

Novel Stereo Visual Odometry Pipeline for Autonomous Robot Navigation on Video Data

Subhobrata Chakraborty
Department of Computer Science
California State University
Northridge, CA, USA
subhobrata.chakraborty.026@my.csun.edu

Abhishek Verma
Department of Computer Science
California State University
Northridge, CA, USA
abhishek.verma@csun.edu

Abstract— Autonomous robot navigation relies on accurate motion estimation. Stereo Visual Odometry is used to estimate the motion of a camera by analyzing the visual information captured by a stereo pair of cameras. Visual odometry refers to the process of estimating the camera’s movement by tracking the visual features in consecutive frames of a video sequence. In a stereo camera system, two cameras are positioned at known baseline separation. The images captured by the cameras are used to create a 3D representation of a scene. Stereo visual odometry leverages the disparity between corresponding points in the pair of stereo images to estimate the motion of camera.

We propose a stereo visual odometry pipeline and perform a comprehensive comparison of different trajectory computation approaches on the challenging KITTI dataset. Different feature detection methods, feature matching, filter matching distance, stereo matching, different types of motion estimation, optimization techniques such as loop closure and bundle adjustment were tested in terms of error metric and computational time. SGBM matcher performed better than the BM matcher on depth and disparity estimation. SIFT was found to outperform the ORB feature detector. Brute Force feature matcher outperformed FLANN matcher. The lowest MAE error rate of 32.54 was obtained with SGBM matcher, brute force approach, PnP RANSAC, and filter matching distance of 0.3 on SIFT features.

Keywords—stereo visual odometry, KITTI, loop closure, bundle adjustment

I. INTRODUCTION

Mobile robotics and autonomous robot navigation is a vital research area due to its various applications such as self-driving vehicles, search and rescue, surveillance and security, deep sea exploration. Localization proves crucial for autonomous robot navigation, especially in environments lacking GPS signals or where identifying distinct landmarks is challenging, such as in indoor or underwater settings. Visual odometry (VO) emerges as a widely employed solution for robot localization, relying solely on camera input. Notably, cameras, being passive sensors, offer energy efficiency compared to active counterparts like sonar or LiDAR, thereby extending deployment durations and minimizing downtime. Visual odometry, depending on the camera configuration, is classified into monocular or multi-camera systems, with stereo VO being the predominant choice among the latter.

Stereo VO [1, 2, 3] typically starts with stereo matching, seeking corresponding features between stereo frames. Triangulation swiftly estimates the 3D positions of objects, followed by the derivation of camera pose (position and orientation) in relation to those 3D points. While this method provides accurate results, stereo matching poses computational challenges. Moreover, many stereo matching algorithms involve rectifying the stereo pair, adding to the time complexity. Challenges also arise in scenes with repetitive, high-frequency textures, where multiple similar patches may lead to ambiguity in determining the best match. Such scenarios are common outdoors, presenting a significant hurdle for field robots like those navigating underwater or exploring mines.

Related work is covered in Section II of this paper. The dataset used in our experiments is described in Section III. Section IV gives details of our proposed pipeline for stereo visual odometry computation that address some of the aforementioned challenges followed by the experiment results in Section V. Conclusion and future work is presented in Section VI.

II. RELATED WORK

Approaches that rely on stereo matching are widely explored in stereo visual odometry. Recent advancements include S-PTAM [4], which is an extension of PTAM where stereo matching is incorporated to generate 3D points [5]. ORB SLAM [6, 7, 8] incorporates stereo matching to perform stereo visual odometry. LSD SLAM [9] was further expanded to a stereo visual odometry system. Monocular LSD SLAM [10] relies on direct method where the photometric error is reduced without relying on feature matching. However, certain drawbacks of stereo matching include failures in repetitive scene textures and computational inefficiency. The stereo matching methods mainly rely on patch appearances like normalized cross-correlation or feature descriptors for determination of the stereo correspondence. To further enhance the process of stereo matching research has been done on global stereo matching to incorporate non-local constraints like smoothness. This can be seen in the stereo VO developed by Stereolab for the ZED stereo camera [11]. Even though this approach is capable of enhancing the localization accuracy, the real time performance is attained through GPU based stereo matching thereby increasing the power consumption and the system complexity.

Forster et al. improved SVO [12, 13] for systems with multiple cameras but not primarily for stereo cameras. Rather



Fig. 1. An example of the first frame of sequence 0 of the KITTI dataset [18].

than depending on stereo matching, all the cameras are brought together into a single function to minimize the photometric error which involves projecting 3D points onto all visible image frames, thereby enhancing the accuracy. The drawback is that it increases the computational cost. Stereo DSO [13] represents a hybrid model that initializes the depth for each keyframe via stereo matching [14] along with stereo images integrated into the error function.

Visual Odometry solely relies on stereo cameras to estimate the depth and motion of features in the environment and might struggle to extract sufficient depth information or features from the environment. Visual inertial odometry [16, 17, 18] combines information from both visual and inertial sensors for better motion estimation while leveraging the data from the inertial measurement unit. The fusion of visual data along with inertial data helps in mitigating the limitations of visual sensors in low-lighting conditions or lack of distinctive features in the environment.

III. DATASET DESCRIPTION

The KITTI dataset [18] is a very widely used benchmark dataset used in the field of computer vision and robotics with its primary focus being on autonomous vehicles. Hence, we test our proposed visual odometry pipeline on this dataset. It includes a variety of sensor data collected from a moving vehicle. The following sensor data are collected:

- LiDAR: This data contains point cloud data collected from Velodyne HDL-64E LiDAR.
- Stereo Camera: This data contains images captured from multiple stereo pairs mounted on the vehicle which helps capture images from different viewpoints.
- GPS and IMU: This data help in precise localization and state estimation.
- Object detection and tracking: KITTI dataset provides ground truth annotations for objects such as cars, pedestrians, cyclists, etc. It includes 3D bounding box

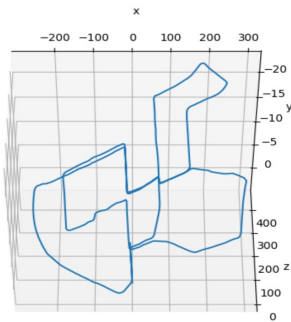


Fig. 2. Ground truth trajectory plot of sequence 0 of the KITTI dataset.

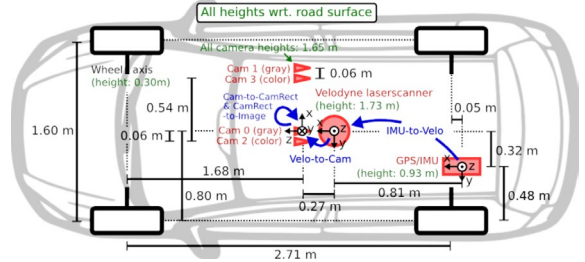


Fig. 3. Sensor configuration of the KITTI dataset.

for objects allowing researchers to focus on 3D object detection models.

Fig. 1 shows the first frame of the sequence 0 of the KITTI dataset, which we use in our experiments.

IV. RESEARCH METHODOLOGY

The KITTI dataset consists of both RGB and grayscale images. For this research, grayscale images were used. There are 4,541 frames in the dataset along with 12 columns that comes from flattening a 3×4 transformation matrix of the left and right cameras with respect to the global coordinate frame, which is established according to the first position of the camera's frame. The transformation matrix is a 3×3 rotation matrix stacked with a 3×1 translation vector. The ground truth poses are extracted from the pose sequence of the KITTI dataset. Fig. 2. represents the ground truth data of the sequence 0 of the KITTI dataset. Fig. 4 shows the proposed stereo visual odometry pipeline, its various components are explained below.

A. Data Acquisition

The trajectory in Fig. 2 ends and starts at the same point. The motion of the camera is tracked with respect to the first camera frame and the timing of the camera is around 10 frames per second.

The sensor calibration data is loaded that gives information about the 3×4 projection matrices for four cameras as well as the transformation matrix for the LiDAR data. It gives us information about the projection matrix of the camera which can be decomposed using singular value decomposition to extract the intrinsic and extrinsic attributes of the camera. Fig. 3 shows the sensor infrastructure for the KITTI dataset.

Finally, all the grayscale images are loaded in the dataset handler and the ground truth frames are compared with the camera frames, the number of rows comes to 4,541.

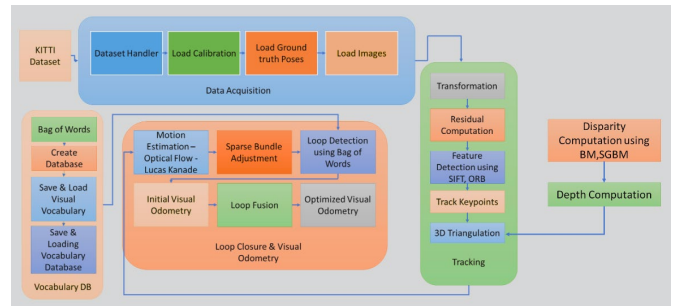


Fig. 4. Architecture of the proposed stereo visual odometry pipeline.

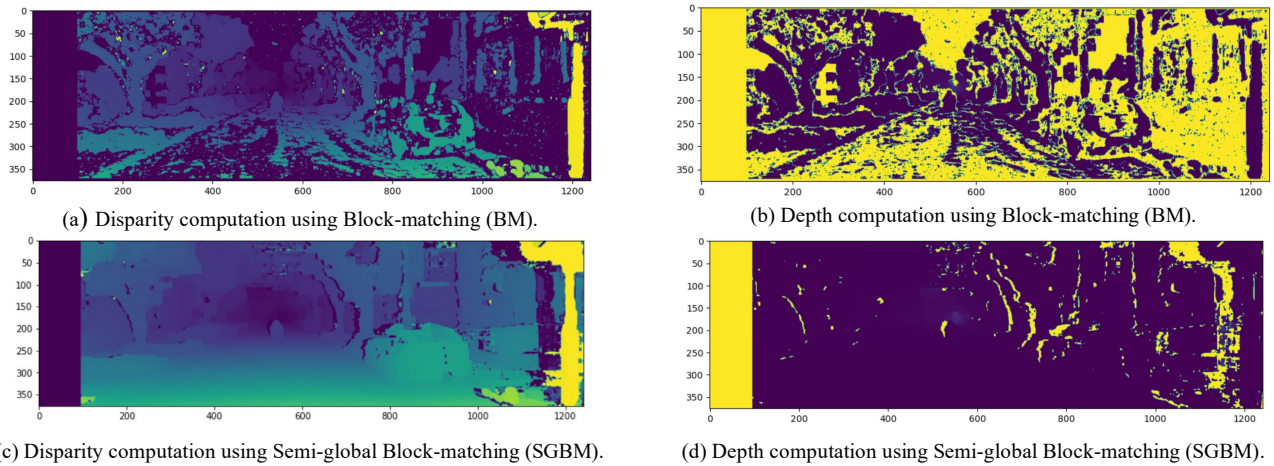


Fig. 5. Comparison between the disparity computation using BM in (a) and SGBM in (c) along with the corresponding depth map estimation using BM in (b) and SGBM in (d) on sequence 0 of the KITTI dataset.

B. Tracking

The transformation matrix is a 3×4 matrix that is generated from the horizontal stacking of the rotation matrix and the translation vector. Following the transformation, the reprojection residuals are computed that allows to extract the difference between the actual and estimated transformation. The following parameters are considered for residual computation – transformation between two frames, feature points in an image, feature points in the subsequent image, 3D points generated from the prior image frame and the 3D points generated from the image frame that follows.

To identify the distinct points in an image, feature detection and matching are done either with SIFT [20] or ORB [21] followed by matching the features with the corresponding ones in the other image. The good features are stored as per the David Lowe’s ratio test and the filter matching distance was set to 0.3.

The keypoints between the frames are tracked using the Lucas Kanade method for optical flow. The keypoints are first converted into a vector of points and the dimensions are expanded to select the good matches. The parameters for tracking keypoints are an image followed by the subsequent image. The first image’s keypoints are considered along with a threshold for maximum acceptable error. The status vector is converted to Boolean to make it usable as a mask, which is later used to select the keypoints. Finally, the keypoints lying outside the image are deleted and the tracking points are returned.

C. Loop Closure and Visual Odometry

Two methods have been used for estimating the motion. One of those is using optical flow while the other uses PnP RANSAC. Lucas Kanade method has been used for the optical flow-based motion estimation. Dense and sparse optical flow are two types of optical flow estimation techniques. Dense optical flow computes motion vectors for every pixel in the image, providing a continuous and dense field of motion information. The result is a dense matrix of motion vectors where each pixel has an associated motion vector. Sparse optical flow computes motion vectors for a subset of feature points in the image. Instead of calculating the motion for every pixel, it focuses on specific, manually or automatically selected interest points. The result is a set of motion vectors corresponding to the selected feature points, providing information about the movement of

these points in the scene. Using sparse optical flow saves computation time where a large number of features are considered.

PnP RANSAC is used to compute a pose that relates points in the global coordinate frame to the pose of the camera. The pose of the camera in the first image has been considered as the global coordinate frame to reconstruct the 3D points of the features using stereo depth estimation. Then a pose is found, which relates the camera in the next frame to those 3D points. While tracking the pose of the vehicle over time, the goal is to relate the points in the camera’s coordinate frame to the global coordinate frame and requires computation of the inverse of the transformation matrix. Furthermore, the vehicle is being tracked from the first camera pose for which the cumulative product of the inverses of each estimated camera pose is required.

The parameters of the 3D reconstruction model are refined to minimize the difference between the observed image features and the features predicted by the model. The goal is to improve the accuracy of the 3D reconstruction by adjusting the parameters to better align with the observed data. Sparse bundle adjustment is used here where only a subset of points in the scene are considered. It is useful while dealing with large-scale reconstruction where considering all the features in all images might prove to be computationally extensive. The focus is to refine the parameters for a sparse set of features.

$$u_l = f_x \frac{x}{z} + o_x, v_l = f_y \frac{y}{z} + o_y \quad (1)$$

$$u_r = f_x \frac{x-b}{z} + o_x, v_r = f_y \frac{y}{z} + o_y \quad (2)$$

$$x = \frac{b(u_l - o_x)}{u_l - u_r}, y = \frac{bf_x(v_l - o_y)}{f_y(u_l - u_r)}, z = \frac{bf_x}{u_l - u_r} \quad (3)$$

Equation 1 denotes the perspective projection of the left camera in a stereo camera configuration and equation 2 represents the perspective projection equation of the right camera. (u_l, v_l) and (u_r, v_r) are points on the image plane of the left and right cameras respectively. f_x and f_y denote the focal lengths across the x and y axis, whereas o_x and o_y represent the optical center. b is the baseline, which is the distance between the optical centers of the stereo pair of cameras. To determine the 3D location of a point in a scene, the x , y and z values need to be determined from equations 1 and 2. Equation 3 represents

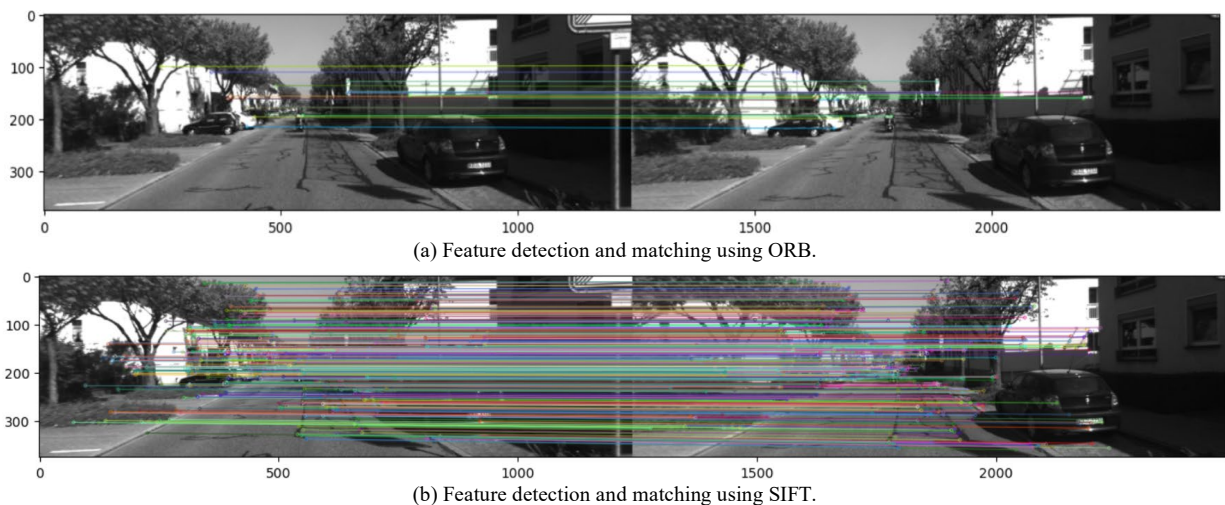


Fig. 6. Comparison of the features detected by ORB in (a) and SIFT in (b) on sequence 0 of the KITTI dataset.

the coordinates of the point in the 3D environment. Here z is also known as the depth of the scene and the denominator, which is the difference between u_l and u_r , represents the disparity. The depth z is always inversely proportional to the disparity.

Loop detection involves identifying when a robot or a camera revisits a location previously visited. Bag of Words (BoW) is a technique for feature representation. As the vehicle navigates the environment local features are extracted using SIFT or ORB. K-Means clustering algorithm is used to quantize the extracted features into a predefined number of visual words. Each image in the frame is represented as a histogram of visual words. As the camera explores further the environment, the BoW representations of the visited locations are stored into a database. Once one of the frames is revisited, the BoW representation is computed for that frame. The current BoW representation is compared to the ones already stored in the database. The Euclidean distance is used to measure the similarities between the BoW histograms. A threshold for the similarity metric is decided. If the similarity between the current BoW representation and any stored representation exceeds the threshold, a potential loop is detected. The threshold may depend on factors such as the environment, sensor noise and the desired level of detection sensitivity. Refinement techniques are further applied to reduce false positives. The goal for loop closure is to improve the overall map consistency.

Once the loop detection is done, the initial visual odometry is computed followed by loop fusion to increase the odometry consistency. The final output is the optimized visual odometry after loop detection and bundle adjustment. The end-to-end system pipeline is shown in Fig. 4.

V. EXPERIMENT SETUP, RESULTS AND DISCUSSION

The following hardware configuration was used to conduct the experiments – Ryzen 9 5900 HX for the CPU, Nvidia RTX 3080 for the GPU, GPU VRAM of 16 GB and the standard RAM of 32 GB. The software libraries include NumPy, Pandas, OpenCV, Matplotlib, and SciPy.

Visual odometry can be computed using feature-based methods and intensity of the image pixel in the sequence can be

leveraged to conduct experiments. It is also referred to as the direct method.

A. Dataset Handling

The grayscale data from the KITTI dataset is used for the experiments. Storing the dataset sequences as lists in attachment with the dataset handling class takes around 20 GB of memory. This can be bypassed by making the low memory argument true, which will create a class with generators rather than lists. The dataset handler takes care of the following components of the sequence – check device for low memory, loading the ground truth poses, loading the calibration details for a scene, loading the images along with the LiDAR data.

B. Disparity Computation

The apparent motion between a pair of stereo images is referred to as disparity. For a stereo pair of images, the pixels in the left and right images are to be matched with each other and the distance between the matching pixels is calculated. The result is a disparity map that represents the distance values as an intensity image. Considering the stereo image pair from left and right cameras of the KITTI dataset, the disparity map is generated using Block-matching (BM) and Semi-global Block-matching (SGBM) algorithms. Fig. 5 (a) and (c) show the comparison of results for depth computation using the BM matcher and the SGBM matcher respectively.

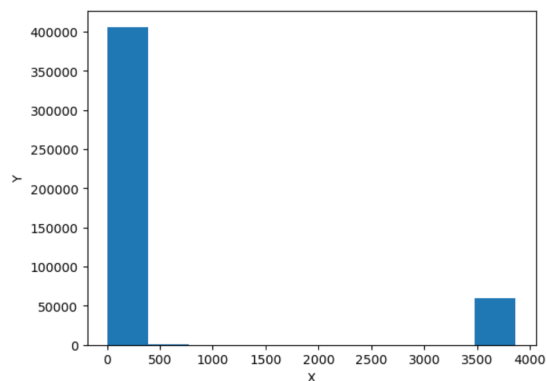
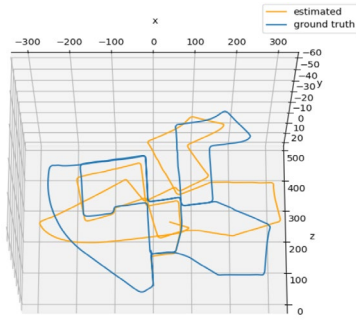
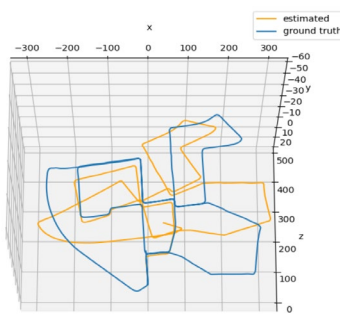


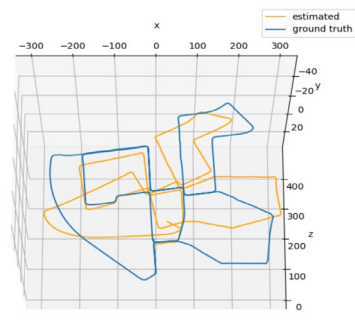
Fig. 7. Histogram analysis demonstrating depth information distribution on sequence 0 of the KITTI dataset.



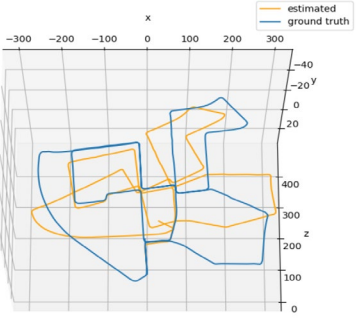
(a) Visual odometry computation using the following parameters-feature detector: SIFT, feature matching: Brute Force (BF), filter matching distance: 0.5, stereo matcher: BM.



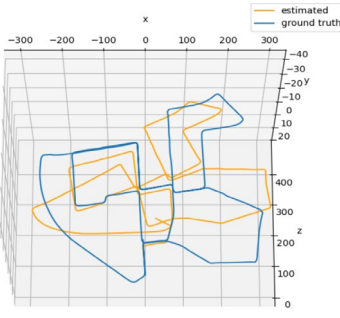
(b) Visual odometry computation using the following parameters-feature detector: SIFT, feature matching: FLANN, filter matching distance: 0.5, stereo matcher: BM.



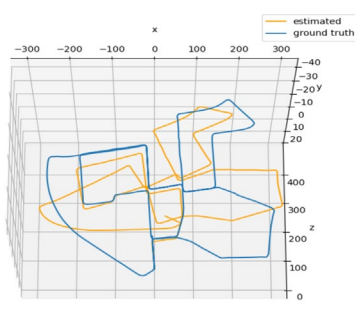
(c) Visual odometry computation using the following parameters-feature detector: SIFT, feature matching: FLANN, filter matching distance: 0.5, stereo matcher: SGBM.



(d) Visual odometry computation using the following parameters-feature detector: SIFT, feature matching: BF, filter matching distance: 0.5, stereo matcher: SGBM.



(e) Visual odometry computation using the following parameters-feature detector: SIFT, feature matching: FLANN, filter matching distance: 0.3, stereo matcher: SGBM.



(f) Visual odometry computation using the following parameters-feature detector: SIFT, feature matching: BF, filter matching distance: 0.3, stereo matcher: SGBM.

Fig. 8. Comparison of visual odometry trajectory using varying parameters of feature detector, feature matching, filter matching distance, and stereo matcher using PnP RANSAC on sequence 0 of the KITTI dataset. FLANN - Fast Approximate Nearest Neighbor Searches.

Stereo SGBM matching takes around three times longer to compute the disparity map when compared to BM matching but produces a much more contiguous map with minimum gaps in depth information.

C. Depth Computation

The projection matrix is decomposed from the dataset and the intrinsic matrix, rotation matrix and the 3D translation vector were obtained. This is eventually used to calculate the baseline between stereo pair of cameras, which is then used for calculating the depth map. Fig. 5. (b) shows the depth computation when BM matcher was used where there are plenty of gaps in the depth information as is evident from the figure. Fig. 5 (d) shows the depth computation when SGBM matcher was used where the depth information is a lot more consistent with less gaps in the depth information.

Fig. 7 shows the histogram analysis that was done to check the distribution of depths in the depth map. This information is

TABLE I. COMPARISON BETWEEN STEREO DEPTH AND LiDAR DEPTH ON SEQUENCE 0 OF THE KITTI DATASET IN METERS

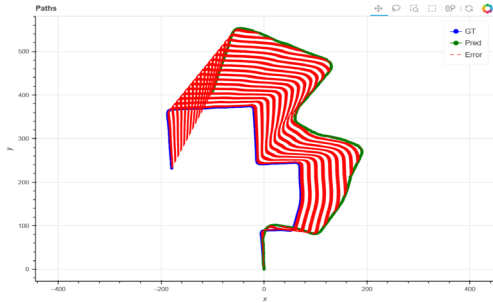
Stereo Depth	24.04	24.04	24.04	24.04	24.04	24.04	24.04
LiDAR Depth	25.08	25.08	25.08	25.08	25.08	25.08	25.08

used to extract relevant depth information from the depth map and discard noisy depth data.

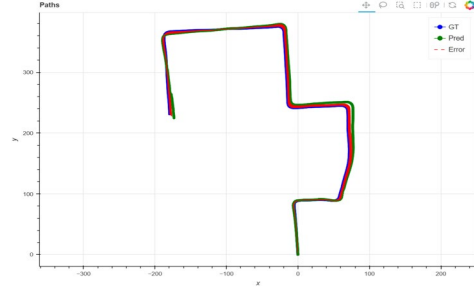
When the LiDAR data was considered with the camera depth information, some discrepancies were noticed for a few results but for most of the data it was found that the LiDAR and the camera depth information were close to each other. Table I shows the comparison between the camera depth and the LiDAR depth on a subset of pixels from the Sequence 0 of the KITTI dataset. LiDAR is a more reliable source of depth information but the primary areas of outlier for the stereo camera lie in the blurry regions or where the camera viewpoint angle does not cover the scene properly. Ignoring the outliers, the depth values between the LiDAR and the stereo were approximately within 10% of each other, hence in real world applications camera can be a cost saving option in comparison to the expensive LiDAR equipment.

D. Feature Detection and Matching

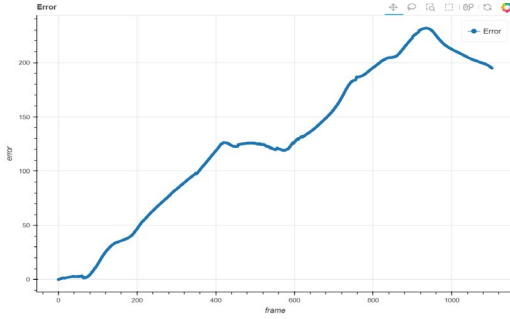
Feature detection and matching is the process of identifying distinct points or features in an image for purposes of matching with other images. The goal is to find correspondence between the features of two images. SIFT and ORB were used for feature detection and matching. The ratio test was applied to determine the uniqueness of SIFT features. Fig. 6 (a) shows feature detection and matching between two frames of sequence 0 of the KITTI dataset using the ORB feature detector and fig. 6 (b)



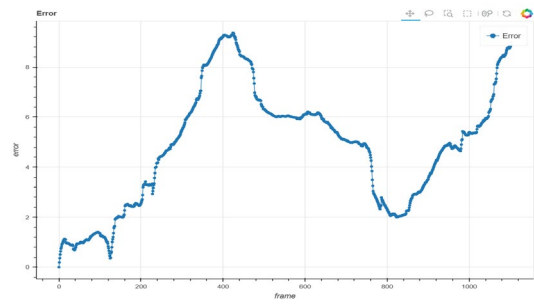
(a) Visual odometry using Lucas Kanade method without bundle adjustment and loop closure.



(b) Visual odometry using Lucas Kanade method with bundle adjustment and loop closure.



(c) Error accumulation while iterating over the frames of the sequence using Lucas Kanade method without bundle adjustment and loop closure.



(d) Error accumulation while iterating over the frames of the sequence when Lucas Kanade method was used with bundle adjustment and loop closure.

Fig. 9. Comparison between the visual odometry computation without bundle adjustment and loop detection in (a) and (c) and with bundle adjustment and loop detection in (b) and (d) on sequence 0 of the KITTI dataset. GT is ground truth, Pred is prediction.

shows the feature detection and matching using the SIFT feature detector and the ratio was set to 0.3.

In the case of ORB, the number of matches before filtering was 500 and after filtering the matches were reduced to 206. As is evident from figures 8 and 9, SIFT performed significantly better than ORB in our experiment. The number of matches before filtering for the SIFT feature detector was 3,206 whereas after applying the filtering method, the number of unique features were reduced to 446.

E. Motion Estimation

Motion estimation is the process of analyzing consecutive frames of the video sequence to determine the motion or movement of objects within a scene. Two approaches were taken for the motion estimation process. The PnP RANSAC involves estimating the pose of a camera given a set of 3D points in a scene and their corresponding 2D projections in the image. Another motion estimation process involved optical flow where the motion of each pixel is estimated. The displacement vectors between the pixels on sequence of two frames represent apparent

motion. The poses are optimized using sparse bundle adjustment that is used to minimize the reprojection errors.

Based on the motion estimation, the visual odometry was computed using several varying parameters to compare the results. Fig. 8 shows the visual odometry computation using the different evaluation parameters to gather conclusive evidence of which combination of parameters works best. Fig. 8 (f) shows results corresponding to the lowest error rate in Table II where SGBM matcher, brute force approach, and filter matching distance of 0.3 are used on SIFT features of the sequence 0 of KITTI dataset.

Table II shows the detailed comparison between the evaluation parameters and the corresponding evaluation results determined by root mean squared error (RMSE) and the mean absolute error (MAE). The time taken to perform visual odometry using Brute Force (BF) feature matcher was observed to be a little less than the FLANN feature matcher, certain frames were dropped when the FLANN matcher was used. While keeping the filter matching distance and the stereo

TABLE II. COMPARISON OF RESULTS ACROSS DIFFERENT EVALUATION PARAMETERS FOR VISUAL ODOMETRY COMPUTATION USING SIFT AND PNP RANSAC ON SEQUENCE 0 OF THE KITTI DATASET

Stereo Matcher Feature Matching Filter Matching Distance						
Error Metric	BM BF 0.5	BM FLANN 0.5	SGBM FLANN 0.5	SGBM BF 0.5	SGBM FLANN 0.3	SGBM BF 0.3
MAE	45.92	45.88	37.91	37.98	37.12	32.54
RMSE	58.00	57.87	47.26	47.37	46.14	39.99

Note: MAE: Mean Absolute Error; RMSE: Root Mean Squared Error; BM: Block-matching; SGBM: Semi-global Block-matching; BF: Brute Force; FLANN: Fast Library for Approximate Nearest Neighbors

matcher same, lower error rate was observed with BF when compared to FLANN particularly as observed in the last two columns of Table II. Furthermore, replacing the BM matcher with SGBM improved the performance. The key difference was observed when the filter matching distance was changed to 0.3. There was not much improvement noticed with the FLANN matching but when BF was used there was a significant improvement over the other combinations as is evident from the RMSE and MAE values in Table II.

Fig. 9 (a) shows the visual odometry computation using the Lucas Kanade method for optical flow and no bundle adjustment or loop closure was used to optimize the trajectory. All the experiments using the optical flow were conducted on 1,103 frames of the sequence 0 of the KITTI dataset. Fig. 9 (c) shows the error accumulation while iterating over the frames of the sequence. Fig. 9 (b) shows the visual odometry computation using the Lucas Kanade method with bundle adjustment and loop closure to optimize the trajectory. Fig. 9 (d) shows the error accumulation while iterating over the frames of the sequence when optical flow was used with bundle adjustment and loop closure. It is evident that bundle adjustment and loop closure when coupled with optical flow provided much more accurate and optimized trajectory.

VI. CONCLUSION AND FUTURE WORK

We proposed a stereo visual odometry pipeline and performed a comprehensive comparison of different trajectory computation approaches on the challenging KITTI dataset. Different feature detection methods, feature matching methods, filter matching distance, stereo matching along with different types of motion estimation, optimization techniques such as loop closure and bundle adjustment were tested in terms of error metric and computational time. SGBM matcher performed better than the BM matcher on depth and disparity estimation. SIFT was found to outperform the ORB feature detector. Brute Force feature matcher marginally outperformed FLANN matcher. The key component for the PnP RANSAC based motion estimation has been the filter matching distance that significantly improved the results. The Lucas Kanade method for optical flow coupled with bundle adjustment and loop closure were used to optimize visual odometry. It improved performance as the drift was reduced significantly. The work could be further extended to fuse other kinds of sensor data like inertial measurement units, LiDAR, and GPS to further reduce the disparity between the ground truth and estimated trajectory.

REFERENCES

- [1] J. Engel, V. Koltun and D. Cremers, "Direct sparse odometry," in *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, no. 3 (2017): 611-625.
- [2] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," in *2015 European Conference on Mobile Robots (ECMR)*, pp. 1-6. IEEE, 2015.
- [3] R. Gomez-Ojeda and J. Gonzalez-Jimenez, "Robust stereo visual odometry through a probabilistic combination of points and line segments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2521-2526. IEEE, 2016.
- [4] T. Pire, T. Fischer, G. Castro, P. D. Cristóforis, J. Civera and J. J. Berles, "S-PTAM: Stereo parallel tracking and mapping," in *Robotics and Autonomous Systems* 93 (2017): 27-42.
- [5] A. Rosinol, M. Abate, Y. Chang and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1689-1696. IEEE, 2020.
- [6] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," in *IEEE Transactions on Robotics* 31, no. 5 (2015): 1147-1163.
- [7] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," in *IEEE Transactions on Robotics* 33, no. 5 (2017): 1255-1262.
- [8] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam," in *IEEE Transactions on Robotics* 37, no. 6 (2021): 1874-1890.
- [9] J. Engel, J. Stückler and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1935-1942. IEEE, 2015.
- [10] J. Engel, T. Schöps and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*, pp. 834-849. Cham: Springer International Publishing, 2014.
- [11] "Getting Started with ROS 2 and ZED," StereoLabs, 2023. [Online]. Available: <https://www.stereolabs.com/docs/ros2>.
- [12] C. Forster, M. Pizzoli and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15-22. IEEE, 2014.
- [13] C. Forster, Z. Zhang, M. Gassner, M. Werlberger and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," in *IEEE Transactions on Robotics* 33, no. 2 (2016): 249-265.
- [14] R. Wang, M. Schworer and D. Cremer, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3903-3911. 2017.
- [15] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," in *The International Journal of Robotics Research* 34, no. 3 (2015): 314-334.
- [16] M. Bloesch, S. Omari, M. Hutter and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 298-304. IEEE, 2015.
- [17] T. Qin, P. Li and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," in *IEEE Transactions on Robotics* 34, no. 4 (2018): 1004-1020.
- [18] P. Geneva, K. Eickenhoff, W. Lee, Y. Yang and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4666-4672. IEEE, 2020.
- [19] Y. Liao, J. Xie and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d," in *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, no. 3 (2022): 3292-3310.
- [20] L. David, "Distinctive image features from scale-invariant keypoints," in *International Journal of Computer Vision* 60 (2004): 91-110.
- [21] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, pp. 2564-2571. IEEE, 2011.
- [22] S. Huang, G. Sun and M. Li, "FAST and FLANN for feature matching based on SURF," in *2021 33rd Chinese Control and Decision Conference (CCDC)*, pp. 1584-1589. IEEE, 2021.