# Part A

There were 3 feature detectors I considered: **Harris**, Scale Invariant Feature Transform (**SIFT**), and Speeded Up Robust Features (**SURF**). Harris was tested, but it failed to find enough suitable to features between images. Hence, SURF and SIFT were both my top choices.

In the end, I went with SURF as I was worried about the speed of my code. Due to my limited understanding of the material, I preferred the more familiar syntax of traditional programming language. Hence, I used loops over vectorized operations in MATLAB, making my code quite inefficient. With SURF’s higher speed, some of the inefficiencies can be compensated for.

# Part B

The feature matching criteria was Euclidian distance for OpenSURF, the open source MATLAB implementation of SURF. OpenSURF attempts to pair up the different feature points in the two images to minimize the Euclidian distance.

# Part C

For outlier rejection, I used **RANSAC** to find the best homography matrix. The high-level overview is as follows:

1. Select four feature pairs at random.
2. Compute homography, **H**, using the four feature pairs.
3. Apply **H** to the other features points and calculate the reprojection error (Euclidian distance).

Where **p** is the unwarped homogenous vector, and **p’** is the corresponding pair for **p**.

A pair is considered to match if the error is below a threshold, **epsilon**.

Total up the error for matching pairs.

Repeat for another four features pairs.

1. Store **H** matrices where there is a sufficient number of matches, where the number of matches required is defined by the **consensus** variable.

Divided the total error by the **consensus** to normalize it.

1. Find the **H** with lowest normalized error and use that as the final homography.

The final homograph is then used to stitch the images into a panaroma.

# Part D

To make the panaroma, I started from the left to the right, so images in order 7, 3, 2, 1.

The high-level overview is as follows:

1. Select two unmodified images for computing homography.

This choice was due to the fact that finding feature points between an unmodified image and a stitched image yields unexpected results.

1. Compute the best homography using RANSAC, finding the transform to project the right image to the left image (So first one would be 3 and 7).
2. Find the location of the four corners of the right image after the homography transform.
3. Using the four new corners as the bounds, bilinearly interpolate the pixels from the former stitched image (right image if this is the first set) and the left image.
4. Repeat for the next set of two images

I was only able to stitch together the bottom four images as image 5 to 7 had a significantly different contrast and brightness, making it difficult to match to the bottom four.