SIMG-782 Introduction to Digital Image Processing
Homework 4 Solutions

1. Write pseudocode algorithm whose input is a pair of vectors h and g of length 256 representing the brightness count at 256 brightness levels and whose output is a point transform $s = T(r)$ that will convert an image with histogram h into one with histogram approximately g. You should provide a way to do this even if h and g do not come from the same size images.
Solution: Construct the cumulative sum expressions or tables for h and g and then scale one of them so that they both reach the same value. Which one is scaled is not important. After that, find matching indexes (s, r) for each $r: \ 0 \leq r \leq 255$ so that $C_h(r) = C_g(s)$. The table formed in this way constitutes the relationship and can be formed from the second part of this table. $s = T[r]$ is the desired matching function.

   (a) Compute cumulative histogram
   $$C_h(n) = \sum_{k=0}^{n} h(k)$$

   (b) Compute cumulative histogram
   $$C_g(n) = \sum_{k=0}^{n} g(k)$$

   (c) Scale
   $$C_g^*(n) = \frac{C_h(255)}{C_g(255)} C_g(n)$$

   (d) Initialize $T[0] = 0$. For each $r = 1, 2, \ldots, 255$ find $\{\text{Smallest } s : C_h(r) \leq C_g^*(s)\}$. Set $T[r] = s$.

2. Implement and test your algorithm by matching the brightness histogram for *EightAM* to the histogram for *Lena*. Your program should also create plots of the histograms of *EightAM* before the transformation, *EightAM* after the transformation, and the histogram of *Lena*.

```
A=read_image(imgpath+'EightAM.png')
B=read_image(imgpath+'lena.png')
;
;Show the images side-by-side
sa=size(A,/dim)
sb=size(B,/dim)
Window,/free,xsize=sa[0]+sb[0]+1,ysize=max([sa[1],sb[1]]),$
    title='Images A and B'
erase,255
TV,A
TV,B,sa[0]+1,0
;
;Get the histograms and cumulative histograms
ha=histogram(A,min=0,max=255)
cha=Total(ha,/cum)
hb=histogram(B,min=0,max=255)
chb=Total(hb,/cum)
```

```
chb=chb*Total(ha)/Total(hb)
T4=intarr(256)


For r=0,255 do T4[r]=min(where(chb GE cha[r]))>0
Am=T4[A] ;A matched to B
;
;Show A before and after.
Window,/free,xsize=2*sa[0]+1,ysize=sa[1],$
    title='Image A before and after'
erase,255
TV,A
TV,Am,sa[0]+1,0
;
;Plot the histograms
ham=histogram(Am,min=0,max=255)
cham=total(ham,/cum)
window,/free,xsize=600,ysize=600
!p.multi=[0,2,3] ;Display a 2x3 array of plots
tickv=indgen(9)*32
plot,ha,xtickv=tickv,xticks=8,xrange=[0,255],title='A Original'
plot,cha,xtickv=tickv,xticks=8,xrange=[0,255],title='A Original'
plot,hb,xtickv=tickv,xticks=8,xrange=[0,255],title='B Original'
plot,chb,xtickv=tickv,xticks=8,xrange=[0,255],title='B Original'
plot,ham,xtickv=tickv,xticks=8,xrange=[0,255],title='B Matched to A'
plot,cham,xtickv=tickv,xticks=8,xrange=[0,255],title='B Matched to A'
!P.Multi=0; Reset the display
;
;Show the cumulative histograms on the same axes
Window,/free,xsize=400,ysize=400,$
    title='Cumulative Histogram Comparisons'
plot,cha,xticks=8,xtickv=tickv,linestyle=1
oplot,chb
oplot,cham,linestyle=2,thick=2

window,/free,xsize=400,ysize=400,title='Transform Function'
plot,T4,xtickv=tickv,xticks=8,ytickv=tickv,$
    yticks=8,xticklen=1,yticklen=1
```

Original EightAM and Lena Images

EightAM before and after matching

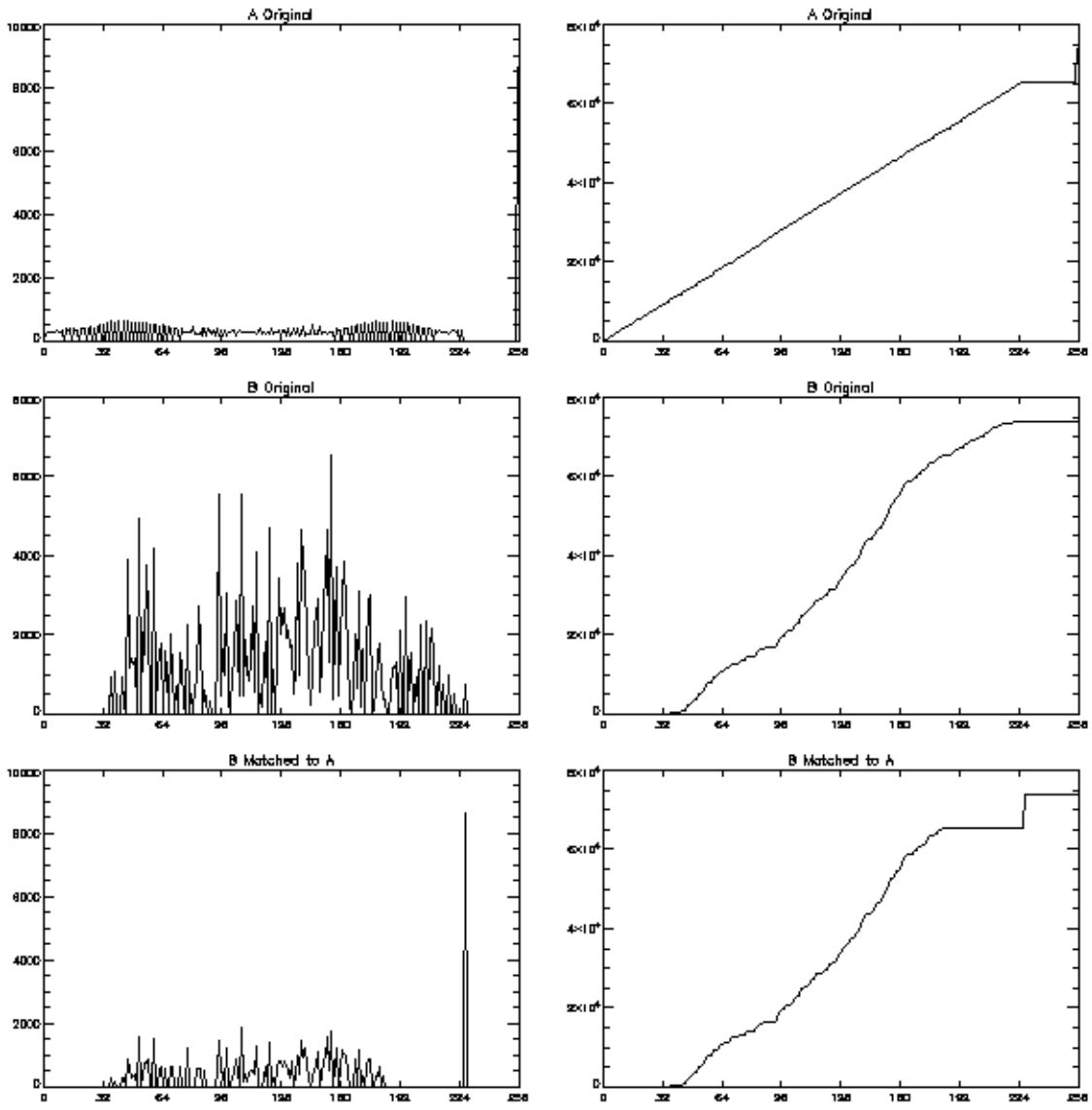
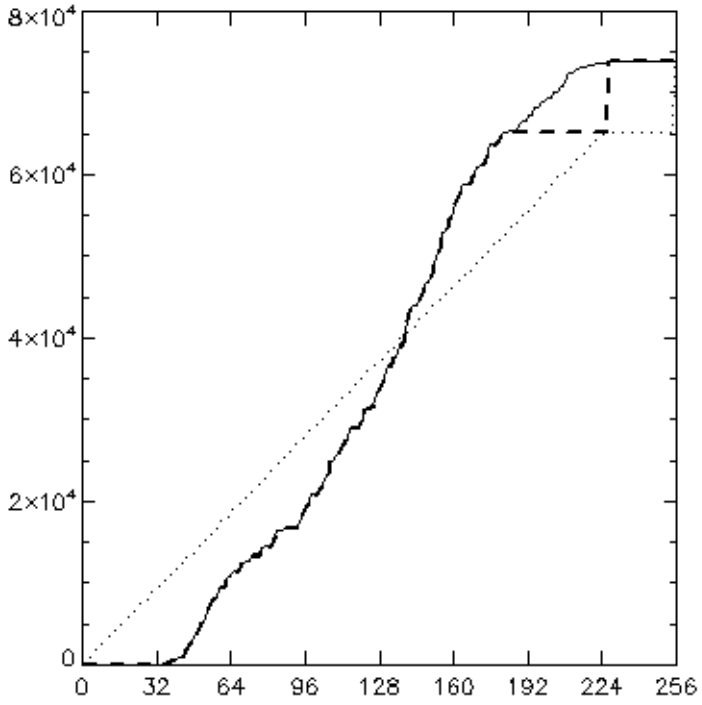


Image Histograms and Cumulative Histograms



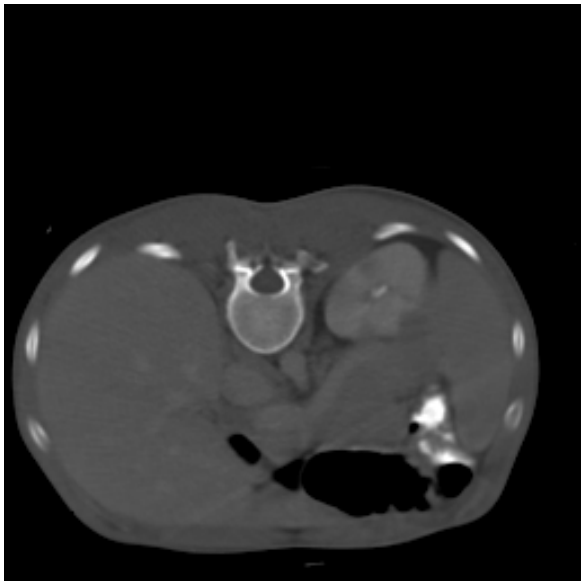
Cumulative Histogram Comparisons. The original EightAM is dotted, the original Lena is solid, and the matched EightAM is dashed. The cumulative EightAM histogram has been scaled to the same image size as the Lena image.

3. It is difficult to see detail in the ctscan.png image. It is also difficult to equalize because of the large area of black background. You can equalize the foreground image by separating it from the background, calculating the equalization transformation on the foreground pixels, and applying the transformation to the whole image. Construct an equalization of the foreground image.

   Solution: The foreground can be extracted from the dark background by a threshold operation, and then an equalization can be constructed from the histogram of the foreground pixels.

```
A=read_image('ctscan.png')
k=where(A GT 0)
h=histogram(A[k])
c=total(h,/cum)
T=c/max(c)*255
B=T[A]
```

   The results are shown below.



Original

Enhanced

Exactly the same result can be achieved by a shorter algorithm.

```
A=read_image('ctscan.png')
h=histogram(A)
h[0]=0
c=total(h,/cum)
T=c/max(c)*255
B=T[A]
```

By setting h[0]=0 before computing the cumulative histogram, the effect of the background pixels is zeroed out before the transform is calculated. This eliminates the need to separate the foreground from the image to calculate the histogram.

4. The image boy.png is an underexposed true color image. Construct a program to equalize the image brightness without creating serious changes in the colors.

   Solution The general idea is to stretch the brightness values of the image while keeping the hue and saturation constant. To get the brightness values, we can convert from RGB space to HSV space. The variable V, which has the range [0,1], represents brightness. There are many ways to stretch V. Here we will use simple histogram equalization. A gamma curve could also be employed.

```
A=read_image('boy.png')
help,A
;IDL PRINTS: A               BYTE      = Array[3, 512, 768]

;We need to get the three color planes for the COLOR_CONVERT procedure
;The REFORM function removes the dimension 1 from the extracted plane.
Ar=reform(A[0,*,*])
Ag=reform(A[1,*,*])
Ab=reform(A[2,*,*])
```

```
;Do the color space conversion
COLOR_CONVERT,Ar,Ag,Ab,H,S,V,/RGB_HSV

;Ge the histogram of V. Use a reasonably large number of bins.
;We don't have to use 255 since we are not going to make V
;into a displayed image.
hist=histogram(V,MIN=0,MAX=1,NBINS=1000)
c=total(hist,/cum)
;Expand the scale and do the mapping
T=c/max(c)*1000
W=T[1000*V]
;Compress the value scale back to [0,1]
W=W/1000.
;Convert back to RGB space
COLOR_CONVERT,H,S,W,Br,Bg,Bb,/HSV_RGB
B=bytarr(3,512,768)
B[0,*,*]=Br
B[1,*,*]=Bg
B[2,*,*]=Bb

disp_image,B,True=1
```

The original and enhanced images are shown in Figure **??**. The equalization of the brightness channel has been a little too strong. This could be improved by a less aggressive enhancement.

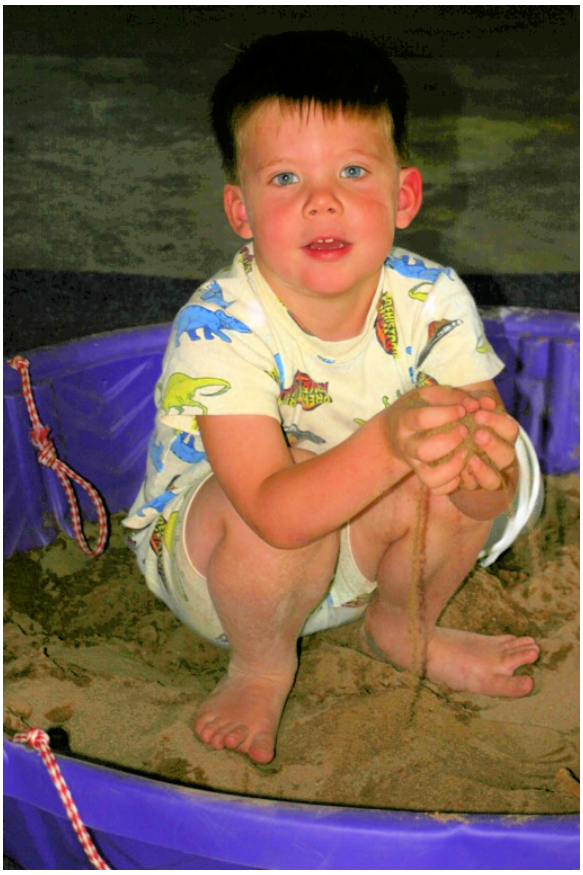
5. The images blobz1.png and blobz2.png (in the images area of the course web page) are shown in figure 1. The difference is that blobz1 has nearly uniform illumination while blobz2 has very nonuniform illumination. The goal of this problem is to construct an algorithm based on global greyscale thresholding for the segmentation of each image.

   (a) Construct an algorithm that will segment the blobz1.png image. This algorithm can use a global threshold that can be chosen by examination of the image histogram.

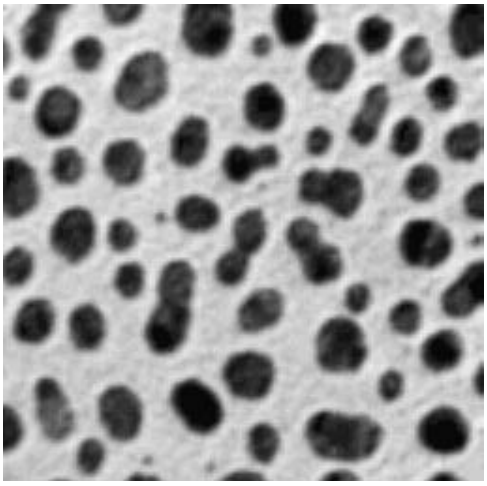
   Solution

   The histogram of the image is shown below. The effect of thresholding with $T = 130$ is shown in the image on the right. Because the illumination is uniform the histogram naturally divides into two relatively distinct regions. This image would be a reasonable candidate for automatic threshold selection.

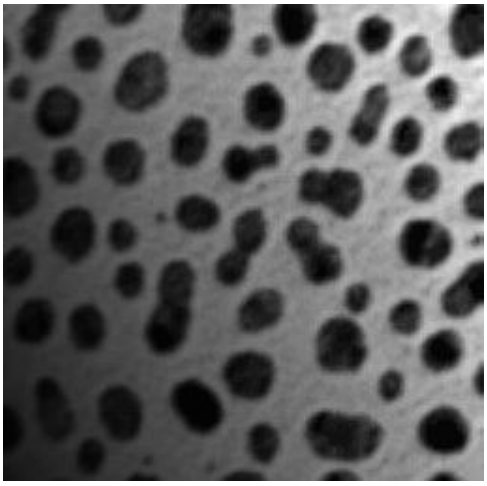
Original



Luminance Equalized

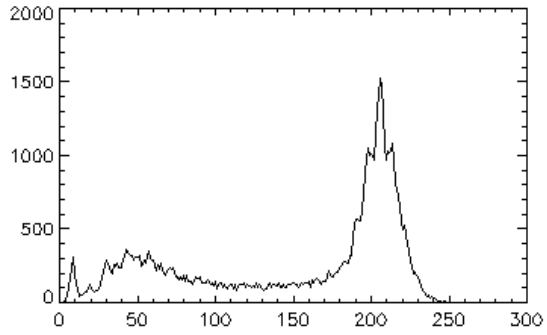


blobz1.png

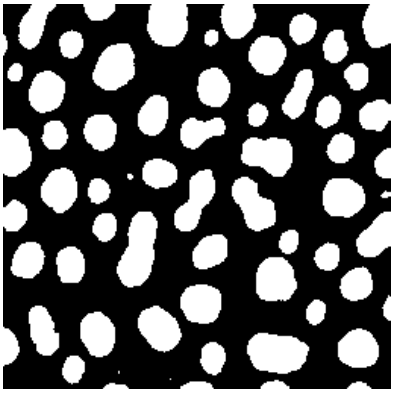


blobz2.png

Figure 1: Microscope images with uniform and nonuniform illumination.



blobz1 histogram



Segmented at $T = 130$

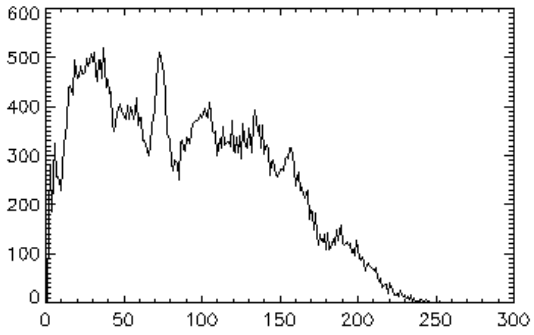
```
;Program for hw4(a)
Fa=read_image('D:\Harvey\Classes\SIMG782\2004\homework\blobz1.png')
sf=size(Fa,/dim)
window,11,xs=sf[0],ys=sf[1],xpos=0,ypos=0
tv,Fa

;Find and display the histogram
ha=histogram(Fa)
window,12,xs=400,ys=sf[1],xpos=2*(sf[0]+5),ypos=0
plot,ha

Ta=130
Ba=FA LT Ta
window,13,xs=sf[0],ys=sf[1],xpos=sf[0]+5,ypos=0
tvscl,Ba
```

(b) Construct an algorithm that will segment the blobz2.png image. This algorithm will require the determination of an illumination function followed by separation of a log-reflectance image

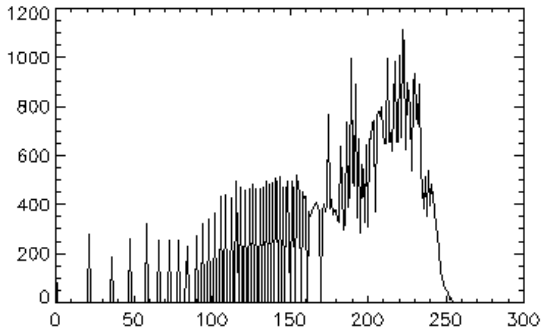
from the illumination. The log-reflectance should then be suitable for global thresholding. [*Note: you are not supposed to use the blobz1 image in the blobz2 algorithm, even though that would be very convenient.* :-)]

Solution

The image cannot be segmented by a global threshold. This can be seen by looking at the histogram, shown below. The log image is somewhat more uniform, but still cannot be segmented globally, as shown in the histogram below right.

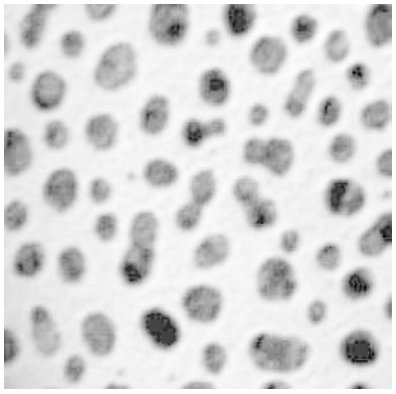


blobz2 histogram

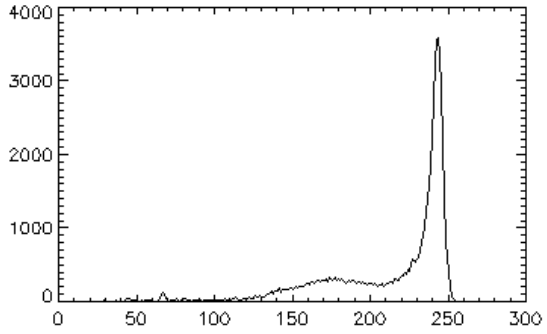


Histogram of log image

The image can be compensated by subtracting a function that is fitted to the background. The resulting compensated image and its histogram are shown below. Note that the histogram is now much "cleaner" and it is possible to separate the tall illumination peak from the rest of the image. A reasonable selection for the threshold level is $T = 210$.

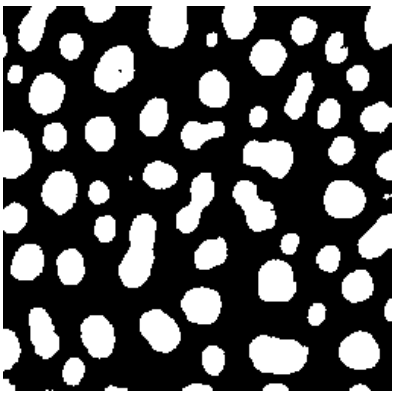


Compensated Image

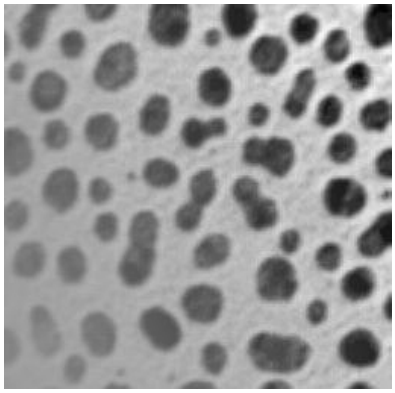


Histogram of compensated log image

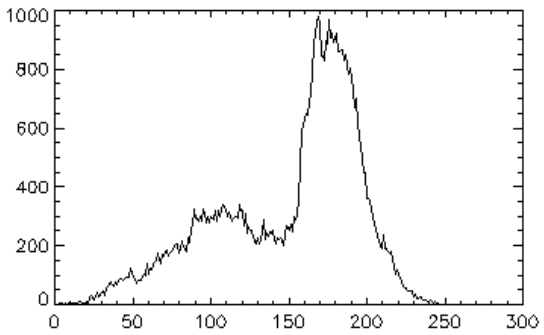
The thresholded image is shown at the right. This segmentation compares favorably with the segmentation of the illuminated image of part (a).



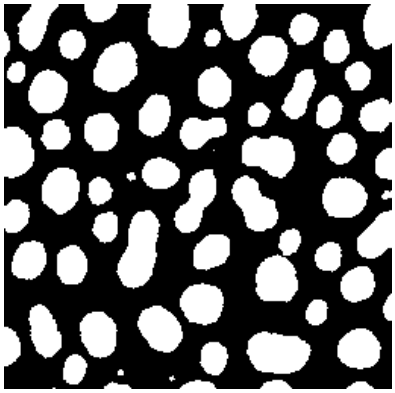
We may also try to compensate the original image and skip the step of forming the log image. We compute an illumination compensation and subtract it from the original image. The histogram of this image is shown below left. The compensated image and the segmented image with a threshold of $T = 130$ are shown below. Although the illumination peak in the histogram is wider, there is still a reasonable threshold to select.



Directly compensated



Histogram



Globally segmented $T = 150$

```
;Program for Prob 4(b)
;Open the blob image
F=read_image('blobz2.png')
sf=size(F,/dim)
window,0,xs=sf[0],ys=sf[1],xpos=0,ypos=0
tv,F

;Construct coordinate arrays for the image
x=findgen(sf[0])#replicate(1,sf[1])
y=findgen(sf[1])##replicate(1,sf[0])

;Construct the moments used to solve for the a coefficients
mx=mean(x) & my=mean(y) & mxy=mean(x*y)
mx2=mean(x^2) & my2=mean(y^2) & mx2y=mean(x^2*y)
mxy2=mean(x*y^2) & mx2y2=mean(x^2*y^2)

;Solve for the coefficients
C=[[1,mx,my,mxy],[mx,mx2,mxy,mx2y],[my,mxy,my2,mxy2],[mxy,mx2y,mxy2,mx2y2]]
CI=invert(C)

;Compute the log image
FL=BYTSCL(ALOG(F))
window,1,xs=sf[0],ys=sf[1],xpos=sf[0]+5,ypos=0
```

```
TV,FL

;Compute the histogram of log image
hLog=histogram(FL)
window,10,xs=400,ys=sf[1],xpos=2*(sf[0]+5),ypos=sf[0]+20,title='Histogram of Log Image'
plot,hLog

;Construct an illumination predictor for FL
;We can use the same x and y coordinate arrays
mL=mean(FL) & mLx=mean(FL*x) & mLy=mean(FL*y) & mLxy=mean(FL*x*y)
w=transpose([mL,mLx,mLy,mLxy])
aL=CI##w
GL=aL[0]+aL[1]*x+aL[2]*y+aL[3]*x*y
BL=bytscl(FL-GL)
window,2,xs=sf[0],ys=sf[1],xpos=2*(sf[0]+5),ypos=0
TVSCL,GL
window,3,xs=sf[0],ys=sf[1],xpos=0,ypos=(sf[1]+25)
TV,BL

hL=histogram(BL)
window,4,xs=400,ys=sf[1],xpos=3*(sf[0]+5),ypos=sf[1]+25,$
    title='Histogram of Compensated Log image'
plot,hL

TL=210
BLT=BL LE TL
window,5,xs=sf[0],ys=sf[1],xpos=(sf[0]+5),ypos=(sf[1]+25)
tvscl,BLT

hf=histogram(F)
window,6,xs=400,ys=sf[1],xpos=3*(sf[0]+5),ypos=0,$
    title='Original Histogram'
plot,hf


;Construct an illumination predictor for F itself
;to see how it would work to not do the log first.
;We can use the same x and y coordinate arrays
m1=mean(F) & m1x=mean(F*x) & m1y=mean(F*y) & m1xy=mean(F*x*y)
w=transpose([m1,m1x,m1y,m1xy])
a1=CI##w
G1=a1[0]+a1[1]*x+a1[2]*y+a1[3]*x*y
B1=bytscl(F-G1)
window,7,xs=sf[0],ys=sf[1],xpos=0,ypos=2*(sf[1]+25)
TVSCL,G1
window,8,xs=sf[0],ys=sf[1],xpos=(sf[0]+5),ypos=2*(sf[1]+25)
TV,B1
```

```
T1=150
B1T=B1 LT T1
window,9,xs=sf[0],ys=sf[1],xpos=2*(sf[0]+5),ypos=2*(sf[1]+25)
tvscl,B1T

h1=histogram(B1)
window,14,xs=400,ys=sf[1],xpos=3*(sf[0]+5),ypos=2*(sf[1]+25)
plot,h1
```