Jacob Oakes
Lab 2

Original Image Used

Reference: http://www.fanpop.com/clubs/roses/images/9842259/title/yellow-roses-photo

Actions Performed

Cut the values of all bands in half

Prediction	The picture will become darker
Actual	The picture became darker since all of the color values were halved.

Double all the bands

Prediction	The picture will become brighter
Actual	The picture became brighter since all of the color values were doubled.

Exchange the green and blue bytes

Prediction	No effect
Actual	This change was surprising to me, however after thinking it through it began to make sense. The yellows became a pinkish color. Since the bands were switched I am guessing the yellows were changed to their complementary color. Looking at a color wheel you can see that opposite of yellow is purple/pink which would be its complementary color.

Double all blue bands

Prediction	The picture will become bluer
Actual	Most of the picture became bluer especially the dark and darker green areas. The yellows were not as affected since they do not have a lot of blue in them.

Double only the blue bands under 64

Prediction	Only parts of the image will become bluer. The mostly blue pixels will not become bluer though.
Actual	Parts of the image did become bluer. The yellows were not really affected at all. Some of the greens became more blueish green. This was not as drastic of a change as doubling the entire blue band.

Converting to Other Color Spaces

H

range is 0 - 0.9988

S

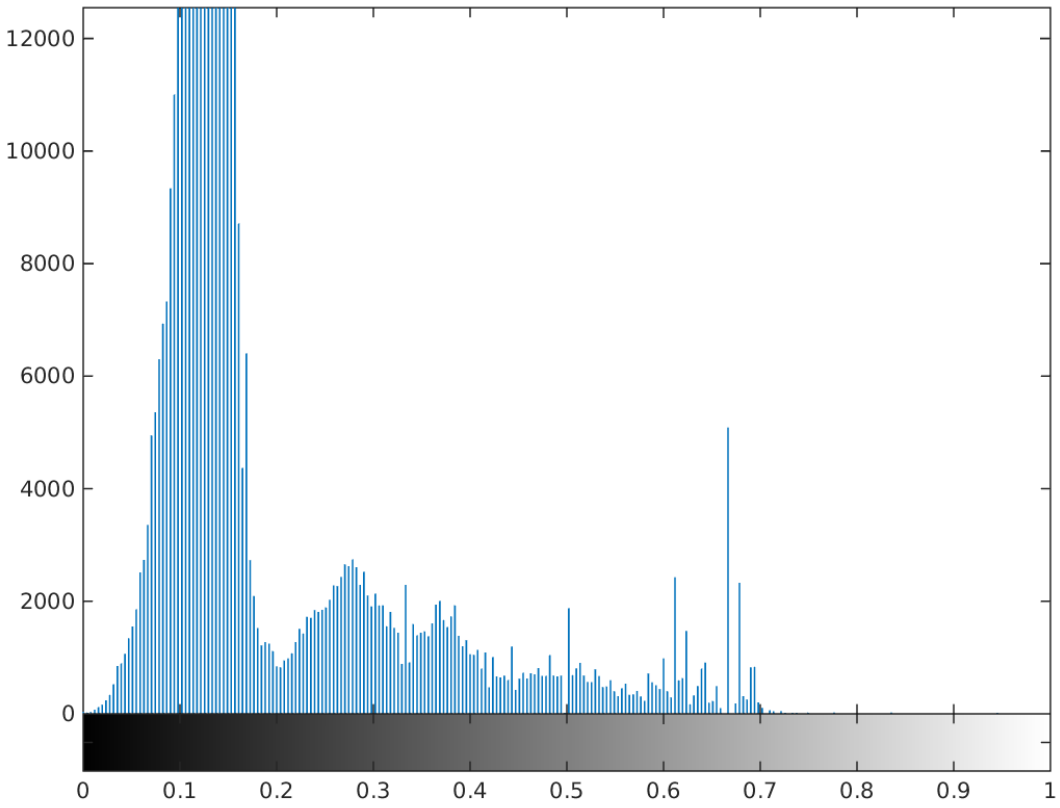
range is 0 - 1

V

range is 0.0471 - 1

HSV

Hue Histogram



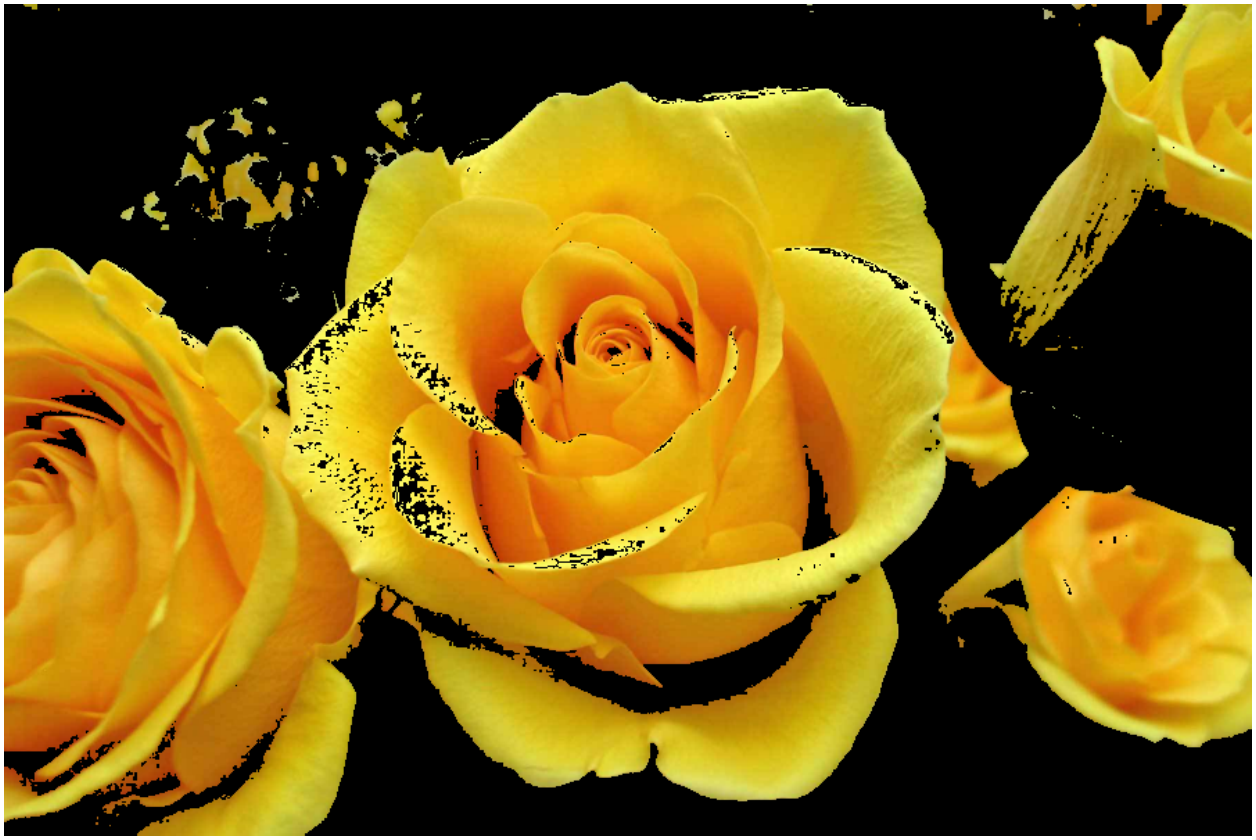
Most of the pixels have the value from a range of 0.1 to about 0.15. This range of hues corresponds yellow. This is valid since most of the picture is yellow. There are very little in the range from .7 to 1. This range of hues corresponds to blue, purple, pink, and red colors. Looking at the picture there are not very many of these colors.

Finding and Operating on Masks

Mask 1

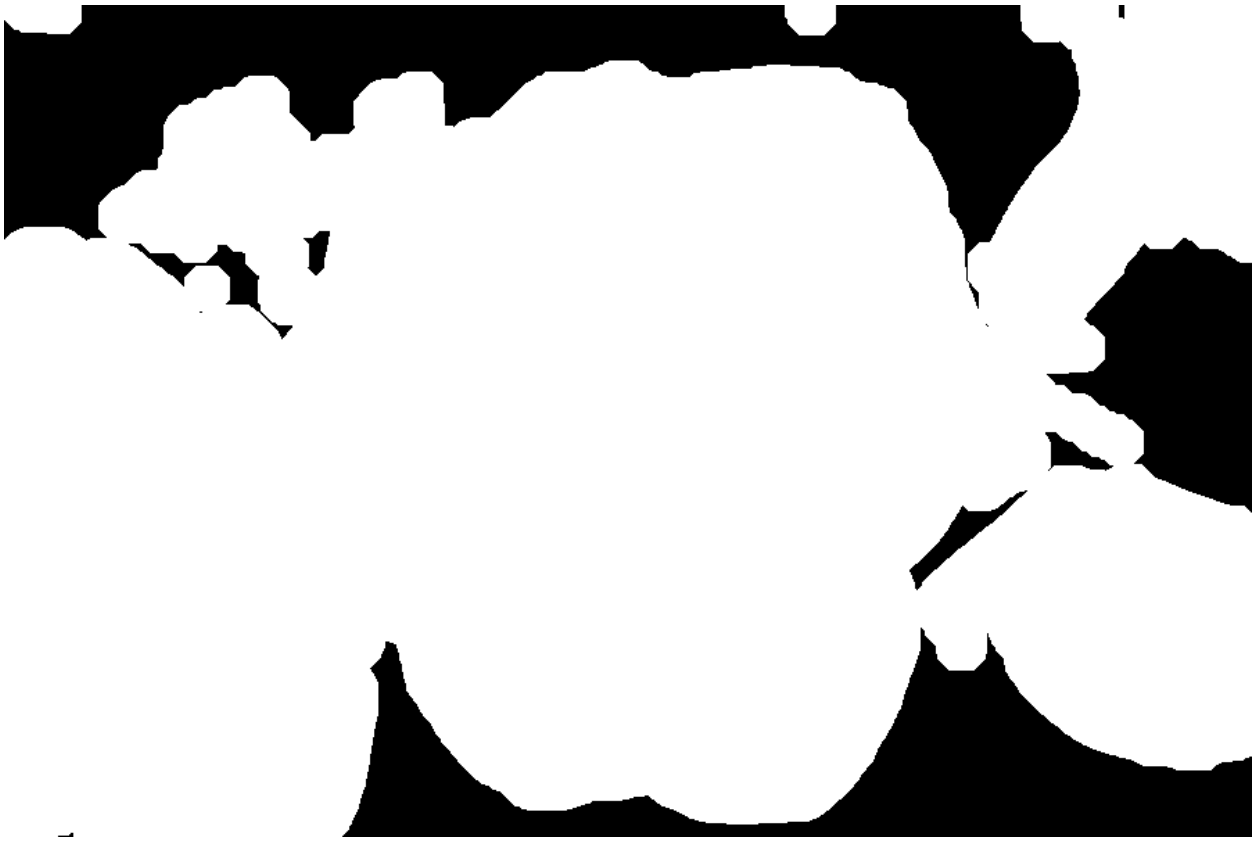
Binary Mask

Mask Applied to Original Image



Mask 2

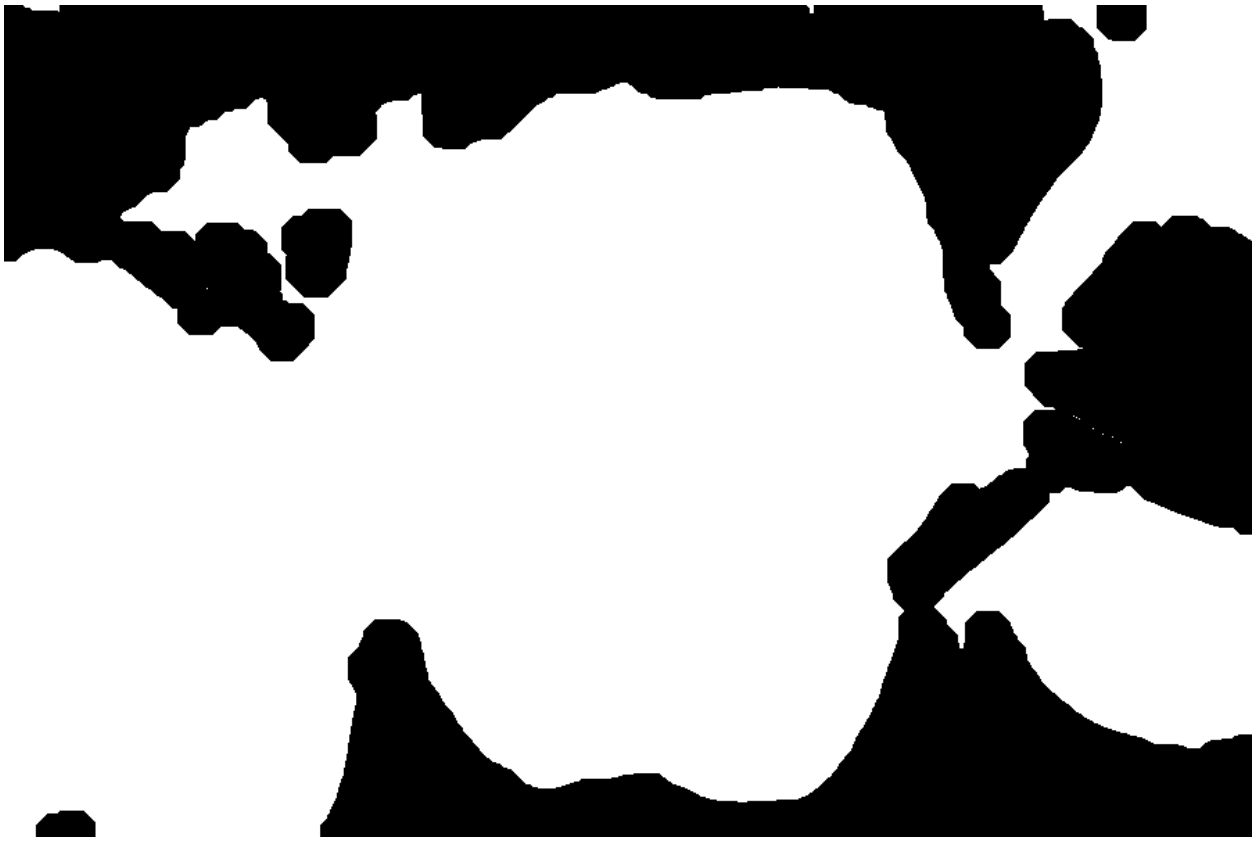
Binary Mask



Mask Applied to Original Image

Mask 3

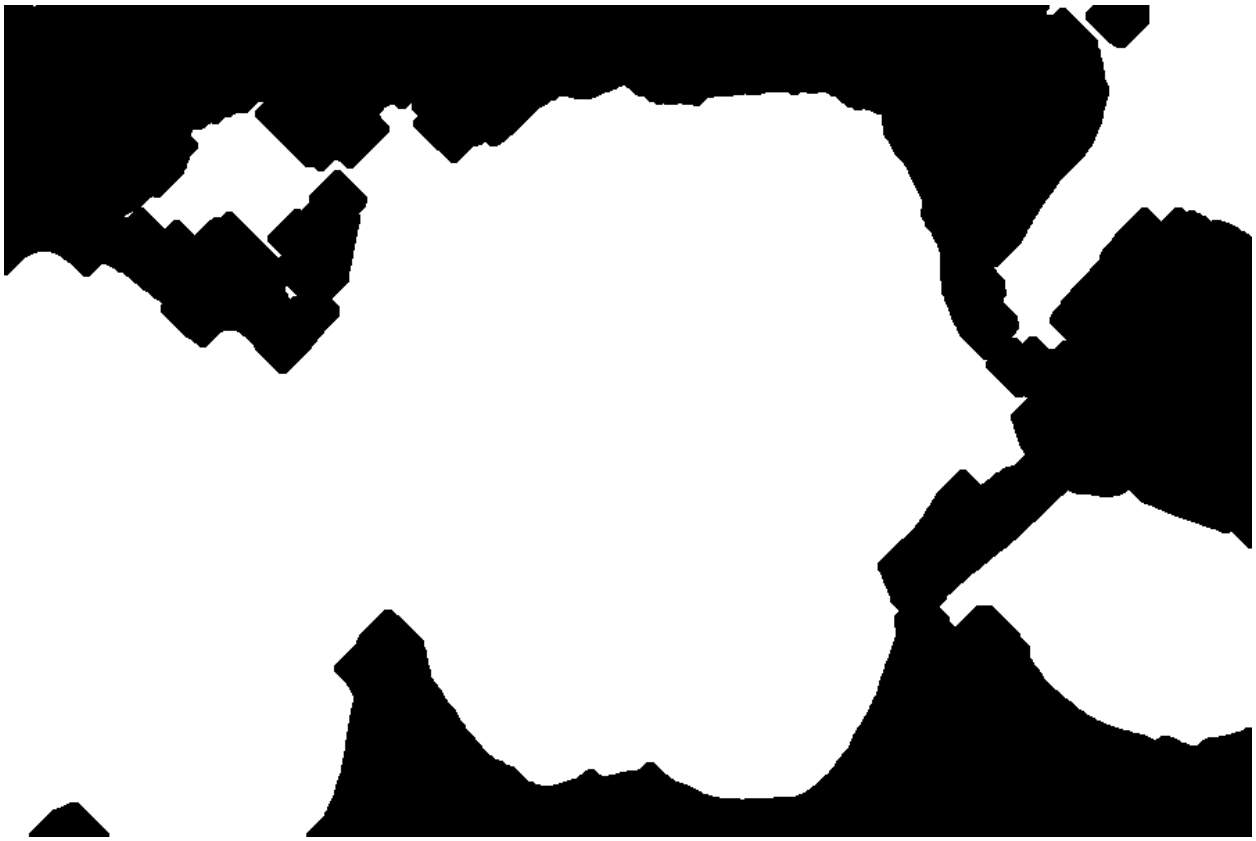
Binary Mask



Mask Applied to Original Image

Mask 4

Binary Mask



Mask Applied to Original Image

Description

The mask was found by the colors of yellows for the roses. A range of 170 - 255 for red, 78 - 255 for green and 0 to 130 for blue was used to find these colors. It was fairly easy to find this range. I used imtool to find the color values for lightest and darkest yellows for the middle rose. I rounded these rgb values to the nearest multiple of 5 which became the ranges used above.

There were a total of 540,000 pixels in the picture. The number of pixels found for each binary masked and the operations used to find them are as follows:

Mask 1
Size - 319,047 pixels
This was the original mask which only uses the rgb range specified above.

Mask 2
Size - 422,076 pixels
This mask was found by dilating the original using a disk of size 17. The size was chosen just big enough to fill in all parts of the center rose.

Mask 3
Size - 353,577 pixels
This mask was found by dilating and then eroding the original using a disk of size 17. This was used just to see how effective imclose would be in this situation. Overall I thought the results from this were very nice with the exception of a few stray pixels.

Mask 4
Size - 339,170 pixels
This mask was found with multiple operations. First the original was dilated using a diamond of size 17. Then it was eroded by a diamond of size 15. Finally it was eroded by a disk of size 5 to give more rounded edges. This was fairly similar to the third mask, however it got rid of more of the picture that was not the roses.

Matlab Quick Reference Guide:

creating a 3D array:

```
>> A(:,:,1) = [1 2; 3 4]
>> A(:,:,2) = [5 6; 7 8]
>> A(:,:,3) = [9 10; 11 12]
```

if statement:

```
if true
    do something
else
    do else
end
```

for loop:

```
for j = 1: 10
    count = count + 1;
end
```

images:

```
img = imread('image.jpg')
imtool(img)
imshow(img)
imwrite(img, 'image.png')          saves image to a file
imhist(img)                        histogram of the image
imresize(img, 0.5, 'bicubic');     cuts the size in half
```

color:

```
red = img(:,:,1)
green = img(:,:,2)
blue = img(:,:,3)
gray = rgb2gray(img)
```

example mask doubling blues with value less than 64:

```
[row, col] = size(blue);
mask = ones(row, col);
mask(blue < 64) = 2;
newBlue = double(blue) .* mask;
doubleSomeBlue(:,:,3) = uint8(newBlue);
```

Applying a binary mask to a color image you must use repmat to include the 3 bands

```
img2 = double(img) .* repmat(mask, [1,1,3]);
```