Footfall Cam Interview #1

Tasks:

- Identify which frames in the clip have the staff present
- Locate the staff xy coordinates when present in the clip (Bonus)

As this seems to be an image registration problem, I shall define what is image registration briefly. Image registration is the process of matching, aligning and overlaying two or more images of a scene, which are captured from different viewpoints. Image registration has five main stages: Feature Detection and Description; Feature Matching; Outlier Rejection; Derivation of Transformation Function; and Image Reconstruction. With these five main stages in mind, this was the framework of how I constructed my code.

Why I chose AKAZE over the other algorithms?

- ORB and BRISK are not suitable in this case. Although these algorithms have a high detection repeatability, it is not suitable because these algorithms are based on corners detection. The sample images that are given are blurry, also do have not much corners to detect as descriptors. Hence, I went ahead with the blob feature detection methods.
- SIFT and SURF are not suitable in this case as the images given are blurry. Even if SIFT were to work here, keep in mind that SIFT has a high computational cost. KAZE and AKAZE are both based on non-linear diffusion filtering and non-linear scale space. This is why both KAZE and AKAZE are invariant to scale, rotation, limited affine, and have more distinctiveness at varying scales. This is why KAZE and AKAZE algorithms are the most suitable in this case.
- As for why AKAZE over KAZE? AKAZE as the name suggest is accelerated by using FED (Fast Explicit Diffusion) framework to construct its non-linear scale spaces. Descriptor of AKAZE is based on MLDB (Modified Local Difference Binary) algorithm which is also highly efficient.
- Side Note: I did try using the same parameters to run all the algorithms mentioned above. The result that I got was AKAZE had the fastest computation with highest accuracy and repeatability.

Why I chose BFMatcher compared to FLANN?

- There are simply too little descriptor to make a difference. Since there are very few descriptors, BFMatcher seems to give a more accurate result with high repeatability.
    - BFMatcher stands for brute-force, it tries all the possibilities and hence it will find the best matches. However, this uses more resources and is definitely slower compared to FLANN.
    - FLANN stands for "Fast Library for Approximate Nearest Neighbors". This matcher is faster when using large dataset because it finds an approximate nearest neighbors. This means it will find a good matching, but not necessarily the best possible one. For such a small amount of descriptor such as this case, even if I manipulate with FLANN's parameters to increase the precision of matchings, it will not be as good as BFMatcher and it will be at the cost of slowing the algorithm.

- Comparison: In a large dataset, FLANN is much faster than BFMatcher but it only finds an approximate nearest neighbor, which is a good matching but not necessarily the best. Since our data set is not that big, BFMatcher would clearly give a more accurate result with approximately the same amount of given time.
- Sidenote: I ran a test to see the difference. The computational time is about the same but the results were clearly in favor of BFMatcher. Because each image has many other objects around, using FLANN gave too many inaccurate descriptors.

Why RANSAC outlier rejection?

- In this case, there really isn't any significant outlier that will affect our results. But just in case we scale this model, I have place it in the code. For now, I just picked one.