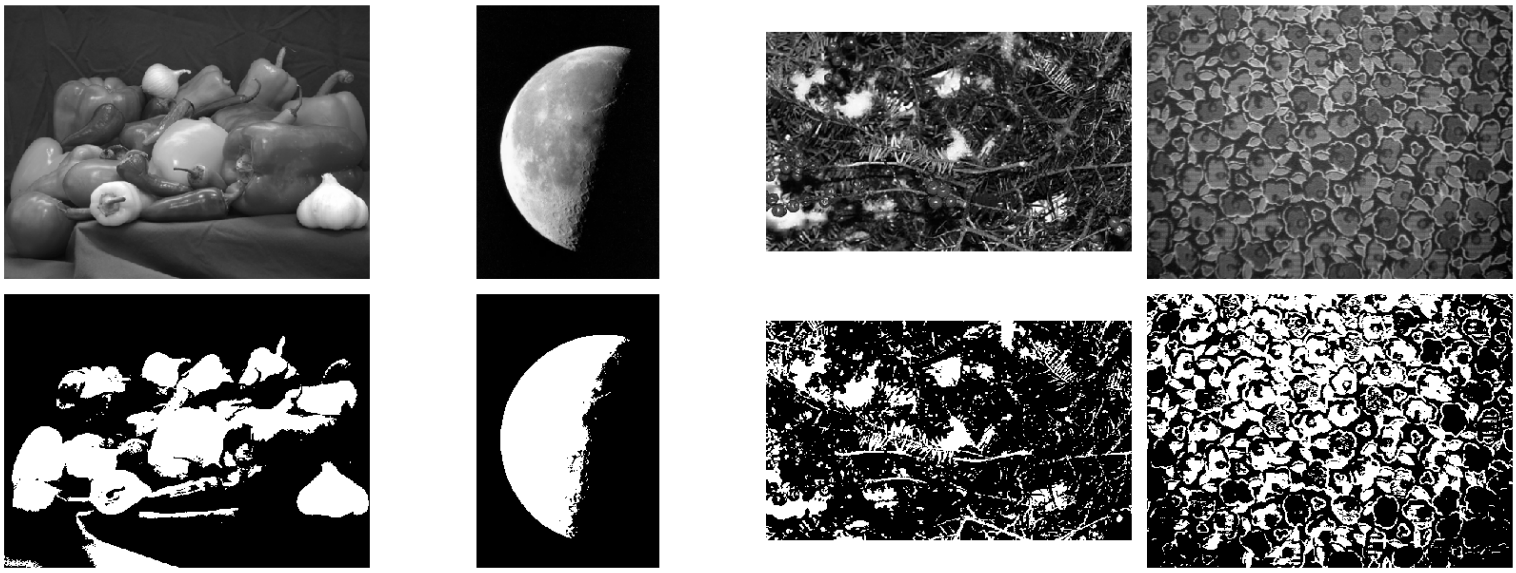
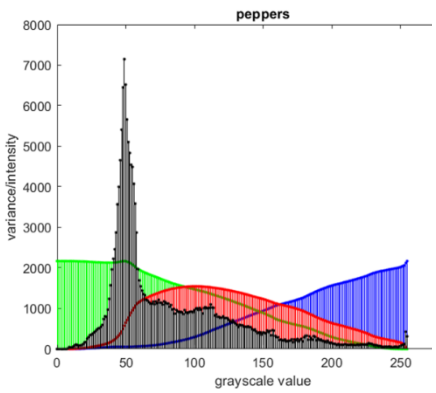
Lab 2 Report

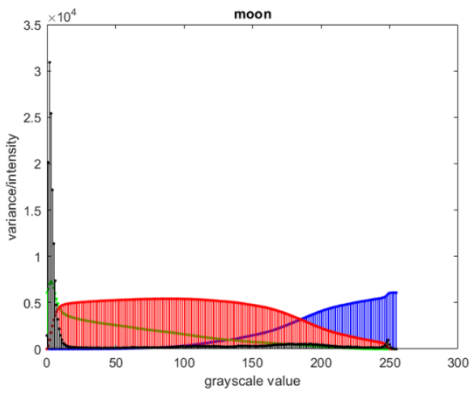
Name: Grant Beatty SID: 862037946

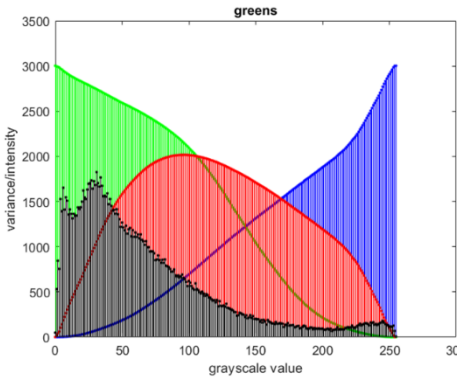
EE146 Section (022)

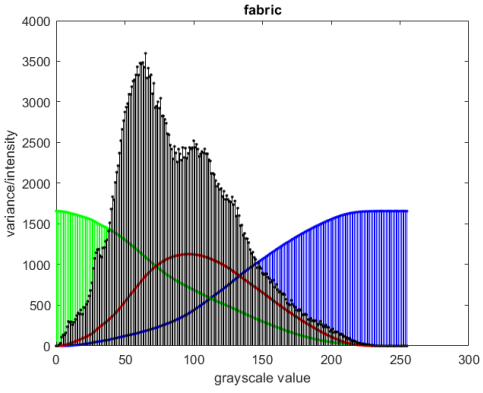
1&2) (peppers, moon, greens, fabric)











= between-class variance

= within-background variance

= within-foreground variance

= histogram

3) The Otsu algorithm works best for images with strong details or patterns. Textured images retain their central pattern. Regular images of things tend to lose their form and become less recognizable.

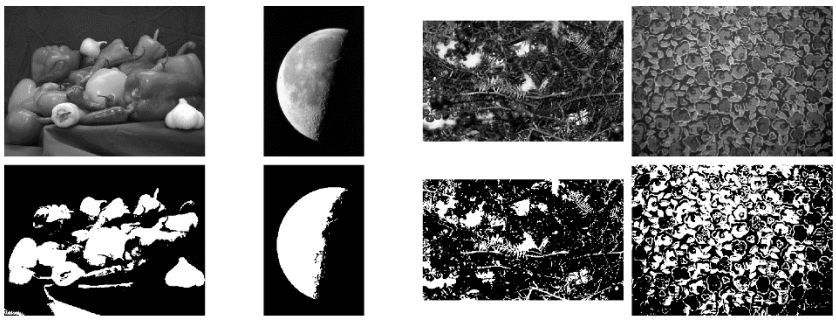
MATLAB Code

```
clear all;
close all;
```

Question 1

1)

```
pic1=imread(['peppers.png']);
pic1=rgb2gray(pic1);
pic2=imread(['moon.tif']);
pic3=imread(['greens.jpg']);
pic3=rgb2gray(pic3);
pic4=imread(['fabric.png']);
pic4=rgb2gray(pic4);


bpic1=imbinarize(pic1);
bpic2=imbinarize(pic2);
bpic3=imbinarize(pic3);
bpic4=imbinarize(pic4);



montage({pic1,pic2,pic3,pic4,bpic1,bpic2,bpic3,bpic4},'size',[2
4],'BackgroundColor',[1 1 1],'BorderSize',8)
```



2)

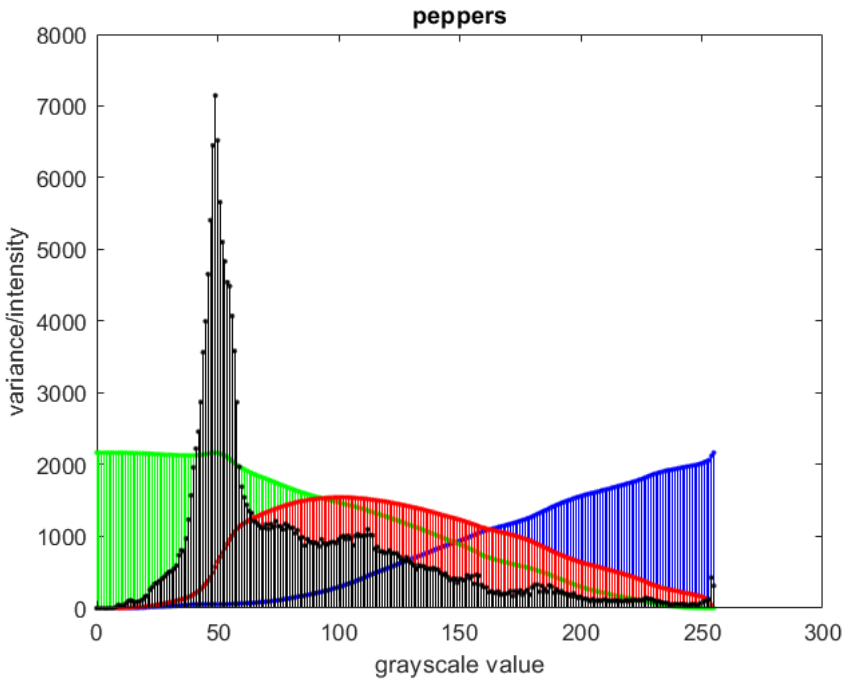
```
graph_name=["peppers","moon","greens","fabric"];
combvar=cell(1,4);
```

```
 listofpictures=cell(1,4);
 listofpictures{1}=pic1;
 listofpictures{2}=pic2;
 listofpictures{3}=pic3;
 listofpictures{4}=pic4;
 for io=1:4
 Gpeppers=listofpictures{io};

 h=imhist(Gpeppers);
 MN=sum(imhist(Gpeppers));
 ch=zeros(1,256);
 for j=1:256
     if (j>1)
 ch(j)=ch(j-1)+h(j);          %ch (cumulative histogram)
 else
 ch(j)=h(j);
     end                      % j represents the Color Scale Value(csv)+1.
(j=csv+1)
     if(ch(j)<(MN/2))         % If ch(j)=(MN/2) the line wont run but j=csv+1 so
we take j.
     median1=j;               % This way we know that the cumulative
     end                      % histogram at the last j value is the
                              % lowest histogram value greater than or equal to
MN/2.
 end
 median1
 A=0;
 for j=1:256                                %
     A=A+(j-1)*h(j);                        % This is the algorithm for A
 end                                        %

 B=0;                                       %
 for j=1:256                                % This is the algorithm for B
     B=B+(j-1)*(j-1)*h(j);                  %
 end

 mew=A/MN %mean
 q2=(1/MN)*(B-(1/MN)*A^2) %variance

 x=0:255;

 cA=zeros(256,1);
 for j=1:256
     if(j>1)
     cA(j)=cA(j-1)+(j-1)*h(j);                      % This is the algorithm
for A at pionts (0-255)
```

```
    else
    cA(j)=(j-1)*h(j);
    end
end


foreground_avg=cA./transpose(ch);
foreground_avg(isnan(foreground_avg))=0;

A1=A-cA;
ch1=MN-ch;
background_avg=A1./transpose(ch1);

cB=zeros(256,1);
for j=1:256
    if(j>1)
    cB(j)=cB(j-1)+(j-1)*(j-1)*h(j);                        % This is the
algorithm for B at pionts (0-255)
    else
    cB(j)=(j-1)*(j-1)*h(j);
    end
end

square_foreground_avg=cB./transpose(ch);
square_foreground_avg(isnan(square_foreground_avg))=0;

B1=B-cB;
square_background_avg=B1./transpose(ch1);


foreground_variance=square_foreground_avg-foreground_avg.^2;
background_variance=square_background_avg-background_avg.^2;
background_variance(isnan(background_variance))=0;


p0=transpose(ch/MN);
p1=transpose(ch1/MN);

between_variance= p0.*(foreground_avg-mew).^2+p1.*(background_avg-mew).^2;
within_variance=p0.*foreground_variance+p1.*background_variance;
variance_total=between_variance+within_variance;
```

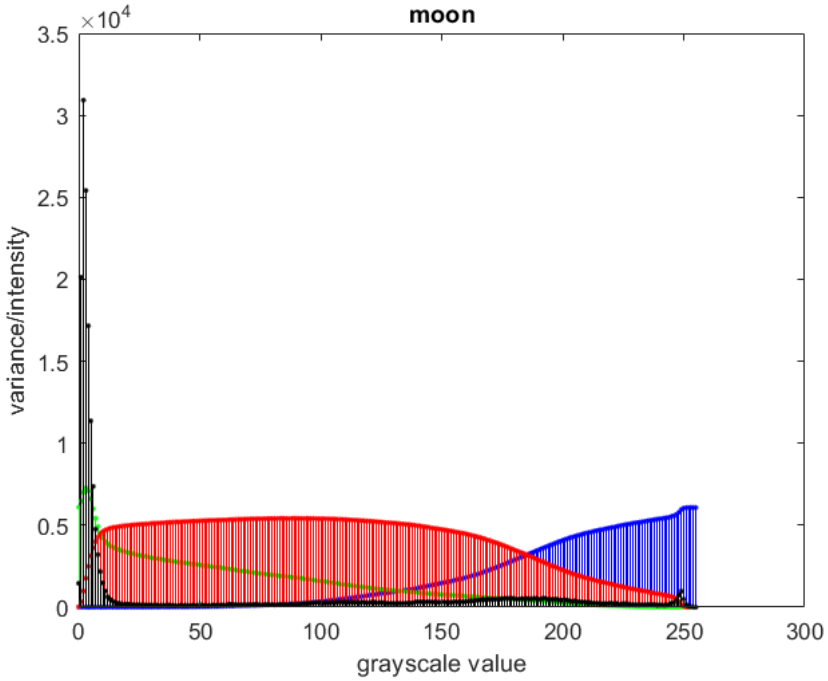
```
combvar{io}=[foreground_variance background_variance between_variance h];
figure
RGBPLOT=stem(x,combvar{io},'.');
set(RGBPLOT, {'color'}, {[0 0 1]; [0 1 0]; [1 0 0]; [0 0 0]});
xlabel('grayscale value')
ylabel('variance/intensity')
title(graph_name(io))



for j=1:256
if(between_variance(j)==max(between_variance))
    thresh=j-1
end
end
disp("Check Answer")
binhistogram=imhist(im2bw(Gpeppers,(thresh/255)))
checkanshist=imhist(imbinarize(Gpeppers))
end
```

```
median1 = 60
mew = 81.6594
q2 = 2.1692e+03
```

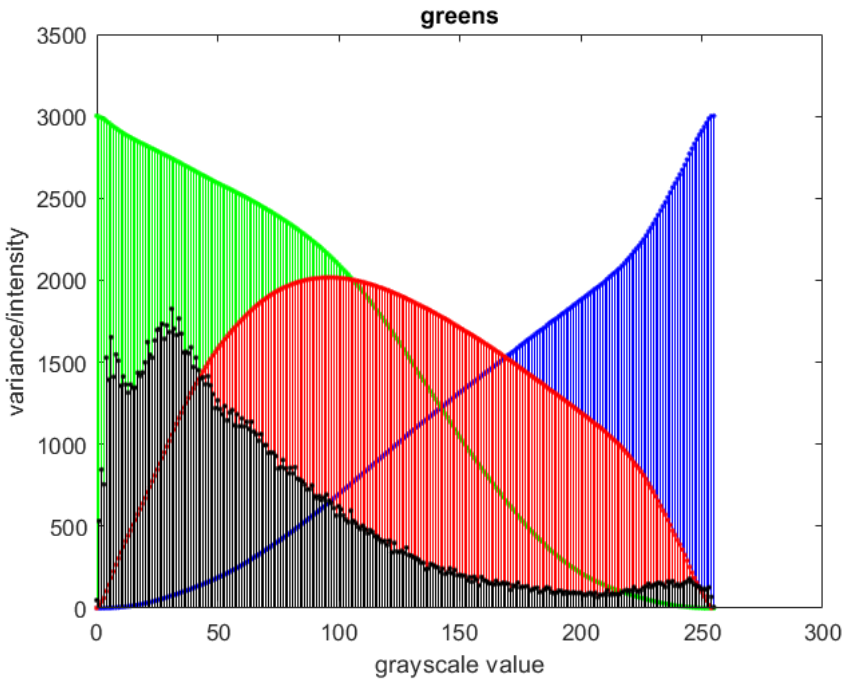



```
thresh = 101
Check Answer
binhistogram = 2×1
       142636
        53972
```

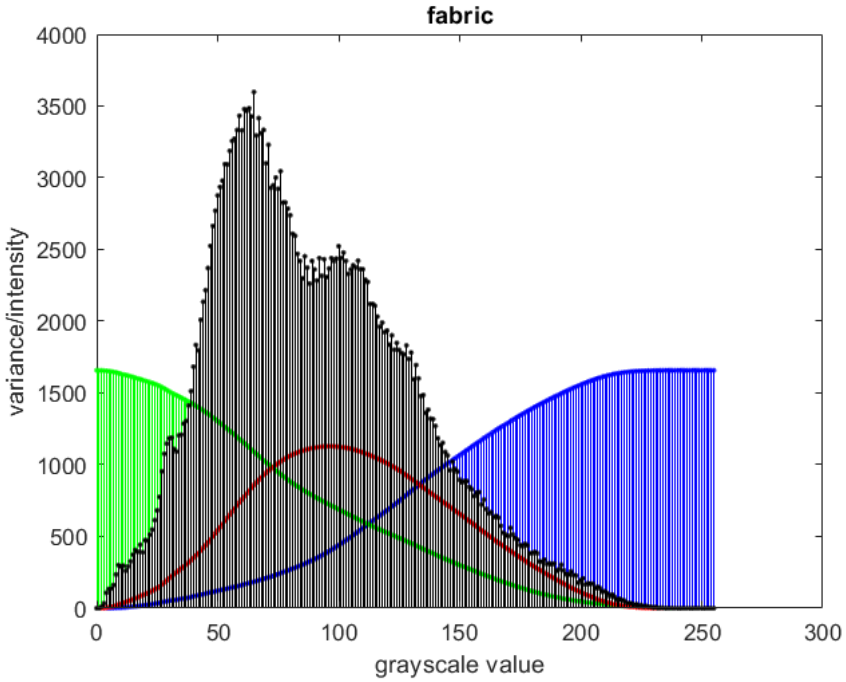
```
checkanshist = 2×1
       142636
        53972
median1 = 5
mew = 52.9154
q2 = 6.0845e+03
```




```
thresh = 89
Check Answer
binhistogram = 2×1
       139134
        53112
checkanshist = 2×1
       139134
        53112
median1 = 52
mew = 67.0060
q2 = 3.0029e+03
```




```
thresh = 96
Check Answer
binhistogram = 2×1
       116204
        33796
checkanshist = 2×1
       116204
        33796
median1 = 84
mew = 90.1428
q2 = 1.6580e+03
```




```
thresh = 96
Check Answer
binhistogram = 2×1
       182754
       124446
checkanshist = 2×1
       182754
       124446
```

The Otsu algorithm works best for images with strong details. Textured images retain their central pattern. Regular images of things tend to lose their form and become less recognizable.