**ORB:**

Introduction:

Oriented FAST and Rotated BRIEF (ORB) is an open-source feature detection algorithm developed by OpenCV. ORB uses FAST for keypoints detection and BRIEF for keypoint description. Working: Using the FAST algorithm for keypoint detection, the pixels' brightness is compared to determine if an edge is present. In addition, ORB uses a multiscale pyramid consisting of images with the resolution scaled-down of the same image. The FAST feature detection is carried out on each of them. After the keypoints are detected, OBR will assign an orientation to each keypoint based on the intensity change around the keypoint. Next, all the keypoints are detected by FAST and fed into BRIEF, which converts it into a binary feature vector. BRIEF uses a Gaussian Filter to smooth out the image. A random keypoint is selected, and the brightness of multiple points in the vicinity of the keypoint is compared with the brightness of the keypoint in order to form a binary vector for each keypoint. ORB uses a rotation-aware BRIEF by using the orientation recorded from FAST. By using the Gaussian Filter on images with different resolutions, stronger features can be determined.

**Harris Corner Detection:**

Introduction: Harris Corner Detection is used to detect edges and corners in an image. It is one of the oldest detectors, introduced back in 1988.

Working: The Harris Corner Detector uses a window of a pre-defined size in order to detect corners and edges in an image. When an edge is present in the window, there is a sudden change in the brightness only in the direction perpendicular to the edge. On the other hand, if a corner is present, there is a junction of two edges in the window, and a significant change in brightness can be measured in all directions.

Introduction: Features from Accelerated Segment Test (FAST) is a computationally efficient corner detection algorithm. Due to the high speed and low computation required, FAST is used in Real Time applications.

Working: A pixel is selected, and the intensity of the pixel is compared with the intensity of the pixels(about 16 pixels) around it, and set a threshold value. In order to be determined as a corner, the threshold criterion is mentioned below.

Threshold Criterion:​

1. All the 16 pixels are brighter than the intensity of the selected point + threshold.​
2. All the 16 pixels are darker than the intensity of the selected point – threshold.​

In order to make the algorithm faster, four pixels opposite each other are checked first, and if and only if the intensities of at least three of these points satisfy the threshold criterion after which, the algorithm considers it as a corner, or else it will simply move to the next pixel.

**BRISK:**

Introduction: Binary Robust Invariant Scalable keypoints is a scale and rotation invariant feature point detection and description algorithm. It was developed in 2011 as a free alternative to SIFT.

3D Structure from Motion

Structure from motion (SfM) is the process of estimating the 3-D structure of a scene from a set of 2-D images. SfM is used in many applications, such as 3-D scanning, augmented reality, and visual simultaneous localization and mapping (vSLAM).

For SfM, open source library OpenSfM is used. It serves as a processing pipeline for reconstructing camera poses and 3D scenes from multiple images. It consists of basic modules for Structure from Motion (feature detection/matching, minimal solvers) focusing on building a robust and scalable reconstruction pipeline. Firstly, we improve the transitions from one photo to another by using images. We generate a triangular mesh to approximate the scene geometry and map the images onto it. This way, we can render the approximate 3D geometry while we move from one camera to another, giving a better perception of the motion.

Fig shows the reconstruction of images with the camera poses the images of the sea bed with the light gray images representing the seabed while the dark gray images representing the objects on the seabed.

Introduction:

Methodology:

We read multiple papers and articles describing the working of the different feature detection algorithms and selected the best-performing algorithms to be compared. After this, using python and MATLAB code, we chose different feature detection algorithms and obtained the output for each by finding the features from the images in the same dataset. After seeing the features, the same stitching algorithm was used for all the images to compare the quality of the features detected. During feature detection, we also measured the different parameters necessary for comparing them to make a concrete conclusion. The comparison is shown below in Table 1. Additionally, we wanted to map the environment in 3D space, and thus we started looking at different SLAM algorithms. We are using the SfM open-source library to map the surface in 3D space and perform SLAM at the same time.

Dataset:

The dataset was collected using a vehicle consisting of a heavy-duty frame on which three low-light level, black and white video cameras with running lights, and a 35mm color strobe camera. The dataset used is called Skerki-D and was part of a larger Skerki project. The Vehicle also consisted of additional sensors and actuators to maneuver the Vehicle underwater. The Vehicle was at a distance of 15m from the bottom. The Vehicle dived down to 3 meters above the surface to perform precision mapping of the shipwreck. The dataset consists of images of an ancient 80 – 60 B.C. Roman Trading shipwreck. The actual area scanned by the Vehicle has a length of 35 meters and a width of 10 meters.

Results:

The above tables show the comparison between the different feature detection algorithms. To get data over a broader set of data, all the algorithms were run on batches of images with varying densities of objects. Table 1 shows the comparison of the algorithms' performance when running on images with no objects present. It can be seen that ORB has the most features detected, but SIFT has the most number of Feature matches even though the number of features detected is low. BRISK took the least amount of time to detect the features but also had the most outliers and no matching features. Harris Corner Detector took the most amount of time to detect the features.

In Table 2, different algorithms' performance is compared when running on images with a few objects present. Similar to the first case, ORB detected the most features in this case, and SIFT could match more features. BRISK was again the fastest performer to detect the features.

Table 3 shows the comparison of the performance of the different feature detection algorithms when run on images with a high density of objects is shown. Here we can see that SIFT was able to detect a more significant number of features and also was able to match the most number of features, but SIFT also took the most amount of time in doing so. BRISK was again the fastest to detect the features and had the highest number of outliers.

Table 4 shows the average of all three cases to better understand the performances of the different feature detection algorithms. Based on these results, SIFT was able to match the most significant number of features. BRISK was the fastest among the algorithms to detect and match a medium number of features. FAST consistently was placed in the middle of the pack with good time complexity and a decent number of features matched. Harris corner detector, the oldest and most primitive type of feature detector, was the slowest and the worst in detecting and matching features. ORB is a feature detector with good time complexity and a good number of features detected and matched.