Matlab Exercises: Color Image Processing

(a) Color Images in Matlab

A color image requires three separate items of information for each pixel. A (true) color image of size $m \times n$ is represented in Matlab by an array of size $m \times n \times 3$: a three-dimensional array. We can think of such an array as a single entity consisting of three separate matrices aligned vertically. Let us read an RGB image:

\[
\begin{align*}
&\text{>> } \text{imread('flowers.tif');} \\
&\text{>> } \text{size(x)}
\end{align*}
\]

We can isolate each color component by the colon operator: Refresh: The colon operator indicates taking ALL elements of that dimension. For e.g. $x(:, :, 1)$ means taking ALL rows (1st dimension) and columns (2nd dimension) of “depth” 1 of the 3rd dimension. I use “depth” just to illustrate that there is a third dimension here. Attempting to show what each of the 3 third-dimension look like:

\[
\begin{align*}
&\text{>> } \text{figure;} \\
&\text{>> } \text{subplot(221); imshow(x(:,:,1)); title('R channel');} \\
&\text{>> } \text{subplot(222); imshow(x(:,:,2)); title('G channel');} \\
&\text{>> } \text{subplot(223); imshow(x(:,:,3)); title('B channel');} \\
&\text{>> } \text{subplot(224); imshow(x); title('Color image');}
\end{align*}
\]

What can you observe from the R, G, and B channel images?

(b) Converting between Color Models

There are functions in Matlab to convert from the RGB model to other color models such as HSV, YIQ (Matlab calls it NTSC), YCbCr, and vice versa.

\[
\begin{align*}
\text{rgb2hsv, rgb2ntsc, rgb2ycbcr,} \\
\text{hsv2rgb, ntsc2rgb, ycbcr2rgb}
\end{align*}
\]

Convert the image to these different color models, show each component and observe what they represent. For example, to convert to HSV,

\[
\begin{align*}
&\text{>> } \text{xh = rgb2hsv(x);} \\
&\text{>> } \text{figure, imshow(xh(:,:,1));} \\
&\text{>> } \text{figure, imshow(xh(:,:,2));} \\
&\text{>> } \text{figure, imshow(xh(:,:,3));}
\end{align*}
\]

Also, displaying the image in HSV model,
does not give us much meaning. Why is there not much usefulness in the HSV image?

The same thing applies to the image using YCbCr model. The image in this model is not much use to us. What then is useful?

Good understand of the various components of these color models can help you to identify which component/channel is best suited to perform certain image processing operations.

(c) Pseudocoloring (Gray-to-Color Transformations)

Converting a grayscale image to color image suggests that we are attempting to move from representing an image in 2D to representing an image in 3D. Is that possible? Yes, it is possible, but it involves some artificial means of inputting color information into the image.

Pseudocoloring means assigning colors to a grayscale image in order to make certain aspects of the image more amenable for visual interpretation – for example, for medical images.

In Matlab, a simply way to view an image with added color is to use imshow with an extra colormap parameter. For example, consider the image saturn.tif. We can add a color map with the colormap function. An example of adding a color map to an image is:

```matlab
>> b = imread('saturn.tif');
>> imshow(b,jet);
```

Here, we use the colormap called jet to add color to the grayscale image. There are other color maps, as listed in the help file for graph3d. Explore and try them.

A bad choice of color map can ruin an image. Try using the vga color map to the saturn image. As the vga color map has only 16 colors (or rows), it is not possible to represent 256 different levels in the saturn grayscale image. The resulting color image has only 2 colors!

The trick is to reduce the number of grayscales in the image down to 16 by resampling the image (which, reduces the quality of the image). This is done with the grayslice function:

```matlab
>> b16 = grayslice(b,16);
>> figure, imshow(b16,vga);
```
Combining Colors with Grayscale Images

**Method 1**

It is also possible to combine or “add” colors into grayscale images, but it involves converting first a 2D grayscale image to a 3D grayscale image. How are different shades of gray representing in a color image?

Having same values for all 3 channels (R, G and B) would produce a gray color.

So, using the Saturn image, your first step is to “concatenate” (using the `cat` function) 3 copies of the original grayscale intensity image along the 3rd dimension, one for each of the three channels. Display the image to check if the newly created 3D truecolor image correctly shows the original grayscale image.

```matlab
>> grayimg = imread('saturn.tif');
>> colorimg = cat(3,grayimg,grayimg,grayimg);
>> figure, imshow(colorimg);
```

Now, you can “color” some of the pixels! Let us attempt to mark one of Saturn’s moons with a red plus sign of about 10-pixel size, located at (305,330) row-column position.

```matlab
>> row = 305;
>> col = 330;
>> colorimg(row-5:row+5,col,1) = 256;
>> colorimg(row-5:row+5,col,2) = 0;
>> colorimg(row-5:row+5,col,3) = 0;
>> colorimg(row,col-5:col+5,1) = 256;
>> colorimg(row,col-5:col+5,2) = 0;
>> colorimg(row,col-5:col+5,3) = 0;
>> figure, imshow(colorimg);
```

Notice how the color red is indicated? In RGB model, red is represented by (256,0,0), which is full RED, no BLUE, no GREEN. The rest of the code does the work of creating the plus sign. The code above can be further simplified...

```matlab
>> red = [256 0 0];
>> for dim = 1:3
    colorimg(row-5:row+5,col,dim) = red(dim);
    colorimg(row,col-5:col+5,dim) = red(dim);
end
>> figure; imshow(colorimg);
```

**Method 2**

Another shorter but probably more complex way (in terms of language syntax), is to make use of the Graphics Toolbox in Matlab. This way avoids converting 2D grayscale images to 3D truecolor images, and allows us to put colored markers directly on grayscale images.
>> a = imread('saturn.tif');
>> figure; imshow(a);
>> hold on;
>> plot(330,305,'r+','MarkerSize',10,'Linewidth',1);

Note that the plot function uses a standard graph-plot axis, which means that the row values are \( y \), while the column values are \( x \). The plot function has many parameters which allows plenty of customization. The hold on command is used to hold the current active figure to enable the plotting to be done on top of the figure.

(e) Processing of Color Images

There are 2 common methods of processing color images:
1. We can transform the color space so that the intensity is separated from the color/hue information, and process the intensity component only.
2. We can process each component of the color image separately before combining back.

Many image processing tasks that we have learnt earlier (for processing grayscale images) can be used in the processing of color images, just that it is common to adopt one of the schemes listed above. Let us consider histogram equalization in color images. Start with the flowers image, and convert it from RGB to YIQ

```matlab
>> c = imread('flowers.tif');
>> cn = rgb2ntsc(c);
```

Then, we isolate the intensity component (Y), which is the first value of the 3\textsuperscript{rd} dimension. Apply histogram equalization to this component, and convert back to RGB for display.

```matlab
>> cn(:,:,1) = histeq(cn(:,:,1));
>> c2 = ntsc2rgb(cn);
>> figure, imshow(c2);
```

View the resulting image, and you can see that the contrast of the image has been enhanced, as intended.

However, suppose we try to apply histogram equalization to each of the RGB components (Red, Green and Blue):

```matlab
>> cr = histeq(c(:,:,1));
>> cg = histeq(c(:,:,2));
>> cb = histeq(c(:,:,3));
```

And then, we combine them all back using the \texttt{cat} function to concatenate three 2D matrices into one 3D matrix:

```matlab
>> c3 = cat(3,cr,cg,cb);
>> figure, imshow(c3);
```
What do you observe in the output image?
Did you noticed strange colors (purplish-blueish tint) introduced in some portions of the image?

Some parts of the flower petals and leaves appear washed out.
This is not the intended result of histogram equalization, and it is unacceptable!

Therefore, it is important to know which components of a color image should be processed and which should not be processed.

As an additional exercise, you can try the following image processing operations on color images, and find out how to correctly process the color image with the correct components:
  ● Spatial Filtering – smoothing and sharpening
  ● Noise Reduction – median filtering