# EN2550 Assignment 2 on Fitting and Alignment

*Name – R.M.P.A.P. Rajapaksha*

*Index Number - 190484T*

*GitHub Repository –* <https://github.com/rajapaksha-rmpap/EN2550-Assignment-2-190484T.git>

### Question 1 - Using RANSAC method for finding the best fitting circle

The following section describes the *algorithm* for finding the best fit circle using RANSAC and *Figure 1* depicts the result. The main code snippet used to implement the algorithm is given in *Code 1* as a function.

* **step 1 - randomly choose a set of 3 points from the data set.**
* **step 2 - find the *circumscribed circle* of the triangle formed by the selected 3 points.**
* **step 3 - find the Inlier count for the created circle.** 
  + Here, the Inliers are the points that reside within a certain threshold distance from the edge of the estimated circle. As given in the code snippet *Listing 1*, these inliers are distributed according to a *Gaussian distribution* with a *variance* value equal to times the radius of the original circle. Therefore, an expression for the threshold value can be found in terms of the radius of the estimated circle such that of the points of inliers are covered by the threshold limit.

|  |  |
| --- | --- |
|  | *Eq.1* |

* + Furthermore, if we assume that the estimated circle has a radius around length (as the original circle), then a common threshold can be found for all the estimations as .
* **step 4 - if there are inliers more than a minimum threshold count, accept the estimation.**
* To decide whether an estimation is exactly a good fitting model, its inlier count must exceed a certain limit, which is known as the *consensus set size* . Since there are around inliers for the circle, and we expect a good fitting model to cover at least a of those points, the consensus set size can be found as

This above process has to be repeated over a number of times to ensure that there is a sufficient probability that at least one sample would be free of any outlier. If the targeted probability of having at least one successful uncontaminated estimation is and the outlier ratio is , then the number of repetition required to achieve can be found by the equation;

|  |  |
| --- | --- |
|  | *Eq.2* |

Let us assume a reasonably higher value as [i.e. a success rate]. Then, a minimum value for can be found out as follows. []

Finally, the circle with the maximum inlier count is chosen as the **best estimated circle** for the point distribution. Then using the *Least-Squares Circle Fit* method proposed by *Randy Bullock*, the **best-fit circle** will be determined for the inliers of the chosen best estimation. If there are more than one estimation with the same maximum inlier count, then the refitted circle with the smallest mean error will be considered to be the best fit.

### Question 2 – Estimating the homography to fit an image onto a specified tetragonal region

* To find the homography for warping, the four corner point of img2 have to be paired with the four locations specified by the mouse clicks on img1.
* Finding the homography between the two images for those point pairs is done through the **Least-Square Algorithm** discussed in the Lecture. A function programmed to carry out this task is given in *Code 2* below.
* After finding the homography H, img2 is warped by cv::warpPerspective function, and then blended onto img1 using a customized function blendColor. This function only blends the color regions common to both img1 and img2 using cv::addWeighted function and keep the outer regions unaffected.
* Here, I have used this operation *to merge the Oxford University logo onto the front wall of the Oxford University library*, and *to visualize the Olympic flag merged on the water surface of an Olympic swimming pool*. Results and the Code is given in *Figure 2* and *Code 3* below.

### *Question 3 -* Computing the Homography between two images using RANSAC algorithm

The algorithm for finding the best homography between two images using RANSAC is discussed below, and the methodology is much similar to the algorithm given in question 1. The calculated homography and the one given in *H1to5p* text file are compared in *Code 4*, and the two matrices are nearly equal. *Figure 4* illustrates **img1** and **img5** added and stitched together, and the main function used to compute the homography using RANSAC is given in *Code 5*.

Code 1 - Finding the Best Fit Circle using the RANSAC Method



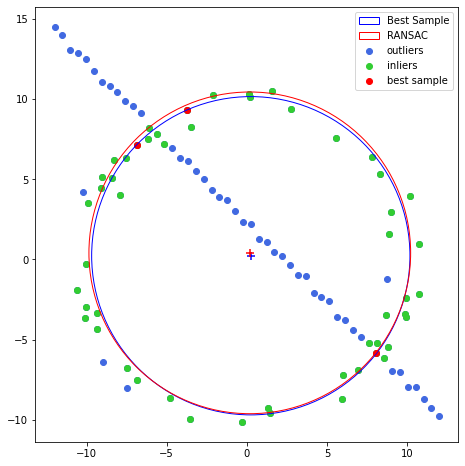


Figure 1 – Result for Question 1

Code 2 - Finding the Homography between two images using the Least-Square Method (for question 2 and 3)



Code 3 - Warping and Blending two images



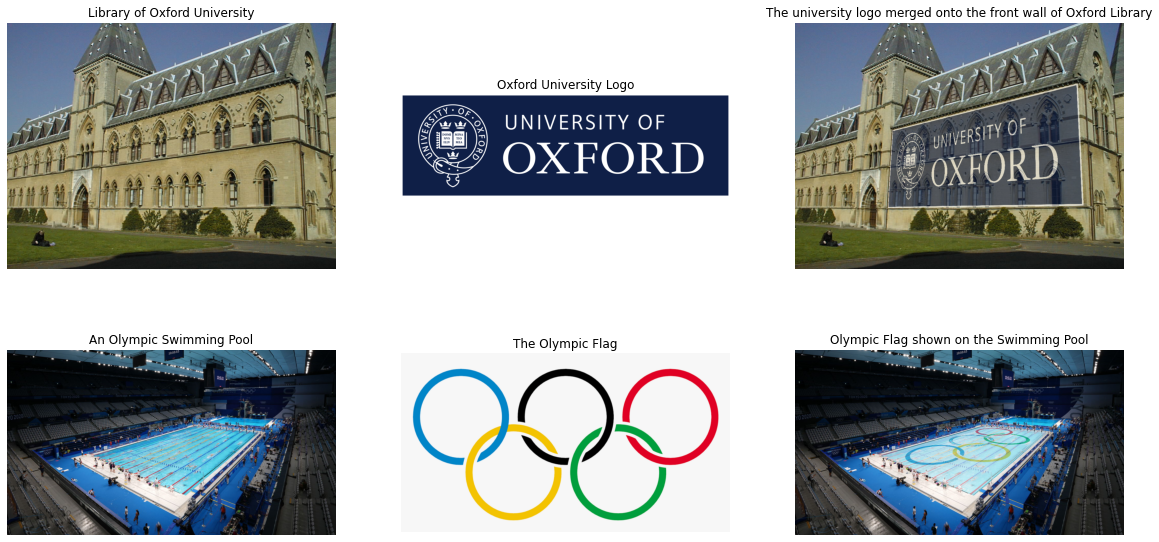


Figure 2 - Result for Question 2

After finding the SIFT features between the two images, choose four matching keypoints pairs randomly . Then find the corresponding homography for those matches using the Least Square algorithm (refer *Code 2*) and count the *Inliers* for that estimated homography. A keypoint pair (say on source/query image to on destination/train image) is considered as an inlier, if ’s projection onto the destination image resides within a threshold distance from . Then, if the inlier count exceeds a given minimum threshold count , accept the estimation. These and values have to be tuned to get a better result depending on the differences between the two images. The *Eq.2* gives the minimum number of samples as 72 for *(*) and [foraveragecase]. Finally, the homography with the maximum inlier count is chosen as the **best estimation**, and the **final homography** is computed as the least-square estimation of its inliers. If there are more than one such best estimations, then take the homography with the least average reprojection error for the inliers.

However, since there is a significant camera angle difference between **img1** and **img5**, their SIFT features mostly contain incorrect matches, thus a homography for these two images cannot be found directly. Therefore, the homography was calculated by combining (multiplying) the homographies from **img4 to img5** and **img1 to img4**.

Code - Results comparison for Qustion 3 (b)

Figure 4 - Result for Question 3 (c)



Code - Computing Homography with RANSAC