Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2014

Lab 6 - Image Processing, Continued: Advanced Techniques

Objectives

To develop Python functions that use more advanced techniques to manipulate digital images; e.g., using nested loops to generate pixel coordinates, and changing individual pixel colours based on the colours of neighbouring pixels.

Attendance/Demo

To receive credit for this lab, you must make reasonable progress towards completing the exercises and demonstrate the code you complete. **Also, you must submit your lab work to cuLearn by the end of the lab period**. (Instructions are provided in the *Wrap Up* section at the end of this handout.)

When you have finished all the exercises, call a TA, who will review the code you wrote. For those who don't finish early, a TA will ask you to demonstrate whatever code you've completed, starting about 30 minutes before the end of the lab period. **Any unfinished exercises should be treated as "homework"; complete these on your own time, before your next lab.**

References

Course materials on image processing are posted on cuLearn.

Getting Started

Step 1: You can use your Lab 5 folder from last week's lab. This folder should contain Cimpl.py, the three image (JPEG) files, and the filters.py module you've been developing over the past couple of weeks.

Step 2: Launch Wing IDE 101. Check the message Python displays in the shell window and verify that Wing is running Python version 3.4. If another version is running, ask your instructor or a TA for help to reconfigure Wing to run Python 3.4.

Step 3: Open filters.py in a Wing IDE editor window. Don't delete the filter functions you developed during Labs 4 and 5!

Exercise 1 - Edge Detection

Edge detection is a technique that results in an image that looks like a pencil sketch, by changing the pixels' colours to black or white.

In filters.py, define a function that is passed an image and transforms it using edge detection. The function header is:

```
def detect_edges(img, threshold):
```

Parameter `threshold` is a positive real number.

A simple algorithm for performing edge detection is: for every pixel that has a pixel below it, check the *contrast* between the two pixels. If the contrast is high, change the top pixel's colour to black, and if the contrast is low, change the top pixel's colour to white.

One way to calculate the contrast between two colours is to calculate the average of the red, green and blue components of the top pixel (with all three components weighted equally), and subtract the average of the red, green and blue components of the pixel below it. We then calculate the absolute value of this difference. If this absolute value is greater than the filter's threshold attribute, the contrast between the two pixels is high, so we change the top pixel's colour to black; otherwise, the contrast between the two pixels is low, so we change the top pixel's colour to white.

Hint: your function needs to process one row of pixels at a time, and for a given pixel, it will need to access the pixel below it. Don't use the

```
for x, y, col in img:
```

pattern that you used in your other filters. Instead, use nested `for` loops to generate the (x, y) coordinates of pixels, and call `get_color` to get the `Color` object for each pixel. Several functions and Online Python Tutor examples that use nested `for` loops are posted on cuLearn.

Interactively test your edge detection filter. When calling the function, use 10.0 as the threshold. Repeat your tests with different thresholds. What range of threshold values result in images that look good to your eyes?

Exercise 2 - Improved Edge Detection

Make a copy of your `detect_edges` function, and rename this copy `detect_edges_better`. Don't modify your original edge detection filter.

Edit `detect_edges_better` so that it checks the contrast between each pixel and the pixel below it *as well as the pixel to the right of it*. As before, we calculate the contrast of two pixels by averaging the red, green and blue components of each pixel, subtracting the averages, and calculating the absolute value of the difference. We change a pixel's colour to black only if the contrast between the pixel and

the one below it is high (i.e., the absolute value of the difference exceeds the filter's threshold attribute) *or* the contrast between the pixel and the one to the right of it is high. Otherwise, we change the pixel's colour to white.

Interactively test your function, using 10.0 as the threshold. In your opinion, does this filter do a better job of edge detection than the one you developed for Exercise 1?

Exercise 3 - Changing the Blurring Filter

From cuLearn, download blur_filter.py, which contains a simple blurring filter. (See the Week 6 lecture materials for the link to this file.) Open the file in Wing IDE 101, and copy/paste the `blur` function into filters.py.

Currently, the filter calculates the component values for the new colour of each blurred pixel by averaging the components of the pixel, the pixel above it, the pixel below it, the pixel to the left of it, and the pixel to the right of it.

Another way to create the new colour for each blurred pixel is to average the pixel's red, green and blue components with the corresponding components of the 8 pixels that surround it; i.e., the new colour is based on the colours of 9 pixels.

Modify `blur` to do this. **Do not call `get_color` 9 times**. Instead, replace these statements:

```
top_red, top_green, top_blue = get_color(source, x, y - 1)
left_red, left_green, left_blue = get_color(source, x - 1, y)
bottom_red, bottom_green, bottom_blue = get_color(source, x, y + 1)
right_red, right_green, right_blue = get_color(source, x + 1, y)
center_red, center_green, center_blue = get_color(source, x, y)
```

with a pair of nested `for` loops. These loops will calculate running sums of the red, green and blue components of the 9 pixels centered at location (`x`, `y`) (a total of three sums). In the body of the inner `for` loop, there will be exactly one call to `get_color`. Use these sums to calculate the average red, green and blue component values.

Interactively test your blurring filter. In your opinion, does this filter do a better job of blurring than the one developed in class?

Exercise 4 (Challenge) - Flipping an Image

In filters.py, define a function that is passed an image and transforms it by flipping it through an imaginary vertical line drawn through the center of the image. The function header is:

```
def flip(img):
```

For example, if you had an image of a person looking to the right, the flipped image would show the

person looking to the left.

You'll need to use nested loops.Within each row, the colour of a pixel some distance from the left side of the image must be swapped with the colour of the pixel the same distance from the right side of the image. This exercise is presented as a challenge, so I'm not going to provide the formulas that determine the (x, y) coordinates of the pairs of pixels that will have their colours swapped. That's something you have to figure out. Before writing any code, spend some time sketching diagrams and deriving the formulas. Hint: the body the inner loop is short (less than 10 lines of code). I'll present my solution after students all the lab sections have had time to work on this exercise.

Wrap-up

1. Remember to have a TA review your solutions to the exercises, assign a grade (Satisfactory, Marginal or Unsatisfactory) and have you initial the demo/sign-out sheet.

2. If you've finished all the exercises, your `filters` module should contain 14 filters: `negative`, `grayscale` and `weighted_grayscale` (from Lab 4) and `solarize`, `black_and_white`, `black_and_white_and_gray`, `extreme_contrast`, `sepia_tint`, `posterize` and `simplify` (from Lab 5), and `detect_edges`, `detect_edges_better`, the revised `blur` function and `flip`. (Any unfinished exercises should be treated as "homework"; complete these on your own time, before your next lab.)

   You'll need your `filters` module during Lab 7, so remember to save a copy of your filters.py file (copy it to a flash drive, or email a copy to yourself, or store it on your M: drive - remember, files left on your desktop or in your Documents folder are not guaranteed to be there the next time you log in).

3. Before you leave the lab, log in to cuLearn and submit filters.py. To do this:

   3.1. Click the Submit Lab 6 link. A page containing instructions and your submission status will be displayed. After you've read the instructions, click the Add submission button. A page containing a File submissions box will appear. Drag filters.py to the File submissions box. Do not submit another type of file (e.g., a zip file, a RAR file, a .txt file, etc.)

   3.2. After the icon for filters.py files appears in the box, click the Save changes button. At this point, the submission status of your file is "Draft (not submitted)". If you're ready to finish submitting the file, jump to Step 3.4. If you instead want to replace or delete your "draft" file submission, follow the instructions in Step 3.3.

   3.3. You can replace or delete the file by clicking the Edit my submission button. The page containing the File submissions box will appear.

      3.3.1. To overwrite a file you previously submitted with a file having the same name, drag another copy of the file to the File submissions box, then click the Overwrite button when you are told the file exists ("There is already a file called..."). After the icon for the file reappears in the box, click the Save

changes button.

3.3.2. To delete a file you previously submitted, click its icon. A dialogue box will appear. Click the Delete button., then click the OK button when you are asked, "Are you sure you want to delete this file?" After the icon for the file disappears, click the Save changes button.

3.4. Once you're sure that you don't want to make any changes, click the Submit assignment button. A Submit assignment page will be displayed containing the message, "Are you sure you want to submit your work for grading? You will not be able to make any more changes." Click the Continue button to confirm that you are ready to submit your lab work. This will change the submission status to "Submitted for grading". **Make sure you do this before you leave the lab.**