CSCI435/CSCI935
Computer Vision: Algorithms and Systems

Spring 2024

Assignment One (15%)

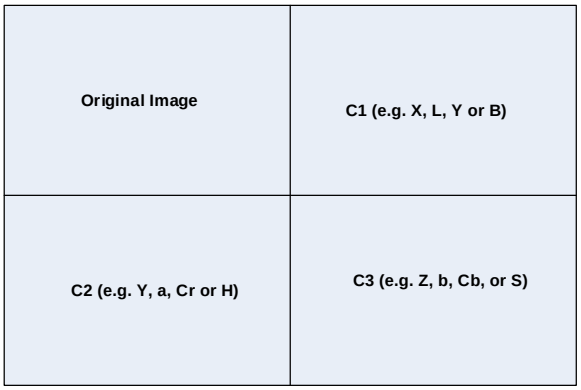
Due Date: 23:30 Tuesday 20 August 2022

Objectives

- Getting familiar with OpenCV 4.10.0
- Reading, processing and displaying images using OpenCV 4.10.0
- Understanding color spaces and transformation

Task One

Many color spaces are available to represent pixel values of a color image. Some are perceptually uniform and others are not. This task is to read an image and display the original color image and its components of a specified color space, such as CIE-XYZ, CIE-Lab, YCrCb and HSB. The color components must be displayed in gray-scale. The original image and its three components must be display in a single viewing window by exactly following the arrangement shown below.



Note that

a) RGB color space is commonly used to represent a digitized images and each color component, also known as a color channel, is digitized into unsigned 8 bits, i.e. the possible values for each component ranges from 0 to 255. (See *Appendix: Digital Representation of Images*)
b) a JPG image file is often decoded into a RGB image, RGB values of each pixel may be packed together with an alpha channel into 32 bits for efficient access by CPU. The value of alpha channel may be ignored or may be used for transparency as needed. When a JPG image is loaded by an OpenCV function, such as `imread(…)`, please check OpenCV online reference to understand how the RGB values of each pixel is packed and stored, and make sure you access these values correctly. (See *Appendix: Digital Representation of Images*)
c) For a color space other than RGB, check the range of its valid values for each component and

you may need to scale the color component values properly so as to display the components in gray-scale images whose pixel values are between 0 to 255, i.e. 8 bits; Note that a gray-scale image means that the RGB values are same for each pixel.

Images are provided to test/debug your code. It is highly recommended that you create testing cases by yourself.

Task Two

A Chroma key is a technique often used in film, television studio and photography to replace a portion of an image with a new image or to place a person, such as a newsreader, on a scenic background. Below shows an example:

(a) A green screen photo with subjects

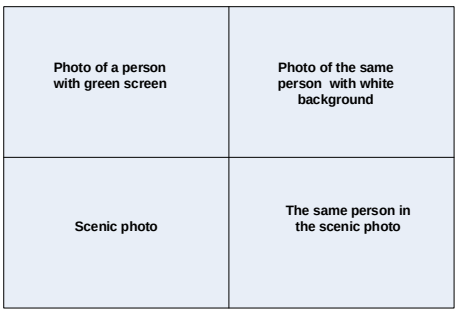
(b) Subjects extracted from the green screen photo are blended with a scenic photo

Source: http://mediacollege.com

You are provided with a few photos of a single person in front of a green screen, referred to as green screen photos, and a few scenic photos. This task is to extract the person in a green screen photo using appropriate Chroma information, e.g. hue and/or other Chroma information, and place the person in a scenic photo according to the following:

a) The combined photo should be of the same size as the scenic photo, and
b) The person should be aligned horizontally to the middle of the scenic photo and aligned vertically to the bottom of the scenic photo.
c) No distortion should be introduced when generating the combined photos.

The program should display in a single viewing window the photo of a person in front of a green screen, person extracted from the green screen photo with white background, scenic photo, photo with the person being in the scenic in a single viewing window as illustrated below.



Requirements on implementation

1. The program should be named as "**ChromaKey**" and shall take

a. one filename and one of the options `-XYZ, -Lab,-YCrCb or -HSB` as the command argument for Task One, e.g.
```
ChromaKey.py -XYZ|-Lab|-YCrCb|-HSB imagefile          # task one
```
or

b. two filename as the command arguments for Task Two, e.g.
```
ChromaKey.py scenicImageFile greenScreenImagefile     # task two
```

*Note command name and options **MUST** be in the exact format as specified above, deviation from the specified format may lead to zero marks.*

2. The size of the viewing window for both tasks should not be over the size of 1280×960 (width x height pixels), and not smaller than 1200×900 in either dimension.
3. There should not be any gap between images and between images and the frames of the viewing window. One way is to combine the four images to be displayed into a single one then display the combined one using `imshow()` in OpenCV
4. Aspect ratios of all images must be kept unchanged while they are being displayed in the viewing window.
5. No other third-party libraries should be used in the program except OpenCV 4.10.0. The code has to be in Python (with OpenCV 4.10.0, assuming `numpy` and `matplotlib` packages exist).
6. The code should be modularized with detail comments AND all source code should be placed in a single file named as "`ChromaKey.py`".
7. Your code will run via a Window PowerShell or Command Terminal. For example,
   The following is to run Task One with option –XYZ and an input image

The following is to run Task Two

Marking Scheme

1. Zero marks may be graded if your code cannot run as specified in the requirements.
2. Program usability (2%)
3. Correct display of the raw image and color components as required (6%)
4. Extraction and display of the person from a green screen photo (4%)
5. Combination of the scenic photo with the extracted person and display of the combined photo (3%)

Submission

Zip the `ChromaKey.py` file to ***your_login_name*.zip**. The zip file must be submitted via Moodle.

IMPORTANT:

a) *The zip file name **MUST be** your UOW login name*

b) Submission through email ***WILL NOT*** be accepted

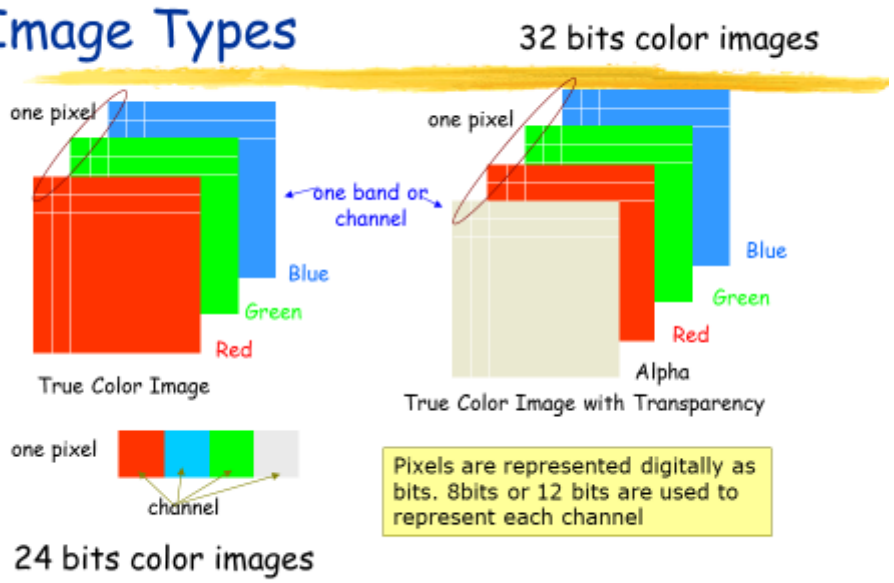
Appendix: Digital Representation of Images

24-bit Color Images

- In a color 24-bit image, each pixel is represented by three bytes, usually representing RGB.
  - This format supports a total of 16,777,216 possible colors.
  - However such flexibility does result in a storage penalty
- **Note**: many 24-bit color images are actually stored as 32-bit images, with the extra byte of data for each pixel used to store an *alpha* value representing special effect information (e.g., transparency).

9/08/2020 51

Image Types



9/08/2020 52

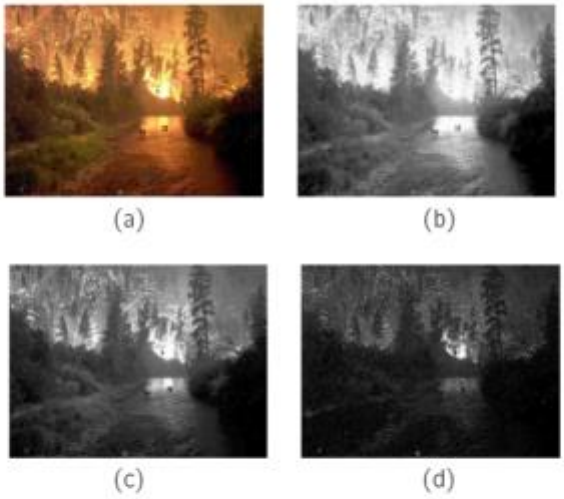


Fig. 3.5 High-resolution color and separate R, G, B color channel images. (a): Example of 24-bit color image "forestfire.bmp". (b, c, d): R, G, and B color channels for this image

9/08/2020 53

8-bit Gray Scale Images

- When a pixel's R, G, B values are equal, it is gray
- Each pixel has a gray-value between 0 and 255. Each pixel is represented by a single byte; e.g., a dark pixel might have a value of 10, and a bright one might be 230.

Gray scale image of Lena

9/08/2020 54