HW5

Chengjiao Yang 04/23/2014

1. **Burger & Burge Exercise 11.5 (page 236) (25 points):** Implement a Java class for describing a binary image region using chain codes. It is up to you, whether you want to use an absolute or differential chain code. The implementation should be able to encode closed contours as chain codes and also reconstruct the contours given a chain code. Call your Java class **`Chain_codes`**

How to use:

1) copy "imagingbook.jar" to ...\ImageJ\plugins\jar

2) copy files in question1 to the ...\ImageJ\plugins

3) use imagej, drag a picture(I use spine.jpg), then run the ChainCode.java

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
2111222102222111222222212212122222323232322223666666666667676665666566656666567665555565656661
0000000000322111110145444543445554556560
2222222222222323222222222322223465501767666666676666666656666667676762
00100122111024543123455465666656560
000000000002124444445331111077243444456556556
222222222221222222222221222222466666666665666666666666666657662
0010121221222122222366666665665655544470
00001010122444545454544570
2121221212122243566666666665656561
0100001114454544444444460070
00124532212122212256665656565666661
21775021223255322366666672
1222213221222322666666665666666662
021111255455650
0010244445470
2323226665770
222222225676666662
013446751
1245571
0134560
2221246666662
2322567773
22124666662
```

The chain code will list like this, each line is a region.

2. **Burger & Burge Exercise 11.12 (page 237) (25 points):** Write an ImageJ plugin that (a) finds (labels) all regions in a binary image (b) computes the orientation and eccentricity for each region, and (c) shows the results as a direction vector and the equivalent ellipse on top of each region (as exemplified in Fig 11.19. Hint: Use Eqn. (11.33) to develop a method for drawing ellipses at arbitrary orientations (not available in ImageJ). Call your ImageJ plugin **`Region_labeling.java`**

How to use:

1) copy "imagingbook.jar" to ...\ImageJ\plugins\jar

2) copy files in question2 to the ...\ImageJ\plugins

3) use imagej, drag a picture(I use spine.jpg), then run the Region_labeling.java

show the orientation and ellipse

the eccentricity will list in log

```
Eccentricity = Infinity
Eccentricity = 22.037957076517515
Eccentricity = 17.701568387825233
Eccentricity = 2.350265157369761
Eccentricity = 2.6666666666666665
Eccentricity = 9.122800060078957
Eccentricity = 41.80574123403406
Eccentricity = 4.333333333333331
Eccentricity = 1.932187273178555
```

3. **(6 points)** Show all your work. By hand convert the following RGB value = (0, 184, 160) to HSV

$Chigh = \max(R,G,B) = 184$

$Clow = \min(R,G,B) = 0$

$Crng = Chigh - Clow = 184 > 0$

$V = Chigh/cMax = 184/255 = 0.72$

$S = Crng / Chigh = 184/184 = 1$

cause $G = Chigh$

So $H' = R' - B' + 2 = (Chigh - R)/Crng - (Chigh - B)/Crng + 2 = 184/184 - 24/184 + 2 = 3.13$

$H = 1/6 * H' = 0.52$

$HSV = (0.52, 1, 0.72)$

4. **(6 points)** how all your work. By hand, convert the following HSV values = (0.67, 0.7, 0.7) to RGB

$H' = (6 * Hhsv) \bmod 6 = (6 * 0.67) \bmod 6 = 4.02$

$c1 = |H'| = 4$

$c2 = H' - c1 = 0.02$

$x = (1 - s) * v = 0.3 * 0.7 = 0.21$

$y = (1-(s*c2))*v = (1 - 0.014) * 0.7 = 0.69$

$z = (1-(s*(1-c2)))*v = (1-0.7*0.98)*0.7 = 0.22$

thus

$(R',G',B') = (z,x,v) = (0.22,0.21,0.7)$

$R = \min(round(R'*255),255) = 56$

$G = \min(round(G'*255),255) = 54$

$B = \min(round(B'*255),255) = 179$

$RGB = (56, 54, 179)$

5. **(6 points)** Show all your work. By hand, convert the following RGB value = (124, 220, 0) to YIQ

$Y = 0.299R + 0.587G + 0.114B = 37.076 + 129.14 + 0 = 166.216$

$I = 0.596R - 0.275G - 0.321B = 73.904 - 60.5 - 0 = 13.404$

$Q = 0.212R - 0.523G + 0.311B = 26.288 - 115.06 + 0 = -88.772$

$YIQ = (166.216, 13.404, -88.772)$

6. **(6 points)** Show all your work. By hand, convert the following YIQ value = (1, 0.3, 0.3) to RGB

$R = Y + 0.956I + 0.621Q = 1 + 0.2868 + 0.1863 = 1.4731$

$G = Y - 0.272I - 0.647Q = 1 - 0.0816 - 0.1941 = 0.7243$

$B = Y - 1.106I + 1.703Q = 1 - 0.3318 + 0.5109 = 1.1791$

RGB = (1.5, 0.7, 1.2)

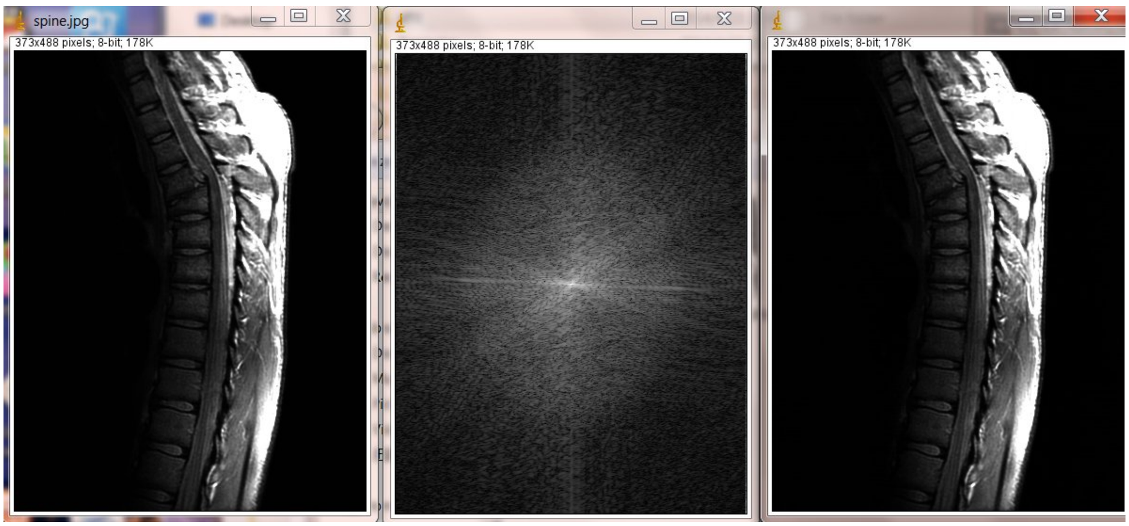
7. (26 Points) Burger & Burge Exercise 14.1 (page 366): Implement the two-dimensional DFT using the one-dimensional DFT, as described in Sec. 14.1.2. Apply the 2D DFT to real intensity images of arbitrary size and display the results (by converting to ImageJ `float` images). Implement the inverse transform and verify that the back-transformed result is identical to the original image

How to use:

1) copy "imagingbook.jar" to ...\ImageJ\plugins\jar

2) copy files in question3 to the ...\ImageJ\plugins

3) use imagej, drag a picture(I use spine.jpg), then run the DFT_COM.java



the original image is in the left, then show the DFT in middle, then reform the image at right part.