Michael Christensen
September 21, 2013
CS 450

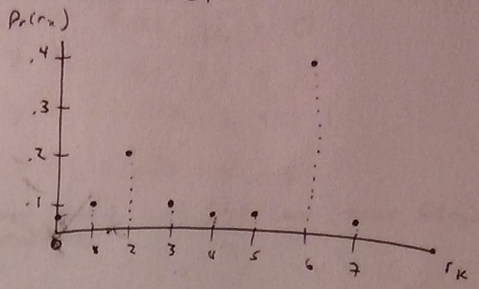
Homework #2

Written Exercises:
See the following 4 pages:

MICHAEL CHRISTENSEN SEPT-21-2013 CS450

(1) INTENSITY DISTRIBUTION:

r_k	n_k	$P_r(r_k) = n_k/MN$, $MN = 32 \times 32 = 1024$
$r_0 = 0$	62	$62/1024 = 0.06$
$r_1 = 1$	100	$100/1024 = 0.10$
$r_2 = 2$	200	$200/1024 = 0.20$
$r_3 = 3$	100	$100/1024 = 0.10$
$r_4 = 4$	50	$50/1024 = 0.05$
$r_5 = 5$	50	$50/1024 = 0.05$
$r_6 = 6$	400	$400/1024 = 0.39$
$r_7 = 7$	62	$62/1024 = 0.05$ (REALLY 0.06, BUT MADE 0.05 TO ADD UP TO 1.00)



$S_k = T(r_k) = (L-1) \sum_{j=0}^{k} P_r(r_j)$, $(L-1) = 8-1 = 7$

$S_0 = T(r_0) = 7\, p_r(r_0) = 7 * 0.06 = 0.42$

$S_1 = T(r_1) = 0.42 + 7\, p_r(r_1) = 0.42 + 7 * .1 = 1.12$

$S_2 = T(r_2) = 1.12 + 7 * (0.2) = 2.52$

$S_3 = T(r_3) = 2.52 + 7 * .1 = 3.22$

$S_4 = T(r_4) = 3.22 + 7 * .05 = 3.57$

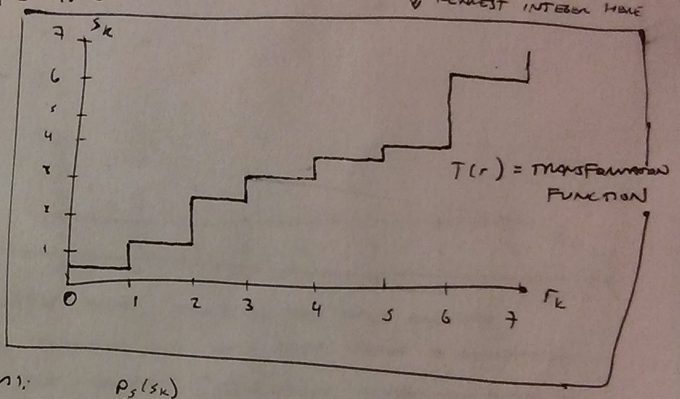
$S_5 = T(r_5) = 3.57 + 7 * .05 = 3.92$

$S_6 = T(r_6) = 3.92 + 7 * 0.39 = 6.65$

$S_7 = T(r_7) = 6.65 + 7 * .05 = 7.00$

SHOULD ROUND

I DON'T THINK THE BOOK ROUNDED TO NEAREST INTEGER HERE



(2) SCALED HISTOGRAM-EQUALIZED VALUES FROM PROBLEM 1:

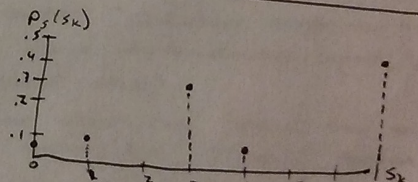
$S_0 = 0.42 \Rightarrow 0$ → $P_s(s_k) = 0.06$

$S_1 = 1.12 \Rightarrow 1 \rightarrow 0.1$

$S_2 = 2.52 \Rightarrow 3$, $S_3 = 3.22 \Rightarrow 3$ } $\frac{200+100}{1024} = 0.3$

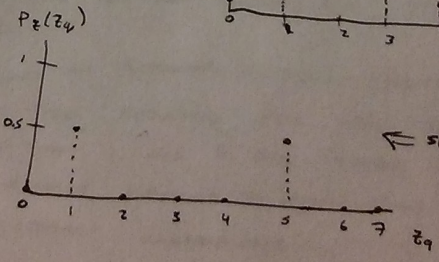
$S_4 = 3.57 \Rightarrow 4$, $S_5 = 3.92 \Rightarrow 4$ } $0.05 + 0.05 = 0.1$

$S_6 = 6.65 \Rightarrow 7$, $S_7 = 7.0 \Rightarrow 7$ } $0.39 + 0.05 = 0.44$



SPECIFIED HISTOGRAM'S INTENSITY DIST:

z_q	n_q	$P_z(z_q) = n_q/MN$
$z_0 = 0$	0	0
$z_1 = 1$	512	$512/1024 = 0.5$
$z_2 = 2$	0	0
$z_3 = 3$	0	0
$z_4 = 4$	0	0
$z_5 = 5$	512	$512/1024 = 0.5$
$z_6 = 6$	0	0
$z_7 = 7$	0	0



← SPECIFIED HISTOGRAM

VALUES OF THE TRANSFORMATION FUNCTION G:

$G(z_0) = (L-1) \sum_{j=0}^{0} p_z(z_0) = 7 * 0 = 0$

$G(z_1) = 7(0 + 0.5) = 3.5 \rightarrow 4$

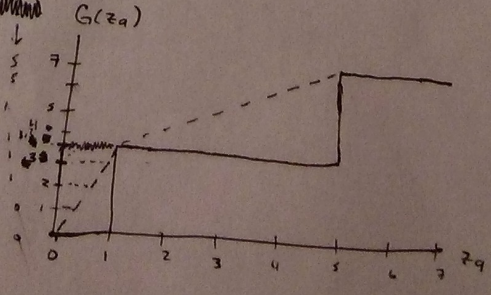
$G(z_2) = 7(0 + 0.5 + 0) = 3.5 \rightarrow 4$

$G(z_3) = 7(0 + 0.5 + 0 + 0) = 3.5 \rightarrow 4$

$G(z_4) = 7(0 + 0.5 + 0 + 0 + 0) = 3.5 \rightarrow 4$

$G(z_5) = 7(0 + 0.5 + 0 + 0 + 0 + 0.5) = 7.0$

$G(z_6) = 7.0$, $G(z_7) = 7.0$



(CONTINUED)

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	4
$z_2 = 2$	4
$z_3 = 3$	4
$z_4 = 4$	4
$z_5 = 5$	7
$z_6 = 6$	7
$z_7 = 7$	7

$G(z) = \text{round}(7 T(z))$	
0	0
1	0
2	1
3	1
4	1
5	1
6	5
7	6

$s_k \rightarrow z_q$

$s_0 = 0 = G(z_0), z_0 = 0$ $(s_0 = z_0)$

$s_1 = 1 \neq G(\ ), z_0 = 0$

$s_2 > 3 \neq G(\ ), z = 1$

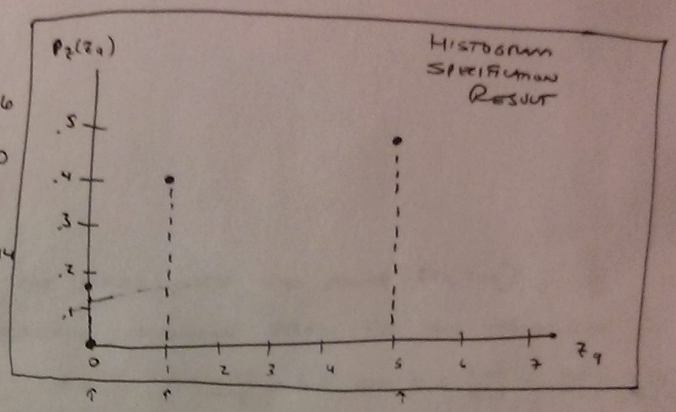
$s_3 > 4 = G(z_1), z_1 = 1$

$s_4 > 7 = G(z_4), z_4 = 5$

FIND SMALLEST VALUE OF z_q S.T. THE VALUE $G(z_q)$ IS THE CLOSEST TO s_k.

SPECIFIED & ACTUAL HISTOGRAMS

z_q	$p_z(z_q)$	$p_z(z_k)$	
$z_0 = 0$	0	$s_0 = 0$ 0.06, $s_1 = 1$ 0.10	> 0.16
$z_1 = 1$	0.5	$s_2 =$ 0.2, $s_3 =$ 0.1, $s_4 =$ 0.05, $s_5 =$ 0.05	> 0.40
$z_2 = 2$	0		
$z_3 = 3$	0		
$z_4 = 4$	0		
$z_5 = 5$	0.5	$s_6 =$ 0.39, $s_7 =$ 0.05	> 0.44
$z_6 = 6$	0		
$z_7 = 7$	0		



IF TWO INPUT VALUES MAP TO THE SAME OUTPUT VALUE, IT IMPLIES THE IMAGE CANNOT BE REPRESENTED WITH A STRICTLY MONOTONIC FUNCTION, AND YOU MUST HAVE A CONVENTION TO CHOOSE THE PARTICULAR VALUE INTO THE OUTPUT FUNCTION WHICH YOU WISH TO USE [FOR EXAMPLE, CHOOSING THE SMALLEST VALUE INTO THE SPECIFIED OUTPUT TRANSFORMATION FUNCTION WHICH GETS YOU CLOSEST TO AN INPUT VALUE].

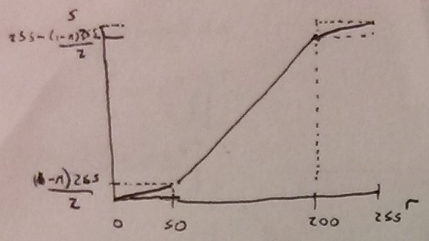
IF NO INPUT VALUE MAPS ONTO A PARTICULAR OUTPUT VALUE, THIS IMPLIES THAT YOU SHALL DON'T HAVE ANY NEED TO DO SHIFTING FOR THAT OUTPUT AND IT WON'T APPEAR IN THE NEW HISTOGRAM.

THE SPECIFIED HISTOGRAM DIFFERS FROM THE RESULTING HISTOGRAM SPECIFICATION SLIGHTLY. SINCE THIS WAS THE DISCRETE CASE, THE RESULTING HIST. SPEC. ISN'T PERFECTLY UNIFORM, BUT LOOKS SIMILAR IN THAT 1 AND 5 ARE HIGHER, BUT NOTICEABLY DIFFERENT IS HOW 0 ALSO HAS A VALUE, SEEMING TO SHOW HOW THE SCALING HISTEQUALIZED VALUE OF 0 AND 1 IN PROBLEM 1 CONTRIBUTED.

(3) THE APPROACH I WOULD USE IS A PIECE-WISE FUNCTION (SIMILAR TO WHAT IS MENTIONED IN THE SLIDES) TO STRETCH THE CONTRAST OF VALUES IN THE RANGE [50, 200], SINCE THAT IS WHERE MOST OF THE IMAGE'S VALUES ARE. THAT, PROPORTION THE OUTPUT TO GIVE MORE WEIGHT TO VALUES IN THAT RANGE.

$$s = \begin{cases} mr, & m = \left(\frac{\left(\frac{1-n}{2}\right) * 255}{2}\right) / (50-0) & \text{IF } r < 50 \\ mr, & m = \frac{n * 255}{(200-50)} & \text{IF } 50 \le r \le 200 \\ mr, & m = \left(\frac{\left(\frac{1-n}{2}\right) * 255}{2}\right) / (255-200) & \text{IF } r > 200 \end{cases}$$

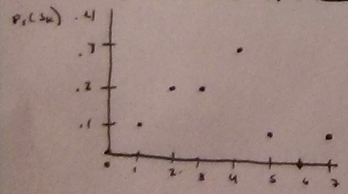
THIS DOES THE FOLLOWING, IN EFFECT:



WHERE n IS THE PERCENTAGE OF THE IMAGE INSIDE THE RANGE [50, 200], SO THAT THE SLOPE OF THE TRANSFORMATION FUNCTION BELOW 50 AND ABOVE 200 IS LESS THAN 1 (EG. IF $n = 95\%$, $1-n = 5\%$, $\frac{0.05 \times 255}{2} = 6.3$, AND $m = \frac{\text{RISE}}{\text{RUN}} = \frac{6.3}{50} = 0.13$, THE SLOPE BETWEEN [0, 50], AND THE SLOPE BETWEEN [50, 200] WOULD BE $\frac{.95 \times 255}{200-50} = 1.615$, > 1). THE CONTENT IN [50, 200] IS STRETCHED INTO MORE OUTPUT, PLACING MORE EMPHASIS ON IT. I DO THIS TO PRESERVE SOME OF THE CONTRAST IN THE LEVELS BELOW 50 AND ABOVE 200, INSTEAD OF JUST DOING A BINARY WHITE OR BLACK. I DON'T THINK HIST. EQUILIZATION WOULD NECESSARILY DO BETTER, SINCE MY APPROACH FOR WINDOWING IS BASICALLY DOING WHAT HIST. EQU. WOULD DO, RELYING ON PROPORTIONS OF THE IMAGE TO DETERMINE HOW MUCH OF THE OUTPUT TO DEVOTE TO THAT SECTION OF INPUT.

(4) GRW 3.7 - SHOW THAT A 2ND PASS OF HIST. EQUI. ON THE HIST-EQUALIZED IMAGE WILL PRODUCE EXACTLY SAME RESULT AS THE 1ST PASS.

TAKE FOR EXAMPLE AN EQUALIZED HISTOGRAM:

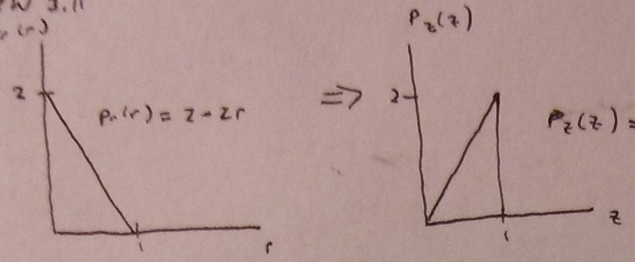


DOING HISTOGRAM EQUATION ON THIS WON'T PRODUCE ANY CHANGE, FOR EXAMPLE GIVEN AN IMAGE OF 1024 PIXELS,

s_k		
0	→	$\frac{0 * 0}{1024} = 0$, SAME AS IN EQUALIZED HISTOGRAM
1	→	$\frac{.1 * 1024}{1024} = .1$, " "
2	→	$\frac{.2 * 1024}{1024} = .2$, " "
3	→	$\frac{.2 * 1024}{1024} = .2$, " "
4	→	$\frac{.3 * 1024}{1024} = .3$, " "
5		
6		⇓
7		ETC.

BY EQUALIZING AN EQUALIZED IMAGE, YOU HAVE NO PROPORTIONS TO CHANGE OR PIXEL VALUES TO SHIFT.

⑤ CRW 3.11



ASSUME CONTINUOUS QUANTITIES, FIND TRANSFORMATION TO ACCOMPLISH THIS:

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$

$$= (L-1)\int_0^r (2 - 2w)\,dw$$

$$= 2(L-1)\int_0^r (1-w)\,dw$$

$$= 2(L-1)\left[w - \frac{w^2}{2}\right]\Big|_0^r$$

$$s = 2(L-1)\left[r - \frac{r^2}{2}\right]$$

$$G(z) = (L-1)\int_0^z p_z(w)\,dw$$

$$= 2(L-1)\int_0^z w\,dw$$

$$= 2(L-1)\left.\frac{w^2}{2}\right|_0^z$$

$$= (L-1)\,z^2 = s$$

$$z = \left[\frac{s}{(L-1)}\right]^{1/2}$$

$$z = \left(\frac{2(L-1)}{(L-1)}\left(r - \frac{r^2}{2}\right)\right)^{1/2}$$

$$= \left(2\left(r - \frac{r^2}{2}\right)\right)^{1/2}$$

$$\boxed{z = \sqrt{2\left(r - \frac{r^2}{2}\right)} = \sqrt{2r - r^2}}$$

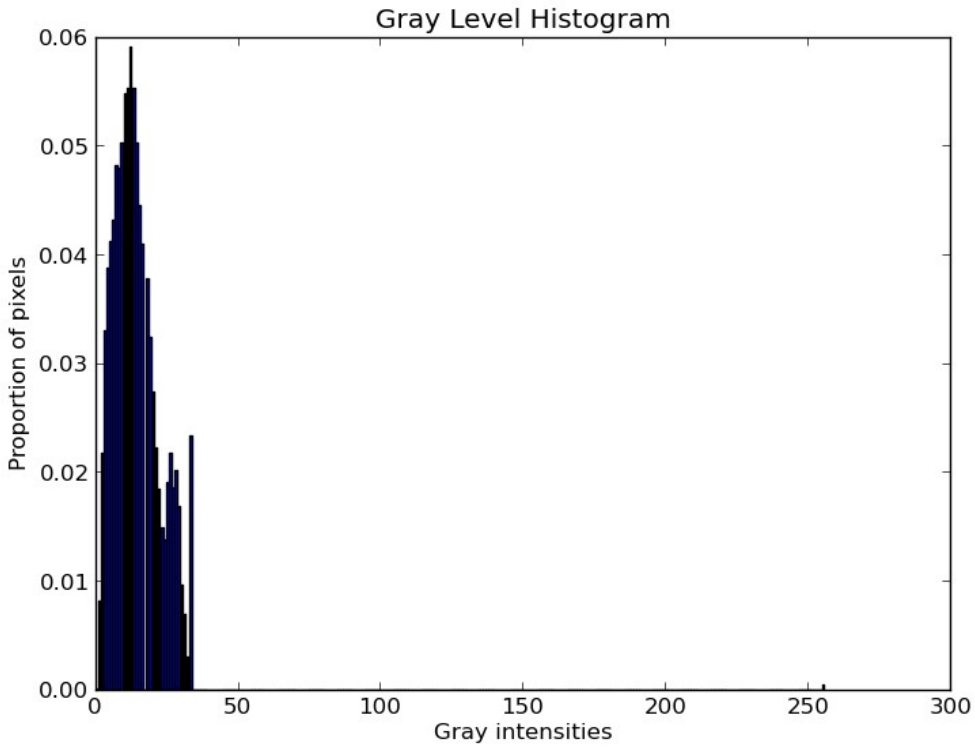
Programming Exercises:
Description of my equalization program (*source code is at end of document*):

I store the frequencies of each intensity, 0 -255, of the converted gray-scale image in an array, found by looping over the image's rows and columns and incrementing the bins of intensity values as I find them. I then create another array to store each intensity's probability density, and fill this array by iterating over the frequency array and dividing the value stored in each element by the total number of pixels in the image. I then proceed to store the transformation function in the form of an array by following equation 3.3-8 in the book, iterating the intensity probability density array one by one until I have computed every T(r_k). At the same time, I determine which integer value the current transformation value T(r_k) will map to and record how many pixels of the original image use that intensity assigned to r_k for later use in creating the equalized histogram (probability density function). Finally, I create a new image by iterating over every value in the old gray-scale image, looking up what that pixel's intensity value now maps to via the transformation lookup table, and store the associated value in the appropriate location in the new image.

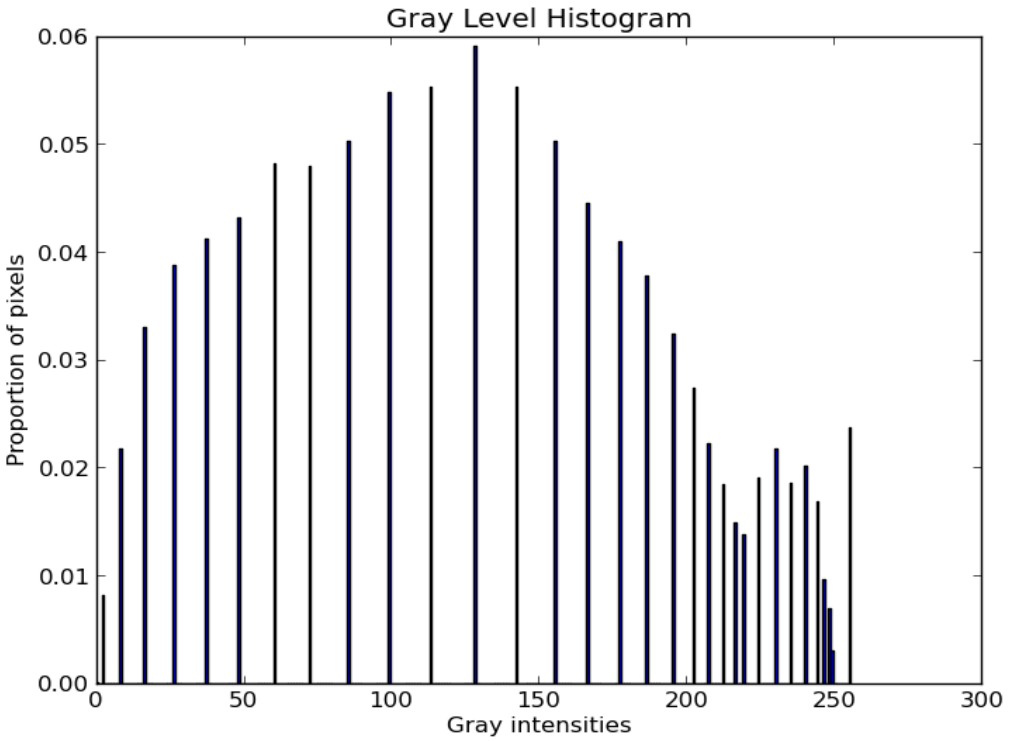
Image *mystery.png* before and after hist. Equalization:
Before:

After:

Scaled histograms of *mystery.png* before and after histogram equalization:
Before:



After:
Although theoretically it should be uniform, in reality this equalization is curved as shown below because of the binning problem. We cannot exactly assign output according the frequency of the input in this discrete case, and part of the problem is that non new intensity levels are allowed to be created. But the equalization does a good job in spreading out the parts of the input intensities where the most pixels are located, cleaning up the contrast of the image.



Lookup table used to equalize the image:

rk	sk
0	0.000000
1	2.000000
2	8.000000
3	16.000000
4	26.000000
5	37.000000
6	48.000000
7	60.000000
8	72.000000
9	85.000000
10	99.000000
11	113.000000
12	128.000000
13	142.000000
14	155.000000
15	166.000000
16	177.000000
17	177.000000
18	186.000000
19	195.000000
20	202.000000
21	207.000000
22	212.000000
23	216.000000
24	219.000000
25	224.000000
26	230.000000
27	235.000000
28	240.000000
29	244.000000
30	246.000000
31	248.000000
32	249.000000
33	255.000000
34	255.000000
35	255.000000
36	255.000000
37	255.000000
38	255.000000
39	255.000000
40	255.000000
41	255.000000
42	255.000000
43	255.000000
44	255.000000
45	255.000000
46	255.000000
47	255.000000
48	255.000000
49	255.000000
50	255.000000
51	255.000000
52	255.000000
53	255.000000
54	255.000000

55 | 255.000000
56 | 255.000000
57 | 255.000000
58 | 255.000000
59 | 255.000000
60 | 255.000000
61 | 255.000000
62 | 255.000000
63 | 255.000000
64 | 255.000000
65 | 255.000000
66 | 255.000000
67 | 255.000000
68 | 255.000000
69 | 255.000000
70 | 255.000000
71 | 255.000000
72 | 255.000000
73 | 255.000000
74 | 255.000000
75 | 255.000000
76 | 255.000000
77 | 255.000000
78 | 255.000000
79 | 255.000000
80 | 255.000000
81 | 255.000000
82 | 255.000000
83 | 255.000000
84 | 255.000000
85 | 255.000000
86 | 255.000000
87 | 255.000000
88 | 255.000000
89 | 255.000000
90 | 255.000000
91 | 255.000000
92 | 255.000000
93 | 255.000000
94 | 255.000000
95 | 255.000000
96 | 255.000000
97 | 255.000000
98 | 255.000000
99 | 255.000000
100 | 255.000000
101 | 255.000000
102 | 255.000000
103 | 255.000000
104 | 255.000000
105 | 255.000000
106 | 255.000000
107 | 255.000000
108 | 255.000000
109 | 255.000000
110 | 255.000000
111 | 255.000000
112 | 255.000000

113 | 255.000000
114 | 255.000000
115 | 255.000000
116 | 255.000000
117 | 255.000000
118 | 255.000000
119 | 255.000000
120 | 255.000000
121 | 255.000000
122 | 255.000000
123 | 255.000000
124 | 255.000000
125 | 255.000000
126 | 255.000000
127 | 255.000000
128 | 255.000000
129 | 255.000000
130 | 255.000000
131 | 255.000000
132 | 255.000000
133 | 255.000000
134 | 255.000000
135 | 255.000000
136 | 255.000000
137 | 255.000000
138 | 255.000000
139 | 255.000000
140 | 255.000000
141 | 255.000000
142 | 255.000000
143 | 255.000000
144 | 255.000000
145 | 255.000000
146 | 255.000000
147 | 255.000000
148 | 255.000000
149 | 255.000000
150 | 255.000000
151 | 255.000000
152 | 255.000000
153 | 255.000000
154 | 255.000000
155 | 255.000000
156 | 255.000000
157 | 255.000000
158 | 255.000000
159 | 255.000000
160 | 255.000000
161 | 255.000000
162 | 255.000000
163 | 255.000000
164 | 255.000000
165 | 255.000000
166 | 255.000000
167 | 255.000000
168 | 255.000000
169 | 255.000000
170 | 255.000000

171 | 255.000000
172 | 255.000000
173 | 255.000000
174 | 255.000000
175 | 255.000000
176 | 255.000000
177 | 255.000000
178 | 255.000000
179 | 255.000000
180 | 255.000000
181 | 255.000000
182 | 255.000000
183 | 255.000000
184 | 255.000000
185 | 255.000000
186 | 255.000000
187 | 255.000000
188 | 255.000000
189 | 255.000000
190 | 255.000000
191 | 255.000000
192 | 255.000000
193 | 255.000000
194 | 255.000000
195 | 255.000000
196 | 255.000000
197 | 255.000000
198 | 255.000000
199 | 255.000000
200 | 255.000000
201 | 255.000000
202 | 255.000000
203 | 255.000000
204 | 255.000000
205 | 255.000000
206 | 255.000000
207 | 255.000000
208 | 255.000000
209 | 255.000000
210 | 255.000000
211 | 255.000000
212 | 255.000000
213 | 255.000000
214 | 255.000000
215 | 255.000000
216 | 255.000000
217 | 255.000000
218 | 255.000000
219 | 255.000000
220 | 255.000000
221 | 255.000000
222 | 255.000000
223 | 255.000000
224 | 255.000000
225 | 255.000000
226 | 255.000000
227 | 255.000000
228 | 255.000000

```
229 | 255.000000
230 | 255.000000
231 | 255.000000
232 | 255.000000
233 | 255.000000
234 | 255.000000
235 | 255.000000
236 | 255.000000
237 | 255.000000
238 | 255.000000
239 | 255.000000
240 | 255.000000
241 | 255.000000
242 | 255.000000
243 | 255.000000
244 | 255.000000
245 | 255.000000
246 | 255.000000
247 | 255.000000
248 | 255.000000
249 | 255.000000
250 | 255.000000
251 | 255.000000
252 | 255.000000
253 | 255.000000
254 | 255.000000
255 | 255.000000
```

Time Spent:

Written: About 6 hours (includes reading the book beforehand)

Programming: Around 6 hours (this took so long because I encountered some trouble with Octave, and choosing not to use Matlab, I instead spent considerable time learning Numpy, PIL, and matplotlib). I think I'll be sticking with Python from now on as I feel like it's becoming less demystified to me. Also, the TA uses Python, so I feel like he's a good resource to turn to as I have questions.

Source Code (I used the TA's example code he sent out as the starting place to know how to open an image in Python and access it's data):

```
import numpy as np
import PIL.Image as pil
import matplotlib.pyplot as plt



def histogramEqualization():

    # load color image
    img = pil.open("mystery.png")
    imgData = img.getdata()
    imgPix = np.array(imgData)

    grayImgPix = np.around(np.dot(imgPix[..., :3],
        [0.299, 0.587, 0.144])).astype(int)
    # Store the frequencies for each intensity, 0 - 255
    grays = np.zeros(256, int)
    for r in range(grayImgPix.size):        # for each pixel in 1-d long array
            grayVal = grayImgPix[r]
            if (grayVal > 255):
                grayVal = 255
            grays[grayVal] = grays[grayVal] + 1
    # Store each intensity's probability density
    pdf = np.zeros(256, np.float64)
    numPixs = img.size[0] * img.size[1]
    for r in range(grays.size):
        pdf[r] = np.float64(grays[r]) / np.float64(numPixs)
    # assert summation pdf from 0 to r = 1.0--verified it does

    # Store the transformation function values, equalization histogram lookup
table
    transVals = np.zeros(256, np.float64)
    histSum = np.zeros(256)
    for r in range(pdf.size):
        prev = None
        if (r == 0.0):
            prev = 0.0
        else:
```

```
        prev = transVals[r - 1]
    transVals[r] = (255.0 * pdf[r]) + prev

    # find the rounded value of the current value we just computed
    rounded = int(round(transVals[r]))
    # be remembering what they map to to store in new histogram
    histSum[rounded] = histSum[rounded] + grays[r]

    #print (transVals[r])
    #print (rounded)
    #print (histSum[rounded])

# Store the equalized pdf
equalPdf = np.zeros(256)
a = 0.0
for r in range(equalPdf.size):
    equalPdf[r] = float(histSum[r] / float(numPixs))
    a += equalPdf[r]
# check to make sure = 1.0
#print (a)

plt.bar(list(range(transVals.size)), transVals)
plt.title("Transformation Function")
plt.xlabel("rk")
plt.ylabel("sk")
#plt.show()

plt.bar(list(range(equalPdf.size)), equalPdf)
plt.title("Gray Level Histogram")
plt.xlabel("Gray intensities")
plt.ylabel("Proportion of pixels")
plt.show()

# store it back in a new image
for r in range(grayImgPix.size):
    # print (transVals[grayImgPix[r]])
    grayVal = grayImgPix[r]
    if (grayVal > 255):  # sometimes 263 pops up...
```

```
            grayVal = 255
        grayImgPix[r] = transVals[grayVal]
    newGrayImg = pil.new("L", img.size)
    newGrayImg.putdata(grayImgPix)
    newGrayImg.show()
    newGrayImg.save("equalized-mystery.png")

    # print out transformation lookup table
    for c in range(transVals.size):
        print ("%i | %f" % (c, int(round(transVals[c]))))


def grayImgExp():
    # do stuff
    # load color image
    img = pil.open("mystery.png")
    imgData = img.getdata()
    imgPix = np.array(imgData)
    numPixs = img.size[0] * img.size[1]

    # make it grayscale, print out histogram manually
    grayImgPix = np.around(np.dot(imgPix[..., :3],
        [0.299, 0.587, 0.144])).astype(int)
    grays = np.zeros(256, int)
    for r in range(grayImgPix.size):        # for each pixel in 1-d long array
            grayVal = grayImgPix[r]
            if (grayVal > 255):
                grayVal = 255
            grays[grayVal] = grays[grayVal] + 1
    # probability distribution
    pdf = np.zeros(256, np.float64)
    for r in range(pdf.size):
        pdf[r] = np.float64(grays[r] / np.float64(numPixs))

    plt.bar(list(range(pdf.size)), pdf)
    plt.title("Gray Level Histogram")
    plt.xlabel("Gray intensities")
    plt.ylabel("Proportion of pixels")
```

```
    plt.show()

def main():
    grayImgExp()
    histogramEqualization()


if __name__ == "__main__":
    main()
```