(1,2,1)
axs[0].set_title("Original")
axs[0].imshow(img2)
plt.axis('off')
axs[1] = fig.add_subplot(1,2,2)
axs[1].set_title("Coloured")
axs[1].imshow(l, cmap = new_map)
plt.axis('off')
fig.savefig("output/2bComparison.jpg")

fig, axs = plt.subplots(1,2,figsize=(9,5))
[axi.set_axis_off() for axi in axs.ravel()]
axs[0] = fig.add_subplot(1,2,1)
axs[0].set_title("Original")
axs[0].imshow(img22)
```

```
plt.axis('off')
axs[1] = fig.add_subplot(1,2,2)
axs[1].set_title("Coloured")
axs[1].imshow(l2, cmap = new_map)
plt.axis('off')
fig.savefig("output/2b-2Comparison.jpg")

fig, axs = plt.subplots(1,2,figsize=(9,5))
[axi.set_axis_off() for axi in axs.ravel()]
axs[0] = fig.add_subplot(1,2,1)
axs[0].set_title("Implementation version")
axs[0].imshow(l, cmap = new_map)
plt.axis('off')
axs[1] = fig.add_subplot(1,2,2)
axs[1].set_title("skimage version")
axs[1].imshow(comparison, cmap = new_map)
plt.axis('off')
fig.savefig("output/2c.jpg")

fig, axs = plt.subplots(1,2,figsize=(9,5))
[axi.set_axis_off() for axi in axs.ravel()]
axs[0] = fig.add_subplot(1,2,1)
axs[0].set_title("Implementation version")
axs[0].imshow(l2, cmap = new_map)
plt.axis('off')
axs[1] = fig.add_subplot(1,2,2)
axs[1].set_title("skimage version")
axs[1].imshow(comparison2, cmap = new_map)
plt.axis('off')
fig.savefig("output/2c-2.jpg")
```

5.3 Question 3

```
import matplotlib.pyplot as plt
import skimage.morphology as m
from skimage.color import rgb2gray
from skimage.filters import threshold_otsu
import numpy as np

img3 = rgb2gray(plt.imread('imgs/blobs.tif'))
SE = m.disk(28)
img31 = m.closing(img3, SE) #remove small blobs
SE2 = m.disk(50)
img32 = m.opening(img31, SE2) #get big side
thresh3 = threshold_otsu(img32)
imgBoundary = img32 - m.erosion(img32, m.disk(5)) > thresh3 #extract boundary
output = np.multiply(imgBoundary,255)+img3
plt.gray()
plt.imsave("output/3.png", output, vmin=0, vmax=255)

fig, axs = plt.subplots(1,2,figsize=(9,5))
[axi.set_axis_off() for axi in axs.ravel()]
axs[0] = fig.add_subplot(1,2,1)
```

```
axs[0].set_title("Original")
axs[0].imshow(img3, vmin = 0, vmax= 255)
plt.axis('off')
axs[1] = fig.add_subplot(1,2,2)
axs[1].set_title("Seperated")
axs[1].imshow(output, vmin = 0, vmax= 255)
plt.axis('off')
fig.savefig("output/3Comparison.jpg")
```

5.4 Question 4

```
import matplotlib.pyplot as plt
import numpy as np
from random import random
import matplotlib
from skimage.filters import threshold_otsu
from skimage.color import rgb2gray
import skimage.morphology as m

def FindPixel(img): #find foreground pixel image
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j] > 0:
                img2 = np.zeros((img.shape[0], img.shape[1]))
                img2[i][j] = img[i][j]
                return img2
    return np.zeros(1)

def connectedExtraction (img, imgE):
    SE = m.square(3)
    img0 = np.logical_and(m.dilation(imgE, SE), img)
    imgTemp = np.ones(1)
    while(not (img0==imgTemp).all()): #check for change in subsequent iterations
        imgTemp = np.logical_and(m.dilation(img0, SE), img)
        img0 = np.logical_and(m.dilation(imgTemp, SE), img)
    return img0

def ExtractAndLabelComponents(img):
    components = []
    imgP = FindPixel(img)
    while imgP.shape[0] != 1: #extract each component
        components.append(connectedExtraction(img, imgP))
        img = np.bitwise_xor(img,components[len(components)-1])
        imgP = FindPixel(img)
    for i in range(len(components)): #label
        components[i] = np.multiply(components[i], i+1)
    return components

colors = [(0,0,0)] #generate colour mapping
for i in range(255):
    if i%25==0 and i >= 50:
        colors.append((i/255,random(),random()))
for i in range(255):
    if i%25==0 and i >= 50:
        colors.append((random(),i/255,random()))
for i in range(255):
    if i%25==0 and i >= 50:
        colors.append((random(),random(),i/255))
new_map = matplotlib.colors.LinearSegmentedColormap.from_list('new_map', colors, N=27)

img4 = rgb2gray(plt.imread('imgs/coins.png'))
SE = m.disk(25)
img41 = m.dilation(img4, SE)
thresh4 = threshold_otsu(img41)
binary4 = img41 < thresh4
c4 = ExtractAndLabelComponents(binary4)
l4 = c4[0]
for i in range(len(c4)-1):
    l4 += c4[i+1]
print("Coin amount is: ", len(c4))
plt.gray()
plt.imsave("output/4.png", l4, cmap=new_map)

fig, axs = plt.subplots(1,2,figsize=(9,5))
[axi.set_axis_off() for axi in axs.ravel()]
axs[0] = fig.add_subplot(1,2,1)
axs[0].set_title("Original")
axs[0].imshow(img4)
plt.axis('off')
axs[1] = fig.add_subplot(1,2,2)
axs[1].set_title("Coloured and Seperated")
axs[1].imshow(l4, cmap = new_map)
plt.axis('off')
fig.savefig("output/4Comparison.jpg")
```