LAB1: Introduction to Image Processing Using MATLAB

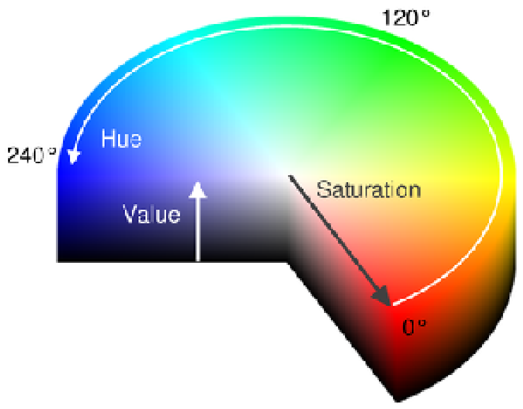
Updated: v1. 2015-10-17; v2. 10-25; v3. 2019-02-27 Module By: Prof. Min Wu (minwu@umd.edu); Version 3 Updated by: Nathaniel Ferlic & Kevin Ho;

Contents

O. Prelab

Your prelab submission should be in published .pdf format. Refer to the prelab from Module 4 and the Publishing Markup documentation page for reference.

Resources:



Question 1:

- What is the Hue of an image?
- What is the Saturation of an image?
- What is the Value of an image?

Question 2:

How do green screens work?

Before moving onto Question 3, make sure to download the image files for this lab here. You will be using these for the remainder of the prelab as well as the rest of Module 5 Lab 1.

O.I: Prelab: Loading Image and Display

- `imread( )`
- `imshow( )`. | | See also `image( )` and `imagesc( )`.

Tips: Learn to use `help` with a function name to learn usage and more details. Reference page can also be viewed in MATLAB's Help browser by `doc` with a function or topic name.

```
clear all   % Use this to clean up the workspace to prepare for this new lab
close all   % Close all figures if needed
img1 = imread('Flower.bmp');  % imread( ) can load images of many formats

figure(1), imshow(img1)
```

O.II. Prelab: Color Representation-1: RGB

- A digital image is a Two-Dimensional signal and described by the brightness or color of picture elements ("pixels") indexed by horizontal and vertical coordinates.
- By default, a color image is stored by MATLAB using 3 matrices, each representing red, green and blue components of pixels. In image/video processing, it is also referred to as R/G/B channels. A matrix is essentially an array indexed by two indexing variables typically for row and column.

Each of the three color matrices can be specified by the third indexing number -- note the number **1** in the example below `img(:,:,1)`, and so on. The horizontal and vertical dimensions of each color matrix matches the size of the image in unit of pixels. The overall color image is thus stored in an array of three indexing variables, as you can see in the summary of Workspace in MATLAB, or run the function `size`.

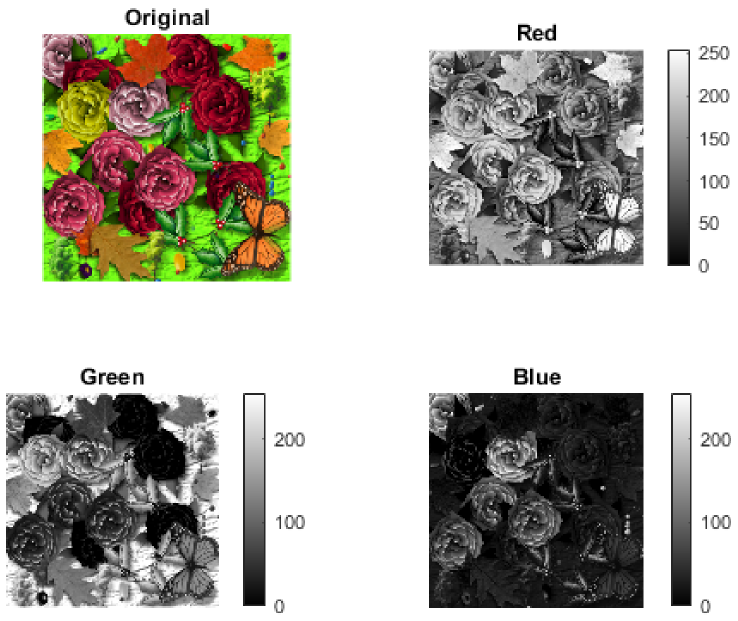
A R/G/B component is represented by dafault using 8 bits (unsigned 8-bit integer or 'uint8' in MATLAB data type. 8-bit representation gives a scale of 0 to 255, with 0 being the darkest and 255 the brightness in the particular color component.

Question 3: Return the size of image one using the `size()` function.

Question 4: Create a variable to read in the green color plane, `img1_g`, and another variable that reads in the blue color plane, `img1_b`. As an example the red color plane is loaded as `img1_r`.

```
img1_r = img1( :, :, 1 );
```

Question 5: Plot each color plane along with the original image in a 2x2 subplot to recreate the image shown below.

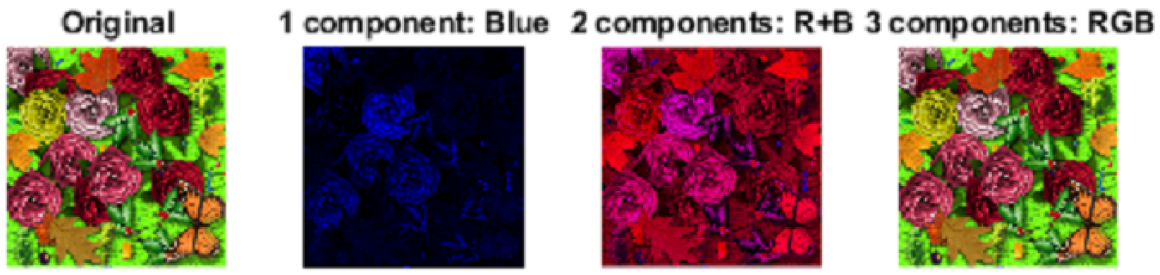


I. Lab: Color Representation-1: Merging RGB Components

In this section, you will continue to use the `Flower.bmp` image you had analyzed in your prelab. For Question 1, we attempt to recreate our image of a flower using each individual color plane.

Question 1: Produce the following plot below by merging each of the color planes. This can be done by defining an image variable, `img2`, where each of the color planes can be input to the 3rd indexing number. Merging RGB components:

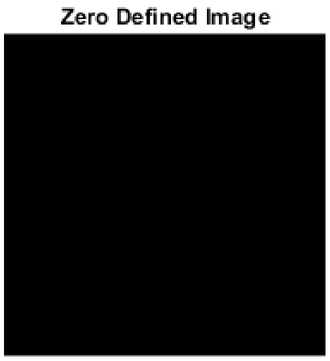
```
img2 = zeros( size(img1) );  %% initialize arrays of the same size as img1
img2 = uint8( img2 );   %% ensure correct data type of unsigned 8-bit integer
```

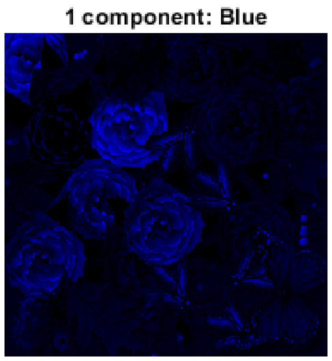


Hint: If only blue color plane is defined then the image will look as

```
img2 = zeros( size(img1) );  %% initialize arrays of the same size as img1
img2 = uint8( img2 );   %% ensure correct data type of unsigned 8-bit integer

figure(4),subplot(1, 2, 1), imshow(img2), title('Zero Defined Image')
img2(:,:,3) = img1_b;
subplot(1, 2, 2), imshow(img2), title('1 component: Blue')
```





You can change to show the results of different orders of adding up the color components. The results of having partial color components mimic what you may see when some R/G/B pin contact of a display connector (such as the VGA traditionally on computers/laptops) becomes loose.

Question 2: Explain what the bright or dark values of each of the three color planes reprensent. In other words, with respect to the original image, what do clusters of bright pixels represent in a color plane? What do clusters of dark pixels represent in a color plane?

II. Lab: Color Representation-2: HSV

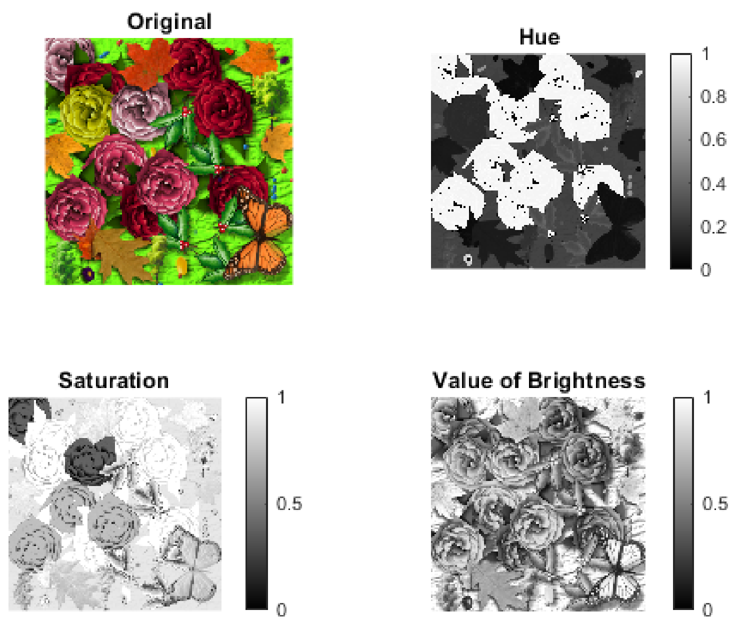
- RGB representation of an image does not always offer complete insights on how a color is seen by our eyes. For example, we may feel two things are of the same color hues but one is lighter and more diluted while the other is more solid.
- HSV is an alternative color representation that offers such perceptual insights by giving numerical descriptions of hue, saturation, and value of brightness.
- HSV is useful in artistic selection of colors, as well as identifying regions of interests in an image (e.g. to locate people in an image by detecting skin tones).
- MATLAB has a built-in function `rgb2hsv` to convert RGB to HSV representation.

```
img_hsv = rgb2hsv(img1);
```

Question 3: What is the size of `img_hsv`?

Like you did with an RGB image, you can extract the three channels of an HSV image.

Question 4: Extract the Hue, Saturation, and Value of `img_hsv` into `img_hue`, `img_sat`, and `img_val`, respectively. Then, make the following 2x2 subplot.



- As **Hue** varies from 0 to 1, the resulting color varies from red through yellow, green, cyan, blue, and magenta, and returns to red.
- When **Saturation** is 0, the colors are unsaturated (i.e., shades of gray); When it is 1, the colors are fully saturated (i.e., they contain no white component and appear to be most solid/pure.
- As **Value** varies from 0 to 1, the brightness increases. Do you see most of look of the image when we display just this component, such as objects in the image, except the color details?

Question 5: Up to this point, how many types of image representations have we encountered, and what are they?

III. Lab: Processing Grayscale Image

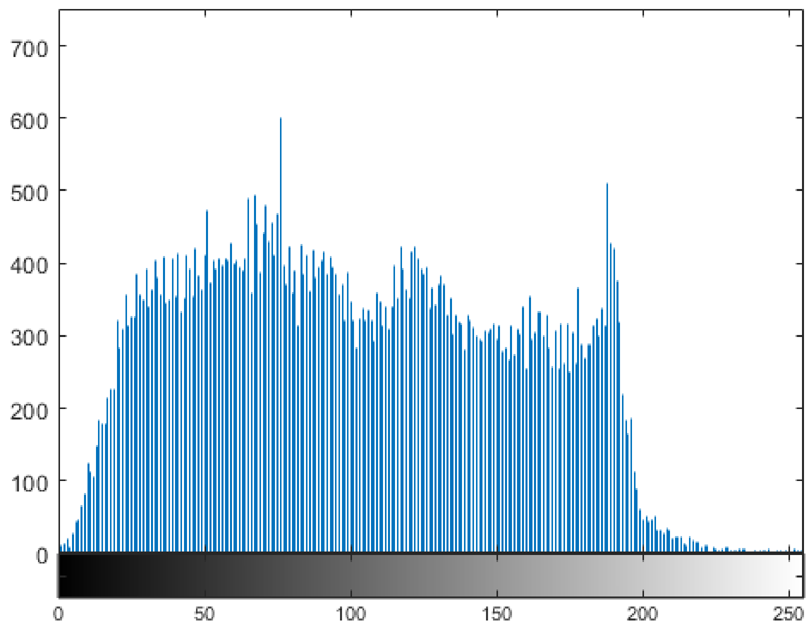
- The Value-of-Brightness component gives a **grayscale** version of the original color image. This value is also known in image processing as the **image intensity**. Some people may call this "black-and-white" pictures, but it actually offers multiple shades of gray instead of just two colors of black and white.
- MATLAB also offers a function `rgb2gray` converting a RGB image (which consists of three arrays) into a grayscale version (which consists of a single array).

```
img3 = rgb2gray(img1);   % convert a color image to grayscalex
```

Question 6: What is the size of `img3`?

We can plot a histogram of `img3` as follows.

```
figure(6), imhist(img3)    % Show *image histogram*
```



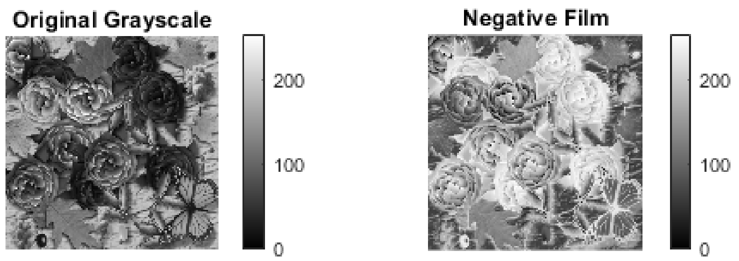
Question 7: What does this histogram show? Label the axes.

Image Processing: we examine some basic processing techniques to grayscale image.

This MATLAB page lists some functions that you may be interested in playing with: http://www.mathworks.com/help/images/contrast-adjustment-.html You can also try out the built-in contrast adjustment tools: http://www.mathworks.com/help/images/adjusting-image-contrast-using-the-adjust-contrast-tool.html

1) Flipping the shades: Change dark to bright, and vice versa; this gives the effect of a negative film

```
img3_1 = 255 - img3;
figure(7), subplot(2, 2, 1), imshow(img3), title('Original Grayscale'), colorbar
subplot(2, 2, 2), imshow(img3_1), title('Negative Film'), colorbar
```



Question 8: In the subplot above, plot each picture's respective histogram below the image.

2) Clipping and contrast stretching: Change all pixels below a low grayshade (`t1`) to black, and above a high shade (`t2`) to white (recall what you did in Module 4), and stretch all shades in between. This technique is especially useful for images with low contrast (where pixels that should be dark is not dark enough and/or those that should be bright is not bright enough).

You can implement this pixel-by-pixel using for-loops and if-then statement as you would do in other languages such as C and Java. Note that in order to better utilize MATLAB's built-in acceleration, a MATLAB-friendly implementation would utilize matrix based operations as much as possible and avoid loops of many iterations, which was especially important historically with older versions of MATLAB.

A partially completed contrast-and-clipping algorithm is provided below. This sample code normalizes the values between t1 and t2.

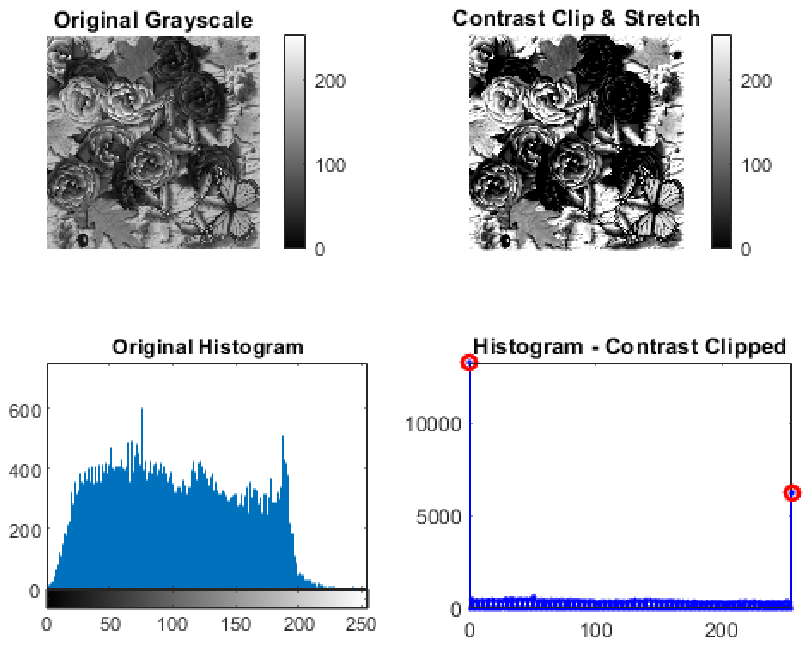
Question 9: Your task is to add two more conditions to change all pixels below a low grayshade (`t1`) to black, and above a high shade (`t2`) to white

```
t1 = 50;  t2 = 180;  % set a low and high threshold for contrast adjustment

img3_2 = double(img3);  % initialize an array with equal size
                        % as img3
for i=1:size(img3, 1)  % loop through all rows
    for j=1:size(img3, 2) % loop through all columns
        if (img3(i, j) >= t1) && (img3(i, j) <= t2) % check to see if current pixel value (i,j) of matrix is between t1 and t2
           img3_2(i, j) = (img3_2(i, j) - t1) * 255 / (t2 - t1); % if so, apply a normalization factor

        end
    end
end
img3_2 = uint8( img3_2 );  % convert the data type to unsigned 8-bit integer
```

The results of clipping and contrast stretching are shown in the figure below.



Question 10: Why are there red circles at the left and right edge of the Contrast Clipped histogram?

Question 11: What if t1 = t2? Would it cause "dividing by zero" by the above implementation? What effect would the processing now become? How would you revise the MATLAB script to handle this special case?

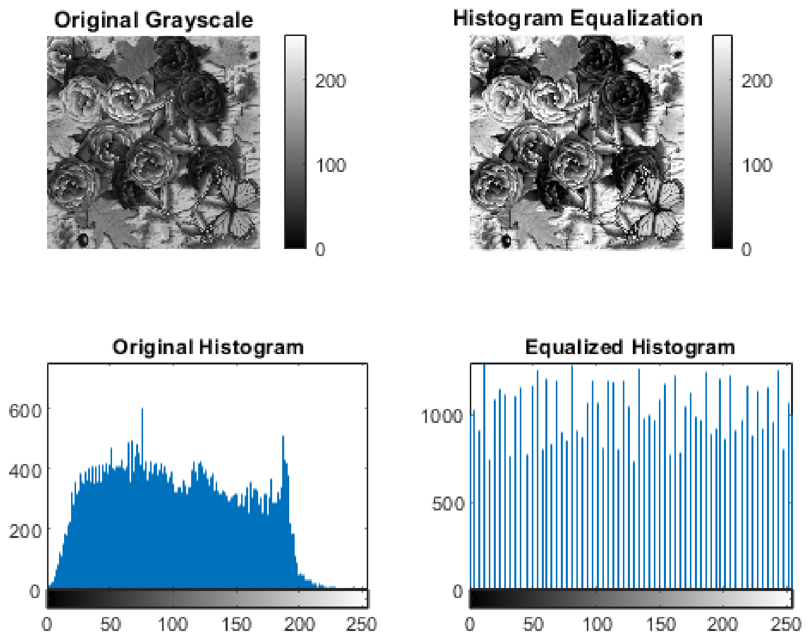
An alternative, MATLAB-friendly implementation: (Don't worry if it doesn't make much sense to you yet; you can become a MATLAB guru in the years to come.)

```
img3_3 = double(img3);   % initialize an array with equal size as img3
img3_3 = double((img3(:,:) >= t1) & (img3(:,:) <= t2)) .* ...
           ((img3_3(:,:) - t1) * 255 / (t2 - t1) ) + ...
         (img3(:,:) > t2) * 255 + (img3(:,:) < t1) * 0;
img3_3 = uint8(img3_3);  % convert data type back to uint8
%subplot(2, 2, 3), imshow(img3_3), title('Contrast Clip & Stretch'), colorbar
```

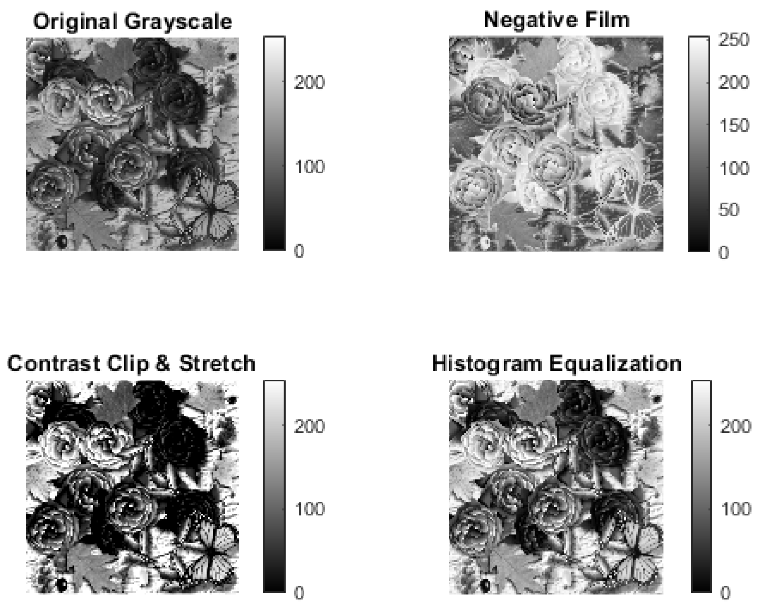
3) Histogram equalization: enhance contrast by adjusting gray shade distribution to nearly uniform

```
img3_4 = histeq(img3);  % apply histogram equalization on img3
```

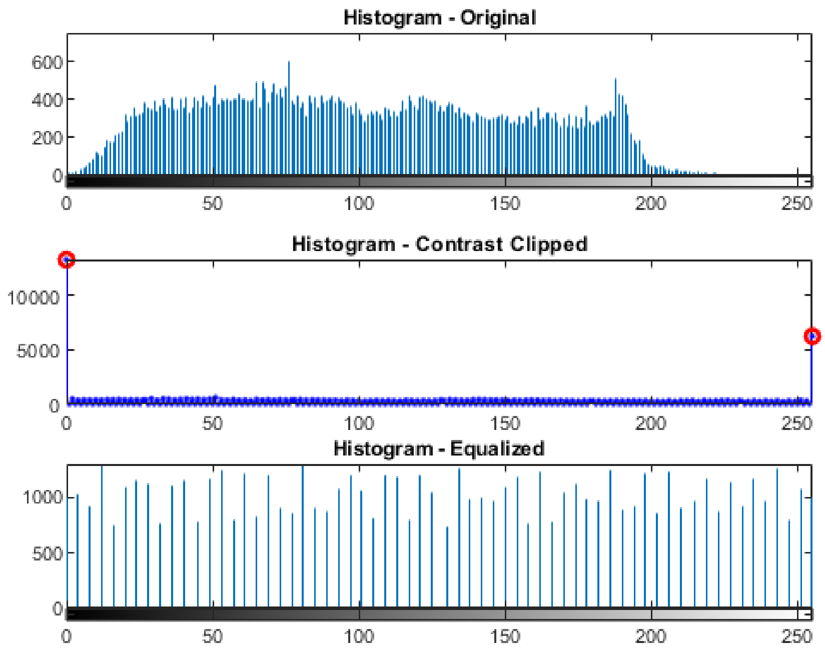
The results of histogram equalization are shown in the figure below.



4a) Display and compare all results:



4b) Display and compare histograms of different processings:



Bonus Question: What if you apply some of the above processing to one or multiple RGB color channels of the color image, and assemble back the color image (which consists of three arrays as shown earlier)? Write MATLAB script to carry out this processing, save and show your final color image, and discuss your findings.

IV. Lab: Green Photography for Fun Background

In a TV weather report, the viewer sees a weatherman standing in front of a map or some other computer-generated image. In reality, the weatherman is standing in front of a green (or blue) curtain. A special camera system will carry out processing and extract the weatherman?s image, and superimpose it upon a computer-generated weather related image.

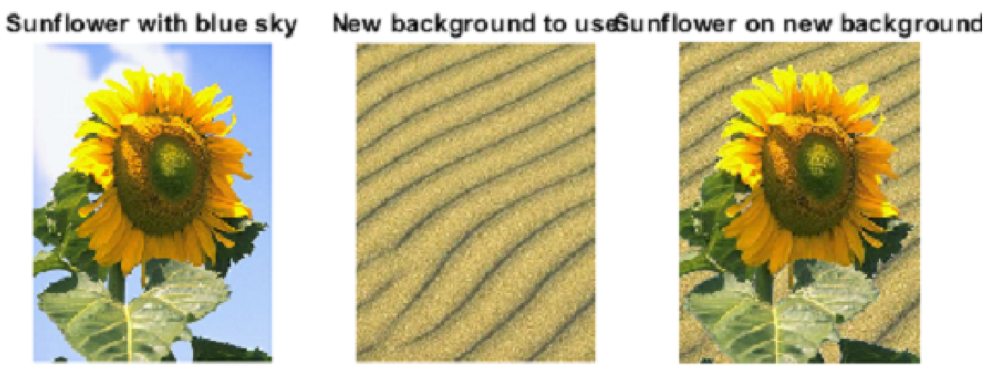
The basic idea of this camera system is to split an image into RGB channels, create a mask based on the information of blue channel, and use this mask to extract the desired image.

Question 12: You have two options -- you are required to do at least one option; bonus points will be awarded to students who do both and documented the results and discussions for both tasks in the report. For each option, in your report, draw a block diagram to explain your scheme, and describe the procedures you followed.

- Option-1: Use Matlab to extract the flower object from `BlueBG.bmp`, and superimpose this flower object upon a background image, `Sand.jpg`. Save the final image of the flower with sand. Below we show you the step-by-step example results. You need to write MATLAB script to generate the third image from the first two.

Hint: To get this mask, you will need to examine each of the three RGB planes pixel by pixel, and extract the flower onto the sand.

```
figure(12),
subplot(1,3,1), imshow( imread('BlueBG.bmp') ), title('Sunflower with blue sky')
subplot(1,3,2), imshow( imread('Sand.jpg') ), title('New background to use')
subplot(1,3,3), imshow( imread('Sand_flower.jpg') ), title('Sunflower on new background')
```

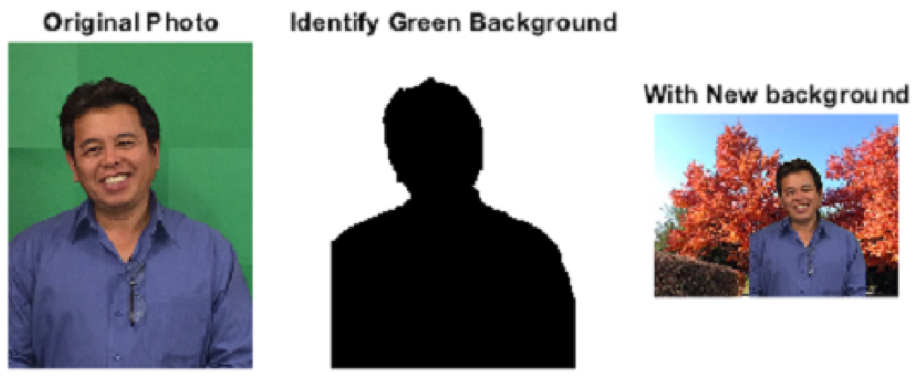


- Option-2: DIY Green Photography -- green construction papers have been pasted on the door of the ENEE101 Lab, with which you can stand in front of it and take a picture of you or your classmate. Transfer this photo into your computer and loaded into MATLAB. Examine this photo and write a MATLAB script to separate the foreground person from the green background. Then put the foreground person/object (without the original green background) onto a different background to create a new fun image; you may adjust the foreground object size if needed using `imresize( )`. To help you get started, we provide you a picture of the fall colors; you may use a different background picture if you wish.

Below we show you the step-by-step example results. You need to write MATLAB script to generate the third image.

Hint: To get this mask, you will need to examine each of the three HSV planes pixel by pixel, and extract the flower onto the sand.

```
figure(13),
subplot(1,3,1), imshow( imresize(imread('Photo_Gomez.jpg'), 0.25) ), title('Original Photo')
subplot(1,3,2), imshow( imread('Photo_Gomez_mask.png'), 'Border', 'tight' ), ...
    title('Identify Green Background')
subplot(1,3,3), imshow( imread('Photo_Gomez_NewBg.jpg') ), title('With New background')
```



Question 13: If the weatherman stands in front of a blue curtain to create the above special background effect, what would happen if the weatherman is wearing a blue jacket of the same color as the curtain? Can the special effect system still work? Why or why not?

References

- Matlab online tutorial: "MATLAB Onramp" https://matlabacademy.mathworks.com/R2015a/
- ENEE408G course material: http://www.ece.umd.edu/class/enee408g.F2006/image/
- Interested students may read more about HSV at this MATLAB documentation page: http://www.mathworks.com/help/images/convert-from-hsv-to-rgb-color-space.html

Published with MATLAB® R2018b