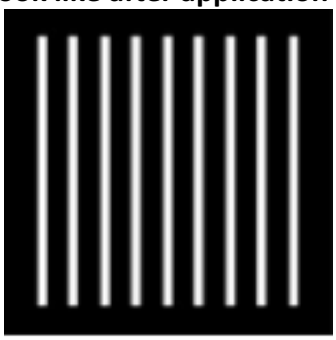
Ray Li
CSC 481
Dr. Furst
10/20/16

5.1 The white bars in the test pattern shown are 7 pixels wide and 210 pixels high. The separation between bars is 17 pixels. What would this image look like after application of



a) A 3 * 3 arithmetic mean filter?

After applying the 3x3 mean filter to the image, the white bars will become blurred. The edge of each white bar is not going to be as defined or sharp as that in the original image. Between the white bars and the black background, there will be one column of gray pixels.

b) A 7 * 7 arithmetic mean filter?

The image will look even more blurred than the one produced in a). The edges of white bars will lose definition by turning gray and the white bars will mostly turn into gray color except the middle pixel column.

c) A 9 * 9 arithmetic mean filter?

The image will be even more blurred than the one produced in b). In this image, it would be really hard to see the edges that separate white bars and the black background and all the white bars will turn gray and there will be no white pixel left in the image because they are all turned into gray pixels.

5.6 Repeat Problem 5.1 using a median filter.

a) A 3 * 3 arithmetic median filter?

The image will have rounded corners because the median value with a 3x3 filter for the corner pixel is 0 instead of 1, meaning it will be transformed into a black pixel. Besides this difference, there is no other change between the filtered and original images.

b) A 7 * 7 arithmetic median filter?

The image will have bigger rounded corners because more corner pixels of white bars are turned into pixel value of 0, or black. Graphically, the ends of white bars will be pointier than those produced in a).

c) A 9 * 9 arithmetic median filter?

The result image will have white bars that have even pointier ends because even more corner pixels are being neutralized into pixel value of 0. Note that, the median filter in this case does not blur the image and instead it only sharpens the corners of the white bars.

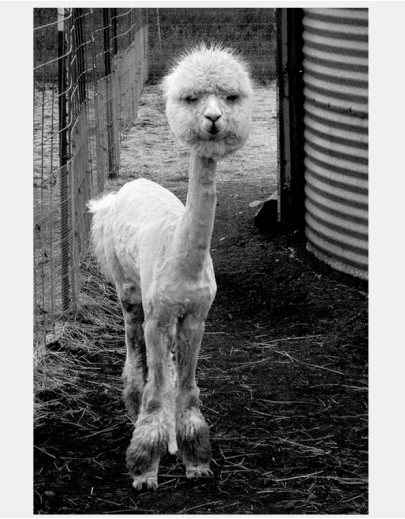
Programming Problems

Problem 1: Edge Detection

Use the edge function to generate masks for Roberts, Canny, Sobel, and Prewitt operators on an image of your choice. Note also that the various edge functions support a number of parameters – feel free to explore those to get more interesting results. State which operator gives the best performance and why you think so.

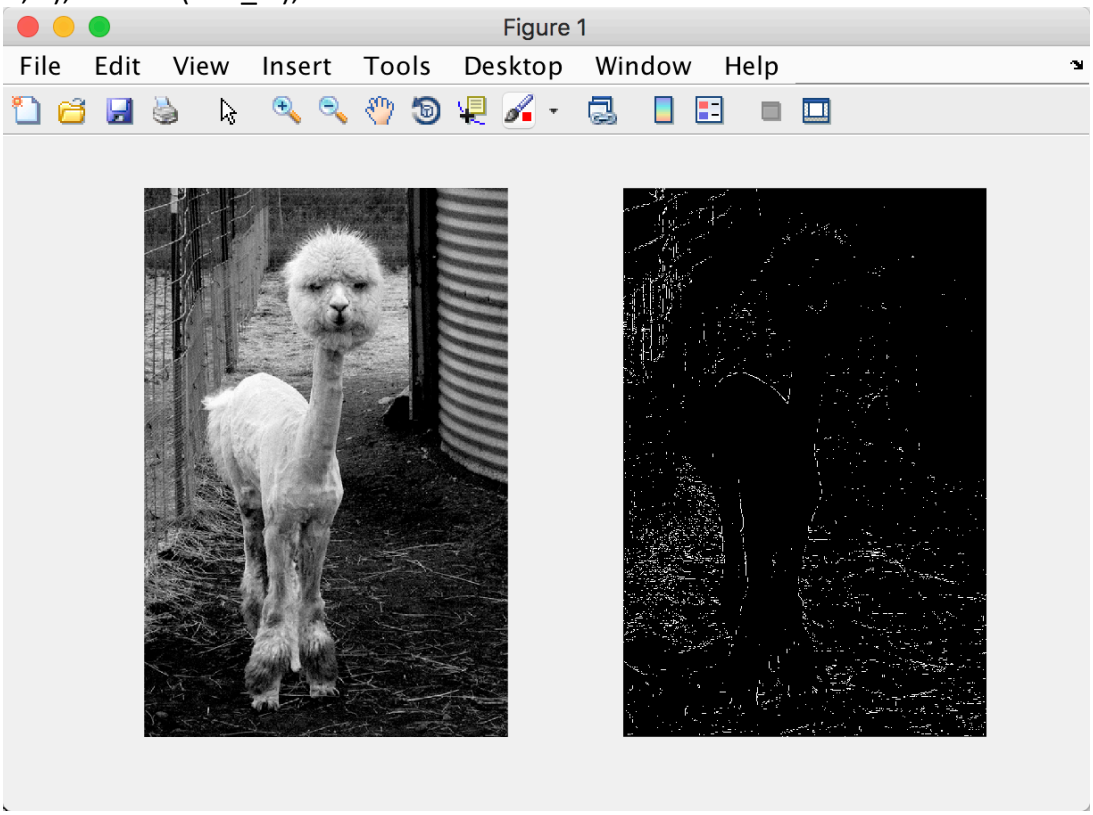
Original Image:

```
>> A = imread('lama.jpg');
>> B = rgb2gray(A);
>> imshow(B);
```



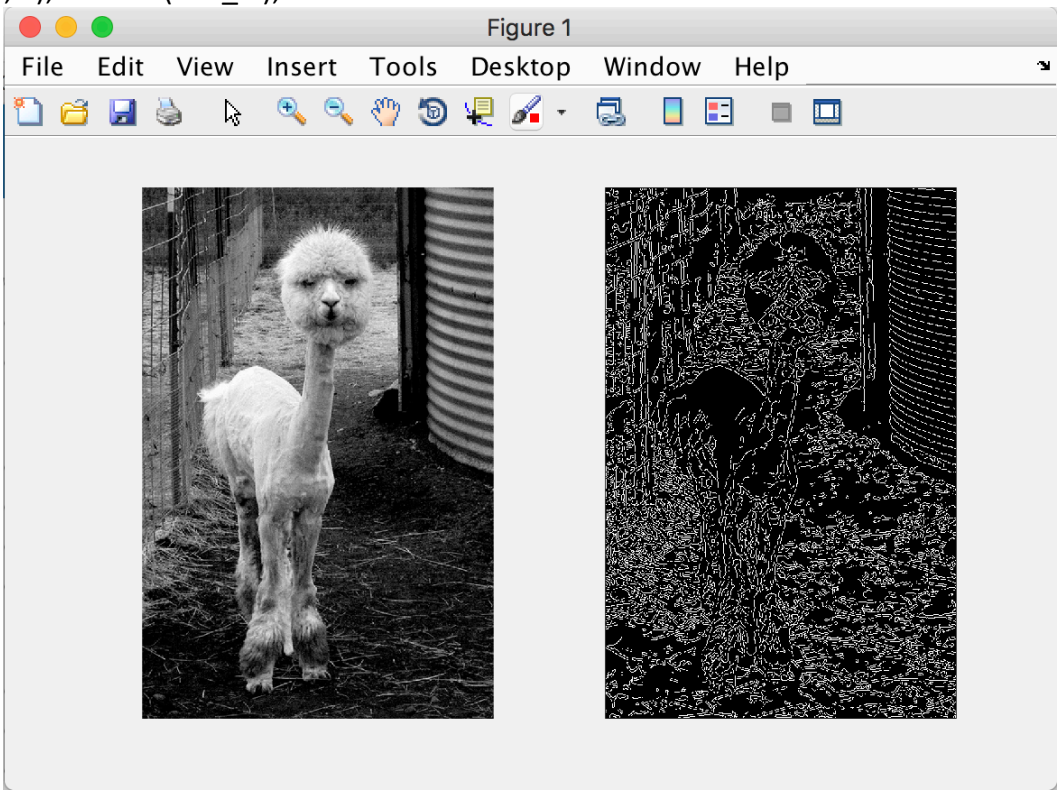
Roberts Mask **(with default parameters)**

```
>> Rob_fil = edge(B, 'Roberts');
>> subplot(1, 2, 1); imshow(B);
>> subplot(1, 2, 2); imshow(Rob_fil);
```



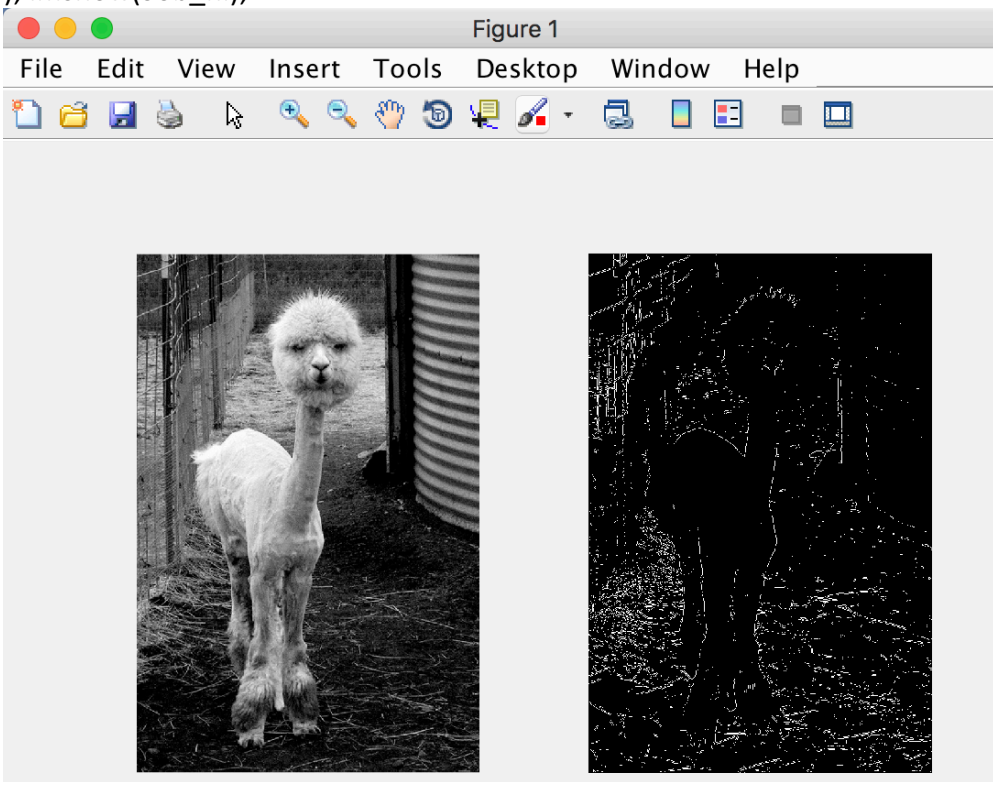
Canny Mask **(with default paramters)**

```
>> Can_fil = edge(B, 'Canny');
>> subplot(1, 2, 1); imshow(B);
>> subplot(1, 2, 2); imshow(Can_fil);
```



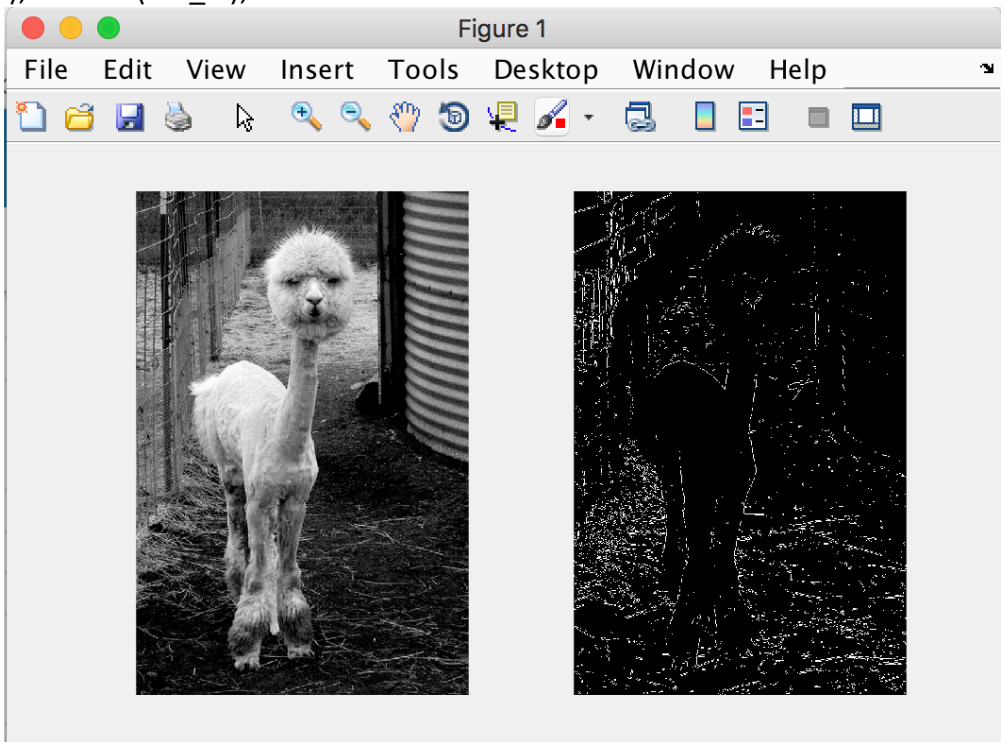
Sobel Mask **(with default parameters)**

```
>> Sob_fil = edge(B, 'Sobel');
>> subplot(1, 2, 1); imshow(B);
>> subplot(1, 2, 2); imshow(Sob_fil);
```

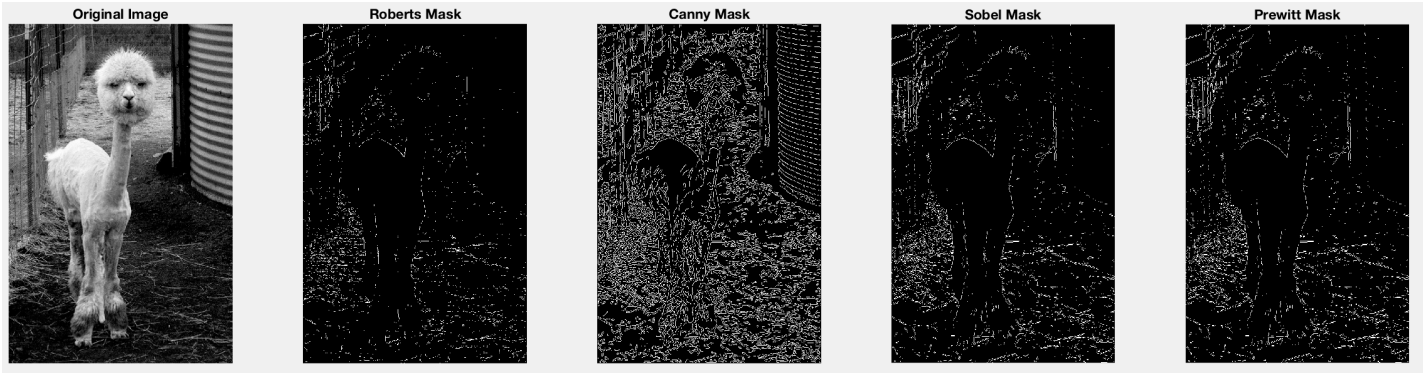


Prewitt Mask **(with default parameters)**

```
>> Pre_fil = edge(B, 'Prewitt');
>> subplot(1, 2, 1); imshow(B);
>> subplot(1, 2, 2); imshow(Pre_fil);
```



Here is the summary of all four masks

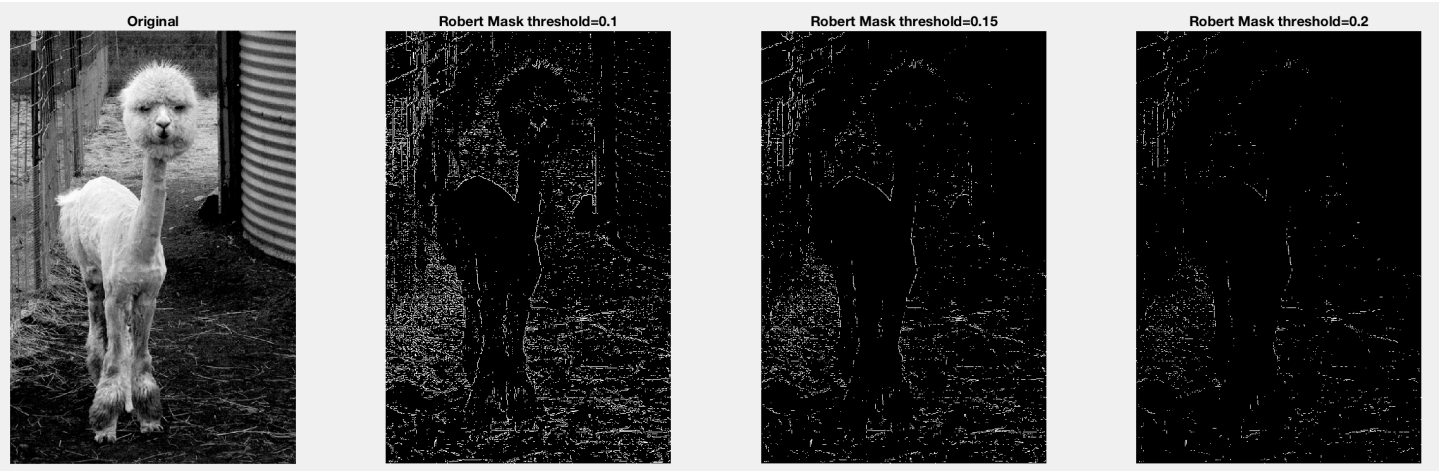


With default parameter settings, as shown above, from my observation, Sobel mask and Prewitt Mask give the best performance because they both produce really similar results that include edges that almost form the shape of the lama in the picture. In the result from Roberts mask, even though we can kind of see the shape of lama, it is not as defined as those in Sobel and Prewitt masks' results. For Canny mask, it captures too much noise and it is not an ideal result if we are doing segmentation.

Threshold Experiment

Roberts Mask **(With Thresholds 0.1, 0.15 and 0.2)**

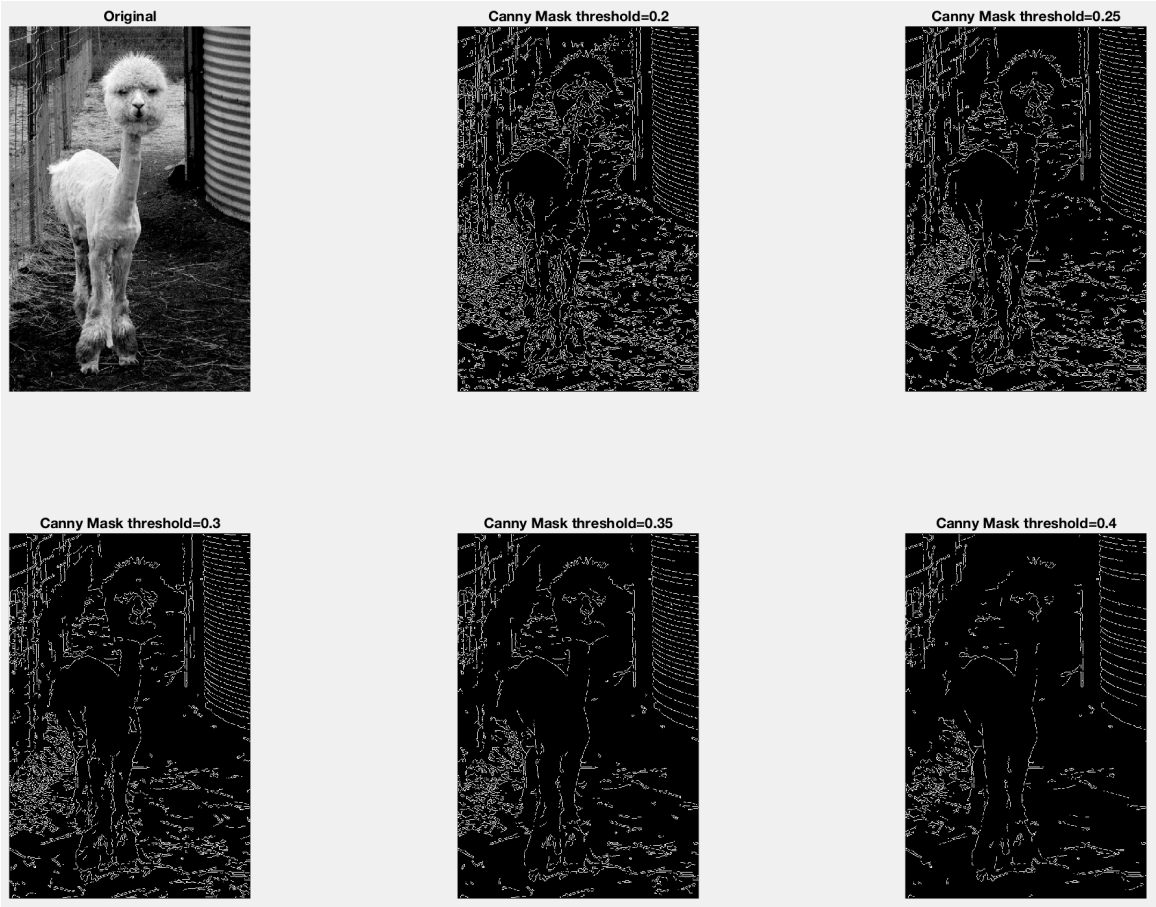
```
>> Rob_fil = edge(B, 'Roberts', 0.5);
>> Rob_fil2 = edge(B, 'Roberts', 0.15);
>> Rob_fil3 = edge(B, 'Roberts', 0.2);
>> subplot(1, 4, 1); imshow(B); title('Original'); subplot(1, 4, 2); imshow(Rob_fil); title('Robert Mask threshold=0.1'); subplot(1, 4, 3); imshow(Rob_fil2); title('Robert Mask threshold=0.15'); subplot(1, 4, 4); imshow(Rob_fil3); title('Robert Mask threshold=0.2');
```



As shown above, for Robert mask, as we increase the threshold, the amount of edges captured is decreasing. Among these three tested threshold values, I think threshold 0.15 gives the best result since the one with threshold of 0.1 contains too much noise and the one with threshold of 0.2 has too much missing information.

Canny Mask **(With Thresholds 0.2, 0,25, 0.3, 0.35 and 0.4)**

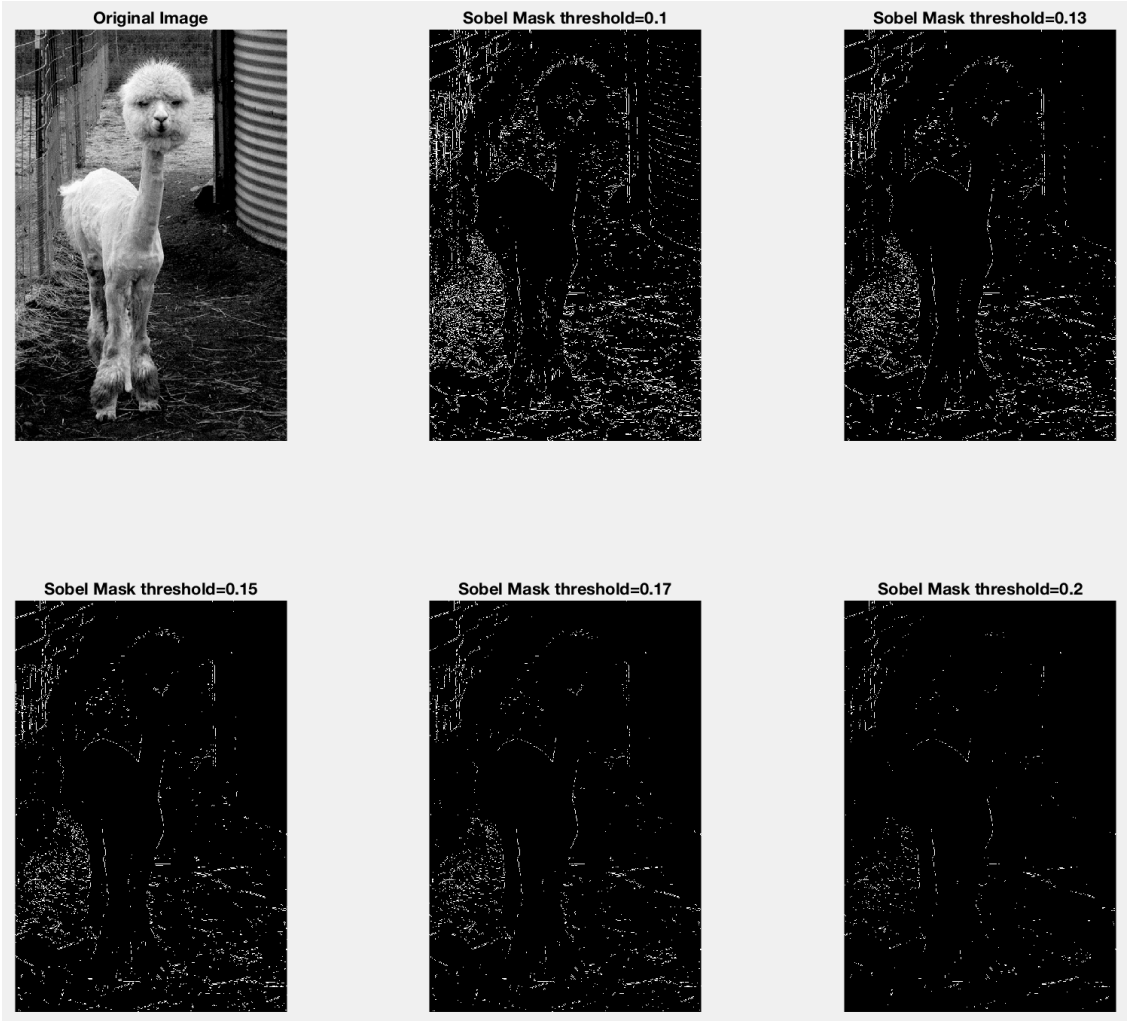
```
>> Can_fil1 = edge(B, 'Canny', 0.2);
>> Can_fil2 = edge(B, 'Canny', 0.25);
>> Can_fil3 = edge(B, 'Canny', 0.3);
>> Can_fil4 = edge(B, 'Canny', 0.35);
>> Can_fil5 = edge(B, 'Canny', 0.4);
>> subplot(2, 3, 1); imshow(B); title('Original'); subplot(2, 3, 2); imshow(Can_fil1); title('Canny Mask threshold=0.2'); subplot(2, 3, 3); imshow(Can_fil2); title('Canny Mask threshold=0.25'); subplot(2, 3, 4); imshow(Can_fil3); title('Canny Mask threshold=0.3'); subplot(2, 3, 5); imshow(Can_fil4); title('Canny Mask threshold=0.35'); subplot(2, 3, 6); imshow(Can_fil5); title('Canny Mask threshold=0.4');
```



As shown above, as we increase the threshold for Canny mask, we lose more and more edges that were captured with lower threshold. For this image, using Canny Mask, the best threshold should be around 0.35.

Sobel Mask **(With Thresholds 0.1, 0.13, 0.15, 0.17 and 0.2)**

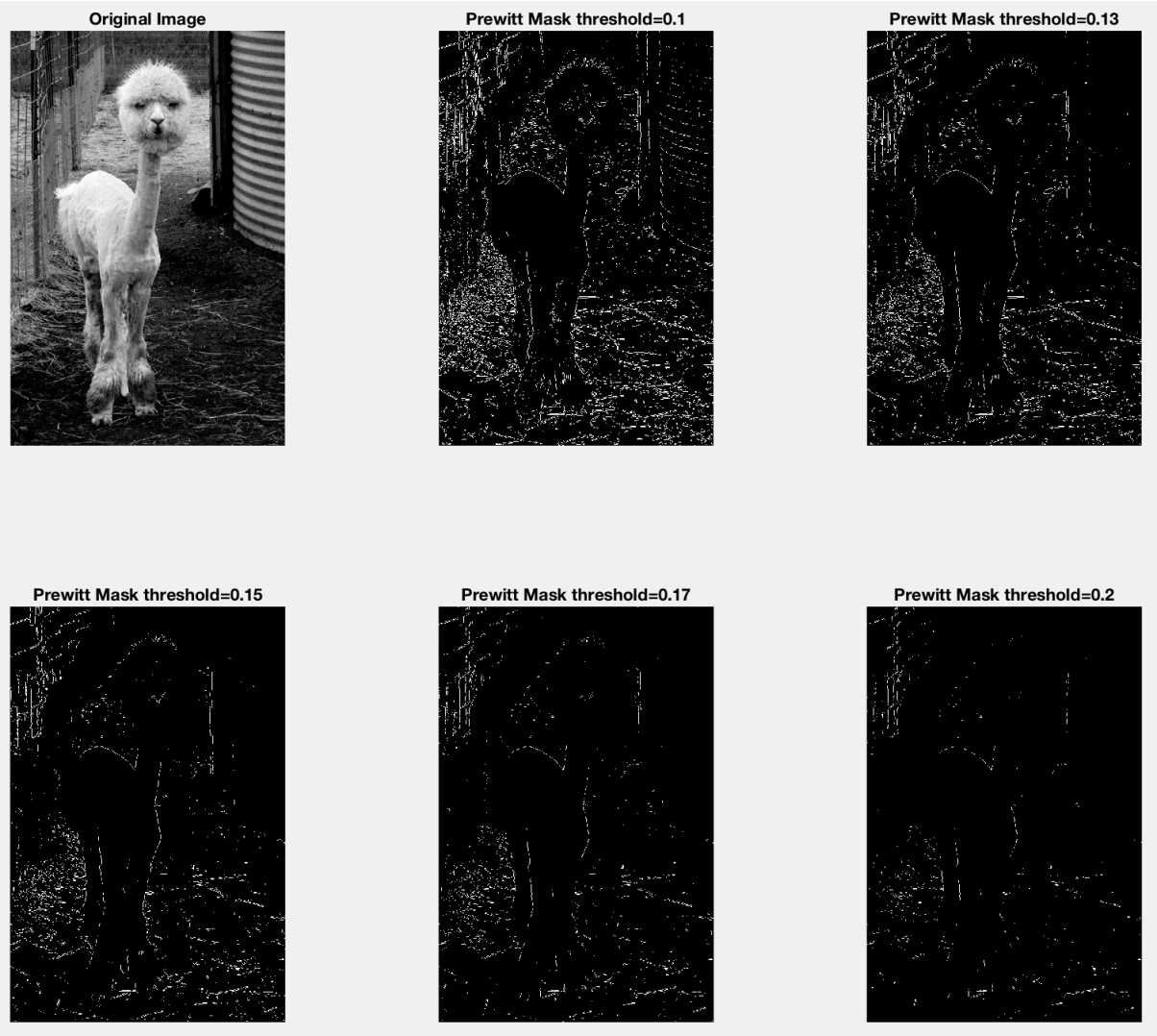
```
>> Sob_fil1 = edge(B, 'Sobel', 0.1);
>> Sob_fil2 = edge(B, 'Sobel', 0.13);
>> Sob_fil3 = edge(B, 'Sobel', 0.15);
>> Sob_fil4 = edge(B, 'Sobel', 0.17);
>> Sob_fil5 = edge(B, 'Sobel', 0.2);
>> subplot(2, 3, 1); imshow(B); title('Original Image'); subplot(2, 3, 2); imshow(Sob_fil1); title('Sobel Mask threshold=0.1'); subplot(2, 3, 3); imshow(Sob_fil2); title('Sobel Mask threshold=0.13'); subplot(2, 3, 4); imshow(Sob_fil3); title('Sobel Mask threshold=0.15'); subplot(2, 3, 5); imshow(Sob_fil4); title('Sobel Mask threshold=0.17'); subplot(2, 3, 6); imshow(Sob_fil5); title('Sobel Mask threshold=0.2');
```



As shown above, with different threshold value experiments, it shows that the Sobel mask behaves similarly to the Roberts mask in this image. Also like the Canny mask, as the threshold increases, we lose more and more edges detected. In this experiment, the best edge detection threshold should be around 0.15.

Prewitt Mask **(With Thresholds 0.1, 0.13, 0.15, 0.17 and 0.2)**

```
>> Pre_fil1 = edge(B, 'Prewitt', 0.1);
>> Pre_fil2 = edge(B, 'Prewitt', 0.13);
>> Pre_fil3 = edge(B, 'Prewitt', 0.15);
>> Pre_fil4 = edge(B, 'Prewitt', 0.17);
>> Pre_fil5 = edge(B, 'Prewitt', 0.2);
>> subplot(2, 3, 1); imshow(B); title('Original Image'); subplot(2, 3, 2); imshow(Pre_fil1); title('Prewitt Mask threshold=0.1'); subplot(2, 3, 3); imshow(Pre_fil2); title('Prewitt Mask threshold=0.13'); subplot(2, 3, 4); imshow(Pre_fil3); title('Prewitt Mask threshold=0.15'); subplot(2, 3, 5); imshow(Pre_fil4); title('Prewitt Mask threshold=0.17'); subplot(2, 3, 6); imshow(Pre_fil5); title('Prewitt Mask threshold=0.2');
```



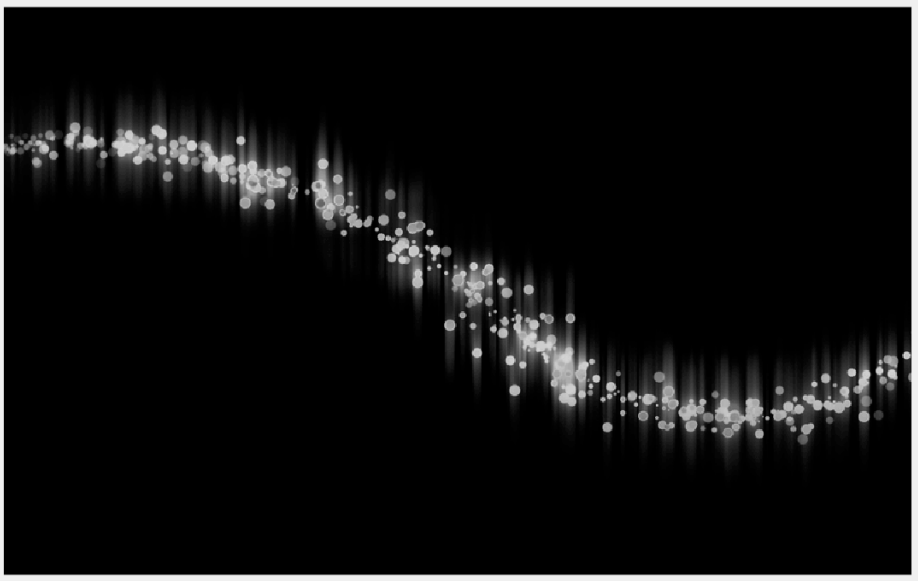
As shown above, the experimented results are extremely similar to the results produced by using Sobel mask. Actually, Sobel and Prewitt masks are really similar to each other and the major difference is that for Sobel mask, the coefficients of the mask are not fixed and can be adjusted.

Problem 2: Color Segmentation

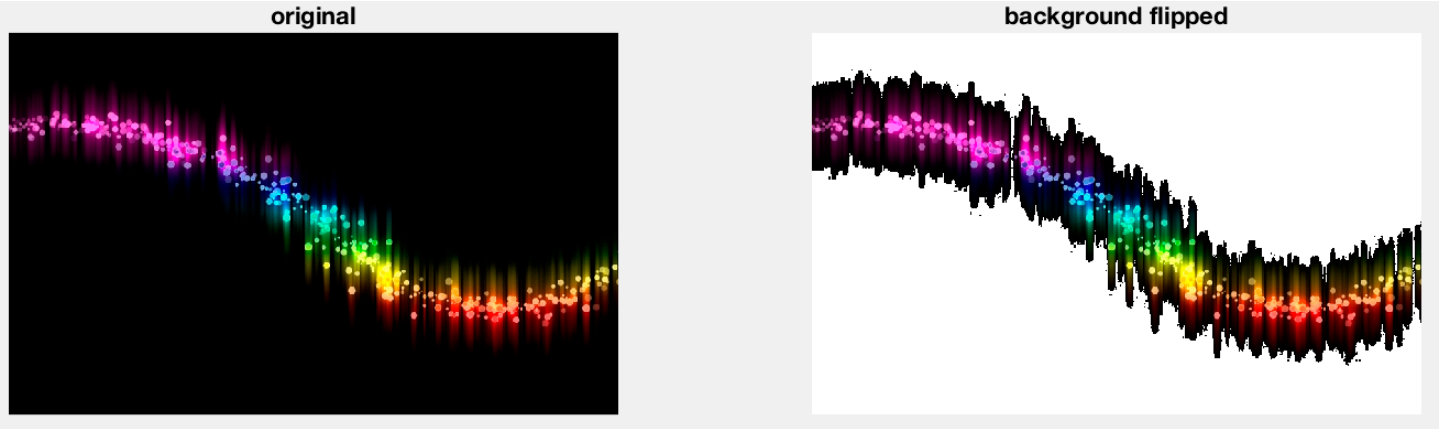
A natural cue to use in segmenting objects from their surroundings in images is color. This problem contrasts segmenting color regions using red, green, and blue thresholds (aligned with the RGB axes of color space) with segmentation using hue, saturation, and intensity bounds (aligned with the coordinate system of HSI or HSV space). For this assignment, it will be more educational to choose an image with strongly colored objects. For example, an image of party balloons works well – make sure they are on a dark or light background.

a) **First, segment your image into objects and background using a threshold on the intensity of the pixels. You can get a grayscale image from an RGB image simply by averaging the three color components of each pixel. Demonstrate your segmentation by replacing the background pixels with a visually distinct color.**

```
>> A = imread('balloon.jpg');
>> r = A(:,:,1); g = A(:,:,2); b = A(:,:,3);
>> gray = r/3 + g/3 + b/3;
>> imshow(gray);
```

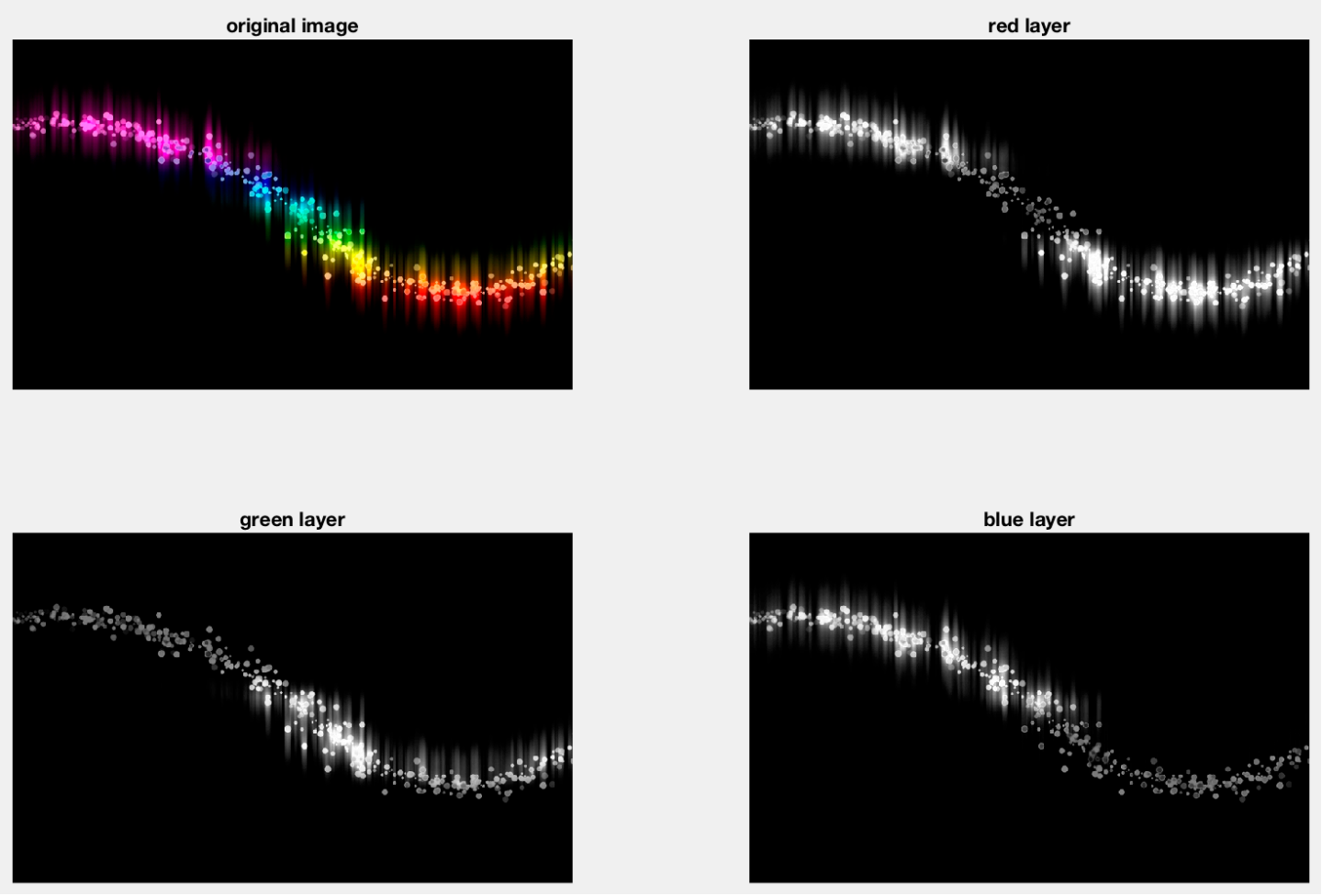


```
>> black = gray == 0;
>> r(black) = 255;
>> g(black) = 255;
>> b(black) = 255;
>> C = cat(3, r, g, b);
>> subplot(1, 2, 1); imshow(A); title('original'); subplot(1, 2, 2); imshow(C); title('background flipped');
```



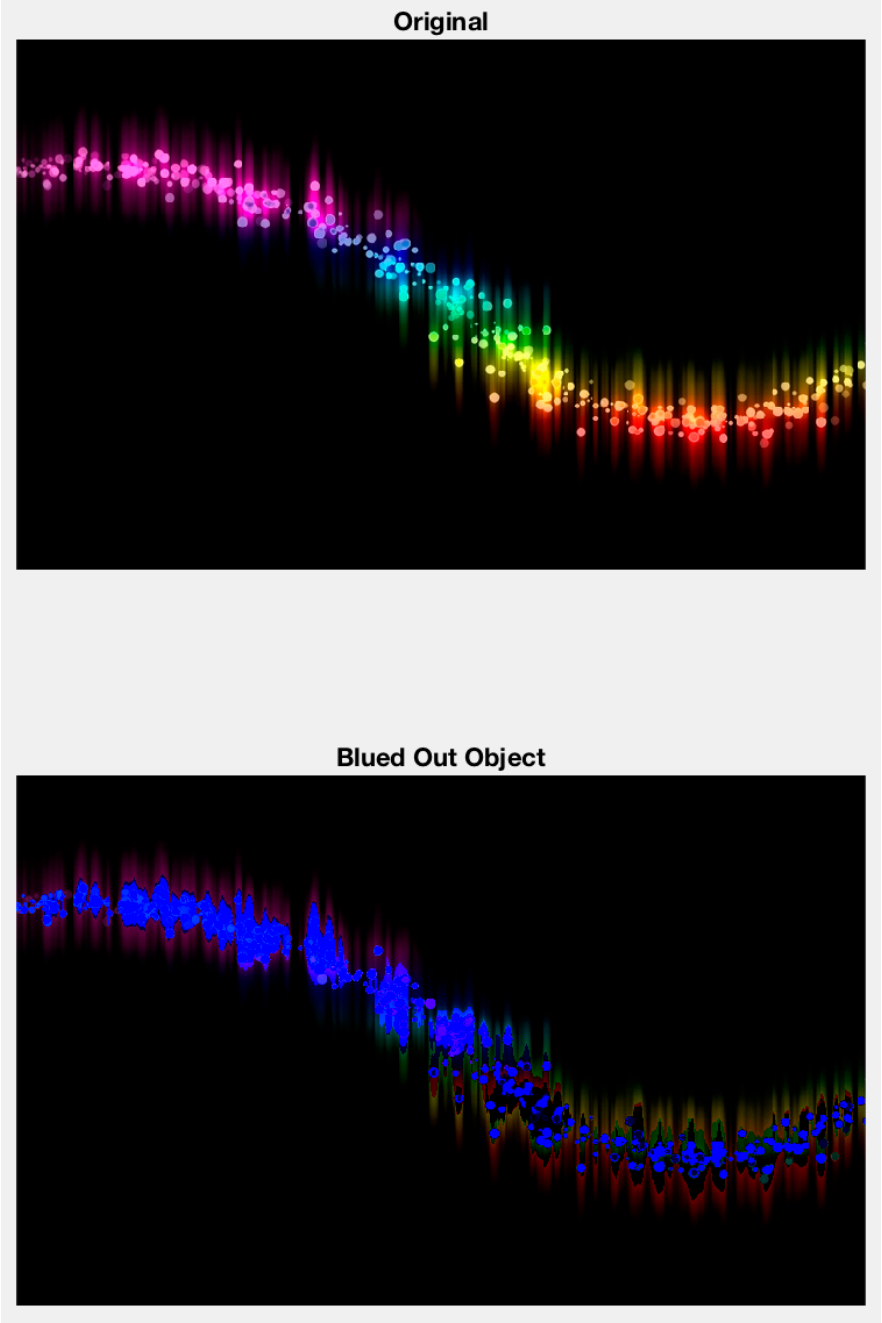
b) Second, use thresholds in each color band (a particular one or two or all three) to isolate the objects in your image. Again, display the results by "bluing out" the intended region. Provide some commentary on how the segmentation succeeded and failed. (481 Students (4): use an automatic thresholding approach instead of choosing thresholds by hand. Hint: look at otsuthresh.)

```
>> r = A(:,:,1);
>> g = A(:,:,2);
>> b = A(:,:,3);
>> [red_counts, rx] = imhist(r);
>> [gre_counts, gx] = imhist(g);
>> [blu_counts, bx] = imhist(b);
>> rthresh = otsuthresh(red_counts);
>> gthresh = otsuthresh(gre_counts);
>> bthresh = otsuthresh(blu_counts);
>> subplot(2, 2, 1); imshow(A); title('original image');
>> subplot(2, 2, 2); imshow(r); title('red layer');
>> subplot(2, 2, 3); imshow(g); title('green layer');
>> subplot(2, 2, 4); imshow(b); title('blue layer');
```



Segment the Entire Object

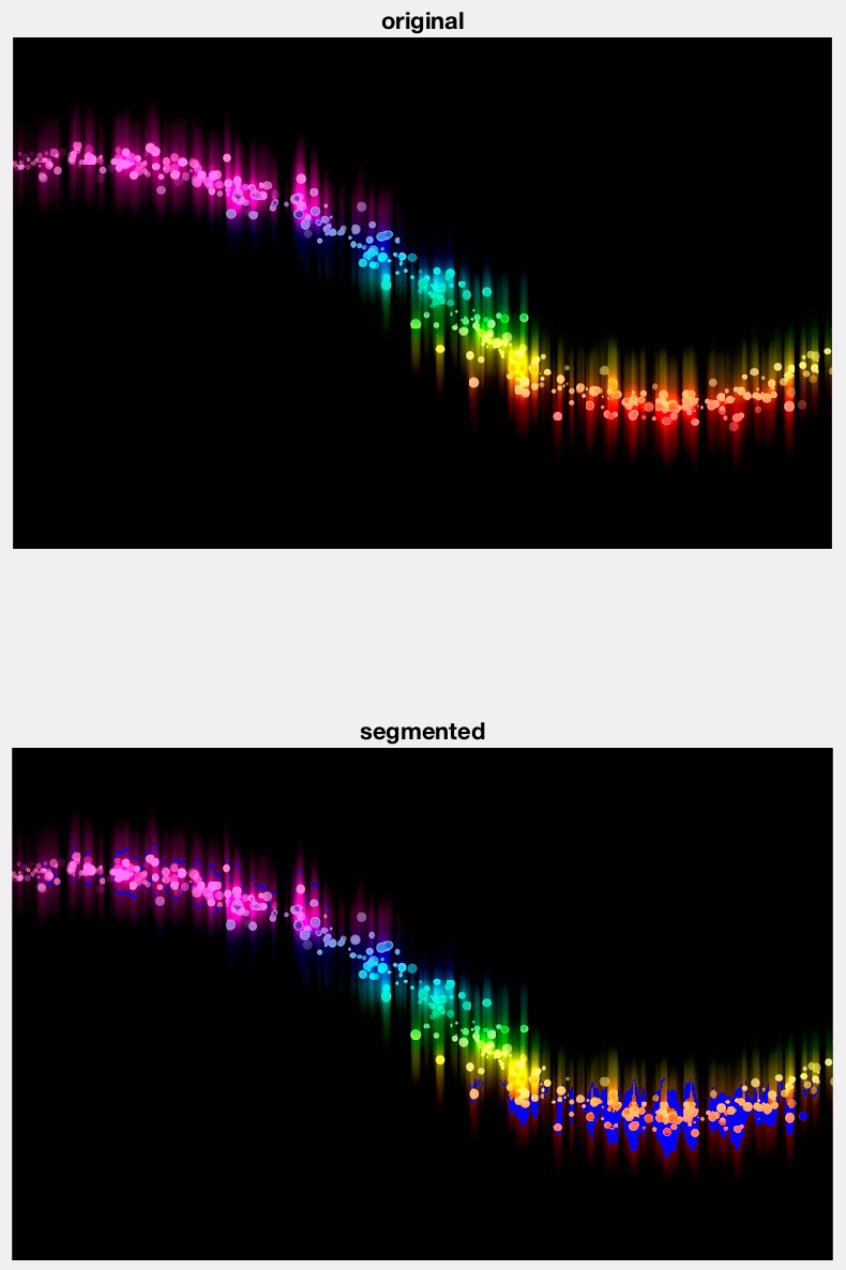
```
>> r(r > rthresh*255) = 0;
>> g(g > gthresh*255) = 0;
>> b(b > bthresh*255) = 255;
>> E = cat(3, r, g, b);
```



Overall, as shown above, the objects are mostly blued out so the segmentation succeeded mostly. The failed part is the “blurred” region in the image.

Segment the Red Component

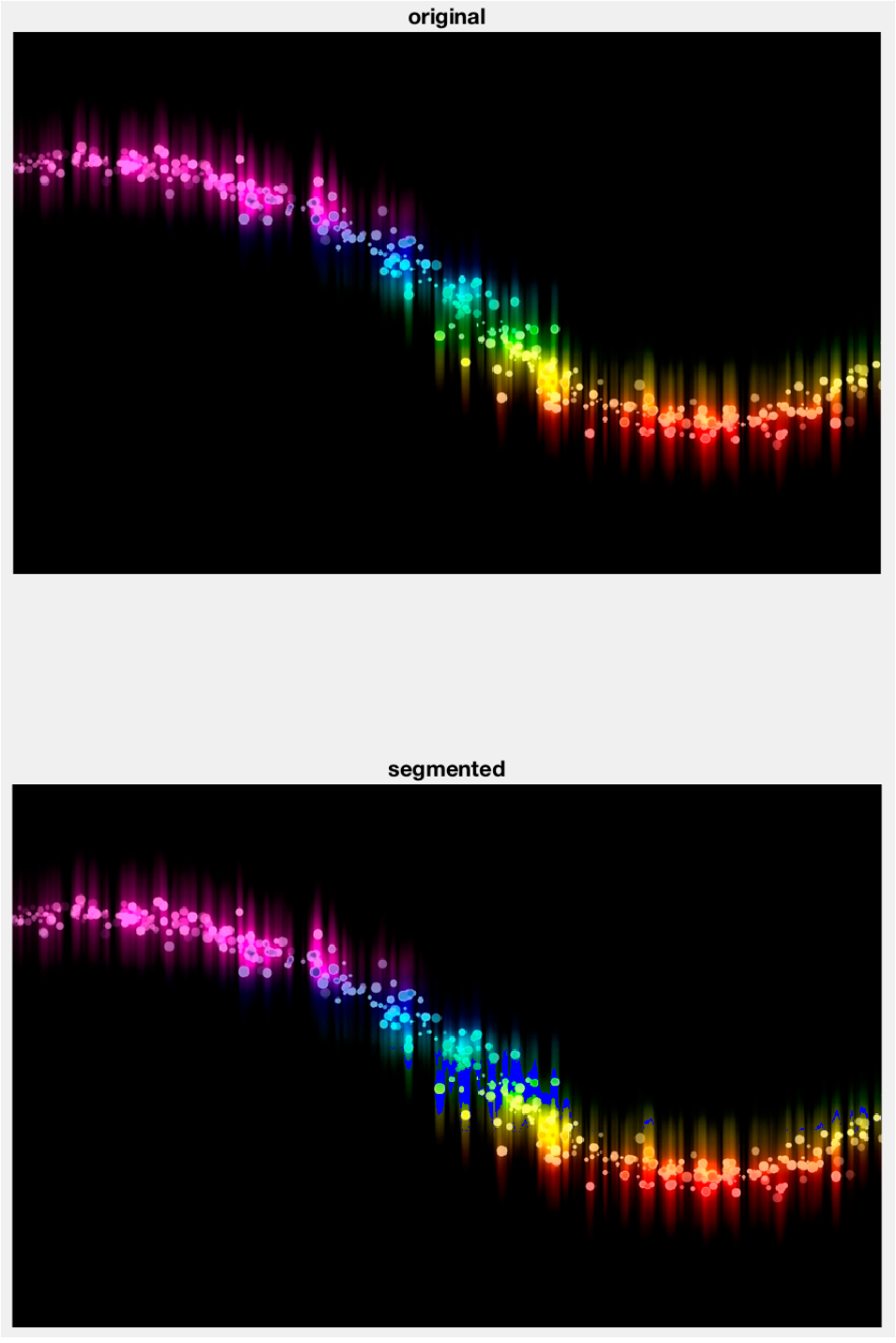
```
>> rule = (r>rthresh*255 & g<gthresh*255 & b<bthresh*255);
>> r = A(:,:,1);
>> g = A(:,:,2);
>> b = A(:,:,3);
>> r(rule) = 0;
>> g(rule) = 0;
>> b(rule) = 255;
>> red_region = cat(3, r, g, b);
>> subplot(2, 1, 1); imshow(A); title('original');
>> subplot(2, 1, 2); imshow(red_region); title('segmented');
```



As shown above, if we only want the red objects from the image, it succeeded in segmenting out objects that have “pure” red color by turning them into “pure” blue.

Segment the Green Component

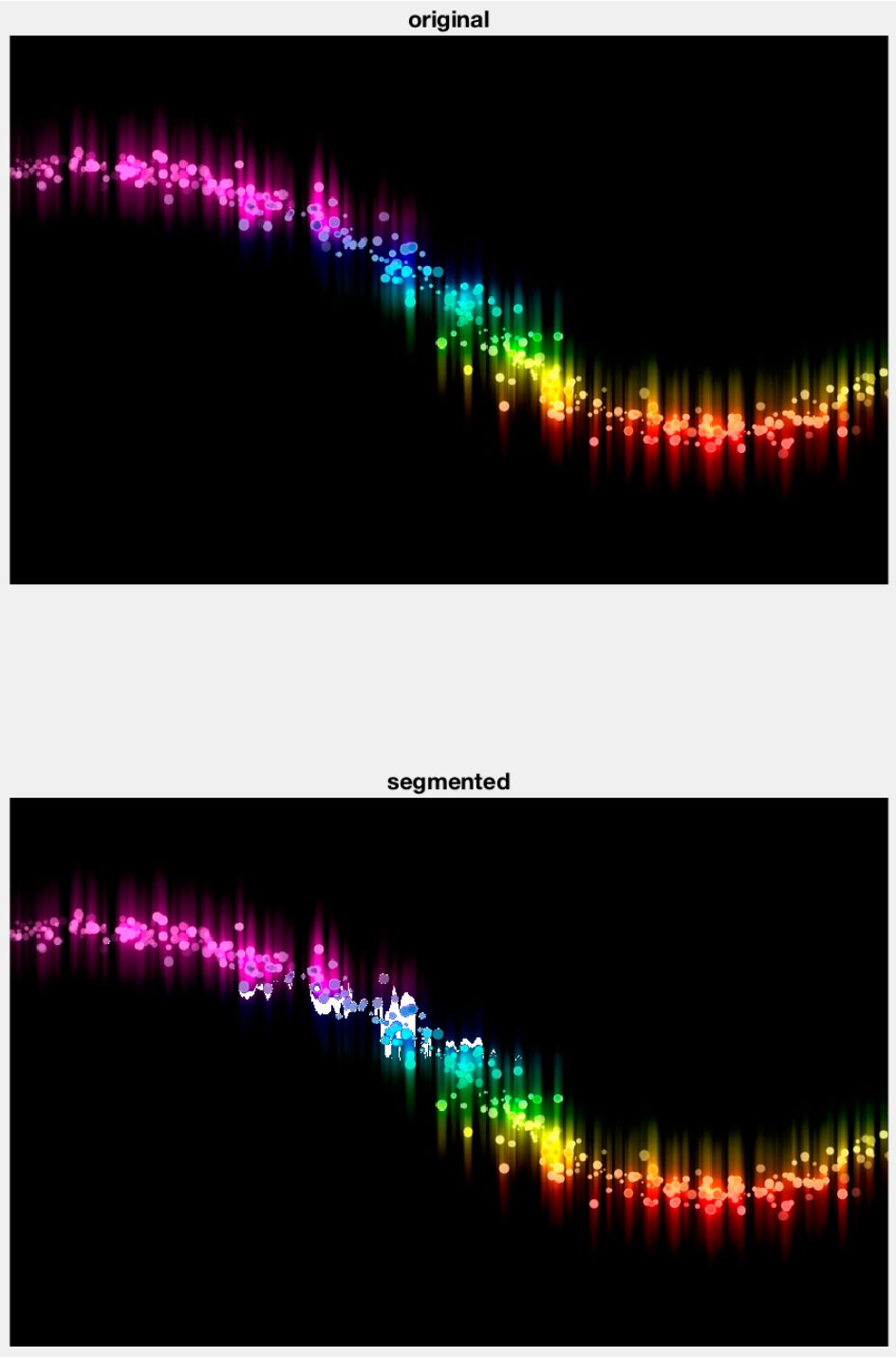
```
>> r = A(:,:,1); g = A(:,:,2); b = A(:,:,3);
>> rule = (r<rthresh*255 & g>gthresh*255 & b<bthresh*255);
>> r(rule) = 0; g(rule) = 0; b(rule) = 255;
>> green_region = cat(3, r, g, b);
>> subplot(2, 1, 1); imshow(A); title('original'); subplot(2, 1, 2); imshow(green_region); title('segmented');
```



As shown above, the objects that have “pure” green color have been successfully segmented out and turned into “pure” blue.

Segment the Blue Component

```
>> r = A(:,:,1); g = A(:,:,2); b = A(:,:,3);
>> rule = (r<rthresh*255 & g<gthresh*255 & b>bthresh*255);
>> r(rule) = 255; g(rule) = 255; b(rule) = 255;
>> blue_region = cat(3, r, g, b);
>> subplot(2, 1, 1); imshow(A); title('original'); subplot(2, 1, 2); imshow(blue_region); title('segmented');
```

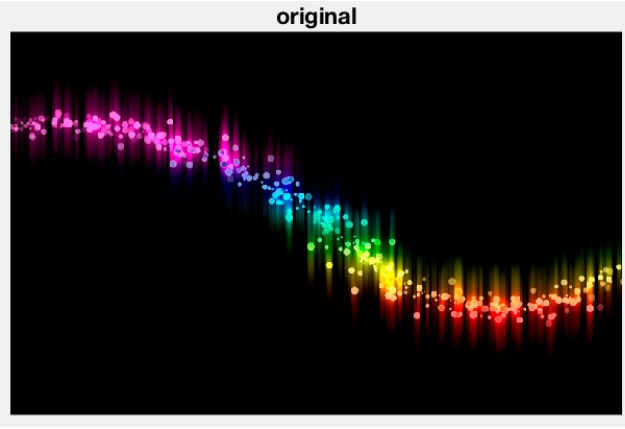


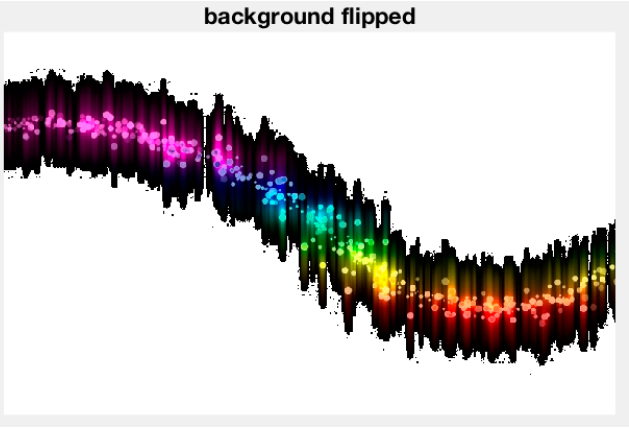
As shown above, objects that have “pure” blue color have been turned into white color. However, we see that part of the fading area between the background the object has been turned into white as well because in the blue layer of the original image, those regions have a high value and be identified as blue even though they are a mix of all three color.

c) Repeat the segmentation using thresholds in HSI space

Change Background Color

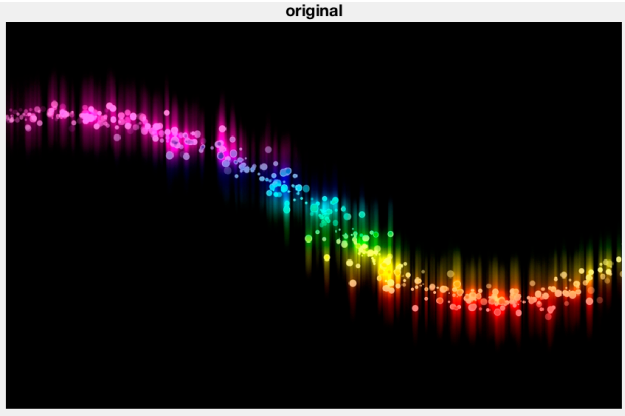
```
>> B = rgb2hsv(A);
>> I = B(:,:,3);
>> I(I == 0) = 1;
>> C = hsv2rgb(cat(3,B(:,:,1),B(:,:,2),I));
```

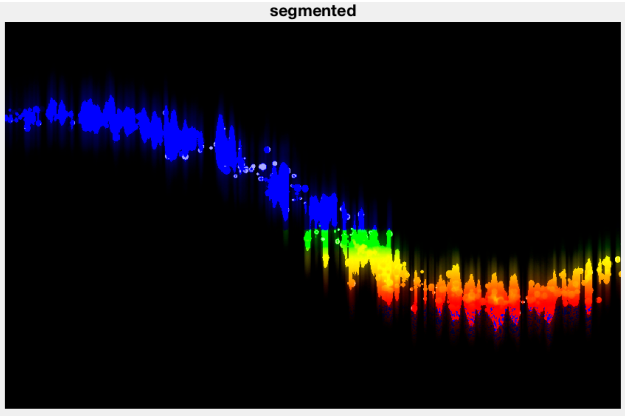




Segment The Object

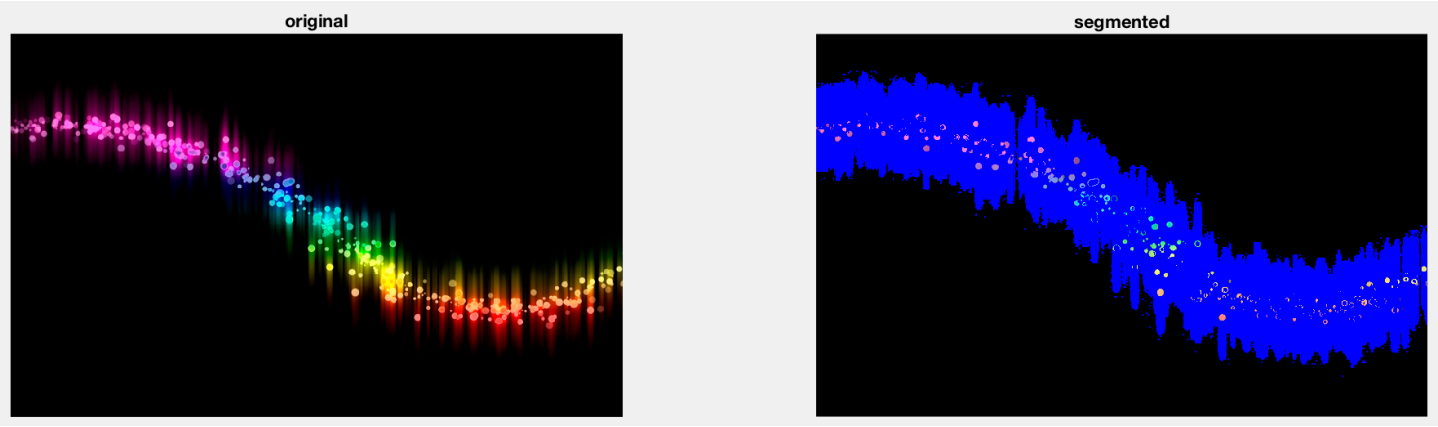
```
>> hue = B(:,:,1); satu = B(:,:,2); inte = B(:,:,3);
>> [hue_counts, hx] = imhist(hue);
>> [satu_counts, sx] = imhist(satu);
>> [inte_counts, ix] = imhist(inte);
>> rthresh = otsuthresh(red_counts);
>> hthresh = otsuthresh(hue_counts);
>> sthresh = otsuthresh(satu_counts);
>> ithresh = otsuthresh(inte_counts);
>> hue(hue>hthresh) = 2/3; % turn the segmented object into blue
>> satu(satu>hthresh) = 1;
>> inte(inte>ithresh) = 1;
>> seg = cat(3, hue, satu, inte);
>> seg_rgb = hsv2rgb(seg);
>> subplot(1, 2, 1); imshow(A); title('original'); subplot(1, 2, 2); imshow(seg_rgb); title('segmented');
```





As shown above, in the HSI space, it failed to segment the entire object since it only blued out half of the object into blue. In order to segment the entire object, manual configuration of selecting the intervals should be performed. Here is my attempt:

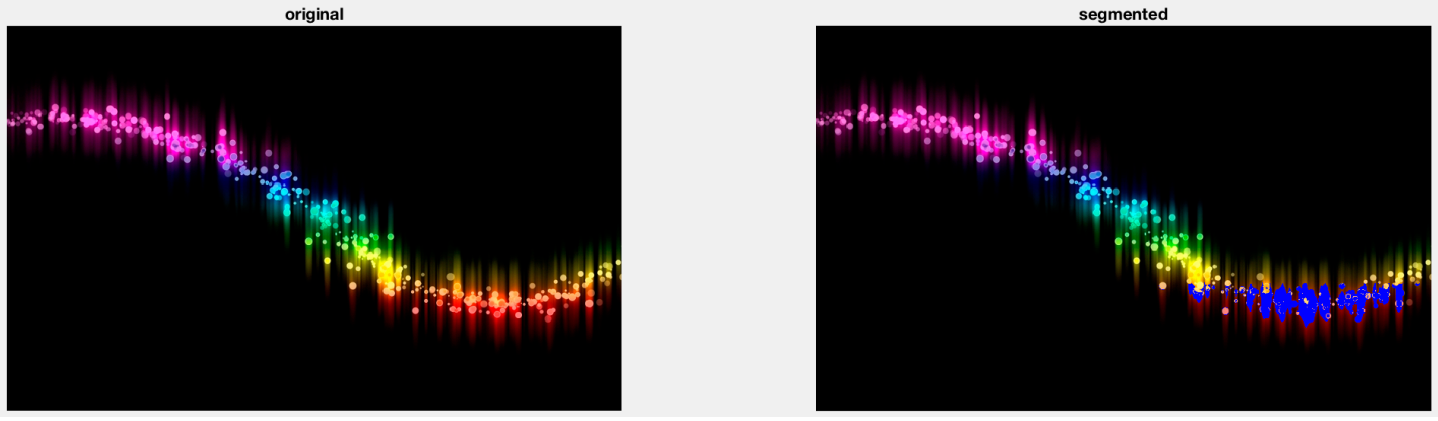
```
>> rule = satu > 0.5 & inte ~= 0;
>> hue = B(:,:,1); satu = B(:,:,2); inte = B(:,:,3);
>> hue(rule) = 2/3; satu(rule) = 1; inte(rule) = 1;
>> test = cat(3, hue, satu, inte);
>> test2 = hsv2rgb(test);
>> subplot(1, 2, 1); imshow(A); title('original'); subplot(1, 2, 2); imshow(test2); title('segmented');
```



As shown above, the parameters I chose over segment the object and it also has some interesting patterns in the middle of the blued out region. To capture the objects themselves, fine tune of the parameters that are used to segment the color should be performed. In general, in this case, segmentation of objects has a better performance in the RGB space than in the HSI space.

Segment Red Region

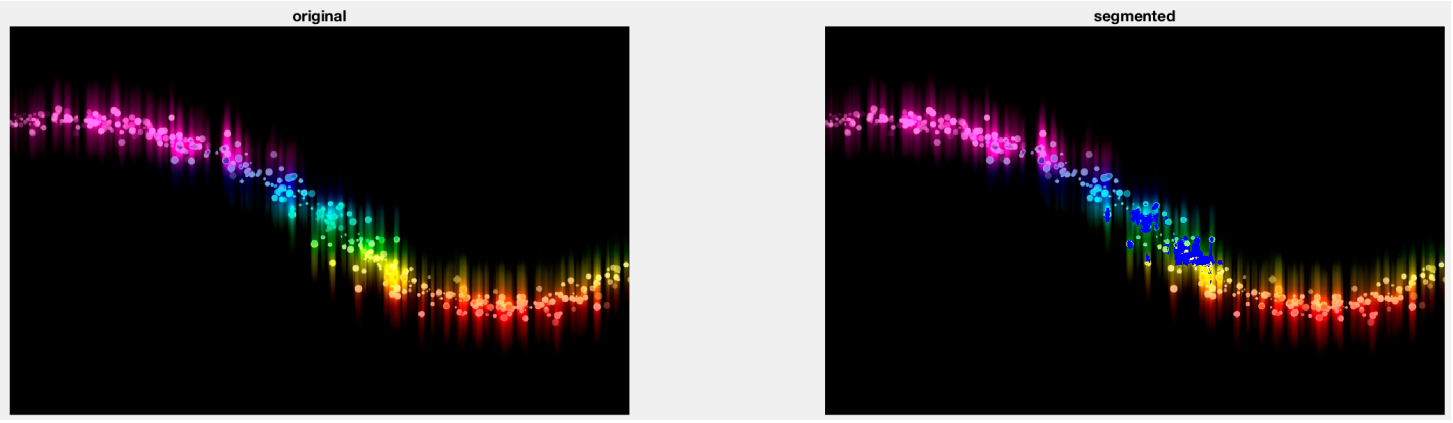
```
>> rule = (hue < 0.1 | hue > 0.9) & satu > 0.5 & inte > 0.5;
>> hue(rule) = 2/3;
>> satu(rule) = 1;
>> inte(rule) = 1;
>> red_region = hsv2rgb(cat(3, hue, satu, inte));
>> subplot(1, 2, 1); imshow(A); title('original'); subplot(1, 2, 2); imshow(red_region); title('segmented');
```



As shown above, the region where has apparent red pigments have been blued out.

Segment Green Region

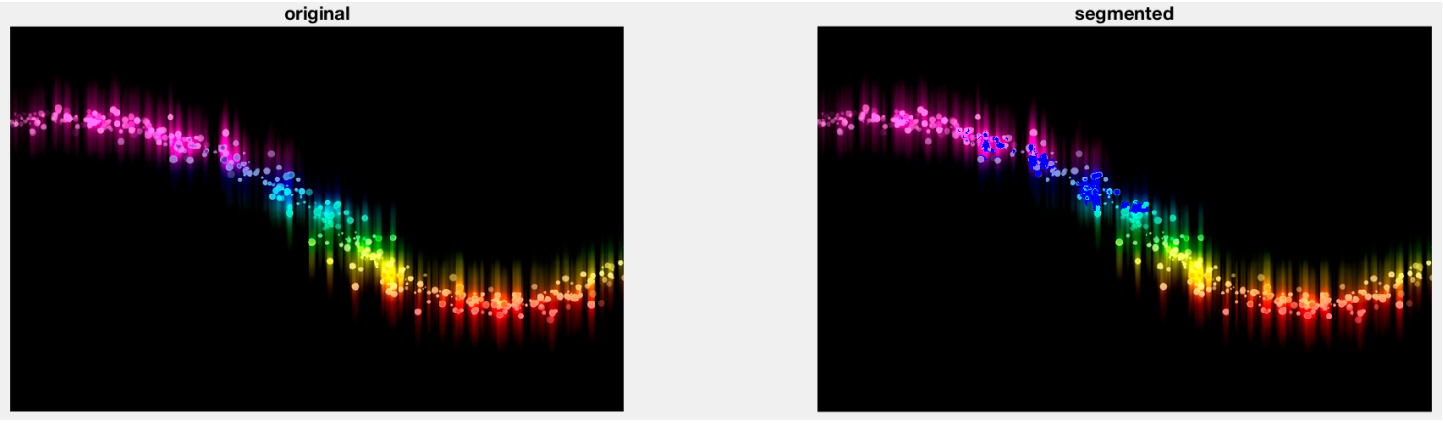
```
>> hue = B(:,:,1); satu = B(:,:,2); inte = B(:,:,3);
>> rule = (hue>1/6 & hue<3/6) & satu > 0.5 & inte > 0.5;
>> hue(rule) = 2/3; satu(rule) = 1; inte(rule) = 1;
>> green_region = hsv2rgb(cat(3, hue, satu, inte));
>> subplot(1, 2, 1); imshow(A); title('original'); subplot(1, 2, 2); imshow(green_region); title('segmented');
```



As shown above, the green region has been blued out.

Segment Blue Region

```
>> hue = B(:,:,1); satu = B(:,:,2); inte = B(:,:,3);
>> rule = (hue>3/6 & hue<5/6) & satu > 0.5 & inte > 0.5;
>> hue(rule) = 2/3; satu(rule) = 1; inte(rule) = 1;
>> blue_region = hsv2rgb(cat(3, hue, satu, inte));
>> subplot(1, 2, 1); imshow(A); title('original'); subplot(1, 2, 2); imshow(blue_region); title('segmented');
```



The blue region has been blued out as well (they are turned into dark blue).