Visual Odometry on KITTI: Pipeline Implementation, Performance Analysis, and Experimental Extensions

Simon Frøyland

September 6, 2024

Abstract

This paper presents the development and evaluation of a SIFT feature-based 2D-2D visual odometry (VO) pipeline. The implemented VO system is evaluated using the KITTI benchmark dataset, a widely recognized resource for testing visual odometry and SLAM systems. The evaluation includes a comparison against ground truth pose information, using selected sequences and performance metrics to assess the accuracy of the system.

In addition to the core implementation and evaluation, I conduct an extended experiment to further analyze the system's performance. This experiment explores a comparative analysis with alternative feature detection methods which are applied to the pipeline. The findings from this experiment provide deeper insights into the strengths, weaknesses, and potential areas for improvement of the implemented VO pipeline.

1 Introduction

Visual odometry is a critical component in autonomous systems, providing essential information about a vehicle's motion by analyzing sequences of images captured by onboard cameras. Among various VO techniques, stereo vision-based approaches have garnered attention due to their ability to exploit depth information directly from stereo camera pairs. This capability is particularly valuable in urban autonomous driving scenarios, where precise localization and motion estimation are crucial for safe and reliable navigation.

In this paper, I present the implementation and evaluation of a feature-based visual odometry pipeline, designed to operate with stereo vision data. The use of stereo cameras allows the system to leverage both 2D image features and 3D spatial information, which can be used for trajectory estimation even in challenging environments with complex geometries and dynamic elements.

The implemented VO system is tested using the KITTI benchmark dataset, a well-established resource for evaluating visual odometry and SLAM systems. By comparing the estimated trajectories with the ground truth provided in the dataset, I assess the accuracy and reliability of my pipeline under various driving conditions. The evaluation metrics and selected sequences are chosen to highlight the strengths and limitations of my approach, ensuring a comprehensive analysis of its performance.

Beyond the core implementation and evaluation, I extend my investigation through an extended experiment. This experiment explores how my VO system performs with other forms of feature detectors, offering insights into the relative advantages and drawbacks of different approaches within my pipeline. This analysis not only underscores the effectiveness of the implementation but also identifies areas where improvements can be made, guiding future work in the field of visual odometry.

2 Methodology: Visual Odometry Pipeline

The visual odometry pipeline implemented in this study follows a structured approach, applying SIFT (Scale-Invariant Feature Transform) algorithm for feature detection, description, and matching. The process includes several steps, such as 2D2D motion estimation, depth estimation from stereo images, and trajectory calculation, to determine the camera's movement through an urban environment. Figure 1 shows the fundamental pipeline overview.

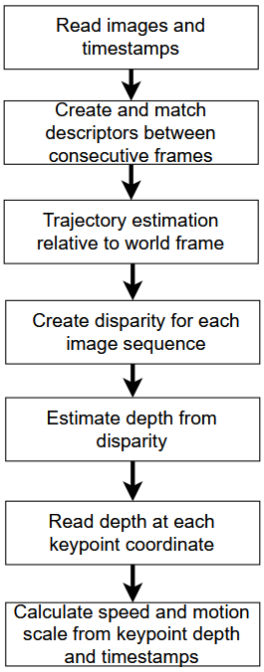


Figure 1: Pipeline overview applying 2D2D motion estimation with SIFT feature detection, and disparity map to scale motion.

2.1 Feature Detection and Matching

The pipeline begins by reading the stereo image sequence from the KITTI dataset [8], where images are first converted to grayscale. SIFT is used to detect keypoints and generate descriptors that are invariant to scale, rotation, and illumination changes. These descriptors are then matched between consecutive frames to establish correspondences.

SIFT works by taking an image $I(x, y)$ and convolving it with a variable-scale gaussian $G(x, y, \sigma)$ to create the scale space, $L(x, y, \sigma)$, of that image.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

The difference between scale-space, $D(x, y, \sigma)$, is then calculated to obtain scale invariant keypoint locations. This is done by subtracting nearby scale-space functions, which are seperated by a constant k.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ D(x, y, \sigma) &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{2}$$

To detect local maxima and minima in the Difference of Gaussians function $D(x, y, \sigma)$, each candidate point is compared with its neighbors. The point is checked against eight neighboring points in the same scale, as well as nine points in the scales immediately above and below it. The point is identified as an extremum if it is either greater than all of these neighboring points (local maximum) or smaller than all of them (local minimum). This process ensures that only the most distinct keypoints, which represent significant local features in the image, are selected [3].

After detecting keypoints, the SIFT descriptors are matched between consecutive frames. The matching process involves comparing the descriptors of keypoints in one frame with those in the subsequent frame. The Euclidean distance between descriptors is used as a measure of similarity.[7]

Lowe's ratio test is applied, which only retains matches where the closest neighbor is significantly closer than the second-closest neighbor. This reduces the likelihood of incorrect matches, which may affect the pose estimation.

2.2 Pose Estimation and Trajectory Calculation

With the matched features, the pipeline estimates the relative pose between consecutive images using a function that computes the motion from 2D-2D correspondences. The motion is represented as an SE3 transformation, capturing the translation and rotation of the camera between frames. These relative poses are accumulated to reconstruct the camera's trajectory.

To calculate the 2D-2D motion an essential matrix is used. The essential matrix E contains epipolar geometry between two views of a scene. It's estimate is computed from the coordinates of matched feature points between two consecutive images taken by the same moving camera. In this case the right camera is used. [6].

For two corresponding points $p_1 = (u1, v1)$ in the first image and $p_2 = (u2, v2)$ in the second image, the epipolar constraint is given by:

$$p_2^T E p_1 = 0 \tag{3}$$

Here p_1 and p_2 are the homogeneous coordinates of the matched points. E is the essential matrix.

To estimate the essential matrix E, the code employs the RANSAC (Random Sample Consensus) algorithm, which iteratively estimates the matrix while rejecting outliers, which are feature matches that don't conform to the model. What is left is a set of inliers, or feature matches that conform to the estimated essential matrix (i.e., $p_2^T E p_1 \approx 0$ within a specified threshold).[1][10]

The essential matrix is then used to recover the pose of the second camera described in the frame of the first. This is done by using singular value decomposition (SVD). Which will return a rotation matrix R and a translation vector t after a chirality check is applied. For more about SVD see Yuefeng Zhang's paper on SVD.[11]

Depth estimation is an integral part of this process, especially when calculating the scale of the motion. The pipeline estimates depth from the disparity between stereo images, using the calibrated focal length and baseline distance.

$$D = \frac{f \cdot B}{disparity} \tag{4}$$

Here D is a matrix containing each pixel depth. Also called a depth map. f is the focal length in pixels, B is the baseline in meters, and $disparity$ is a matrix containing disparity values for each pixel in the image. Also called a disparity map.

Consecutive images from the same camera are then matched. The matched feature point coordinates are mirrored over the depth map D, which is created for each image, and the depth at that coordinate is read and stored together with the other feature point depths. The median of the difference in feature point depths between consecutive images are then calculated to obtain the scale.

$$S = median(d_2 - d_1) \tag{5}$$

Where S is the scale, d_2 is a list/vector of all feature point depths in the last image, and d_1 is the list/vector with all feature point depths in the first image.

The scale factor is then applied to the translation component of the SE3 transformations, allowing the pipeline to convert the relative pose into an absolute scale of the trajectory.

By dividing the scale with the two corresponding timestamps of the images, it is also possible to obtain the speed of the cameras:

$$x = \frac{S}{t_2 - t_1} \tag{6}$$

Where t_1 and t_2 is the time passed in seconds until image 1 and 2 are taken, and x is the speed in $\frac{m}{s}$.

3 Other methods

3.1 Comparing feature detection methods

In urban autonomous driving, choosing an appropriate method is important. The implemented pipeline applies a combination of SIFT and 2D to 2D motion detection. And while SIFT contains many advantages such as scale and rotation invariance, low sensitivity to illumination changes, and rich descriptors, which makes it excellent for matching distinct points across frames[3]. It also has a major disadvantage. It is computationally expensive compared to other methods [5]. In this paper SIFT is compared with three other feature detection methods. ORB (Oriented FAST and Rotated BRIEF), BRISK (Binary Robust Invariant Scalable Keypoints) and AKAZE (Accelerated-KAZE), which are all relevant competitors to replace it.

- ORB is faster and more efficient than SIFT, but may struggle with textured regions or scenes where precise matching is needed, such as in urban environments with complex architectures. In addition ORB has limited scale invariance, making it less robust to significant changes in object size [4].
- Like ORB, BRISK is fast and efficient, and it also offers scale invariance in addition to rotation just like SIFT. It is a strong contender to surpass SIFT when it comes to precision and repeatability. Its disadvantage lies in its descriptor quality. While BRISK is fast, its descriptors are less distinctive than SIFT, and may not perform as well in environments with many similar features like repetitive building facades or along treelines and woods, as seen in [2].
- The final feature detection method is AKAZE. This algorithm is more efficient than SIFT while still providing scale and rotation invariance. It performs well on low-texture regions, which can be beneficial in urban environments with varied textures. AKAZE uses M-LDB descriptors, which are binary and fast to compute, while decently handling noise. AKAZE is less accurate in highly textured environments where precision in feature matching is essential. Its robustness and distinctiveness fall short compared to the SIFT descriptor, particularly in the presence of rotation, perspective, and scale transformations.[9].

3.2 Motion estimation methods

When comparing 2D to 2D motion estimation to other forms like 2D to 3D and 3D to 3D, The nature of the data has to be considered, the application context, and the trade-offs involved in terms of accuracy, computational cost, and real-time performance. Each approach has its strengths and weaknesses.

- 2D2D involves detecting and matching 2D image features between consecutive frames to estimate the relative motion of the camera. 2D2D motion estimation works directly with image coordinates, which makes it straightforward to implement and compute. Since only 2D image points are required, the method is well-suited to feature-based approaches like SIFT, ORB, or AKAZE. It requires less processing power than 3D methods because it doesn't involve reconstructing the 3D structure of the scene.

  2D2D motion cannot recover absolute scale. For example, it can track movement between images but cannot determine how far objects have moved in the real world. Large rotations or translations may cause the method to lose track of features. Since only 2D correspondences are used, this method does not provide depth or 3D structure information, which can be limiting in tasks like obstacle avoidance or path planning.

- 2D3D method involves matching 2D image points to corresponding 3D points in the scene. The 3D points are usually obtained through stereo vision or depth sensors. By using 3D points, the system can recover absolute scale and depth information, providing a more accurate estimate of motion in the real world. Since 3D points are available, this method is generally more robust to large camera motions and can handle scenarios like rapid vehicle turns or motion in dynamic urban environments. The use of 3D points gives the vehicle better awareness of its surroundings, including object locations, distances, and potential obstacles.

  Matching 2D points to 3D structures is computationally expensive, requiring a depth sensor or stereo vision system to estimate the 3D points. The process of aligning 2D image points with 3D points (through methods like PnP – Perspective-n-Point) is more complex and may be slower than 2D2D methods.

- In 3D3D motion estimation, the system estimates the camera motion by comparing two sets of 3D points, typically generated using stereo vision or depth sensors like LiDAR. Since 3D points are already available, this method can directly estimate the camera motion in real-world units, making it highly accurate. This method is accurate even with large camera translations, rotations, and dynamic scene changes, making them ideal for complex driving scenarios where precise vehicle movement estimation is necessary.

  3D motion estimation requires a depth sensor or stereo vision setup, increasing the hardware cost and complexity of the system. The 3D data processing required is computationally expensive and may be slower in real-time systems, making it less suitable for certain real-time applications without high-performance computing resources. In urban driving, where the majority of the scene is flat (roads, sidewalks), a lot of 3D data may be redundant, and simpler 2D2D methods may suffice for motion estimation.

3.3 Pipeline: expected failure cases with SIFT

- Low Texture or Homogeneous Surfaces

  SIFT relies on distinctive keypoints such as corners and edges, but certain environments often have large areas with low texture like roads and open landscapes. It may fail to detect enough keypoints in such areas, leading to poor feature matching and inaccurate motion estimation.

- Dynamic Objects

  Moving objects in the scene, like cars, people and bicycles can introduce features that are inconsistent between frames. SIFT may detect and match features on

moving objects, which can lead to erroneous essential matrix estimation and incorrect pose recovery, as the algorithm assumes static scene structure.

4 Testing the implemented pipeline

In this section, I evaluate the performance of the 2D-2D motion estimation system implemented using SIFT features, tested on the KITTI dataset. The selected sequences span diverse urban driving environments, including city (sequence 0018), highway (sequence 0052), and residential areas (sequence 0035), to assess the algorithm's robustness across different scenarios. I chose these specific sequences due to their representative nature within the dataset and their relatively short lengths, allowing for efficient computation while still providing meaningful results. Evaluation metrics such as Absolute Trajectory Error (ATE), the number of matched features, and the number of inliers were used to compare the system's estimated trajectories with the ground truth. My results highlight that while the SIFT-based approach achieves relatively accurate pose estimations in city and residential environments, it struggles in highway sequences due to dynamic objects and a depth threshold within the python code set at 0-50 meters. Adjusting the depth threshold to 35 meters and beyond improved the accuracy considerably on the highway sequence. The detailed analysis of each scenario and the algorithm's performance is presented in the subsequent sections.

4.1 Experimental Setup and Evaluation Protocol

For this experiment, three sequences from the KITTI dataset were chosen: sequence 0018 (city), sequence 0052 (highway), and sequence 0035 (residential). These sequences were selected to evaluate the algorithm's performance across distinct driving environments:

- City (0018): This sequence contains dense urban settings with a variety of structures, traffic, and potential occlusions, making it an ideal test case for feature detection and matching in highly textured scenes.
- Highway (0052): The highway sequence introduces high-speed motion with relatively sparse features, posing a challenge for feature detection and tracking. Additionally, it includes dynamic objects like other vehicles, which can interfere with motion estimation.
- Residential (0035): This sequence includes a suburban environment with moderate speeds, low dynamic interference, and a mix of structured and unstructured features. It provides a balanced scenario between the urban density of the city sequence and the high-speed motion of the highway sequence.

These sequences were chosen to ensure that the performance of the algorithm could be evaluated under a variety of real-world conditions, from high-speed highway driving to dense urban environments. Furthermore, the relatively short length of each sequence allows for a more computationally efficient evaluation. Although having a shorter sequence may not show difficulties the pipeline may encounter in a longer sequence.

To assess the performance of the VO-pipeline, The following metrics were used:

- Absolute Trajectory Error (ATE):

  ATE measures the difference between the estimated and ground truth positions of the vehicle over time. The mean, median, maximum, and root mean square error (RMSE) of the ATE was computed for each sequence. This metric was chosen as it directly quantifies the accuracy of the estimated trajectory in relation to the ground truth.

- Matched Features:

  The number of matched features between consecutive frames was recorded. This metric reflects how well SIFT detects and matches features in different environments. A high number of matches indicates better performance, as it provides more data for estimating motion.

- Inliers:

  Inliers represent the number of feature matches that were successfully used for motion estimation. This metric is essential for understanding the reliability of the feature matching process and the effectiveness of RANSAC in eliminating unreliable matches.

4.2 Results and Performance Analysis

In the **city sequence**, the algorithm performed well, with a low mean ATE of 922.45 mm and RMSE of 673.90 mm, benefiting from abundant features. The RANSAC is particularly active in this scenario. removing a lot of outliers. This can be seen by the fluctuating inlier curve in figure 2d.

The **residential sequence** showed similar performance, with a mean ATE of 1099.89 mm and RMSE of 828.09 mm, indicating that the algorithm is robust in suburban settings. Figures 2a to 2b shows the estimated trajectory with the ground truth for the residential and city sequence. These sequences had a somewhat high match count compared to the highway scenario, which was the most challanging.

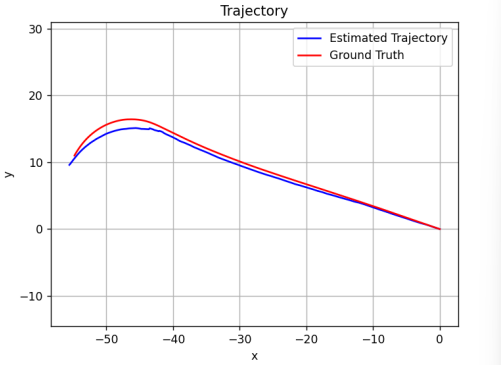
The **highway sequence**, where the mean ATE was initially 2735.59 mm due to dynamic objects and sparse features as was expected. However, adjusting the depth threshold from 0-50m to 35m and above improved the results significantly, reducing the mean ATE to 1291.50 mm and RMSE to 1167.75 mm. Despite these improvements, the max ATE remained high, reflecting difficulties in handling high-speed motion and dynamic objects. Figures 3a to 3d shows the results from the threshold change. Matches in the two images seems to be quite equal to each other, although relatively low compared to the two other sequences.

The results align with the expected performance of the VO-pipeline. It performed well in urban environments as anticipated, but faced difficulties in highway scenarios, confirming the algorithm's known weaknesses in handling sparse and dynamic scenes.

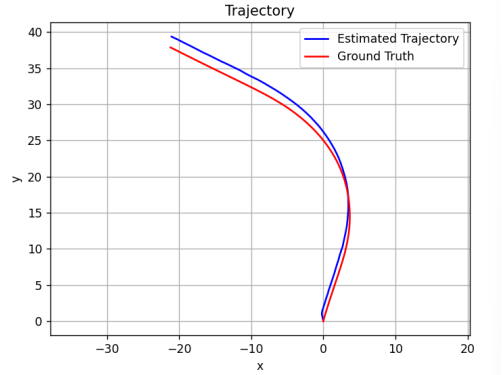
Table 1: Performance Evaluation Results on KITTI Sequences

Sequence	Mean ATE (mm)	Median ATE (mm)	Max ATE (mm)	RMSE (mm)
City (0018)	922.45	979.09	1529.37	673.90
Highway (0052)	2735.59 (1291.50)*	3399.53 (938.57)*	3736.39 (3989.06)*	2118.67 (1167.75)*
Residential (0035)	1099.89	1174.64	1713.19	828.09

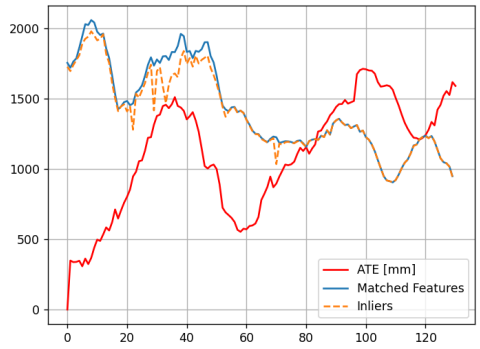
*Results after adjusting depth threshold to 35m.



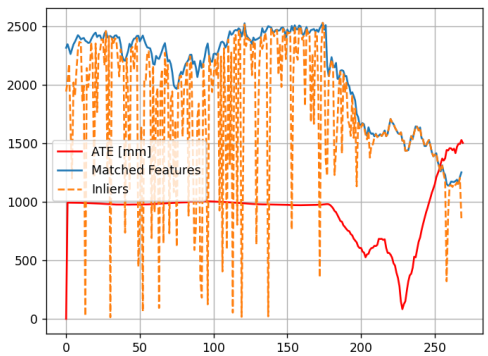
(a) Trajectory estimation for the residential sequence (0035).



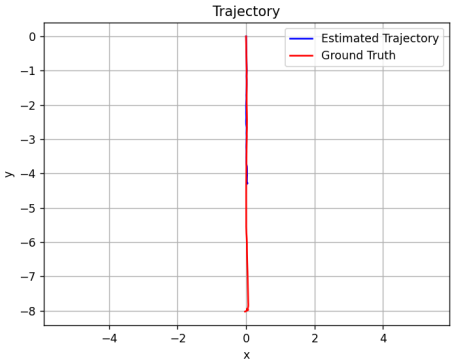
(b) Trajectory estimation for the city sequence (0018).



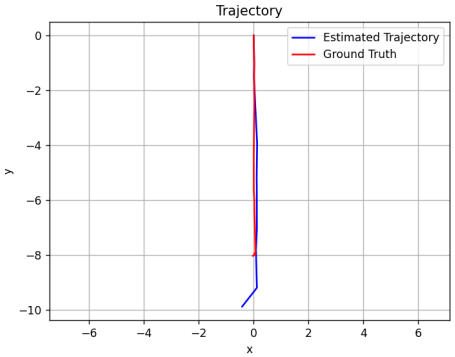
(c) Results on residential sequence (0035).



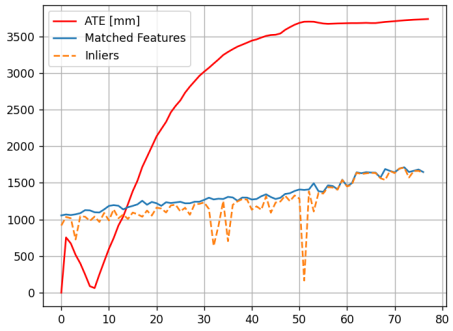
(d) Results from city sequence (0018).



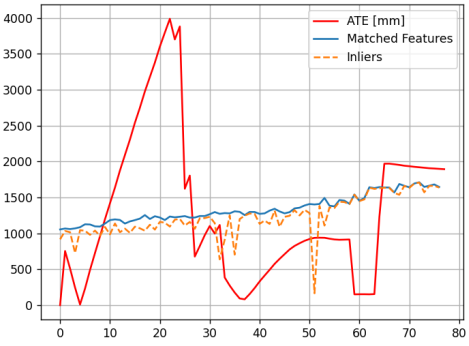
(a) Trajectory estimation on highway sequence (0052) using depth threshold < 50m.



(b) Changing depth threshold to > 35m.



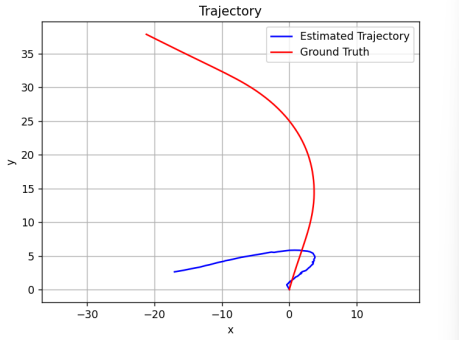
(c) Results from using depth threshold $<$ 50m.



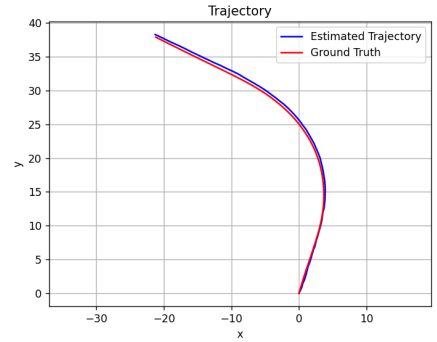
(d) Results from changing depth threshold to > 35m.

5 Extended experiment

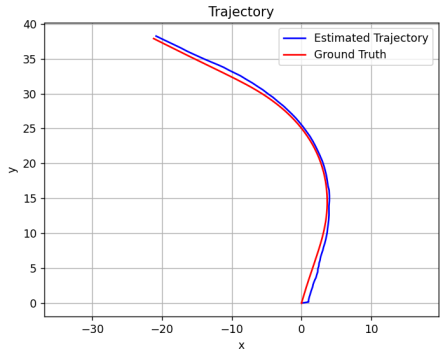
To further extend the experimentation on the pipeline the SIFT algorithm was replaced by the other feature detection methods discussed in this paper. ORB, AKAZE, and BRISK. In the following experiment the city (0018) sequence is used. Table of the results can be seen in the appendix 2.



(a) ORB trajectory



(b) AKAZE trajectory



(c) BRISK trajectory

- **ORB** showed the highest mean ATE of 4942.57 mm and RMSE of 7079.80 mm. While ORB is computationally efficient, its performance suffered due to poor handling of texture-rich areas and dynamic scenes, leading to large max ATE values of 35,507.05 mm. Its lower Median ATE suggests that it performs well in some regions but struggles with more challenging scenarios, especially when scale and rotation variance is present.
- **AKAZE** demonstrated a strong balance between performance and computational efficiency, with a mean ATE of 899.64 mm and a low RMSE of 655.89 mm. It excelled in both feature-rich and low-texture environments, making it the most consistent of the methods tested. The relatively low max ATE of 1095.95 mm highlights its reliability, even in challenging situations.
- **BRISK** achieved decent results with a mean ATE of 1017.63 mm and an RMSE of 730.49 mm. While it offered robust feature matching in moderately challenging environments, it was outperformed by AKAZE in terms of overall accuracy. Its max ATE of 1663.32 mm indicates it struggles more in complex scenarios compared to AKAZE but is more reliable than ORB.

Among the four methods, AKAZE proved to be the most reliable and balanced for this sequence, achieving a low mean ATE of 899.64 mm and an RMSE of 655.89 mm. It performed well in both feature-rich and low-texture environments, with a relatively low max ATE of 1095.95 mm, showcasing its consistent accuracy across different scenarios. BRISK also showed good performance with a mean ATE of 1017.63 mm and RMSE of 730.49 mm, but it was less accurate than AKAZE, especially in complex scenes. SIFT performed comparably with a mean ATE of 922.45 mm and RMSE of 673.90 mm, benefiting from the rich visual features in urban environments, although it experienced a high rate of outliers. In contrast, ORB demonstrated the weakest performance, with the highest mean ATE of 4942.57 mm and RMSE of 7079.80 mm, along with a max ATE of 35,507.05 mm. This indicates that while ORB is computationally efficient, it struggles significantly in texture-rich and dynamic scenes, resulting in lower accuracy compared to the other methods.

6 Conclusion

The evaluation of the 2D-2D motion estimation pipeline utilizing SIFT features, tested across diverse sequences from the KITTI dataset, provides insightful findings on the algorithm's performance. The selected sequences—city (0018), highway (0052), and residential (0035)—were chosen to assess the pipeline's efficacy in varied driving environments.

In urban settings (sequence 0018) and suburban environments (sequence 0035), the SIFT-based approach demonstrated strong performance, yielding accurate pose estimations with a low mean Absolute Trajectory Error (ATE) and Root Mean Square Error (RMSE). This indicates that SIFT is effective in handling dense, textured scenes and moderate dynamic interference.

However, the highway sequence (0052) presented significant challenges due to sparse features and dynamic objects. Initially, the pipeline struggled with a high mean ATE, but adjustments to the depth threshold improved accuracy considerably. Despite these improvements, the algorithm continued to face difficulties, particularly in high-speed scenarios.

The comparative analysis with alternative feature detection methods—ORB, AKAZE, and BRISK—revealed that AKAZE offered the most balanced performance, combining accuracy with computational efficiency. ORB, while efficient, struggled in texture-rich and dynamic scenes, leading to the highest errors. BRISK showed decent performance but was outperformed by AKAZE. SIFT performed competitively in urban settings but had limitations in high-speed and dynamic environments.

Overall, the results underscore the strengths of the SIFT-based pipeline in urban and suburban scenarios while highlighting areas for improvement in handling dynamic and high-speed conditions. Future work could explore further adjustments or alternative feature detection methods to enhance performance across a wider range of driving conditions.

References

[1] Ragon Ebker. Random sample consensus explained. https://www.baeldung.com/cs/ransac.

[2] S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE, 2011.

[3] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 5–7, 2004.

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 11 2011.

[5] Shaharyar Ahmed Khan Tareen — Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8346440.

[6] Cyrill Stachniss. Fundamental and essential matrix - 5 minutes with cyrill. https://www.youtube.com/watch?v=auhpPoAqprk.

[7] Brown university. Efficient sift feature matching with matrices. https://cs.brown.edu/courses/csci1430/2021_Spring/proj2_featurematching/matching/.

[8] A. Geiger — P. Lenz — C. Stiller — R. Urtasun. Kitty-360. https://www.cvlibs.net/datasets/kitti/.

[9] Dan Li — Qiannan Xu — Wennian Yu2 — Bing Wang. Srp-akaze: an improved accelerated kazealgorithm based on sparse random projection. https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/iet-cvi.2019.0622.

[10] XI WU. Project 3 / camera calibration and fundamental matrix estimation with ransac. https://sites.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/xwu72/index.html.

[11] Yuefeng Zhang. An introduction to matrix factorization and factorization machines in recommendation system, and beyond. https://arxiv.org/pdf/2203.11026.

Appendix

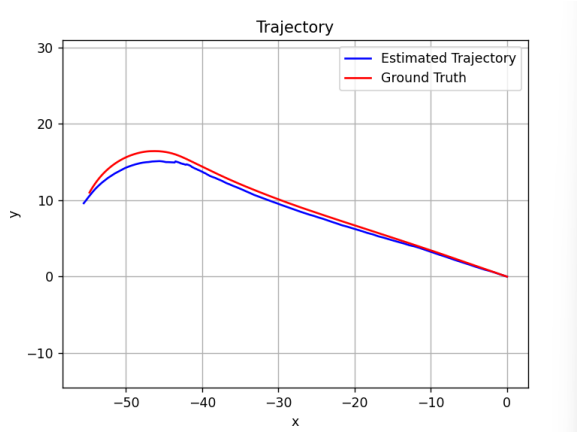
Extended Experiment Table

Table 2: Performance Evaluation Results by Feature Detection Method for city sequence (0018)

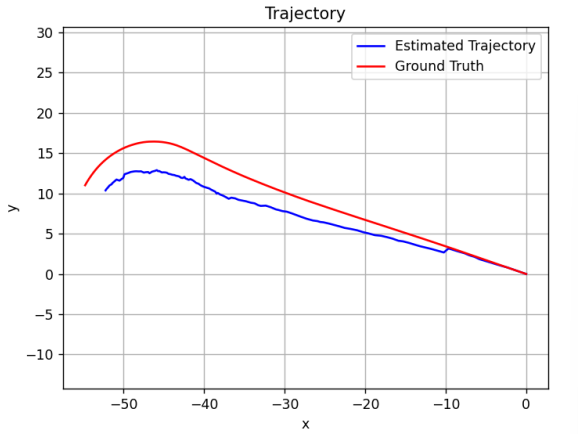
Method	Mean ATE (mm)	Median ATE (mm)	Max ATE (mm)	RMSE (mm)
ORB	4942.57	971.31	35507.05	7079.80
AKAZE	899.64	997.07	1095.95	655.89
BRISK	1017.63	1012.46	1663.32	730.49
SIFT	922.45	979.09	1529.37	673.90

Extended Experiment: The Extended Edition

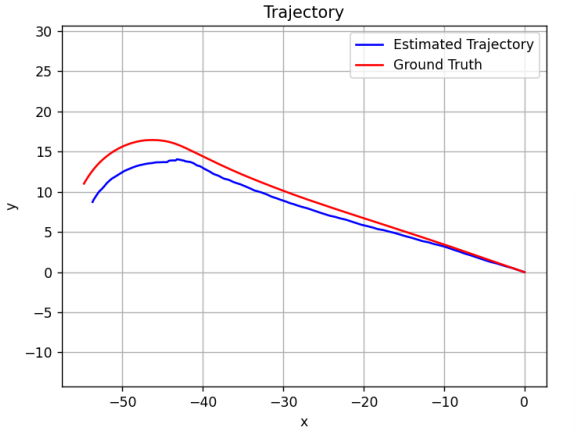
Residential results (0035):



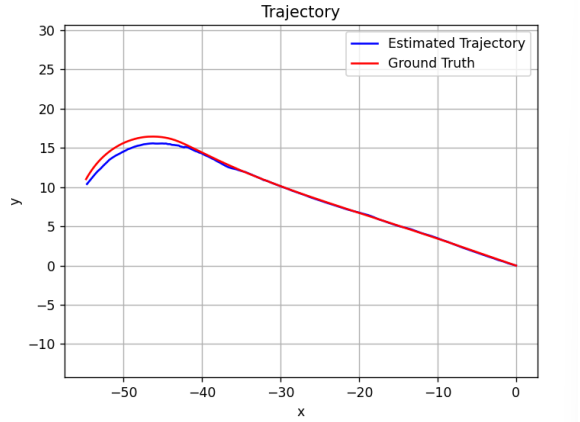
(a) SIFT trajectory estimation for the residential sequence (0035).



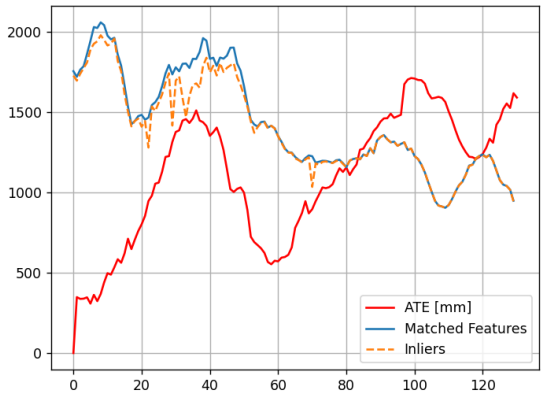
(b) ORB trajectory estimation for the residential sequence (0035).



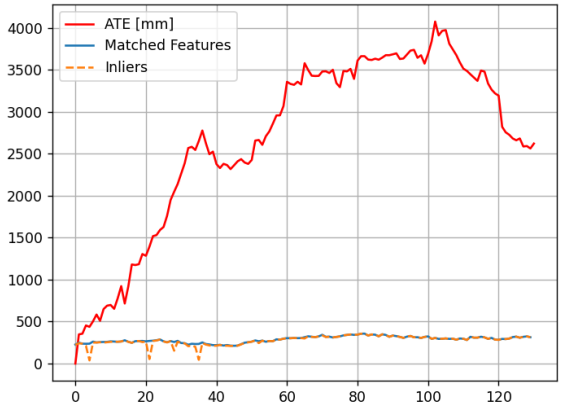
(c) BRISK trajectory estimation for the residential sequence (0035).



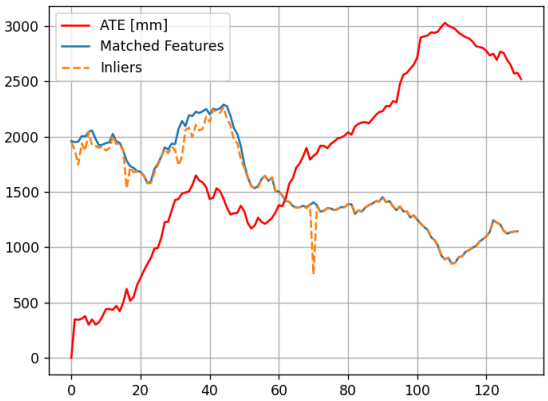
(d) AKAZE trajectory estimation for the residential sequence (0035).



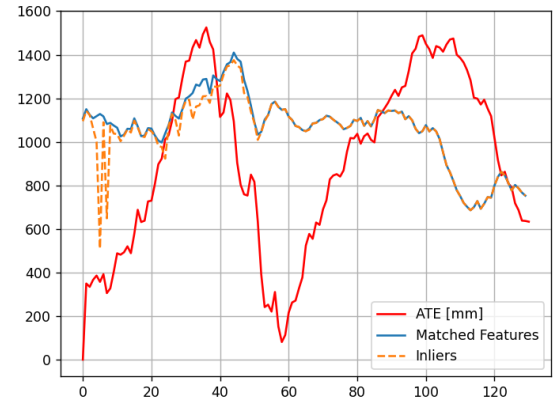
(a) SIFT trajectory ATE, matches and inliers for the residential sequence (0035).



(b) ORB trajectory ATE, matches and inliers for the residential sequence (0035).



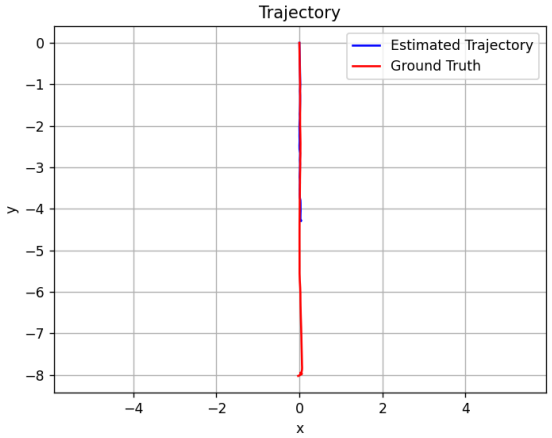
(c) BRISK trajectory ATE, matches and inliers for the residential sequence (0035).



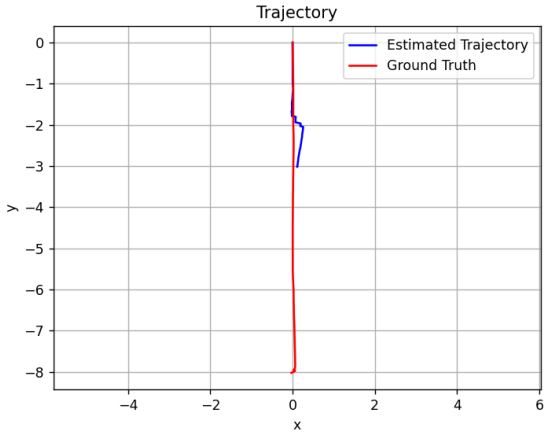
(d) AKAZE trajectory ATE, matches and inliers for the residential sequence (0035).

Highway results (0052):

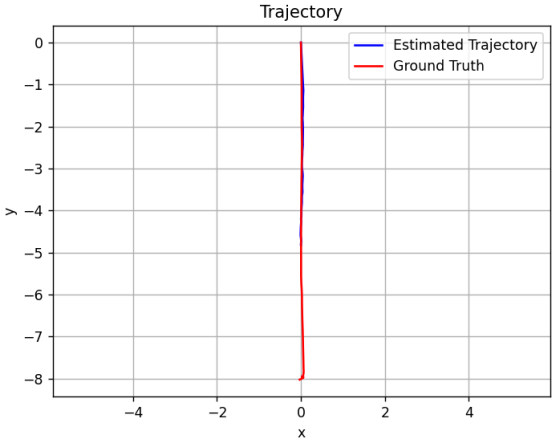
Here the threshold of the depth was at fault for creating scales out of proportions.



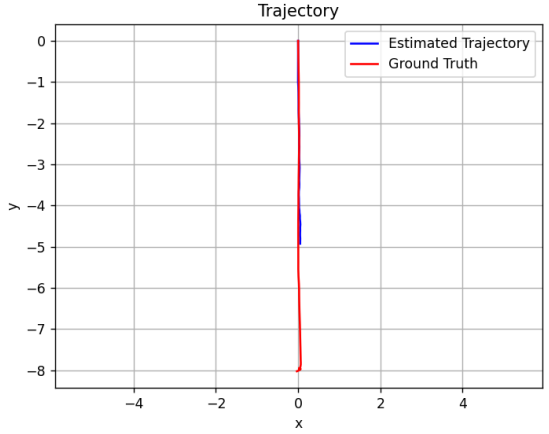
(a) SIFT trajectory estimation for the highway sequence (0052).



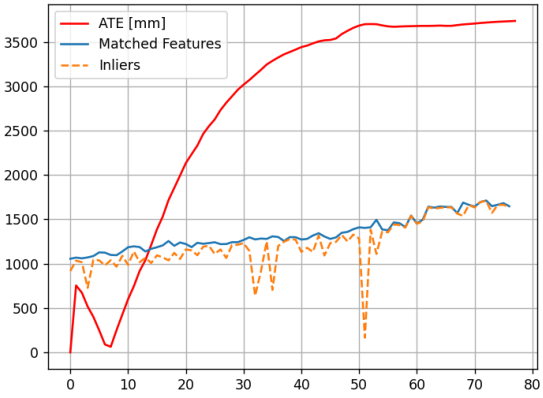
(b) ORB trajectory estimation for the highway sequence (0052).



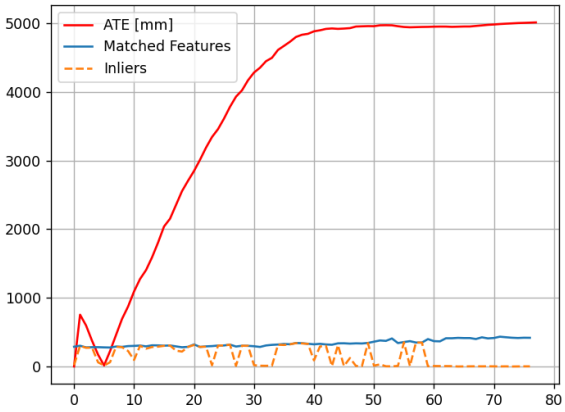
(c) BRISK trajectory estimation for the highway sequence (0052).



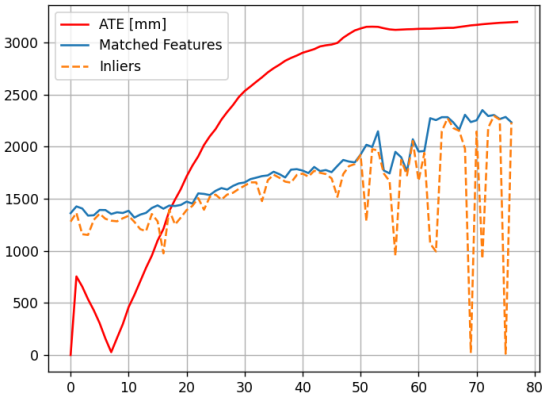
(d) AKAZE trajectory estimation for the highway sequence (0052).



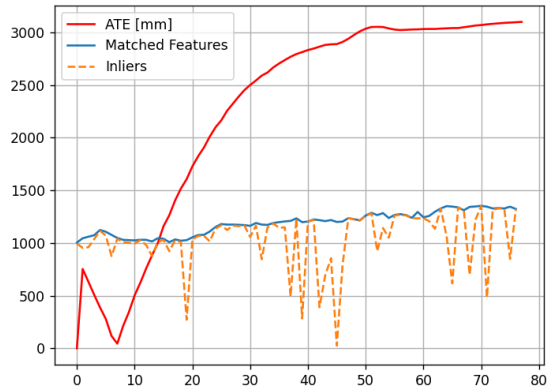
(a) SIFT trajectory ATE, matches and inliers for the highway sequence (0052).



(b) ORB trajectory ATE, matches and inliers for the highway sequence (0052).



(c) BRISK trajectory ATE, matches and inliers for the highway sequence (0052).



(d) AKAZE trajectory ATE, matches and inliers for the highway sequence (0052).