Image Transformations

CST 205

Review: Grayscale

```
def grayscale(p):
    new_red = int(p[0] * 0.299)
    new_green = int(p[1] * 0.587)
    new_blue = int(p[2] * 0.114)
    lumi = new_red + new_green + new_blue
    return (lumi,) * 3

def gray_list(pic):
    gray_list = [grayscale(p) for p in pic.getdata()]
    return gray_list
```

One more filter

- **Sepia:** a brown pigment obtained from the inklike secretion of various cuttlefish and used with brush or pen in drawing.

Sepia

```
def sepia(p):
    # tint shadows
    if p[0] < 63:
        r,g,b = int(p[0] * 1.1), p[1], int(p[2] * 0.9)

    # tint midtones
    elif p[0] > 62 and p[0] < 192:
        r,g,b = int(p[0] * 1.15), p[1], int(p[2] * 0.85)

    # tint highlights
    else:
        r = int(p[0] * 1.08)
        if r > 255:
            r = 255
        g,b = p[1], int(p[2] * 0.5)

    return r, g, b
```

Moving pixels

- We need to keep track of where we're getting pixels from and where we're putting the pixels
    - *Source* index variables: where we're getting the pixels from
    - *Target* index variables: where we're putting the pixels
- (We are not actually copying pixels, but rather just replicating the colors.)

Uses

- Collages
- Cropping
- Scaling

Review: Empty images in Pillow

```
from PIL import Image

im = Image.new("RGB", (640, 480), "salmon")

im.save("images/empty.png")
```

How to copy an image to a blank canvas

Say hi to Jeanne!

Copy image code

```
def copy_jeanne():

    # source image
    jeanne = Image.open("images/modigliani.png")

    # destination image
    canvas = Image.new("RGB", (640,480), "white")

    target_x = 0
    for source_x in range(jeanne.width):
        target_y = 0

        for source_y in range(jeanne.height):
            color = jeanne.getpixel((source_x, source_y)) # get pixels from the source
            canvas.putpixel((target_x, target_y), color) # put pixels onto target
            target_y += 1
        target_x +=1

    canvas.save("images/copy_jeanne.png")


copy_jeanne()
```

Transformations

- Making small changes to this basic copying program can make a variety of transformations
    - Change the `target_x` and `target_y` to copy wherever you want
    - **Cropping**: Change the `source_x` and `source_y` range and you can copy only part of the image
    - **Rotating**: Swap `target_x` and `target_y` and you copy sideways
    - **Scaling**: Change the increment on `source_x` and `source_y` and you either grow or shrink the image

Shift and copy

```
source = Image.open(pic)
target = Image.new('RGB', (640, 480), 'salmon')

target_x = 100
for source_x in range(source.width):
    target_y = 100
    for source_y in range(source.height):
        color = source.getpixel((source_x, source_y))
        target.putpixel((target_x, target_y), color)
        target_y += 1
    target_x += 1

target.show()
```

Scaling

- When we just copy, we sample every pixel
- If we want a smaller copy, we skip some pixels
  - We sample fewer pixels
- If we want a larger copy, we duplicate some pixels
  - We *oversample* some pixels

Sample fewer pixels

```
source = Image.open(pic)
target = Image.new('RGB', (640, 480), 'salmon')

target_x = 0
for source_x in range(0, source.width, 2):
    target_y = 0
    for source_y in range(0, source.height, 2):
        color = source.getpixel((source_x, source_y))
        target.putpixel((target_x, target_y), color)
        target_y += 1
    target_x += 1
target.show()
```

Quick **numpy** aside

```
import numpy as np

my_list = range(1,6)

dupe_list = np.repeat(my_list,2)

print(dupe_list)
print(repr(dupe_list))
print(dupe_list[4])
print(type(dupe_list[4]))
```

Code for scaling-up

```
source = Image.open(pic)
target = Image.new('RGB', (source.width*mf, source.height*mf))

target_x = 0
for source_x in np.repeat(range(source.width), mf):
    target_y = 0
    for source_y in np.repeat(range(source.height), mf):
        color = source.getpixel((int(source_x), int(source_y)))
        target.putpixel((target_x, target_y), color)
        target_y += 1
    target_x += 1
target.show()
```

Blending Pictures

- Instead of copying from source to target, we can combine the source and target to create a new image

- Simple technique
  - Average the red, green, and blue from the source and target

- Let's put Jeanne on the beach!

Blending code

```
source = Image.open(pic_1)
target = Image.open(pic_2)

source_x = 0
for x in range(240, 240 + source.width):
    source_y = 0
    for y in range(80, 80 + source.height):

        r_s,g_s,b_s = source.getpixel((source_x,source_y))
        r_t,g_t,b_t = target.getpixel((x,y))

        color = (
                    int( (r_s + r_t)/2 ),
                    int( (g_s + g_t)/2 ),
                    int( (b_s + b_t)/2 )
                )

        target.putpixel((x,y), color)

        source_y += 1
    source_x += 1
target.show()
```

Chroma key compositing

- Think of the weather person on the news
- Pose in front of a blue or green screen
- Swap all blue or green for the background

Chroma key code

```
import math
from PIL import Image


def distance(color_1, color_2):
    red_diff = math.pow((color_1[0] - color_2[0]), 2)
    green_diff = math.pow((color_1[1] - color_2[1]), 2)
    blue_diff = math.pow((color_1[2] - color_2[2]), 2)
    return math.sqrt(red_diff + green_diff + blue_diff)


def chromakey(source, bg):
    for x in range(source.width):
        for y in range(source.height):

            cur_pixel = source.getpixel((x,y))
            green = (0, 190, 60)

            if distance(cur_pixel, green) < 250:
               # grab the color at the same spot from the new background
                source.putpixel((x,y), bg.getpixel((x,y)))

    source.save("images/chromakeyed.png")


weather = Image.open("images/story_pic.png")
fruit = Image.open("images/fruit.png")

chromakey(weather, fruit)
```