AI728: Assignment
Digital Image Processing

Due on March 31, 2016

Dr. Arun Agarwal

Submitted By

Suman Bhukar(15MCMT29)
Mohan Reddy(15MCMT10)
Rosni Kv(15MCMI15)

Contents

Problem 1

Take the hubble image and count number of large stars and colour them with different colours based on their sizes.

Method 1

The listing below shows a MATLAB script(main.m) that was used to generate the required output image. In this main program we used the function colour which is defined seperately. Figure 1 shows the input image we used.

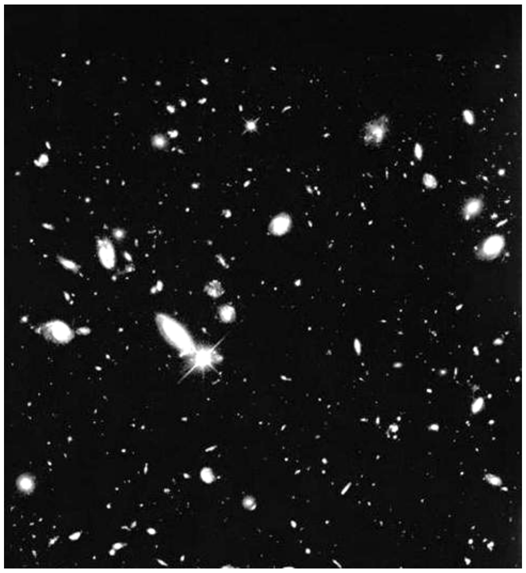

Figure 1: Input image

```
%Assignment Part 1
input=imread('/home/freestyler/Courses/test/hubble.tif');
[number,I_new,I_imadjust,RGB_label]=colour(input);
disp('The number of large stars are') ;disp(number)
figure,imshow(I_new),title('coloured hubble image');
imwrite(I_new,'Colured_hubble_image.png');
```

Below Code shows the procedure colour used in previous listing. Since the input is gray scale image, It is converted to RGB format (explained in lines 10 - 20). Then we are correcting the non uniform illumination for enhancement purpose. Next created a new binary image by thresholding the adjusted image. Remove background noise with **bwareaopen** (code section 30).

From the binary image we are finding the connected components using **bwconncomp** (code section 35). *CC = bwconncomp(BW)* returns the connected components CC found in BW. The binary image BW can have any dimension. CC is a structure with four fields.

Connectivity [*Connectivity of the connected components (objects)*], **ImageSize** [*Size of BW*], **NumObjects** [*Number of connected components (objects) in BW*],**PixelIdxList** [*1-by-NumObjects cell array where the kth element in the cell array is a vector containing the linear indices of the pixels in the kth object.*].

From the structure field NumObjects we return the number of objects. That is **counted the number of large stars in the image(30 in this example).** Using the built in function **regionprops** also we can

Problem 1 continued on next page. . .

measure the properties of image regions (code section 40). The variable N_object has large stars count. We can colour the objects with different colour using **label2rgb** method. That is shown in Figure 2.

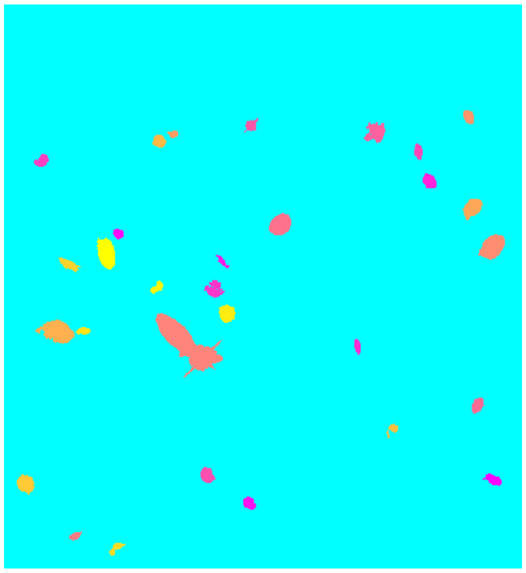

Figure 2: label2rgb method

In Code section 55, we set the threshold values, based on that we are accessing different size stars. Then colour the different stars based on their area. Colouring is explained in Code section 60 -75, and it is done by changing pixel by pixel values. The coloured hubble image is shown in Figure 3.

```

function [number,I_new,I_imadjust,RGB_label] = colour(I1)

figure, imshow(I1), title('original image');
5
%%%contrast gray scale image
I_imadjust=imadjust(I1);
figure, imshow(I_imadjust), title('contrast of image');

10 %%%conversion of gray to rgb 3d format
size(I_imadjust,3);
[m,n,r]=size(I_imadjust);
rgb=zeros(m,n,3);
rgb(:,:,1)=I_imadjust;
15 rgb(:,:,2)=rgb(:,:,1);
rgb(:,:,3)=rgb(:,:,1);
I_new=rgb/255;
figure,imshow(I_new),title('converted rgb');
imwrite(I_new,'gray_to_rgb.png');
20
%%%correcting non uniform illumination
background = imopen(I1,strel('disk',15));
I2 = I1 - background;
I3=imadjust(I2);
25 figure, imshow(I3), title('non uniform illumination corrected');
```

```
%Create a new binary image by thresholding the adjusted image. Remove background
%noise with bwareaopen.
level = graythresh(I3);
bw = im2bw(I3,level);
bw = bwareaopen(bw, 50);

%number of objects in the image
cc = bwconncomp(bw);
number  = cc.NumObjects;

%number ofobjects in image using regionprops
s  = regionprops(bw, 'Area');
N_objects = numel(s);

%Colouring using label2rgb method, objects are displayed with different
%colours
labeled=labelmatrix(cc);
RGB_label = label2rgb(labeled, @spring, 'c', 'shuffle');
imshow(RGB_label,'InitialMagnification','fit')
imwrite(RGB_label,'labeled.png');

for n=1:number
    so(n)=numel(cc.PixelIdxList{n});
end
so=sort(so);

%set the threshold
th0=43;
th1=92;
th2=150;
th3=1373;

%colour the stars based on their sizes manually
for i=1:number
    if ((th0<= numel(cc.PixelIdxList{1,i})) & (numel(cc.PixelIdxList{1,i}) <= th1))
        for j=1:numel(cc.PixelIdxList{1,i})
            I_new(cc.PixelIdxList{1,i}(j))=.173;
        end
    end
    if ((th1 < numel(cc.PixelIdxList{1,i})) & (numel(cc.PixelIdxList{1,i}) <= th2))
        for j=1:numel(cc.PixelIdxList{1,i})
            I_new(cc.PixelIdxList{1,i}(j))=2.56;
        end
    end
    if ((th2 < numel(cc.PixelIdxList{1,i})) & (numel(cc.PixelIdxList{1,i}) <= th3))
        for j=1:numel(cc.PixelIdxList{1,i})
            I_new(cc.PixelIdxList{1,i}(j))=0.2;
        end
    end
end
figure,imshow(I_new),title('Coloured hubble image');
end
```

Problem 1 continued on next page. . .

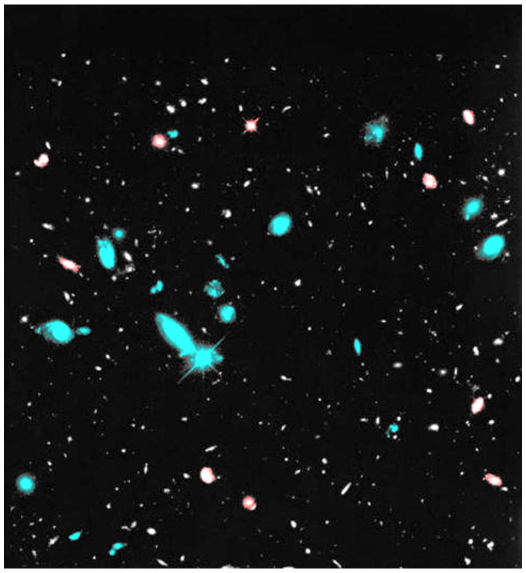

Figure 3: Final output:Coloured Hubble image

Method 2

Count the number of large stars and colouring them is acheived through another method, which is described in this section. The code below shows the matlab procedure (code.m).

The input image shown in Figure 1 is converted into binary image. Code section 5-10 make use of **regionprops** function and count the number of stars(here the count is 358). By using RGB color table we can display the image and activate color map. Colouring is done based on the relative area. Code section 25 is used to display the image on black background. The final output image is shown in Figure 4.

```
   
   I=imread('/home/freestyler/Courses/test/gray_to_rgb.png');
   
   %%% convert to binary image
5  bw = im2bw(I, graythresh(I));
   
   %%% Use regionprops to get objects area and count
   [labelBW,no_obj] = bwlabel(bw);
   reginfo = regionprops(logical(bw), 'Area');
10 disp('The number of large stars are') ;disp(no_obj)
   
   %%% Finding relative areas
   all_area = [reginfo.Area];
   rel_area = all_area(:) ./ max(all_area);
15 
   %%% code the relative area as being the hue.To prevent color confusion, we
   %%% limit the upper value of the hue.
   hue = rel_area * 0.85;
   hsvtab = [hue, ones(size(hue,1),2)];
20 rgbtab = hsv2rgb(hsvtab);
   
   %%%we have an RGB color table we can display the image and activate the colormap
   image(labelBW);
   colormap(rgbtab);
```

```

%%% display on a black background
map = colormap;
map(1,:) = [0,0,0];
colormap(map)
```

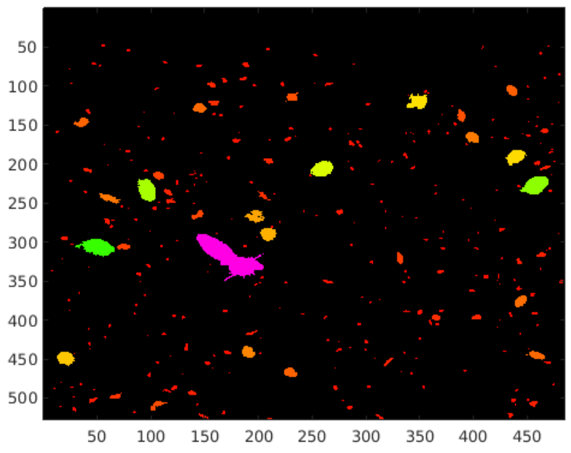

Figure 4: Final output2:Coloured Hubble image

Problem 2

Find stars whose area is greater than some threshold and color it RED.

The listing below shows a MATLAB script(main2.m) that was used to generate the required output image. In this main program we used the function **red** which is defined seperately. Figure 1 shows the input image we used.

```
  %Assignment Part 2
  input=imread('/home/freestyler/Courses/test/hubble.tif');
  [I_new1]=red(input);
  figure,imshow(I_new1),title('Red coloured hubble image');
5 imwrite(I_new1,'Red_Colured_hubble_image.png');
```

Below Code shows the procedure **red** used in previous listing. Since the input is gray scale image, It is converted to RGB format (explained in lines 5-10). Then we are correcting the non uniform illumination for enhancement purpose. Next created a new binary image by thresholding the adjusted image. Remove background noise with **bwareaopen** (code section 25).

All the procedures are same as we did in Problem 1. The output image obtained from this procedure is shown in Figure 5. That is RED coloured the big stars using the user defined threshold value.

```
   function [I_new1]=red(I11)

   %conversion gray to rgb
   size(I11,3);
5  [m,n,r]=size(I11);
   rgb=zeros(m,n,3);
   rgb(:,:,1)=I11;
   rgb(:,:,2)=rgb(:,:,1);
   rgb(:,:,3)=rgb(:,:,1);
10 I_new1=rgb/255;
   %%figure,imshow(I_new1),title('converted rgb');

   %%%contrast gray scale image
   I_imadjust=imadjust(I11);
15
   %%%correcting non uniform illumination
   background = imopen(I11,strel('disk',15));
   I21 = I11 - background;
   I31=imadjust(I21);
20
   %Create a new binary image by thresholding the adjusted image. Remove background
   %noise with bwareaopen.
   level1 = graythresh(I31);
   bw1 = im2bw(I31,level1);
25 bw1 = bwareaopen(bw1, 50);

   %number of objects in the image
   cc2 = bwconncomp(bw1);
   number2  = cc2.NumObjects;
30
   for n=1:30
       so(n)=numel(cc2.PixelIdxList{n});
   end
```

```matlab
so=sort(so);
th1=92;
th2=150;
th3=1373;

%colour the stars those have above thresold with 'red color'

for i=1:30
    if (numel(cc2.PixelIdxList{1,i}) <= th1)
        for j=1:numel(cc2.PixelIdxList{1,i})
            I_new1(cc2.PixelIdxList{1,i}(j))=255;
        end
    end
    if ((th1 < numel(cc2.PixelIdxList{1,i})) & (numel(cc2.PixelIdxList{1,i}) <= th2))
        for j=1:numel(cc2.PixelIdxList{1,i})
            I_new1(cc2.PixelIdxList{1,i}(j))=255;
        end
    end
    if ((th2 < numel(cc2.PixelIdxList{1,i})) & (numel(cc2.PixelIdxList{1,i}) <= th3))
        for j=1:numel(cc2.PixelIdxList{1,i})
            I_new1(cc2.PixelIdxList{1,i}(j))=1;
        end
    end
end

figure,imshow(I_new1),title('Red Coloured hubble image');
%%%imwrite(I_new1,'Red_coloured_hubble.png');

end
```

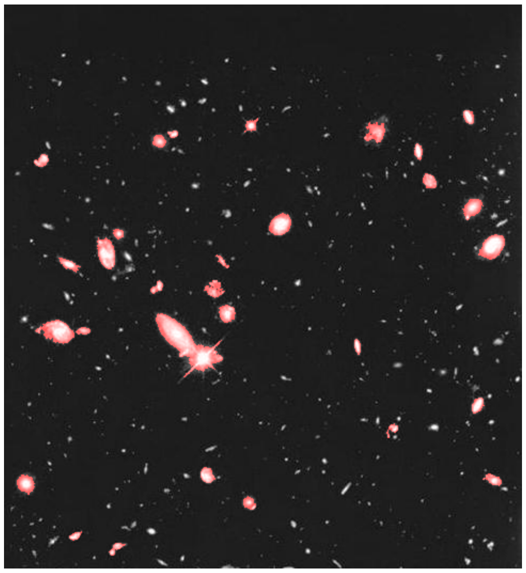

Figure 5: Final output:Red Coloured Hubble image

□