Abelardo Riojas
ISC 4221C Professor Quaife
Fall 2019

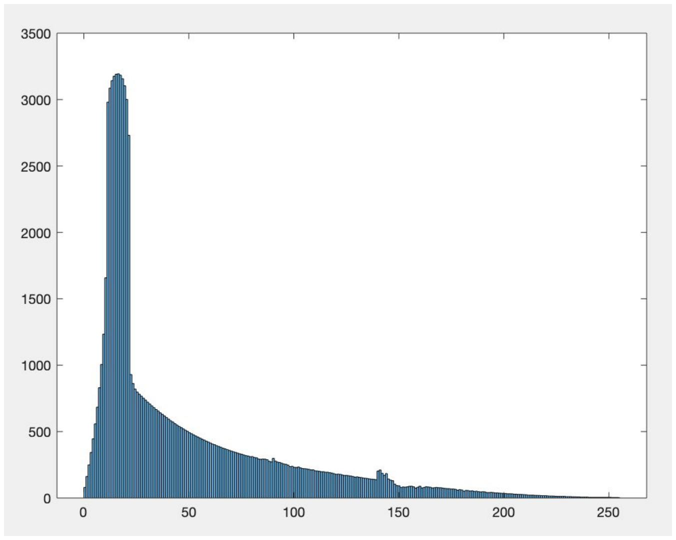
Lab 5 Report

1. Contrast Stretching
   i. Read in the image file and display the image.

```
mona = imread("mona_lisa.png");
imshow(mona);
```

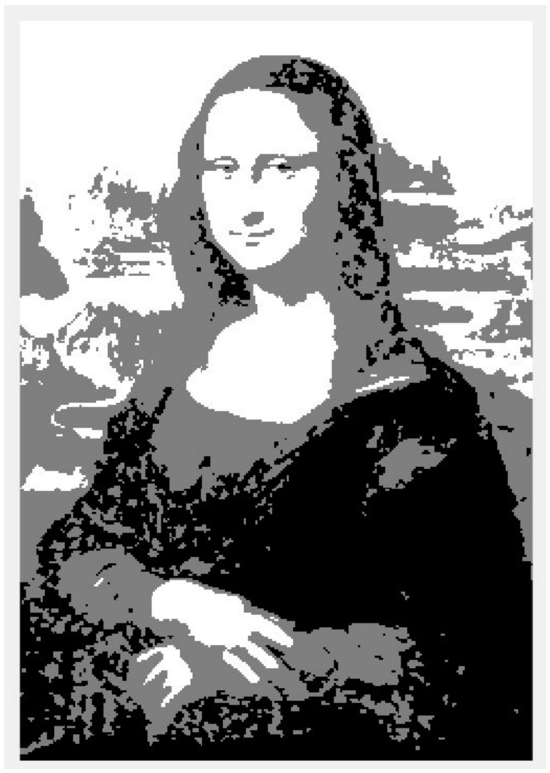
   ii. Make a histogram of the image with 256 bins.

```
m1 = double(mona);
m1 = m1(:);
histogram(m1,256);
```



iii. Divide the range [0, 255] into three ranges with about the same number of pixels (based on the histogram). Reset pixels in the first range to 0, the middle range to 127, and the third range to 255. Display the 3-level image.

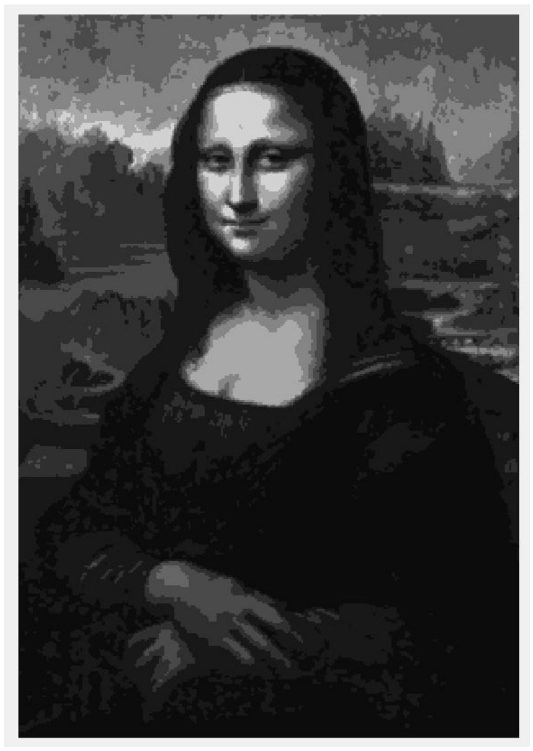
```
m3 = m1; %copy

t1 = 18;
t2 = 50;

i1 = m3 < t1;
i2 = t1 <= m3 & m3 < t2;
i3 = t2 <= m3;

m3(i1) = 0;
m3(i2) = 127;
m3(i3) = 255;
m4 = imshow(uint8(m3));
```



iv. Apply k-means with K = 8 representative gray levels. You may use a built-in k-means code or the one you developed in Lab 4. Display the image.

```
[m, n] = size (mona);
p = double (mona);
p = reshape (p, m * n, 1);
```

```
k = 8;
[c, ptoc] =  kmeans(p, k);
ptoc = round (ptoc);
p = ptoc(c);
p = reshape (p, m, n);
g = uint8 (p);
imshow(g);
```

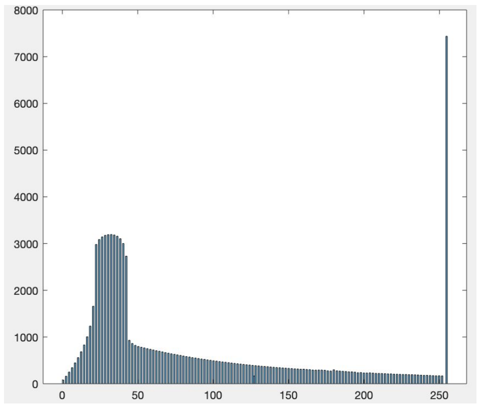


v. Double the value of each pixel with a value in [0, 127] and set all the other pixels to a value of 255. Display the brighter image.

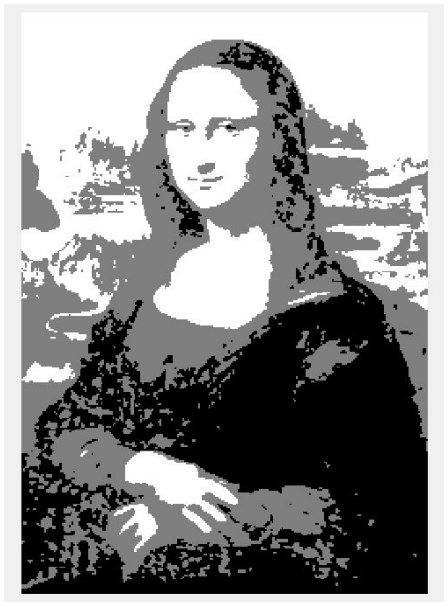
```
m4 = m1; %copy

i1 = m4 < 127;
i2 = 128 <= m4;
m4(i1) = 2*m4(i1);
m4(i2) = 255;
imshow(uint8(m4));

%%PICTURE BELOW
```

vi. Repeat parts (ii)–(iv) with the brighter image.

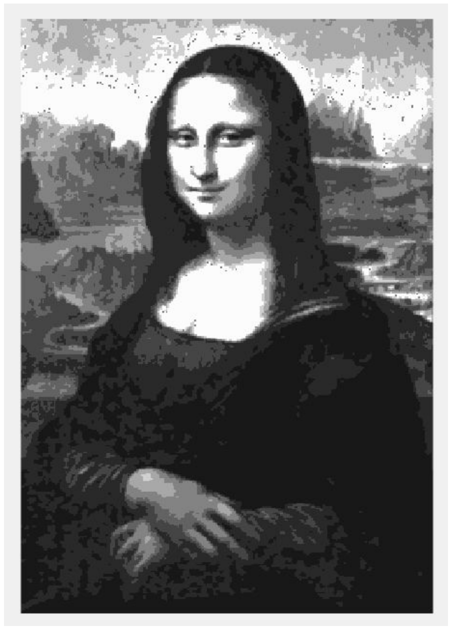
```
histogram(m4(:),256);
```



```
t1 = 35;
t2 = 100;

i1 = m4 < t1;
i2 = t1 <= m4 & m4 < t2;
i3 = t2 <= m4;

m4(i1) = 0;
m4(i2) = 127;
m4(i3) = 255;
m4 = imshow(uint8(m4));
```



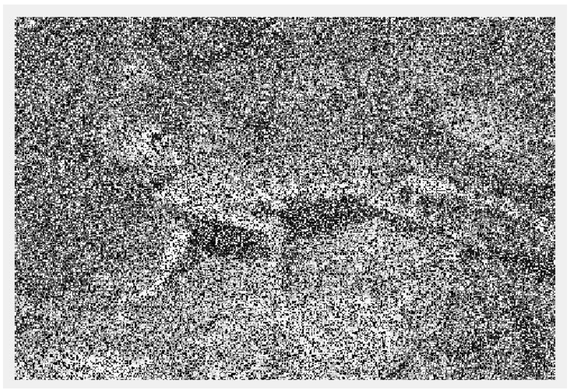
```
[m, n] = size (m4);
p = double (m4);
p = reshape (p, m * n, 1);
k = 8;
[c, ptoc] =  kmeans(p, k);
ptoc = round (ptoc);
p = ptoc(c);
p = reshape (p, m, n);
g = uint8 (p);
imshow(g);
```



2. Noise Removal

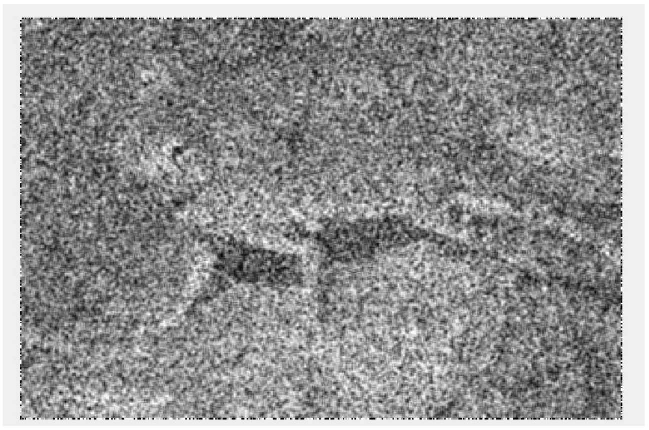
   i. Read in the image file and display the image.

```
liz = imread("lizard_noisy.png");
imshow(liz);
```



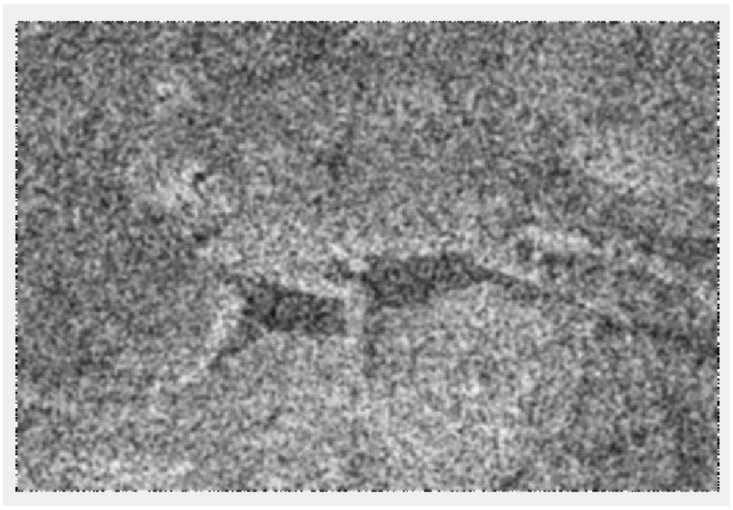
   ii. Denoise the image by replacing each pixel with the average value of its 3 × 3 neighborhood.

```
p = double ( liz );
[ m, n ] = size ( liz );
p2 = p;
for i = 2 : m - 1
    for j = 2 : n - 1
        p2(i,j) = mean( ...
            [p(i+1,j-1), p(i+1,j), p(i+1,j+1), ...
            p(i, j-1),  p(i, j),  p(i, j+1), ...
            p(i-1,j-1), p(i-1,j), p(i-1,j+1) ] );
    end
end
lizmean = uint8 ( p2 );
imshow(lizmean);
```



iii. Apply the same averaging technique a second time. Does the image improve?

```
p = double ( lizmean );
[ m, n ] = size ( lizmean );
p2 = p;
for i = 2 : m - 1
    for j = 2 : n - 1
        p2(i,j) = mean( ...
            [p(i+1,j-1), p(i+1,j), p(i+1,j+1), ...
            p(i, j-1),  p(i, j),  p(i, j+1), ...
            p(i-1,j-1), p(i-1,j), p(i-1,j+1) ] );
    end
end
lizmean2 = uint8 ( p2 );
imshow(lizmean2);
```

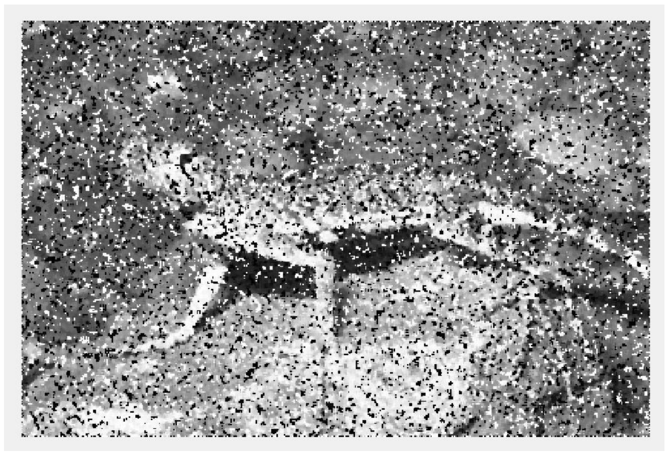


The image did not improve.

iv. Denoise the image by replacing each pixel with the median value of its 3 × 3 neighborhood.

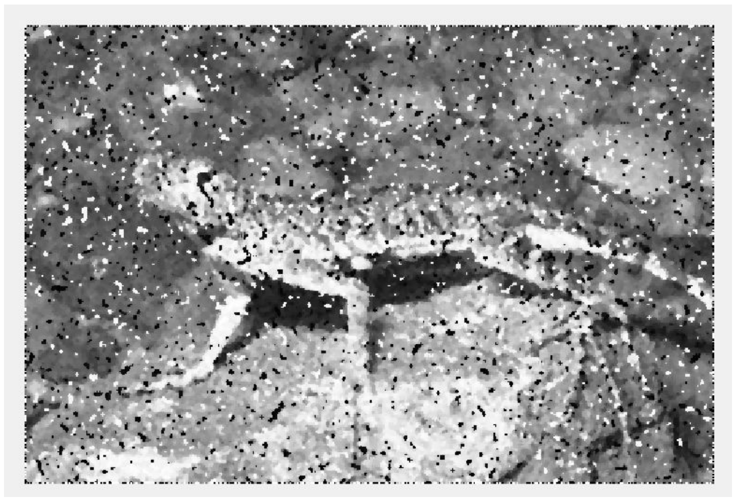
```
p = double ( liz );
[ m, n ] = size ( liz );
p2 = p;
for i = 2 : m - 1
    for j = 2 : n - 1
        p2(i,j) = median( ...
            [p(i+1,j-1), p(i+1,j), p(i+1,j+1), ...
            p(i, j-1),  p(i, j),  p(i, j+1), ...
            p(i-1,j-1), p(i-1,j), p(i-1,j+1) ] );
    end
```

```
end
lizmed = uint8 ( p2 );
imshow(lizmed);
```



v. Apply the same median technique a second time. Does the image improve?

```
p = double ( lizmed );
[ m, n ] = size ( lizmed );
p2 = p;
for i = 2 : m - 1
    for j = 2 : n - 1
        p2(i,j) = median( ...
            [p(i+1,j-1), p(i+1,j), p(i+1,j+1), ...
            p(i, j-1),  p(i, j),  p(i, j+1), ...
            p(i-1,j-1), p(i-1,j), p(i-1,j+1) ] );
    end
end
lizmed2 = uint8 ( p2 );
imshow(lizmed2);
```

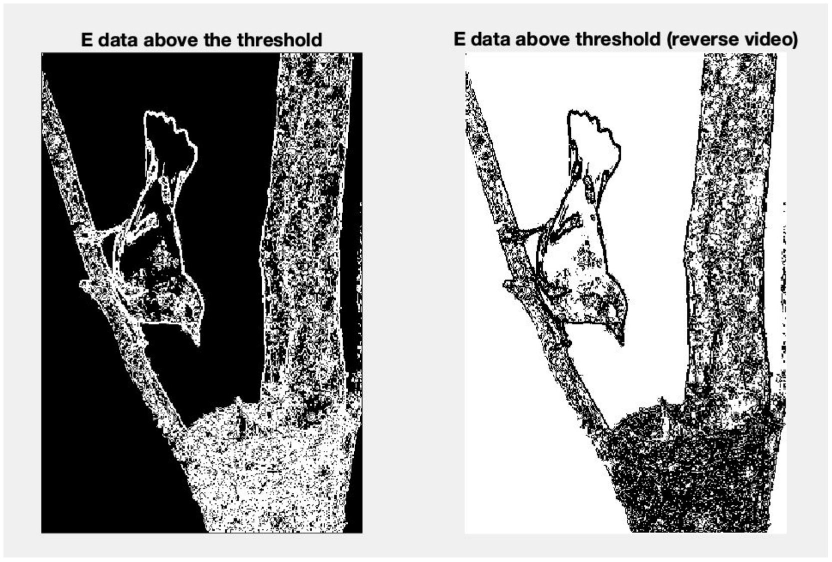


The image improved substantially.

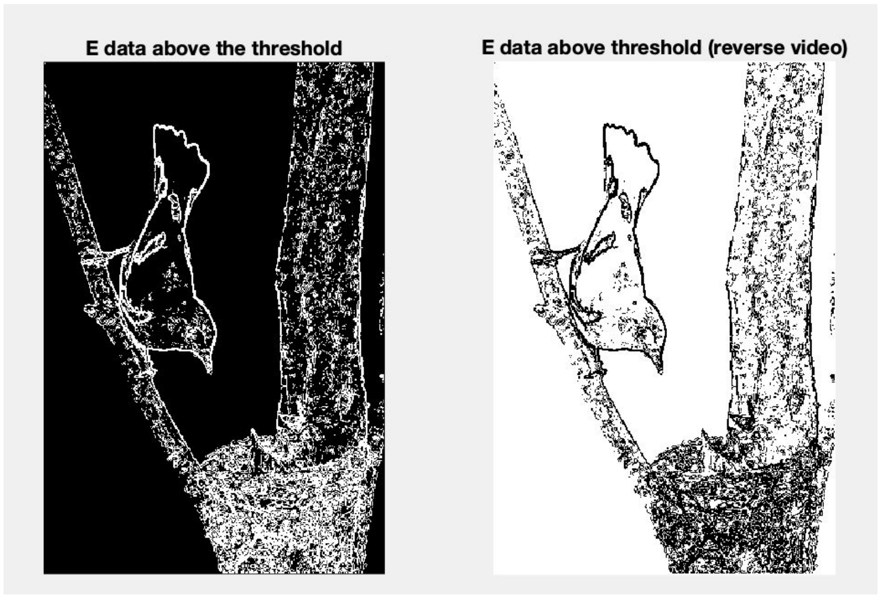
3. Edge Detection

```
a = imread ( 'bird.png');
[ m, n ] = size ( a );
b = double ( a );

e = zeros ( m, n );
e(2:m-1,2:n-1) = abs ( b(3:m,2:n-1) - b(1:m-2,2:n-1) ) ...
               + abs ( b(2:m-1,3:n) - b(2:m-1,1:n-2) );

emin = min ( min ( e ) );
emax = max ( max ( e ) );

e = round ( 255 * ( e - emin ) / ( emax - emin ) );

thresh = 20;
e = 255 * ( thresh < e );
subplot(1,2,1);
imshow ( uint8 ( e ) );
title ('E data above the threshold');
% reverse video
e_reverse = 255 - e;
subplot(1,2,2);
imshow ( uint8 ( e_reverse ) );
title ('E data above threshold (reverse video)');
```

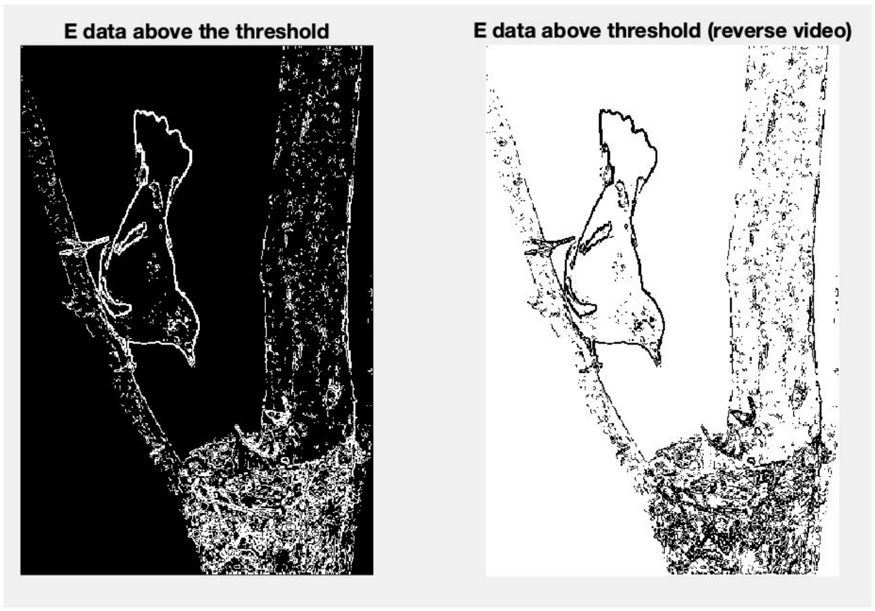
Thresh = 20


E data above the threshold
E data above threshold (reverse video)

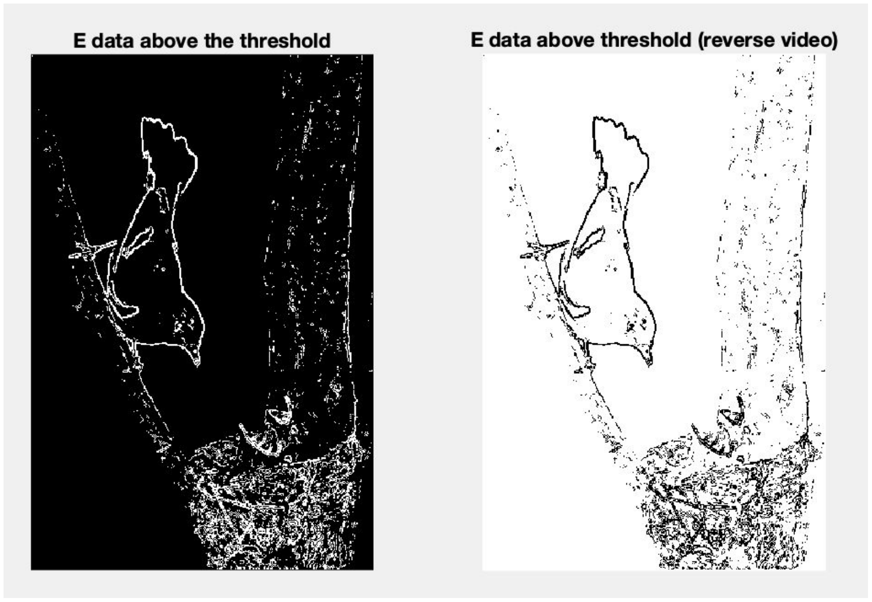
Thresh = 30


E data above the threshold
E data above threshold (reverse video)

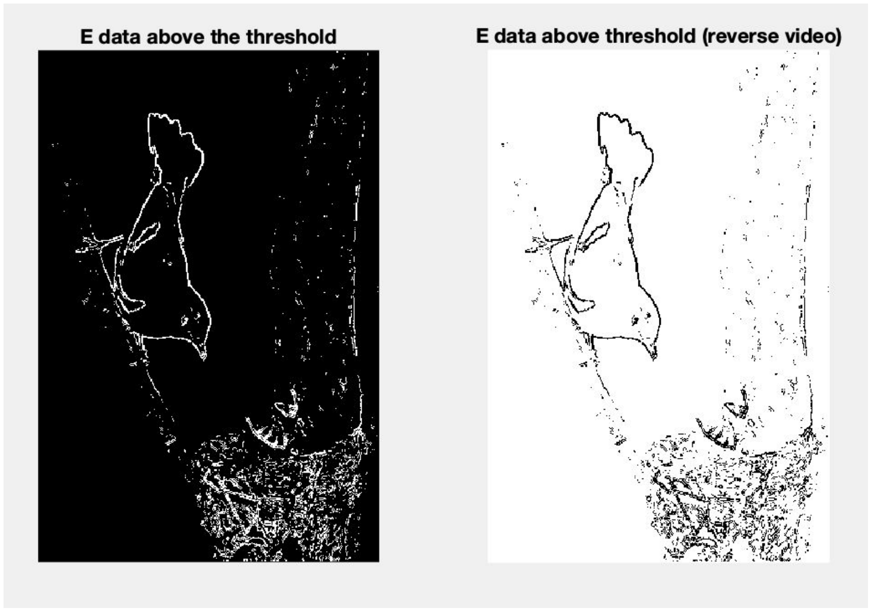
Thresh = 40


E data above the threshold
E data above threshold (reverse video)

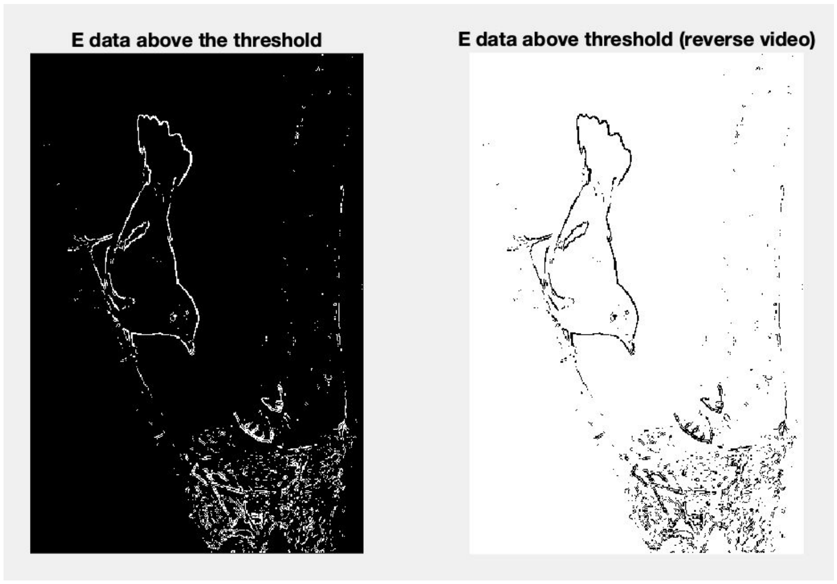
Thresh = 50


E data above the threshold
E data above threshold (reverse video)

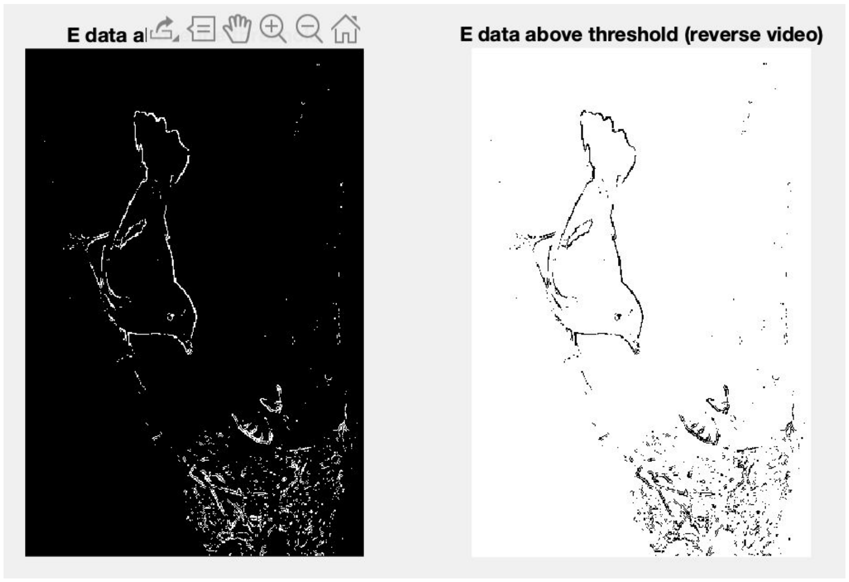
Thresh = 60


E data above the threshold
E data above threshold (reverse video)

Thresh = 70


E data above the threshold
E data above threshold (reverse video)

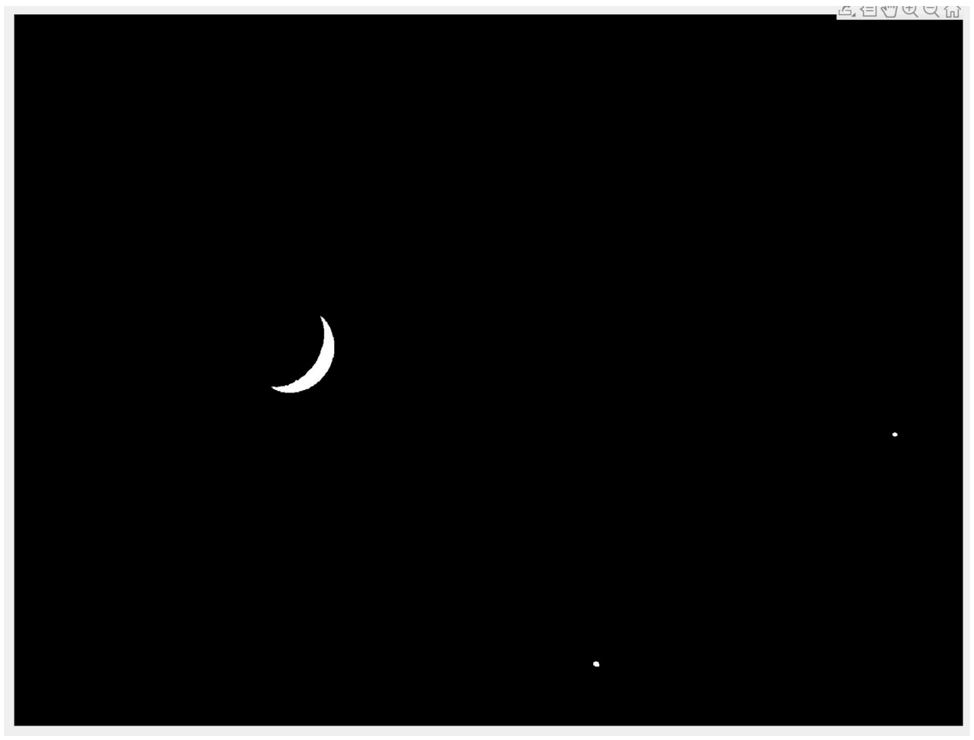
Thresh = 80



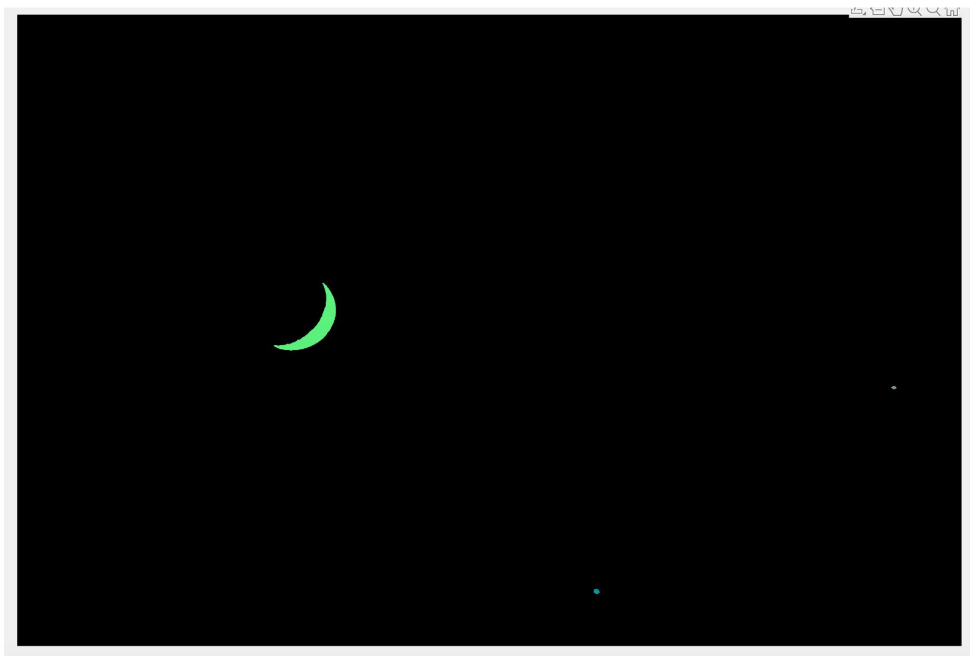
Looks good to me!

4. Component Identification
   i. For the conjunction gray file, determine a threshold parameter t between 0 and 255 so that when you threshold the image, everything is black but the two stars and the moon, which are white (255).

Thresh = 240

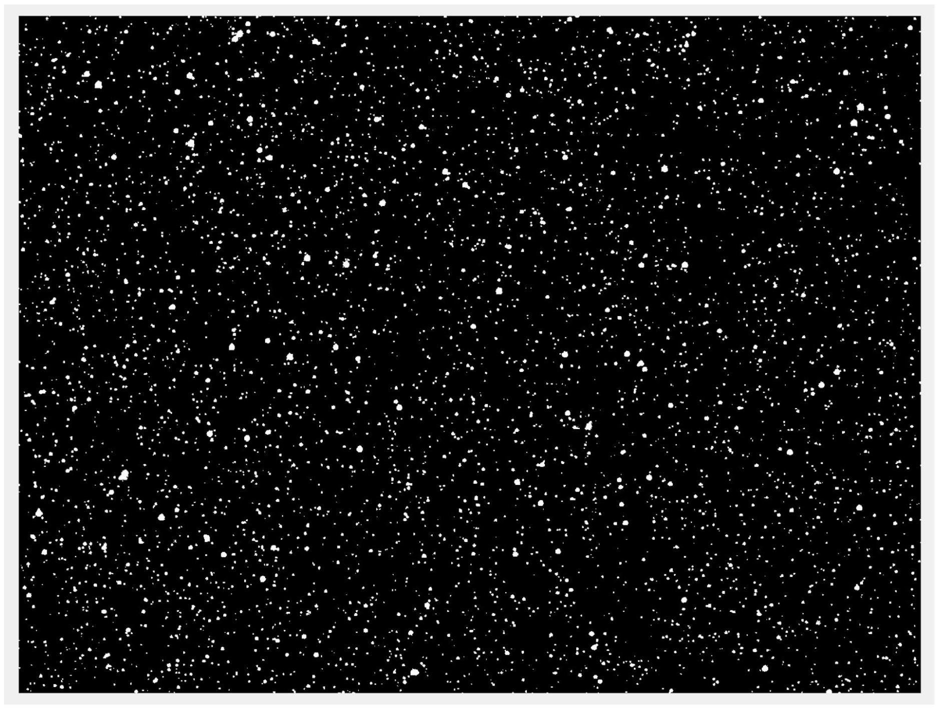


ii. Apply a connected component identification algorithm to your thresholded image, which returns the number of components.
Number of components: 3 (components given random color).



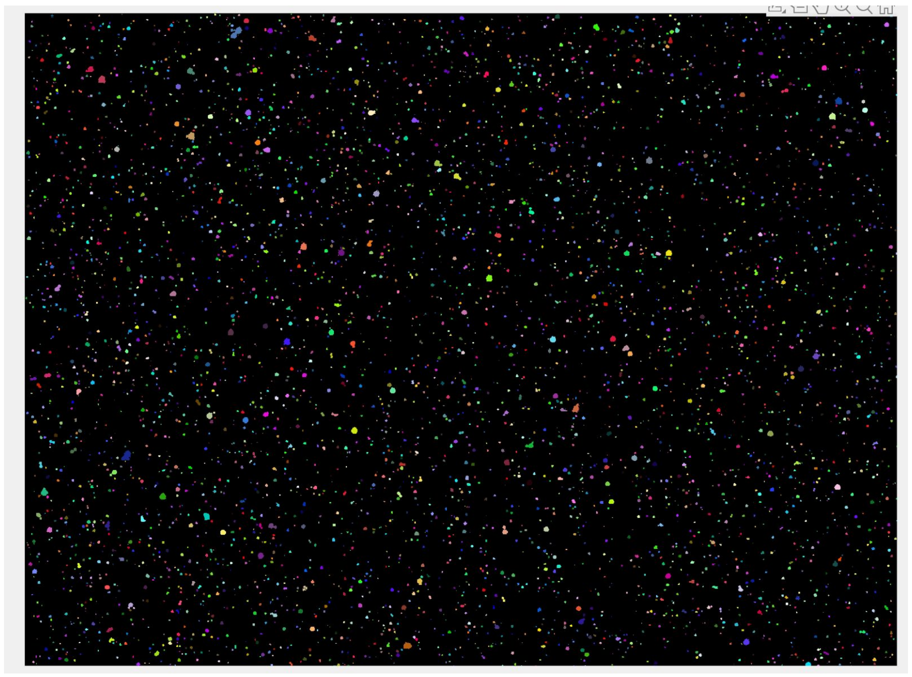
iii. For the star field gray.png file, determine a threshold parameter t between 0 and 255 so that the thresholded image separates white blobs of stars. Include a plot of your thresholded image along with your thresholding parameter.

Thresh = 200



iv. Apply a connected component identification algorithm to your thresholded image. How many stars does your code count?

5159 stars. (components given random color).



Code for #4.

```
clear all
%a = imread ( 'conjunction_gray.png');
a = imread ( 'star_field_gray.png');
%a = imread ( 'test.png');
[ m, n ] = size ( a );
b = double ( a );




thresh = 200;
e = 255 * ( thresh < a );
subplot(2,1,1);
imshow ( uint8 ( e ) );
title ('E data above the threshold');
% reverse video
e_reverse = 255 - e;
subplot(2,1,2);
 imshow ( uint8 ( e_reverse ) );
 title ('E data above threshold (reverse video)');
p = uint8(e);
label = zeros(m,n);
l = 0;
for i = 2 : m
    for j = 2 : n
        %disp([i j])
        if ( p(i,j) == 0 ) %if the current pixel is zero, skip
            label(i,j) = 0;
        elseif ( p(i-1,j) == 0 && p(i,j-1) == 0 ) %if left and above are zero,
assign new label
            l = l + 1;
            label(i,j) = l;
        elseif ( p(i-1,j) == 0 ) %if the left label is just zero, then assign
it the above label.
            label(i,j) = label(i,j-1);
        elseif ( p(i,j-1) == 0 ) %if the above label is just zero, then assign
it the left label.
            label(i,j) = label(i-1,j);
        else %otherwise, give it the lowest label between above and left.
            label(i,j) = min (label(i-1,j), label(i,j-1));
        end
    end
end


labelN = numel(unique(label(:))); %number of unique labels
index = 1:labelN-1; %index created to assign numbers to labels.
index = index';
index = [index index]; %transpose and make a copy

%label reassignment
cond = 0;
count = 1;
while cond == 0
    if count == 0
        cond = 1;
    end
count = 0;
for i = 1:m
    for j = 1:n
        if label(i,j) ~= 0 %if this label is nonzero
            adj = [label(i,j) label(i-1,j) label(i,j-1)];
            adj(adj == 0) = [];%remove zero
            if range(adj) ~= 0 %if they don't match, set them equal to the
smallest one.
                minlabel = min(adj);
                if label(i-1,j) == 0
                    label(i,j) = minlabel;
                    label(i,j-1) = minlabel;
                elseif label(i,j-1) == 0
                    label(i,j) = minlabel;
                    label(i-1,j) = minlabel;
                else
                    label(i,j) = minlabel;
                    label(i-1,j) = minlabel;
                    label(i,j-1) = minlabel;
                end
                count = count +1;
            end
        end
    end
end
end



cond = 0;
count = 1;
while cond == 0
    if count == 100
        cond = 1;
    end
    %count = 0;
    for i = 1:size(index,1)
        if index(i,1) == index(i,2)
```

```
            continue;
        elseif index(index(i,2),1)  == index(index(i,2),2)
            continue;
        else
            count = count + 1;
            index(i,2) = index(index(i,2),2);
            %disp(count);
        end
    end
    count = count + 1;
end

num = numel(unique(index(:,2)));
labels = 1:num;
labels = labels';
labels = [unique(index(:,2)) labels];

for i = 1:num
    final = find(index(:,2) == labels(i,1));
    index(final,2) = i;
end

for i = 1:m
    for j = 1:n
        if label(i,j) ~= 0
            label(i,j) = index(label(i,j),2);
        end
    end
end

colors = uint8(256*rand(3,num));
img = zeros(m,n,3);

for i = 1:m
    for j = 1:n
        if label(i,j) ~= 0
            img(i,j,1) = colors(1,label(i,j));
            img(i,j,2) = colors(2,label(i,j));
            img(i,j,3) = colors(3,label(i,j));
        end
    end
end

num = numel(unique(label(:))) - 1;

img = uint8(img);
imwrite(e,"200.png")
imwrite(img,"colored.png")
```