1 Assignment 6 (100 points)

To submit your work, collect your inputs not provided as part of the assignment, the required outputs, and your answers to the written questions in pdf format (preferred) or Word, or .txt . Clearly label all of your work. For the math-oriented portion of the assignment, you may typeset in LaTeX or Word or provide a clear photograph of a hand-written solution. Work that is not legible will not be graded. Name a zip file `CS585_Assignment6_username.zip` and submit with web-submit. Collect your .cpp files and label code that you changed with comments containing your username like: `//Modified by deht`

1.1 Learning Objectives

- Use Feature detection and matching to compute a transformation between images
- Use computed transformations to manipulate the images
- Understand the mosaic creation pipeline
- Introduce basic concepts that we will need to study feature tracking

1.2 Written Questions (10 points)

1. Summarize the steps for computing a transformation matrix that describes the geometric relationships between two images
2. Given a transformation matrix T that transforms image I_2 into the same coordinate frame as image I_1, if the size of I_2 is (w, h), write down a formula for each of the four corners of I_2 after it has been warped using T.

1.3 Technical Task (70 points)

For this assignment, we will implement the mosaic creation pipeline to use with two and three images. Skeleton code is given for you to fill in for the case of two images. It is up to you to extend and modify the code in any way necessary in order to create a mosaic with three source images.

Your input images should be no larger than 640 x 480. Submit your input images along with your output images.

1. Given: code to extract and match features from images and compute a transformation mapping between I_2 and I_1.
2. Required: Given the estimated transformation, project the corners of I_2 into the same coordinate frame as I_1.
3. Required: Using the transformed corners, determine the size of the finished mosaic.
4. Required: To save you some debugging time, I am pointing out that the second image may end up with portions above or to the left of the first image, in which case, you will need to translate the entire thing down and to the right so that the entire mosaic will appear in the canvas. Along with the mosaic size, you must compute the correct offset.

5. Required: Given the transformation, and having computed the mosaic size and offset, combine the mosaic images using a "copy and paste" operation similar to your collages, using the warpPerspective function.

   Given your two source images, you should create the mosaic using both orders for the images, to be sure that the mosaic footprint is being created correctly. Example output is provided in `Example_Assignment6_Part1_Result1.png` and `Example_Assignment6_Part1_Result1_backwards.png`.

   Save your output as `Assignment6_Part1_Result1.png` and `Assignment6_Part1_Result1_backwards.png`.

6. Required: Combine the source images into a mosaic by using the data from the source image that is "closest" to each output pixel. My solution uses the OpenCV `distanceTransform` function, but yours does not have to. Save your output as `Assignment6_Part1_Result2.png`.

7. Required: Write a new program to extend your work from part 5 to use three images instead of just two images. (A mostly-empty Visual Studio project is set up under `Assignment6_Part2` with the correct libraries and paths.). Save your output as `Assignment6_Part2.png`.

   You may find that you will have misregistration errors that cause the edges of your source images to not line up precisely in the finished mosaic. This is okay. The output should look reasonable, but it is not expected to look perfect.

8. Optional: The `Lab6` project contains code demonstrating how to use the OpenCV image stitching interface. Create a mosaic with many images using OpenCV's functionality.

1.4 Lecture Preparation (20 pts)

Next week and after Spring Break, we will discuss different ways of working with video to compute motion. There are many ways of thinking about motion in images. We did one of the simplest forms of tracking when we used a ball to draw a shape to place around a face in HW3. Please look at this Wikipedia page to get a flavor of what we will be discussing: `http://en.wikipedia.org/wiki/Video_tracking`.

Here are a few things that we will need to use to be able to talk about feature tracking.

(a) The Sum of Squared Differences (SSD) is a measure of the similarity between two image regions. Let the pixel values from image I in a particular region of interest W_I be represented by x_i and the corresponding pixel values from image J in a region of interest W_J be represented by y_i. Then the Sum of Squared Differences (SSD) is given by

$$SSD = \sum_{i=1}^{|W|} (x_i - y_i)^2$$

If you were using the Sum of Squared Differences to compute the similarity between two images of an object in your hand, how would the results be affected if your rude roommate came in while you were working and turned the light on or off?

(b) The Normalized Correlation Coefficient (or Normalized Cross Correlation) is a measure of similarity that is resistant to changes in lighting (both brightness and contrast). `http://en.wikipedia.org/wiki/Cross-correlation#Normalized_cross-correlation`

Remember the formulas for the mean, $\bar{x}$ and standard deviation σ_x of a quantity are given by

$$\bar{x} = \frac{1}{|W|} \sum_{i=1}^{|W|} x_i$$

$$\sigma_x^2 = \frac{1}{|W|} \sum_{i=1}^{|W|} (x_i - \bar{x})^2$$

Then, the normalized Correlation Coefficient between x and y is given by

$$\frac{1}{|W|} \sum_{i=1}^{|W|} \frac{(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x \sigma_y}$$

Prove that this is invariant to changes in brightness and contrast by showing that the result will not change if you replace x_i with $ax_i + b$. (Hint: Plug and chug.)

(c) The papers we will talk about next week refer to the Newton-Raphson method. Read the first few sections of this article `http://en.wikipedia.org/wiki/Newton's_method`.

In one sentence, state the goal of the Newton-Raphson method.

Given a differentiable function of one variable, $f(x)$ write down a formula for the line that is tangent to f at x_0.

(d) Please brush up on your intuition of eigenvalues and eigenvectors. I recommend these two pages: `http://anothermathgeek.hubpages.com/hub/What-the-Heck-are-Eigenvalues-and-Eigenvecto`
`http://math.stackexchange.com/questions/23312/what-is-the-importance-of-eigenvalues-eigenv`