

INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 2 - COLOR SPACES

Submission Date: 26.11.18

Introduction

In this assignment we will learn the basics of color spaces and manipulation of image colors. Although MATLAB has many built in functions, our goal is to learn image processing through doing things ourselves. Please avoid using built-in MATLAB functions unless clearly stated otherwise. However, feel free to compare your own functions to the built-in functions.

Before you start, please read the submission instructions at the end of the exercise and follow them during your work. For any question regarding this assignment please refer to the course forum on the moodle web site, for personal questions **only** please email hgazit@post.bgu.ac.il, davidit@post.bgu.ac.il, royshau@post.bgu.ac.il or arbellea@post.bgu.ac.il.

1 RGB and Grayscale

In this exercise we will examine the relation between the grayscale and RGB image.

1. Read a color image in MATLAB using the `'imread()'` function. Convert the image to “double” and normalize to the range $[0,1]$.
2. Display the image using MATLAB's `'imshow()'` function.
3. Extract the separate R, G and B channels from the image and display them.
4. Write your own function that converts an RGB image to grayscale `'dip_rgb2gray()'`.
(a) $\text{Gray} = 0.2989 * R + 0.5870 * G + 0.1140 * B$
5. Compare your results to MATLABs `'rgb2gray()'`

6. Manipulate the values and the order of the channels and display the grayscale image. **Show 3 different manipulation.** Explain the manipulations you chose and their effects in the report. You may choose to explain on only 2 manipulations in your report. For example:

- (a) Add a constant to a single channel.
- (b) Switch between the channels.
- (c) Apply some function on a specific channel. (Make sure that the output of the function stays in the range $[0,1]$)

Note: When repeating this step for the next sections, choose manipulations that suite the color space.

2 Additive vs Subtractive Color space

In this exercise we will examine the difference between additive and subtractive color spaces. We will examine the visual properties of the CYMK colorspace

1. Briefly explain about the CYMK color space.
2. Create the CYMK channels from the RGB channels and display each separately:
 - (a) $\text{Black} = \text{minimum}(1-\text{Red}, 1-\text{Green}, 1-\text{Blue})$
 - (b) $\text{Cyan} = (1-\text{Red}-\text{Black})/(1-\text{Black})$
 - (c) $\text{Magenta} = (1-\text{Green}-\text{Black})/(1-\text{Black})$
 - (d) $\text{Yellow} = (1-\text{Blue}-\text{Black})/(1-\text{Black})$
 - (e) Display each channel separately
3. Display the separate channels using the provided '*displayCYMK()*' function.
4. Repeat step 6 from section 1. Display the results using the provided '*imshowCYMK()*' function.

3 HSV

In this exercise we will examine the difference between the HSV and RGB representations. We will see the effects of changing the values of separate channels.

1. Briefly explain about the HSV color space.
2. Write your own function that converts an RGB image to HSV '*dip_rgb2hsv()*'.
3. Convert the image to HSV and display the separate channels
4. Compare your results to MATLAB's '*rgb2hsv()*' function.

5. Repeat step 6 from section 1. Display the results using the provided `'imshowHSV()'` function.
6. Switch the order of the RGB channels and then convert to HSV using `'dip_rgb2hsv()'`. Display the separate channels. Which channels changed? Why?

Note: When showing the HSV channels try using matlab `'hsv'` colormap if needed

4 L*a*b

In this exercise we will examine the difference between the L*a*b representations. We will see the effects of changing the values of separate channels.

1. Briefly explain about the L*a*b color space.
2. Convert the image to L*a*b using MATLAB's `'rgb2lab()'` function and display the separate channels.
3. Repeat step 6 from section 1. Display the results using the provided `'imshowLab()'` function.
4. Compare the channels to the HSV channels from the section 3.3. Explain the relations and differences in the report.

5 Playing With Colors

In this exercise we will try to solve real world problems with your newly acquired knowledge.

1. Use color spaces and other previously learned subjects to automatically circle the blue cap (of the soda bottle) in the `'cap1/2.png'` images.
 - (a) Read the `'cap1/2.png'` images in MATLAB using the `'imread()'` function and normalize them to $[0 - 1]$.
 - (b) Find the blue cap of the soda bottle in the images and circle it in each image. Explain the algorithm you've used and show the final result images together with the binary masks you've found.
Example the algorithm steps on a chosen image.
You should come up with an algorithm that finds the cap in the images, regardless on what the input image is (of the given images).
 - (c) Test your algorithm on `'cap_test.png'` image. Don't modify the algorithm, it's alright if it fails. Show and analyze the results. See example in Figure 1:

Notes:

- (a) Your algorithm should be robust, automatic and without any prior knowledge about the spatial location of the cap. Meaning that you can't assume the cap is in the bottom right of the image or build a mask of the cap and search for the closest match in the image. Please use the color spaces and other previously learned subjects.
- (b) In this question you may use MATLAB's built in functions to post-process a mask and to draw a circle (e.g. `'medfilt2()'`, `'insertShape()'`).

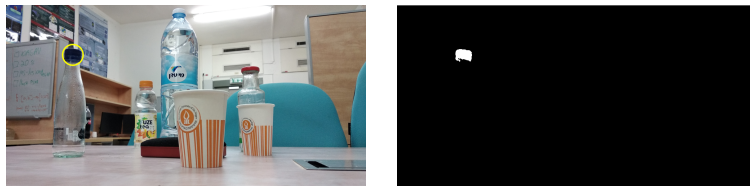


Figure 1: Example of a possible solution

- 2. At night time, when you turn on the light, some colors may look different then they really are (See Figure 2). In this section you will use white balancing to remove yellow light from an image.
 - (a) Take a picture of a white page in your apartment at night time with the lights on. Make sure to turn off the auto white balance in your camera. If you can't find this option ask google or take a picture with a friend's camera.
 - (b) Read about white balancing and design an algorithm for removal of the yellow light from the picture.
 - (c) Explain the algorithm you've used and show the results.



Figure 2: Example of yellow light influence. Photo taken from: <https://www.itsalwaysautumn.com/fix-photo-remove-yellow-color-cast.html>

6 Optical Illusion (Optional)

In this exercise we will trick our mind into seeing colors!
This section is not graded, but we encourage you to do it.

1. Read a color image in MATLAB using the `'imread()'` function and normalize to $[0 - 1]$.
2. Create the negative RGB image: $nRGB = 1 - RGB$
3. Create the grayscale image using your function from the first sections
4. Display the negative image for exactly 10 seconds and then immediately switch to the gray scale.
5. Run the function and stare at the center of the image. What happens when it switches to gray scale?

7 Bonus Question

Choose an image as you wish and manipulate it in the coolest way you can. Use methods from this exercise and optionally from the previous exercise. Display the initial image together with the modified image in the document. **Be creative! One of the images will be chosen by the course staff and it's authors will receive one bonus point to the final grade.** The staff will judge by the visual result, originality and the code.

Submission Instructions

The assignment is to be done in MATLAB and submitted to the course moodle page in the form of a *.zip (**not RAR**) containing *.m files and images along with a report in the form of a PDF file (**not .doc**). The file name should be the initials and ID of both of the team members ex. 'HG-1234567-AA-7654321.pdf'. The use of built-in MATLAB image processing functions is strictly forbidden unless the instructions says differently.

Document Instructions

The following instructions are mandatory and will be checked and graded by the course staff. Failing to follow these instructions **will** reduce points from you grade.

- Each section should have the relevant title as is in this document.
- Every explanation should be accompanied with the relevant image and every image should be explained in the text.

- The displayed images should be large enough for us to see them.
- The document should be organized and readable.

Code Instructions

The following instructions are mandatory and will be checked and graded by the course staff. Failing to follow these instructions **will** reduce points from you grade.

- Use MATLAB version 2014b or later. If you don't have one on your computer, you can work from the computer laboratories in building 33.
- A **main** function should call all the section functions in the correct order and should be named *Ex1.m*.
- The first line of *Ex1.m* should print the full names and IDs of all team members. Use MATLAB's *disp()* function.
- Write modular functions for the subsections and reuse those functions throughout your code whenever possible.
- Every **.m* file should start with a comment containing the full names and IDs of all team members.
- The code should be clearly written with correct indentations (you could use automatic indentation Ctrl+A followed by Ctrl+I).
- Use meaningful names for all functions and variables.
- Try to avoid overriding variables.
- Write comments for every line of code that is not completely self explanatory.
- For every image displayed give a meaningful title using MATLAB's *title()* function.
- Use subplots whenever possible.
- All paths to files should be relative paths. If you are using subfolders use MATLAB's *fullfile()* function to construct the path to the file. Do not hard code '/' or '\' in the paths.
- The code should run completely without errors. A project with errors **will not be checked!**