Lab 5

John Munyi

26/09/2018

Table of Contents

Table of Contents **2**

Introduction **3**

1. Stereo-pairs **3**

1.1 Results 3

1.2 Discussion 6

2. Conclusion **6**

3. Appendix **7**

Introduction

This assignment covers the concept of stereo pairs images, this is where you have 2 images each representing the left and the right eye respectively, when the 2 images are processed and viewed on the Oculus Go in the 360 degrees / 3D mode the user only see it as a single image.

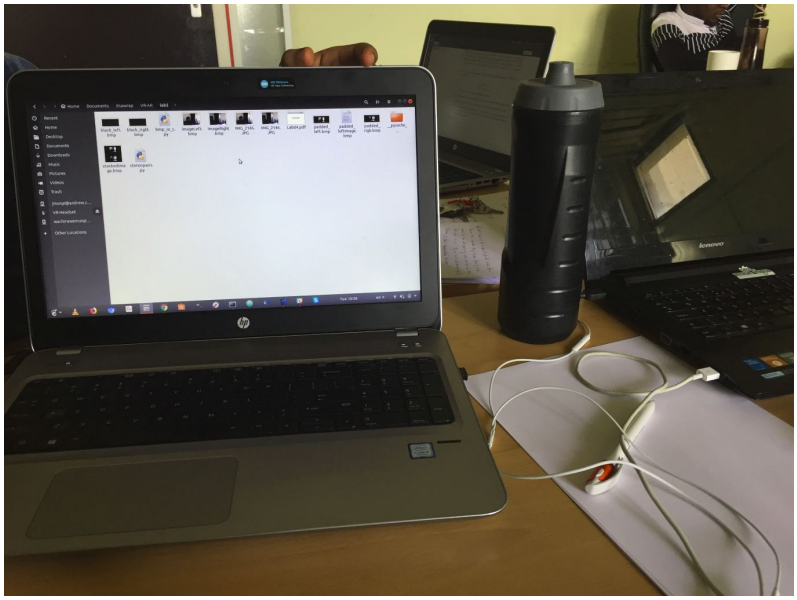
1. Stereo-pairs

The aim of this lab is to eventually implement stacking of **stereoscopic** image, such that when the images are viewed with naked eye or in 2D they look like 2 stacked images, however when viewed in the Oculus Go in 3D/360 mode, the image is viewed as a single 3D image.

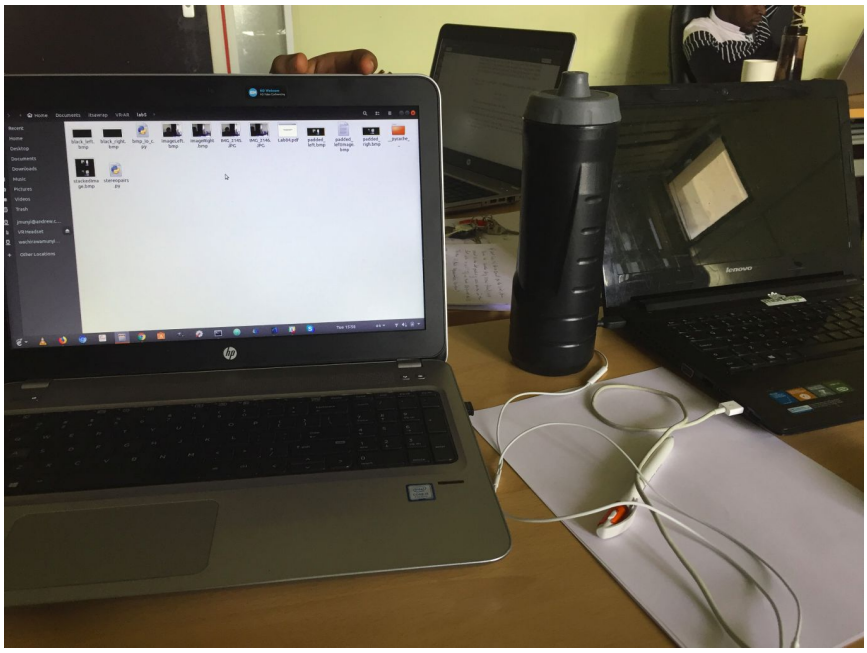
1.1 Results

Original images

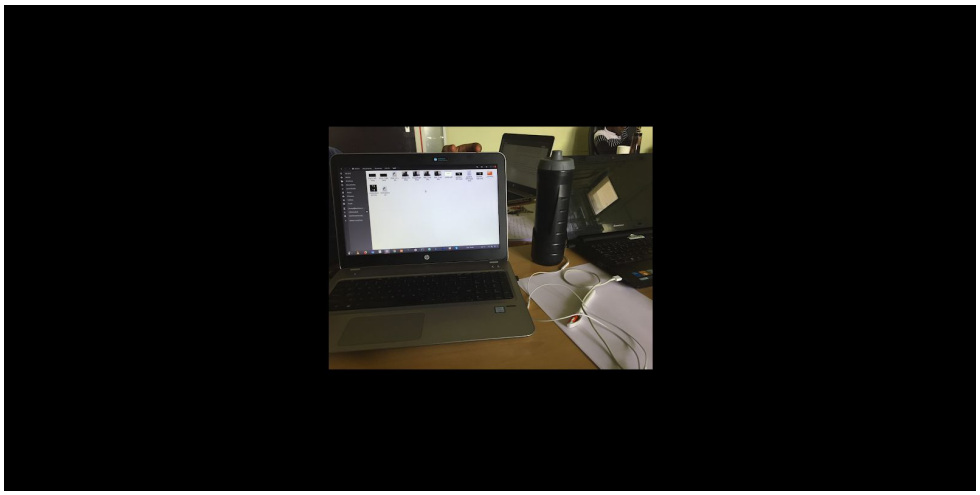
Left

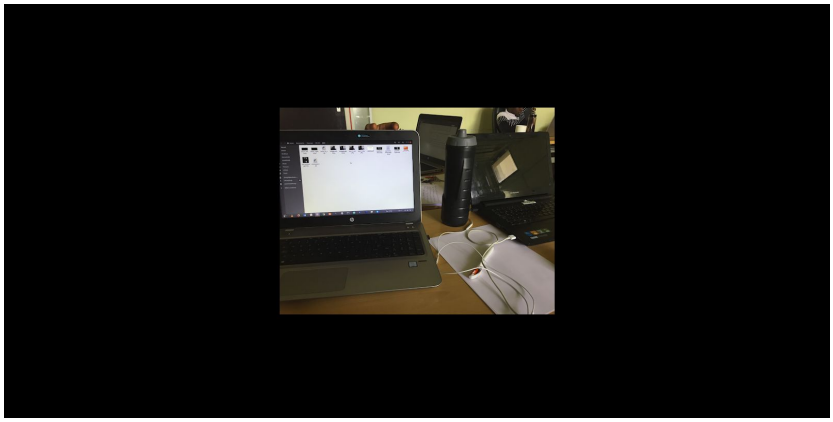


Right



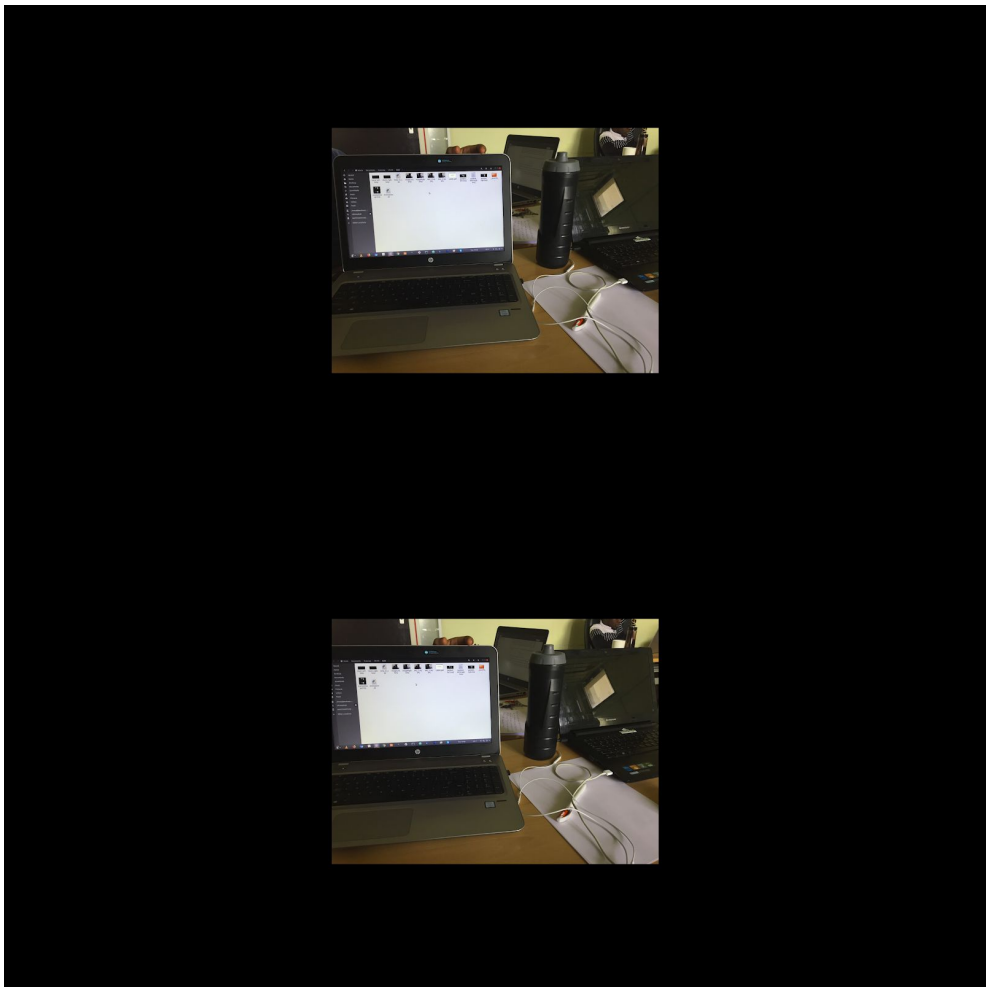
After processing the Images and adding the padding (4096 X 2048), the images looks like shown below:
Left padded image





Right padded image

The next after this is to stack the 2 images into a 4096 by 4096, the resulting image is shown below



1.2 Discussion

Noticeably in the Oculus Go the image is warped inwards, also if the user closes one eye, he or she is able to see two distinct images, i.e the left image if the right eye is closed and vice versa.

2. Conclusion

To conclude this exercise reveals several things, one is that fact each eye see an independent image. Also it was very important when taking the pictures to only move the pictures a minimal distance, otherwise the resulting image is not very clear, it actually stresses the eye.

3. Appendix

Stereopairs.py

```
import numpy as np
import bmp_io_c
import math
from numpy.linalg import inv

# reading the left phone camera image
rows_left_image, cols_left_image, pixels_left_image =
bmp_io_c.input_bmp_c("imageLeft.bmp")
print(rows_left_image, cols_left_image)
# print(pixels_left_image)

# reading the right phone camera image
rows_right_image, cols_right_image, pixels_right_image =
bmp_io_c.input_bmp_c("imageRight.bmp")
print(rows_right_image, cols_right_image)
# print(pixels_right_image)

# creating a 4096 by 2048 black images
black_left = np.zeros([3, 2048, 4096], np.uint8)
bmp_io_c.output_bmp_c("black_left.bmp", black_left)

black_right = np.zeros([3, 2048, 4096], np.uint8)
bmp_io_c.output_bmp_c("black_right.bmp", black_left)

# padding the left and right images with the black frame
```

```
padded_left1 = np.pad(pixels_left_image[0], ((513, 513), (1367, 1366)),
'constant')
padded_left2 = np.pad(pixels_left_image[1], ((513, 513), (1367, 1366)),
'constant')
padded_left3 = np.pad(pixels_left_image[2], ((513, 513), (1367, 1366)),
'constant')

padded_left = np.array([padded_left1, padded_left2, padded_left3], np.uint8)
bmp_io_c.output_bmp_c("padded_left.bmp", padded_left)

padded_right1 = np.pad(pixels_right_image[0], ((513, 513), (1367, 1366)),
'constant')
padded_right2 = np.pad(pixels_right_image[1], ((513, 513), (1367, 1366)),
'constant')
padded_right3 = np.pad(pixels_right_image[2], ((513, 513), (1367, 1366)),
'constant')

padded_right = np.array([padded_right1, padded_right2, padded_right3],
np.uint8)
bmp_io_c.output_bmp_c("padded_righ.bmp", padded_right)

#stacking the 2 images
stacked_image = np.concatenate((padded_left, padded_right), 1)
print(stacked_image.shape)
bmp_io_c.output_bmp_c("stackedImage.bmp", stacked_image)
```