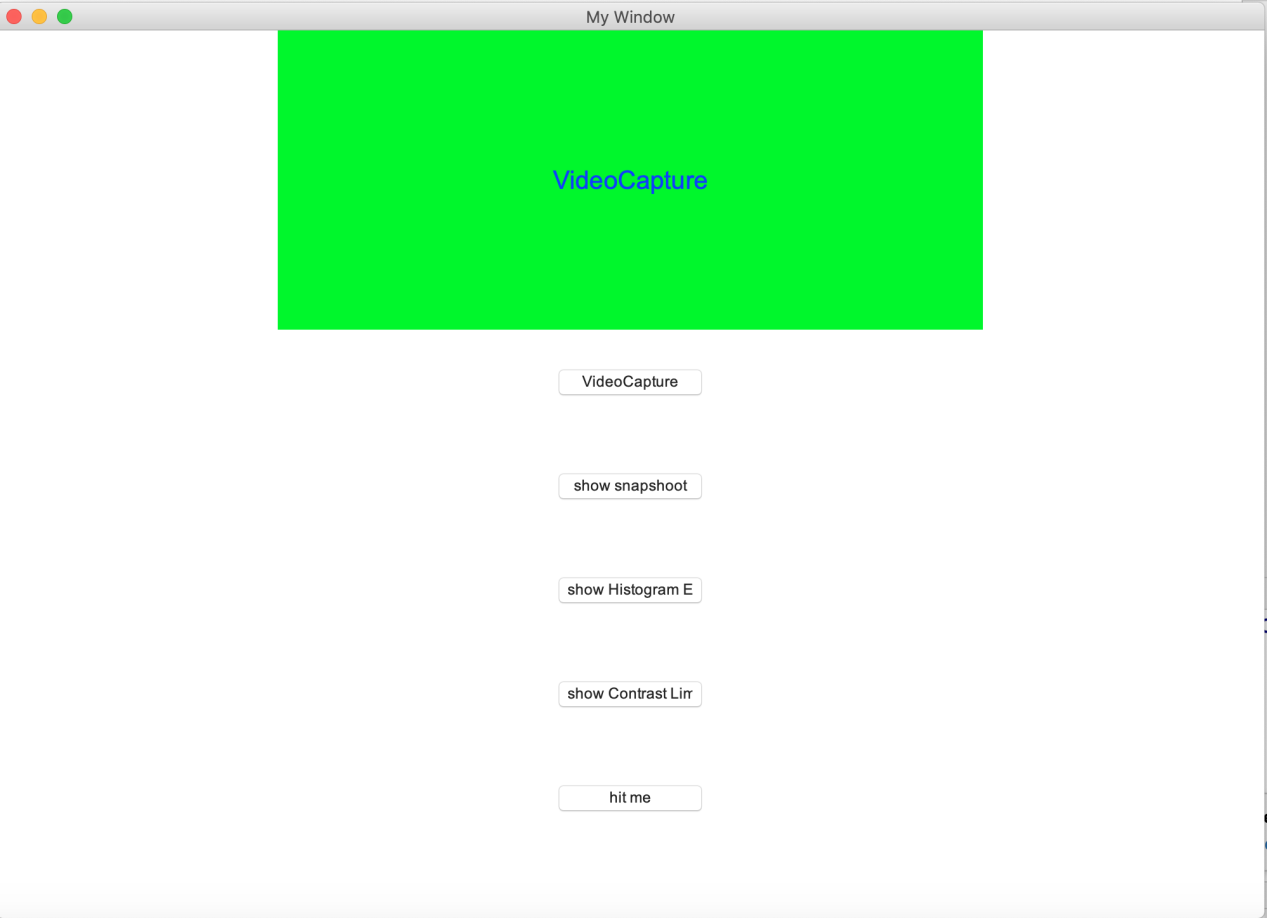
**Assignment Report**

**Xiang Wang 1106943**

This document shows my assignment details which include the diagram of the procedure, the definition of my functions, and description of the method. Also I post snapshot and the result in this document. And I described some method through coding comment step by step,which can clear demonstrate my idea. According the comment (**Begin with # signal** ) , it can let the readers know my ideas easily.Finally I write a conclusion to review this assignment.

First, I designed a GUI interface. In this interface, I designed a number of buttons. Clicking these buttons will open the camera, display the image, display the histogram equalization, and contrast locally adaptive histogram equalization.

1. Design a interface to implement GUI



**def** center\_window(w, h):

*# Get the screen width*, *height*

ws = window.winfo\_screenwidth()

    hs = window.winfo\_screenheight()

*# Calculate x*, *y position*

x = (ws/2) - (w/2)

    y = (hs/2) - (h/2)

    window.geometry(**'%dx%d+%d+%d'** % (w, h, x, y))

center\_window(1000, 700)

*# Step 4*, *set the label on the graphical interface*

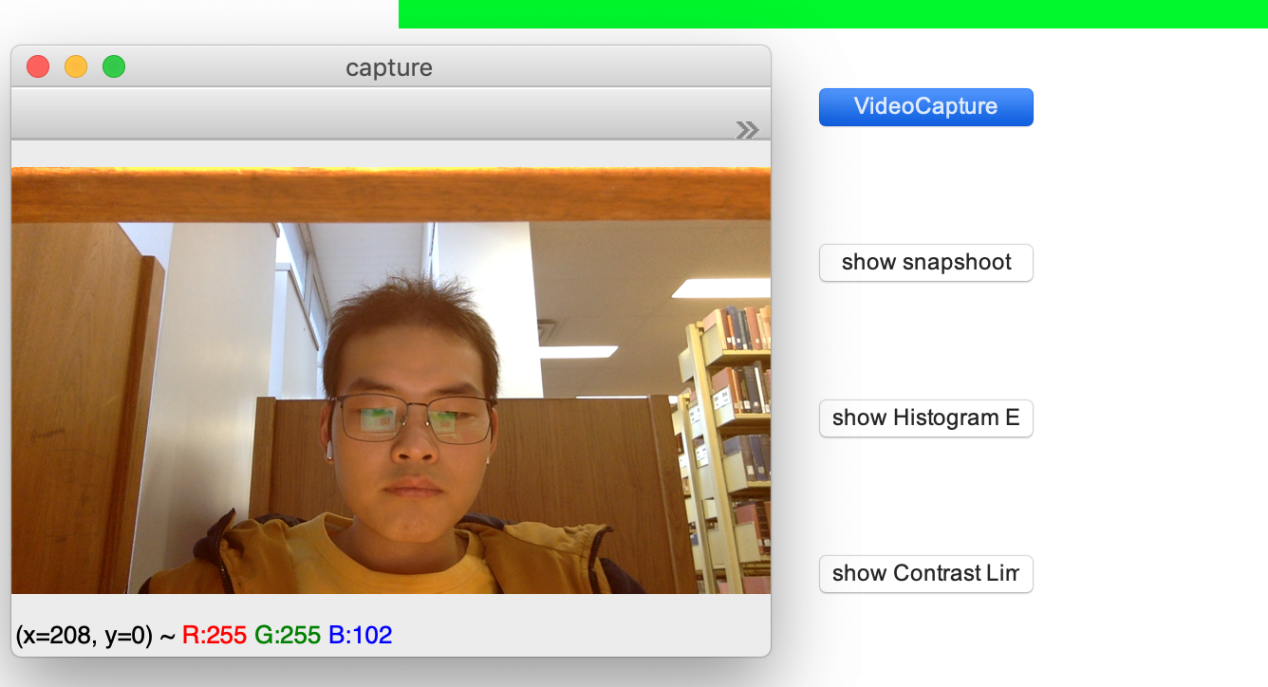
var = tk.StringVar()  *# Set the contents of the label tag to the character type*, *and use var to receive the outgoing content of the hit\_me function for display on the label.*

l = tk.Label(window, textvariable=var, bg=**'green'**, fg=**'blue'**, font=(**'Arial'**, 20), width=50, height=10)

*# Description: bg is the background*, *fg is the font color*, *font is the font*, *width is long*, *and height is high. The length and height here are the length and height of the character. For example*, *height=2*, *the label has 2 characters so high.*

l.pack()

1. design a method to open the camera



on\_Webcam = **False**

**def** Webcam\_capture():

**global** on\_Webcam

    var.set(**'VideoCapture'**)

**if** on\_Webcam == **False**:

*#set a video capture instance*

cap = cv2.VideoCapture(0)  *# open the video*

*# Setting the size of the screen*

*# Picture width is set to 9920*

cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 9920)

*# setting the size of the height 9080*

cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 9080)

*#Create a window called "capture"*

*#Window attribute flags*

*# Widow\_normal* ： *Window can shrink*

*# widow\_keepration :Maintain ratio during window scaling*

*# window\_Gui\_expanded : Gui window enhanced with new version features*

cv2.namedWindow(**'capture'**, flags=cv2.WINDOW\_NORMAL | cv2.WINDOW\_KEEPRATIO | cv2.WINDOW\_GUI\_EXPANDED)

**while** (1):

*# get a frame*

*# If the screen is read successfully*, *ret= true*, *frame is the image object read* (*numpy ndarry format*)

ret, frame = cap.read()*#Get the picture frame by frame If the picture is read successfully*

*# show a frame*

*# Update window The picture window in capture is automatically adjusted to the image size.*

*# The first parameter is the name of the window*, *followed by the image. Can create multiple windows*, *but must give them different names*

cv2.imshow(**"capture"**, frame)  *# Generate camera window*

**if** cv2.waitKey(1) & 0xFF == ord(**'q'**):  *# If you press q*, *the screenshot is saved and exited.*

cv2.imwrite(**"/Users/wangxiang/Downloads/test.png"**, frame)

**break**

cap.release()

        cv2.destroyAllWindows()

        var.set(**'VideoCapture'**)

**else**:

        on\_Webcam = **False**

var.set(**''**)

1. 显示摄像头截图



on\_showimg = **False**

**def** Show\_img():

**global** on\_showimg

    var.set(**'Show Snapshoot!!!'**)

**if** on\_showimg == **False**:

*# on\_showimg = True*

*#Read the file under the specified directory folder and this file*

img = cv2.imread(**"/Users/wangxiang/Downloads/test.png"**)

cv2.imshow(**"test"**, img)

cv2.waitKey(0)

*#cv2.destroyAllWindows*()

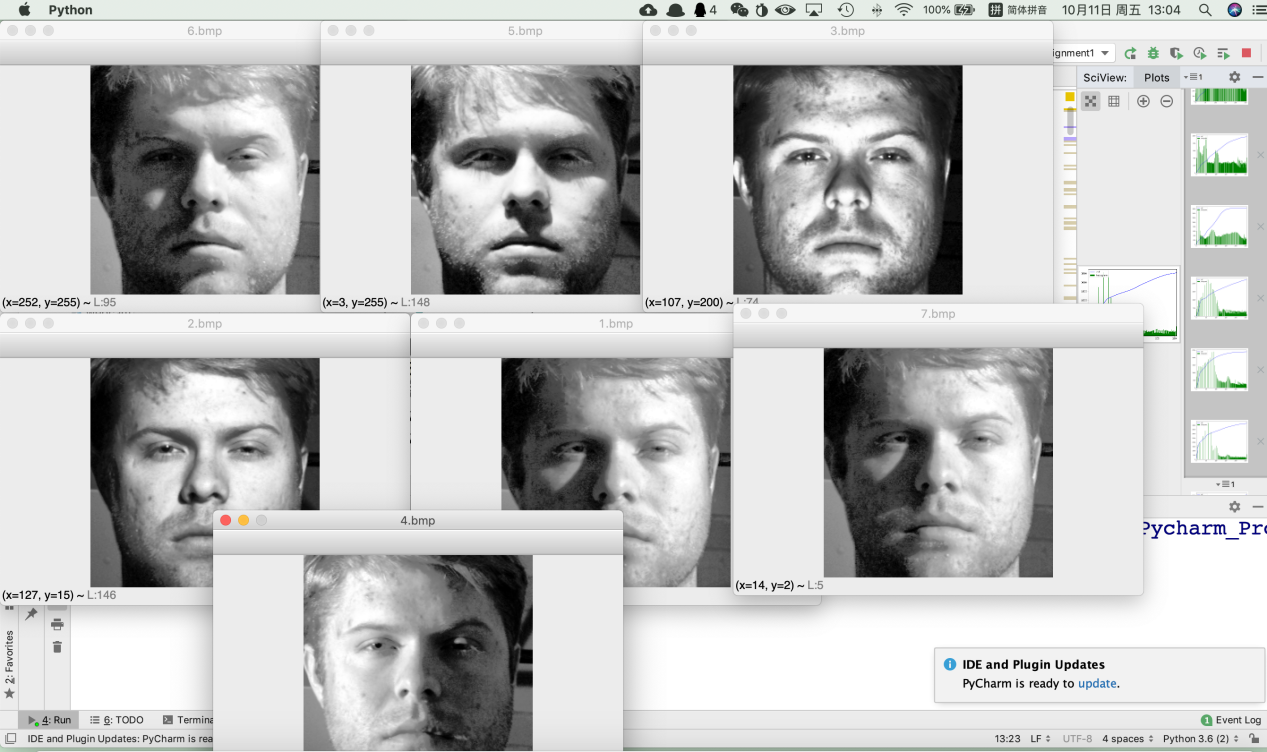
**else**:

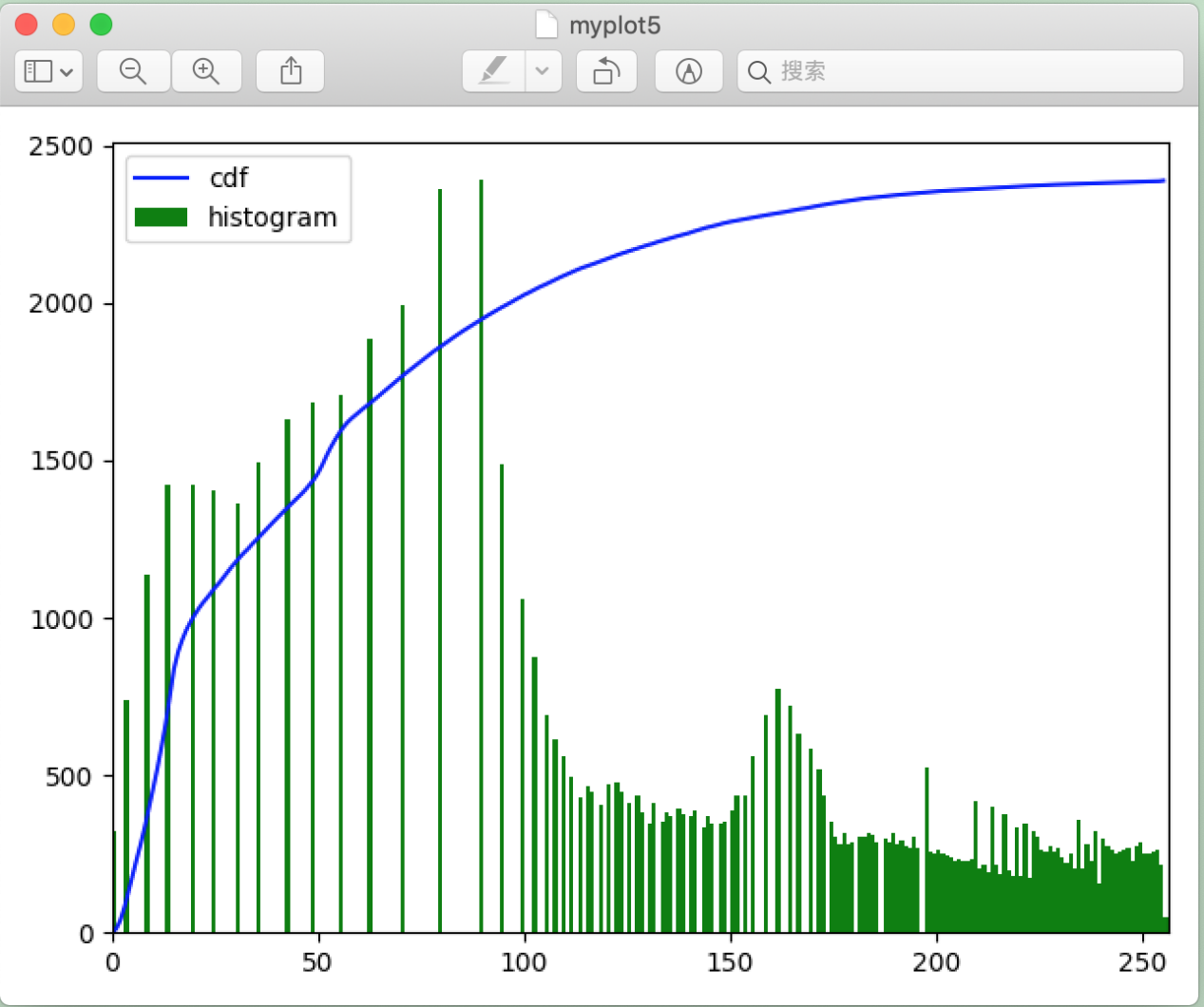
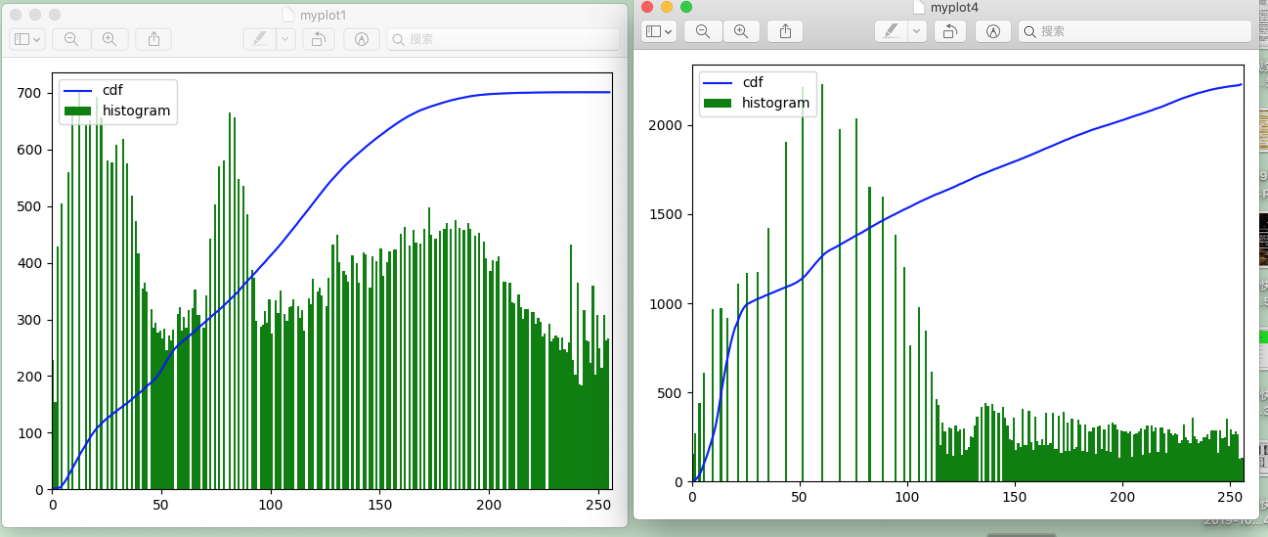
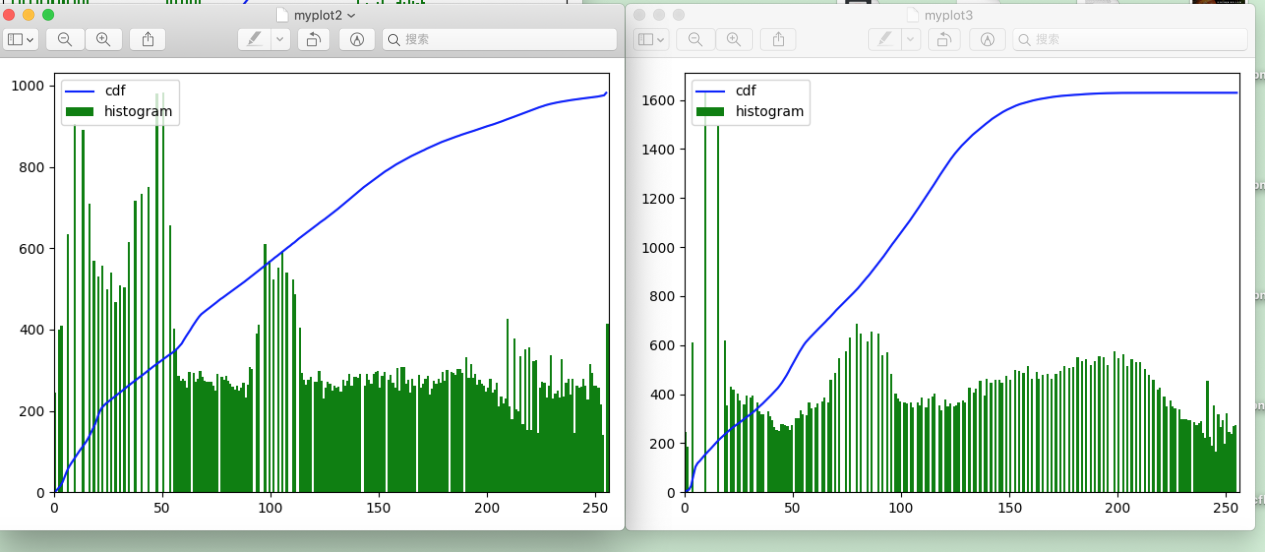
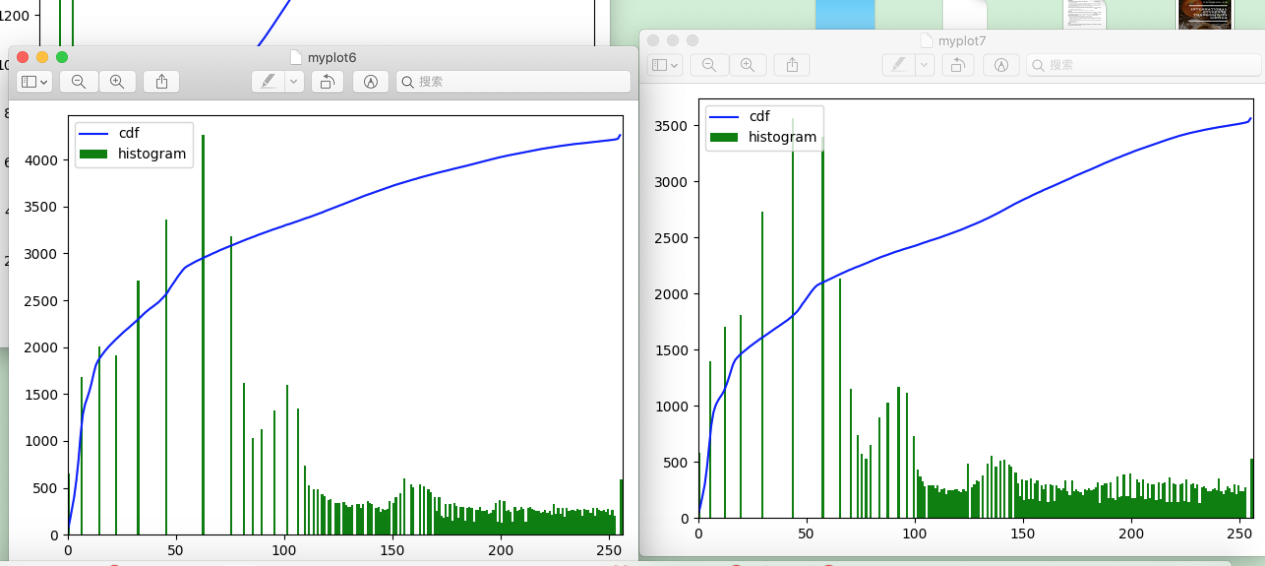
        on\_showimg = **False**

var.set(**''**)

1. Use histogram equalization to process photos

And the result showed the below.





#Histogram equalization is used to make all pictures have the same brightness conditions. The histogram is equalized so that pictures of different brightnesses reach the same brightness condition.

**def** Histo\_Equ():

**global** on\_he

**if** on\_he == **False**:

*#on\_hit = True*

**for** i **in** range(1,8):  *# choose loop times*

img=cv2.imread(**"/Users/wangxiang/Downloads/%d.bmp"**%(i), 0)#choic the image read path

var.set(**'Histogram Equalization'**)

#The numpy.histogram() function takes the input array and bin as two arguments. The contiguous elements in the bin array are used as the bounds of each bin.

            hist, bins = np.histogram(img.flatten(), 256, [0, 256])  *# img.flatten*

*#Turn an array into a one-dimensional array*

*# Calculate histogram*

cdf = hist.cumsum()

cdf\_normalized = cdf \* hist.max() / cdf.max()

#Build a Numpy mask array, cdf is the original array, and mask when the array element is 0

            cdf\_m = np.ma.masked\_equal(cdf, 0)

            cdf\_m = (cdf\_m - cdf\_m.min()) \* 255 / (cdf\_m.max() - cdf\_m.min())

*#* Assign a value to the masked element, where the assignment is 0

cdf = np.ma.filled(cdf\_m, 0).astype(**'uint8'**)

#Now I have a table, I can find the value of the output pixel corresponding to the input pixel by looking up the table. I only need to apply this transformation to the image.

            img2 = cdf[img]

            cv2.imshow(**'1.bmp'**, img2)

            cv2.waitKey(1)

            cv2.imshow(**'%d.bmp'**%(i), img2)

            plt.plot(cdf\_normalized, color=**'b'**)

#Using Matplotlib : Matplotlib has a histogram drawing function: matplotlib.pyplot.hist()

#Matplotlib converts the digital representation of a histogram into a graph.The plt() function of the pyplot submodule takes an array of data and bin arrays as arguments and converts them into a histogram.

            plt.hist(img2.flatten(), 256, [0, 256], color=**'green'**)

            plt.xlim([0, 256])

            plt.legend((**'cdf'**, **'histogram'**), loc=**'upper left'**)

            plt.show()

            cv2.waitKey(1)

**else**:

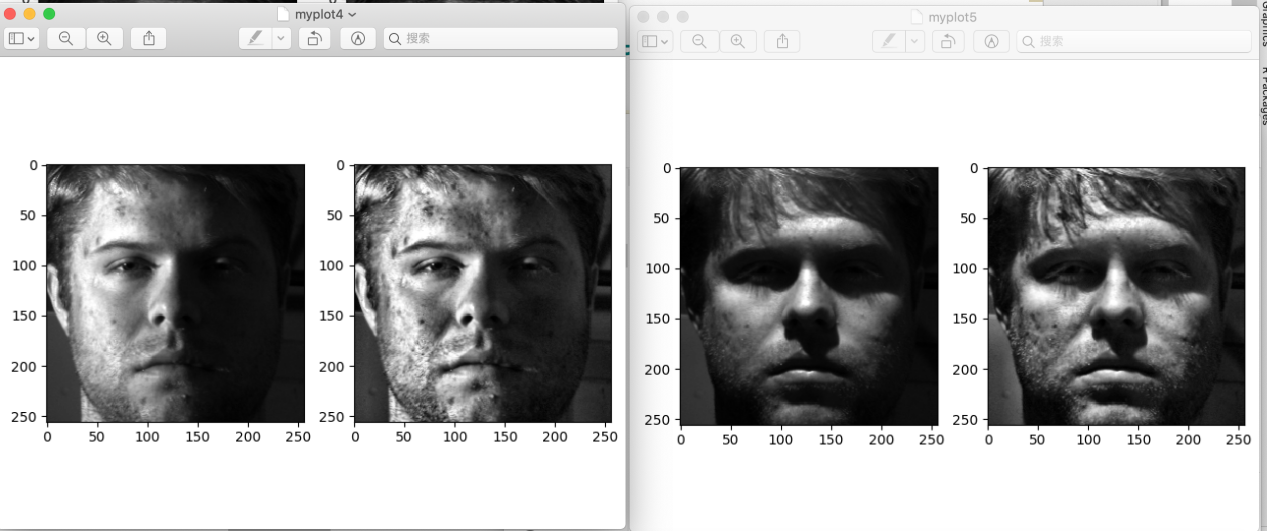
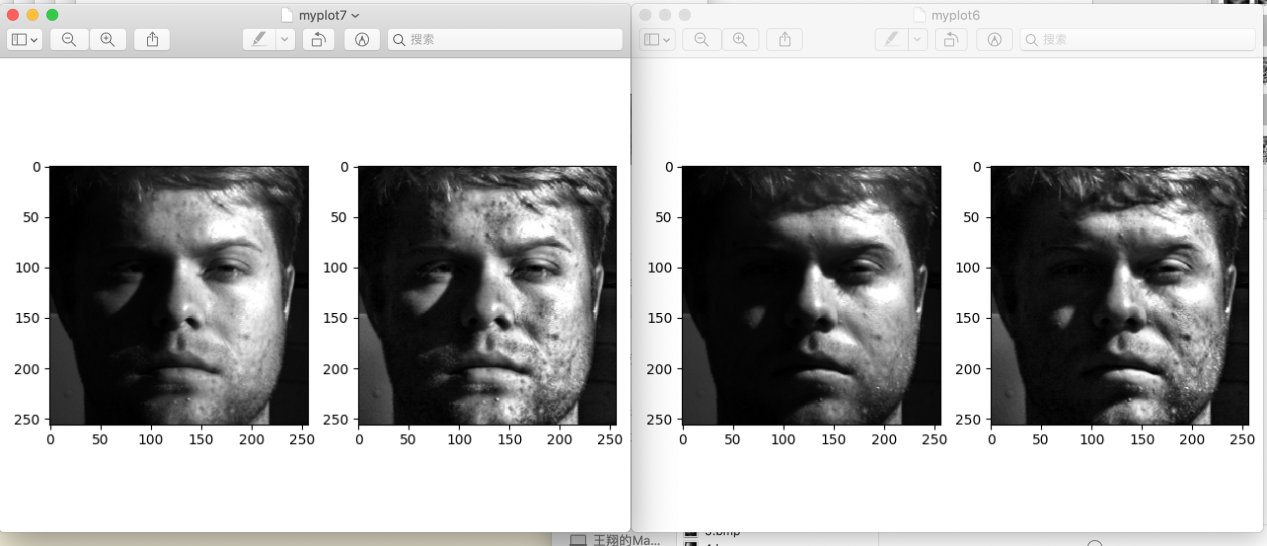
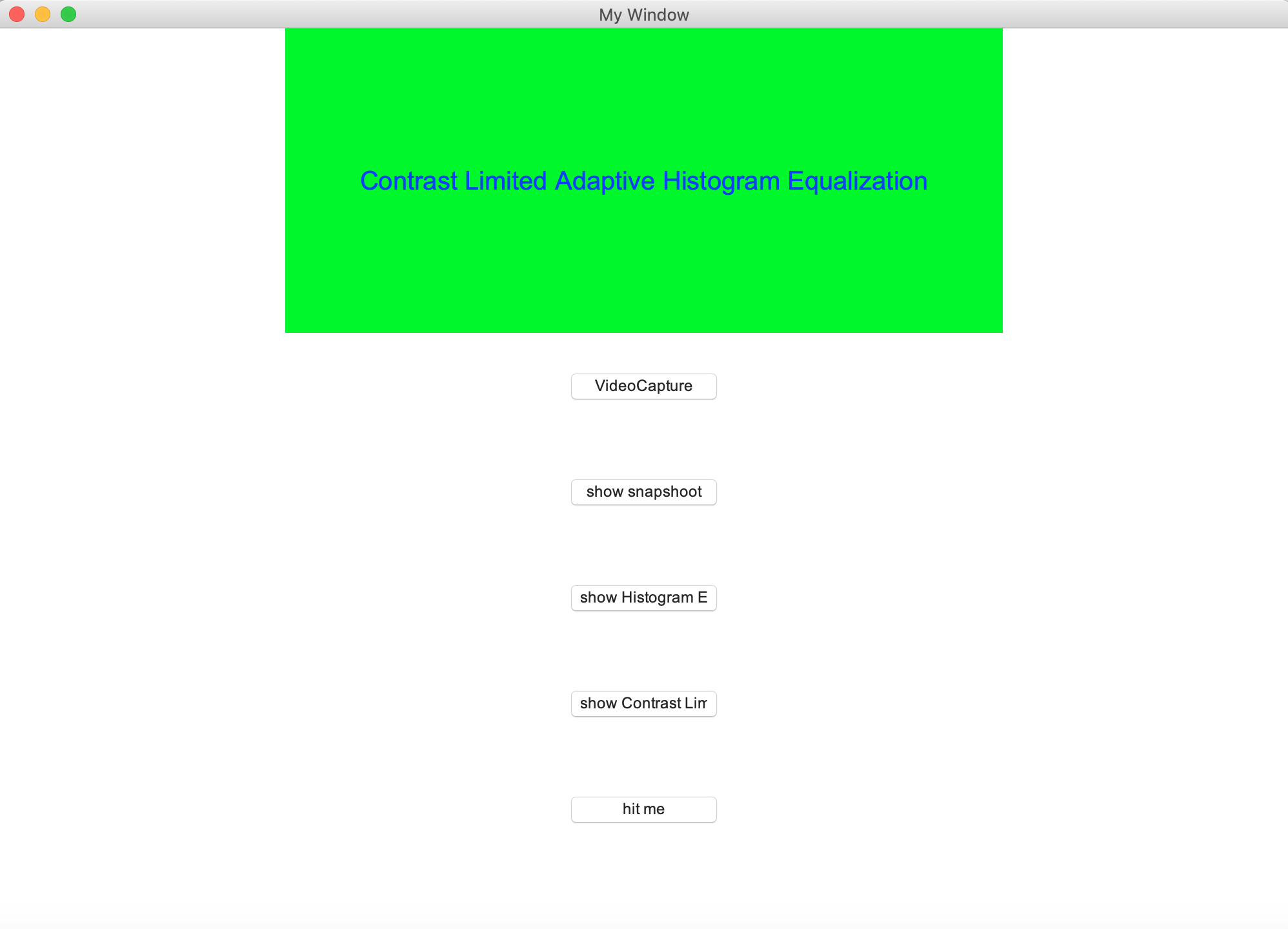
        on\_he = **False**

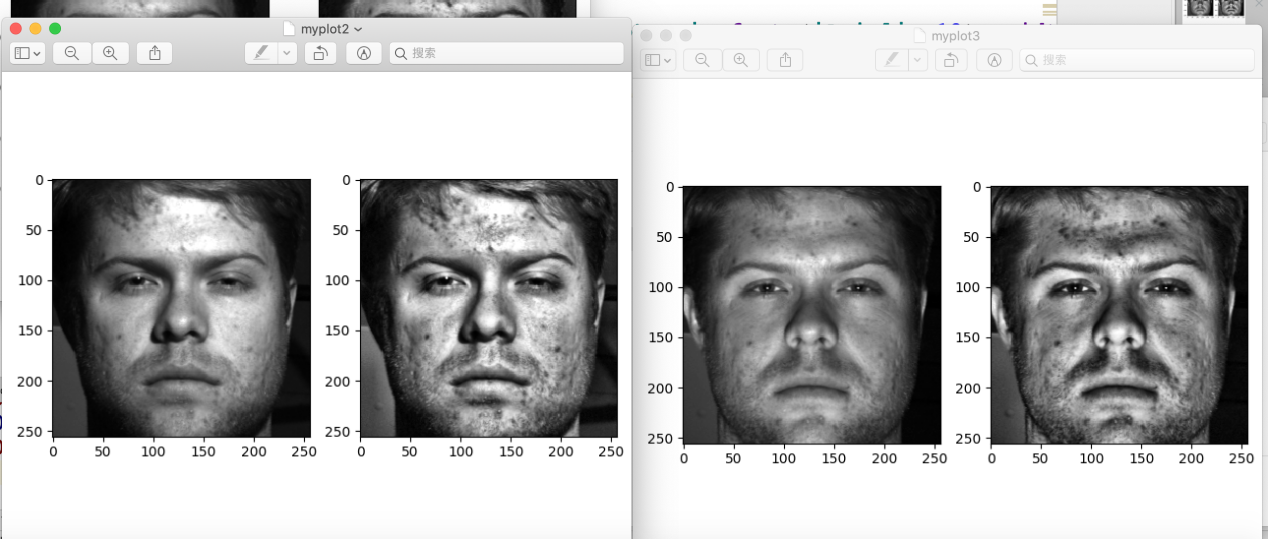
var.set(**''**)

5.design contrast locally adaptive histogram equalization

#Due to the histogram equalization, the contrast is changed, which causes the problem of losing picture information. The root cause of this result is that the histogram of this image is not concentrated in a certain area. In order to solve this problem, we need to use adaptive histogram equalization. In this case, the entire image is divided into a number of small blocks, and then each of the small blocks is separately histogram equalized. So in each region, the histogram will be concentrated in a small area . To avoid noise, use contrast limited histogram equalization. For each small block, if the bin in the histogram exceeds the upper limit of the contrast, the pixels are evenly dispersed into other bins, and then histogram equalization is performed. Finally, in order to remove the "artificial" boundary between each small block, the small block is stitched using the bilinear difference.

on\_clahe = **False**

****

****

**def** CLAHE\_img():

**global** on\_clahe

**if** on\_clahe == **False**:

        var.set(**'Contrast Limited Adaptive Histogram Equalization'**)

**for** i **in** range(1, 8):

            img = cv2.imread(**'/Users/wangxiang/Downloads/%d.bmp'** %(i), 0)  *#* Read directly as a grayscale image

#CLAHE Function parameter：clipLimit is the contrast and tileGridSize controls the size of each processing area.

clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(10, 10))

            cl1 = clahe.apply(img)

            plt.subplot(121)

            plt.imshow(img, **'gray'**)

            plt.subplot(122)

            plt.imshow(cl1, **'gray'**)

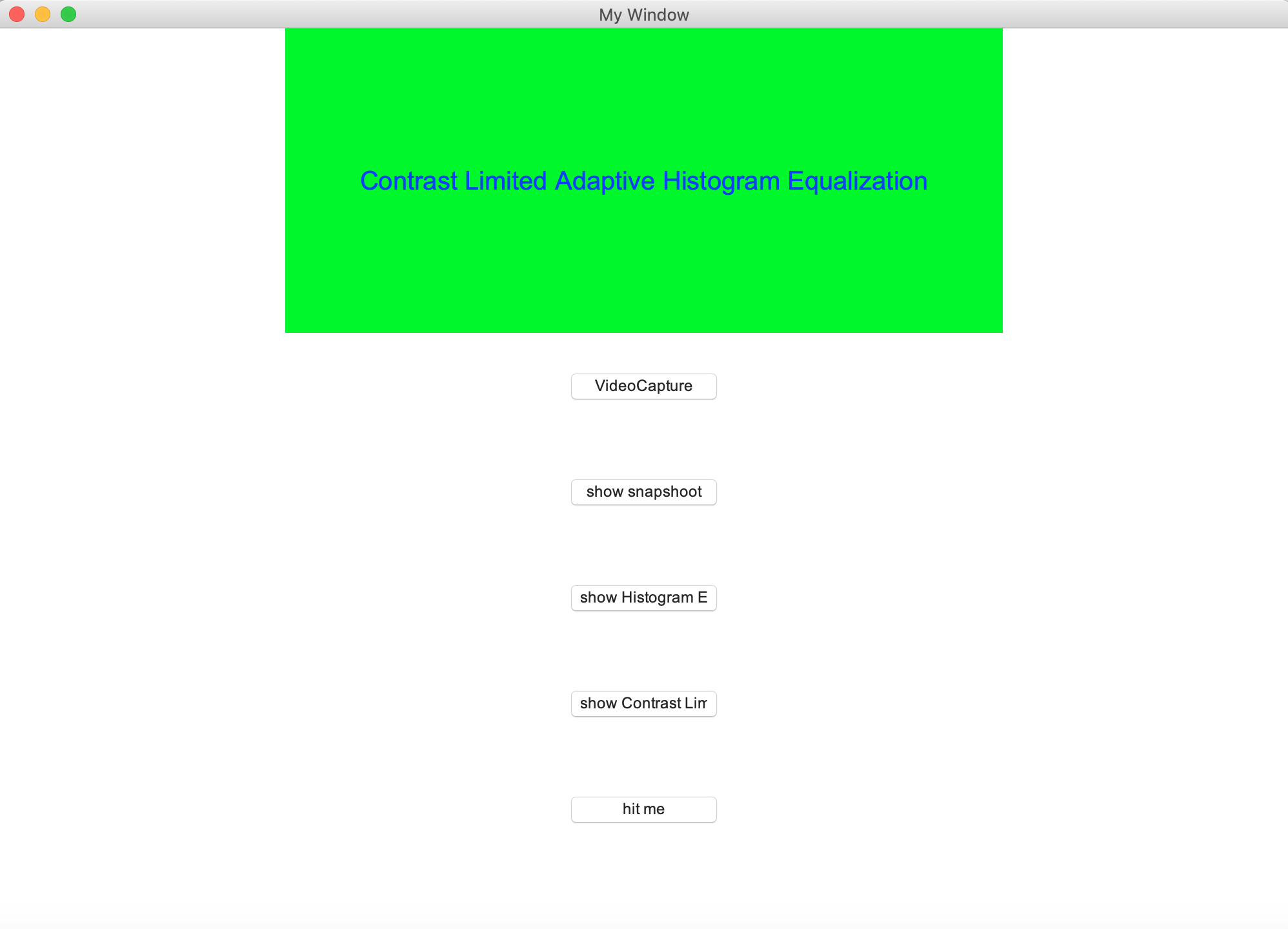
            plt.show()

**else**:

        on\_clahe = **False**

var.set(**''**)

1. *set the Button button in the window interface setting*

**

b1 = tk.Button(window, text=**'VideoCapture'**, font=(**'Arial'**, 12), width=15, height=5, command=Webcam\_capture)

b1.pack()

b2 = tk.Button(window, text=**'show snapshoot'**, font=(**'Arial'**, 12), width=15, height=5, command=Show\_img)

b2.pack()

b3 = tk.Button(window, text=**'show Histogram Equalization'**, font=(**'Arial'**, 12), width=15, height=5, command=Histo\_Equ)

b3.pack()

b4 = tk.Button(window, text=**'show Contrast Limited Histogram Equalization'**, font=(**'Arial'**, 12), width=15, height=5, command=CLAHE\_img)

b4.pack()

b5 = tk.Button(window, text=**'hit me'**, font=(**'Arial'**, 12), width=15, height=5, command=hit\_me)

b5.pack()

*#main window loop display*

window.mainloop()

**Conclusion**：In this assignment, I designed the GUI interface, which can be implemented by clicking the button in the interface. Using opencv to open the camera, capture the screen, and show the screenshot. According to the pictures given by the teacher, I designed histogram equalization to process the pictures. In image processing, the histogram is mostly located in a brighter area, and it is desirable to cover the entire area more gently. To do this, I designed a conversion function, the histogram equalization function, which maps the input pixels of the bright area to the output pixels of the entire area. Finally, the histogram equalization will cause the loss of important information of the picture. I used the Contrast Limited Adaptive Historgram Equalization function, the image will be divided into several small blocks, and the histogram equalization will be performed for each small block, which can achieve good results .