1 Assignment 1 (100 points)

In this assignment, we will brush up some C++ fundamentals while working with various data structures in OpenCV to make a very simple object tracker that will allow us to draw a shape using a red ball, and then place that shape around a user's face.

To submit your work, put answers to the written questions in a file(s) at the top level directory. Clearly label all of your work. For the math-oriented portion of the assignment, you may typeset in LaTeX or Word. If you don't know how to correctly typeset math equations, write your solution very neatly, with good labels, on a piece of paper and take a clear, high resolution photograph of the paper. Work that is not legible will not be graded. Clean your Visual Studio Solution, removing all temporary files, then zip up your code, output files, and written answers. Name the file `CS585_Assignment3_username.zip`. Submit the code with web-submit.

1.1 Learning Objectives

- Practice using the Viola and Jones face detector implementation in OpenCV
- Become familiar with basic image region properties
- Become familiar with evaluating a binary classifier
- Learn about using translation and scaling to modify geometric objects
- Learn about technical C++ topics
  - pointers, memory, and type-casting
  - pointers and references
  - STL vectors to hold sequences of things

1.2 Technical Task (points)

Below is the list of given and required functionality.

1. (Given) Finding the largest red object in an image
2. (Given) Finding the outline of an image region
3. (Given) Calculating the area and centroid of an image region bounded by an outline
4. (Given) Drawing the outline of an image region
5. (Given) Using a webcam with OpenCV
6. (Given) Using the Viola and Jones face detector in OpenCV
7. (Required) Moving (translating) and scaling the outline of an image region (Assignment3_Part1)

   Using the given skeleton code that reads in an image, you will implement translation and scaling using simple arithmetic operations performed on Point objects that are stored in a C++ STL vector. The skeleton code has event handling to capture the < > + and - keys. After implementing the scaleOutline and translateOutline functions, you should fill in the body of the if statements for those keys to call your functions to translate left, translate right, increase the scale, and decrease the scale.

When you scale the object, you may notice some strange behavior that doesn't match your expectations. Your program should scale your shapes so that the distance between the outline and the center increases or decreases, but so that the center of the object doesn't move. You may want to use the computeObjectAreaAndCenter to find the center of your outline so that you can do some extra operations to be sure that the center of your object remains fixed when you scale it.

The program takes a command line arguments containing the filename for your shape (example: redStar.png) .In the provided skeleton code, there is a slider for the threshold on the red object. You should ensure that your red shape has been detected if your results do not match your expectations. If you press the space key, the program wil save your current image as output.

An example image for your debugging is "redStar.png"

8. (Required) Putting a shape around a picture of a face (Assignment3_Part2)

   You will use the provided function that wraps the OpenCV implementation of the Viola and Jones face detector. Using your translation and scaling functions, you will place the outline of a shape so that it is centered on the face and scaled to the size of the face. To determine an appropriate scale factor, you can use the area of the shape and the size of the detected face.

   If the size of the object on top of the faces in your image do not match your expectations, you should check your units. You may need to compute a non-linear function of the area of the shape in order to make the units for the scale factor match.

   The program takes two command line arguments: the filename for your shape (example: redStar.png) and an image with your faces (example: faceImage.png). In the provided skeleton code, there is a slider for the threshold on the red object. You should ensure that your red shape has been detected if your results do not match your expectations. If you press the space key, the program will save your current image as output.

9. (Required) Using a C++ STL vector to keep track of the position of a red object over time (Assignment3_Part3)

   A function to find red objects, their centers, and outlines, is provided. Using an STL vector, you should keep a record of the position of the detected red object. Using the drawOutline function, you will display the record of the position of the object over time.

   In the provided skeleton code, there is a slider for the threshold on the red object. You should ensure that your red shape has been detected, if your results do not match your expectations. If you press the space key, the program wil save your current image as output. If you press 't', it will start and stop the tracker. You should ensure that the red object has been detected before you try to run the tracker.

10. (Required) Using a red object to draw a shape, then drawing that shape around an image of a face in live video (Assignment3)

For this part of the assignment, you will need to assemble the parts that you developed in parts 2 and 3 to create a live demo to amaze and dazzle your friends. Suggestions for where to insert your code are given in the assignment skeleton.

When you press "t" it will start and stop the live tracker. When the tracker is stopped, it will draw the shape you have drawn on top of any faces present in the video. If you press the space bar, it will save a video sequence containing both the original video and the resulting video with the tracks and /or faces marked.

The program accepts three command line arguments: a string containing the name of a directory containing images (example: ../trackingSequence2) and a start frame number and end frame number (example 1 and 90).

To summarize:

1. (Given) Finding the largest red object in an image
2. (Given) Finding the outline of an image region
3. (Given) Calculating the area and centroid of an image region bounded by an outline
4. (Given) Drawing the outline of an image region
5. (Given) Using a webcam with OpenCV
6. (Given) Using the Viola and Jones face detector in OpenCV
7. (Required) Moving (translating) and scaling the outline of an image region
8. (Required) Putting a shape around a picture of a face
9. (Required) Using a C++ STL vector to keep track of the position of a red object over time
10. (Required) Using a red object to draw a shape, then drawing that shape around an image of a face in live video

1.3 Programming Assignment Questions (points)

1. Write the formulas for the area (A) and centroid ($\hat{x}, \hat{y}$) of an object (in terms of the pixels (x, y) contained in the region). Your answer should be in the form of a double summation.
2. Write a formula for how to translate each point (x, y) of the outline of your object so it is centered at any point (u, v) in the scene
3. Write a formula for how to manipulate the points in the outline so that the overall size changes by a factor of α, but the center does not move. You will need to use translation for intermediate steps.
4. Draw an outline of a red object in MS Paint (or other program). Use your program to interactively translate your shape two steps to the right, and scale your shape up two "ticks" by pressing > twice and + twice. Save your image as Part1_result.png

5. Take two pictures of yourself (with your phone or with the webcam demo from Lab) where your face appears at two different locations and two different scales. Use your program to display the outline from your red object as a frame around your face. Save your image as Part2_result.png.

6. Record a video of yourself with a red object. As outlined above, fill in the code to use an STL vector to keep track of the position of a red object over time. Draw the trajectory of the red object on top of the final image and save your result as Part3_result.png

7. Using the pieces that you built in parts 2 and 3, fill in the relevant sections of the final part of the assignment. Using a red object as a cursor, draw a shape and place it around any faces in the image. Save a picture of your final output as Assignment3_Result.png

1.4 Written Questions

1. Using the numbers below

   (a) find a threshold that gives the best possible accuracy.

   (b) compute the number of false positives and false negatives

   (c) find a threshold that gives perfect detection of the "1" class

   (d) compute the number of false positives

0.6718	1
- 1.3566	1
1.4658	1
1.0305	1
- 0.4165	1
- 1.9628	0
- 1.2355	0
- 1.7760	0
- 1.6207	0
- 0.3682	0

2. Design a filter to find black dots

3. How would you use the same filter to find white dots?

4. Describe two different ways of finding black and white dots of different sizes

5. Design a filter to blur an image so that, each pixel in the result of the convolution is the mean of the corresponding 3 x 3 neighborhood in the original image

6. How would you modify your filter to compute a mean with wider support? (So, instead of the mean of a 3x3 neighborhood, using a 5 x 5 neighborhood)

7. Describe how you could use an integral image to compute a blurred image.

8. One formula for the variance of a set of numbers is given by

$$\frac{1}{n}\sum_i (x_i^2) - (\frac{1}{n}\sum_i x_i)^2$$

.

   The variance of the image values in a pixel neighborhood cannot be computed using convolution because it is a non-linear operation (it has a square). By examining the two terms in the equation, propose a way that you could compute the variance of pixel neighborhoods using two integral images: one for the sum of the pixel values, and the other for the sum of the pixel values squared.

9. Read this description of the Laplacian of a Gaussian (`http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm`)and write a description of how you could compute an extremely crude approximation using the integral image