1. Methodology

Python was chosen as the preferred interpreter, given the availability of scientific and machine learning tools for it. OpenCV will be used for Image Processing Tasks.
The project can be divided into the following parts: - Image pre-processing Find the tie-points by using suitable algorithms using feature detection Refine those tie points by removing outliers Re-sampling
The Working of the program is as follows: -

1.1. Pre-Processing and Noise Removal

This is followed by normalization of pixel values between 0-255 or an 8 bit size. The extraction of image data is followed by pre-processing of the image. Image pre-processing is performed using histogram equalization.

By using histogram equalization you get a much flatter histogram response. This actually increases the overall contrast of the image. Increasing the contrast makes the processing of the images fast and efficient as pixels can be easily differentiated after histogram equalization. After this image pre-processing step the next is feature detection. Feature extraction is done to obtain tie points. A tie point is a point within the image that has the following characteristics:- It has a clear, preferably mathematically well-founded, definition, It has a well-defined position in image space, The local image structure around the interest point is rich in terms of local information contents. There are basically four aspects of our Methodology, viz., Feature Detection, Descriptor Extraction, Descriptor matching and Outlier Rejection.

1.2. Feature Detection: The following algorithms have been used in our program:

1.2.1. **Harris Corner Detection for feature extraction and Cross-Correlation for Point Detection**: In this method, the image was split into 5x5 equal sized tiles and the most prominent corner in each tile was classified as a feature. The corresponding point for each feature on the Reference Image was detected using Cross Correlation with a template image of 50 pixel padding around the feature detected.

1.2.1.1. Outcome: The algorithm is translation and rotation invariant, but is not scale invariant, and is highly susceptible to illumination differences. Also, it requires high processing powers, which are beyond the capacities of normal PC computers as of today.
Time taken to process entire image: 13 minutes
Time taken to process image resized by half: 6 minutes

1.2.2. **FAST (Features from Accelerated Segment Test)** for Feature extraction Correlation with Normed distance: FAST algorithm is based on a paper by Edward Rosten and Tom Drummond - “Machine learning for high-speed corner detection. For detection, Normed distance was used instead of Euclidean distance and Image was preprocessed by histogram equalization before applying FAST to minimize differences due to illumination variation.

1,2.2.1. Outcome: The algorithm is translation and rotation invariant, and works for images across different illuminations, but is NOT scale invariant. It is reasonably fast, and does not require very high processing powers to compute.
Time taken to process entire image: 100-110 sec
Time taken to process image resized by half: 50 sec

1.2.3. **ORB (Oriented-fast and Brisk Rotation):** This algorithm uses FAST algorithm for Feature Extraction on both images, and uses BRISK (Binary Robust Invariant Scalable Key points) to find relation between the found key points. BRISK is a much optimized algorithm. Image was preprocessed by histogram equalization before applying ORB to minimize differences due to Illumination variation.

1.2.3.1. Outcome: The algorithm is translation and rotation invariant, and works for images across different illuminations. The algorithm is partially scale invariant, and could handle variations in scale up to a factor of 2. It is very fast, and does not require very high processing powers to compute. This algorithm was found to be the best algorithm to calculate transformation if scale variance is within sufficient bounds.
Time taken to process entire image: 51 sec
Time taken to process image resized by half: 17 sec

1.2.4. **SIFT (Scale Invariant Feature Transform)** for Feature Extraction and **RANSAC (RANdom SAmpling Consensus)**: This algorithm uses SIFT algorithm for Feature Extraction on BOTH IMAGES, and uses RANSAC to find relation between the found key points. RANSAC is a training-based machine learning model. The image was pre-processed by histogram equalization before applying ORB to minimize the differences due to illumination variation.

1.2.4.1. Outcome: The algorithm is translation and rotation invariant, and works for images across different illuminations. The algorithm is fully scale invariant, which makes it different from all other algorithms. Calculating SIFT features directly is a very computationally expensive task, but it can be fastened by using SURF (Speeded-Up Robust Features) that reduces the computations to a good extent. This algorithm

was found to work robustly even for satellite imagery. Hence, this algorithm was chosen for further work.
Time taken to process entire image (using SURF): 5 minute
Time taken to process image resized by half: 50 sec

1.2.5. **BRIEF (Binary Robust Independent Elementary Features):** BRIEF is a faster method feature descriptor calculation and matching. It also provides high recognition rate unless there is large in-plane rotation

1.2.6. **BRISK (Binary Robust Invariant Scalable Keypoints):** BRISK relies on an easily configurable circular sampling pattern from which it computes brightness comparisons to form a binary descriptor string. The unique properties of BRISK can be useful for a wide spectrum of applications, in particular for tasks with hard real-time constraints or limited computation power: BRISK finally offers the quality of high-end features in such time-demanding applications. BRISK achieves comparable quality of matching at much less computation time.

1.2.7. **AGAST (Adaptive and Generic Corner Detection Based on the Accelerated Segment Test)**: This algorithm finds the optimal decision tree in an extended configuration space which can be combined to yield an adaptive and generic accelerated segment test. The resulting method provides high performance for arbitrary environments and so unlike FAST, the corner detector does not have to be trained for a specific scene, but it dynamically adapts to the environment while processing an image.

1.2.8. **A-KAZE (Accelerated KAZE)**: KAZE is an edge-preserving nonlinear filtering strategy to locate image features. The filtering is based on the image diffusion equation. A fast explicit diffusion (FED) technique is used to solve the diffusion equation efficiently, contributing to an accelerated variation of the KAZE algorithm, called AKAZE. Keypoints are located by finding the extrema of the second-order derivatives of the image over the non-linear multi-scale pyramid built from the principle of image diffusion. A-KAZE deploys a technique similar to SURF to estimate the direction of a patch. A modified local difference binary (LDB) representation of the rotation-compensated patch is then extracted as its binary descriptor.

1.3. Descriptor Extraction After detecting key points we go on to compute a descriptor for every one of them. A local descriptor is a compact representation of a point's local neighborhood. In contrast to global descriptors, describing a complete object or point cloud, local descriptors try to resemble shape and appearance only in a local neighborhood around a point and thus are very suitable for representing it in terms of matching. OpenCV comes with several implementations for feature detection. The algorithms used by us include SIFT, BRISK, FREAK, AKAZE.

1.3.1. **FREAK (Fast REtinA Keypoint):** FREAK suggests using the retinal sampling grid which is also circular with the difference of having higher density of points near the center. The density of points drops exponentially. Each sampling point is smoothed with a gaussian kernel where the radius of the circle illustrates the size of the standard deviation of the kernel.

1.4. Correspondence Estimation (Descriptor Matcher) The next task is to find correspondences between the key points found in both images. Therefore the extracted features are placed in a structure that can be searched efficiently. Usually it is sufficient to look up all local feature-descriptors and match each one of them to its corresponding counterpart from the other image. However due to the fact that two images from a similar scene don't necessarily have the same number of feature-descriptors as one cloud can have more data than the other, we need to run a separate correspondence rejection process. The following algorithms have been taken into use for our program: -

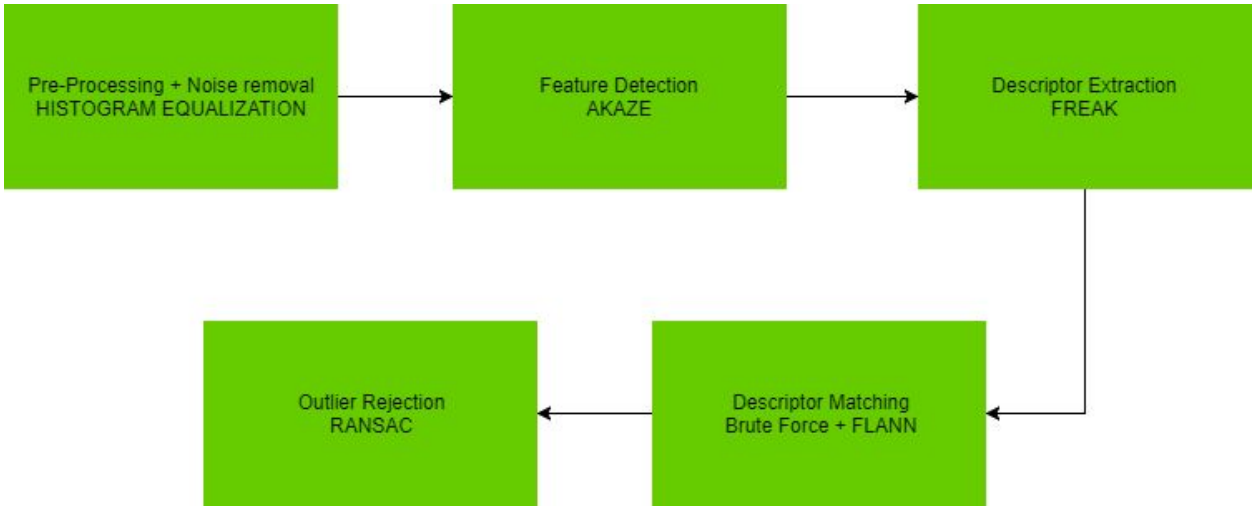
1.4.1. **BF (Brute Force)**: It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.

1.4.2. **FLANN (Fast Library for Approximate Nearest Neighbors)**: FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It contains a collection of algorithms we found to work best for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset.

1.5. Outliers Rejection The next step is outliers rejection or rejecting those points which exceed the threshold of the **RMSE (Root Mean Square Error)** value. One of the most common approaches to perform correspondence rejection is to use RANSAC (Random Sample Consensus).

1.5.1. **RAndom SAmple Consensus (RANSAC):** The RANSAC algorithm is a learning technique to estimate parameters of a model by random sampling of observed data. Given a dataset whose data elements contain both inliers and outliers, RANSAC uses the voting scheme to find the optimal fitting result. Data elements in the dataset are used to vote for one or multiple models. The implementation of this voting scheme is based on two assumptions: that the noisy features will not vote consistently for any single model (few outliers) and there are enough features to agree on a good model.

2. Work Flow



3. CONCLUSION

Though SIFT is by far the oldest of the algorithms tested in this study it still proved to have superior performance to the newer algorithms. The claim from the creators of KAZE, that it could surpass SIFT are not supported by the test results from this study and in fact AKAZE outperformed KAZE in most cases leaving few reasons to choose KAZE over AKAZE. In general, SIFT had among the highest score in each tests and if one was to use one algorithm for general detection purposes SIFT would be the clear choice. There is however the issue with the copyright. If this is an issue the second best performing algorithm is AKAZE. While having lower detection rates on the general detection test than SIFT it still had high detection rates on most tests and managed to score 100% on the test of illumination invariance. AKAZE is also faster than SIFT which can be a significant advantage on devices with low computational power. If speed is of the essence, for example if objects need to be tracked or for use in a moving car ORB might the correct choice. While having lower accuracy than the other algorithms it can still detect the object in simpler cases and is extremely fast. The lower detection rate can also be mitigated by prior knowledge of a scene. If you for example know that the traffic signs will appear in the top half of the left part of the image, detection will be significantly easier. This is also true for tracking since you have prior knowledge of the scene and the location of the sought after objects. AKAZE is both faster and had a higher total score.

7. Comparison of Applied Algorithms

	Translation Invariant	Rotation Invariant	Scale Invariant	Illumination Invariant	Process Time* Full Image	Process Time* Half Image
Harris Corner +Correlation	Yes	Partially	No	No	13min	6min
FAST +Correlation*	Yes	Yes	No	Yes	110sec	50sec
ORB + FLANN	Yes	Yes	Partially	Yes	51sec	17sec
SURF + RANSAC	Yes	Yes	Yes	Yes	5min	50sec
AGAST + FREAK	Yes	Yes	Yes	Yes	7min	2min
BRISK	Yes	Yes	Yes	Yes	6.5min	2min
AKAZE	Yes	Yes	Yes	Yes	3min	40sec

Reference Research paper :
http://www.diva-portal.org/smash/get/diva2:927480/FULLTEXT01.pdf