The Impact of the Image Feature Detector and Descriptor Choice on Visual Odometry Accuracy

Chunkai Yao[1], Danish Syed[1], Joseph Kim[2], Peerayos Pongsachai[2], and Teerachart Soratana[3]

***Abstract*—Building a fully autonomous mobile robotic system is a difficult task, which requires accurate sensors and techniques for localization and mapping. One such techniques is Visual Odometry (VO), which uses stereo or monocular camera sensors to estimate the poses of a vehicle. The goal of this project was to implement VO and to analyze the accuracy of the localization when using different combinations of descriptor and detector. Simulation results were obtained by analyzing 10 KITTI dataset sequences, with Relative Pose Error (RPE), Absolute Trajectory Error (ATE) and runtime in term of frames per second (FPS) as evaluation metrics. Comparison of RPE and ATE for each combination of descriptor and detector were obtained as heatmap. Outliers in RPE and error accumulation in ATE were discussed, and future work were suggested for conclusive analysis as benchmark. Code of our report is publicly available at https://github.com/dysdsyd/VO_benchmark.**

***Index Terms*—Visual Odometry, SLAM, Mobile robotics, Computer vision, Descriptors, Detectors, Convolution Neural Network**

I. Introduction

Significant progress has been achieved in the area of mobile robotics. Advancement in both hardware and software have made such system a safe and reliable technology. For such systems, making robots understand the world and recognize / distinguish what they see has become an important research area. Furthermore, localization and mapping have become a central technology in the perception and state estimation of mobile robotics. In this paper, we recapped relative techniques used in monocular visual odometry (VO) to determine the position and orientation of a vehicle with pre-existing KITTI dataset [1]. Then, we implemented and compared different feature descriptors to compare their performances using Relative Pose Error (RPE), Absolute Trajectory Error (ATE) and runtime.

The structure of this paper is organized as follows: section II discuss existing techniques in VO, and detector and descriptor algorithms. Section III discuss the details of techniques we implemented for performance evaluation. Section IV discuss about the results of the evaluation. Section V summarize the conclusions of the performance evaluation and discussed what we can improve in the future.

[1] Chunkai Yao and Danish Syed are with Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, USA nickyao@umich.edu and dasyed@umich.edu

[2] Joseph Kim and Peerayos Pongsachai are with Robotics Institute, University of Michigan, Ann Arbor, USA jthkim@umich.edu and ppongsa@umich.edu

[3] Teerachart Soratana is with Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, USA tsorat@umich.edu

II. Background

For a robot to autonomously track its path, and detect / avoid any obstacles, localization is the fundamental technique for the pose estimation. Below are the descriptions of VO, and of detection and descriptor algorithms for data acquisition and transformation estimation.

A. Visual Odometry (VO)

VO is a pose estimation on a moving agent with camera video inputs [2]. By analyzing sequence of images in camera, VO incrementally estimates the position (translation and rotation with respect to a reference frame) of a vehicle. The idea first came out in the early 1980s [3], and VO was extensively researched in NASA, preparing Mars Mission in 2004 [4], [5]. They found that VO is an inexpensive application with some of distinct advantages like functionality in GPS-denied environment, no slippage effect in uneven terrain, lightweight and simple to integrate with other computer vision based algorithms. Thus, VO has become an alternative to conventional localization method such as wheel odometry, GPS, INS, sonar localization, etc [5]. Particularly, VO is one of the robust techniques to do such localization, known to have a relative position error of 0.1 to 2% [5]. To provide the comparison, Table I below briefly summarizes pros and cons of different localization techniques.

Furthermore, it is noteworthy to distinguish the difference between VO and Simultaneous Localization and Mapping (SLAM). In VO, primary objective is to estimate the pose of the vehicle by incrementally taking the camera input and by maintaining local consistency between the sequential frames. In contract, the objective of SLAM is to update the unknown map environment incrementally, and use this updated map to estimate the pose of vehicle in the newly acquired map data. In this sense, SLAM maintains globally consistent estimation of the trajectory. Figure 1 below illustrates the high-level overview of VO and SLAM.

VO is broadly classified as two categories (i.e. stereo vision, monocular vision) based on the types of camera used in position estimation. In our interest of VO with image feature detectors, we restrict our scope to the monocular VO, but the generality is not lost. However, it is worth noting the main performance difference between stereo and monocular cameras used in VO. In stereo camera, depth information and image scale is relatively easy to obtain, but requires more calibration effort than monocular cameras, and also can be more costly and difficult in terms of interfacing. In contrast, monocular camera is low cost, low weight, and simpler to implement,

TABLE I
COMPARISON OF DIFFERENT LOCALIZATION TECHNIQUES

Techniques	Pros	Cons
Global Positioning System (GPS)	Error does not accumulate over time	Cannot operate indoor or underwater
Wheel Odometry	Cheap and simple way to estimate position / orientation	error accumulates over time
Laser	High accuracy and high scan rate	Expensive, and subject to reflection in object surface
Visual Sensor Based Odometry	High accuracy and low cost	Relatively high computational cost

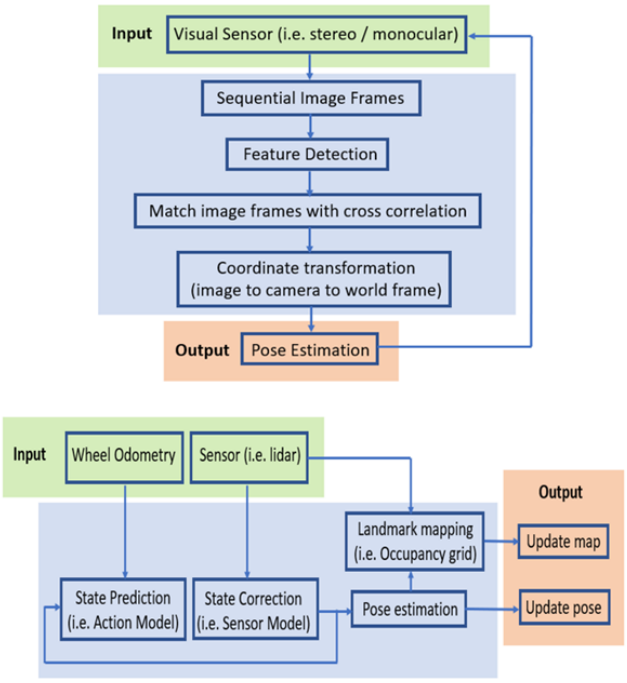

Fig. 1. Schematics of VO (top) and SLAM (bottom).

requiring less effort in calibration. Monocular camera is particularly suitable for micro/small robots. However, monocular vision has a drawback of uncertainty in image scaling.

B. Detector and Descriptor Algorithms

In traditional visual SLAM or VO systems, visual data association tools are used to keep track of objects within key frames for camera localization. These tracking tools consist of a detector to identify keypoints, a descriptor to characterize the keypoints and mathematically represent the features, and a matching algorithm to compare found features of different frames [6].

Detection algorithms identify keypoints in the image and keep track of these features between keyframes. This method of data association and tracking takes significant computational power if done for every frames. To fix this issue, detector algorithms often utilize keyframe method of tracking data for some selected frames and disregard the rest to ease computational burden [2]. Typical detector tools like the ORB created short-term and mid-term data association of features in the keyframes by creating association map of the tracked elements. This association is done on pixel level in order to track the visual elements over time [7].

A common feature tracking algorithm, Shi-Tomasi [8], was one of the detector algorithms used in this project. Shi-Tomasi tracks features in camera frames based on dissimilarity of the image's affine changes. The algorithm calculates the feature's affine transformation using Newton-Raphson stile minimization procedure and determine a dissimilarity between the previous and current frame to disregard features with high dissimilarity. Additionally, the algorithm selects features that optimize the detector's accuracy to track, thereby maximizing the detector's accuracy [8].

Descriptor algorithms extract a vector attribute that represent the interested pixel extracted by the detector. The association of features by descriptors enable feature transformation between frames to be determined [6]. For many descriptors, this process is done on pixel level and using mathematically models. As such, many descriptor algorithms are computationally intensive.

Instead of describing the features on pixel-levels, some descriptors utilize neural network methods as an alternative method of classifying and tracking visual elements [7]. Unlike traditional tracking tools, neural network based algorithms are not susceptible to error accumulation that is the result of keyframe tracking. Neural network tools can identify visual elements based on trained data and track these elements over time, allowing for mid-term and long-term data association [7].

III. METHODOLOGY

A. Visual Odometry

The project was based on the pySLAM v2 implementation [9]. In monocular VO pose estimation, feature points were sequentially detected at least in three consecutive frames in order to observe / confirm the features, and then calculated the transformation (rotation and translation) between the frames. The procedure of monocular VO with three minimum consecutive frame analysis in pySLAM v2 is described below:

- Select number of features to detect and track
- Choose feature tracker configurations (i.e. SIFT, SUPERPOINT, BRISK, etc)
- Create VO object with camera calibration, ground truth data and feature tracker
- Process first image by extracting regional / augmented features, and keypoints
- Process consecutive frames

- Track / extract features by using specified detector and descriptors, and match the keypoints
- Estimate the relative poses among three consecutive frames by using five-point algorithm [10] (i.e. estimate the Essential matrix in normalized image coordinates), and use RANSAC to refine the matches, and estimate the transformation.
- calculate current translation scale from ground-truth to obtain inter-frame scale
- Update the history: reference / current keypoints and descriptors
- Repeat the process for every iteration

B. Detector and Descriptor Selection

In this project, we conducted a study with descriptor and detector combinations as our independent variable. There were 30 combinations between detectors and descriptors in total.

1) Group 1: Common detector-descriptors: Based on the literature review on visual odometry, we chose 5 common detectors and 5 common descriptors and assigned them to group 1. In this group, all of the detectors and descriptors were matched together to form a detectors-descriptors pair, except for SIFT-ORB2 pair which did not run in the code. 5 detectors are: BRISK [11], FAST [12], ORB2 [13], SHI-TOMASI [8], and SIFT [14]. 5 descriptors are: BRISK [11], ORB2 [13], SIFT [14], TFEAT [7], and no descriptor (using optical flow method such as Lucas-Kanade (LK) [15]). A total of 24 combinations belong to this group. Note that the ORB2 is the detector and descriptor from ORB-SLAM2 implementation [13] which used a different Bag-of-Words implementation comparing to ORB features [16].

2) Group 2: ORB2 with deep-learning descriptors: In the second group, we want to evaluate the performance of the method that uses ORB2 as detector and neural network algorithm to extract the features. We chose HARDNET [17], L2NET [18], SOSNET [19], and VGG [20] as the descriptors. A total of 4 combinations belong to this group.

3) Group 3: End-to-end deep-learning VO: In the third group, we run the evaluation on a few end-to-end neural network methods. End-to-end method refers to algorithm that uses its default pipeline to function as both detector and descriptor. In our evaluation, we have SUPERPOINT [21] and D2NET [22] as the end-to-end method. These methods were chosen because both were designed to find correspondence between two frames, and thus can function as both detector and descriptor. A total of 2 combinations belong to this group.

C. Evaluation Metrics

We obtained estimated trajectory: $P_1, \ldots, P_n \in SE(3)$, ground truth trajectory: $Q_1, \ldots, Q_n \in SE(3)$, and fixed time interval Δ. We introduced ATE and RPE for evaluation [23].

1) ATE: To calculate ATE, the first step is to find the rigid-body transformation S mapping the estimated trajectory $P_{1:n}$ onto the ground truth trajectory $Q_{1:n}$. We calculated ATE by:

$$F_i := {Q_i}^{-1} S P_i \tag{1}$$

2) RPE: We defined the relative pose error at time step i as:

$$E_i := ({Q_i}^{-1} Q_{i+\Delta})^{-1} ({P_i}^{-1} P_{i+\Delta}) \tag{2}$$

For both errors, we computed the root mean squared error (RMSE) to evaluate the overall performances of different feature combinations we tested. Cumulative RMSE is computed by

$$RMSE = \sqrt{\frac{\sum_{j=1}^{J} \sum_{i=1}^{N_j} (x_{pij} - x_{oij})^2}{\sum_{j=1}^{J} N_j}} \tag{3}$$

Here, we have x_p denotes the predicted value and x_o as observed value (i.e. ground truth). j denotes each KITTI sequences, N_j denotes the number of observation for the j^{th} KITTI sequence, and i denotes each predicted frame. Pooled standard deviation is computed using equation 4

$$sd_{pooled} = \sqrt{\frac{\sum_{j=1}^{J} (N_j - 1) sd_j^2}{\sum_{j=1}^{J} (N_j - 1)}} \tag{4}$$

3) Runtime: Runtime was evaluated by running each detector-descriptor combinations on KITTI dataset sequence 00. This sequence contains 4541 images, and each image is a gray scale image of size 1241 X 376 pixel. The average frame per second (FPS) and its standard deviation were used as runtime metric.

D. Implementation and Running

In practice, we tested on 30 different detector-descriptor combinations. For each test, we used the same detector-descriptor combination for estimating pose and trajectory on 10 different sequences. This strategy was chosen to eliminate the effect that one combination worked particularly well or bad for one specific dataset and that performance was not representative.

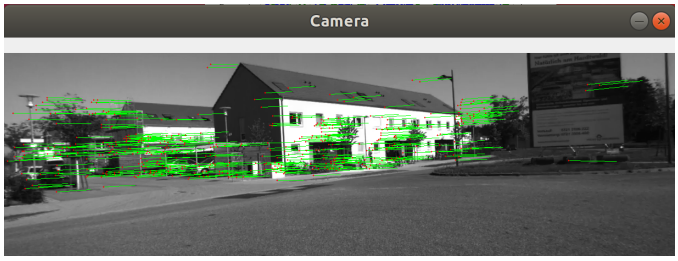

Fig. 2. Video output during VO evaluation

In Figure 2, during the evaluation process, the evaluation software matches the interest points detected from two images of the same scene. The green lines visualize the transformations between matched features.

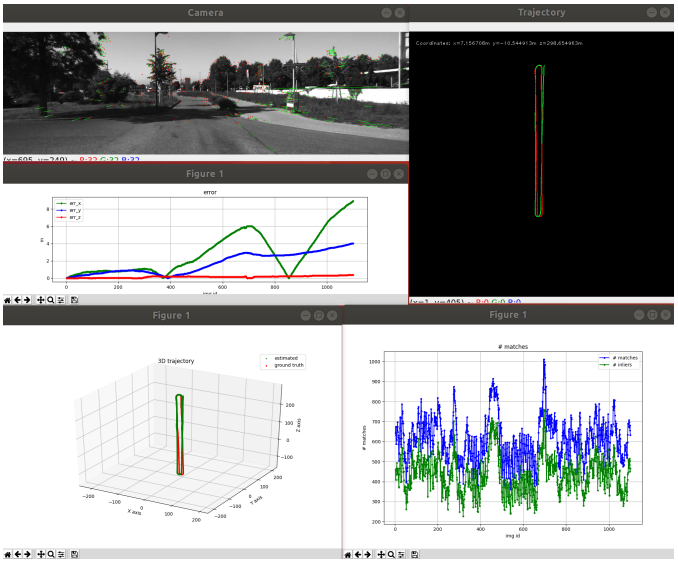

Fig. 3. Combined output during evaluation. It shows real-time trajectory visualization in 2-D (top-right) and 3-D (bottom-left), translation error (center-left) and number of matches (blue) and inliers (green) (bottom-right).

While we were performing the evaluation, we output the plots in Figure 3 to monitor the performance of our VO. The windows in top-right and bottom-left provided real-time trajectory visualization in both 2-D and 3-D. The window in the center showed runtime errors for x, y and z axis. Also, the window in the bottom-right indicated the number of matches and inliers between each consecutive frame.

In order to match the format we want for evaluation, we converted rotation and translation matrix into $SE(3)$. We took estimated trajectory in $SE(3)$ and ground truth as inputs to compute the ATE and RPE. Our evaluations relied on an open source evaluation tool named evo [24]. RPE and ATE were then computed for all 30 combinations.

IV. Results

A. RPE

The results for RPE indicated that SIFT-SIFT performed best out of all other combinations, followed by BRISK-BRISK pair. High standard deviation in RPE indicated that there were many outliers. Out of 300 evaluations (from 10 sequences for each of the 30 combinations), 291 (97%) of the evaluations has the mean higher than the median, which indicated that the normal behavior of RPE in VO would generally contains occasional high error frames. Out of the 9 evaluations with mean lower than the median, 7 of them was from KITTI dataset sequence 01, which is a free way drive scenario.

In KITTI dataset sequence 01, out of 30 combinations between detector and descriptor evaluated, 27 (90%) of the combinations has the highest RPE in this sequence out of all other KITTI sequences in 00 - 09. This might be due to its free way driving scenario where there are little fiducials on the road side comparing to other scenario, which makes it challenging to perform VO. There are also moving cars in the same direction at the similar speed, or in the opposite direction. SIFT-SIFT pair performed the best in this scenario compared to all other combinations, followed by BRISK-BRISK.

Our results also indicated that using optical flow method (specifically LK method) in place of a descriptor yield consistently good results in FAST, ORB2, and SHI-TOMASI detector. As for ORB2 detector with deep-learning descriptor, we found that L2NET and SOSNET paired with ORB2 performs relatively better than with HARDNET and VGG with ORB2. End-to-end deep-learning performed relatively well comparing to ORB2 with deep-learning.

The estimated trajectory plots of SIFT-SIFT combination and of SHI-SIFT combination with RPE value at particular pose are shown in Figure 6 and 8 respectively. The colored bars on the right side of the plots are the labels of the RPE values at each pose. As shown in the trajectory plots, the SIFT-SIFT combination performs VO very well and the SHI-SIFT combination has a terrible performance.

The plots of RPE for each frame, shown in Figure 7 and 9 for SIFT-SIFT combination and SHI-SIFT combination respectively, show that there are many outliers in the RPE values. Figures 7 and 9 show the outliers as sudden spikes in the RPE values. Figure 7 shows relatively small number of outliers, illustrating the reason for low pooled standard deviation value for the SIFT-SIFT combination. On the other hand, Figure 9 shows large number of outliers causing the RMS value of RPE and value of pooled standard deviation to be high for SHI-SIFT case.

In order to evaluate the data and investigate the cause of spikes in RPE plots, we looked at the real-time plot of number of matches and inliers which are the real-time output of pySLAM code. Figure 3 shows an example of the output of the pySLAM code with the plot of pose error, estimated trajectory, and plot of number of matches and inliers for each frame. We attempted to find correlation between the number of matches and inliers, and the value of RPE at the same frame. We identified the frame that resulted in sudden changes in the RPE value and tried to observe if there are any significant changes in the number of matches and inliers at that frame. However, we did not find any correlation between number of matches and inliers, and the sudden increases in RPE. We observed that there are certain frames in the KITTI data that causes the sudden increases in RPE across many detector-descriptor combinations. Even so, we cannot make any conclusive remarks regarding the cause of the sudden spikes in RPE values observed in Figure 7 and 9.

B. ATE

The ATE value is the lowest for SIFT-SIFT combination followed by BRISK-BRISK combination, which is consistent with the results for RPE. The combination with the highest ATE is the SHI-SIFT combination as seen in Figure 10. Furthermore, the error values for Group 3, end-to-end deep-learning tools, are lower than error of Group 2, ORB2 with deep-learning features. The result indicated that using LK in place of a descriptor yield relatively low ATE values in SIFT,

	Group 1				
Descriptor	BRISK	NONE	ORB2	SIFT	TFEAT
Detector					
BRISK	0.151	0.229	0.183	0.160	0.174
FAST	0.457	0.200	0.373	0.745	0.486
ORB2	0.684	0.188	0.619	0.622	0.654
SHI	0.530	0.212	0.347	0.987	0.539
SIFT	0.190	0.203		0.106	0.160

Group 2	
	ORB2
HARDNET	0.773
L2NET	0.549
SOSNET	0.516
VGG	0.623

Group 3	
D2NET	0.304
SUPERPOINT	0.299

Fig. 4. Heatmap of Relative Pose Error (RPE) across different combinations of detectors and descriptors. (Green/ orange / red indicates low / medium / high error). Lower error means better performance.

	Group 1				
Descriptor	BRISK	NONE	ORB2	SIFT	TFEAT
Detector					
BRISK	0.146	0.218	0.176	0.154	0.168
FAST	0.371	0.190	0.324	0.514	0.376
ORB2	0.493	0.182	0.468	0.452	0.501
SHI	0.443	0.202	0.314	0.680	0.432
SIFT	0.184	0.194		0.103	0.156

Group 2	
	ORB2
HARDNET	0.432
L2NET	0.413
SOSNET	0.488
VGG	0.557

Group 3	
D2NET	0.277
SUPERPOINT	0.275

Fig. 5. Heatmap of Relative Pose Error (RPE) standard deviation across different combinations of detectors and descriptors. (Green/ orange / red indicates low / medium / high error). Lower error means better performance.

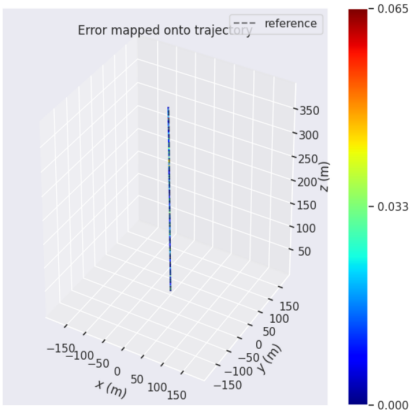

Fig. 6. Trajectory Plot with Relative Pose Error (RPE) of SIFT-SIFT in KITTI dataset 04

ORB2, and SHI-TOMASI detectors. Moreover, L2NET and SOSNET paired with ORB2 yields relatively lower error than HARDNET and VGG with ORB2.

Comparing the trajectory plots with the ATE results for each combination confirms that lower ATE values indicate better VO performance. The combination with the lowest ATE, SIFT-SIFT, has a trajectory plot with very low deviation from the ground truth, as seen in Figure 7. Likewise, the combination with highest ATE, SHI-SIFT, has a trajectory plot that greatly deviated from the ground truth, seen in Figure 9. Since low ATE values suggest better VO performance, the result seems to indicate that SIFT-SIFT combination performed the best at VO compared to the other chosen combinations. The ATE values of SIFT-SIFT and BRISK-BRISK combinations are

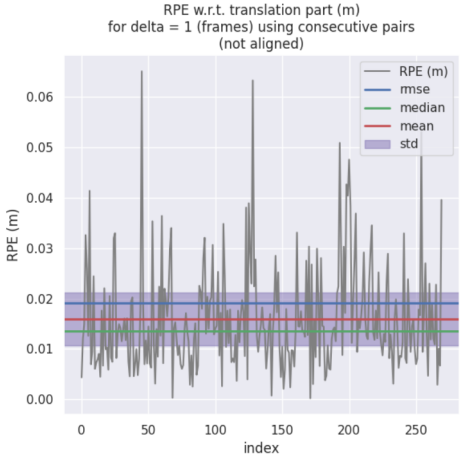

Fig. 7. Relative Pose Error (RPE) of SIFT-SIFT in KITTI dataset 04

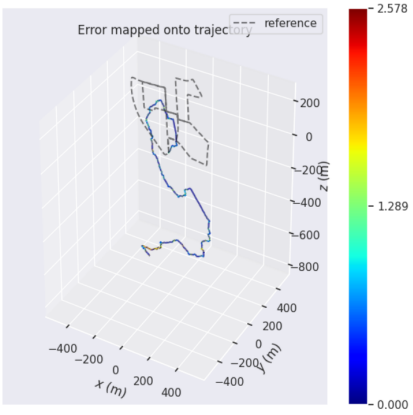

Fig. 8. Trajectory Plot with Relative Pose Error (RPE) of SHI-SIFT in KITTI dataset 00

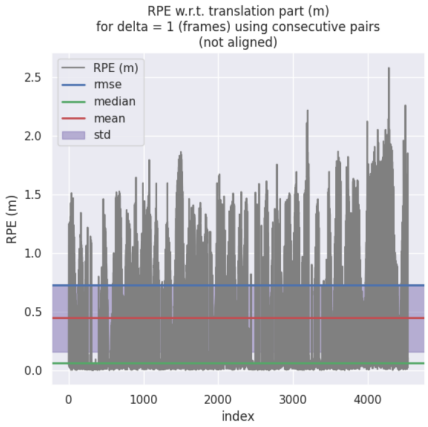

Fig. 9. Relative Pose Error (RPE) of SIFT-SIFT in KITTI dataset 04

lower than error values of deep-learning tools like D2NET and SUPERPOINT, suggesting that these combinations have better VO performance than deep-learning tools.

Figure 11 shows that the standard deviation values of the

ATE are high relative to the cumulative root mean square values of ATE. The reason for high relative standard deviation was due to the error accumulating over time. Figure 12 shows how the ATE value increases for subsequent frames, resulting in high standard deviation for each trajectory. As such, the pooled standard deviation of each combination across the 10 trajectories is relatively high and is close to the cumulative root mean square value of ATE.

	Group 1				
Descriptor	BRISK	NONE	ORB2	SIFT	TFEAT
Detector					
BRISK	20.33	129.19	26.92	24.51	26.24
FAST	127.75	116.36	86.10	289.82	141.14
ORB2	328.03	28.50	311.02	275.75	353.81
SHI	355.82	51.77	155.10	478.01	300.87
SIFT	42.64	62.92		15.48	22.24

Group 2	
	ORB2
HARDNET	262.70
L2NET	252.57
SOSNET	421.67
VGG	375.42

Group 3	
D2NET	113.08
SUPERPOINT	73.31

Fig. 10. Heatmap of Absolute Trajectory Error (ATE) across different combinations of detectors and descriptors. (Green/ orange / red indicates low / medium / high error). Lower error means better performance.

	Group 1				
Descriptor	BRISK	NONE	ORB2	SIFT	TFEAT
Detector					
BRISK	11.53	90.33	15.92	14.25	15.22
FAST	74.64	75.40	49.86	161.36	83.13
ORB2	190.56	17.84	180.70	153.43	196.01
SHI	197.30	31.69	91.56	266.22	166.32
SIFT	32.05	38.36		8.73	12.76

Group 2	
	ORB2
HARDNET	150.67
L2NET	147.80
SOSNET	231.83
VGG	216.86

Group 3	
D2NET	58.07
SUPERPOINT	42.19

Fig. 11. Heatmap of Absolute Trajectory Error (ATE) standard deviation across different combinations of detectors and descriptors. (Green/ orange / red indicates low / medium / high error). Lower error means better performance.

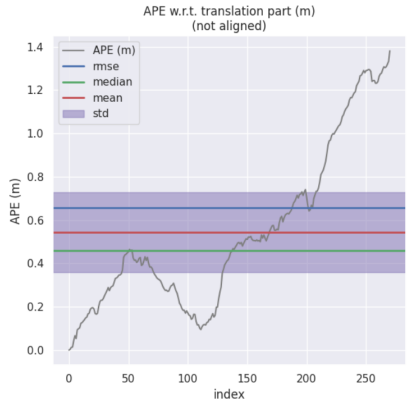

Fig. 12. Absolute Trajectory Error (ATE) of SIFT-SIFT in KITTI dataset 04

C. Runtime Metrics

The results of runtime calculation indicated that BRISK-BRISK combination has the best average FPS with its standard deviation. Also, neural network based descriptor-detector combinations performed worse compared to classical ones. This matches our expectation as neural networks takes significantly more computation time and resources. As for the descriptor, using LK algorithm yield overall high frame rate in group 1.

	Group 1				
Descriptor	BRISK	NONE	ORB2	SIFT	TFEAT
Detector					
BRISK	8.22	9.89	7.83	4.16	6.11
FAST	4.92	12.35	4.99	2.14	3.93
ORB2	5.09	5.81	5.22	1.04	3.44
SHI	4.50	9.78	4.78	2.35	4.28
SIFT	4.89	3.78		5.10	4.96

Group 2	
	ORB2
HARDNET	2.66
L2NET	2.63
SOSNET	2.69
VGG	0.91

Group 3	
D2NET	1.57
SUPERPOINT	4.34

Fig. 13. Heatmap of Average FPS across different combinations of detectors and descriptors. (Green/ orange / red indicates High / medium / low FPS). Higher FPS means better performance.

	Group 1				
Descriptor	BRISK	NONE	ORB2	SIFT	TFEAT
Detector					
BRISK	3.08	3.97	2.66	1.50	2.22
FAST	2.72	5.60	2.48	0.76	1.56
ORB2	3.35	2.15	3.65	0.21	1.58
SHI	2.53	4.07	2.74	0.94	1.88
SIFT	0.82	1.03		1.22	0.99

Group 2	
	ORB2
HARDNET	0.96
L2NET	0.84
SOSNET	0.83
VGG	0.17

Group 3	
D2NET	0.11
SUPERPOINT	2.09

Fig. 14. Heatmap of FPS standard deviation across different combinations of detectors and descriptors. (Green/ orange / red indicates low / medium / high standard deviation). Lower standard deviation means consistent performance.

V. Conclusions

The result demonstrates that SIFT-SIFT combination, followed by BRISK-BRISK, performed the best at visual odometry compared to other chosen combinations and end-to-end deep-learning tools. Superpoint performs better than deep-learning descriptor combined with ORB2 based on our choice of deep-learning descriptor-detector. The SHI-SIFT combination had the worst performance out of the chosen combinations.

Taking runtime into account, the BRISK-BRISK combination is the best out of the chosen combinations. While SIFT-SIFT performance is the best, its runtime makes it unappealing for live implementation of VO. Additionally, our implementation of ORB2 using pySLAM had relatively slower frame rate compared to the frame rate of the BRISK detector.

The project was based on the pySLAM in python. We simplified the implementation for our visual odometry. In the future, we can explore more features of slam and visual odometry by comparing our results to more sophisticated ones like stereo VO, Omnidirectional VO [25], and C++-based ORB-SLAM. Our current result shows generally classical computer vision features do a better job than deep-learning features. It is possible to introduce more features like CenterNet [26] and check if more recent deep-learning feature combinations could have better results or not. Additionally, the experiment could be run using other dataset to confirm that the result is valid for any dataset. Lastly, loop closer could be implemented for more accurate pose estimation.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] 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 multi-map slam," *arXiv preprint arXiv:2007.11898*, 2020.

[3] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover." Stanford Univ CA Dept of Computer Science, Tech. Rep., 1980.

[4] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. Ieee, 2004, pp. I–I.

[5] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[6] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[7] R. Kang, J. Shi, X. Li, Y. Liu, and X. Liu, "Df-slam: A deep-learning enhanced visual slam system based on deep local features," 2019.

[8] Jianbo Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[9] L. Freda and M. Pierenkemper, "pyslam v2," https://github.com/luigifreda/pyslam, 2019.

[10] H. Li and R. Hartley, "Five-point motion estimation made easy," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 1. IEEE, 2006, pp. 630–633.

[11] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[12] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2019.

[13] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[14] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[15] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." Vancouver, British Columbia, 1981.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[17] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," *arXiv preprint arXiv:1705.10872*, 2017.

[18] Y. Tian, B. Fan, and F. Wu, "L2-net: Deep learning of discriminative patch descriptor in euclidean space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 661–669.

[19] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, "Sosnet: Second order similarity regularization for local descriptor learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 016–11 025.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman, "Learning local feature descriptors using convex optimisation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1573–1585, 2014.

[21] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[22] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint detection and description of local features," *arXiv preprint arXiv:1905.03561*, 2019.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.

[24] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[25] P. Corke, D. Strelow, and S. Singh, "Omnidirectional visual odometry for a planetary rover," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 4. IEEE, 2004, pp. 4007–4012.

[26] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019.