Name:

Partner:

**Module 7 Lab 2 – Facial Recognition**

Module by: Cheng Peng; Edited by: Kevin Ho and Cheng Qu

**Objective:**

1. Lab objective:   
   Students will learn the basic theory behind facial recognition. Students will gain experience with Kinect hardware and OpenCV in C++. By the end of this lab, students will be able to implement facial recognition in OpenCV to: (1) perform a Gaussian blur on faces recognized in a scene, (2) replace faces in a scene with an image (of your lab partner!).
2. Motivation:   
   Facial recognition is a massively important area of Computer Vision. Not only does it save you precious seconds of your life by allowing you to look at your phone and automatically unlock it, but also it saves you precious minutes of your life by automatically tagging your friends on social media! On a more serious note, facial recognition can be used to take classroom attendance, find missing persons, or track down runaway criminals. The field is still growing; various problems still exist. How do lighting, facial expressions, occlusion, and more affect facial recognition algorithms? Although you won’t be answering these questions in today’s lab, we hope this exercise may help spark your curiosity.
3. Key concepts:
   1. OpenCV in C++
   2. Haar cascade
   3. Haar-like features

**Procedure:**

***I. Introduction***

1. Open computer\_vision.sln in Visual Studio and make sure Lab\_2\_skeleton.cpp is your only file under the *Source Files* folder.
2. Read through the main function of the code to observe how we implemented hand gesture recognition and facial recognition. Answer the following questions and **copy the lines of code onto this document to support your answers.** (*Hint: Around line 163*)

**Question 1:** How many different kinds of gestures can the code detect? What are they?

**Question 2:** How many different hands can the code track at the same time?

**Question 3:** What happens when a hand gesture is detected?

1. Navigate to line 291. All of the code you write will go in our custom function detectAndDisplay. The input parameter Mat frame\_input represents the current frame taken from the Kinect’s RGB camera.

**Question 4:** Under what conditions does the custom function detectAndDisplay get called by the main function? Name the three events which cause flag to toggle.

1. Google the function detectMultiScale (called on line 300) to find its documentation; answer the following questions:

**Question 5:** What does the function detectMultiScale do?

**Question 6:** The output of our call to detectMultiScale is stored into the variable called faces. What is the data type of faces? What does this variable represent?

***Gaussian Blur***

1. We will begin to perform a Gaussian blur on the detected faces. You will use the function [GaussianBlur(...)](https://docs.opencv.org/3.3.1/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1) to do this. Note that both the input and output of this function MUST be a Mat class. Initialize a Mat variable to store the areas of the frame representing faces with the following line:

cv::Mat faces\_matrix;

1. A Rect object contains 4 values that define a rectangle: the x and y pixel coordinates of the upper-left corner, the width, and the height. This object is typically used in OpenCV to define a region of interest within a frame—i.e. one can index a Mat variable by using a Rect variable. Assuming each object has already been initialized, this indexing can be done by the following line of code:

matrix\_var = frame(rect\_var);

**Question 7:** Using the above line of code as a guide, write a line of code which indexes frame by using each of the detected faces, and stores them into the faces\_matrix you initialized in Step 5. This line of code should go after your initialization of faces\_matrix. Copy the line of code and paste it below.

1. Apply a Gaussian blur. You may wish to read up on the documentation of [GaussianBlur(...)](https://docs.opencv.org/3.3.1/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1). Your function call should look somewhat like this. You need to figure out what to put in place of input and output (*Hint: recall the note from step 5*)

GaussianBlur(input, output, Size(23,23), 0, 0)

**Question 8:** What are some constraints on input and output?

**Question 9:** What are the constraints placed on the Size object created in the third parameter?

1. Build and run your code. If you implemented the Gaussian Blur code correctly, you can now wave at the camera, turn on the face searching ability, and subsequently blur out your face!

**Question 10:** Take a screenshot of your Gaussian Blur code and results. Paste them below.

***Face Swapping***

1. We will now begin the process of face swapping. Comment out your Gaussian Blur code before moving on.
2. Take a few pictures of your lab partner. Import the pictures to the computer.
3. Now that you are familiar with some of the operations, here are the steps for replacing the faces in each frame with an image you took:
4. Before the main function, **read in** an image file into a Mat variable. We provide an example of reading an image on line 32.
5. In detectAndDisplay, **resize** the image you loaded to be the same size as the faces detected. You will use the function [cv::resize(...)](https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html?highlight=resize#resize). DO NOT overwrite the original Mat for your image. You will also need to look up how to get the width and height from the Rect object of each face.
6. As you did with the Gaussian Blur, **initialize** a Mat variable to store the regions within frame which represent the faces (see steps 5-6).
7. **Copy** the image you loaded to the Mat variable you initialized in step III. This will rewrite the face region in a frame to the picture of your desire. An example of copyTo is shown on line 295
8. Now, try to use all the pictures you took and go through them at different frames to create an animation effect, where the replaced picture will change with time. You will need to:
9. Read in each image to different Mat variables before the main function.
10. Use a counter to count the number of frames. Use an if conditional on the counter to decide which image should replace the detected faces.
    1. The Kinect records images at 30fps.

**Question 11:** Take a screenshot of your face swap code and results. Paste them below.