# A Benchmark Stereo-Visual Odometry variant for MAV based Autonomous Navigation

**Jerrin Bright**[1]

[1]Vellore Institute of Technology, School of Mechanical Engineering,
Chennai, Tamil Nadu, India
Jerrin.bright2018@vitstudent.ac.in

**Abstract:** ***This work deals with determining the best algorithm with respect to the environment (indoor or outdoor) for determining the visual odometry which results in efficient pose estimation for visual based autonomous navigation. Thus, various approaches and combinations of visual odometry has been studied with various combinations and examined closely in terms of accuracy and computational cost. We thus in this research examine all the commonly used feature extraction methods including ORB, SIFT, SURF, AKAZE, BRISK, Shi-Tomasi and Matching methods including Brute-Force Matcher and Fast Library for Approximate Nearest Neighbor. Then outlier rejection techniques including Random Sample Consensus (RANSAC), M-estimator Sample Consensus (MSAC), and Progressive Sample Consensus (PROSAC) are compared with and resulting stats are logged. Followingly, estimation techniques including various versions of perspective-n-points (PnP) including EPnP and MLPnP are studied for efficient retrieval of the rotational and translational vectors. Additionally, various pre-processing methods including Bilateral Filtering, Gaussian Blur, Rotation, Warping are experimented. The trajectory of the pose estimated via various combinations as mentioned above are studied and evaluated using KITTI and EUROC benchmarked datasets and conclusions stating the best blend and environment conditions for efficient stereo visual odometry are conferred. The source code is available at https://github.com/jerriebright/visual-odometry.***

**Keywords:** SLAM, Visual Odometry, Computer Vision, Pose Estimation

## 1. INTRODUCTION

Visual Odometry (VO) is a very vital part of autonomous navigation used with autonomous ground vehicles, aerial vehicles and underwater or surface vehicles. Visual odometry is in layman terms, determining the position information for accurate navigation of the robot using camera image sequences. Usage of encoders has slippage issues which in turn will reduce the accuracy of estimation. Thus, VO provides a computationally cheaper and accurate odometry results when compared to GPS, wheel odometry, sonar localization or INS.

There are 2 approaches for VO- appearance based and feature based. Appearance based VO make use of the pixel intensities to directly estimate the dense correspondences; feature-based VO uses the image intensities to determine the keypoints and matching of the keypoints with corresponding image frame to estimate the ego-location of the robot. We will be evaluating each and every technique along with its classifications in detail and suggest the best fit for Stereo Visual Odometry (SVO). Our main contributions include:

- The best feature extractors and descriptors combination to be used.
- The most efficient feature matcher and outlier removal technique to be used.
- The most efficient method to estimate the rotational and translational vectors of the camera.

The remainder of this paper is structured as follows: In Sec. 2 we discuss various other research works orienting our research work. In Sec. 3 we recall all the subtypes involved in the architecture of VO. In Sec. 4 we do experimental analysis comparing all the types and establish a perfect fit for SVO. Conclusions will be finally drawn in Sec. 5.

## 2. RELATED WORKS

Previously, we have seen a lot of research works relating to phases of visual odometry that's detailed comparison on feature extractors or feature descriptors, but there hasn't been much research benchmarks stating the best combination of stereo visual odometry which could result in accurate pose estimation. In particular, Tareen et al [1] did a very detailed analysis on SIFT, SURF, KAZE, AKAZE, ORB and BRISK in terms of efficiency and accurate algorithms using Oxford, Matlab, VLFeat and OpenCv. Monika et al [2] performed a comparative study between SIFT, SURF and ORB descriptors. Chien et al [3] used SIFT, SURF, ORB and AKAZE for mono visual odometry, analyzed and suggested the best fit for mono VO. Ebrahim et al [4] compared SIFT, SURF, BRIEF and ORB for Distorted Images explicitly. Thus, this research concentrates on comparing various visual odometry

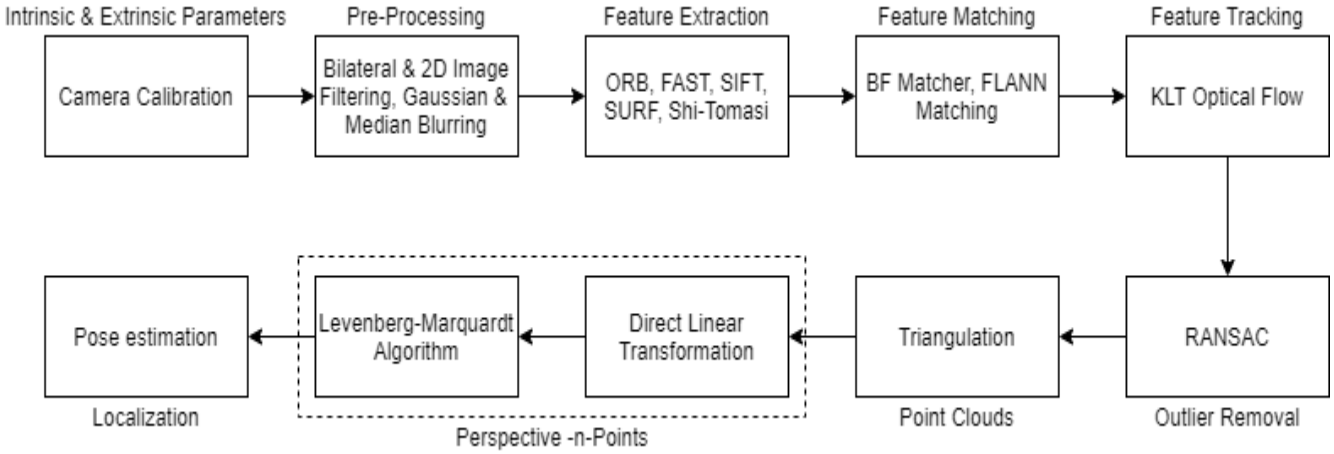



**Figure 1.** Architecture of the Visual Odometry

combinations from scratch and benchmark the most accurate variants considering the influence of indoor and outdoor scenarios as well.

## 3. METHODOLOGY

The basic workflow of stereo visual odometry includes Feature Extractor, Feature Matching, Feature Tracking, Outlier removal, Triangulation and PnP for efficient estimation of the pose using Direct Linear Transformation and Levenberg Marquardt Algorithm in its backend. The architecture can be visualized in fig. 1. In this section we will explain all the phases involved in the visual odometry algorithm in detail.

### *Camera Calibration*

Calibration reduces the radial and tangential distortion. Calibration helps to determine the intrinsic and extrinsic parameters along with the distortion coefficients. Intrinsic parameters include focal length and optical centers and extrinsic correspond to the rotational and translational vectors. Implemented using OpenCV library and OpenCV's sample dataset of chess board. First the image points are determined in the chess board as in the first image fig. 1 (a). Then, calibration can be done using the **calibrateCamera()** library. The second image that's fig. 1 (b) shows the undistorted image after calibration is done.

### *Image Enhancement*

**Smoothening/ blurring-** Helps in removing noises in the image leaving most of the image pixels intact. Some of the smoothening methods experimented are image filtering, gaussian blurring, bilateral filtering and median blurring.

**Rotation and Warping** were also experimented in the pre-processing steps to observe the results with such types of pre-processing steps. It for sure will increase computational cost of the system. Its effect on the accuracy of the feature extraction was thus tested precisely.

### *Feature Extraction*

There are a lot of feature extraction methods including FAST, SIFT, ORB, SURF, Shi-Tomasi, BRISK, KAZE, AKAZE, etc. Basic working of all these extractors and comparison with outdoor and indoor environment with different contrasts are shown in this section.

**FAST-** FAST stands for Features from Accelerated Segment Test [5]. It is one of the fastest feature extraction technique. It is best for live real-time application point of view with efficient computation. The working of FAST is as follows: Corner detection by drawing the bresenham circle around the pixel p and labelling each of the circle from 1-16. Then intensity of N random pixels is compared.

$$S_p = \begin{cases} Ip \rightarrow x \leq Ip - T, \ darker \\ Ip - T \leq Ip \rightarrow x < Ip + T, \ similar \\ Ip + T \leq Ip \rightarrow x, \ brighter \end{cases} \quad (1)$$

Multiple interest points are cast off using the non-maximum suppression which is based on the difference in distance between subsequent keypoints.

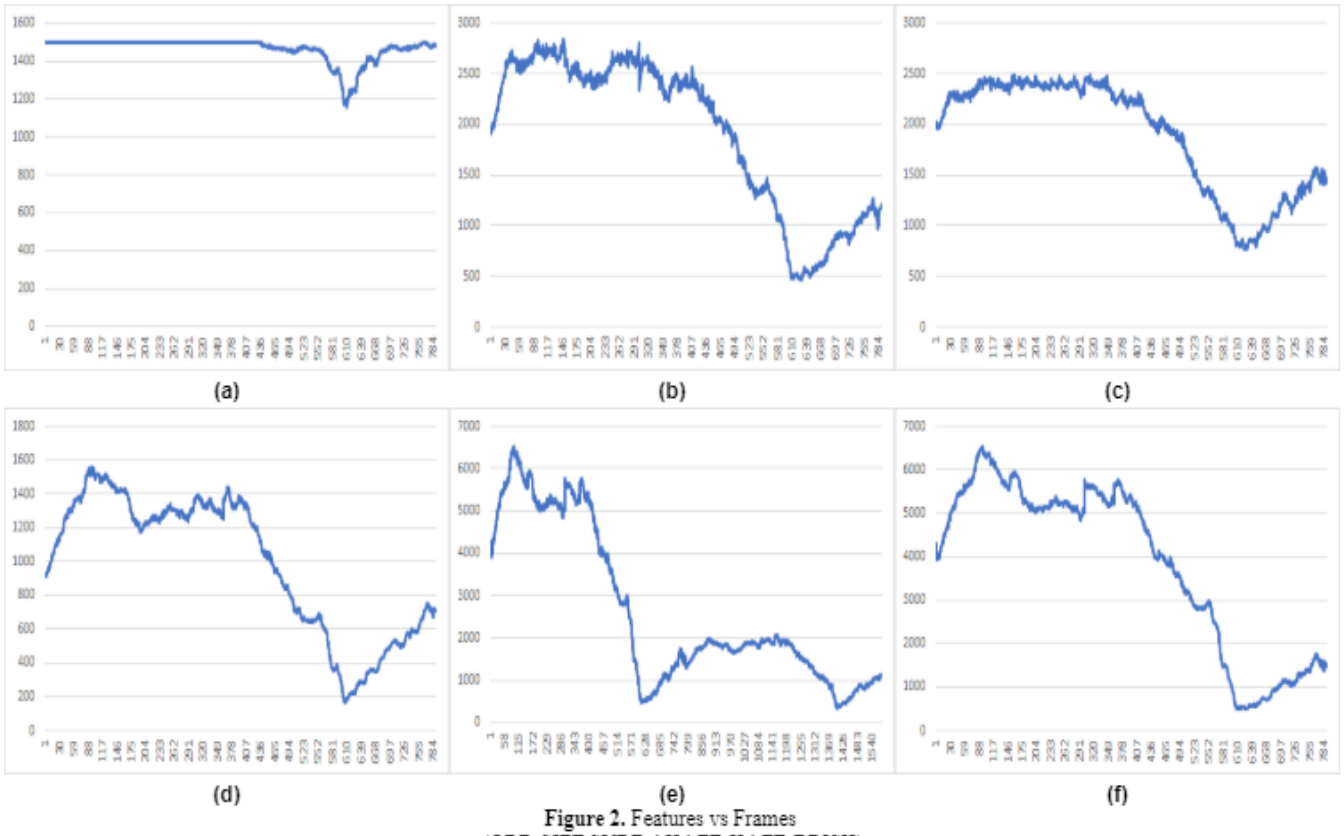



**Figure 2.** Features vs Frames
(ORB-SIFT-SURF-AKAZE-KAZE-BRISK)

**SIFT**- SIFT stands for Scale-Invariance Feature Transform [6]. Its working is as follows: First interest points are scaled and localized followed by blurring using the gaussian blur operation.

$$L = G\,(x, y, \sigma) \times (I\,(x, y)) \tag{2}$$

Comparing of the interest points with neighboring octaves is done to determine keypoints based on its extrema values obtained. Rejection of edges and flat region happens next based on intensity measures.

$$Intensity < 0.03 \tag{3}$$

$$HM = \begin{pmatrix} Dxx & Dxy \\ Dxy & Dyy \end{pmatrix}, \frac{Tr\,(HM)}{Det(HM)} = \frac{(R+1)^2}{R} \tag{4}$$

Equation (3) represents rejection criteria for flat regions and (4) for edges. Then the orientation of the keypoints obtained is based on high probable distribution of the orientation of each and every keypoint. Then the descriptors based on orientation and scale of the keypoint is assigned.

**SURF**- SURF stands for Speeded Up Robust Features [7]. It is a robust and fast technique preferred for its fast computation of operators using box filters, thus enabling real-time applications such as tracking and object recognition. Consists of 2 steps, Feature Extraction and Feature Description. Feature extraction uses integral image, which is a way of calculating the average intensity of pixels in a selected box.

$$Image\ Integral = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i, j) \tag{5}$$

Surf uses the Hessian matrix because of its good performance in computation time and accuracy. Descriptors are assigned by first assigning a fixed orientation to each interest point, and then extract feature descriptor.

**ORB**- ORB stands for Oriented FAST and Rotated Brief [9]. In ORB, oriented FAST algorithm is used to calculate and find the keypoints from the image and rotated BRIEF descriptors are used. Coming to the feature detection part, image pyramidal representation of frames is done for scale invariance and then features from each pyramidal layer is extracted. Firstly, to estimate the orientation, centroid will be found which will be estimated from the moment equations.

$$C = \left(\frac{m10}{m00}, \frac{m01}{m00}\right) \tag{6}$$

**Table 1:** Various Feature Extractors

| | ORB | SIFT | SURF | KAZE | AKAZE | BRISK |
|---|---|---|---|---|---|---|
| **Origin** | 2011 | 1999 | 2006 | 2012 | 2013 | 2011 |
| **Scale Invariance** | YES | YES | YES | YES | YES | YES |
| **Rotation Invariance** | YES | YES | YES | YES | YES | YES |
| **Keypoint Type** | FAST | DoG | Hessian | Hessian | Hessian | - |
| **Descriptor Type** | Binary | Integer | Real | Real | MLDB | Binary |
| **Descriptor Length** | 32 | 128 | 64 | 128 | - | 64 |

Then, orientation patch can be obtained.

$$Orientation = a \times \tan 2\ (m01, m10) \qquad (7)$$

Rotated Brief descriptors are used once extraction is done. The orientation patch obtained is used to compute the BRIEF descriptor operation:

$$g\ (p, \theta)\ =\ f(p)\ |\ (x, y)\ \in\ S \qquad (8)$$

**SHI-TOMASI-** Harris corner detector takes the differential of the corner score into account with reference to direction directly. A corner is an important feature in an image, as they are interest points which are invariant to translation, rotation, and illumination. The idea is to consider a small window around each pixel p in an image. Then we measure gradient shift of the central pixel and the pixels around in all 8 directions (using SSD values of pixel value). If shift is above a threshold value, a new feature descriptor is mapped. Small modifications in the Harris Corner detection resulted in the Shi-Tomasi corner detection [10] using the good features to track algorithm.

**KAZE-** The working of KAZE [13] is as follows: Nonlinear scale-space extrema to detect the features accurately; Nonlinear diffusion filtering with Additive Operator Splitting (AOS) is used instead of gaussian blurring to keep the object boundaries intact.

$$c(x, y, t)\ =\ g(|\ \nabla Io(x, y, t)\ |) \qquad (9)$$

Orientation of the feature points are done very similar to SIFT, but instead of aligning in a histogram, points are used in the vector space. Since the descriptors has to operate at nonlinear scale-space, a modified version of SURF descriptor was used here in KAZE.

**AKAZE-** AKAZE [12] was built over the KAZE algorithm. It uses an accelerated version of nonlinear scale-space called as Fast Explicit Diffusion (FED). This change was made considering how computationally expensive the AOS process of KAZE was. Also, AKAZE uses a form of local difference binary (LDB) descriptors to increase the computation of the extraction process further.

**BRISK-** BRISK stands for Binary Robust Invariant Scalable Keypoints [11]. Similar to ORB, BRISK uses pyramidal image representation, wherein stable points are extracted using the adaptive corner detection operator. Unlike other algorithms, BRISK uses binary bitstring [14]. The feature descriptor is generated from this equation:

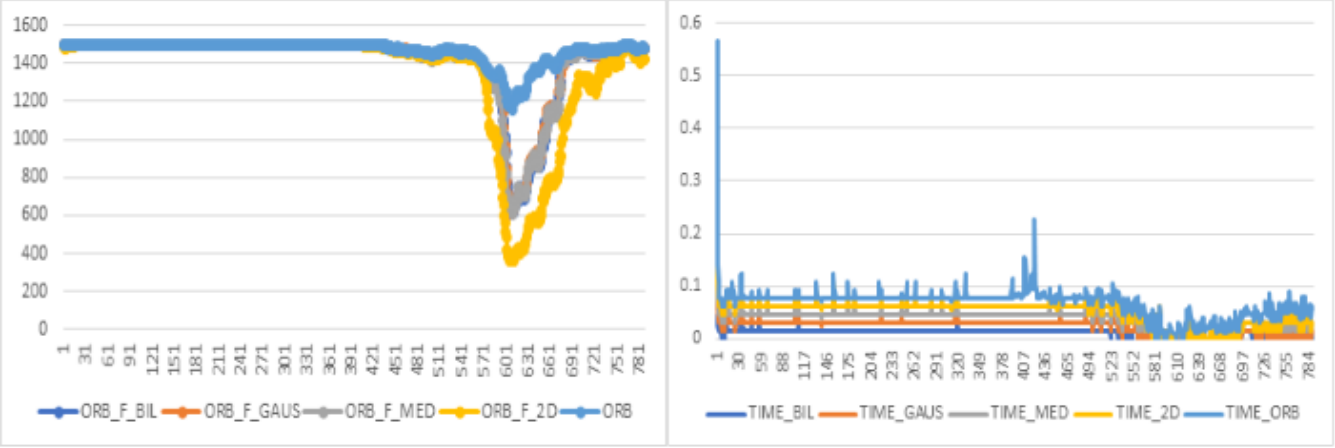
$$B = \begin{cases} 1 & I(Pj, \sigma j) > I(Pi, \sigma i) \\ 0 & otherwise \end{cases} (Pi, Pj)\ \in\ S \qquad (10)$$

***Feature Description***

**BRIEF-** BRIEF stands for Binary Robust Independent Elementary Features [8]. It uses binary strings as feature point descriptor. BRIEF identifies the patches around the keypoint and converts it into a binary vector, thereby representing an object. Thus, each keypoint will be described with the help of the binary 1's and 0's. One major consideration about the BRIEF is that it is very noise sensitive as it deals closely with pixel-level images. Thus, smoothening is very important to reduce the noise from the image. Smoothening is done using Gaussian Kernel in BRIEF. Once smoothening is done, the next step is the feature descriptor. Now the problem will come when we will have to calculate the n (x, y), that's the x, y pairs. This (x, y) pairs are called random pair inside the patch. N is the length of the binary vector and τ is the binary test response of the vector. Finding the random pair can be easily done if the length of the binary vector is decided as we can pick from the patch easily then. There are 5 different methods used to calculate the length. They are uniform (GI), gaussian (GII), gaussian (GIII), coarse polar grid (GIV), coarse polar grid (GV). Also, BRIEF relies on less intensity difference between the image patches.

***Feature Matching***

**BF Matcher-** It takes the descriptor of each and every feature of first set and matches with all the other features from the second set using distance calculation. It is time




(a) (b)

**Figure 3: Preprocessing**

consuming and matches all features. There are a lot of distance measurement methods including NORM_L2/ NORM_L1, NORM_HAMMING, etc. Also, Lowes's test is done after matching in general to remove outliers.

**FLANN**- Stands for Fast Library for Approximate Nearest Neighbor. It has a collection of optimized algorithms to for searching the nearest neighbors in big datasets. It is faster and more accurate than BF Matcher. It has two dictionaries index and search parameters.

***Feature Tracking***

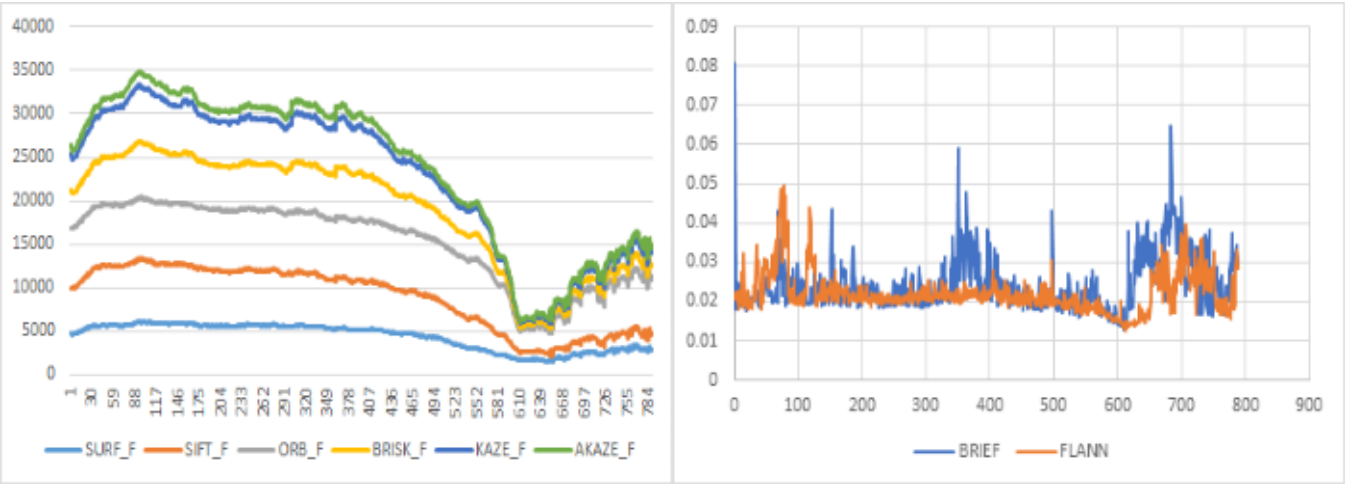
***Outlier Removal***

RANSAC stands for Random Sample Consensus. Deals with removing outliers from inliers contained in a data. No real-world sensor readings are perfect. Thus, we use RANSAC which is a simple trial and error method with groups the inliers and outliers separately. Thus, it helps us to throw away the outliers and work with inliers alone which will save our computation and time. It involves a 4-step processing, and they are sampling, scoring, computing and repeating.

Let's assume there are 2D points (Fig. 14 (a)) in which a line has to be fitted. Now we will sample and randomly take one line marked in fig. 14 (b) as our inlier. Now we will compute the number of supporting inliers satisfying the line we drew. This will be done by parallelly extending imaginary lines on both the side of the line assumed to be the inlier with a uniform distance of **δ.** Now each and every point lying inside these imaginary lines constitute to the scoring of the inliers. Now for the fig. 14 (b) there are 4 inliers.

Now we will repeat the steps mentioned above repeatedly and the score will be overwritten with the best score after every iteration. In the Fig. 14 (c), that's after some iterations, we got a line which has 12 points satisfying that line model which is the best score to be obtained. Thus, it will be best fit for the line. This way we can eliminate all the outliers easily.

***Rotation and Translation***




(a) (b)

**Figure 4:** Feature Extractors and Matching techniques

$$S \times p_c = K \times [R \mid T] \times p_w \quad (1)$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2)$$

SIFT and SURF performs similarly in outdoor environment and is computationally expensive focusing on sparse extraction. Controlling the number of features to be extracted of SIFT and SURF is not possible thereby making it difficult to manage tradeoff in extraction process.

**Table 2.** Comparing various Pose Estimation Results

| | ORB | SIFT | SURF | KAZE | AKAZE | BRISK |
|---|---|---|---|---|---|---|
| **Number of Features** | 9920257 | 7880352 | 9121625 | 7647275 | 6038516 | 11324156 |
| **Total Time taken** | 387.1427 | 1035.8436 | 724.0894 | 4087.2750 | 922.7840 | 675.3332 |
| **Total % Error** | 346.2213 | 412.6922 | 241.2476 | 164.7603 | 400.5994 | 678.8387 |

Some of the assumptions taken will estimating the pose of the robot using stereo vision are:

- Camera parameters are known.
- Image co-ordinate is known.

For mono, we will first estimate the estimation matrix derived from the Fundamental matrix. Then we will use recover pose to find the rotation and translational vectors. In stereo, we will use a variant of PnP (that's Perceptive-n-Points). Some of PnP variants include EPnP, MLPnP and the frame-PnP. Basic framework of PnP is as follows:

- Direct Linear Transform (DLT) finds approximate value of Rv and Tv using the projection matrix.
- Levenberg-Marquardt Algorithm is a trial-and-error optimization step used to reduce the reprojection error thereby optimizing the estimated Rv and Tv values.

## 4. Discussion/ Analysis

### Pre-processing

In fig. 6 represents time taken for ORB extractors with and without smoothening (a) using Image Filtering, (b) using Gaussian Filtering, (c) using median blurring and (d) using bilateral filtering technique. In fig. 7, we can see the number of features extracted in each and every frame after applying different smoothening and edge sharpening filters. It is observed that there is a significant decrease in the number of features extracted when filters are used over the frames. Fig. 8 depicts that smoothening reduces the total time for processing and finding the number of features. This is an effect of reducing the number of features while smoothening occurs as visualized in fig. 7.

### Feature Extraction

Fig. 5 visualizes all the number of features with respect to the frames of various feature extractors used throughout this section. Some of the extractors included in this figure are ORB, SURF, SIFT and FAST.

Thus, ORB focuses on close range and does dense extraction thereby leaving out far off features unnoticed.

Shi-Tomasi has the least number of features comparatively in outdoor environment. But has the sparsest features extracted. Clearly defines the boundary of the environment compared to other techniques. Rotation based preprocessing isn't required. Bilateral filtering is most preferred when compared with the computation cost and the number of features extracted.

### Feature Matching

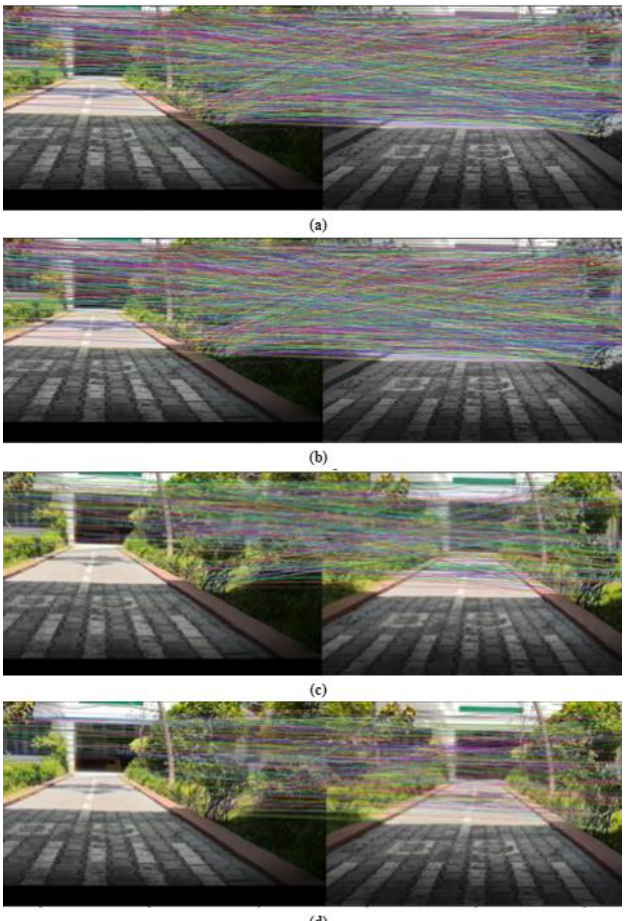



**Figure 5:** Feature Matching


**Figure 6.** Trajectory using various extractors ORB-SIFT-SURF-AKAZE-KAZE-BRISK

In fig. 11, (a) represents feature matching using FLANN with ORB extractors, (b) represents feature matching with SIFT extractors and (c) represents feature matching with SURF extractors. Fig 12 clearly visualizes the FLANN and BF Matcher using ORB feature extractors. It is analyzed that FLANN is constantly faster than BF Matcher with respect to individual frames.

**Pose estimation**

We are using KITTI dataset for comparison and estimating the error/ overlap percentage of our pose estimation algorithm. Some of the specs of KITTI dataset are Grayscale frames, consists of 22 stereo sequences (We have used 1$^{st}$ sequence which has 4500 stereo frames), 11 sequences with ground truth (for finding error/ overlap ratios). We are experimenting with 0$^{th}$ sequence of grayscale frames (right and left frames). The ground truth trajectory in fig. 7 in red color is derived from the KITTI data taken using the GPS and IMU readings.

In fig. 16, (a) represents the error vs frame number for FAST, (b) represents for SIFT, (c) for SURF and (d) for ORB.

## 5. Conclusion

When it comes to feature extractors- ORB is most efficient, SIFT is most stable, SURF is faster than SIFT, Shi-Tomasi is fastest among all the extractors, FAST is second most efficient when compared to ORB in terms of accuracy and computation. When it comes to feature matchers, FLANN is more efficient than BF Matcher. For outlier removal, RANSAC … For PnP, EPnP … Thus, we are proposing ORB extractors; FLANN matchers; RANSAC outlier removal and EPnP matrix retrieval to be the best techniques that can be used for SVO and the results are proved in the previous section.

## 6. References

[1]. S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and

BRISK," 2018 International Conference on Computing, Mathematics and Engineering Technologies 2018 pp. 1-10.
[2]. Bansal, M., Kumar, M. & Kumar, M. 2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors. Multimed Tools Appl 2021
[3]. Chien, H.-J., Chuang, C.-C., Chen, C.-Y., & Klette, R. (2016). When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry. 2016.
[4]. Ebrahim Karami and Siva Prasad and Mohamed Shehata Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images 2017
[5]. Rosten, Edward; Drummond, Tom (2006). "Machine Learning for High-speed Corner Detection". Computer Vision – ECCV 2006. Lecture Notes in Computer Science. 3951. pp. 430–443
[6]. D. G. Lowe, "Object recognition from local scale-invariant features." In Proc. ICCV, vol.2, pp.1150–1157, 1999
[7]. H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features." In Proc. European Conf. Computer Vision, 2006.
[8]. Calonder M, Lepetit V, Strecha C, Fua P. 2010. Brief: binary robust independent elementary features. In: Computer vision-eccv 2010; Crete: Springer; p. 778–792.
[9]. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF." In Proc. Int. Conf. Computer Vision, pp. 2564–2571, 2011
[10]. J. Shi and C. Tomasi (1994) Good Features to Track.
[11]. S. Leutenegger et al., "BRISK: Binary robust invariant scalable keypoints," in IEEE International Conference on Computer Vision, Barcelona, ICCV, 2011, pp. 2548-2555.
[12]. P. F. Alcantarilla et al., "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in British Machine Vision Conference, Bristol, BMVC, 2013.
[13]. P. F. Alcantarilla et al., "KAZE features," in European Conference on Computer Vision, Berlin, ECCV, 2012, pp. 214-227.
[14]. Mair E., Hager G.D., Burschka D., Suppa M., Hirzinger G. Adaptive and Generic Corner Detection Based on the Accelerated Segment Test; Proceedings of the 11th European Conference on Computer Vision; Heraklion, Greece. 5–11 September 2010; pp. 183–196