Lab 02: Python Warmup

Due: September 5, 8:30am

Python Basics

- **Indentation matters!**
  - If you're new to Python, pay attention to it
- **Typeless**
  - However, Numpy allows you to strictly control data types in a "C"-like fashion, and declare arrays similar to malloc( )
- **Array indexing starts at 0**
- **Can put ";" at end of lines if you wish, but is not mandatory**
- **if-then-else and for-loop syntax is intuitive (see the "Python Intro" manual)**

Good Code

- **Good code is documented**
- **Good code uses variable names that make sense**
- **Good code is logically organized and easy for someone who *did not write the code* to understand**
- **The quality of your code matters for your grade: it isn't enough to "get the right answer"**
  - Software skills are indispensable for engineers, as is the ability to clearly present results in written form

BMP Images

- **Images comprise 3 color planes: Red, Green, and Blue (RGB). They are stored in raster-scan order**
- **There is a python file, bmp_io_c.py, on Canvas that supports bmp image input and output.**
- **After input, pixels are arranged as a 3-D matrix**
  - Index 1 is the color plane
  - Indices 2 and 3 are the pixel location in rows/columns
    - ✓ Origin is at upper left of image
    - ✓ The values stored are intensities
- **Each pixel is 8 bits (this is a numpy np.uint8 type)**

Assignment

- **Download the file bmp_io_c.py on canvas (look in the ”miscellaneous” module)**
- **Write a Python script named rgb2yuv.py that does the following**
  - Inputs a bmp image as an RGB color image (a 3-D array with three color planes: R, G, and B)
    - ✓ Pass the name of the input file on the command line of your script
    - ✓ See section 7 for an example of how to use bmp_io_c.py to input and output images
  - Continues on next page...

Assignment

- Loops over the image and applies the following matrix transformation on a pixel-by-pixel basis to compute a new "YUV" image (i.e. a new 3-D array) with 3 color planes. The transformation to apply is:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

  - ✓ This calculation must be done using "numpy" as described in the "intro to Python" document (see Section 9.5)

- Prints the maximum and minimum value of each image plane (both the input RGB image and the new YUV image). Six numbers total

- Continues on next page...

Assignment

- Has a command line parameter that determines which color plane to output for viewing: Y, U, or V. The name of the output file to hold the image should also be passed on the command line
  - ✓ Prior to output, the selected color plane --- Y, U, or V --- needs to be copied into the other color plane locations. In other words, each colorplane at every pixel location should have values Y, Y, Y if "Y" was selected on the command line for output
  - ✓ Each color plane should be "clipped" so that values less than 0 are 0 and values greater than 255 are 255 (float to uint8 conversion)
- Run your script on the image "test_image01.bmp" (see Canvas). Output the Y, U, V image planes, view them, and *write a paragraph or two* describing what you see. Can you say anything about U and V? Run your script on "colorbars.bmp" and repeat the above.

Assignment

- **Write a Python script named histo.py that does the following**
  - Inputs a bmp image as an RGB color image (a 3-D array with three color planes: R, G, and B)
    - ✓ Pass the name of the input file on the command line
  - Pass a color plane choice on the command line
    - ✓ R, G, or B
  - Create an array of size 256, and for the selected color plane, count how many pixels of each intensity level occur in that color plane (0 to 255 are the allowed levels). This is a histogram.
  - Normalize the count values so that they add to 1. This creates a pdf. Compute the mean and variance.
  - Generate a graph of the pdf for each color plane of test_image01.bmp. Label the axes and provide a title. See Section 8 of the Python intro for examples

Submission

- **Organize your results and scripts into a single lab report (you will submit one pdf file)**
  - Include your name, the lab number, and the name of the course at the top of your report
  - Include a brief introduction describing what you did in this lab and what is in the subsequent sections of your report.
  - Put your results into clearly name sections. Label each item contained in your report. There should be text descriptions preceding each item in your report (for example, before each graph or included image) explaining what that item is.
  - Include in the report: your observations and answers as requested, your scripts, histogram graphs, and any other items you deem relevant
  - The quality of your report matters for your grade