Sebastian Zdarowski

October 27, 2019

Assignment 4 (50 points)

From the text:

481 Students: 5.1 (3/0) “The white bars in the test pattern shown are 7 pixels wide…”

Arithmetic mean filter is the simplest of the mean filters, a mean filter will smooths local variation in an image and noise is reduced as a result of blurring.

A) 3x3 arithmetic mean filter

Ans: Once you apply an 3x3 filter on the image in question. The white bars will become blurred, the edges of the white bar will not be as defined or sharp within the original image. There will be a column of gray pixels between the black background and white bars.

B) 7x7 arithmetic mean filter

Ans: By doing a 7x7 filter on the image, the image will be blurry than the previous 3x3 filter mask we have applied on the image. The edges of the white bars will lose their definition by turning gray. White bars will start to turn even more gray as we apply the mean filter throughout the image.

C) 9x9 arithmetic mean filter

Ans: By doing a 9x9 filter on the image, the image will be blurry than the previous filter of 7x7, there would be no white pixels left in the image, since we have averaged the pixels and by mixing black and white values we will see gray bars instead of white bars within the image.

481 Students: 5.6 (3/0) “Repeat problem 5.1 using a median filter.”

Median Filter which as its name implies, replaces the value of a pixel by the median of the intensity levels in a predefined neighborhood of that pixel. Provide excellent noise reduction capabilities with considerable less blurring than linear smoothing filters of similar size.

A) 3x3 median filter

Ans: No significant change would happen within the image, besides the image being rounded as this filter does less blurring than the mean filter. The white bars would stay the same.

B) 7x7 median filter

Ans: Image would be more rounded than the previous filter, white bars would be more pronounced.

C) 9x9 median filter

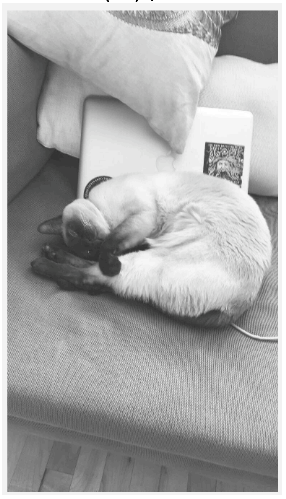
Ans: White bars would be even more pronounced in this filter than the previous image. Since the filters purpose is not to blur, it will sharpen the white bars more so than the previous filter.

Programming:

Problem 1: Edge Detection (15/20)

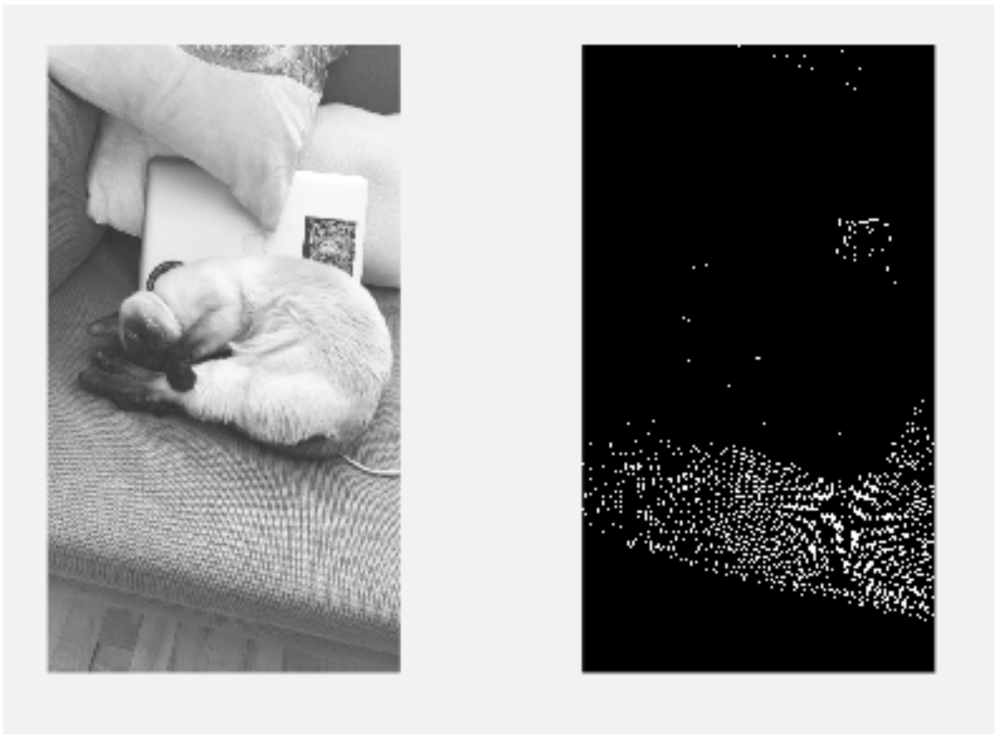
Use the `edge` function to generate results for Roberts, Canny, Sobel, and Prewitt operators on an image of your choice. Note also that the various edge functions support a number of parameters – feel free to explore those to get more interesting results. State which operator gives the best performance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

```
X = imread('collins4.jpeg');
Y = rgb2gray(X);
imshow(Y);
```



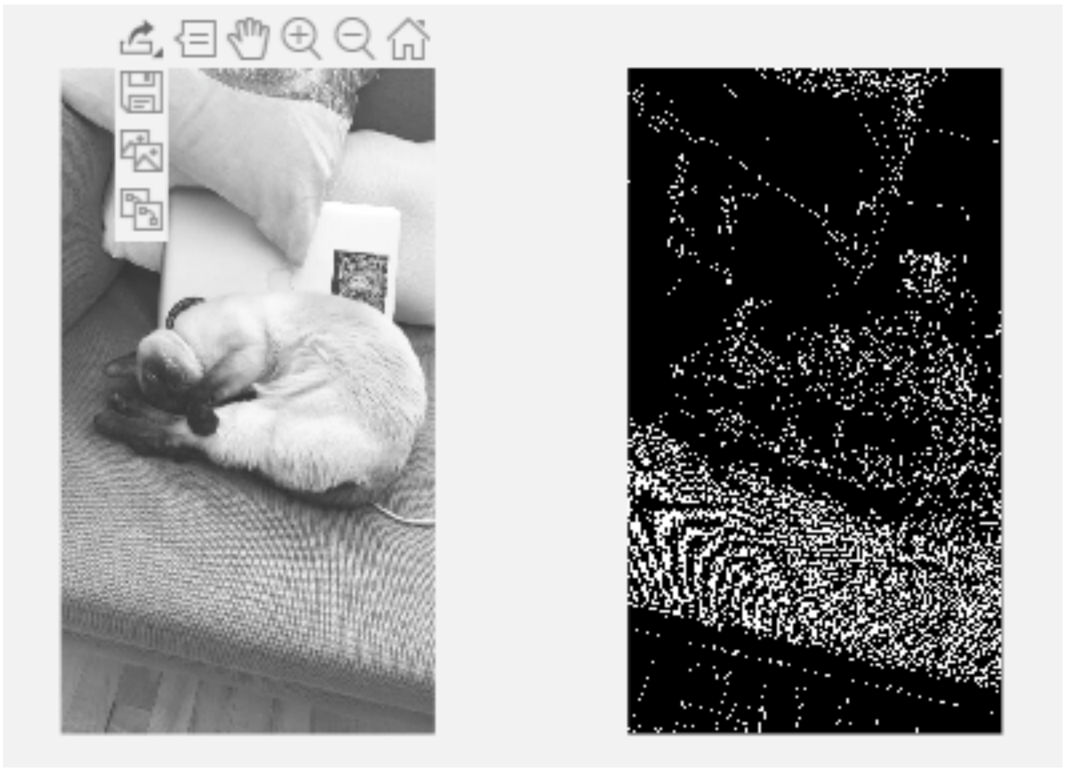
Roberts edge function:

```
robert = edge(Y,'Roberts')
subplot(2,2,1);
imshow(Y);
subplot(2,2,2);
imshow(robert);
```



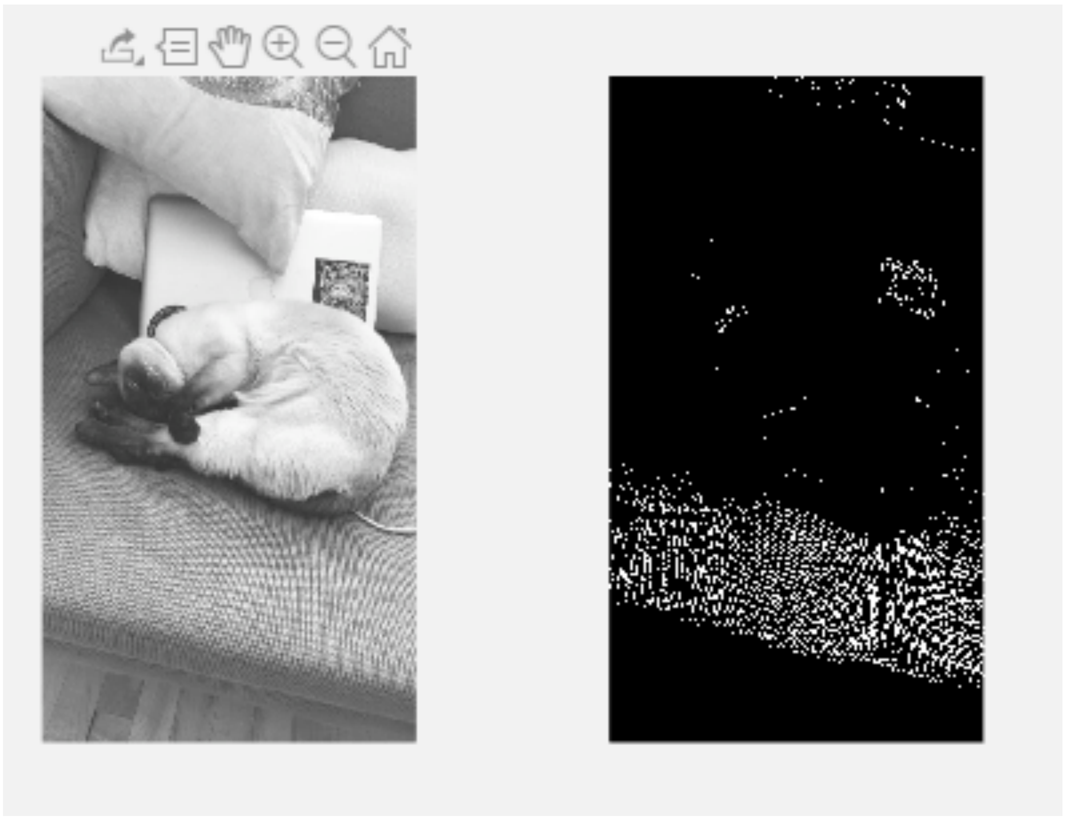
Canny edge function:

```
canny = edge(Y,'Canny')
subplot(2,2,1);
imshow(Y);
subplot(2,2,2);
imshow(canny);
```



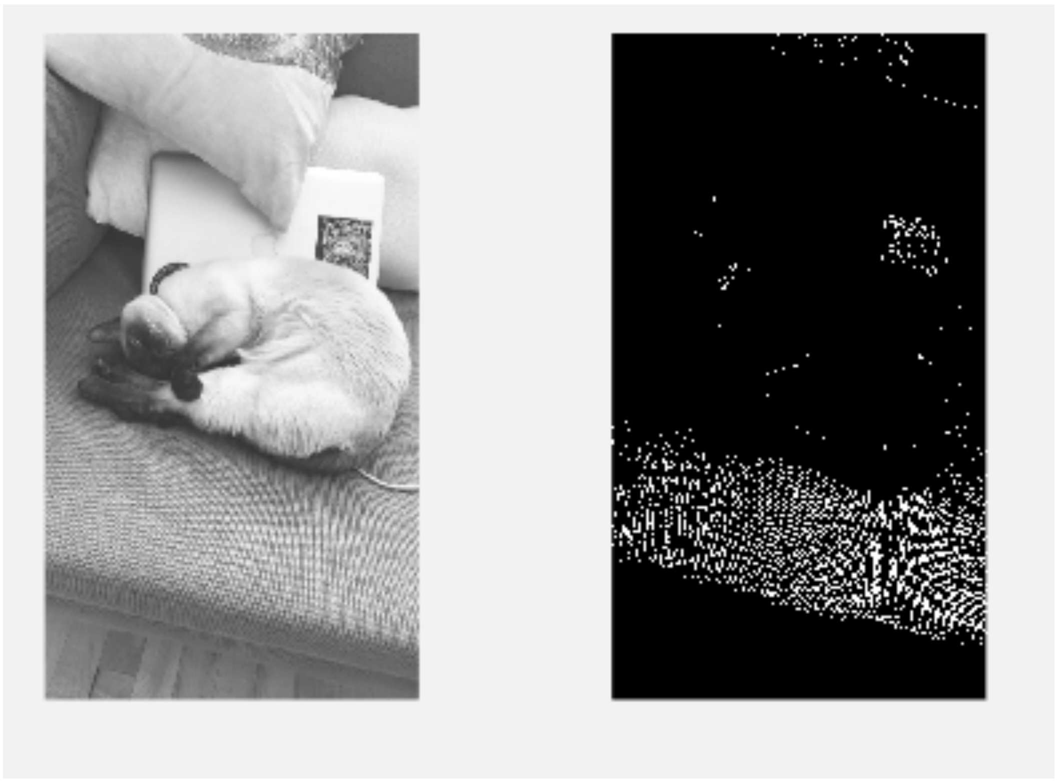
Sobel Edge Function:

```
sobel = edge(Y,'Sobel')
subplot(2,2,1);
imshow(Y);
subplot(2,2,2);
imshow(sobel);
```



Prewitt Edge function:

```
pret = edge(Y,'Prewitt')
subplot(2,2,1);
imshow(Y);
subplot(2,2,2);
imshow(pret);
```



Ans: I don’t think my image was best suited for these edge functions, but to find edges I think that all of functions did poorly to find the edges except the canny function, it did pick up on a lot more edges than the other functions but I feel that it did give us too much information. I was able to distinguish the couch, the couch pillow, and the cat itself. But additional information kind made it too noisy, like picking up the wood floor patterns. The other 3 functions mainly picked up on the couch texture and wherever the cat was located in the photo it was black.

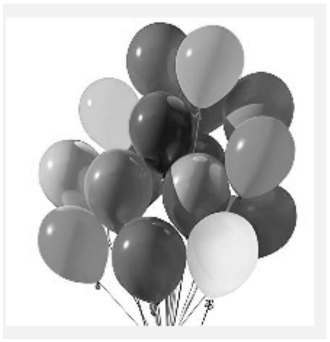
Problem 2: Color Segmentation

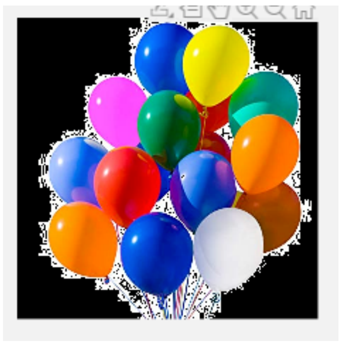
A natural cue to use in segmenting objects from their surroundings in images is color. **This problem contrasts segmenting color regions using red, green, and blue thresholds** (aligned with the RGB axes of color space) **with segmentation using hue, saturation, and intensity bounds** (aligned with the 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.

```
X = imread('ballon.jpg');
Y = rgb2gray(X);
imshow(X);
```

a) **(5/10)** 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. (In fact, just the reverse -- replacing blue or green pixels with those of some preset image -- is the technique used in TV or movies to superimpose objects against some preset background. Since thresholding is used, this (and not fashion) is why so few weathercasters wear saturated blue items. This technique is called travelling matte. Blue is good because it turns black under a red filter; green is good because most digital cameras have less noise in the green channel. Matte techniques have become even more sophisticated as digital video becomes more sophisticated).

```
red = X(:,:,1);
green = X(:,:,2);
blue = X(:,:,3);

gray = red/3 + green/3 + blue/3;
imshow(gray);
```





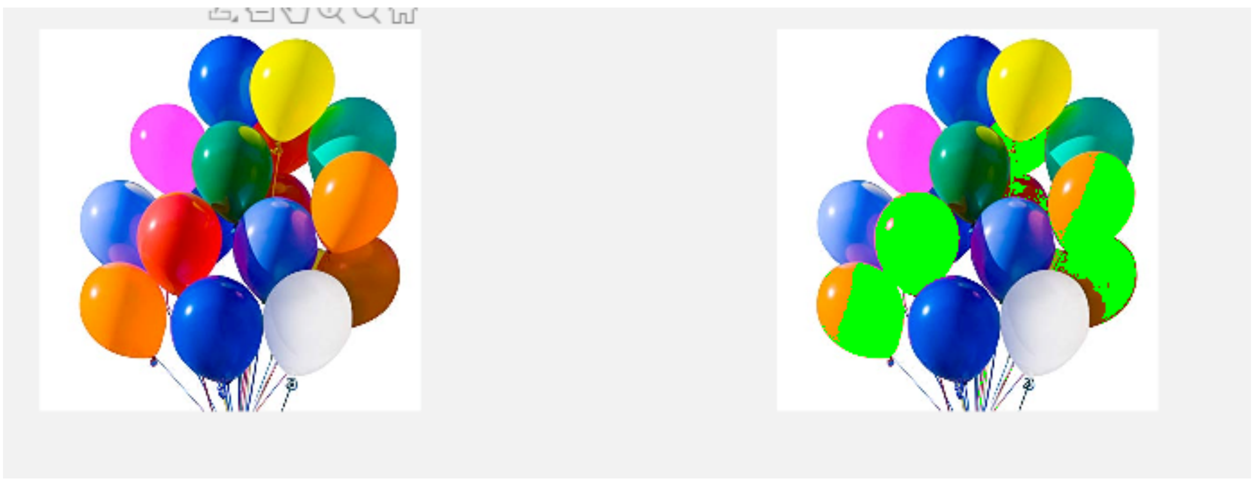
```
black = gray == 255;

red(black) = 0;
green(black) = 0;
blue(black) = 0;

flip = cat(3,red,green,blue);
imshow(flip);
```

b) **(5/10)** 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`.)

```
    %grab histogram of each color
    [count_red] = imhist(red);
    [count_green] = imhist(green);
    [count_blue] = imhist(blue);


%Set up automatic threshold
Redthresh = otsuthresh(count_red);
Greenthresh = otsuthresh(count_red);
Bluethresh = otsuthresh(count_red);
```

Red segmentation:

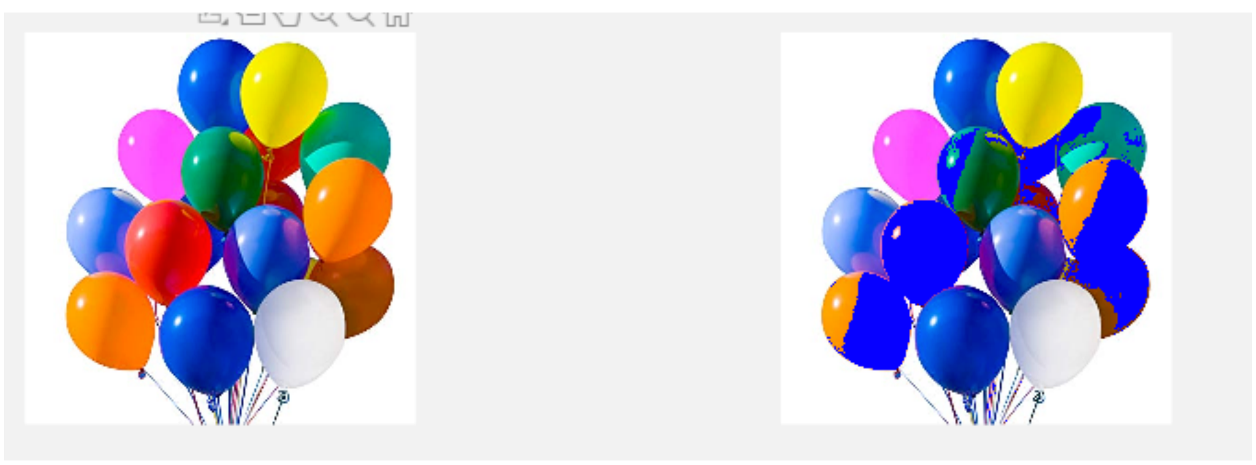
```
%%%red segmentation:
segment = (red>Redthresh*255 & green<Greenthresh*255 & blue<Bluethresh*255);
red(segment) = 0;
green(segment) = 255;
blue(segment) = 0;

red_seg = cat(3,red,green,blue);
imshow(red_seg);
```



Red segmentation picked up on the red balloons fairly well, we did pick up the shaded orange half of the balloon and I think the balloon below it also was orange but it was shaded in the image, so it was picked up as a red color.

Green Segmentation:

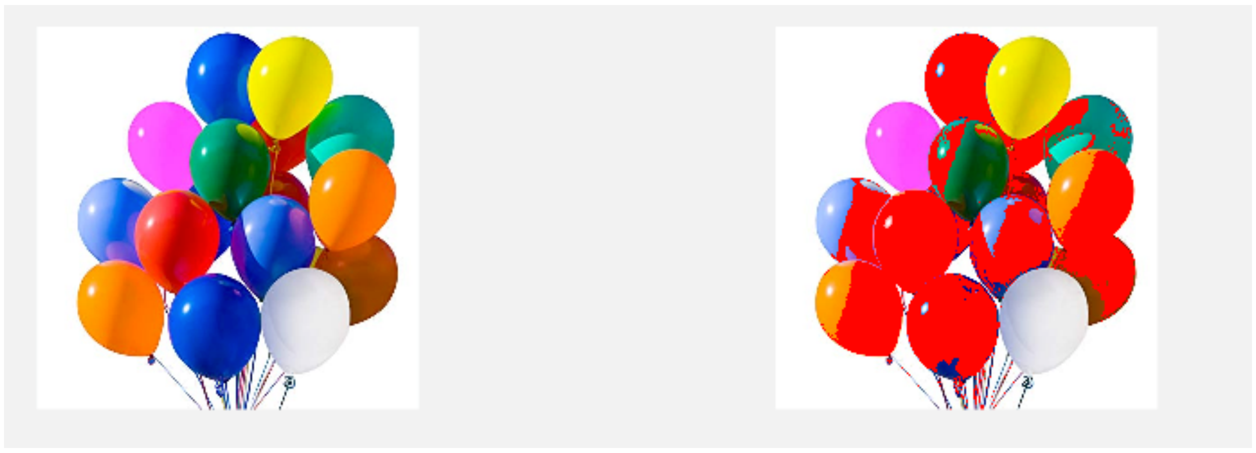
```
%%%Green segmentation:
segment = (red<Redthresh*255 & green>Greenthresh*255 & blue<Bluethresh*255);
red(segment) = 0;
green(segment) = 0;
blue(segment) = 255;

green_seg = cat(3,red,green,blue);

subplot(2,2,1);
imshow(X);
subplot(2,2,2);
imshow(green_seg);
```



Function captured the green balloons and filled them with blue coloring but also it captured the red balloons as well.

Blue Segmentation:

```
%%%Blue segmentation:
segment = (red<Redthresh*255 & green<Greenthresh*255 & blue>Bluethresh*255);
red(segment) = 255;
green(segment) = 0;
blue(segment) = 0;

blue_seg = cat(3,red,green,blue);

subplot(2,2,1);
imshow(X);
subplot(2,2,2);
imshow(blue_seg);
```



Blue segment was captured within the image, you can see that the blue balloons have been filled in with red, its interesting to see that the other colors of green and red were also being captured by this segmentation.

c) **(5/10)** Repeat the segmentation using thresholds in HSI space. MatLab has a function for converting from RGB to HSI (MatLab calls it HSV):

```
B = rgb2hsv(A);
```

Note: When described in matlab's HSV space, the hue (first component) of a pixel ranges from 0.0 (red) to 1.0 (red again), passing through orange, yellow, green, cyan, blue, purple, and magenta along the way. The second component, saturation, varies from 0.0 (grayscale) to 1.0 (completely saturated -- no white at all). The final component, intensity (or "value"), also ranges from 0.0 (no intensity) to 1.0 (max intensity).

There is also an inverse function

```
A = hsv2rgb(B);
```

It returns an RGB image with pixel components between 0.0 and 1.0

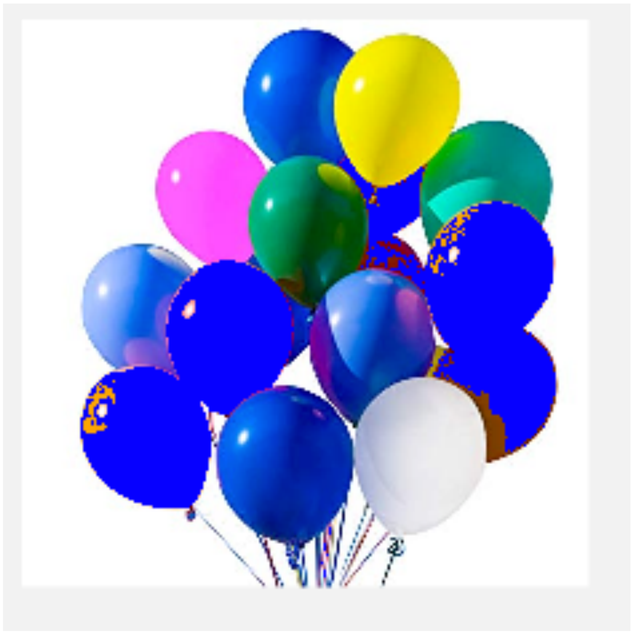
Note: A "threshold" of the hue component of pixels must be an interval, because the hue actually wraps around and is best envisioned as a circle. Thus, to segment a blue region, you need to accept only hues around 2/3 (0 = red, 1/3 = green, 2/3 = blue).

```
X = imread('ballon.jpg');
Y = rgb2gray(X);
imshow(X);

Y = rgb2hsv(X);

Hue = Y(:,:,1);
Sat = Y(:,:,2);
Inten = Y(:,:,3);
```

Red Segmentation:

```
%%%% Segment out the red
Segment=(Hue<0.1|Hue >0.9) & Sat >0.5 & Inten >0.5;

Hue(Segment) = 2/3
Sat(Segment) = 1;
Inten(Segment) =1;
```

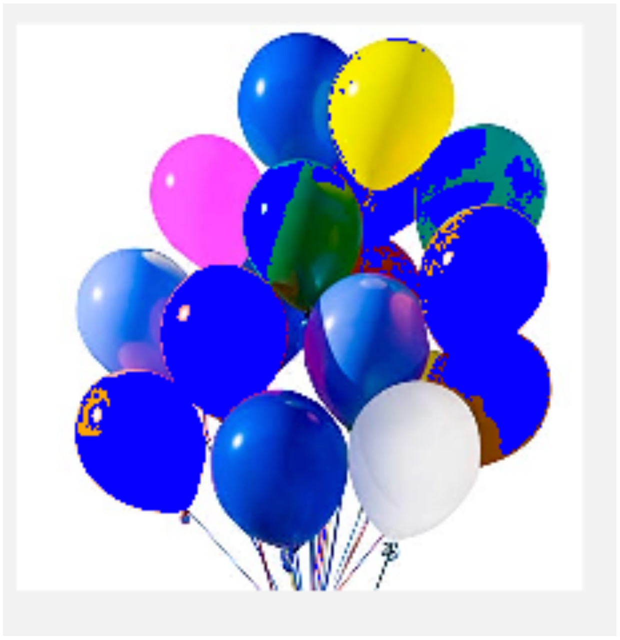
```
red_seg = hsv2rgb(cat(3,Hue,Sat,Inten));
imshow(red_seg);
```



Using the HSI space for segmentation seems to be working better for me than using it in the RGB space, I was able to fully capture the colors of red and orange balloons pretty well.

Green Segmentation:

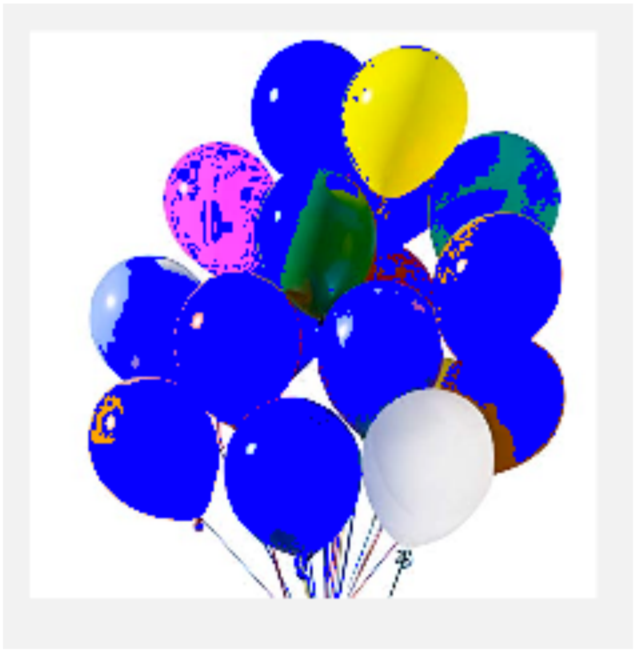
```
%%%% Segment out the green
Segment=(Hue > 1/6 & Hue<3/6) & Sat >0.5 & Inten >0.5;

Hue(Segment) = 2/3
Sat(Segment) = 1;
Inten(Segment) =1;

green_seg = hsv2rgb(cat(3,Hue,Sat,Inten));
imshow(green_seg);
```



Not capturing the green balloons all the way, but also picking up some of the red balloons as well. Seems to have failed, since the red balloons were included.

Blue Segmentation:

```
%%%% Segment out the Blue
Segment=(Hue > 3/6 & Hue<5/6) & Sat >0.5 & Inten >0.5;

Hue(Segment) = 2/3
Sat(Segment) = 1;
Inten(Segment) =1;

blue_seg = hsv2rgb(cat(3,Hue,Sat,Inten));
imshow(blue_seg);
```



This segmentation did capture the blue balloons for the most part. It seemed to pick up on the red and orange balloons. Seems to be a trend with the segmentation that I have been doing throughout this assignment. My Red segmentation works fairly well, but when trying to segment greens and blues, I always seem to bring in the red values as well.

General submission instructions:

(a) Be kind to your aging, over-worked professor and submit only a single document. This can be pdf, MS Word, OpenOffice, etc. Do not submit a zip file.

(b) Your single document should include the input image for your problem, if required, and answers to each of the sub-problems (text, image or both, as appropriate). Your document should also include code that you wrote to generate your answers.

(c) You may use any images you like for the programming; I encourage you to use images that might be useful/interesting for your final project.

(d) Feel free to use whatever functions MatLab supplies, except where otherwise specified. Also feel free to write your own, if you are so inclined; it will take more time, but you will gain a deeper understanding of the material. It is one thing, for example, to implement Otsu thresholding using `otsuthresh`, quite another to write an thresholding technique yourself.

(e) Point values for each question are indicated as ***x/y*** in which ***x*** is the point value for 481 students and ***y*** is the point value for 381 students.