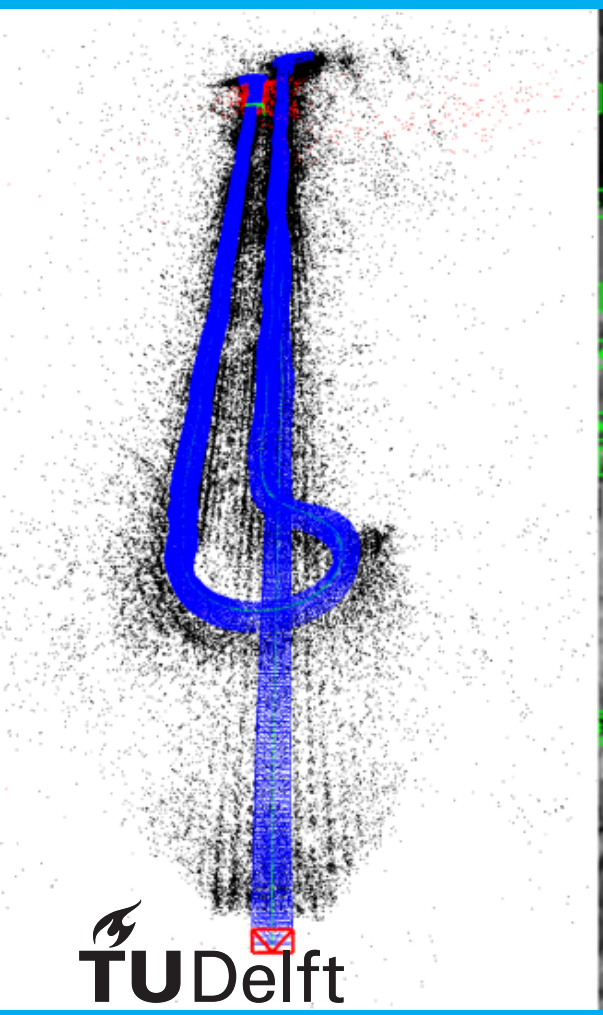
On the robustness of ORB matching in feature-based SLAM

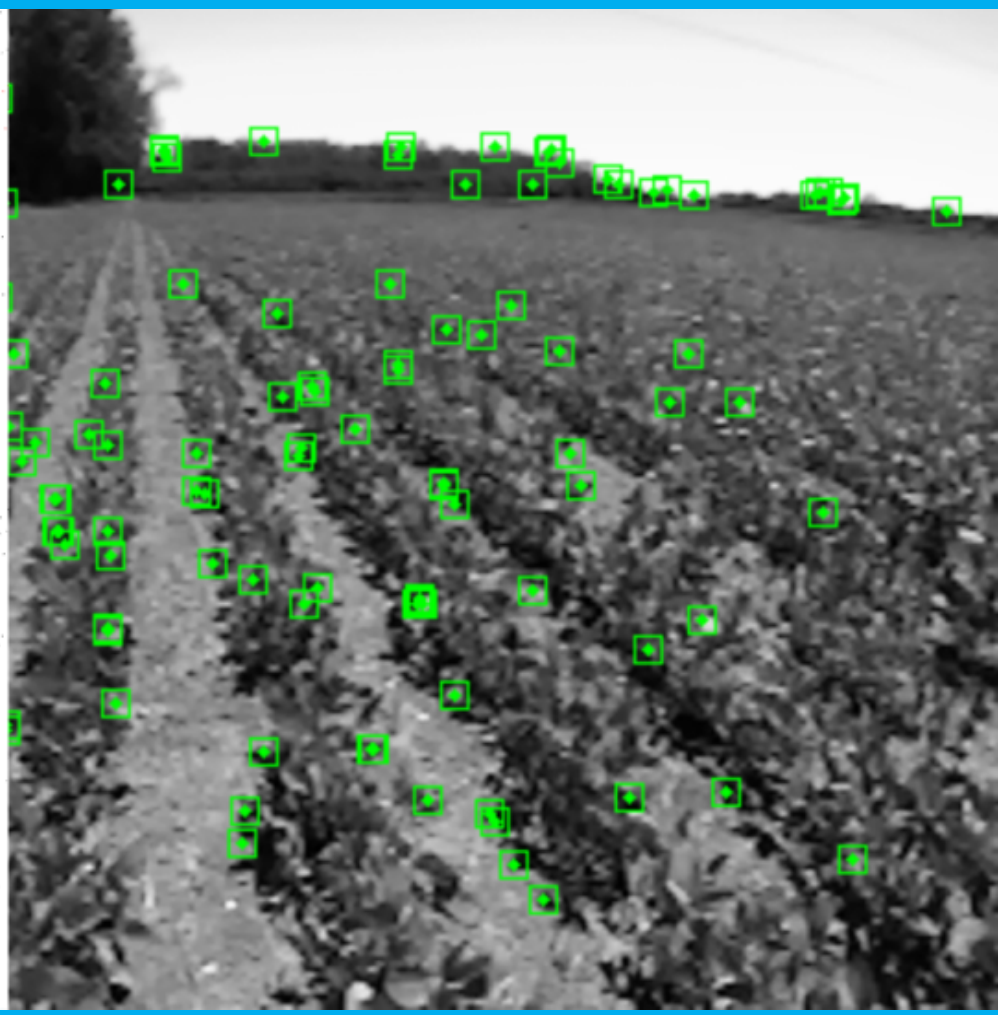
M. Coroz

Student number: 4430565

Supervisors: Dr. J.F.P. Kooij & Dr. Y. Li

Date: 28-11-2022





TU Delft

On the robustness of ORB matching in feature-based SLAM

by

M. Coroz

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday November 28, 2022 at 11:00 AM.

Student number:	4430565	
Project duration:	March 1, 2022 – November 30, 2022	
Thesis committee:	Dr. J.F.P. Kooij,	TU Delft, supervisor
	Dr. Y. Li,	Lely Technologies, supervisor
	Dr.ir. Y.B. Eisma,	TUDelft

This thesis is confidential and cannot be made public until November 28, 2024.

TU Delft

Abstract

A robotic vehicle must continuously determine its position within the map to traverse a path safely; this is called self-localization. Current localization methods use mainly sensors like LIDARS. However, a LIDAR does not return data points if the environment is an empty field; the laser scan of the LIDAR does not reflect without obstacles, making self-localization in those environments impossible. Self-localization with visual information from cameras can be an alternative for vehicles operating in an open environment, such as the Lely Juno, an agricultural robot operating on farms. A widely used visual Simultaneous Localization and Mapping (SLAM) method in literature, ORB-SLAM3, lacks robustness on Lely Juno's visual data; the track is lost multiple times due to a sudden drop in the number of feature matches.

This work aims to experimentally determine (a) the cause of the drop in the feature matches that eventually causes the tracking error and (b) the robustness and accuracy of ORB-SLAM3 compared to a simple visual odometry (VO) system. The cause of the drop in feature matches is investigated outside the ORB-SLAM3 pipeline by replicating the matching thread of ORB-SLAM3 with OpenCV ORB feature matching. The robustness and accuracy of ORB-SLAM3 are compared with a simple visual odometry system by comparing the trajectories and the minimum number of matches found in the sequences.

The sudden drops in feature matches could not be replicated outside the ORB-SLAM3 pipeline with the brute-force-based ORB matcher in OpenCV. While in some cases, the decrease in feature matches is related to the image itself (e.g., blur, movement), the leading cause is complicated. In this work, the leading cause is further reduced to the quality of the disparity map, the type of matcher used, and the use of grids when extracting features. The simple VO system is more robust than ORB-SLAM3. However, the absolute pose error is worse and unsuitable for reliable navigation for long trajectories.

For the Lely Juno, a simple visual odometry system can only be used for short distances outside. Therefore, investigating the issues in ORB-SLAM3 is an excellent direction from the project's point of view. More understanding is needed of the exact effects of the disparity map, the type of matcher used, and the use of grids on the number of feature matches.

The contributions of this work are insights into the reasons behind the tracking errors, exploring the effect of different ORB parameters and datasets on the matching performance, and creating two new datasets in low-textured and repetitive agricultural environments. Investigating the exact impact of the disparity map, the type of matcher used, and the use of grids on the number of feature matches is left for future work.

Preface

After spending seven years at the TU in Delft, my time as a student is ending. My journey started with aerospace engineering. I soon discovered that aerospace was not the degree I wanted to study for the coming few years. I ended up pursuing mechanical engineering for my bachelor's. During this time, I developed an interest in software development and autonomous vehicles. This interest led me to the master's in robotics, which I am completing with this final work.

I want to thank my supervisors, Julian Kooij and Yan Li, for their support and guidance during the final phase of my studies. I also would like to thank the ORB-SLAM3 community for answering my questions about the code, which has helped me understand the pipeline better.

Finally, I would like to thank my friends and family that supported me during my time as a student and everyone that took the time to proofread my work.

M. Coroz
Delft, November 2022

Contents

1 Introduction

Localizing a robotic vehicle is essential to perform tasks without human intervention safely. The vehicle needs to know its position on a map to follow a planned route and perform tasks.

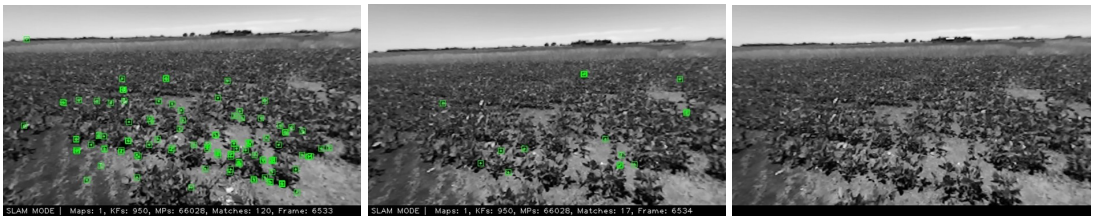
Current low-cost sensors that generate a point cloud, such as a 2D-LIDAR, do not provide data in an open outdoor environment as there are no objects that reflect the laser rays on the vehicle. A Global Positioning System (GPS) solely is not accurate enough to use for localization and is not able to provide data indoors without setting up beacons. An Inertial Navigation System (INS) uses accelerometers and gyroscopes to calculate a pose. An INS cannot provide accurate pose estimation when there is no acceleration, which is the case for most ground robots operating at a constant speed. Moreover, the pose is estimated by integrating the accelerations twice. A small measurement error can result in significant errors in the estimated pose. This error accumulates over time as the new pose is calculated from the previously determined position. This calculation method is called dead reckoning. Wheel odometry also uses dead reckoning to calculate the current pose based on how many times the wheels of the vehicle have rotated, given the wheels' circumference. The positional error with wheel odometry increases rapidly if the wheels slip.

This research focuses on localizing a robotic vehicle in an agricultural environment. The Lely Juno vehicle follows a path from a charging station to multiple barns and eventually back to the charging station. The charging station can be located inside a barn or on the farm property. The vehicle follows an outdoor path to visit multiple barns and pushes feed for the cattle to the fences inside the barn. The vehicle is equipped with a stereo camera, a 2D-LIDAR, an Inertial Measurement Unit (IMU), and wheel encoders. The vehicle has limited computational capacity onboard. Multiple processes share a single CPU, including localization. Moreover, the Lely Juno is a product that many customers in the agricultural sector use. Therefore, the vehicle should operate without training on the data of the target environment.

Many robotic vehicles are already equipped with low-cost cameras for object detection, including the Lely Juno, which makes using cameras for localization attractive. Moreover, cameras provide rich data compared to other sensors. Using a camera for self-localization is not yet common in the industry. However, there is much ongoing research on visual self-localization for robotic vehicles. A well-known method is ORB-SLAM(3) [8]. ORB-SLAM extracts and matches ORB features in frames and estimates a camera pose based on these matches. A map of the unknown environment is created simultaneously while localizing the vehicle on the map, which is the SLAM principle. SLAM is an extension of Visual Odometry (VO). Visual Odometry only tracks the camera pose with dead reckoning. SLAM extends this with place recognition to close loops, correct the pose that drifts over time, and simultaneously build and optimize a map.

Much research is done in Visual Odometry and Visual SLAM, all focusing on improving the pose estimation accuracy. Almost all research is conducted on specific robotic datasets, such as the EuRoC MAV, KITTI, and TUM-VI. Testing ORB-SLAM3 on agricultural datasets, such as the Rosario and our datasets, resulted in many tracking failures, impeaching the method's robustness. The tracking failures occur due to a drop in the number of ORB feature matches in the tracked frames (see Figure 1.1). The drop in the number of feature matches does not always occur in the same frames when the sequence

is rerun due to the randomness of the keyframe insertion of the ORB-SLAM3 pipeline, which makes the issue difficult to trace.



(a) Frame t-2. Many feature matches can be found in this frame. The total amount of feature matches found in the whole map is 120.

(b) One frame later, at frame t-1, the number of feature matches starts to reduce. The total amount of feature matches found in the whole map is 17.

(c) No feature can be matched at frame t and the rest of the map; this leads to a loss of track. If this system is unable to re-localize, ORB-SLAM3 will start a new empty map.

Figure 1.1: Three subsequent frames of a sequence of the Rosario dataset run on ORB-SLAM3. A green rectangle indicates a feature match. Within three frames, the number of total feature matches in the whole map drops from 120 to a value under 15, which is the minimum amount of feature matches needed for ORB-SLAM3 to calculate a pose. Visually, there is no apparent reason why the number of feature matches could reduce.

This research focuses on the robustness of ORB features in feature-based SLAM or ORB-SLAM3. There are two main research questions:

1. *Why does feature matching often return a small number of matches for the agricultural datasets in ORB-SLAM3?*

   Hypothesis: Two possibilities are expected for the cause of the drop in feature matches; 1. The ORB feature descriptor might not be descriptive enough to robustly match features correctly. This can cause drops in feature matches in scenes where the environment is especially repetitive or textureless. 2. The drop in feature matches can be related to the ORB-SLAM3 pipeline itself. A specific design choice or a bug in the pipeline can cause these drops.

   To be able to answer the research question above, the following three subquestions will be answered first.

   (a) *What ORB feature extraction settings can we use for our experiments that work well for all datasets?*
   Hypothesis: The ORB feature extraction parameters can affect the feature responses' strength. Weak responses can also be more challenging to match. To reduce the time needed to execute the experiments, determining a good set of ORB feature extraction parameters that work well on all datasets would be crucial.

   (b) *Can we replicate the issue of sudden drops in feature matches outside of the ORB-SLAM3 pipeline?*
   Hypothesis: If the issue of the drop in feature matches also occurs outside of the ORB-SLAM3 pipeline, it is more likely due to the ORB features. If not, the issue is probably within the ORB-SLAM3 pipeline.

   (c) *Are feature matches affected by the type of dataset used, and which properties of the dataset affect the feature matching?*
   Hypothesis: Specific scenes or frames might also affect the feature matches, such as blur and illumination changes.

   (d) *Can we improve the ORB-SLAM3 matcher with tuning?*
   Hypothesis: Tuning the parameters of the matcher in the ORB-SLAM3 pipeline might help to reduce the drops in feature matches.

2. *Can we implement a simple camera pose estimator that outperforms ORB-SLAM3 in terms of robustness without compromising too much of the accuracy?*

   Hypothesis: The ORB-SLAM3 pipeline might not be needed for the Lely Juno, as a simple Visual Odometry system might suffice in terms of accuracy for the length of the trajectory of the Lely Juno. A simple Visual Odometry is probably also more robust than the ORB-SLAM3 pipeline with many different sub-components that work in parallel.

2

Related Work

2.1. Visual Odometry

Visual odometry [37] is a method used to estimate the relative camera pose using sequential images. The robot's pose can be determined without needing a prior database. It is a relative method, meaning the pose is calculated relative to the previous frame(s), called dead-reckoning. The pose will drift more and more over time unless corrected. The pose is optimized or corrected with bundle adjustment, pose graph optimization or loop closures ([48]) which is often implemented in the map maintenance part of visual SLAM applications (Chapter 2.3). Different approaches exist to optimize a camera pose, such as geometric approaches where the pose is optimized with projective geometry, learning-based approaches that solely use machine learning, or hybrid approaches that combine projective geometry with machine learning.

Geometric approaches use projective geometry to calculate poses from matches [48]. The motion is estimated by associating 3D to 3D points, 3D to 2D points, or 2D to 2D points (see Figure 2.1). Epipolar geometry solves the camera pose in the 2D to 2D point association, while Perspective-n-Point is used in the 3D-2D case. The 3D to 3D point association is done directly by extracting the 3D position from the stereo-camera but is rarely used as it is not as accurate as the 3D-2D points association [39]. The matching method is dense when all pixels are matched. The method is semi-dense if some pixels are matched, such as those with a high gradient. The methods that try to minimize the number of pixels or features matched are sparse methods.

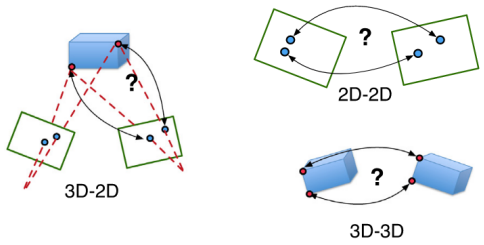


Figure 2.1: Data association methods [48].

In general, there are two geometric approaches; feature-based and appearance-based. Feature-based methods extract features from pixels and match these features to the features in the other image. Appearance-based methods use the pixels without any feature extraction to match them to the pixels in the other image.

Sometimes different versions of the same method exist for monocular, and stereo cameras [26]. Stereo cameras are convenient for directly extracting depth information; however, it is susceptible to correct camera calibration. Shocks and vibrations during use can degrade the extrinsic calibration over time. Furthermore, the stereo cameras' baseline affects the depth information's accuracy. Long baselines can provide more accurate depth information for objects far away, while shorter baselines

can provide accurate depth information for objects nearby. The baseline is, however, fixed and limited by the vehicle's size.

On the other hand, monocular methods are not limited by the baseline problem. A drawback, however, is that they need at least three different images to find the depth with triangulation. Because the translation and rotation vectors between the first two images are unknown, the distance between the first two frames is set to a predetermined value. This value needs to be deducted, for example, from IMU or LiDAR data. Also, loop closures are crucial for handling scale drift ([49]), which happens due to errors during triangulation.

Feature-based visual odometry methods have the limitation of failing in low-textured environments ([26]). Appearance-based methods are more robust to low-textured environments but less to illumination changes. Also, for the geometry-based methods with 2D to 2D association, epipolar geometry is used to solve the essential matrix [48]. This solver generally works well, but in some cases, it cannot estimate a camera pose; 1. When there is pure rotation, the rotation matrix becomes unsolvable. 2. When the camera translation is small, the solution becomes unstable. 3. When the scale is ambiguous. Monocular methods are also prone to scale drift unless corrected in SLAM methods by keeping a scale-consistent map by performing global bundle adjustments for scale optimization or with additional prior assumptions. SLAM methods perform well for long-term localization, primarily due to the long-term data association with loop closures. However, when loop closures are not working because the environment is too repetitive or there are no loops in the trajectory, it is hard to recover when tracking failures happen or to keep an accurate estimation in the long term.

2.2. Visual Place Recognition

Visual place recognition [24] is a method to determine whether the robot is in a previously visited place based on the sensory input from the camera(s) and a prior database. Visual place recognition can be a means for camera pose estimation, or it can be used to correct for the drift in visual odometry (chapter 2.1) or SLAM (chapter 2.3) methods by detecting loop-closures.

Visual place recognition is challenging due to several reasons; The appearance of a place can change drastically due to changes in day and night, seasons, viewpoints, and occlusions. Furthermore, perceptual aliasing can occur due to multiple places that look similar. Visual place recognition systems have, in general, three components;

1. The image processing component; interprets the incoming visual data.
2. The map; represents the robot's knowledge of the environment.
3. Belief generation component; uses the sensor data in combination with the map to update the systems' belief of where the robot is localized.

Hand-crafted features are widely used to extract features from images. The feature descriptors are subsequently matched to the descriptors in the database. The descriptors in the database can be based on images or a 3D model. Although feature-based matching methods are quite outdated, they are still competitive compared to learning-based methods [4]. Feature-based methods are still state-of-the-art on medium and large-scale datasets.

In image-based approaches, hand-crafted features are used to extract features in the query image, and the feature descriptors are matched to the feature descriptors from the database image. A popular approach for image retrieval is the bag of visual words model and variations thereof ([41]; [17]; [21]; [9]; [10]) due to its efficiency. In this model, the feature descriptors are converted to codewords (sets of similar features), and each image is represented by a frequency histogram of the features present in the image. A similar image can be retrieved by comparing their frequency histogram. In this approach, spatial relationships are lost. However, spatial verification and outlier removal are often done afterward with Random Sample Consensus (RANSAC) based methods ([17];[10]) during pose estimation together with Perspective-n-Point (PnP).

2.3. Visual SLAM

SLAM stands for Simultaneous Localization and Mapping [48]. SLAM methods build a map during localization, which also helps localization itself. SLAM methods can be filter-based or optimization-based (also called keyframe-based). Most visual SLAM methods use the latter due to higher accuracy.

Visual SLAM approaches combine visual odometry with place recognition. Initialization, data association, pose optimization, and keyframe management are part of visual odometry and SLAM. The place recognition module detects loop closures. Map maintenance and map expansion modules are added for the mapping part of visual SLAM to generate and correct the map and simultaneously the pose estimation. Pure visual odometry methods do not have a global optimization module, such as the loop closures in visual SLAM.

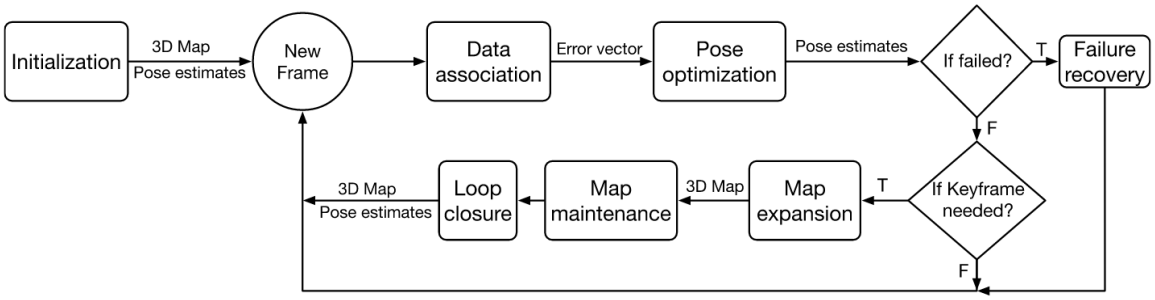


Figure 2.2: Flow-chart of a general keyframe-based visual SLAM method [48].

ORB-SLAM is a widely used feature-based method that is integrated into a visual SLAM method, first introduced in 2015 [27]. The initial method was meant for monocular cameras and could operate in real-time in small and large indoor and outdoor environments. The current version is called ORB-SLAM3 [8]. It can perform visual, visual-inertial, and Multimap SLAM on monocular and stereo cameras in real time on a CPU. They show state-of-the-art results on the trajectory error compared to other feature-based and appearance-based visual and visual-inertial methods in the EuRoC and the TUM VI datasets (Figure 2.3 and Figure 2.4). The authors achieve remarkable accuracy in ORB-SLAM3 by using short-term, mid-term, and long-term data associations.

Short-term data association matches map elements from the last few frames; the elements are forgotten when they leave the view. Most visual odometry methods use this association which is prone to drift. Mid-term data association matches map elements near the camera while the accumulated drift is still small. Long-term data association uses place recognition to reset the accumulated drift. Most visual SLAM methods do not use all associations, which is the reason for the lower accuracy compared to ORB-SLAM3, according to the authors of ORB-SLAM3.

			MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg[1]
Monocular	ORB-SLAM [4]	ATE[2,3]	0.071	0.067	0.071	0.082	**0.060**	**0.015**	0.020	-	**0.021**	**0.018**	-	0.047*
	DSO [27]	ATE	0.046	0.046	0.172	3.810	0.110	0.089	0.107	0.903	0.044	0.132	1.152	0.601
	SVO [24]	ATE	0.100	0.120	0.410	0.430	0.300	0.070	0.210	-	0.110	0.110	1.080	0.294*
	DSM [31]	ATE	0.039	0.036	0.055	**0.057**	0.067	0.095	0.059	0.076	0.056	0.057	**0.784**	**0.126**
	ORB-SLAM3 (ours)	ATE	**0.016**	**0.027**	**0.028**	0.138	0.072	0.033	**0.015**	**0.033**	0.023	0.029	-	0.041*
Stereo	ORB-SLAM2 [3]	ATE	0.035	**0.018**	0.028	0.119	0.060	**0.035**	**0.020**	**0.048**	**0.037**	0.035	-	0.044*
	VINS-Fusion [44]	ATE	0.540	0.460	0.330	0.780	0.500	0.550	0.230	-	0.230	0.200	-	0.424*
	SVO [24]	ATE	0.040	0.070	0.270	0.170	0.120	0.040	0.040	0.070	0.050	0.090	0.790	0.159
	ORB-SLAM3 (ours)	ATE	**0.029**	0.019	**0.024**	**0.085**	**0.052**	**0.035**	0.025	0.061	0.041	**0.028**	**0.521**	**0.084**
Monocular Inertial	MCSKF [33]	ATE[5]	0.420	0.450	0.230	0.370	0.480	0.340	0.200	0.670	0.100	0.160	1.130	0.414
	OKVIS [39]	ATE[5]	0.160	0.220	0.240	0.340	0.470	0.090	0.200	0.240	0.130	0.160	0.290	0.231
	ROVIO [42]	ATE[5]	0.210	0.250	0.250	0.490	0.520	0.100	0.100	0.140	0.120	0.140	0.140	0.224
	ORBSLAM-VI [4]	ATE[2,3] scale error[2,3]	0.075 0.5	0.084 0.8	0.087 1.5	0.217 3.5	0.082 0.5	**0.027** 0.9	0.028 0.8	- -	**0.032** 0.2	0.041 1.4	0.074 0.7	0.075* 1.1*
	VINS-Mono [7]	ATE[4]	0.084	0.105	0.074	0.122	0.147	0.047	0.066	0.180	0.056	0.090	0.244	0.110
	VI-DSO [46]	ATE scale error	**0.062** 1.1	0.044 0.5	0.117 0.4	0.132 0.2	0.121 0.8	0.059 1.1	0.067 1.1	0.096 0.8	0.040 1.2	0.062 0.3	0.174 0.4	0.089 0.7
	ORB-SLAM3 (ours)	ATE scale error	**0.062** 1.4	**0.037** 0.3	**0.046** 0.8	**0.075** 0.5	**0.057** 0.3	0.049 2.0	**0.015** 0.6	**0.037** 2.2	0.042 0.7	**0.021** 0.4	**0.027** 1.0	**0.043** 0.9
Stereo Inertial	VINS-Fusion [44]	ATE[4]	0.166	0.152	0.125	0.280	0.284	0.076	0.069	0.114	0.066	0.091	0.096	0.138
	BASALT [47]	ATE[3]	0.080	0.060	0.050	0.100	**0.080**	0.040	0.020	0.030	**0.030**	0.020	-	0.051*
	Kimera [8]	ATE	0.080	0.090	0.110	0.150	0.240	0.050	0.110	0.120	0.070	0.100	0.190	0.119
	ORB-SLAM3 (ours)	ATE scale error	**0.036** 0.6	**0.033** 0.2	**0.035** 0.6	**0.051** 0.2	0.082 0.9	**0.038** 0.8	**0.014** 0.6	**0.024** 0.8	0.032 1.1	**0.014** 0.2	**0.024** 0.2	**0.035** 0.6

Figure 2.3: ORB-SLAM3 performance comparison on the EuRoC dataset. Systems that did not complete all sequences are denoted by * and are not marked in bold. ([8])

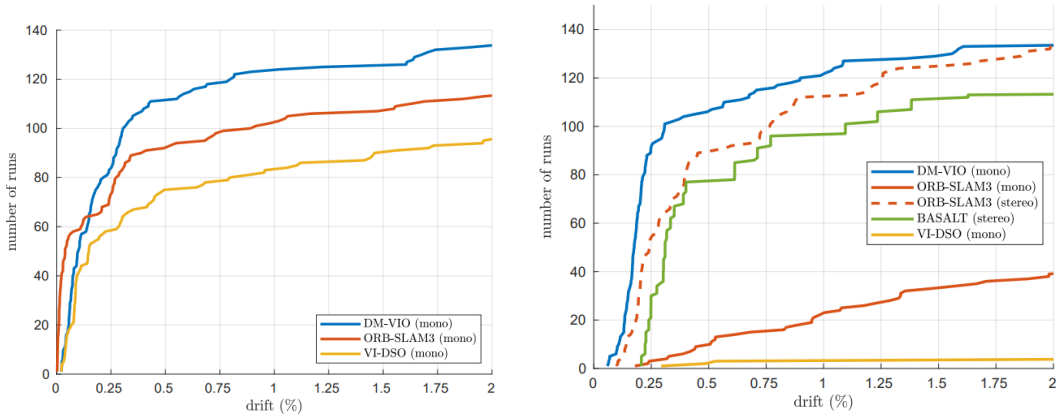
Seq.	Mono-Inertial		Stereo-Inertial				Length (m)	LC
	VINS-Mono	ORB-SLAM3	OKVIS	ROVIO	BASALT	ORB-SLAM3		
corridor1	0.63	**0.04**	0.33	0.47	0.34	**0.03**	305	✓
corridor2	0.95	**0.02**	0.47	0.75	0.42	**0.02**	322	✓
corridor3	1.56	**0.31**	0.57	0.85	0.35	**0.02**	300	✓
corridor4	0.25	**0.17**	0.26	**0.13**	0.21	0.21	114	
corridor5	0.77	**0.03**	0.39	2.09	0.37	**0.01**	270	✓
magistrale1	2.19	**0.56**	3.49	4.52	1.20	**0.24**	918	✓
magistrale2	3.11	**0.52**	2.73	13.43	1.11	**0.52**	561	✓
magistrale3	**0.40**	4.89	1.22	14.80	**0.74**	1.86	566	
magistrale4	5.12	**0.13**	0.77	39.73	1.58	**0.16**	688	✓
magistrale5	**0.85**	1.03	1.62	3.47	**0.60**	1.13	458	✓
magistrale6	2.29	**1.30**	3.91	X	3.23	**0.97**	771	
outdoors1	74.96	**70.79**	X	101.95	255.04	**32.23**	2656	
outdoors2	133.46	**14.98**	73.86	21.67	64.61	**10.42**	1601	
outdoors3	**36.99**	39.63*	32.38	**26.10**	38.26	54.77	1531	
outdoors4	**16.46**	25.26	19.51	X	17.53	**11.61**	928	
outdoors5	130.63	**14.87**	13.12	54.32	**7.89**	8.95	1168	✓
outdoors6	133.60	**16.84**	96.51	149.14	65.50	**10.70**	2045	
outdoors7	21.90	**7.59**	13.61	49.01	**4.07**	4.58	1748	✓
outdoors8	83.36	**27.88**	16.31	36.03	13.53	**11.02**	986	
room1	0.07	**0.01**	0.06	0.16	0.09	**0.01**	146	✓
room2	0.07	**0.02**	0.11	0.33	0.07	**0.01**	142	✓
room3	0.11	**0.04**	0.07	0.15	0.13	**0.01**	135	✓
room4	0.04	**0.01**	0.03	0.09	0.05	**0.01**	68	✓
room5	0.20	**0.02**	0.07	0.12	0.13	**0.01**	131	✓
room6	0.08	**0.01**	0.04	0.05	0.02	**0.01**	67	✓
slides1	**0.68**	0.97	0.86	13.73	**0.32**	0.41	289	
slides2	**0.84**	1.06	2.15	0.81	**0.32**	0.49	299	
slides3	**0.69**	**0.69**	2.58	4.68	0.89	**0.47**	383	

Figure 2.4: ORB-SLAM3 performance comparison on the TUM VI dataset, measured with RMS ATE (m). * indicates that one out of the three runs was not successful. ([8])

The authors claim that the leading failure cause of ORB-SLAM3 is low-textured environments. Also, the matching of feature descriptors is less robust for tracking than the Lucas-Kanade method, which uses photometric information. Another failure cause can be vehicles driving on a flat area with no roll and pitch or with slow motions, which can cause the IMU sensor to have difficulty initializing in the visual-inertial case. In that case, the stereo configuration should be used.

DM-VIO The appearance-based Delayed Marginalization Visual-Inertial Odometry (DM-VIO) [42] is the only method that slightly outperforms ORB-SLAM3 in terms of localization accuracy. This monocular method builds on top of DSO ([11]) by adding IMU integration and two novel adjustments; delayed marginalization (removal of unused map points and keyframes) and pose graph bundle adjustment. In delayed marginalization, a second factor graph is maintained where marginalization is delayed compared to the first graph. The delayed graph can be used to get an updated marginalization prior and enables the addition of IMU information into already marginalized states. The Pose Graph Bundle Adjustment (PGBA) combines pose graph optimization and bundle adjustment, which is faster than applying full bundle adjustment and more accurate than pose graph optimization. PGBA uses the delayed graph for IMU initialization. The system starts with visual-only odometry and runs an IMU initializer in parallel. The scale is initially unobservable due to the monocular camera, so the method continuously optimizes scale and the direction of gravity in the main system after IMU initialization is done. Monocular visual-inertial methods are generally less accurate than stereo-inertial methods as scale is hard to determine with a monocular camera. The inertial system can be used to find scale. However, this fails when the motion is constant. DM-VIO has excellent robustness compared to other visual-inertial methods (Figure 2.5); it even outperforms stereo-inertial methods. DM-VIO copes with a scale that is initially unobservable by continuously optimizing the scale and gravity direction in the primary system after IMU initialization.

The results show that DM-VIO is more robust in terms of accuracy with an increasing number of runs on the TUM-VI and 4Seasons dataset compared to different mono- and stereo-inertial methods (Figure 2.5). DM-VIO even outperforms the stereo-inertial ORB-SLAM3, which uses loop closures. Note that all experiments in the paper are performed in real-time mode on the datasets on a CPU.



(a) robustness in terms of accumulated drift with an increasing number of runs on the TUM-VI dataset. DM-VIO outperforms ORB-SLAM3 and VI-DSO. On some sequences, ORB-SLAM3 has better performance due to its loop-closure system.

(b) robustness in terms of accumulated drift with an increasing number of runs on the 4Seasons dataset. The 4Seasons dataset is challenging for monocular visual-inertial systems as scale can not be observed due to constant motion and, therefore, challenging for IMU initialization.

Figure 2.5: Robustness of DM-VIO compared to different visual-inertial methods on the TUM-VI and 4Seasons dataset. ORB-SLAM3 uses loop closures. All other methods are without loop-closing.

The RMS ATE on the TUM-VI dataset (Figure 2.6) and the EuRoC dataset (2.7) further show that DM-VIO outperforms other methods on most sequences in terms of RMSE, even stereo-inertial methods. Basalt is the closest competitor in terms of RMSE on the TUM-VI dataset (Figure 2.6).

Sequence	ROVIO stereo	VINS mono	OKVIS stereo	BASALT stereo	DM-VIO mono	length [m]
corridor1	0.47	0.63	0.33	0.34	**0.19**	305
corridor2	0.75	0.95	0.47	**0.42**	0.47	322
corridor3	0.85	1.56	0.57	0.35	**0.24**	300
corridor4	**0.13**	0.25	0.26	0.21	**0.13**	114
corridor5	2.09	0.77	0.39	0.37	**0.16**	270
magistrale1	4.52	2.19	3.49	**1.20**	2.35	918
magistrale2	13.43	3.11	2.73	**1.11**	2.24	561
magistrale3	14.80	**0.40**	1.22	0.74	1.69	566
magistrale4	39.73	5.12	**0.77**	1.58	1.02	688
magistrale5	3.47	0.85	1.62	**0.60**	0.73	458
magistrale6	X	2.29	3.91	3.23	**1.19**	771
outdoors1	101.95	**74.96**	X	255.04	123.24	2656
outdoors2	21.67	133.46	73.86	64.61	**12.76**	1601
outdoors3	26.10	36.99	32.38	38.26	**8.92**	1531
outdoors4	X	16.46	19.51	17.53	**15.25**	928
outdoors5	54.32	130.63	13.12	7.89	**7.16**	1168
outdoors6	149.14	133.60	96.51	65.50	**34.86**	2045
outdoors7	49.01	21.90	13.61	**4.07**	5.00	1748
outdoors8	36.03	83.36	16.31	13.53	**2.11**	986
room1	0.16	0.07	0.06	0.09	**0.03**	146
room2	0.33	**0.07**	0.11	**0.07**	0.13	142
room3	0.15	0.11	**0.07**	0.13	0.09	135
room4	0.09	0.04	**0.03**	0.05	0.04	68
room5	0.12	0.20	0.07	0.13	**0.06**	131
room6	0.05	0.08	0.04	**0.02**	**0.02**	67
slides1	13.73	0.68	0.86	0.32	**0.31**	289
slides2	0.81	0.84	2.15	**0.32**	0.87	299
slides3	4.68	0.69	2.58	0.89	**0.60**	383
avg drift%	16.83*	1.700	0.815*	0.939	**0.472**	normalized

Figure 2.6: RMSE ATE (m) on the TUM-VI dataset. The methods do not have loop closing. A full SLAM system using loop closures could achieve even more accurate results.

Sequence		MH1	MH2	MH3	MH4	MH5	V11	V12	V13	V21	V22	V23	Avg
MCSKF[2] [1] (M)	RMSE	0.42	0.45	0.23	0.37	0.48	0.34	0.20	0.67	0.10	0.16	1.13	0.414
OKVIS[1] [19] (M)	RMSE	0.33	0.37	0.25	0.27	0.39	0.094	0.14	0.21	0.090	0.17	0.23	0.231
ROVIO[2] [18] (M)	RMSE	0.21	0.25	0.25	0.49	0.52	0.10	0.10	0.14	0.12	0.14	0.14	0.224
VINS-Mono [3] (M)	RMSE	0.15	0.15	0.22	0.32	0.30	0.079	0.11	0.18	0.080	0.16	0.27	0.184
Kimera [21] (S)	RMSE	0.11	0.10	0.16	0.24	0.35	0.05	0.08	0.07	0.08	0.10	0.21	0.141
Online VIO [23] (M)	RMSE	0.14	0.13	0.20	0.22	0.20	0.05	0.07	0.16	0.04	0.11	0.17	0.135
VI-DSO [6] (M)	RMSE	**0.062**	**0.044**	0.117	0.132	0.121	0.059	0.067	0.096	0.040	0.062	0.174	0.089
	Scale Error (%)	1.1	0.5	0.4	0.2	0.8	1.1	1.1	0.8	1.2	0.3	0.4	0.7
BASALT [20] (S)	RMSE	0.07	0.06	**0.07**	0.13	0.11	**0.04**	0.05	0.10	0.04	**0.05**	-	0.072
DM-VIO (M)	RMSE	0.065	**0.044**	0.097	**0.102**	**0.096**	0.048	**0.045**	**0.069**	**0.029**	**0.050**	**0.114**	**0.069**
	Scale Error (%)	1.3	0.9	0.4	0.2	0.4	0.4	1.0	0.3	0.02	0.6	0.8	0.6

Figure 2.7: RMSE ATE (m) on the TUM-VI dataset. The methods do not have loop closing.

2.4. SLAM evaluation

Most of the literature focuses on improving the accuracy of a visual SLAM or visual odometry (VO) system. Some literature focus on robustness in terms of repeatability of the same accuracy [32]. The accuracy of a SLAM/VO system is often measured with the root mean squared absolute trajectory error (ATE) [50]. However, ATE is sensitive to the time an estimation error occurs. An estimation error at the beginning of the trajectory gives a larger ATE than an error at the end. Therefore, relative position error (RPE) is also used in addition to the ATE to provide a more informative error.

The datasets used in most SLAM/VO literature can be found in Tabel 2.1. Most datasets are recorded in an urban or indoor environment with relatively many (distinctive) features compared to an agricultural setting. Therefore, most visual localization methods are not tested extensively in low-textured environments.

Evaluation of the pose estimation seems biased in most cases [4]. The performance of the method depends on how the ground truth is obtained. For example, if a SLAM method is used to generate

ground truth data, the accuracy is often higher if the bench-marked method is similar; this is due to the similar optimized cost functions, which have the same local minima.

Table 2.1: Commonly used datasets in visual SLAM research. A '-' means not present and 'x' means present.

Dataset	Setting	Pose data	Camera	Seasons	Illumination	Image capture
CMU Seasons ([36])	Suburban	6 DoF	Stereo	x	-	Trajectory
RobotCar Seasons ([36])	Urban	6 DoF	Stereo	x	x	Trajectory
KITTI Vision Benchmark Suite ([18])	Urban	6 DoF	Stereo	-	-	Trajectory
TUM monoVO ([12])	Indoor, Suburban	6 DoF	Mono	-	-	Trajectory
TUM RGB-D ([43])	Indoor	6 DoF	Stereo	-	-	Trajectory
TUM LSI ([46])	Indoor	6 DoF, GPS	Mono	-	-	Sets of contiguous captures
TUM VI Benchmark ([38])	Indoor, Suburban	6 DoF	Mono	-	-	Trajectory
EuRoC MAV ([6])	Indoor	6 DoF, 3D position from laser	Stereo	-	-	Trajectory
7 Scenes ([40])	Indoor	6 DoF	Stereo	-	-	Trajectory

2.5. Contributions

The literature focuses on improving localization accuracy and robustness in terms of repeatability of the accuracy. To the best of our knowledge, there is no literature about the understanding of tracking failures. Failures in feature matching in low-textured or repetitive environments are addressed; however, it needs further investigation or improvement. The general cause of sudden low feature matches that cause tracking failures in any environment is not yet clear. The main contributions of this work can be summarized as follows:

- The robustness of ORB-SLAM3 regarding tracking errors due to sudden low feature matches is investigated for agricultural datasets and the commonly used datasets in the literature. The number of feature matches is recorded in an OpenCV implementation of the ORB feature matching outside the ORB-SLAM3 pipeline.
- The effect of different ORB extraction parameters on the matching performance is investigated for the different datasets, giving insights for proper parameter tuning.
- Two new datasets are created on existing Lely machines for the agricultural application. One dataset contains recordings of a machine working on a grass field. The cameras point downwards, which means that the cameras see mainly grass; therefore, the environment is visually repetitive. The second dataset contains recordings of a machine working in the barn environment. The cameras point straight to the horizon. The dataset contains many distinct features inside the barn. However, the environment is texture-less when the machine travels outside to the different barns due to concrete floor slabs.

3

Methods

This work focuses mainly on the feature extraction and matching part of ORB-SLAM3. The ORB-SLAM3 feature extractor is compared to a general OpenCV-based feature extraction pipeline for the experiments. A general approach for feature extraction and matching is explained in Chapter 3.1. The ORB-SLAM3 pipeline is explained in Chapter 3.2. ORB features, which are the baseline for the tracking in ORB-SLAM3, are explained in Chapter 3.3. The Bag of Binary Words method, which ORB-SLAM3 uses for feature matching, is explained in more detail in Chapter 3.4. For the second research question, a simple visual odometry system is implemented. Therefore, the principles of stereo visual odometry can be found in Chapter 3.5. In stereo visual odometry, triangulation is used for pose estimation after the general feature extraction and matching principle, which is explained in more detail in Chapter 3.6. Correlation factors and plots are used to evaluate specific experiments, which is explained in Chapter 3.7.

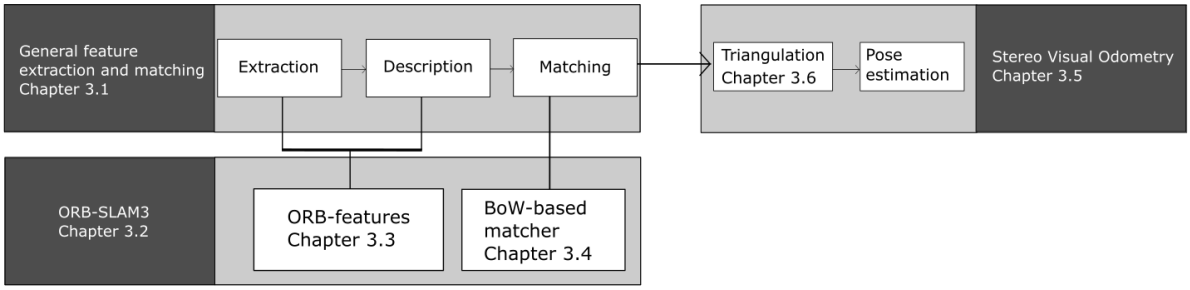


Figure 3.1: The relationship between methods used in this work.

3.1. Feature extraction and matching

Features [44] are local interest points or areas in an image. These features include lines, edges, corners (also called interest points), patches, and more. Feature detection and matching are used to find point correspondences in an image pair. There are many applications for using these feature correspondences; they can be used to align images for image stitching, to construct a 3D model, or to estimate a camera pose in visual odometry or SLAM systems.

The features can be hand-crafted or extracted with deep networks. However, the most popular features that are used for matching are hand-crafted corners [25]. These corners are easy to extract and are simple to describe. A good feature is distinctive and fast to compute.

After feature extraction, the feature is described with a local patch descriptor; this is needed to recognize a feature in different images and subsequently match the features. There are also several different descriptors, such as binary descriptors commonly used with ORB features or non-binary descriptors commonly used with, for example, SIFT features. Different features can be combined with different descriptors. An advantage of binary descriptors is that they are fast to compute. However, they are less distinctive compared to non-binary descriptors.

The matching is generally done with Brute Force matching or a Fast Library for Approximate Nearest Neighbors (FLANN) based matching [5]. The Brute Force matcher is slower than the FLANN matcher as each feature in an image is matched to all features in the corresponding image. The best k matches can be found with the k-nearest neighbor algorithm. The FLANN matcher is based on algorithms optimized for fast nearest neighbor search in large datasets and high dimensional features. After matching with Brute Force or FLANN, many matches can still be incorrect. Most of the incorrect matches can be filtered with Lowe's ratio test [23] and Random Sample Consensus (RANSAC) [14] outlier filtering afterward. Matches that remain after RANSAC filtering are called inliers. The OpenCV library [5] provides many feature extractors and matching functions that are widely used for computer vision applications. For the experiments, ORB features are combined with a Brute-force matcher to find feature matches. The outliers are subsequently filtered with Lowe's ratio test. Note that this implementation differs in two major ways from the ORB extractor and matcher in ORB-SLAM3:

- In ORB-SLAM3, the image is divided into grids. The features are extracted from each grid to make the spread of the features more homogeneous over the image.
- ORB-SLAM3 uses a Bag of Words model to compare features initially with a brute force matching, assuming that features close in the descriptor space will also be close in the bag of words model. Subsequently, there are no parameters for the k-nearest neighbor in ORB-SLAM3. It also means that descriptors are only compared if the visual words are close enough.

The exact steps in the pipelines of ORB-SLAM3 are as follows;

1. Extracting ORB features from each grid of the image.
2. Calculating the descriptors of the extracted features.
3. Finding correspondences between features in two images by comparing only the features associated with the same nodes at a preset level in the vocabulary tree of the Bag of Words model (see Chapter 3.4).
4. The correspondences are further filtered by comparing their descriptors and keeping the correspondences that have a distance in the feature space under a certain threshold and also pass the Lowe's ratio test.

The steps in the OpenCV implementation are as follows;

1. Extracting ORB features from the image
2. Calculating the descriptors of the extracted features.
3. Finding correspondences by applying a brute-force matcher.
4. The correspondences are further refined with Lowe's ratio test and possibly RANSAC.

As the ORB-SLAM3 implementation does not use RANSAC, the OpenCV implementation will be tested with and without RANSAC in an experiment to determine if RANSAC is needed.

3.2. ORB-SLAM3

ORB-SLAM3 [8] is a state-of-the-art feature-based visual localization method. ORB-SLAM3 uses ORB features, which are lightweight and widely used for real-time applications. The appearance-based DM-VIO [42] is the only method that slightly outperforms ORB-SLAM3 in terms of localization accuracy. However, unlike DM-VIO, ORB-SLAM3 has a very active code repository, which also played a role in choosing this method. Furthermore, our main concern is robustness in tracking failures. Both methods were only tested for accuracy and not for tracking robustness in the literature. We use the stereo configuration in ORB-SLAM3, which is more accurate than the monocular configuration. Fusing inertial information does not work for our use case since the Lely Juno runs with constant speed. Therefore, we will explain the ORB-SLAM3 pipeline for the stereo configuration. Furthermore, ORB-SLAM3 also has a localization mode, where the local mapping and loop closing threads are disabled. The localization mode uses visual odometry and relocalization to localize within an existing map. The pipeline consists of the tracking, local mapping, loop- and map merging threads.

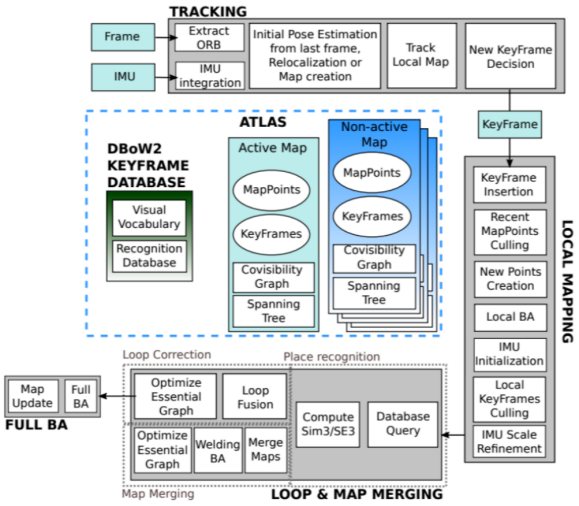


Figure 3.2: The full ORB-SLAM3 pipeline [8] .

3.2.1. The tracking thread

The tracking thread tracks the pose of the camera. The thread extracts ORB features from incoming frames. The features are matched between the last frame and its reference keyframe to make an initial pose estimation of the last frame. This matching is done by projecting the feature points seen in the previous frame with a motion model (TrackWithMotionModel() in Figure 3.4). The pose is estimated if enough matches are found by minimizing the re-projection error. If there are not enough matches, the tracking matches all features in the reference keyframe to the features in the last frame that have a similar visual word (TrackReferenceKeyframe() in Figure 3.4). The bag of words model used for place recognition and feature matching is explained in detail in Chapter 3.4. If this step fails due to insufficient matches for pose estimation, the system will relocate or start a new map if relocalization fails. Relocalization is done with place recognition. The relocalization thread matches ORB features from the current frame to the features in each candidate place from the query database. A pose is estimated if a candidate has enough matches.

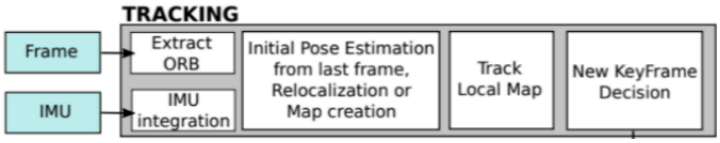


Figure 3.3: The tracking component of ORB-SLAM3 [8].

Suppose there are enough matches from the tracking, either from the motion model, the tracking with the reference keyframe, or relocalization. In that case, the system continues tracking the local map. Initially, there are some feature matches in the frame and an initial camera pose estimation. The camera pose can be further refined by projecting the local map to the frame and searching for map point correspondences. The local map consists of a set of keyframes that share map points with the current frame and a set of their co-visible keyframes. The local map has a reference keyframe that shares most map points with the current frame. After refining the pose, the tracking thread determines if the current frame should become a new keyframe depending on the following conditions [27]:

- More than 20 frames have passed from the last global relocalization.
- Local mapping thread is idle, or more than 20 frames have passed from the last keyframe insertion.
- Current frame tracks at least 50 map points.
- Current frame tracks less than 90% map points than the reference keyframe.

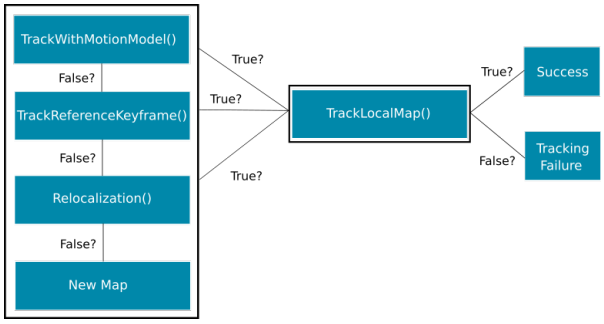


Figure 3.4: The initial pose estimation flow in more detail. The track local map thread is only triggered if an initial pose estimation can be made.

3.2.2. The local mapping thread

The local mapping thread [27] inserts the keyframe and updates the co-visibility graph if a new keyframe insertion is decided in the tracking thread. To ensure correct data association, the tracking thread continues to cull recent map points observed by less than three keyframes. Map points can at any time be seen by less than three keyframes when a keyframe is removed or when local bundle adjustment removes outliers. New map points are created if ORB matches between the inserted keyframe and the connected keyframes from the co-visibility graph can be triangulated after the positive depth in both cameras, the parallax, re-projection error, and scale consistency are checked. Afterward, a local bundle adjustment is performed with the new keyframe, the co-visible keyframes, and all the map points seen by the keyframes. All other keyframes that see the same map points are also used for the optimization but remain fixed. The local keyframes culling removes redundant keyframes in the co-visible keyframes set whose 90% of the map points have been seen in at least three other frames on the same or finer scale. The scale condition ensures that the map points that originate from the keyframe, measured with the most accuracy, are maintained.

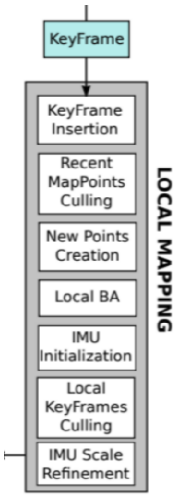


Figure 3.5: The local mapping thread [8].

3.2.3. The loop & map merging thread

This thread performs place recognition and corrects the loop subsequently. For place recognition, the bag of words database in the Atlas is queried to find the three most similar keyframes to the active keyframe, excluding the co-visible keyframes of the active keyframe. Subsequently, a local window is defined for each candidate keyframe, including the keyframe, its best co-visible keyframes, and all the map points observed. A rigid body transformation is calculated that aligns the map points in the local window of the candidate keyframe and the map points of the active keyframe. All the map points in the local window are transformed using the transformation found to find more matches with the key points in the active keyframe and vice versa. The transformation can be further optimized with non-linear

optimization using all the matches found.

Place recognition is verified by searching two keyframes in the active map co-visible with the active keyframe, where the number of matches with map points in the local window is over a certain threshold. The validation continues until three keyframes verify the transformation or fail if two consecutive new keyframes fail to verify it.

The map merging process is initiated if the keyframes matched by place recognition belong to different maps. The active map is brought to the reference of the matched map. Merging the whole map could take a long time. Therefore, the merging process is split into two operations. First, the merge is performed in a welding window defined by the neighbors of the active and the matched keyframe in the co-visibility graph. In the second operation, the correction is propagated to the rest of the merged map with a pose-graph optimization. Loop closing is analogous to map merging, but keyframes matched by place recognition belong to the same active map.

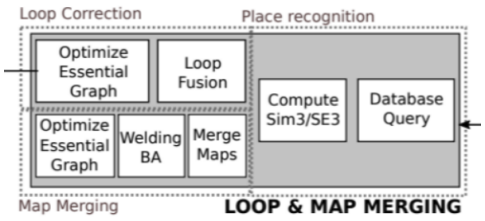


Figure 3.6: The loop and map merging thread [8].

3.2.4. The ATLAS

The ATLAS consists of a set of disconnected maps. The local mapping thread continuously grows the active map with new keyframes and map points. The non-active maps can become active after relocalization or map merging. A bag of words database is also maintained in the ATLAS, consisting of keyframes. This database is used for relocalization, loop closing, and map merging.

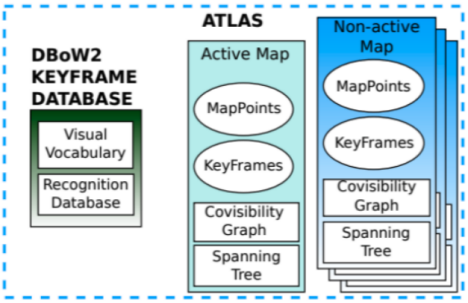


Figure 3.7: The ATLAS [8].

3.3. ORB Features

ORB features [35] consists of Oriented FAST keypoint detectors and Rotated BRIEF descriptors. ORB was introduced as an efficient alternative to SIFT and SURF features. ORB features are rotation invariant and resistant to noise.

FAST keypoint detector The FAST (Features from Accelerated Segment Test) keypoint detector [34] is a high-speed keypoint detector suitable for real-time applications. The initial segment test takes a circle of sixteen pixels around a candidate corner or keypoint p (see Figure 3.8). The candidate corner is classified as a corner when at least $n = 12$ contiguous pixels in the circle are brighter or darker than the candidate corner, including a threshold value (fastThreshold parameter in ORB OpenCV). The reason to check 12 contiguous pixels is that the test can be sped up by only considering pixel numbers 1, 5, 9, and 13 in the circle. If p is a corner, then at least three pixels are darker or brighter than the candidate corner p, and a full segment test can be done to check the remaining pixels in the circle. Otherwise, p can not be a corner. There are, however, several problems with this approach:

1. The high-speed test does not generalize well for $n < 12$

2. The choice and the ordering of the fast test pixels contain implicit assumptions about the distribution of feature appearance
3. Knowledge from the first four tests (the high-speed test) is discarded
4. Multiple features are detected adjacent to one another

The first three problems are addressed using machine learning for the corner detector. The corner detector consists of two stages; First, the corners are detected using the slow segment test that tests all 16 pixels in the circle (for a given n and threshold). This is done on a set of images, preferably from the target application domain. For each corner, the 16 pixels with their location x around it are stored as a vector. A feature vector P is created by doing this for all images. For every corner, all 16 pixels are either classified as darker, similar, or brighter than the corner p. Depending on the states, the feature vector P is divided into three subsets; P_d, P_s, or P_b. A Boolean variable K_p is created, which is true if p is a corner and false otherwise. The second stage consists of making a decision tree. This stage starts with selecting the circle pixels at location x, which contain the most information about whether the candidate pixel is a corner measured with the entropy of K_p:

$$H(P) = (c + \bar{c})log_2(c + \bar{c}) - clog_2(c) - \bar{c}log_2(\bar{c}) \tag{3.1}$$

where $c = |p|K_p$ is true | (number of corners) and $\bar{c}$ = $|p|K_p$ is false | (number of non corners)

The information gained with the choice of x is equal to:

$$H(P) - H(P_d) - H(P_s) - H(P_b) \tag{3.2}$$

This process is applied to all three subsets. Each x is chosen to yield maximum information about the set to which it applies. The process stops when a subset's entropy is zero, meaning that all p in the subset are either all corners or all non-corners. The decision tree created can be used for FAST detection in other images.

An indirect non-maximum suppression is applied to address the last problem. A score function V is computed for each detected corner. This V is the sum of the absolute difference between the corner p and the 16 surrounding pixel values. The score functions of two adjacent corners are compared the one with a lower V value is discarded.

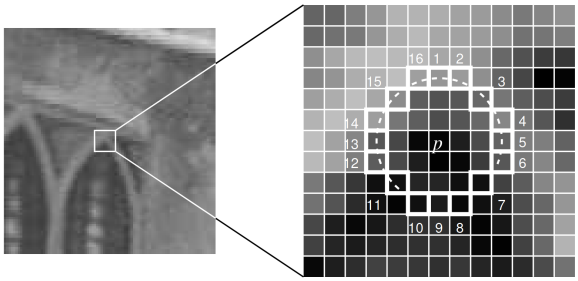


Figure 3.8: The 12-point segment test corner detection in an image patch [34]

The original FAST keypoint detector is not invariant to rotation and scale. The FAST keypoint detector is made rotation and partially scale invariant in ORB. Rotation invariance is done by using the intensity centroid as a corner orientation measure, which assumes that a corner's intensity is the offset from its center. This vector is used to define an orientation.

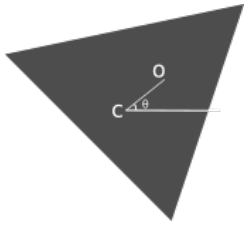


Figure 3.9: Illustration of the orientation measure θ. Defined through the line between the center of mass C and the center O of a patch.

The moments of a patch have to be found first to calculate the center of mass:

$$m_p q = \sum_{x,y} x^p y^q I(x,y) \tag{3.3}$$

The center of mass is then:

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \tag{3.4}$$

Furthermore, the orientation can be found with (see Figure 3.9):

$$\theta = atan2(m_{01}, m_{10}) \tag{3.5}$$

Scale invariance is partially achieved by using an image pyramid. Each pyramid level consists of a downscaled version of the original image. Keypoints are detected at each level of this pyramid, which makes ORB partially scale invariant. It is partially invariant to scale as the image can only be downsampled. Larger features than the features in the original image will not be detected.

BRIEF descriptors

The BRIEF (Binary Robust Independent Elementary Features) descriptor [7] describes the feature point in a binary feature vector. The feature vector is described as a n_d-dimensional bit string with n_d being the bit size:

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; x_i, y_i) \tag{3.6}$$

where
p = patch of size SxS
and $\tau(p; x_i, y_i)$ is the test at bit i. The test is defined as:

$$\tau(p; x, y) = \begin{cases} 1, & \text{if } p(x) < p(y) \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

where $p(x)$ and $p(y)$ are the pixel intensity values in the smoothed p. These (x, y) pairs are chosen randomly within the patch. The smoothing is done to reduce the effect of noise, as this method uses pixel-level tests.

The (x, y) pairs can be randomly drawn from different distributions. The ORB method uses the Gaussian distribution centered around the key point. ORB also uses a vector length of n = 256.

The ORB method steers BRIEF according to the orientation of the key points to make the BRIEF descriptors invariant to in-plane rotation. A $2xn$ matrix is defined for n binary tests for any feature set at location (x_i, y_i).

$$S = \begin{pmatrix} x_1, \dots x_n \\ y_1, \dots y_n \end{pmatrix} \tag{3.8}$$

A rotation matrix found from the calculated patch orientation in (3.5) is used to transform the matrix to the steered S:

$$S_\theta = R_\theta S \tag{3.9}$$

The steered BRIEF feature vector subsequently becomes:

$$g_n(p, \theta) = f_n(p) | (x_i, y_i) \in S_\theta \tag{3.10}$$

A lookup table is subsequently generated of precomputed BRIEF patterns from discretized angles (per $2\pi/30$) to speed up computation time.

The OpenCV implementation of the ORB feature extractor has different parameters that can be tuned:

- **Number of features** The number of features limits the total number of features found in an image to the value set. More features are better for increasing feature matches. However, it is a trade-off between speed and performance.

- **FAST threshold** The FAST threshold is the parameter that determines when a pixel around a reference pixel is counted as a distinct pixel. When a certain amount of pixels around the reference pixel are distinct, the reference pixel becomes a feature point.
- **Patch size** The patch size is a defined neighborhood around a pixel, which is used to compare the brightness between the reference pixel and the surrounding pixels to detect key points (see Figure 3.8).
- **Scale factor** The scale factor is the factor at which the image is downsampled each level in the pyramid.
- **number of levels** The number of levels represents the layers of the image pyramid. It is the number of times the image is downsampled. Keypoints are detected at each level to detect features at different scales.
- **Edge threshold** The edge threshold is the size of the border where the features are not detected.

3.4. Bags of binary words

ORB-SLAM3 uses a hierarchical bags of binary words model [17] [28] for loop detection. Moreover, as a novelty compared to other bag-of-word approaches, ORB-SLAM uses the bag-of-words vocabulary also to find initial feature matches efficiently by approximating the nearest neighbor distance ratio policy [23] to speed up the feature matching process. The latter is crucial for this research, as this affects how many correspondences can be found for feature matching.

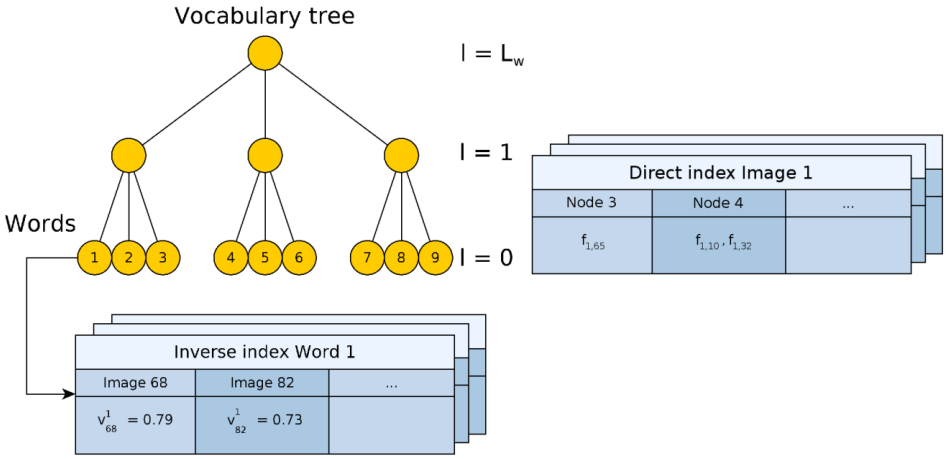


Figure 3.10: example of the hierarchical vocabulary tree and the direct and inverse indexes [17]

.

Creating the vocabulary tree (offline)

The hierarchical bag of words is created using a vocabulary structured as a tree (see Figure 3.10). The vocabulary is created (offline) as follows:

1. ORB features are extracted from approximately 10k training images (Bovisa 2008-09-01 dataset), unrelated to the images used online.
2. The descriptors are discretized in binary clusters by k-medians clustering with k-means++ seeding. The clusters with a value of 0 forms the first level of nodes in the vocabulary tree.
3. More levels are created by repeating this process with the descriptors associated with each node, up to L_w times.

The ORB-SLAM3 vocabulary consists of a vocabulary of 6 levels and 10 clusters per level, with in total of one million words (the leaves in the vocabulary tree). Each word has a weight associated with

the word's relevance in the training set. Words that are seen more frequently are given a lower weight as it is less discriminative.

Converting an image into a bag-of-words vector (online)

ORB features are extracted, and the descriptors of the features traverse the tree from the root to the leaves (top node to leaves in the bottom in Figure 3.10) to convert an image into a bag-of-words vector. At each level, the nodes that minimize the Hamming distance are selected.

An inverse index is maintained by storing for each word in the vocabulary the list of images where the word is present to quickly access the weight of the word in the image. The inverse index is updated every time a new image is added to the online database.

As a novelty compared to other bag-of-word approaches, a direct index is also created and maintained. For each image, the nodes that are ancestors of the words present in the image are stored for each level. A list of local features associated with each node is stored as well. This direct index is used to speed up the feature matching process for both geometrical verification in loop closing candidates, and for feature matching, in general, to find initial correspondences in the TrackWithReferenceKeyframe function (see Chapter 3.2.1). Correspondences between features in two images are found by comparing only the features associated with the same nodes at a preset level in the vocabulary tree.

There is a trade-off between the number of correspondences and computation speed. Setting this preset level to the top node (level 6) means that no speed in computation is gained and that all features are compared. Setting the level to a lower value means that only very similar features are compared (level 0 compares only features belonging to the same word), resulting in less computation time as fewer features are compared and fewer correspondences.

3.5. Stereo visual odometry

The main components of a feature-based visual odometry system can be simplified in a block diagram [15] (see Figure 3.11).

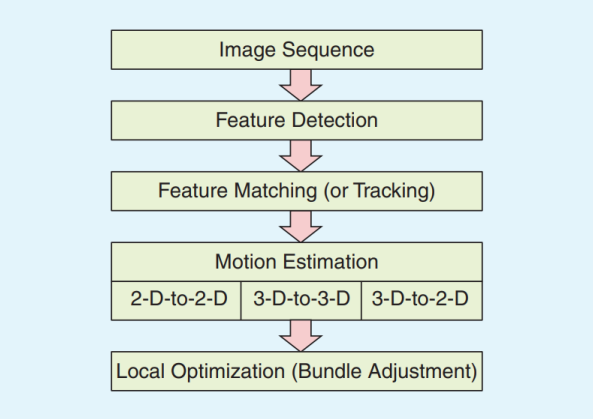


Figure 3.11: Main components of a visual odometry pipeline simplified in a block diagram [15]. Features are detected from the incoming image sequence. The features are matched or tracked. A motion is estimated with the preferred association (3D-2D, 2D-2D or 3D-3D) [15]. The estimation is further optimized with local optimization methods such as bundle adjustment.

Feature matching (or tracking) After features are detected in the images, there are two methods to find corresponding features in the other image; through feature matching or feature tracking. The difference is that when features are matched, the features are detected independently in all images and subsequently matched. With feature tracking, features are detected in one image and tracked in the following images using a local search technique such as correlation, a sum of squared differences (SSD), KanadeLucasTomasi (KLT) tracker, and many more [16].

Motion estimation Most motion estimation methods for stereo cameras use 3D-to-2D feature correspondences [15]. 3D-to-2D correspondences are more accurate than 3D-to-3D correspondences because it minimizes the image re-projection error instead of the feature position error. With 3D-to-2D association, the features in the previous image are in 3D (found through triangulating the 2D features in the stereo image pair), and the features in the current image are the corresponding 2D re-projections

on the current image. The general formulation is to find a transformation T_k that minimizes the image re-projection error:

$$\arg\min_{T_k} \sum_i ||p_k^i - \hat{p}_{k-1}^i||^2 \tag{3.11}$$

where p_k^i is a 2D-feature point in the current image and $\hat{p}_{k-1}^i$ the re-projection of the 3D point in the previous image onto the current image according to the transformation T_k. Estimating a pose by minimizing the re-projection error of a 3D-to-2D correspondence is also known as the Perspective-n-Point (PnP) problem. The motion estimation algorithm is summarized in Figure 3.12.

Algorithm 3. VO from 3-D-to-2-D Correspondences.

1) Do only once:
1.1) Capture two frames I_{k-2}, I_{k-1}
1.2) Extract and match features between them
1.3) Triangulate features from I_{k-2}, I_{k-1}
2) Do at each iteration:
2.1) Capture new frame I_k
2.2) Extract features and match with previous frame I_{k-1}
2.3) Compute camera pose (PnP) from 3-D-to-2-D matches
2.4) Triangulate all new feature matches between I_k and I_{k-1}
2.5) Iterate from 2.1).

Figure 3.12: The 3D-to-2D feature correspondence algorithm [15].

Windowed (or Local) Bundle Adjustment After pose estimation, a bundle adjustment can be performed over the last frames to obtain a more accurate camera pose estimation, and 3D landmarks [16]. A windowed bundle adjustment takes a window of n images and performs parameter optimization of the camera poses and the 3D landmarks for this window. This optimization is done by minimizing the image re-projection error:

$$\arg\min_{X^i, C_k} \sum_{i,k} ||p_k^i - g(X^i, C_k)||^2 \tag{3.12}$$

where p_k^i is the ith image point of the 3D landmark X^i in the kth (current) image and $g(X^i, C_k)$ is the image projection of p_k^i according to the current camera pose C_k. As this optimization is a non-linear function, it is usually solved using the Levenberg-Marquardt algorithm [22].

3.6. Triangulation

Triangulation of 3D points is done with the epipolar geometry (see Figure 3.13). From similar triangles, depth can be written as [45]:

$$Z = \frac{b * f}{x_L - x_R} = \frac{b * f}{d} \tag{3.13}$$

where Z is the depth of the 3D world coordinate P, b is the camera baseline, f is the focal length, x_L and x_R are the x coordinates of the point in the left and right image planes, and d ($d = x_L - x_R$) is the disparity.

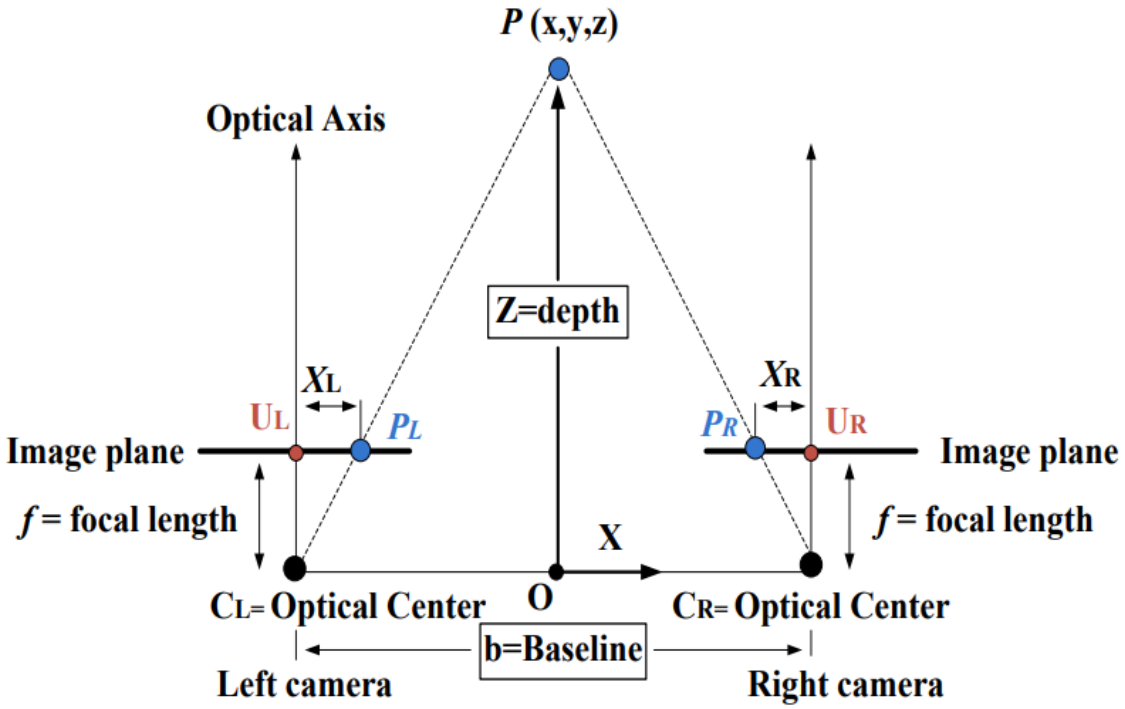


Figure 3.13: Epipolar geometry with parallel optical axes [45].

A correspondence problem has to be solved to determine the disparity of a pixel in an image. These correspondences form the disparity map of an image. The widely used method in OpenCV, called StereoSGBM, is a modified version of the H. Hirschmuller algorithm [19], which solves the correspondences problem with the so-called Semi-Global Matching. The differences between the StereoSGBM and the H. Hirschmuller algorithm are [5]:

- By default, the algorithm is single-pass, which means that only five directions are considered instead of 8 to reduce memory usage.
- By default, the algorithm matches blocks instead of pixels. Setting the blockSize to 1 reduces the blocks to pixels.
- Instead of the mutual cost function, a simpler Birchfield-Tomasi sub-pixel metric [3] is implemented.
- Certain pre- and post-processing steps are included, such as the x-Sobel pre-filter and post-filters, such as a uniqueness check, quadratic interpolation, and speckle filtering.

The OpenCV StereoSGBM has specific parameters that can be tuned. The most trivial parameters are;

- minDisparity, which sets the minimum possible disparity value.
- numDisparity, which is the maximum disparity minus the minimum disparity. This value should be a multiple of 16. Note that high values of numDisparity create a large black column in the disparity map, as no correspondences can be matched to the right image. The camera baseline affects how large this value can be set. The smaller the camera baseline, the larger the disparities that can be calculated.
- blockSize, the block size that is used for matching. A value of 1 matches individual pixels.
- The parameters P1 and P2 control the disparity smoothness. P1 is the penalty for the disparity change of ±1 between neighboring pixels, while P2 is the penalty for more than one between the neighboring pixels.

3.7. Correlation factors and plots

Calculating a correlation factor or plotting data points of one or multiple variables can show if there is a relationship between the variables. There are two widely used correlation coefficients; Spearman's rank correlation coefficient and Pearson's product-moment correlation coefficient [2]. Spearman's correlation coefficient calculates the monotonic correlation between the different variables. Spearman's correlation coefficient is a more general approach than Pearson; while Pearson only describes linear relationships, Spearman can describe monotonic relationships for linear as well as non-linear relationships.

In addition to a correlation coefficient, a visual plot of the data points of the different variables can also show a relationship between the different variables. Each variable can have a different (unknown) distribution. A transformation can be done to transform the data to a normal distribution to determine if there is a relationship between the datasets [33]. For the transformation, percentile rank scores are calculated for each variable (similar to how Spearman's rank correlation is calculated). Subsequently, the inverse of the cumulative distribution function is taken of the percentile rank score to get a z-score. The relationship can subsequently be visualized by plotting the z-scores of the variables against each other.

4

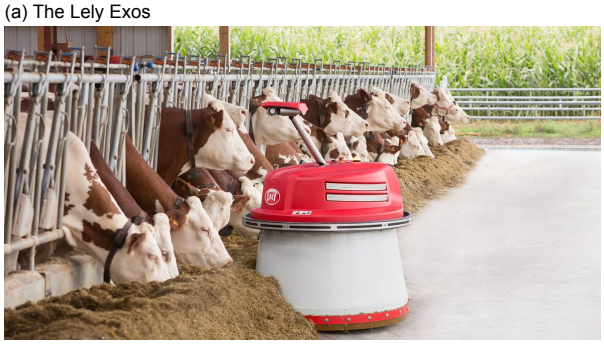
Experiments and Results

4.1. Datasets

Besides using publicly available datasets, we also want to be able to experiment with the Lely data. Different Rosbag recordings are created on two different machines; the Exos and Juno. The machines already save videos with left and right images during their daily operation on the farm. These videos are converted to a ROS1 (Melodic Morenia) [47] Rosbag with a playback function. The Rosbag is used to run the sequences on ORB-SLAM3. The ground truth pose is generated with the current Adaptive Monte Carlo Localization (AMCL) method running on the machines. The ground truth poses are generated with the same frequency as the frame rate; each frame in the video corresponds to a ground truth value with an identical timestamp. The ground truth poses are saved in the TUM format [20]. Recording rotation was not possible. Therefore the quaternion value of each pose point is set to the identity quaternion. The rotation is not necessary for evaluating the positional error, as the x,y, and z positions are used for evaluation. However, evaluating rotational error is not possible with only positional ground truth values.

The Exos operates mainly outside on grass fields, cutting the grass and bringing the freshly cut grass into the barn. The Lely Juno visits each barn and pushes feed to the fences. The Lely Juno operates inside the barn and travels on the outside terrain to the barns. Even though we also experiment on the Exos machine, this research's primary purpose is to localize robustly on the Juno machine visually.

(a) The Lely Exos



(b) The Lely Juno

The Exos dataset consists of recordings on one farm with the same machine. The recordings are made on three separate days in different conditions (rainy, at night, and cloudy weather with occasional sun) and split into smaller sequences (barn environment, grass field, road). Example frames can be seen in Figure 4.2

The Juno dataset consists of recordings on two different farms, thus also on two different Juno machines. On each farm, we have recorded the Juno operating the entire trajectory in 'sunny,' 'dark,' and 'after rain' conditions. On one farm, we also have 'during rain' conditions. An entire sequence consists of the following actions; detaching from the charging station, traveling to a barn, pushing feed inside the barn and traveling to other barns, and finally back to the charging station. Example frames can be seen in Figure 4.3.

Properties of all datasets used in the experiments are summarized in Table 4.1 (own datasets and publicly available datasets that are commonly used in the literature).

(a) The 'barn dark' sequence

(b) The 'barn day' sequence

(c) The 'grass dark' sequence

(d) The 'grass normal' sequence

(e) The 'grass rain' sequence

(f) The 'road dark' sequence

(g) The 'road normal' sequence

(h) The 'road rain' sequence

Figure 4.2: Example frames in the sequences of the Exos dataset

(a) The 'den Boer sunny' sequence

(b) The 'den Boer rain' sequence

(c) The 'den Boer dark' sequence

(d) The 'den Boer after rain' sequence

(e) The 'van Adrichem sunny' sequence

(f) The 'van Adrichem dark' sequence

(g) The 'van Adrichem after rain' sequence

Figure 4.3: Example frames in the sequences of the Juno dataset

Table 4.1: Overview of all datasets and their properties

Dataset	Width (px)	Height (px)	Frame rate (Hz)
4Seasons	800	400	30
EuRoC MAV	752	480	20
KITTI	1241	376	10
Flourish	752	480	25
Rosario	672	376	15
Lely-Juno	720	480	10
Lely-Exos	672	376	15

4.2. Effect of RANSAC

ORB-SLAM3 does not use RANSAC in its feature-matching pipeline. ORB-SLAM3 uses a BoW model to match only the descriptors with similar visual words, which can be seen as a replacement for the brute force k-NN matcher with RANSAC filtering. This is done because the BoW model can match and filter matches faster than the standard Brute Force (with k-NN) matcher and RANSAC. To determine if RANSAC is needed in the OpenCV pipeline to filter wrong matches, an ORB extractor with the same settings as the setting in ORB-SLAM3 ($nfeatures = 2000$, $scaleFactor = 1.2$, $nlevels = 8$, $edgeThreshold = 19$, $firstLevel = 0$, $WTA_K = 2$, $scoreType = HARRIS\_SCORE$, $patchSize = 31$, $fastThreshold = 40$ or 14) is implemented. The matcher is a brute-force matcher combined with a k-nearest neighbor, which is set to k=2. We filter the two nearest neighbors with the Lowe ratio test where the ratio is equal to 0.75, which removes most incorrect matches (Figure 4.4) [23].

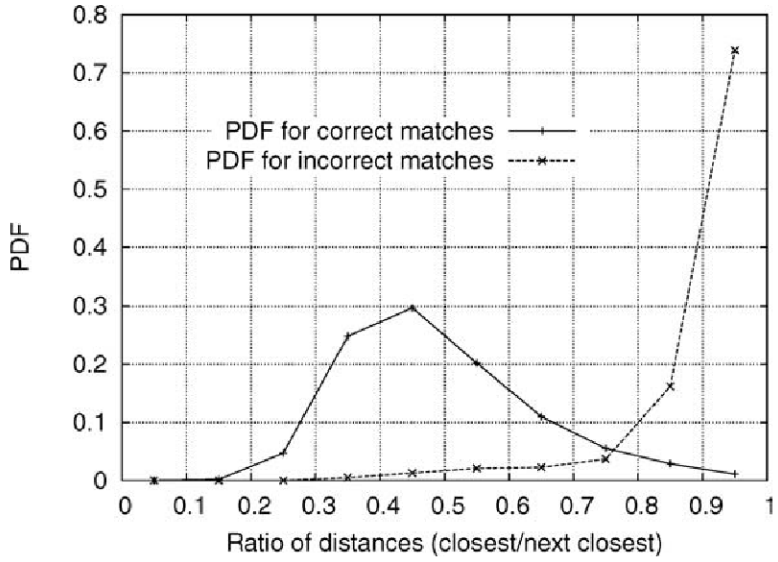


Figure 4.4: The probability that a match is correct (solid line) or incorrect (dotted line) based on a database of 40000 key points [23]

4.2.1. Results

Figure 4.5 shows a visible result of our implementation. We can see that after using Lowe's filtering method, there are still a lot of wrong matches (see the diagonal matches). Using RANSAC can help filter these out, which we did not initially add because ORB-SLAM3 does not use RANSAC at the stage where the failures happen in the TrackReferenceKeyframe function.

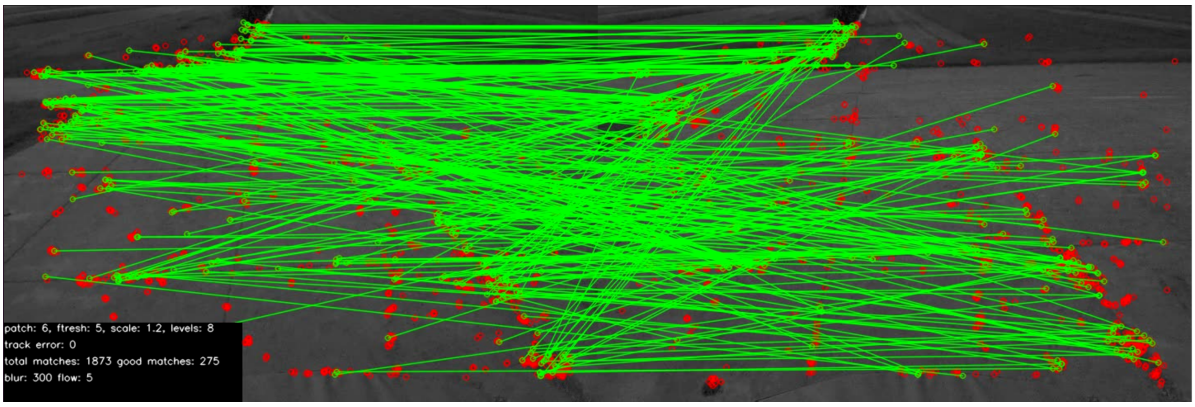


Figure 4.5: Matches made with the OpenCV ORB in an image of the Juno that failed in ORB-SLAM3. Outliers are only filtered with Lowe's ratio test.

After adding a RANSAC filter (with $ransacReprojThreshold = 5$), it is clear that most of these matches are false matches (see Figure 4.6). The amount of inliers after RANSAC filtering drops to 42. Because the difference is significant, RANSAC is added to the OpenCV pipeline.

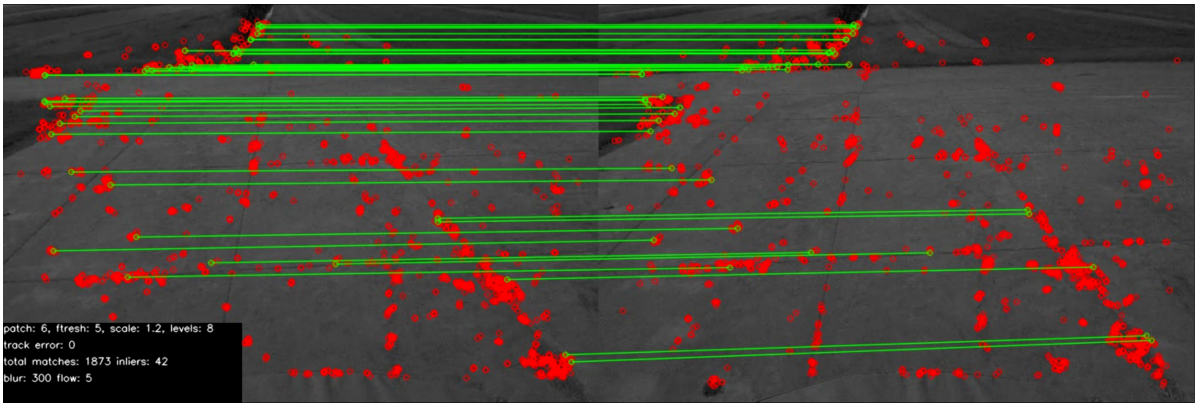


Figure 4.6: Matches made with the OpenCV ORB in an image of the Juno that failed in ORB-SLAM3. Outliers are filtered with Lowe's ratio test and RANSAC.

4.3. Effect of ORB settings on matching performance

Tuning the ORB parameters is essential for finding distinct key points affecting the matching performance. To determine which settings are essential for the performance of the feature matching and to determine settings that work for all datasets without a greedy search, the effect of the settings on the mean and minimum matching performance for each parameter is investigated in the OpenCV pipeline. For each sequence in each dataset, the mean and the minimum amount of inliers are recorded for different parameter settings. The results are summarized as follows;

- the average mean and the average minimum inliers for each setting across all sequences and their standard deviation
- the coefficient of variance across the settings for both the mean and minimum matches

We keep two parameters fixed; the number of features and the edge threshold. The number of features is kept at 2000, which should be enough for large images such as the images in the KITTI dataset. We also determined that ORB-SLAM3 can run in real time with the number of features set to 2000. For the edge threshold, we follow the recommendation of the OpenCV ORB documentation to set the value roughly equal to the patch size. We set the edge threshold to the exact value of the patch size parameter.

The number of inliers is recorded for each sequence of each dataset. This is done by an ordered tuning approach. We start by changing the FAST threshold while keeping all other parameters fixed to the default value of OpenCV ORB. Subsequently, we try out different patch sizes while using the best FAST threshold based on the maximum average minimum inliers for most datasets and keeping all other parameters fixed to the default value. We continue the same approach for the scale factor and number of levels. The reason for this ordered approach is that some default values only work well for some datasets. Specific orders result in zero average mean and average minimum matches no matter the change in the setting. We have discovered that tuning the FAST threshold first and patch size second prevents this from happening. The mean inliers for each sequence and setting are averaged and represented by a single average mean value for each dataset, including their standard deviations. The same is done for the minimum amount of inliers.

We compare different settings of ORB and the feature extractor for each parameter separately. We experiment twice, first on the original images and second on the images that are equalized with histogram equalization. The results can be found in the following subsections.

4.3.1. Results on the original images

FAST threshold The results in Table 4.2 show that, in general, a low FAST threshold setting results in more mean inliers and a higher minimum in the inliers. When looking at the coefficient of variance across the settings, the average mean and minimum inliers in most datasets are not affected much by the FAST threshold setting, especially on the KITTI dataset. Both of our datasets (Exos and Juno) are the most sensitive to the FAST threshold setting. This sensitivity might be explained by low-contrast

images in our datasets, which we further investigated (see Chapter 4.3.2) by applying histogram equalization on the images to increase the contrast.

There is also a significant difference between the frames in the sequences of some datasets; The average minimum amount of inliers deviates significantly from the mean in the Exos and Juno datasets, which deviates more than its average value. Euroc, Flourish, and the 4Seasons datasets are easy to the feature-matcher based on the average mean inlier values. Based on the absolute best average minimum matches for each dataset, ORB performs the best on Rosario, Flourish, and the 4Seasons datasets. The Exos and Juno datasets contain, on average, complex frames in the sequences for the ORB to match features successfully.

Table 4.2: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for different FAST thresholds. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Fast threshold	**5**	**10**	**20**	**40**	**coeff of variance of the setting**
EUROC	**avg mean**	**659.8 ±105.6**	652.0 ±113.4	605.6 ±152.0	459.4 ±200.5	0.2
	avg min	**65.6 ±44.0**	54.6 ±35.7	39.5 ±34.7	14.3 ±14.7	0.5
KITTI	**avg mean**	**316.3 ±58.8**	316.1 ±58.8	315.2 ±59.5	297.3 ±66.0	0.0
	avg min	**74.0** ±37.7	**72.9** ±37.0	72.8 ±38.6	74.0 ±38.0	0.0
ROSARIO	**avg mean**	433.0 ±87.6	**433.9 ±87.9**	418.5 ±82.4	329.5 ±72.0	0.1
	avg min	129.8 ±53.7	**133.2 ±59.5**	127.7 ±52.8	98.0 ±48.8	0.1
FLOURISH	**avg mean**	786.2 ±5.5	786.0 ±6.9	**786.4 ±5.8**	659.4 ±9.5	0.1
	avg min	**444.5 ±149.2**	435.5 ±132.2	444.0 ±116.0	164.5 ±21.9	0.4
EXOS	**avg mean**	**295.0 ±241.8**	241.1 ±225.9	87.7 ±121.0	9.0 ±14.0	0.8
	avg min	**59.4 ±101.9**	38.6 ±83.5	4.3 ±12.0	0.0 ±0.0	1.1
JUNO	**avg mean**	**540.3 ±234.9**	498.7 ±250.8	357.4 ±274.5	155.7 ±170.1	0.4
	avg min	**64.3 ±82.1**	50.9 ±81.0	27.7 ±50.2	6.6 ±12.6	0.7
4SEASONS	**avg mean**	**744.0 ±126.9**	731.5 ±132.6	666.3 ±146.4	465.2 ±166.9	0.2
	avg min	**161.0 ±83.6**	152.1 ±81.9	111.3 ±62.4	34.2 ±30.5	0.5

Patch size The results in Table 4.3 show the average mean and average minimum inliers for different patch sizes on each dataset. All datasets seem to be affected comparably for all datasets; The coefficient of variance of the various patch sizes is relatively close for all datasets, between 0.4 and 0.6, except for 0.3 for the average mean inliers of the Juno dataset. The value is not negligibly small, so tuning the patch size is still important. Patch size of 48 works for almost all datasets based on the average mean and minimum inliers. Which can mean that a large patch results in more distinct features as you look at a larger patch of the image, which can be related to the scale of the subjects in the image. The average minimum matches are not always optimal at patch size 48. However, the optimal value does not seem to differ much from the value at patch size 24, so a patch size of 48 is still a good setting for the patch size.

Table 4.3: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for different patch sizes. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Patch size	6	12	24	48	coeff of variance of the setting
EUROC	**avg mean**	236.6 ±44.2	397.4 ±62.9	602.0 ±88.1	**719.5 ±129.9**	0.4
	avg min	17.9 ±8.6	40.5 ±24.0	**73.1 ±42.9**	54.5 ±33.3	0.5
KITTI	**avg mean**	105.3 ±24.1	186.6 ±35.5	287.1 ±52.8	**296.9 ±63.0**	0.4
	avg min	16.1 ±11.1	47.5 ±23.1	**71.5 ±35.3**	68.8 ±32.4	0.5
ROSARIO	**avg mean**	124.4 ±33.0	237.0 ±52.5	379.4 ±75.9	**437.3 ±100.6**	0.5
	avg min	26.8 ±11.7	83.0 ±27.8	**121.5 ±44.9**	101.7 ±59.3	0.5
FLOURISH	**avg mean**	174.3 ±11.7	390.5 ±11.5	673.9 ±10.1	**888.3 ±12.0**	0.6
	avg min	72.5 ±14.8	206.5 ±60.1	383.5 ±84.1	**480.0 ±169.7**	0.6
EXOS	**avg mean**	98.1 ±82.4	185.3 ±138.8	**278.3 ±207.4**	254.2 ±244.4	0.4
	avg min	19.1 ±28.4	46.3 ±61.6	**68.6 ±94.0**	50.1 ±90.7	0.4
JUNO	**avg mean**	250.2 ±116.7	365.9 ±153.8	501.9 ±205.3	**538.2 ±248.8**	0.3
	avg min	22.7 ±27.1	43.3 ±48.2	**60.9 ±68.8**	51.0 ±65.5	0.4
4SEASONS	**avg mean**	310.8 ±121.0	469.2 ±127.2	666.0 ±126.7	**735.7 ±129.8**	0.4
	avg min	31.5 ±22.1	90.3 ±39.7	155.0 ±71.6	**162.9 ±85.4**	0.6

Scale factor The scale factor does not seem to affect the datasets much, as the coefficient of variance across the settings is very low. The default scale factor of 1.2, recommended in the OpenCV implementation of ORB, works well for all datasets based on the average mean and average min matches.

Table 4.4: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for different scale factors. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Scale factor	1.1	1.2	1.3	1.4	coeff of variance of the setting
EUROC	**avg mean**	**586.9 ±126.1**	570.9 ±140.2	549.8 ±147.8	518.8 ±143.7	0.1
	avg min	67.5 ±41.9	**74.2 ±42.5**	62.7 ±39.9	55.8 ±37.8	0.1
KITTI	**avg mean**	**316.3 ±54.8**	291.1 ±55.0	262.1 ±53.0	241.3 ±50.1	0.1
	avg min	70.4 ±39.1	**74.7 ±35.1**	66.1 ±33.9	56.5 ±32.3	0.1
ROSARIO	**avg mean**	**399.8** ±81.8	395.0 ±95.2	373.0 ±82.9	348.6 ±79.1	0.1
	avg min	**141.8 ±55.3**	125.0 ±51.5	111.7 ±41.8	103.8 ±34.3	0.1
FLOURISH	**avg mean**	857.8 ±14.7	**888.3 ±12.0**	788.0 ±15.1	737.4 ±16.3	0.1
	avg min	478.5 ±217.1	**480.0 ±169.7**	435.5 ±147.8	419.5 ±125.2	0.1
EXOS	**avg mean**	**293.9 ±214.9**	267.9 ±218.7	241.9 ±216.6	225.3 ±208.5	0.1
	avg min	**79.3 ±104.8**	68.9 ±93.8	53.4 ±74.4	48.3 ±74.6	0.2
JUNO	**avg mean**	**479.3 ±208.3**	466.9 ±220.4	441.4 ±227.6	413.0 ±216.1	0.1
	avg min	60.7 ±67.7	**61.0 ±68.1**	53.1 ±61.4	53.0 ±66.1	0.1
4SEASONS	**avg mean**	**710.4 ±130.6**	704.6 ±134.2	649.3 ±128.1	607.2 ±121.4	0.1
	avg min	161.9 ±85.0	**172.1 ±82.1**	151.1 ±73.8	143.1 ±74.2	0.1

Number of levels The coefficient of variance across the settings shows that these parameters do not affect the datasets much. The average mean and minimum of inliers are quite close for all settings. The number of levels can be set to 8 for all datasets, resulting in a reasonable amount of inliers. At the same time, it is also computationally less expensive than setting the number of levels to 12 or 16.

Table 4.5: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for the different number of levels. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Number of levels	4	8	12	16	coeff of variance of the setting
EUROC	**avg mean**	553.8 ±151.8	574.7 ±143.4	**575.3 ±143.4**	564.4 ±142.9	0.0
	avg min	50.6 ±28.4	76.4 ±43.9	81.8 ±42.3	**85.5 ±44.0**	0.2
KITTI	**avg mean**	280.4 ±59.0	**298.1 ±61.6**	291.7 ±68.0	283.9 ±68.4	0.0
	avg min	68.4 ±37.3	**77.5 ±35.2**	72.3 ±32.6	71.0 ±36.5	0.1
ROSARIO	**avg mean**	344.4 ±76.3	399.8 ±81.8	433.9 ±87.5	**444.1 ±76.8**	0.1
	avg min	129.2 ±44.1	141.8 ±55.3	150.0 ±62.2	**150.7 ±49.2**	0.1
FLOURISH	**avg mean**	836.0 ±14.4	**874.0 ±8.2**	834.1 ±63.8	822.8 ±109.5	0.0
	avg min	491.5 ±125.2	**496.0 ±192.3**	455.0 ±200.8	435.5 ±222.7	0.1
EXOS	**avg mean**	265.4 ±209.7	293.4 ±215.6	305.0 ±219.2	**309.5 ±220.6**	0.1
	avg min	70.5 ±105.5	79.4 ±104.7	**81.6 ±105.5**	76.6 ±93.7	0.1
JUNO	**avg mean**	434.1 ±219.6	476.6 ±211.7	**484.9 ±214.2**	482.3 ±213.4	0.1
	avg min	**65.0 ±81.3**	63.9 ±69.9	61.4 ±63.0	62.3 ±66.8	0.1
4SEASONS	**avg mean**	679.7 ±146.5	**707.0 ±134.2**	691.2 ±127.7	665.0 ±117.9	0.0
	avg min	159.7 ±81.0	**177.9 ±83.4**	168.4 ±78.2	159.8 ±73.2	0.1

4.3.2. Results on the equalized images

The previous experiment is repeated with images on which histogram equalization [1] is applied before extracting and matching features. Histogram equalization improves contrast in the image by stretching the intensity range in the image. Histogram equalization is used to make the images from the different datasets more equal.

FAST threshold The results of the effect of the FAST threshold settings on the matching performance for equalized images can be seen in Table 4.6. The coefficient of variance of the setting is lower for all datasets compared to the coefficient for non-equalized images (see Table 4.2). This result means that the datasets are less sensitive to the FAST threshold setting, including the Exos and Juno datasets which were highly sensitive during the previous experiment (see Chapter 4.3.1). The absolute average minimum inliers affect the KITTI and the Juno datasets negatively. All other datasets do not show a significant change.

Table 4.6: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for different FAST thresholds on the equalized images. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Fast threshold	5	10	20	40	coeff of variance of the setting
EUROC	**avg mean**	**662.0 ±122.0**	660.2 ±123.9	641.3 ±141.7	534.3 ±189.4	0.1
	avg min	**50.4 ±32.9**	50.1 ±32.9	48.3 ±35.3	21.5 ±18.5	0.3
KITTI	**avg mean**	261.2 ±53.8	**261.5 ±53.9**	261.2 ±53.9	250.7 ±52.0	0.0
	avg min	36.2 ±24.4	**37.4** ±26.2	36.9 ±24.9	35.0 ±23.7	0.0
ROSARIO	**avg mean**	411.6 ±89.7	411.3 ±89.2	**412.2 ±89.1**	384.0 ±76.2	0.0
	avg min	125.2 ±62.6	**126.0 ±60.3**	120.8 ±58.7	113.3 ±50.5	0.0
FLOURISH	**avg mean**	**812.8 ±14.6**	811.1 ±14.5	811.1 ±13.3	810.9 ±14.2	0.0
	avg min	**555.5 ±0.7**	552.5 ±10.6	526.0 ±0.0	538.0 ±4.2	0.0
EXOS	**avg mean**	**307.8 ±247.5**	307.5 ±247.6	294.2 ±240.2	212.6 ±183.9	0.2
	avg min	**68.4 ±99.4**	66.4 ±95.8	57.6 ±95.7	19.5 ±24.0	0.4
JUNO	**avg mean**	**490.0 ±238.6**	489.2 ±239.1	471.4 ±243.4	358.4 ±214.3	0.1
	avg min	**20.0 ±15.1**	19.3 ±16.2	18.4 ±17.1	11.0 ±10.4	0.2
4SEASONS	**avg mean**	721.2 ±116.4	**721.4 ±116.2**	715.7 ±119.5	630.1 ±150.3	0.1
	avg min	**157.4 ±74.4**	155.8 ±73.8	151.6 ±79.9	122.0 ±84.8	0.1

Patch size Equalizing the images with histogram equalization did not affect the response of the datasets to different patch sizes; the coefficient of variance of the settings did not change significantly for both the average mean and average minimum inliers compared to the results of non-equalized images. The absolute values of the average minimum inliers are affected negatively for the KITTI, Rosario, and Juno datasets. All other datasets do not show a significant change compared to the results of non-equalized images.

Table 4.7: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for different patch sizes on the equalized images. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Patch size	**6**	**12**	**24**	**48**	**coeff of variance of the setting**
EUROC	**avg mean**	227.5 ±44.0	392.4 ±70.9	597.1 ±102.5	**707.9 ±144.8**	0.4
	avg min	13.2 ±5.0	35.2 ±22.4	**61.5 ±36.5**	43.5 ±27.4	0.5
KITTI	**avg mean**	78.6 ±21.8	152.7 ±32.4	236.1 ±46.9	**250.8 ±56.8**	0.4
	avg min	8.9 ±1.7	21.9 ±18.8	**36.6 ±25.6**	34.0 ±20.9	0.5
ROSARIO	**avg mean**	112.4 ±33.5	219.9 ±53.0	355.9 ±79.0	**415.3 ±112.0**	0.5
	avg min	17.2 ±13.1	73.8 ±28.7	**116.3 ±52.1**	95.8 ±58.2	0.6
FLOURISH	**avg mean**	180.8 ±12.1	408.4 ±13.8	691.4 ±15.5	**913.6 ±16.1**	0.6
	avg min	87.5 ±4.9	252.0 ±15.6	440.5 ±16.3	**614.5 ±23.3**	0.7
EXOS	**avg mean**	104.6 ±88.0	192.3 ±143.0	**285.8 ±210.7**	269.2 ±254.1	0.4
	avg min	19.5 ±30.2	49.9 ±58.9	**74.0 ±95.2**	50.5 ±85.4	0.5
JUNO	**avg mean**	217.6 ±109.8	323.4 ±152.3	447.9 ±207.3	**498.3 ±237.8**	0.3
	avg min	9.7 ±2.9	15.6 ±8.9	**23.9 ±16.8**	16.9 ±12.3	0.4
4SEASONS	**avg mean**	274.2 ±107.3	435.9 ±116.3	632.6 ±119.6	**729.2 ±127.4**	0.4
	avg min	30.7 ±18.4	87.8 ±40.0	146.7 ±63.5	**157.6 ±88.9**	0.6

Scale factor Histogram equalization did not affect the response of the datasets to different scale factors either; the coefficient of variance of the settings did not change significantly for both the average mean and average minimum inliers compared to the results of non-equalized images. The absolute values of the average mean and minimum inliers did not change significantly for most datasets compared to the non-equalized images except for the KITTI, Flourish, and Juno datasets. The KITTI and Juno datasets are affected negatively regarding average minimum inliers, while the Flourish dataset is affected positively.

Table 4.8: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for different scale factors on the equalized images. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Scale factor	**1.1**	**1.2**	**1.3**	**1.4**	**coeff of variance of the setting**
EUROC	**avg mean**	**566.2 ±147.4**	553.1 ±162.6	529.1 ±168.4	496.0 ±162.5	0.0
	avg min	53.3 ±30.7	**61.9 ±36.0**	56.6 ±36.9	48.4 ±29.6	0.1
KITTI	**avg mean**	**258.9 ±49.1**	245.1 ±51.6	222.4 ±48.4	204.6 ±45.4	0.1
	avg min	31.5 ±25.2	**40.3 ±25.1**	33.5 ±21.7	29.3 ±20.2	0.1
ROSARIO	**avg mean**	**360.2 ±66.6**	355.9 ±79.0	344.7 ±82.8	322.4 ±77.4	0.0
	avg min	**120.8 ±62.0**	116.3 ±52.1	99.3 ±48.6	96.5 ±47.0	0.1
FLOURISH	**avg mean**	879.4 ±20.6	**913.6 ±16.1**	813.8 ±13.0	764.7 ±13.7	0.1
	avg min	**627.5 ±46.0**	614.5 ±23.3	516.0 ±14.1	480.0 ±31.1	0.1
EXOS	**avg mean**	**295.3 ±213.9**	275.9 ±221.9	253.3 ±225.0	232.4 ±212.8	0.1
	avg min	**85.3 ±102.2**	74.3 ±94.9	57.8 ±73.0	52.1 ±79.6	0.2
JUNO	**avg mean**	426.1 ±218.3	**428.0 ±223.3**	404.5 ±226.3	376.9 ±214.5	0.1
	avg min	17.9 ±11.9	**24.9 ±15.7**	22.4 ±13.3	18.9 ±12.9	0.2
4SEASONS	**avg mean**	670.9 ±141.2	**682.7 ±148.5**	628.5 ±128.7	584.8 ±121.3	0.1
	avg min	163.7 ±81.8	**168.4 ±83.1**	149.9 ±76.0	138.5 ±71.5	0.1

Number of levels Tabel 4.9 shows that the number of levels parameter is not affected by the histogram equalization in the same manner as the scale factor and patch size. The absolute values of the average mean and minimum inliers did not change significantly for most datasets compared to the non-equalized images except for the KITTI, Flourish, and Juno datasets. The KITTI and Juno datasets are affected negatively regarding average minimum inliers, while the Flourish dataset is affected positively.

Table 4.9: The average and standard deviation of the mean inliers (noted as avg mean ±std) and minimum inliers (noted as avg min ±std) of each dataset are shown for different numbers of levels on the equalized images. Also, the coefficient of variance of the average mean and average minimum inliers across all settings is noted (last column).

	Number of levels	**4**	**8**	**12**	**16**	**coeff of variance of the setting**
EUROC	**avg mean**	527.0 ±165.1	**558.4 ±151.6**	558.3 ±142.9	542.0 ±126.9	0.0
	avg min	43.7 ±28.0	64.5 ±37.7	66.6 ±37.1	**67.0 ±39.5**	0.2
KITTI	**avg mean**	228.6 ±50.5	**243.3 ±52.3**	230.8 ±51.7	221.7 ±48.9	0.0
	avg min	30.1 ±20.9	**40.7 ±24.9**	36.7 ±24.2	36.8 ±26.3	0.1
ROSARIO	**avg mean**	310.3 ±63.9	353.6 ±73.1	374.5 ±90.1	**386.3 ±106.3**	0.1
	avg min	104.7 ±43.6	124.7 ±57.8	139.3 ±66.1	**140.2 ±75.3**	0.1
FLOURISH	**avg mean**	868.5 ±4.6	**898.1 ±5.8**	869.3 ±54.1	859.0 ±102.0	0.0
	avg min	614.0 ±15.6	**629.0 ±43.8**	589.5 ±57.3	573.0 ±56.6	0.0
EXOS	**avg mean**	258.3 ±205.8	288.8 ±216.0	303.1 ±223.8	**309.3 ±228.9**	0.1
	avg min	69.0 ±91.7	85.8 ±101.8	88.4 ±98.6	**89.3 ±92.7**	0.1
JUNO	**avg mean**	382.4 ±227.8	**418.3 ±205.7**	405.5 ±188.8	397.6 ±183.8	0.0
	avg min	18.7 ±16.7	26.0 ±16.3	**27.1 ±18.1**	26.0 ±16.9	0.2
4SEASONS	**avg mean**	644.4 ±156.7	**676.3 ±148.8**	651.2 ±136.8	628.1 ±130.6	0.0
	avg min	156.3 ±81.2	**175.6 ±86.7**	157.1 ±75.0	152.1 ±71.4	0.1

4.4. Qualitative results and discussion

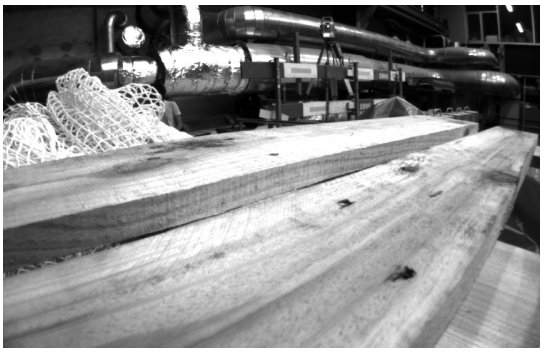
Using the ORB extraction parameters that maximize the average minimum inliers ($nfeatures = 2000$, $scaleFactor = 1.2$, $nlevels = 8$, $edgeThreshold = patchSize$, $firstLevel = 0$, $patchSize = 48$, $fastThreshold = 5$), example frames are saved when the inliers were at the maximum of the sequence and the minimum for both the regular and equalized images.

Figure 4.7b of the MH01 sequence of the Euroc dataset shows that images with less texture cause a drop in the features extracted. However, in this example, the minimum features of 1550 in Figure 4.7b are still close to the aimed value of 2000 features. The minimum inliers seem to be caused by blur in Figure 4.7f and 4.7d. After equalization, the images where the maximum and minimum features and the maximum and minimum inliers are found did not change (Figures 4.7g - 4.7j). Also, the number of features extracted and matched is similar to the non-equalized images. The most significant impact of equalization seems to be on the number of features extracted in Figure 4.7b. The extracted features increased to 1850 in Figure 4.7h.

Figure 4.7: Qualitative examples of the MH01 sequence of the Euroc dataset

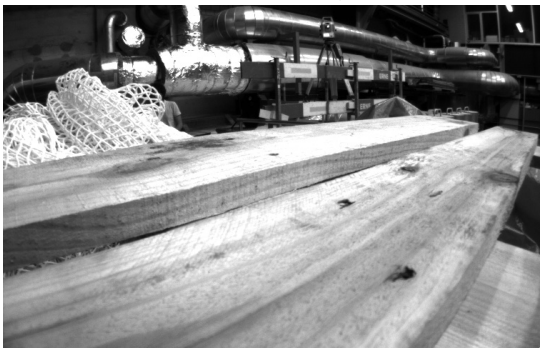
(a) Maximum features of 2000

(b) Minimum features of 1550

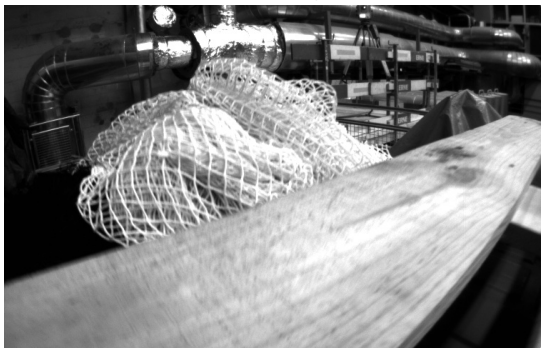


(c) Maximum inliers of 1839, current image of the pair

(d) Minimum inliers of 62, current image of the pair



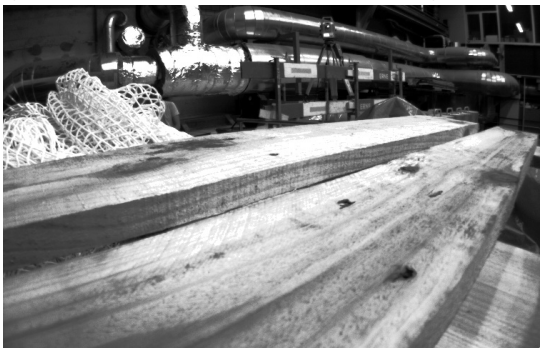
(e) Maximum inliers of 1839, previous image of the pair



(f) Minimum inliers of 62, previous image of the pair

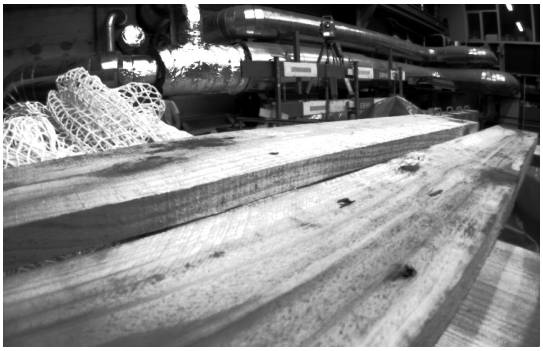
(g) Maximum features of 2000 on the equalized images

(h) Minimum features of 1850 on the equalized images

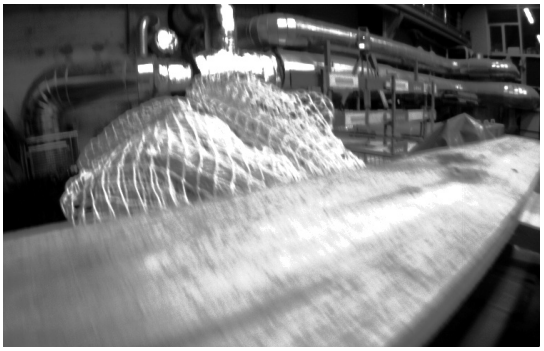


(i) Maximum inliers of 1836 on the equalized images, current image of the pair

(j) Minimum inliers of 65 on the equalized images, current image of the pair



(k) Maximum inliers of 1836 on the equalized images, previous image of the pair



(l) Minimum inliers of 65 on the equalized images, previous image of the pair

The ORB extractor can extract the aimed amount of features in the images of sequence A of the Flourish dataset as the maximum and minimum features are 2000 features. The number of inliers drops to 600 at the minimum (Figures 4.7f and 4.7d). The previous and the current image are very similar, and there is no apparent difference between the images of the maximum inliers (Figures 4.7e and 4.7c) that could explain this drop. Although there is a drop in inliers, it is not problematically low. After equalization, the images where the maximum and minimum features and the maximum and minimum inliers are found did not change (Figures 4.8g - 4.8j). Also, the number of features extracted and matched is similar to the non-equalized images.

Figure 4.8: Qualitative examples of sequence A of the Flourish dataset

(a) Maximum features of 2000

(b) Minimum features of 2000

(c) Maximum inliers of 1680, current image of the pair

(d) Minimum inliers of 600, current image of the pair

(e) Maximum inliers of 1680, previous image of the pair

(f) Minimum inliers of 600, previous image of the pair

(g) Maximum features of 2000 on the equalized images

(h) Minimum features of 2000 on the equalized images

(i) Maximum inliers of 1683 on the equalized images, current image of the pair

(j) Minimum inliers of 563 on the equalized images, current image of the pair

(a) Maximum inliers of 1683 on the equalized images, previous image of the pair

(b) Minimum inliers of 563 on the equalized images, previous image of the pair

The results on the 00 sequence of the KITTI dataset show that the minimum inliers can drop significantly depending on the scene. In Figure 4.10c, the maximum amount of inliers of 1708 is found, while the amount of inliers in Figure 4.10d drops to 52. A possible cause can be the reflections on the road, which seem to result in overexposure in parts of the image. Applying histogram equalization results in a different image with the maximum number of features in the sequence (Figure 4.10a compared to Figure 4.10g). This result means that an image that did not have 2000 features and is temporally seen before Figure 4.10a has 2000 features after equalization. The minimum inliers found after equalization has dropped to 13 (Figure 4.10j) on a different scene. In this case, histogram equalization had a negative effect; a scene that did not have such a low amount of inliers in the non-equalized images now has the worst amount of inliers of the whole scene. Visually, the image looks noisy. Histogram equalization might have increased the noise in this image, which decreased the matching performance.

Figure 4.10: Qualitative examples of the 00 sequence of the KITTI dataset

(a) Maximum features of 2000

(b) Minimum features of 1896

(c) Maximum inliers of 1708, current image of the pair

(d) Minimum inliers of 52, current image of the pair

(e) Maximum inliers of 1708, previous image of the pair

(f) Minimum inliers of 52, previous image of the pair

(g) Maximum features of 2000 on the equalized images

(h) Minimum features of 1910 on the equalized images

(i) Maximum inliers of 1703 on the equalized images, current image of the pair

(j) Minimum inliers of 13 on the equalized images, current image of the pair

(k) Maximum inliers of 1703 on the equalized images, previous image of the pair

(l) Minimum inliers of 13 on the equalized images, previous image of the pair

The drop in inliers for the rainy barn sequence of the Exos dataset can not be explained from the qualitative examples. A very similar pathway with the same lighting conditions has 1604 inliers in one frame (Figure 4.11c) and only 272 inliers in the other frame (Figure 4.11d). Equalizing the images does not affect the number of features and inliers much; minor differences in the number of inliers and features extracted cause various frames to appear in the equalized results.

Figure 4.11: Qualitative examples of the rainy barn sequence of the Exos dataset

(a) Maximum features of 1982

(b) Minimum features of 1903

(c) Maximum inliers of 1604, current image of the pair

(d) Minimum inliers of 272, current image of the pair

(e) Maximum inliers of 1604, previous image of the pair

(f) Minimum inliers of 272, previous image of the pair

(g) Maximum features of 2000 on the equalized images

(h) Minimum features of 1970 on the equalized images

(i) Maximum inliers of 1653 on the equalized images, current image of the pair

(j) Minimum inliers of 255 on the equalized images, current image of the pair

(k) Maximum inliers of 1653 on the equalized images, previous image of the pair

(l) Minimum inliers of 255 on the equalized images, previous image of the pair

The drop in amount of inliers in the den Boer sunny sequence of the Juno dataset is dramatic (Figure 4.12d) compared to the maximum amount of inliers of 1869 that could be found in Figure 4.12c. The reason seems to be a textureless floor in combination with a lot of shadows. Equalization did not help to improve the minimum number of inliers nor the frame at which the minimum number of inliers is present. Equalization did, however, help to increase the number of features that could be extracted. Figure 4.12a is no longer the frame with the minimum amount of features extracted in the sequence.

Figure 4.12: Qualitative examples of the den Boer sunny sequence of the Juno dataset

(a) Maximum features of 2000

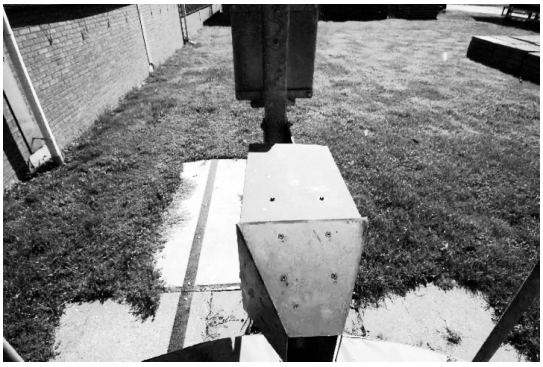
(b) Minimum features of 1113

(c) Maximum inliers of 1869, current image of the pair

(d) Minimum inliers of 15, current image of the pair

(e) Maximum inliers of 1869, previous image of the pair

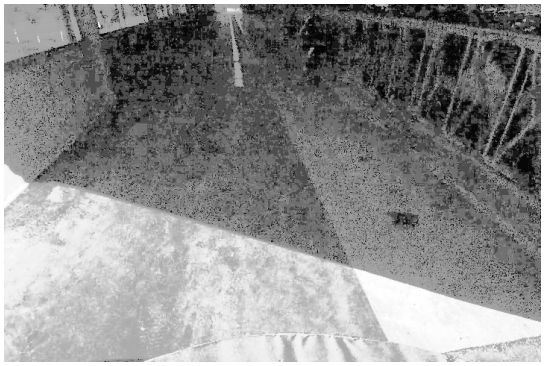
(f) Minimum inliers of 15, previous image of the pair



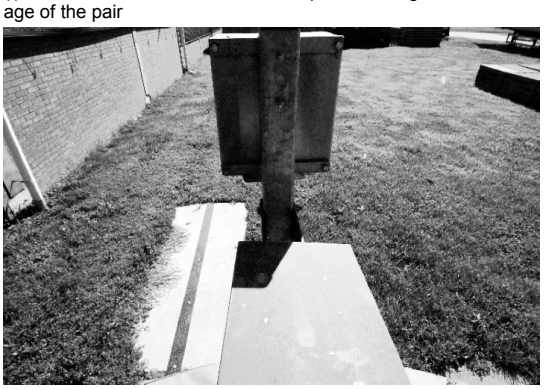
(g) Maximum features of 2000 on the equalized images

(h) Minimum features of 1925 on the equalized images

(i) Maximum inliers of 1892 on the equalized images, current image of the pair



(j) Minimum inliers of 15 on the equalized images, current image of the pair



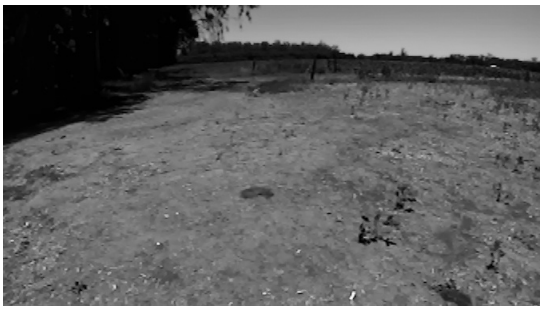
(k) Maximum inliers of 1892 on the equalized images, previous image of the pair

(l) Minimum inliers of 15 on the equalized images, previous image of the pair

Sequence 1 of the Rosario dataset also shows significant drops in the number of inliers. However, there are no significant differences between the image with the maximum inliers (Figure 4.13c) and the image with the minimum inliers (Figure 4.13d) to hypothesize about the reason for this drop. Equalization does not show any significant improvements. The minimum amount of inliers has dropped to 35 for a different frame (Figure 4.13j).

Figure 4.13: Qualitative examples of sequence 1 of the Rosario dataset

(a) Maximum features of 1955

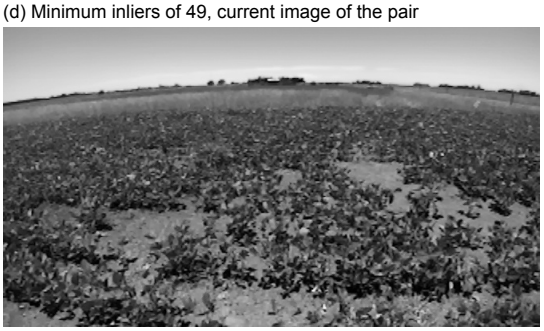


(b) Minimum features of 1909

(c) Maximum inliers of 1657, current image of the pair

(d) Minimum inliers of 49, current image of the pair

(e) Maximum inliers of 1657, previous image of the pair



(f) Minimum inliers of 49, previous image of the pair

(g) Maximum features of 1959 on the equalized images

(h) Minimum features of 1918 on the equalized images

(i) Maximum inliers of 1644 on the equalized images, current image of the pair

(j) Minimum inliers of 35 on the equalized images, current image of the pair

(k) Maximum inliers of 1644 on the equalized images, previous image of the pair

(l) Minimum inliers of 35 on the equalized images, previous image of the pair

Loop 1 of the business campus sequence of the 4Seasons dataset shows that the amount of features extracted (Figure 4.14b) and inliers Figure 4.14d) can drop dramatically due to underexposed areas caused by shadow in the image. Histogram equalization does increase the amount of extracted features (Figure 4.14h) and inliers (4.14j). However, the minimum inliers in Figure 4.14j can not be explained well as the image seems to have good lighting and enough objects for texture.

Figure 4.14: Qualitative examples of loop 1 of the business campus sequence of the 4Seasons dataset

(a) Maximum features of 1985

(b) Minimum features of 1375

(c) Maximum inliers of 1868, current image of the pair

(d) Minimum inliers of 109, current image of the pair

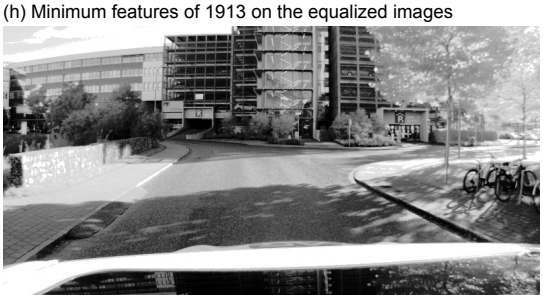
(e) Maximum inliers of 1868, previous image of the pair

(f) Minimum inliers of 109, previous image of the pair

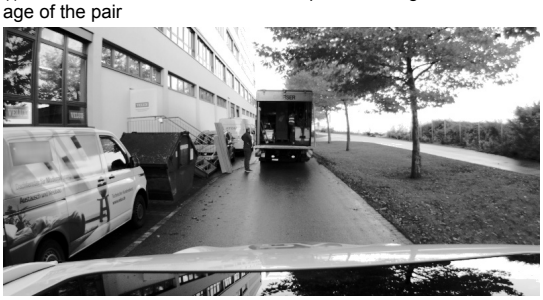
(g) Maximum features of 2000 on the equalized images

(h) Minimum features of 1913 on the equalized images

(i) Maximum inliers of 1857 on the equalized images, current image of the pair



(j) Minimum inliers of 126 on the equalized images, current image of the pair



(k) Maximum inliers of 1857 on the equalized images, previous image of the pair

(l) Minimum inliers of 126 on the equalized images, previous image of the pair

4.5. Replicating the problem

In this experiment, the feature extraction and matching pipeline is replicated with an OpenCV implementation of ORB in Python. The results are subsequently compared to the results of the ORB-SLAM3 pipeline. This experiment is executed in the SLAM mode of ORB-SLAM3, as the localization mode does not work correctly. Images with too few matches for tracking (under 15 matches) in ORB-SLAM3 are saved for the Juno dataset. The images are saved within the TrackReferenceKeyFrame function before and after outlier removal. From this, it could be determined that all failures in the TrackReferenceKeyframe function happen before outlier removal for the Juno dataset.

Results can be seen in Table 4.10. Note that we could not save a set of images for each sequence due to issues with reading the Rosbags during the run in ORB-SLAM3. The 'den Boer after rain' and 'van Adrichem sunny' sequences are missing in Table 4.10 because there was only one failure and too few samples as ORB-SLAM3 crashed close to the start of the sequence.

Table 4.10: The average matches and the standard deviation found in the frames of the sequences of the Juno dataset in ORB-SLAM3 and our OpenCV implementation. The average features found for each sequence in ORB-SLAM3 and the OpenCV implementation are recorded to see any relation with the total number of features found in a frame. The ORB settings in ORB-SLAM3 and the OpenCV implementation are: ($nfeatures = 2000$, $scaleFactor = 1.2$, $nlevels = 8$, $edgeThreshold = 19$, $firstLevel = 0$, $patchSize = 31$, $fastThreshold = 14$)

Sequence		**van Adrichem dark**	**van Adrichem after rain**	**den Boer dark**	**den Boer rain**	**den Boer sunny**
Amount of failures total samples		6/391	3/458	193/3809	4/2353	5/3940
Averaged matches	**ORB-SLAM3**	8.0 ±2.2	13.7 ±0.5	4.7 ±4.1	8.0 ±3.0	11.4 ±0.8
	OpenCV	**219.2 ±80.7**	**393.0 ±92.7**	**31.0 ±20.1**	**90.5 ±127.7**	**49.8 ±44.5**
Averaged features	**ORB-SLAM3**	599.7 ±52.1	1717.7 ±146.1	1242.7 ±98.1	**356.0 ±211.0**	842.2 ±644.9
	OpenCV	**1058.3 ±153.7**	**2000.0 ±0.0**	**1302.0 ±101.5**	352.3 ±149.3	**902.8 ±717.7**

Table 4.11: The average matches and the standard deviation found in the frames of the sequences of the Juno dataset in ORB-SLAM3 and our OpenCV implementation. The average features found for each sequence in ORB-SLAM3 and the OpenCV implementation are recorded to see any relation with the total number of features found in a frame. The ORB settings in ORB-SLAM3 and the OpenCV implementation are the best settings found in the previous experiment: ($nfeatures = 2000$, $scaleFactor = 1.2$, $nlevels = 8$, $edgeThreshold = patchSize$, $firstLevel = 0$, $patchSize = 48$, $fastThreshold = 5$)

Sequence		**van Adrichem dark**	**van Adrichem after rain**	**den Boer dark**	**den Boer rain**	**den Boer sunny**
Amount of failures total samples		6/896	2/590	23/4616	0/5248	0/5248
Averaged matches	**ORB-SLAM3**	10.7 ±2.9	12.0 ±0.0	11.6 ±2.4	x	14.0 ±0.0
	OpenCV	**328.5 ±204.9**	**667.5 ±155.5**	**32.0 ±33.0**	x	**44.5 ±22.5**
Averaged features	**ORB-SLAM3**	1641.7 ±446.7	**1991.0 ±0.0**	**1815.3 ±75.6**	x	**1314.0 ±0.0**
	OpenCV	**1655.3 ±420.5**	1980.0 ±2.0	1801.8 ±78.2	x	1227.5 ±99.5

For both the standard and best settings found in the previous experiment, the OpenCV implementation can find many matches on the images where ORB-SLAM3 failed to find a sufficient number for pose estimation. The average feature matches for the Juno dataset (run with the best settings, Table 4.11) in OpenCV is 268.125, and the average feature matches in ORB-SLAM3 is 12.075; The OpenCV pipeline outperforms ORB-SLAM3 with a factor of 22.2.

4.6. Correlations

To determine the reasons that affect the feature matches, we record the sharpness, optical flow, contrast, average intensity, percentage of black pixels, and percentage of white pixels for each frame for all datasets. The recorded variables are calculated as follows:

- Sharpness is calculated by taking the variance of the Laplacian, as described in [30].
- The optical flow is calculated with Gunnar Farneback's algorithm [13].

- The contrast is calculated with the root mean square contrast method, which describes the standard deviation of the pixel intensities [31].
- For the average intensity, the mean gray intensity value is calculated in the image.
- For the percentage of black pixels, all pixel values between 0 and 5 are counted and divided by the total amount of pixels in the image.
- For the percentage of white pixels, a similar calculation is done for the percentage of black pixels. The difference is that the counted pixel values are between 250 and 255.

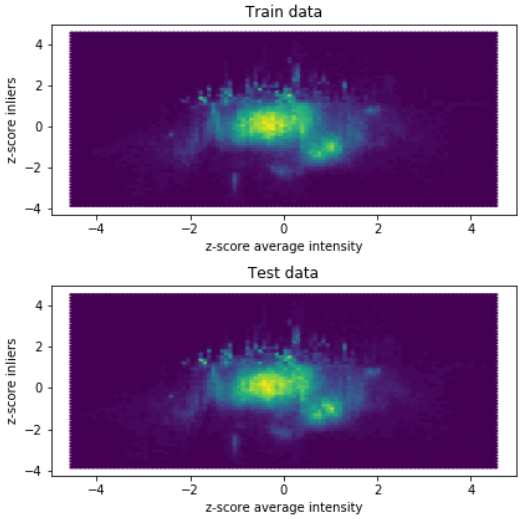
Based on the previous experiments, we run this experiment on the non-equalized images with the following ORB feature extractor settings: ($nfeatures = 2000$, $scaleFactor = 1.2$, $nlevels = 8$, $edgeThreshold = patchSize$, $firstLevel = 0$, $WTA_K = 2$, $scoreType = HARRIS\_SCORE$, $patchSize = 48$, $fastThreshold = 5$). The reason for running this experiment on non-equalized images is that even if histogram equalization makes the performance less sensitive to the proper tuning of the FAST threshold, it degrades the matching performance overall (see Chapter 4.3).

In this experiment, all variables are calculated for each frame and combined per dataset or sequence to calculate Spearman's correlation coefficient. In addition to Spearman's correlation, plots are made of the relationships between the different variables against the amount of inliers variable. As each variable has a different (unknown) distribution, a transformation is done to transform the data to a normal distribution. The relationship is visualized by plotting the z-scores of the variables against each other.

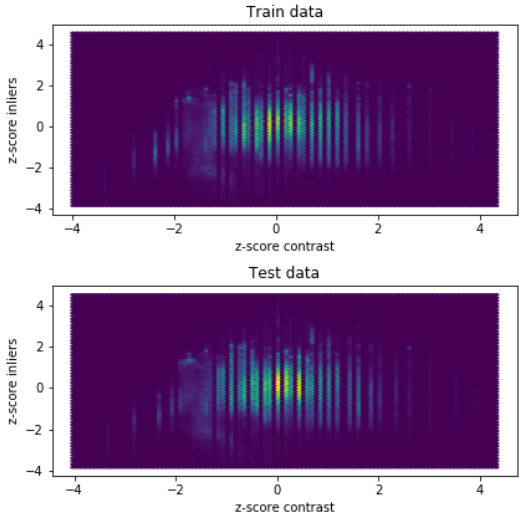
The Spearman's correlation and the z-scores are calculated for all datasets combined and for each sequence of Exos and Juno separately. To cross-validate, the data (which contains all recorded variables per frame) is shuffled with random state one and split in half to make a train and test set.

4.6.1. Results

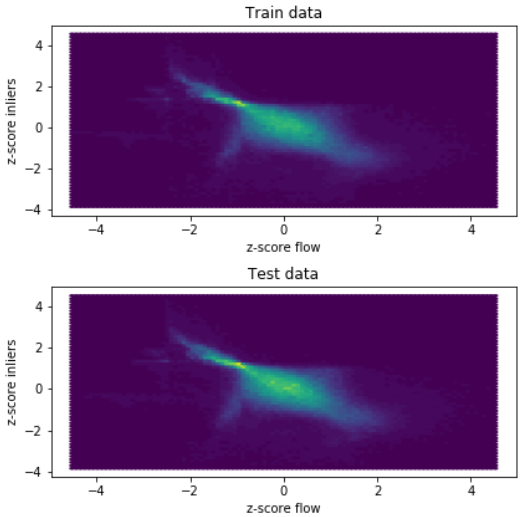
Figure 4.15: Transformed 2D-histograms of all datasets combined. Z-scores are plotted for different variables compared to the z-score of the number of inliers.



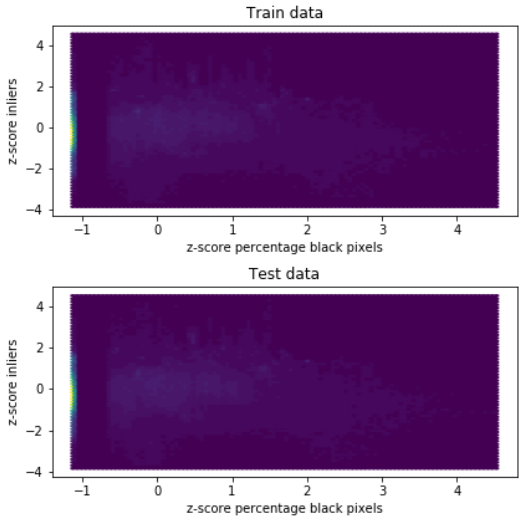
(a) Spearman correlations: train data = -0.14, test data = -0.14



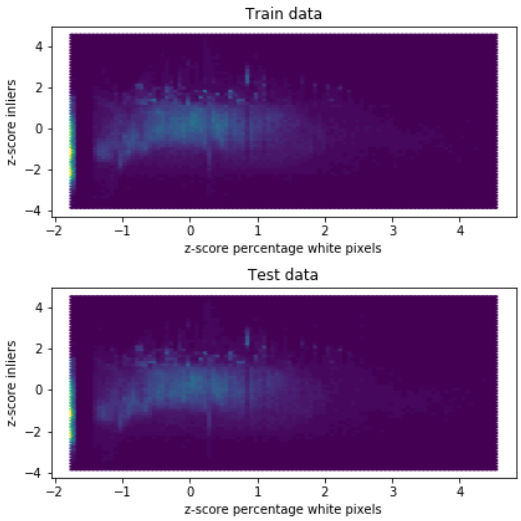
(b) Spearman correlations: train data = 0.07, test data = 0.07



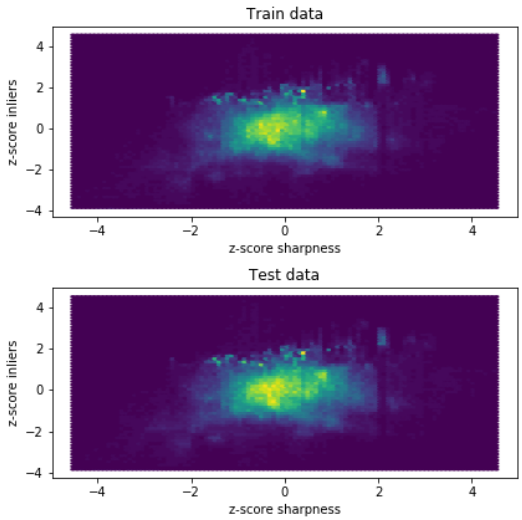
(c) Spearman correlations: train data = -0.6, test data = -0.6



(d) Spearman correlations: train data = 0.14, test data = 0.14

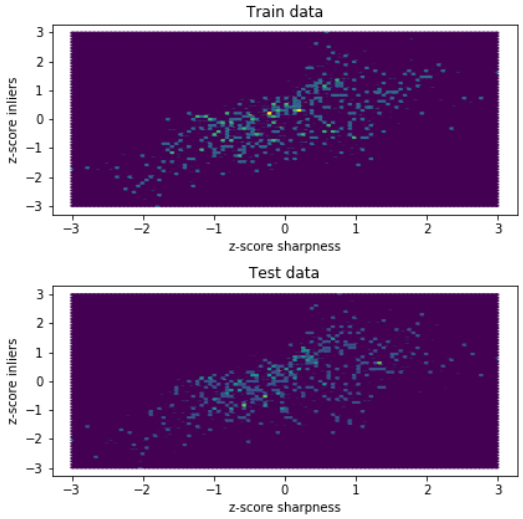


(e) Spearman correlations: train data = 0.18, test data = 0.19

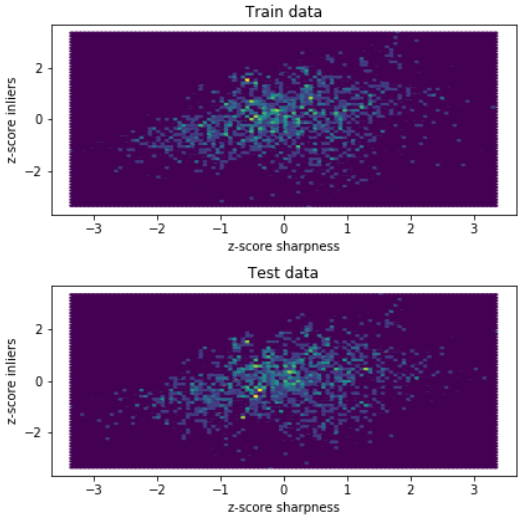


(f) Spearman correlations: train data = 0.15, test data = 0.15

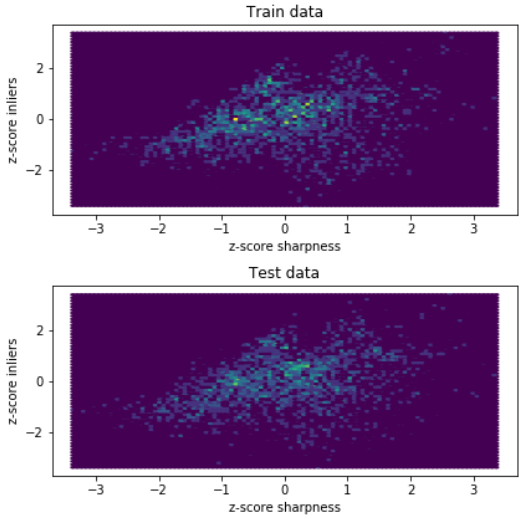
Figure 4.16: Transformed 2D-histograms of other correlations seen in the Juno and Exos datasets. Z-scores are plotted for different variables compared to the z-score of the number of inliers.



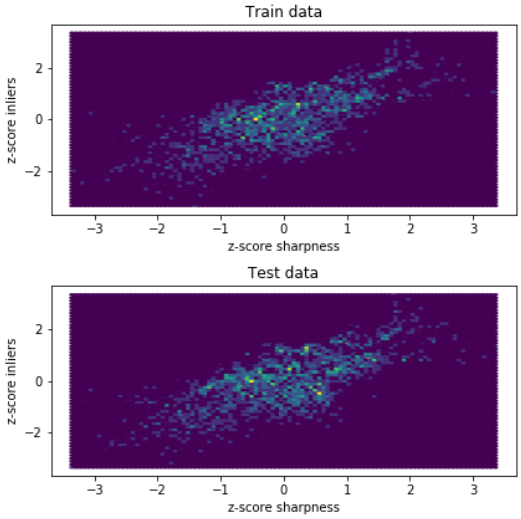
(a) Exos barn dark sequence. Spearman correlations: train data = 0.65, test data = 0.66



(b) Juno den Boer sunny sequence. Spearman correlations: train data = 0.28, test data = 0.31



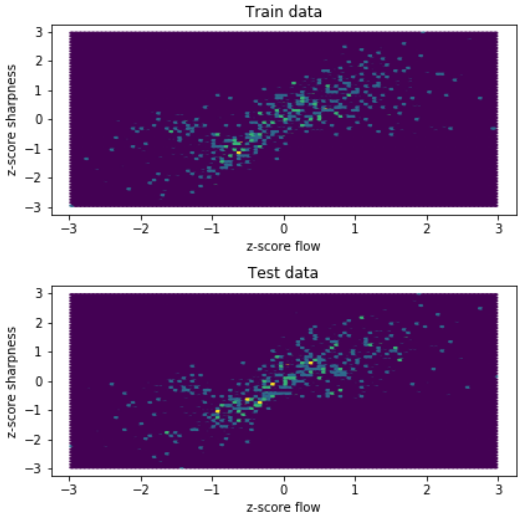
(c) Juno den Boer rain sequence. Spearman correlations: train data = 0.33, test data = 0.35



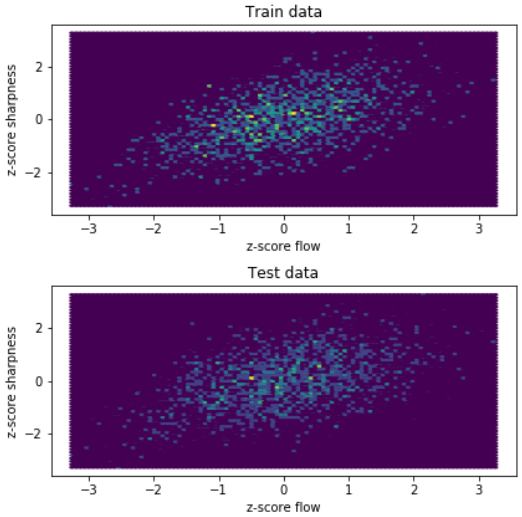
(d) Juno van Adrichem dark sequence. Spearman correlations: train data = 0.66, test data = 0.66

The results on all datasets combined can be seen in Figure 4.15. All plots are clusters with no apparent relationship except the optical flow plot in Figure 4.15c. As all datasets are combined, the data is averaged out. The optical flow shows a strong relationship, nevertheless. The Spearman's coefficient is -0.6, a relatively strong negative correlation with the number of inliers; this means fewer inliers are found if there is more apparent motion in a frame. For the separate sequences in the Exos and Juno datasets, plots that show other correlations besides optical flow are shown in Figure 4.16. Only sharpness shows a relatively strong correlation with the number of inliers in some sequences in the Exos and Juno datasets.; this means that the less blur there is in the frame, the more inliers can be found. To further investigate if motion blur could explain the inlier drops, the transformed 2D-histograms between sharpness and optical flow are plotted in the following Figure (Figure 4.17). Note that we can not directly deduce causality from correlations. However, hypothetically it is unlikely that blur causes apparent motion that is mainly caused by pure motion than the other way around.

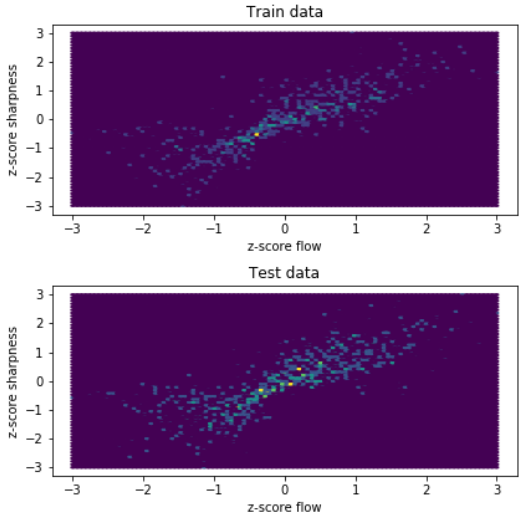
Figure 4.17: Transformed 2D-histograms of other correlations seen in the Juno and Exos datasets. Z-scores are plotted for different variables compared to the z-score of the number of inliers.



(a) Exos road rainy sequence. Spearman correlations: train data = 0.76, test data = 0.78



(b) Exos grass dark sequence. Spearman correlations: train data = 0.50, test data = 0.44



(c) Exos road normal sequence. Spearman correlations: train data = 0.87, test data = 0.86

From the Exos and Juno datasets' sequences, only three sequences from the Exos dataset show a strong correlation between the optical flow and sharpness. The correlation is positive, which means that more apparent motion causes sharper images in the frame or the other way around. From a practical point of view, it is unlikely that pure motion causes sharper images. A more likely explanation is that the optical flow measurement in these frames can not be purely seen as motion only; the change in illumination is significant, and the sharpness and optical flow calculations are based on similar properties in the image.

4.7. Tuning ORB-SLAM3 matcher

The TrackReferenceKeyframe function in ORB-SLAM3 uses the Bag-of-Words model to find feature correspondences between two given frames. The Bag-of-Words model finds similar initial features by their nodes, without directly comparing the descriptors, through the direct index as explained in Chapter 3.4. The ORB-matcher subsequently compares the descriptors of the features returned by the direct index afterward. The node level, which the Bag-of-World model uses to compare features associated with each node, can be tuned. The default value in ORB-SLAM3 is set to level 4.

The ORB-matcher also has two parameters that can be tuned; The nearest neighbor ratio and the descriptor distance threshold. The descriptor distance threshold ensures the closest match is under a certain distance in the descriptor space. As explained in [23], the nearest neighbor ratio filters most incorrect matches by checking the distance ratio between a descriptor in one image to the first and second closest match from the other image in the descriptor space. The two distances should be different enough to assume that the closest match is the possible correct match. A ratio close to one allows more ambiguous matches, while a lower value keeps only well-discriminative matches.

Different values for the three parameters are used to record the resulting matches and inliers found. Also, the number of total features found is recorded to not falsely relate a drop in feature matches to the change in parameters but to a general drop in the number of features extracted. The number of maps created is also recorded, which is eventually the consequence of the drop in feature matches that also needs to be minimized; a new map is started when both tracking and relocalization fail. A failure in the TrackReferenceKeyframe occurs when the number of matches is under 15 and if, after outlier filtering, the number of inliers is under 10. We record the number of occurrences the matches drop under 15 and the number of inliers falls under 10. Also, the mean of the number of matches and inliers is recorded for the whole sequence. This experiment is only run on the den Boer sunny sequence, as it is time-consuming to run on all sequences. The standard value of the nearest neighbor ratio set in ORB-SLAM3 is 0.7, and the descriptor distance threshold is set to 50. For each parameter, we use different values while keeping all other parameters fixed to their default value.

4.7.1. Results

In Table 4.12, the effect of changing the node level can be seen. With increasing node level, the number of times the number of matches falls under 15 also decreases. After outlier filtering, the number of times the inliers are under ten is also reduced, indicating that the increase in the number of matches did not increase the number of false matches.

Table 4.12: effect of changing the node level on the den Boer sunny sequence.

Node level		**2**	**4**	**6**
Number of maps created		25	11	**5**
Features	**mean**	1829.6	**1831.6**	1826.6
	min	**1231**	1111	1111
Matches	**mean**	51.6	86.5	**117.8**
	<15	136	54	**34**
Inliers	**mean**	51.6	86.5	**117.8**
	<10	38	12	**10**

The effect of changing the nearest neighbor ratio can be seen in Table 4.13. A value of 0.5 results in many matches under 15, as the ratio is too discriminative. A higher ratio seems to decrease the failure up until a certain point. After a ratio of 0.9, the effect seems less clear. The randomness in the trend after 0.9 can be explained by the randomness of the keyframe insertion and, thus, the selected features to be matched in ORB-SLAM3. This randomness becomes more dominant when the ratio change does not significantly affect the feature matching.

Table 4.13: effect of changing the nearest neighbor ratio on the den Boer sunny sequence.

Nearest neighbor ratio		**0.5**	**0.7**	**0.9**	**0.95**	**0.99**
Number of maps created		22	11	**10**	11	12
Number of features	**mean**	**1834.8**	1831.6	1825.7	1826.4	1828.1
	min	**1347**	1111	1111	1111	1111
Number of matches	**mean**	62.9	86.5	106.5	104.0	**109.8**
	<15	107	54	**30**	35	31
Number of inliers	**mean**	62.9	86.5	106.5	104.0	**109.8**
	<10	16	12	13	**11**	17

Changing the descriptor distance threshold to a smaller value increases the number of times the matches drop under 15. The larger the distance threshold, the less strict the difference in pixel intensities, and the more matches can be made. Increasing the distance threshold too much, as in the case of 100, does depreciate the performance again. A possible explanation is that more descriptors are kept, which makes matching more difficult, as too many descriptors in an area can be seen as noise.

Table 4.14: effect of changing the descriptor distance threshold on the den Boer sunny sequence.

Descriptor distance threshold		**20**	**50**	**80**	**100**
Number of maps created		**4**	11	6	8
Features	**mean**	**1850.8**	1831.6	1831.4	1831.4
	min	**1397**	1111	1169	1111
Matches	**mean**	42.3	86.5	**116.2**	114.5
	<15	123	54	**20**	27
Inliers	**mean**	42.3	86.5	**116.2**	114.5
	<10	12	12	**6**	9

In Table 4.15, the experiment is rerun with the best settings for the den Boer sunny sequence based on the settings that minimize the number of times the number of matches drops under 15. The number of times the matches fall under 15 is reduced to 15. However, seven maps were started, which means that tracking was lost and unable to relocalize seven times.

Table 4.15: Results on the matching with node level=6, descriptor distance threshold=80, nearest neighbor ratio=0.9 on the den Boer sunny dataset.

Number of maps created		7
Features	**mean**	1818.2
	min	1111
Matches	**mean**	193.6
	<15	15
Inliers	**mean**	193.6
	<10	7

In Table 4.16, the experiment is run with the same best setting as in Table 4.16, but this time with the descriptor distance threshold set to 20 as this threshold resulted in the least amount of maps in Table 4.14. The number of maps started is slightly lower. However, there are many more matches that did drop under 15.

Table 4.16: Results on the matching with node level=6, descriptor distance threshold=20, nearest neighbor ratio=0.9 on the den Boer sunny dataset.

Number of maps created		6
Features	**mean**	1842.4
	min	1347
Matches	**mean**	51.0
	<15	110
Inliers	**mean**	51.0
	<10	13

4.8. A simple stereo visual odometry implementation

This chapter explores the possibility of using a simple visual odometry (VO) system instead of ORB-SLAM3; to explore the robustness of a simple VO system and compare it to ORB-SLAM3. There will be a trade-off between robustness and accuracy, as the simple visual odometry system is less optimized than ORB-SLAM3. Also, the error builds up over time as we explore an odometry system. For Lely Juno, this error is expected to be reasonable, as the trajectory is relatively short.

An online repository is used for the stereo visual odometry implementation [29]. The repository is adjusted to match features with a brute force matcher. The original implementation works as follows; The image is divided into patches to extract more evenly distributed features. Features are extracted from each patch in frame t-1. An estimation of the location of the features in frame t is made with an optical flow tracker. A disparity map is created from the left and right images at time t, and the inbound feature points with the minimum and maximum disparity are used to triangulate the features to get 3D points. A pose is calculated by selecting six feature points and minimizing the re-projection error with the least squares for x amount of iterations. The pose estimation with the least amount of error from these iterations is the final estimated pose.

In the adjustments, the estimate of the feature points in frame t with the optical flow is replaced by a brute-force feature match. Moreover, features are extracted from the full image size instead of patches. The reason for not using patches is that the parameters of the ORB extractor need to be set relatively small to detect features in a small patch; this resulted in many wrong matches, even after RANSAC filtering. A plausible explanation for the incorrect matches is that the feature response of the features is too weak. The feature extractor becomes sensitive to any response, even noise. Thus, the feature descriptors become too indistinct to correctly match the correct feature point in the following image. Therefore, the features are extracted from the entire image instead with the previously determined parameters ($nfeatures = 2000$, $scaleFactor = 1.2$, $nlevels = 8$, $edgeThreshold = patchSize$, $firstLevel = 0$, $WTA_K = 2$, $scoreType = HARRIS\_SCORE$, $patchSize = 48$, $fastThreshold = 5$) for the feature extractor.

The settings for the disparity map are set to: $minDisparity = 1$, $numDisparities = 48$, $blockSize = 3$, $P1 = int(block * block * 8 * smooth)$, $P2 = int(block * block * 32 * smooth)$, $speckleRange = 0$, $speckleWindowSize = 0$ with the smoothness factor for P1 and P2 set to $smooth = 3.0$.

4.8.1. Results

KITTI The estimated paths of the three different methods (ORB-SLAM3, VO with brute force matcher, and VO with optical flow tracker) for the KITTI sequences can be seen in Figure 4.18. The localization mode of ORB-SLAM3 did not work properly; therefore, the sequences are evaluated with loop closure. Not all sequences have loops. The sequences without loops (Figures 4.18b, 4.18d, 4.18e, and 4.18i) are fairer to compare the VO methods to the ORB-SLAM3 performance.

In sequence 00 (Figure 4.18a), ORB-SLAM3 is the most accurate method based on their paths and absolute position error (APE, as defined in [32]). However, ORB-SLAM3 shows a sudden straight line connecting two positions. The reason can be because of different maps that are not connected correctly or failed place recognition. In sequence 01 (Figure 4.18b), all methods fail to estimate the path accurately. The most accurate method seems to be the VO with feature matching based on the APE.

The results of sequence 02 (Figure 4.18c) show that ORB-SLAM3 lost track and started a new map at the end of the trajectory. However, it could not merge this map with any other existing maps. The

APE is low for ORB-SLAM3, but this is only based on a small part of the trajectory. The VO with optical flow tracking seems to be the most robust and accurate method for this sequence. The same holds for sequence 10 (Figure 4.18k).

All methods are quite accurate in sequence 02, especially for the first part. The error builds up toward the end of the sequence. ORB-SLAM3 is the most accurate of all methods, followed by VO with optical flow. The same holds for sequence 04 (Figure 4.18e).

In sequence 05 (Figure 4.18f) ORB-SLAM3 outperforms both VO methods by a large margin, which is also the case in sequence 06 (Figure 4.18g).

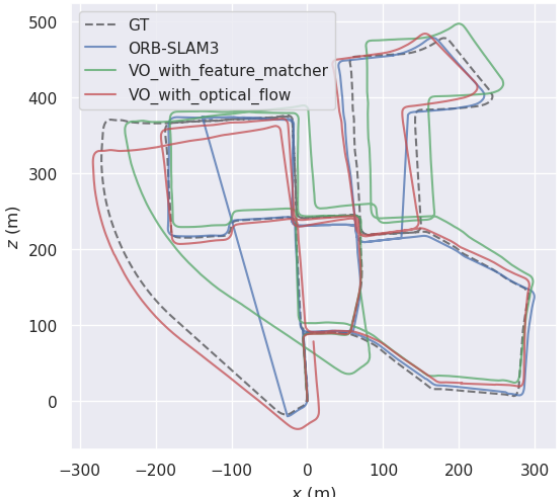
Regarding APE, ORB-SLAM3 outperforms both VO methods again in sequence 07 (Figure 4.18h). However, ORB-SLAM3 misses a large part of the trajectory again, which is also the case in sequence 08 (Figure 4.18i) and sequence 09 (Figure 4.18j). When looking at the entire trajectory, the VO method with the feature matcher is the most accurate in sequence 08 based on the APE, while the VO method with the optical flow is more accurate in sequence 09.

Both VO methods did not have any problems finding enough feature matches for pose estimation, unlike ORB-SLAM3. The minimum amount of matches (and inliers for the VO method) can be seen in Table 4.17.

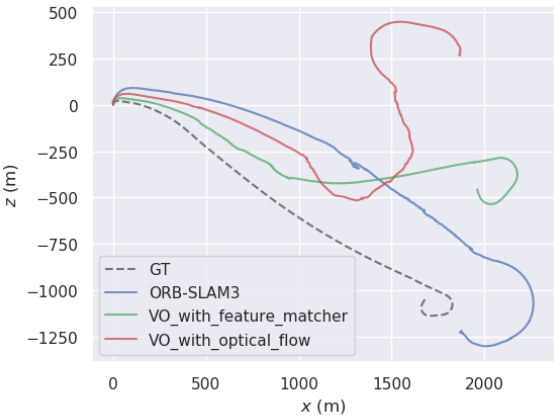
Table 4.17: Minimum amount of matches seen in the sequences of the KITTI dataset for ORB-SLAM3 and the matching-based VO. The average minimum matches of the matching based VO is 49 times more than the average minimum matches in ORB-SLAM3, even when compared with the minimum number of inliers in the VO method, the factor is still 48.9.

KITTI sequence		**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
ORB-SLAM3	min matches	0	12	0	1	39	0	2	0	0	2	0
VO	min matches	**218**	**274**	**157**	**350**	**493**	**305**	**246**	**312**	**240**	**272**	**159**
	min inliers	185	247	140	324	484	280	220	258	191	262	150

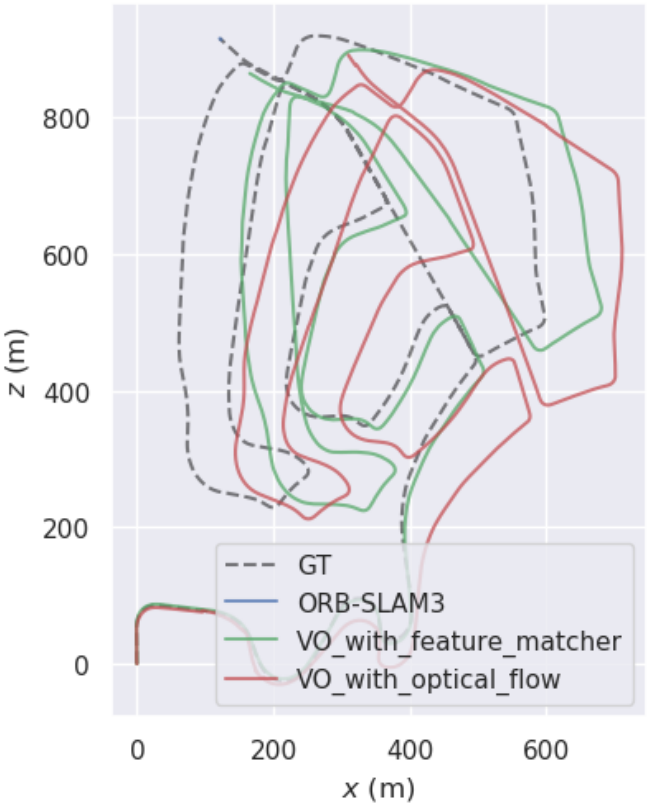
Figure 4.18: Estimated trajectories of the KITTI sequences. The trajectories are estimated with ORB-SLAM3, the VO method with the brute force matcher, and the VO method with the optical flow tracker.



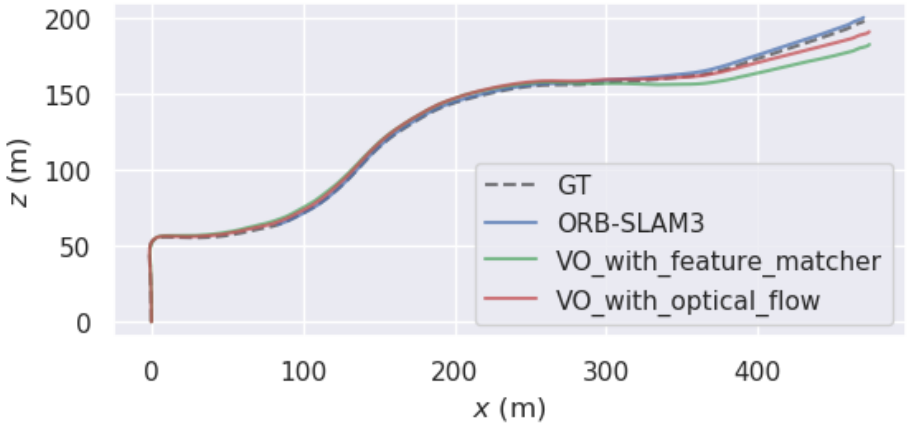
(a) KITTI sequence 00. ORB-SLAM3: APE_{rmse} = 8.5m. VO with feature matcher: APE_{rmse} = 30.3m. VO with optical flow: APE_{rmse} = 15.6m.



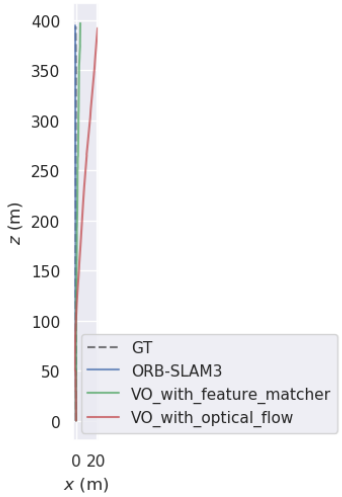
(b) KITTI sequence 01. ORB-SLAM3: APE_{rmse} = 574.3m. VO with feature matcher: APE_{rmse} = 279.1m. VO with optical flow: APE_{rmse} = 591.6m.



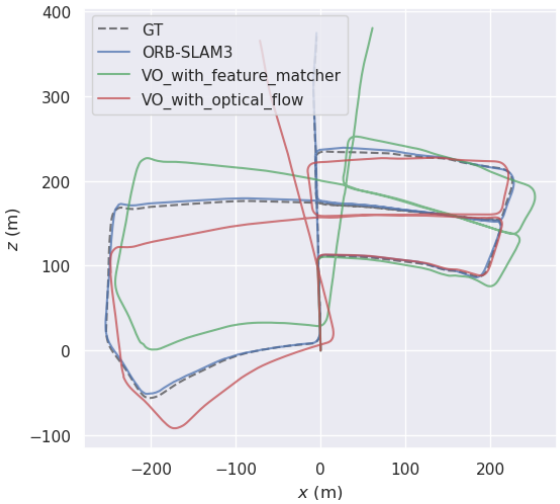
(c) KITTI sequence 02. ORB-SLAM3: APE_{rmse} = 0.02m. VO with feature matcher: APE_{rmse} = 49.0m. VO with optical flow: APE_{rmse} = 16.1m.



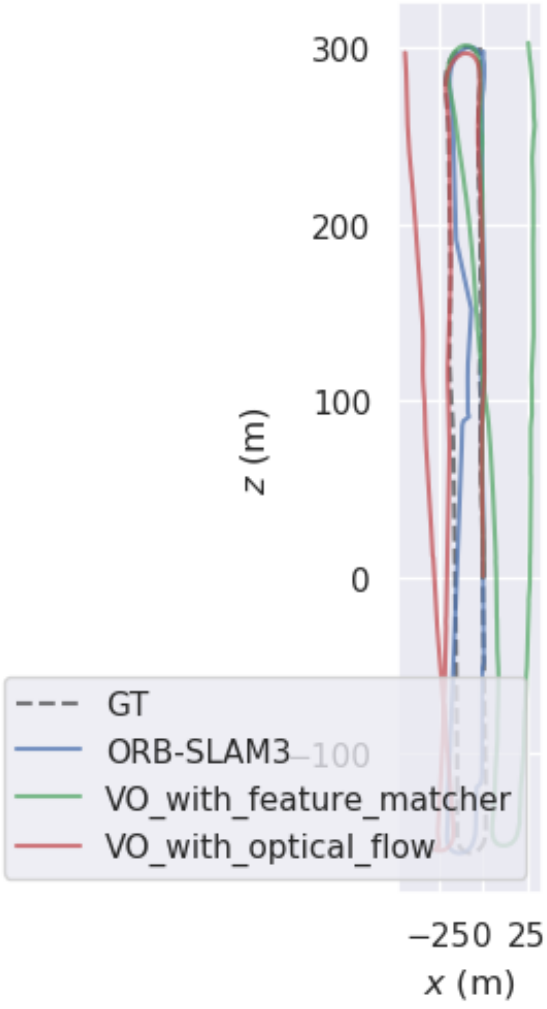
(d) KITTI sequence 03. ORB-SLAM3: APE_{rmse} = 3.3m. VO with feature matcher: APE_{rmse} = 9.6m. VO with optical flow: APE_{rmse} = 8.4m.



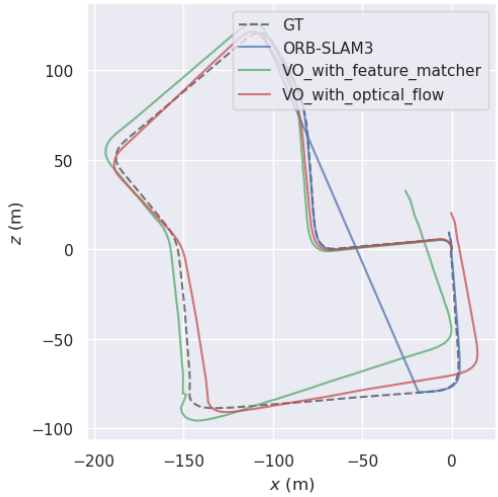
(e) KITTI sequence 04. ORB-SLAM3: APE_{rmse} = 1.5m. VO with feature matcher: APE_{rmse} = 3.8m. VO with optical flow: APE_{rmse} = 1.9m.



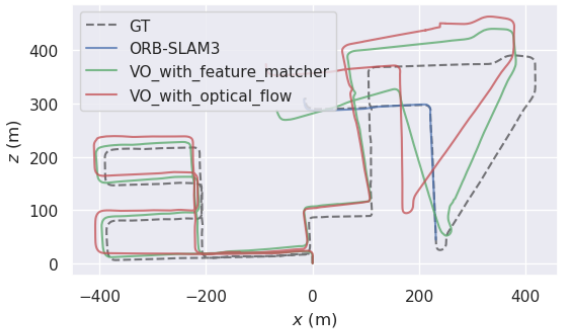
(f) KITTI sequence 05. ORB-SLAM3: APE_{rmse} = 2.0m. VO with feature matcher: APE_{rmse} = 12.0m. VO with optical flow: APE_{rmse} = 11.3m.



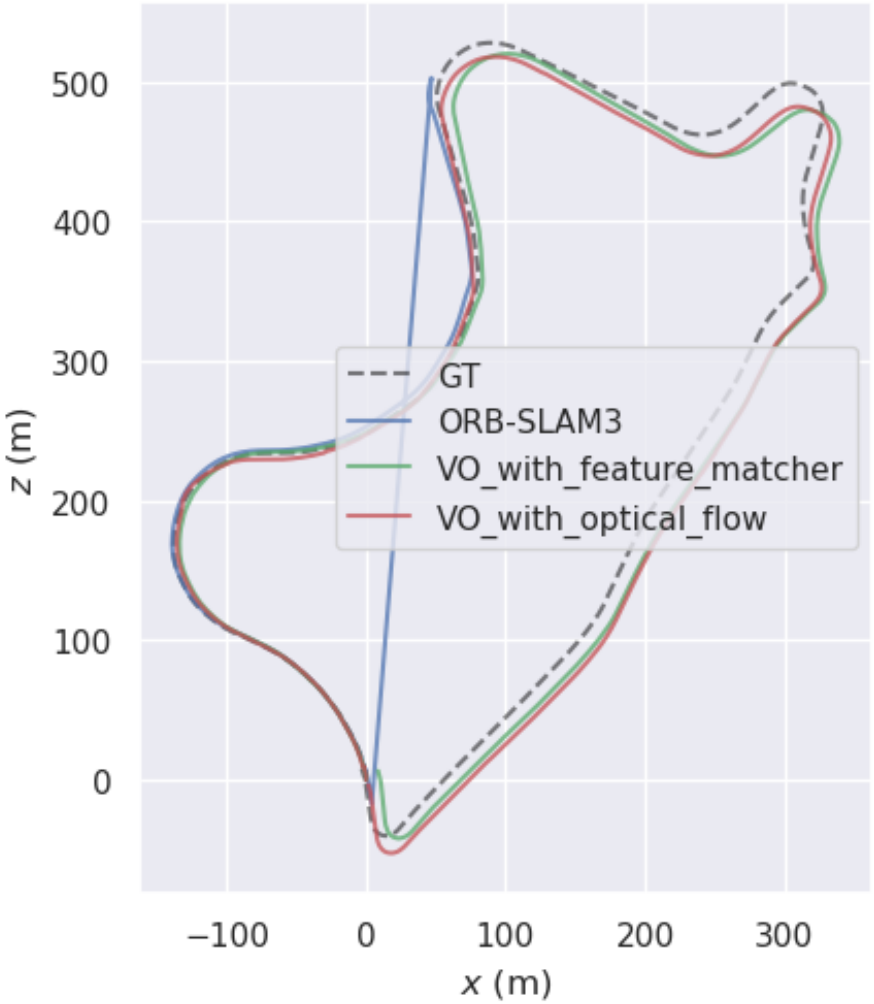
(g) KITTI sequence 06. ORB-SLAM3: APE_{rmse} = 3.6m. VO with feature matcher: APE_{rmse} = 13.8m. VO with optical flow: APE_{rmse} = 15.4m.



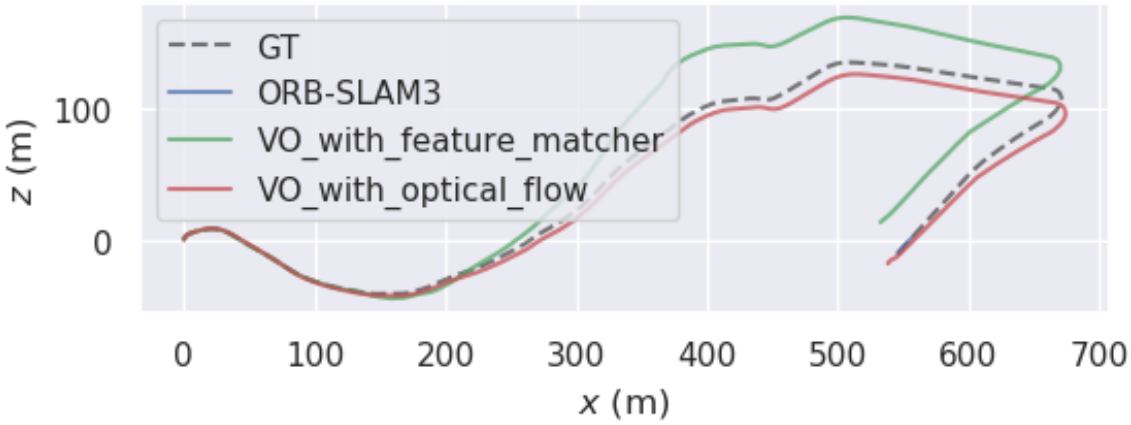
(h) KITTI sequence 07. ORB-SLAM3: APE_{rmse} = 0.51m. VO with feature matcher: APE_{rmse} = 11.3m. VO with optical flow: APE_{rmse} = 3.3m.



(i) KITTI sequence 08. ORB-SLAM3: APE_{rmse} = 1.0m. VO with feature matcher: APE_{rmse} = 20.9m. VO with optical flow: APE_{rmse} = 28.9m.



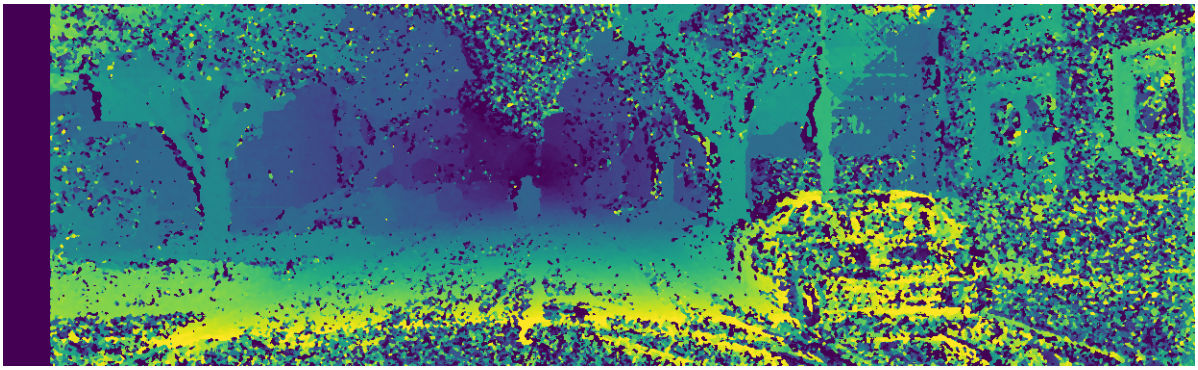
(j) KITTI sequence 09. ORB-SLAM3: APE_{rmse} = 1.4m. VO with feature matcher: APE_{rmse} = 11.7m. VO with the optical flow: APE_{rmse} = 7.4m.



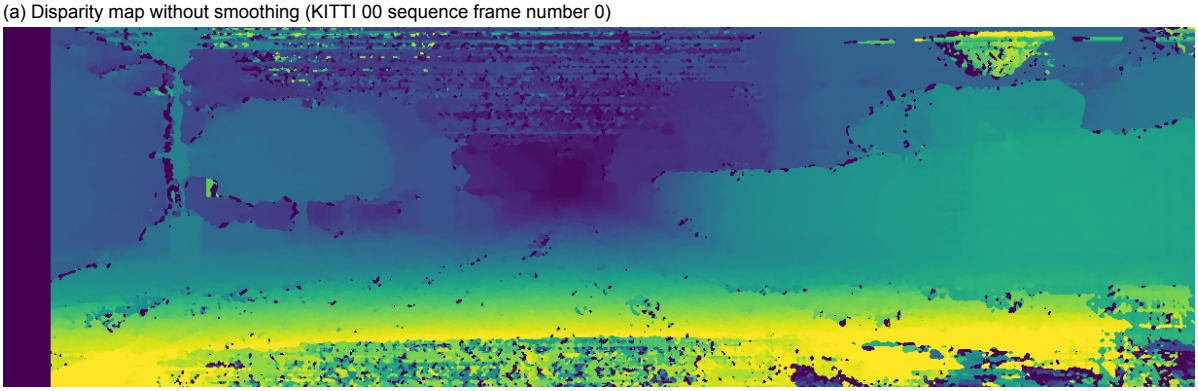
(k) KITTI sequence 10. ORB-SLAM3: APE_{rmse} = 1.5m. VO with feature matcher: APE_{rmse} = 13.2m. VO with optical flow: APE_{rmse} = 4.1m.

The sequences where ORB-SLAM3 did not fail are sequences 01, 03, 04, 05, and 06. From these sequences, sequences 05 and 06 contain loop closures. The APE_{rmse} of the best VO method in sequence 01 is 0.48 times better than the APE_{rmse} of ORB-SLAM3. The APE_{rmse} of the best VO method in sequence 03 is 2.5 times worse compared to ORB-SLAM3 and 1.26 times worse in sequence 04. The sequences with loop closure show that APE_{rmse} of the best VO method is even worse if ORB-SLAM3 can make loop-closures; the factor is 5.65 for sequence 05 and 3.83 for sequence 06.

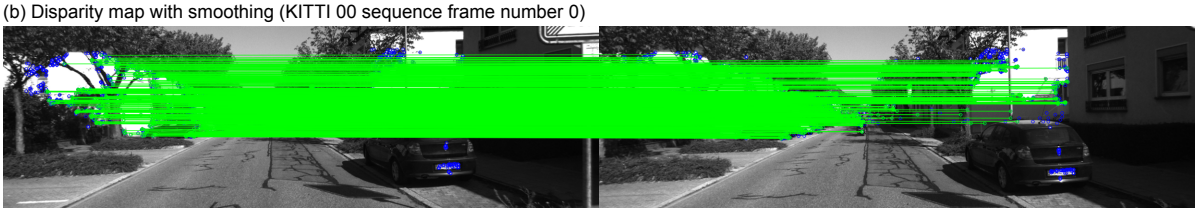
The accuracy of the VO methods is affected a lot by correct matches and a good disparity map, as there are no continuous optimization steps like in ORB-SLAM3. The paths in sequence 00 and sequence 02 show that the VO methods estimate the shape of the trajectory well but not the depth. The more noise there is in the disparity map, the less accurate the depth estimation, which can be seen in Figure 4.20. If the features used for pose estimate correspond to noisy locations in the disparity map, the pose estimation becomes worse. An example of a noisy and smoothed disparity map is given in Figures 4.19a and 4.19b as well as (the locations of) the feature matches in the image in Figure 4.19c.



(a) Disparity map without smoothing (KITTI 00 sequence frame number 0)



(b) Disparity map with smoothing (KITTI 00 sequence frame number 0)



(c) ORB feature matches extracted between KITTI 00 sequence frame number 0 and 1

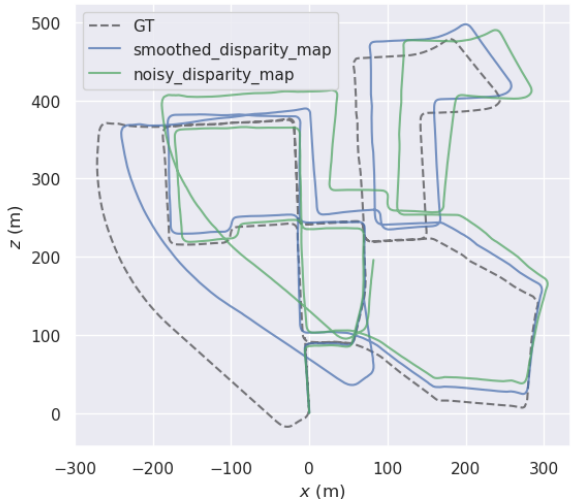


Figure 4.20: Path of KITTI sequence 00. VO with feature matching run with a smooth and noisy disparity map. Smooth disparity map (smooth factor set to 3.0): APE_{rmse} = 30.3m. Noisy disparity map (smooth factor set to 0.0): APE_{rmse} = 47.4m. This is a factor of 1.56 difference in the APE_{rmse}.

Juno The estimated paths of the two different VO methods (VO with brute force matcher and VO with optical flow tracker) can be seen in Figure 4.21 for the Juno sequences. There are no results

of the ORB-SLAM3 path due to an unsolvable bug that crashed the ORB-SLAM3 program before a trajectory could be saved. Both VO methods failed for all dark sequences due to too few inbound features that could be extracted with the minimum and maximum disparity. Both VO methods could follow the trajectory well for the Van Adrichem sequences, and both methods have a similar APE_{rmse} score. The results for the den Boer sequence are less promising. Both VO methods fail to follow the ground truth after the turn (around x=42 and y=94). The shape of the trajectory is still somewhat similar to the ground truth for the VO method with the feature matcher. However, the VO with optical flow fails almost completely for the den Boer sunny and den Boer after rain sequence.

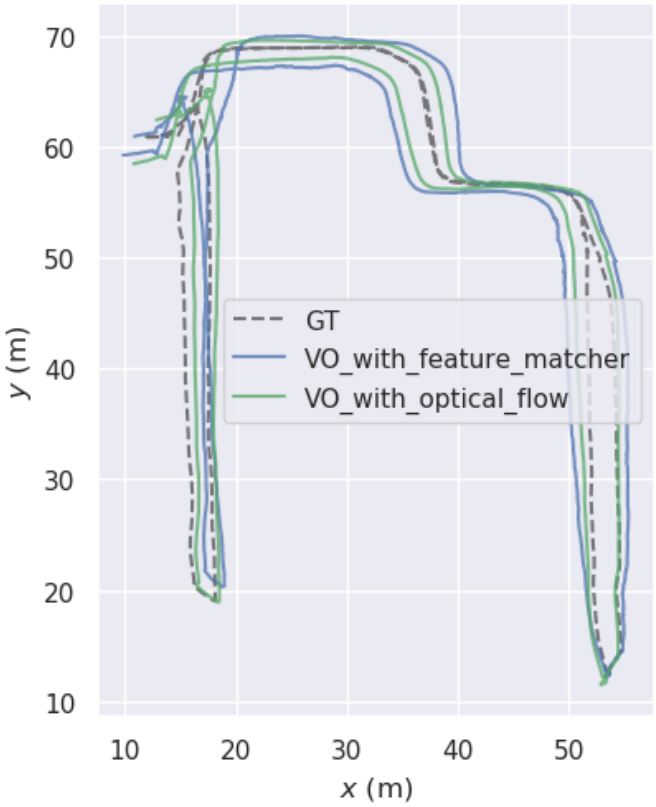
The disparity maps and feature matches around this turn at x=42 and y=94 can be seen for each den Boer sequence in Figures 4.25, 4.23, and 4.24.

The disparity map of the den Boer sunny sequence is relatively smooth compared to the disparity maps of the den Boer rain (Figure 4.24b) and after rain (Figure 4.25b) sequences. Also, more features are extracted from the floor because of the feed spread on the floor.

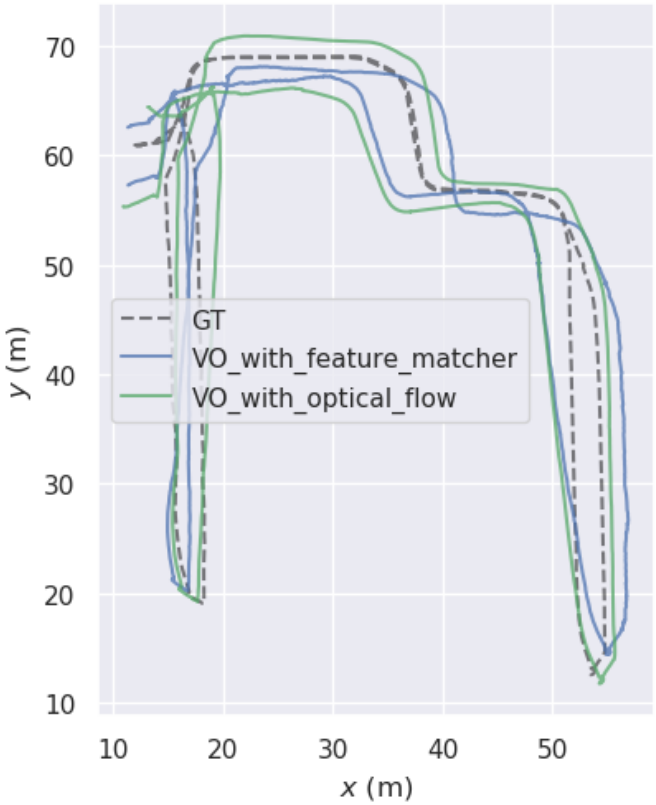
The feature matches in the den Boer rain sequence (Figure 4.24) show that most features are extracted from the fence and the feed that is close to the fence. The disparity map contains very noisy patches. The reason for the noise can not be explained by looking at the images.

The feature matches in the den Boer after rain sequence (Figure 4.25) show that most features are extracted from the fence and less from the floor as the floor does not have much texture. The disparity map is quite noisy; a noisy patch is present on the left side of the disparity map and the upper right side. The reason for this noise can not be explained by looking at the images.

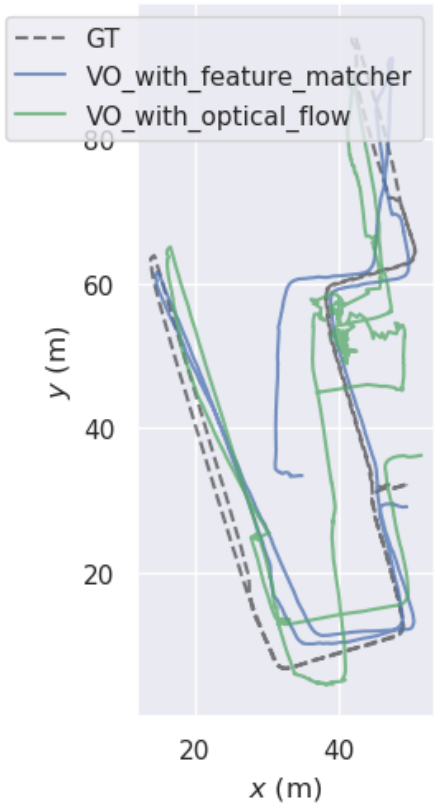
Figure 4.21: Estimated trajectories of the Juno sequences. The trajectories are estimated with the VO method with the brute force matcher and the VO method with the optical flow.



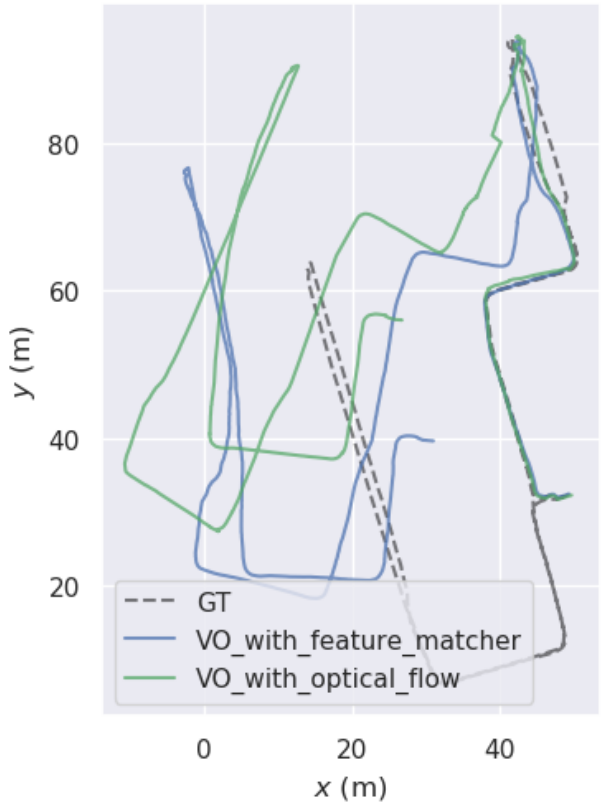
(a) Van Adrichem sunny sequence. VO with feature matcher: APE_{rmse} = 55.7m. VO with optical flow: APE_{rmse} = 55.6m.



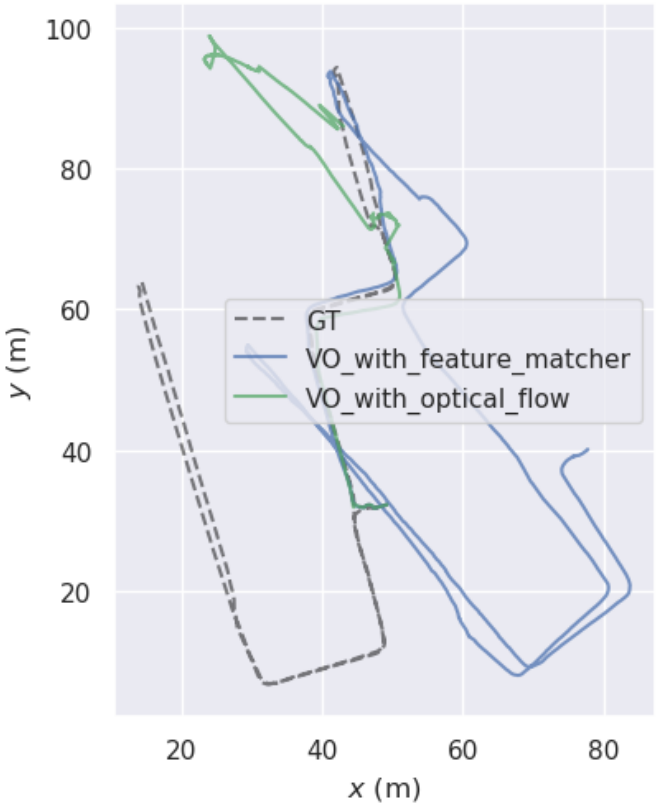
(b) Van Adrichem after rain. VO with feature matcher: APE_{rmse} = 55.6m. VO with optical flow: APE_{rmse} = 60.6m.



(c) den Boer Sunny sequence. VO with feature matcher: APE_{rmse} = 40.9m. VO with optical flow: APE_{mean} = 38.6m, APE_{rmse} = 40.6m.

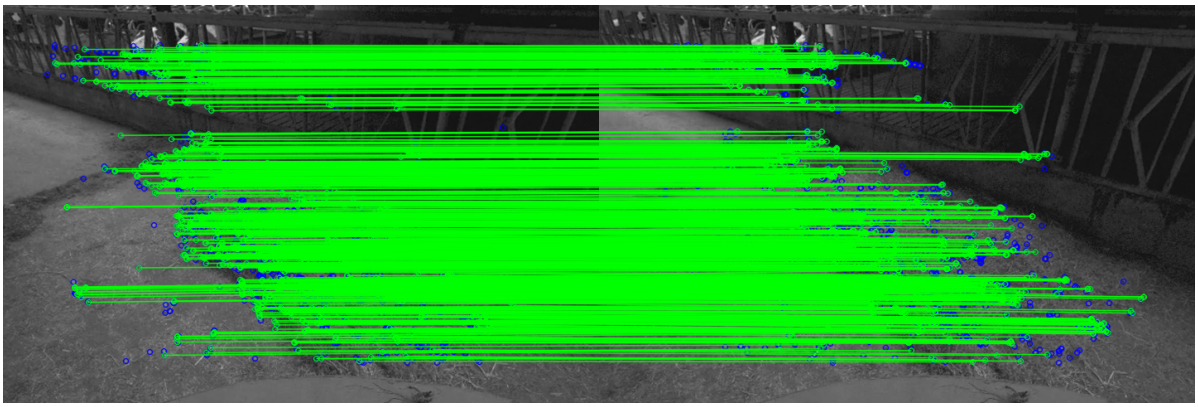


(d) den Boer rain sequence. VO with feature matcher: APE_{rmse} = 52.4m. VO with optical flow: APE_{rmse} = 53.0m.

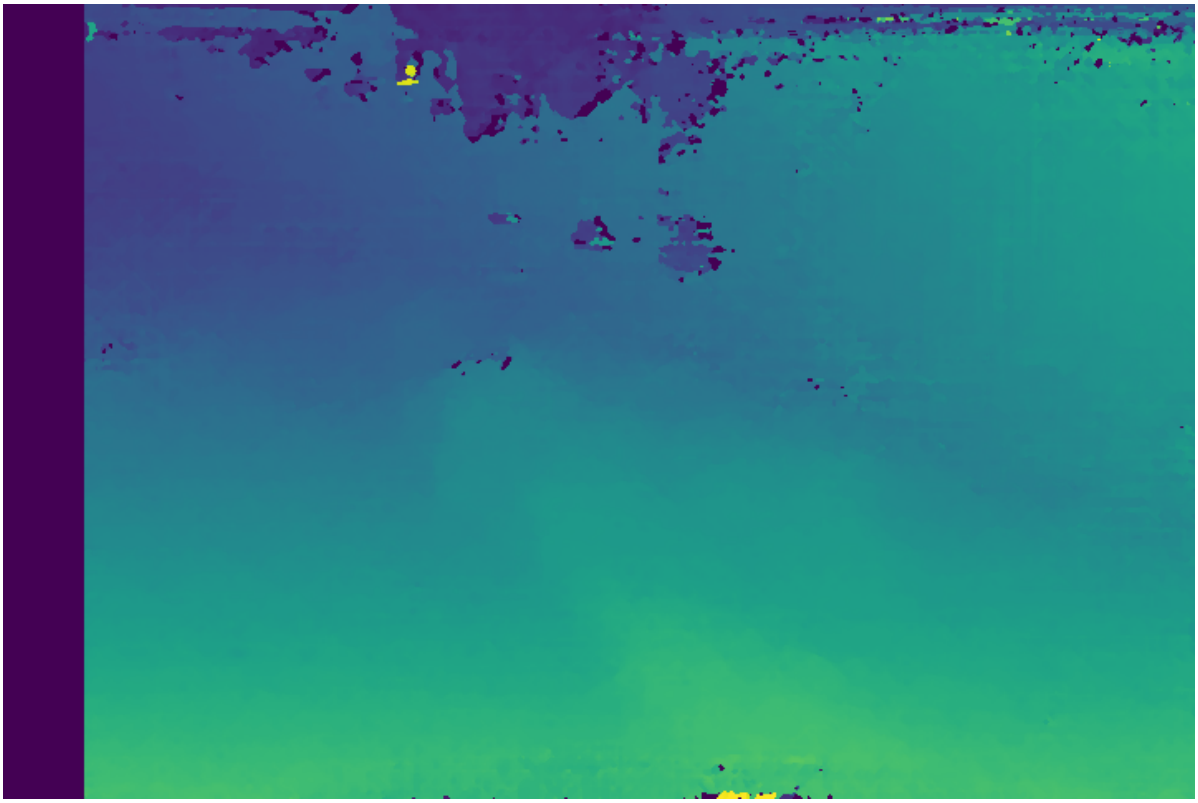


(a) den Boer after rain sequence. VO with feature matcher: APE_{rmse} = 45.3m. VO with optical flow: APE_{rmse} = 59.3m.

Figure 4.23: The matches and disparity map during the turn around x=42 and y=94 for the den Boer sunny sequence.

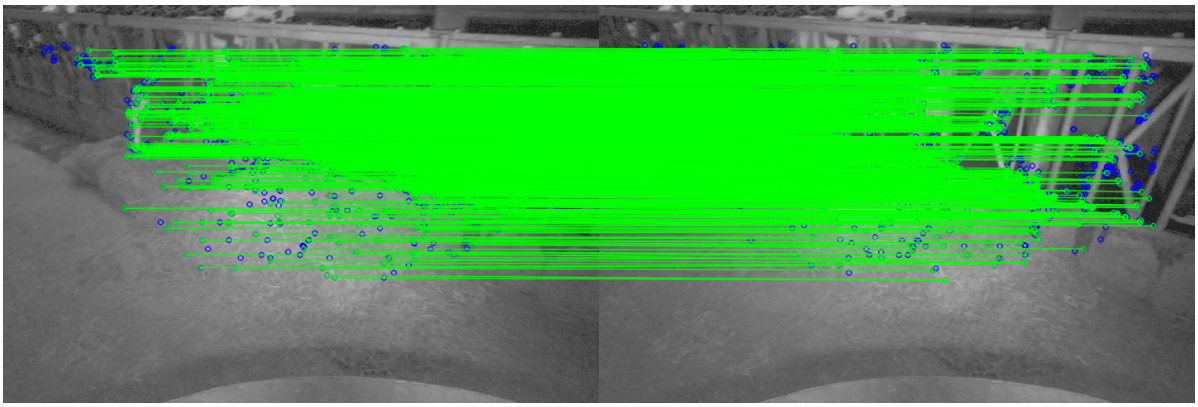


(a) Matches made between t and t-1 for the den Boer sunny sequence.

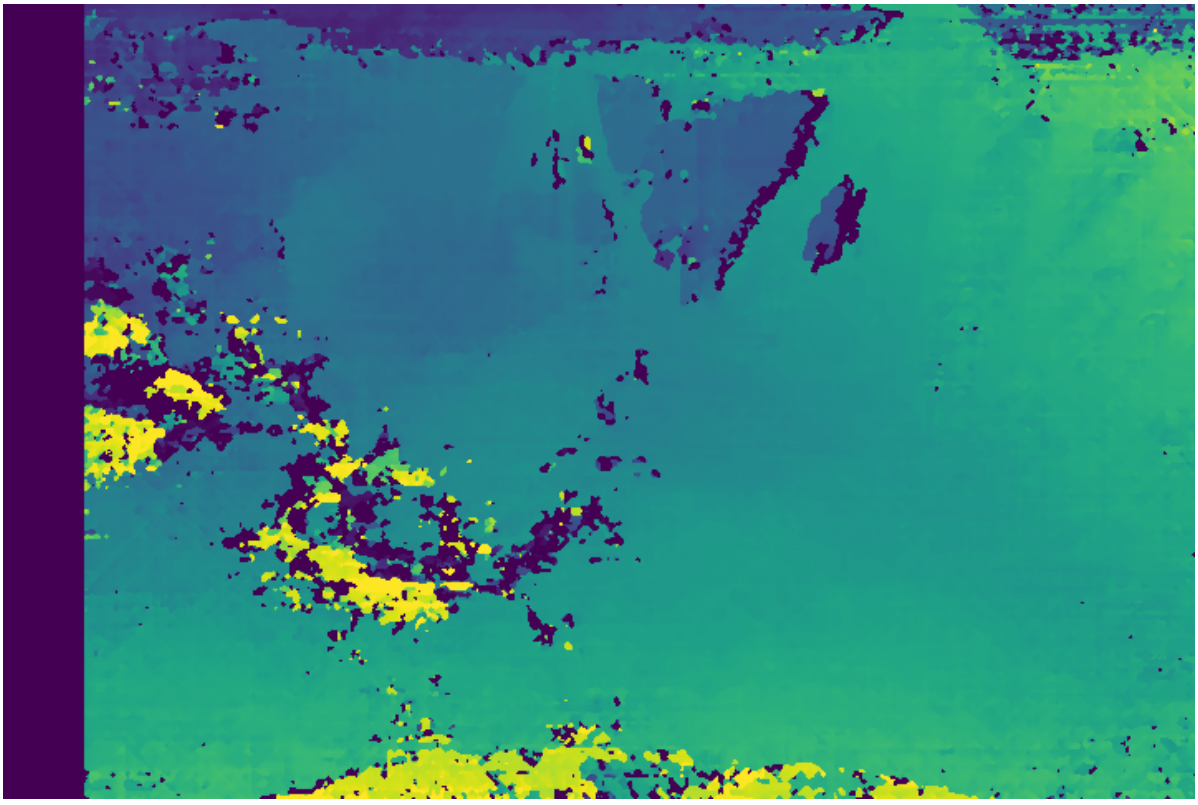


(b) Disparity map at time t for the den Boer sunny sequence.

Figure 4.24: The matches and disparity map during the turn around x=42 and y=94 for the den Boer rain sequence.

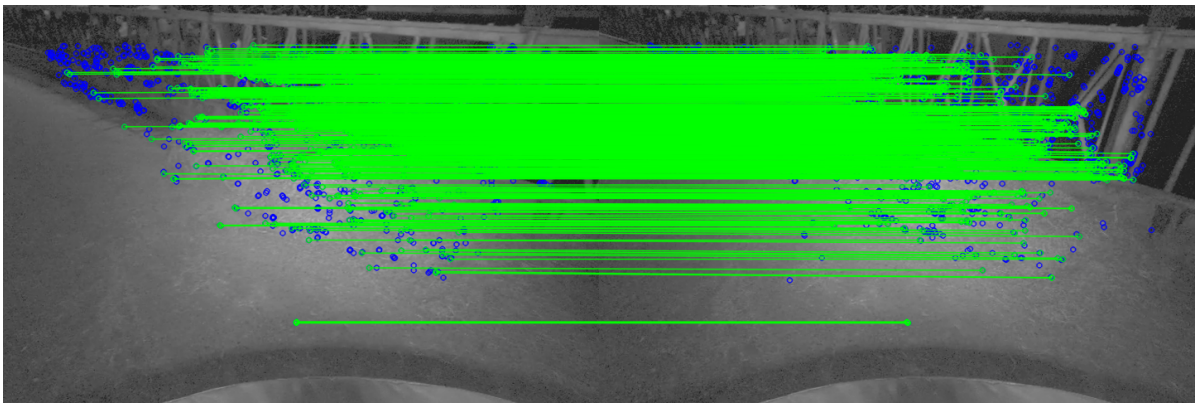


(a) Matches made between t and t-1 for the den Boer rain sequence.

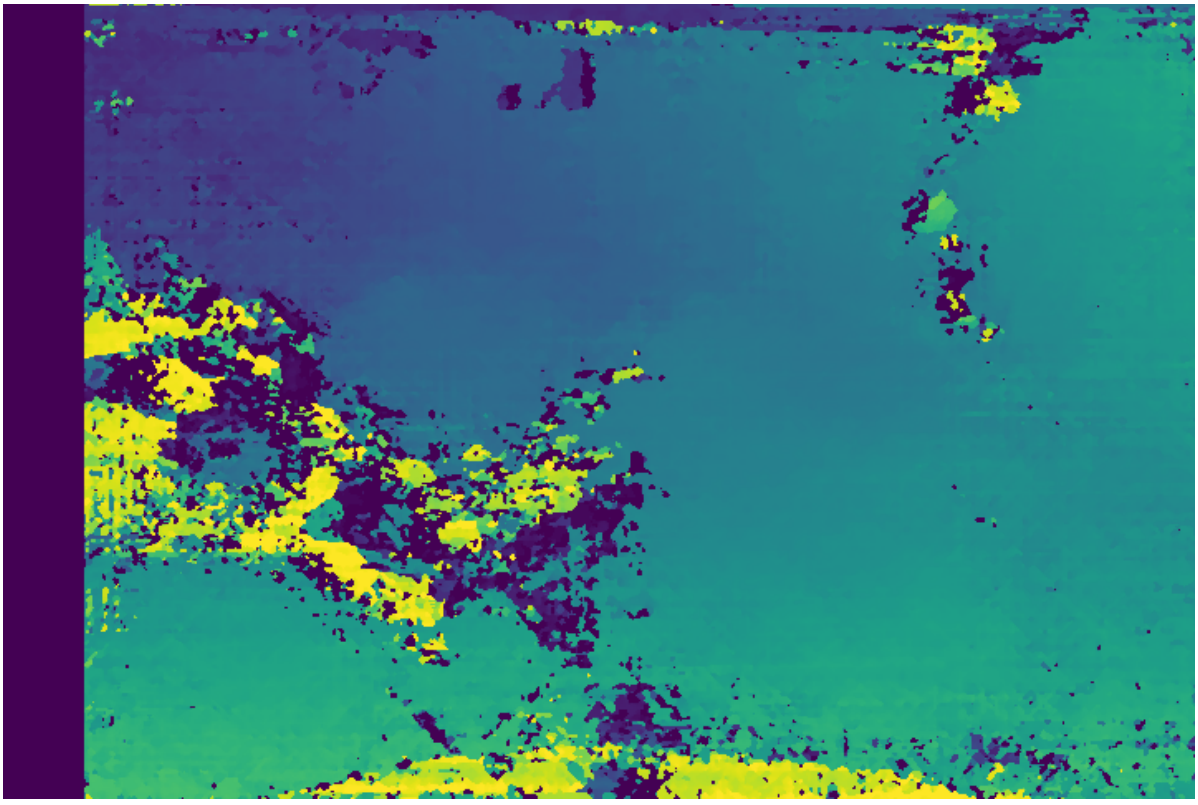


(b) Disparity map at time t for the den Boer rain sequence.

Figure 4.25: The matches and disparity map during the turn around x=42 and y=94 for the den Boer after rain sequence.



(a) Matches made between t and t-1 for the den Boer after rain sequence.



(b) Disparity map at time t for the den Boer after rain sequence.

5

Discussion & Conclusion

5.1. Discussion

While determining suitable ORB parameters for the experiments, it became clear that a specific change in the settings had similar effects on the outcome of the number of feature matches in most datasets. From this, a single setting for the ORB features is found that works well for all datasets; this means that if suitable parameters are used, it should only affect the number of feature matches a little when a different dataset is used. From qualitative results in Chapter 4.4, the effect of the image on the matches became slightly more apparent; the number of feature matches reduced due to apparent reasons such as blur, overexposure, and underexposure in the image. However, there were also many examples without an apparent reason for such a sudden drop in feature matches. Equalizing the images did not positively affect the number of matches, probably due to the noise it introduced. The noise could be seen clearly in some qualitative examples. Moreover, the drop in the number of feature matches happens in agricultural datasets and the widely used indoor and outdoor robotics datasets.

Although the images affect the feature matches in a way, a more significant reason for the drop in the number of feature matches in ORB-SLAM3 is the type of matcher used. Tuning the matcher and the ORB extraction parameters did improve the drop in the number of matches (Chapter 4.7). The mean number of inliers increased from 86.5 to 193.6 compared to the default matcher settings, and the number of times the inliers dropped under ten was reduced from 12 to 10. However, it did not eliminate the problem. The brute-force-based matcher consistently found enough matches and inliers needed for pose estimation compared to ORB-SLAM3's bag-of-words-based matcher (Chapter 4.5); the OpenCV pipeline outperforms ORB-SLAM3 on average by a factor of 22.2 on the Juno dataset.

Moreover, ORB-SLAM3 divides the images into patches before features are extracted. These patches significantly affect the number of correct matches in the simple visual odometry implementation. The ORB settings must be set to small values to extract features in a small patch, which causes the ORB extractor to extract noise as feature points. The simple visual odometry system failed to match features extracted from small patches in Chapter 4.8. Therefore, the patches are not used in the experiment.

Furthermore, it is essential to have a disparity map of good quality. The camera baseline should match the environment. The baseline should be based on where most features are extracted in a dataset. Suppose the camera baseline is not suitable for generating a specific disparity range where most features are extracted. In that case, many features are discarded because there is no depth information on the locations of the features. The baseline should be large (for example, 60 cm as in the KITTI dataset) if most features are extracted on the horizon and small (11 cm as in the EUROC MAV dataset) if most features are extracted close to the camera. The noise in the disparity image also affects the quality of the depth estimation at noisy locations and, thus, the pose estimation. An example of the KITTI dataset in Chapter refch:simpleVOresults shows that the trajectory's APE_{rmse} is 1.56 times worse when a noisy disparity map is used. The low texture in the image can cause noise. The H. Hirschmuller algorithm could have difficulty matching correct blocks or pixels for the disparity map if the texture in the image is low, the camera calibration parameters are not accurate, or if there are illumination differences in the images from the two cameras.

Thus, ORB features are not the direct cause of the drop in the number of feature matches and are suited for real-time visual SLAM. Aspects in the implementation of ORB-SLAM3, such as the division of the image in patches and the type of matcher, are likely causes of the drop in feature matches. Also, the dataset's quality is trivial to extract a good disparity map from the images.

For the Juno project, a simple visual odometry system would suffice for tiny distances. It is recommended to use the current system with 2D-lidar for indoor navigation and to switch to visual odometry for outdoor localization. For the long-term future goal of using the camera solely, more research in ORB-SLAM3 is needed. A starting point would be to test ORB-SLAM3 without using patches for feature extraction and different types of matchers and to experiment on a high-quality disparity map dataset.

5.2. Conclusion

This work aimed to understand the reasons behind the tracking errors caused by the drop in feature matches in ORB-SLAM3 in agricultural datasets. To conclude this work, the two previously determined research questions are answered based on the results seen in the experiments:

- *Why does feature matching often return a small number of matches for the agricultural datasets in ORB-SLAM3?*

  The experiments showed that the drop in the number of feature matches is not necessarily only present in agricultural datasets but also in the widely used indoor and outdoor robotics datasets. There are multiple reasons for this sudden drop in the number of feature matches, which are:

  - Blur, apparent motion, under- and over-exposure in the image. Spearman's correlation coefficient showed a correlation between the number of inliers and optical flow (or apparent motion) of -0.6 on all datasets combined. The sharpness strongly correlates in some of the Exos and Juno sequences with a Spearman's coefficient of 0.66.
  - The type of matcher used can affect the number of matches that can be made. The brute-fore matcher could consistently find more matches and inliers than the bag-of-words-based matcher in ORB-SLAM3, even in frames where ORB-SLAM3 failed. The OpenCV pipeline outperforms ORB-SLAM3 on average by a factor of 22.2 concerning the average number of feature matches on the Juno dataset.
  - The division of the image in patches negatively affects the number of matches. The ORB extraction parameters must be set small to find features in small patches. The features found in such a small patch are mostly noise that the matcher cannot match due to the indistinctiveness of these noisy features.
  - The quality of the disparity map affects the number of feature matches that are used for pose estimation. The disparity map should have accurate depth information at the locations where most features are extracted from the dataset; otherwise, many feature matches are discarded. Moreover, noise in the disparity map causes wrong depth information at the noisy locations and affects the pose estimation accuracy. An example of the KITTI dataset shows that the trajectory's APE_{rmse} is 1.56 times worse when a noisy disparity map is used. However, more work is needed to determine at which stage in the ORB-SLAM3 pipeline the disparity map affects the matches; this is left for future work.

  So the initial hypotheses are both true; however, there is more to the story. The ORB feature descriptor is not descriptive to match features correctly because of multiple reasons, such as the image quality and the division of the image in patches. The drop in the number of feature matches can be related to the ORB-SLAM3 pipeline itself, such as the type of matcher used and the implementation of the image divisor. Additionally, the quality of the disparity map also affects the number of feature matches which was not in the hypothesis initially.

- *Can we implement a simple camera pose estimator that outperforms ORB-SLAM3 in terms of robustness without compromising too much of the accuracy?*

  In the final experiment, a visual odometry system is implemented and compared to ORB-SLAM3 in terms of absolute trajectory error, amount of maps started, and the number of feature matches and inliers. ORB-SLAM3 is the most accurate system in the KITTI dataset, as expected, especially

when loop closures are made. The sequences where ORB-SLAM3 did not fail are sequences 01, 03, 04, 05, and 06. From these sequences, sequences 05 and 06 contain loop closures. The APE_{rmse} of ORB-SLAM3 is lower than the APE_{rmse} of the VO methods for all sequences except sequence 01. The APE_{rmse} of the best VO method in sequence 03 is 2.5 times worse compared to ORB-SLAM3 and 1.26 times worse in sequence 04. The sequences with loop closure show that APE_{rmse} of the best VO method is even worse if ORB-SLAM3 can make loop-closures; the factor is 5.65 for sequence 05 and 3.83 for sequence 06. However, there are still many cases where tracking is lost, after which ORB-SLAM3 could not recover. This resulted in a new map that could not be merged with the previous maps. In general, the visual odometry methods are much more robust compared to ORB-SLAM3. The matching-based VO system has, on average, 49 times more the minimum number of inliers compared to ORB-SLAM3 in the KITTI dataset. However, both VO systems fail in Lely Juno's dark sequences. So, in conclusion, a simple camera pose estimator that outperforms ORB-SLAM3 in terms of robustness could be implemented, but the accuracy is compromised a lot. The initial hypothesis that a simple visual odometry system is more robust than ORB-SLAM3 is true. However, better accuracy is needed for the entire sequences of the KITTI and Lely Juno datasets.

5.3. Future Work

Future work could investigate the effect of the patches in the ORB-SLAM3 pipeline, with larger patches or no patches. Furthermore, the exact effect of the disparity map on the feature matches in ORB-SLAM3 can be investigated further in more detail by analyzing the pipeline and by experimenting with the camera baseline as well as exposure changes and different texture difficulties. Also, the reason for random noisy patches in the disparity maps can be investigated with this approach.

Another future work could be to investigate the effect of different types of matchers in ORB-SLAM3, such as brute force with k-NN, FLANN-based matcher, or any other matcher found in the literature. Preferably matchers that still allow the system to run in real time. For the Juno project, a comparison can be made between the wheel and visual odometry. Wheel odometry can also suffice for the small distance Juno has to travel outside the barn.

Bibliography

[1] Tinku Acharya and Ajoy K Ray. *Image processing: principles and applications*. John Wiley & Sons, 2005.

[2] Pritha Bhandari. *Correlation coefficient*. en. https://www.scribbr.com/statistics/correlation-coefficient/. Accessed: 2022-10-7. Aug. 2021.

[3] Stan Birchfield and Carlo Tomasi. "A pixel dissimilarity measure that is insensitive to image sampling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.4 (1998), pp. 401–406.

[4] Eric Brachmann et al. "On the limits of pseudo ground truth in visual camera re-localisation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6218–6228.

[5] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[6] Michael Burri et al. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1157–1163.

[7] Michael Calonder et al. "Brief: Binary robust independent elementary features". In: *European conference on computer vision*. Springer. 2010, pp. 778–792.

[8] Carlos Campos et al. "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam". In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[9] Gabriella Csurka et al. "Visual categorization with bags of keypoints". In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. Prague. 2004, pp. 1–2.

[10] Mark Cummins and Paul Newman. "Appearance-only SLAM at large scale with FAB-MAP 2.0". In: *The International Journal of Robotics Research* 30.9 (2011), pp. 1100–1123.

[11] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[12] Jakob Engel, Vladyslav Usenko, and Daniel Cremers. "A photometrically calibrated benchmark for monocular visual odometry". In: *arXiv preprint arXiv:1607.02555* (2016).

[13] Gunnar Farnebäck. "Two-frame motion estimation based on polynomial expansion". In: *Scandinavian conference on Image analysis*. Springer. 2003, pp. 363–370.

[14] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[15] Friedrich Fraundorfer and Davide Scaramuzza. "Visual odometry: Part i: The first 30 years and fundamentals". In: *IEEE Robotics and Automation Magazine* 18.4 (2011), pp. 80–92.

[16] Friedrich Fraundorfer and Davide Scaramuzza. "Visual odometry: Part ii: Matching, robustness, optimization, and applications". In: *IEEE Robotics & Automation Magazine* 19.2 (2012), pp. 78–90.

[17] Dorian Gálvez-López and Juan D Tardos. "Bags of binary words for fast place recognition in image sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197.

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[19] Heiko Hirschmuller. "Stereo processing by semiglobal matching and mutual information". In: *IEEE Transactions on pattern analysis and machine intelligence* 30.2 (2007), pp. 328–341.

[20] *Informatik IX Computer Vision Group*. Mar. 2016. URL: https://vision.in.tum.de/data/datasets/rgbd-dataset/file_formats.

[21] Hervé Jégou et al. “Aggregating local descriptors into a compact image representation”. In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 3304–3311.

[22] Kenneth Levenberg. “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[23] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[24] Stephanie Lowry et al. “Visual place recognition: A survey”. In: *IEEE Transactions on Robotics* 32.1 (2015), pp. 1–19.

[25] Jiayi Ma et al. “Image matching from handcrafted to deep features: A survey”. In: *International Journal of Computer Vision* 129.1 (2021), pp. 23–79.

[26] Sherif AS Mohamed et al. “A survey on odometry for autonomous navigation systems”. In: *IEEE Access* 7 (2019), pp. 97466–97486.

[27] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[28] Raúl Mur-Artal and Juan D Tardós. “Fast relocalisation and loop closing in keyframe-based SLAM”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 846–853.

[29] N.H. Nielsen. *ComputerVision*. `https://github.com/niconielsen32/ComputerVision/tree/master/VisualOdometry`. 2022.

[30] José Luis Pech-Pacheco et al. “Diatom autofocusing in brightfield microscopy: a comparative study”. In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 3. IEEE. 2000, pp. 314–317.

[31] Eli Peli. “Contrast in complex images”. In: *JOSA A* 7.10 (1990), pp. 2032–2040.

[32] David Prokhorov et al. “Measuring robustness of Visual SLAM”. In: *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE. 2019, pp. 1–6.

[33] Vijay K Rohatgi and AK Md Ehsanes Saleh. *An introduction to probability and statistics*. John Wiley & Sons, 2015.

[34] Edward Rosten and Tom Drummond. “Machine learning for high-speed corner detection”. In: *European conference on computer vision*. Springer. 2006, pp. 430–443.

[35] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.

[36] Torsten Sattler et al. “Benchmarking 6dof outdoor visual localization in changing conditions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8601–8610.

[37] Davide Scaramuzza and Friedrich Fraundorfer. “Visual odometry [tutorial]”. In: *IEEE robotics & automation magazine* 18.4 (2011), pp. 80–92.

[38] David Schubert et al. “The TUM VI benchmark for evaluating visual-inertial odometry”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1680–1687.

[39] Myriam Servières et al. “Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking”. In: *Journal of Sensors* 2021 (2021).

[40] Jamie Shotton et al. “Scene coordinate regression forests for camera relocalization in RGB-D images”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2930–2937.

[41] Josef Sivic and Andrew Zisserman. “Video Google: A text retrieval approach to object matching in videos”. In: *Computer Vision, IEEE International Conference on*. Vol. 3. IEEE Computer Society. 2003, pp. 1470–1470.

[42] Lukas von Stumberg and Daniel Cremers. “DM-VIO: Delayed Marginalization Visual-Inertial Odometry”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1408–1415.

[43] Jürgen Sturm, Wolfram Burgard, and Daniel Cremers. “Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark”. In: *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*. Vol. 13. 2012.

[44] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[45] Raad H Thaher and Zaid K Hussein. “Stereo vision distance estimation employing SAD with canny edge detector”. In: *International Journal of Computer Applications* 107.3 (2014).

[46] F. Walch et al. “Image-based localization using LSTMs for structured feature correlation”. In: *ICCV*. Oct. 2017.

[47] *Wiki*. URL: `http://wiki.ros.org/melodic`.

[48] Georges Younes et al. “Keyframe-based monocular SLAM: design, survey, and future directions”. In: *Robotics and Autonomous Systems* 98 (2017), pp. 67–88.

[49] Huangying Zhan et al. “Visual odometry revisited: What should be learnt?” In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4203–4210.

[50] Zichao Zhang and Davide Scaramuzza. “A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7244–7251.