Comp Photography
Spring 2016
Final Project

Brian Wagner
bpwagner@gatech.edu

https://cs6475.wordpress.com/2016-spring/

Tilt-Shifter

Take any photo and create a simulated tilt-shift photo.

Tilt-shift photos make your photos look like they are pictures of miniature toys!

The Goal of Your Project

The goal of my project is to simulate, using computational photography techniques, a tilt-shifted photo. A traditional tilt-shifted photo uses a special tilt-shift lens that can be physically moved to adjust the plane of focus. This gives a very unique, toy like quality to the photos. I have always wanted a tilt-shift lens for my camera but they are expensive. This gives me the opportunity to simulate the experience using software.

Tilt-shift Lens

Below is a sample tilt-shift photo created my by software.

Input

Output

Notice the toy like quality of the barge as well as the blurred focus of the top and bottom of the photo. Also the colors on the final photo are saturated.

Below is a sample tilt-shift photo created my by software.

Input

Output

Creek Street in Ketchikan Alaska

Below is a sample tilt-shift photo created my by software.

Input

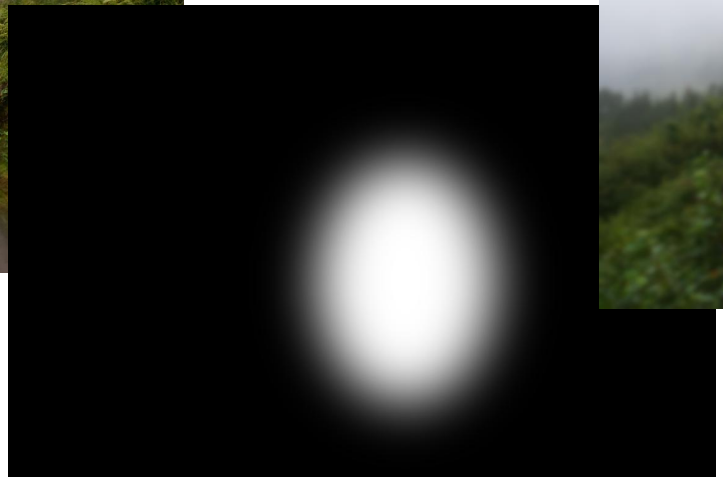
Output

Ketchikan Alaska Harbor. No extra saturation.

Below is a sample tilt-shift photo created my by software.

Input

Custom mask



Output

This image used a custom mask to create the output. The mask was created in Photoshop.

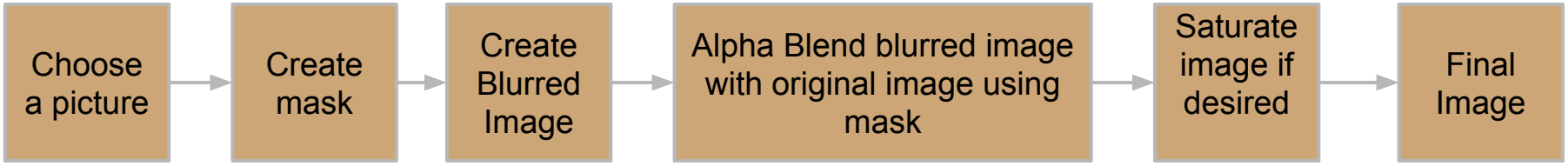
Below is a sample tilt-shift photo created my by software.

Input

Output

This photo was taken at Roy C. Manchester Boy Scout Camp in Kentucky.

Image Pipeline



1: Choose your image. Landscapes work well.

2: Tilt-shifter will generate a mask for you or you can make one yourself and use that.

3: Tilt-shifter will take your original image and make a gaussian blurred copy to use.

4: Tilt-shifter will take the original image and the blurred image from step 3 and alpha blend the two images using the mask that was created in step 2.

Basically, the original (in focus) image will show when the white area of the mask corresponds to that area. The black areas of the mask are where the blurred image appears.

5: Tilt-shift image often look more interesting if the colors of the image are more saturated. Tilt-shifter will increase the saturation of the image if desired.

6: The final image can be displayed and/or the final image is saved.

What is the best way to see your project?

The code for this project can be found at:

https://github.com/bpwagner/tilt-shifter

The main program is tiltshifter.py and it only requires numpy and OpenCV (cv2).

To run it try:

Python tiltshifter.py -i pic.jpg -o out.jpg -d -s

What worked

The project worked and solution was more obvious than I originally thought. Combining a blurred image with the original image using an alpha blend mask worked well. Also, creating the simple horizontal mask also worked well.

Making a function that would saturate an image was easier than expected, once I figured out how to do it.

What did not work? Why?

Creating a project like this is full of trial and error. When I started this project, I thought I was going to have to gradually apply blurring techniques, ramping up and down, as the mask was being produced. I spend a great deal of time experimenting with this and discovered it really did not ‘look’ any better than a simple blur on the entire image.

When I was trying to saturate the image I tried to adjust the RGB values to no success. I had to do quite a bit of research to discover the HSV color model and how that works. Once I got past that hurdle, saturating the image was simple.

More Details…

Tilt-shifter is a command-line python program. It has several parameters that control its execution. Here is tilt-shifter help:

```
tiltshifter.py -i <inputfile> [-o <outputfile>] [-m <mask file>]
   other options
        -h help
        -p percent (10 is default)
        -y (vert center of image is default)
        -d display images
        -s saturate image
```

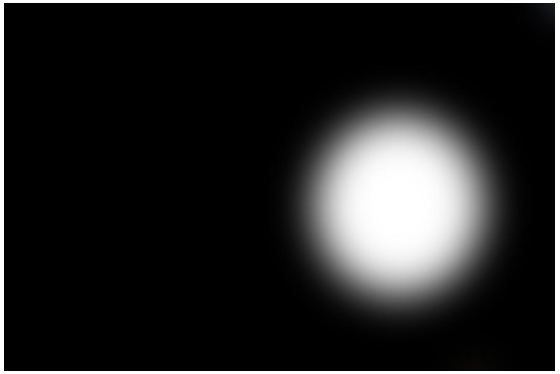
Tilt-shifter requires an input file, but the mask file and output file are optional. If the -m option is used, the program will use a user created mask file. If the -o option is used, the program will save the output to a file. The -p option sets how much (%) of a horizontal mask bar the program will create. The -y option is how far down (in pixels) the horizontal mask bar will be created. The -d option will pop up the input image, mask image and output image to view. The -s option will saturate the final image.

The mask...

Tilt-shifter creates a mask that is black with a grey and white horizontal bar running through it. This bar is 10% (configurable with the -p option) of the size of the image. The position of the bar is in the center of the image unless the -y option is used. The horizontal bar mask works well with landscape images.

A user supplied mask can also be used. The mask has to be the exact size of the original image. It is chosen using the -m option.

Example generated mask



Example user created mask

Blurring

Blurring of the original image is done using the Gaussain Blur function of OpenCV. I experimented with the sigma parameter to adjust the blurriness of the image. 4 was found to look good. The actual command is:

```
cv2.GaussianBlur(image,(25,25),4)
```

Alpha blending

Alpha blending was done to blend the original image with the blurred image using the mask. Each pixel in the final image is composed of a combination of the two images weighted by the values of the corresponding pixel in the mask. Here is the relevant code:

```
for row in range(image.shape[0]):
    for col in range(image.shape[1]):
        alpha = mask[row,col] / 255.0
        pixel = image[row,col] * alpha + blurred_image[row,col] * (1-alpha)
        image_out[row,col] = pixel
```

Saturating

Saturating the image was done by changing the color space of the image to HSV (hue, saturation, value), adjusting the saturation and then changing the color space back to BGR (blue, green, red). In HSV mode, a value of 20 was added to each saturation value and a value of 10 was added to each value, for each pixel. I experimented quite a bit to get the saturation to look good for these photos. Here is the code that performed the saturating:

```
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
for row in range(hsv_image.shape[0]):
    for col in range(hsv_image.shape[1]):
        hsv_image[row,col][1] = hsv_image[row,col][1] + 20
        if hsv_image[row,col][1] > 255:
            hsv_image[row,col][1] = 255
        hsv_image[row,col][2] = hsv_image[row,col][2] + 10
        if hsv_image[row,col][2] > 255:
            hsv_image[row,col][2] = 255
image = cv2.cvtColor(hsv_image, cv2.COLOR_HSV2BGR)
```

Final thoughts...

Overall, I happy with the results of this project. I did spend quite a bit of time trying to make this application GUI based using QT, and I gave up. That will have to be another project as it became very frustrating. Eventually I settled on making the app be command-line based, and that worked well for me.

Future improvements for this app would be to have more command line arguments to control the level of Gaussian blur and saturation.

Another improvement would be the ability to take a video, break down the frames, apply the tilt-shifter and put the video back together. That would be fun!

References / Pointers

- https://en.wikipedia.org/wiki/Tilt%E2%80%93shift_photography
- https://en.wikipedia.org/wiki/Scheimpflug_principle
- https://en.wikipedia.org/wiki/HSL_and_HSV
- http://mashable.com/2013/09/24/photoshop-tilt-shift/#8_WXfVVVu8qH
- https://inst.eecs.berkeley.edu/~cs194-26/fa14/upload/files/projFinalUndergrad/cs194-ko/part1/
- http://www.tiltshiftphotography.net/
- http://lensbaby.com/videos-instructional/edge80-tilt-photo-tips
- http://tiltshiftmaker.com/

Credits/Thanks

- I would like to thank my daughter, Sarah, who has an actual lens-baby lens and put me on to tilt-shift photography. Enjoy some of her wonderful photos… http://swagner.smugmug.com/
- I would like to thank Dr. Irfan Essa and the wonderful TA’s for OMSCS CS 6475 - Spring 2016