Chris Piech and Mehran Sahami
CS 106A

Assignment #3
May 4, 2020

Assignment #3: Images

Goal Date: Wednesday, May 13 at 11:59pm Anywhere on Earth.

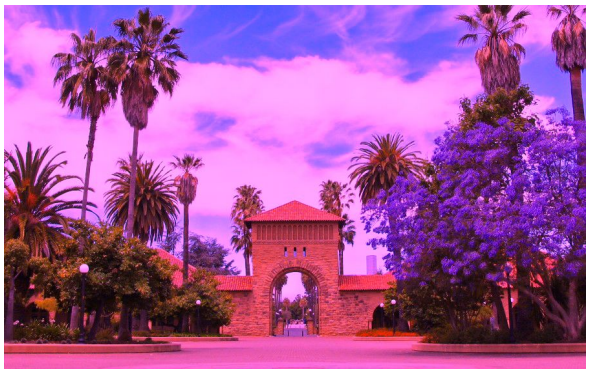
Based on an Image library by Nick Parlante. Problems written by Chris Piech and Sonja Johnson-Yu. Inspired by Julie Zelenski

The handout for this assignment is on the longer side, but don't be intimidated by its length! It is long because it's designed to give you step-by-step instructions for implementing several cool algorithms. We would like everyone to implement the first two tasks. The rest are optional challenges.

Setup: Installing Pillow

Before you get started on the assignment, make sure that you have run through the Pillow installation instructions, which can be found in the "Image Reference" handout on the Code in Place website. If you cannot get Pillow installed please post on ed.

1. Code in Place Filter



Write a program that asks the user to enter an image file name. It then loads that file and applies the "Code in Place" filter. To apply the Code in Place filter, you are going to change every **pixel** to have the following new red green and blue values:

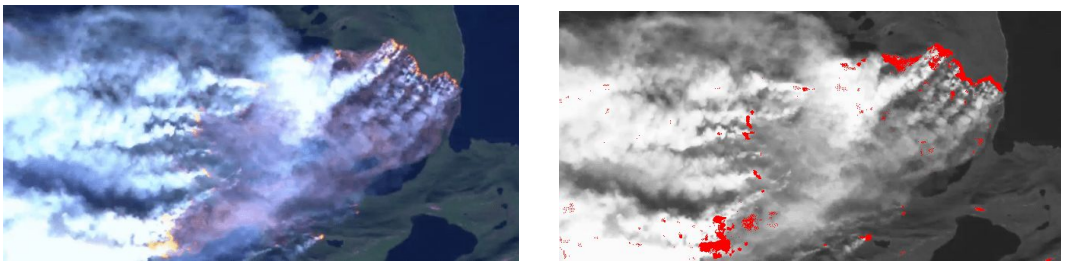
```
new red value = old red value * 1.5
new green value = old green value * 0.7
new blue value = old blue value * 1.5
```

2. Finding forest flames.

We're going to start by writing a function called **find_flames** (in the file **forestfire.py**) that highlights the areas where a forest fire is active. You're given a satellite image of Greenland's 2017 fires (photo credit: Stef Lhermitte, Delft University of Technology). Your job is to detect all of the "sufficiently red" pixels in the image, which are indicative of where fires are burning in the image. As we did in class with the "redscreening" example, we consider a pixel "sufficiently red" if its red value is greater than or equal to the average of the pixel's three RGB values.

When you detect a "sufficiently red" pixel in the original image, you set its red value to 255 and its green and blue values to 0. This will highlight the pixel by making it entirely red. For all other pixels (i.e., those that are not "sufficiently red"), you should convert them to their grayscale equivalent, so that we can more easily see where the fire is originating from. You can grayscale a pixel by summing together its red, green, and blue values and dividing by three (finding the average), and then setting the pixel's red, green, and blue values to all have this same "average" value.

Once you highlight the areas that are on fire in the image (and greyscale all the remaining pixels), you should see an image like that shown on the right in the image bellow. On the left side of Figure 1, we should the original image for comparison.



Original forest fire image on left, and highlighted version of image on right.

Congratulations, you have finished Assignment 3. As always we encourage you to go above and beyond. Make any cool effect you like. Here are a few ideas!

Extensions

3. [Optional, Medium Challenge]: Reflection

Write a function that returns an output image that is twice the height of the original. The top half of the output image should be identical to the original image. The bottom half, however, should look like a reflection of the top half. The highest row in the top half should be "reflected" to be the lowest row in the bottom half. This results in a cool effect.

4. [Optional, Hard Challenge] Warhol Effect

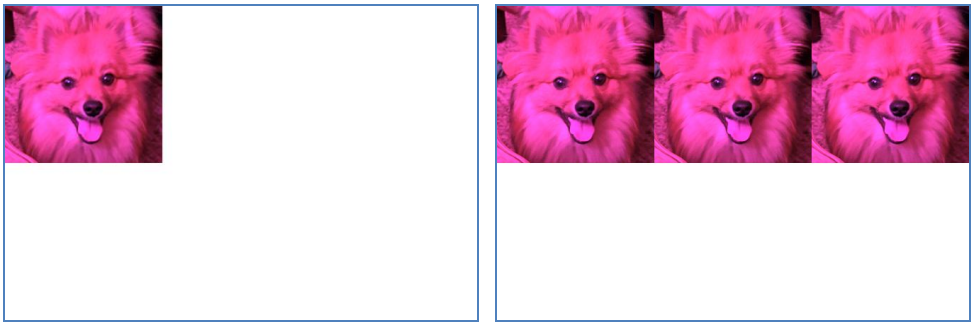
Write an algorithm that takes in a square patch like this photo of Simba the Dog:

And creates an image which has the patch copied 6 times (in 2 rows and 3 columns) where each patch gets recolored. This effect is inspired by some of Andy Warhol's paintings.

We strongly recommend implementing a function:

```
def make_recolored_patch(red_scale, green_scale, blue_scale):
```

Which returns a new colored patch. See the starter code for details. A few milestones:



Don't try to match the colors in this example exactly. Experiment with different combinations of red_amount, green_amount and blue_amount. The pink Simba was generated by

```
make_recolored_patch(1.5, 0, 1.5)
```

Define other functions too! How about a function which adds a colored patch to the final_image at a given row, column?