# User Manual

He Liu

10/8/15

## **Write at the beginning**:

I tried to make this program executable, but somehow there are some import problem with OpenCV that I cannot actually solve, so what I hope is you can run this code on a python compiler. This small program was make in less than a week, and I believe there are many place that I can improve, like the optical flow and motion vector and block matching. I did tested some of them but it needs more time to make these techniques stable. And I know there must be bugs in this program, but the basic purposes were implemented in this program.

## **Requirements:**

This program was wrote in Python 2.7 in Windows 8.1 so it needs a python 27 compiler. And this program needs the following packages to run:

1. Tkinter (buid-in in python 2.7)
2. Pillow (PIL)
3. OpenCV version 3.0
4. Numpy

## File Includes:

1. test.py: the main python file to run

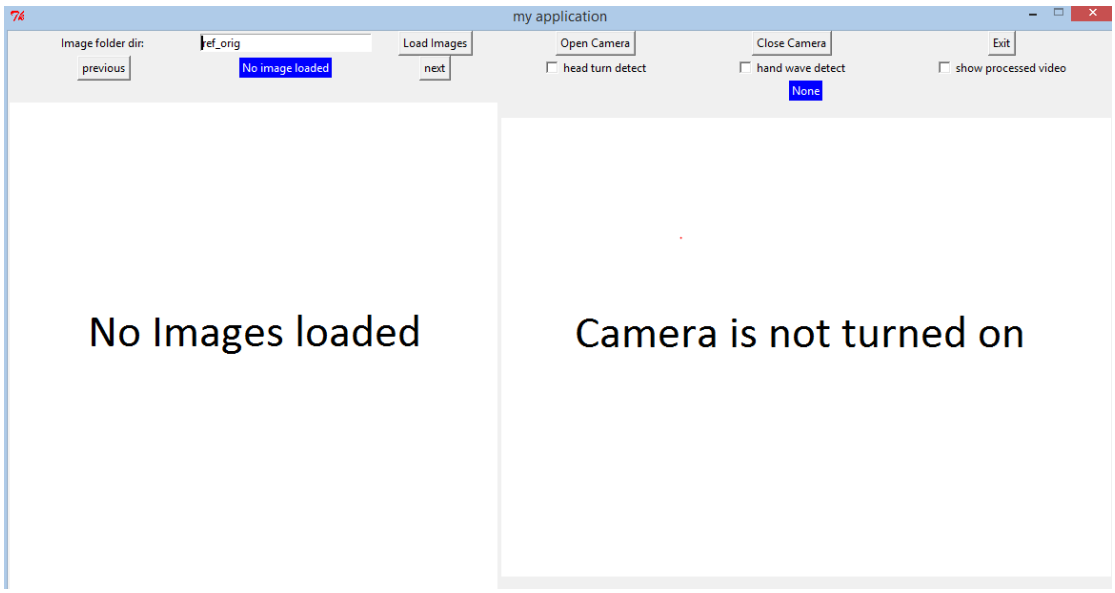
2. A folder called ref_orig, which contains several 512*512 images from SCIQ database from Oklahoma State University Computational Perception and Image Quality Lab. This folder gives the default image data for this testing program.

3. Image: ‘Camera.png’ and ‘No Image loaded.png’, the default image to show on the program.

4. haarcascade_eye.xml: the data for the program to detect the eyes.

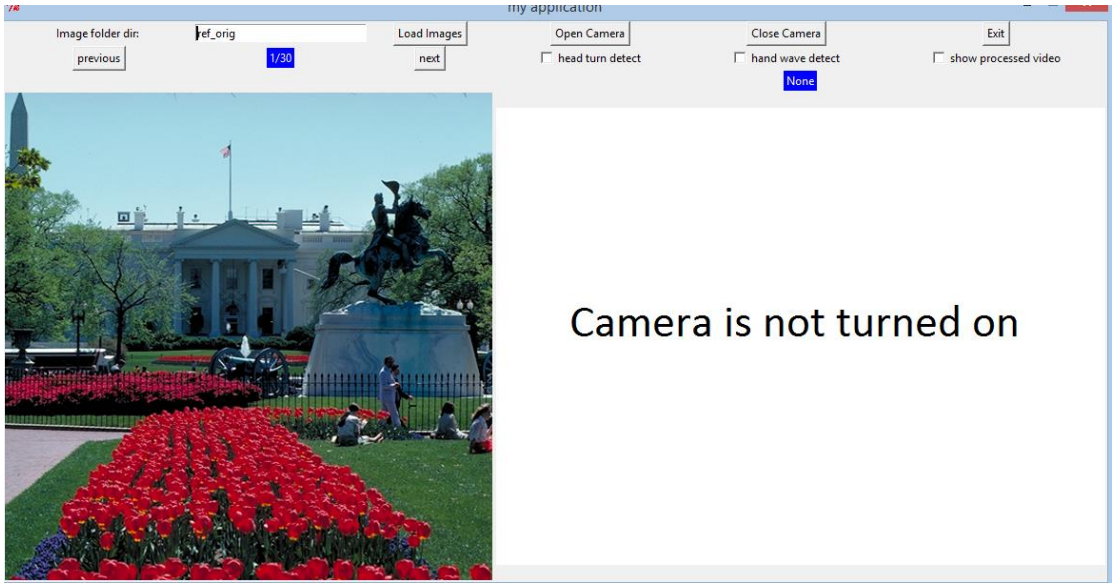
## User instructions:

When the program is executing, the following figure shows the GUI.




## Image presenting

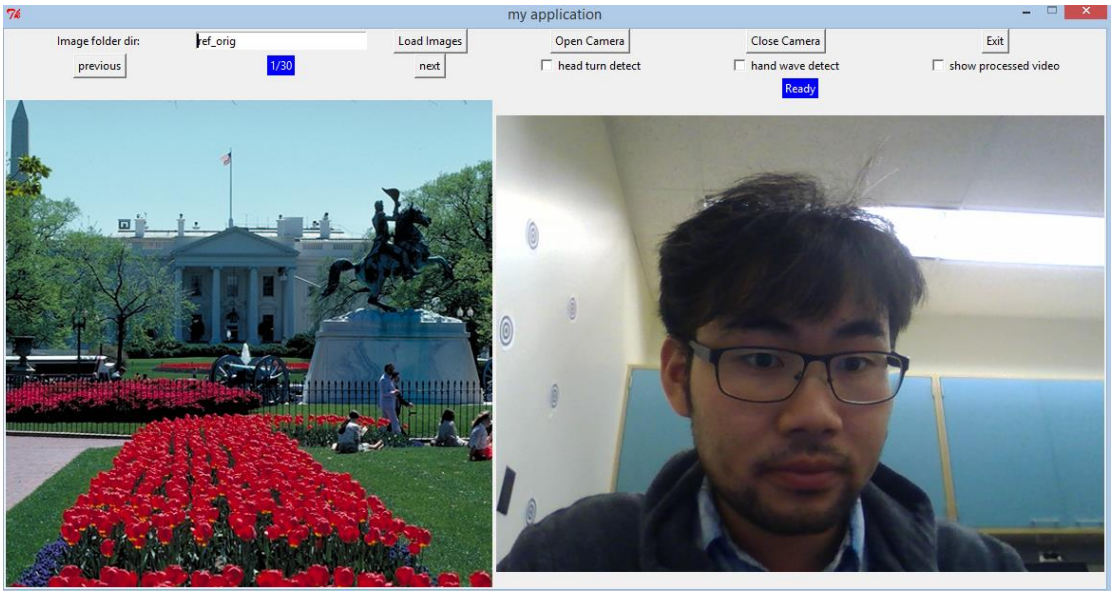
The left side of the program is image showing part. There is a text entre and here you can put your own image folder into it. It is initialized by the CSIQ images. If the image folder is not correct it will not show anything. If we use the default image folder and press load image button, we get image loaded.




Then we can press next and previous button to find change the images inside the folder. The index above shows the image index.

This program supports image formats: 'PNG','JPG','JP2','BMP','TIF' and 'GIF'. All images will be resized to 512*512 to fit this program.

On the right side we have the camera part. By pressing open camera button, and if the camera is working well, it will show the image from the camera. The close camera button is to shut camera down. Exit button can be pressed anytime to exit the program.



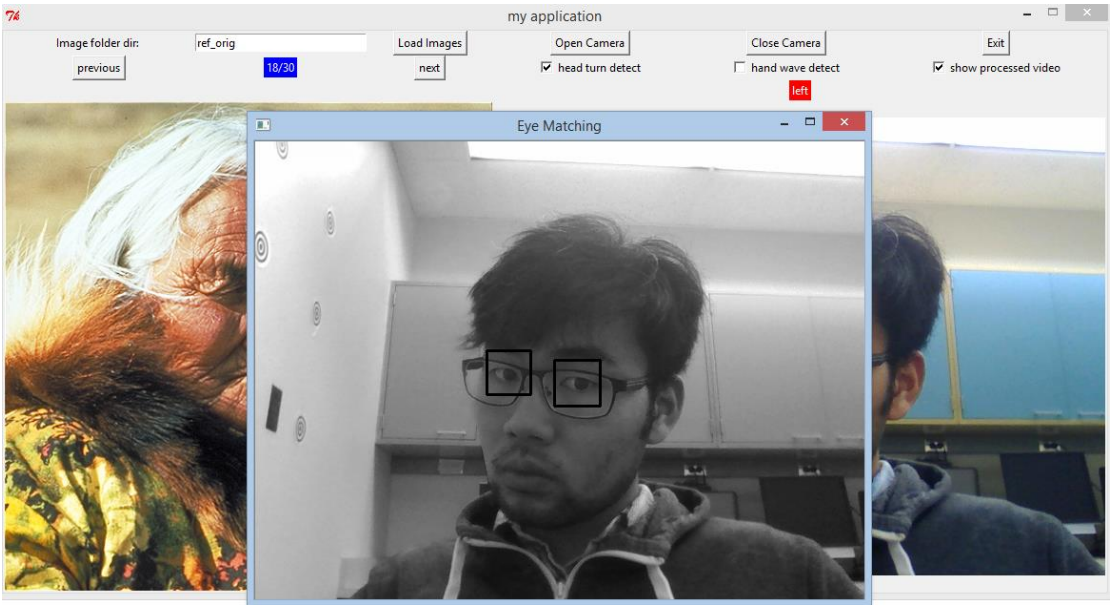

There are three tickers above the camera image. First and second ticker gives two detection modes and last one is to show helper video.

## Head turn detect mode:

By ticking ‘Head turn detect’, the program will detect the eyes of the user through camera and once the user turning the head, the program will show the next or previous image on the left side. And the prompt above the camera will show ‘left’ or ‘right’ in red which side head turns. In addition, I made a ‘Beep’ sound for both turning right and turning left.

If the prompt turns to blue and show ‘Ready’ again, it can detect head turning again, this is for the case when the user did not turn his or her head back and face the camera. By ticking the ticker ‘show processed video’ will show the eye detected video, with eyes rectangular in a new window. By ticking the ticker ‘show processed video’ again, it will close the

gray image window.



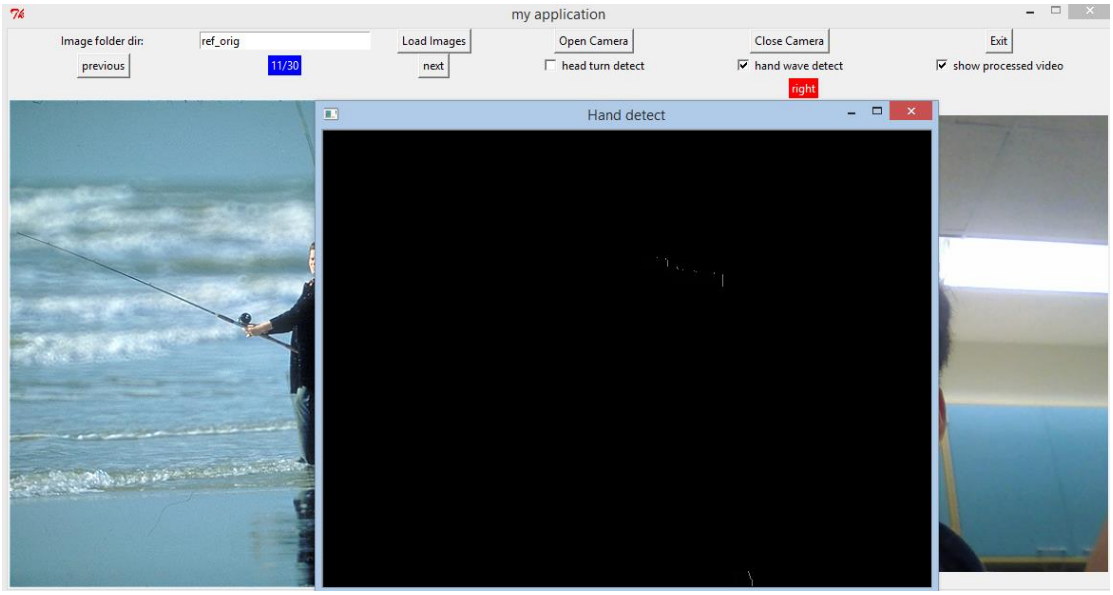

If you turn your head so fast, it probably will not detect that. And if your hair or glasses cover your eyes, it is better to remove them if the program does not detect your eyes. You can swing your head a little bit and the program will not determine it is a turning.

The basic idea of this is to measure the center position of eyes and see if it moves left or right. The eye detection function and data are from OpenCV.

## Hand wave detect mode

For ‘hand wave detect’ ticker, it will detect if there is a hand waving in front of the webcam. You can put your hand close to the camera and wave

your hand from left to right or right to left, and of course you will hear the ‘Beep’ sound and see the image change. Also the prompt above will show the direction in red and it will go back to ‘Ready’ in blue after your hand move out of the screen. And there is also a processed video for this detection. The processed video shows the changes of the background.




For the part, the best case is that your hand can cover 1/5 of the camera screen area when you move your hand in front of the camera. When the red prompt turns blue again, you can wave your hand again. The speed of waving is the hardest part of this, it is hard to control. The webcam on my PC is slow, so if I move my hand so fast that the camera cannot catch 3 frame of my hands, it will not work. But if I move so slow and the program will turn my hand into the background.

The basic idea is Gaussian background model and background subtraction. It will eliminate all the un-moving parts of the camera and detect the moving parts in the camera. The processed video shows the frame after

background subtraction.

## Other situations

If you would like to turn both detection on together, it can work but the PC will get slow for both detections. So if you will move your hand, you need to slow down a little bit for the program to process. And for eye detections, you probably need to move your head faster for this case. However, I set the program to stop detect once a motion is already detected. So if the prompt is red, it will not detect again once it goes back.

Two parts on the left and right side are kind of independent, you really do not need to turn on the webcam and load the image or try the webcam after you load the images.

For a general reminder, it is better to close the program by pressing the exit button when the camera is on because if you click the right top X button on the window, it will close the window but not stop the camera loop in the program.