UNIVERSIDADE D
COIMBRA

Rita do Rosário Singéis

PERFORMANCE ANALYSIS OF VISUAL ODOMETRY AND VISUAL SLAM IN FOGGY ENVIRONMENTS

Master's Dissertation in MIEEC, supervised by Professor Doctor Lino Marques and Doctor Sedat Dogru and presented to the Department of Electrical and Computer Engineering of the Faculty of Science and Technology of the University of Coimbra

Setembro de 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Performance Analysis of Visual Odometry and Visual SLAM in Foggy Environments

Rita Singéis

Coimbra, September 2023

UNIVERSIDADE D
COIMBRA

Performance Analysis of Visual Odometry and Visual SLAM in Foggy Environments

Supervisor:

Prof. Doutor Lino José Forte Marques

Co-Supervisor:

Doutor Sedat Dogru

Jury:

Prof. Doutor Rui Rocha

Prof. Doutor Cristiano Premebida

Prof. Doutor Lino Marques

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2023

Acknowledgements

Here I'd like to thank everyone who has helped me in my academic journey.

First, to my supervisor Prof. Dr. Lino Marques, I thank the opportunity of working in a project that greatly interests me and for all the advice and help provided in the last year.

Then, I would like to thank my co-supervisor, Dr. Sedat Dogru for all the patience, transferred knowledge and especially assistance that allowed me to finish this project of which I'm so proud.

To my parents for always pushing me to follow my dreams, wherever they take me, and for supporting me. I wouldn't have made this far without all your help, encouragement and willingness to be better and believe in myself. To my cousins, Marta and Patrícia I literally wouldn't be in this degree, in this university, if it wasn't for you two, so thank you very much.

Last but not least, thank you Marta and André for always hearing me (even when you don't understand what I'm talking about), for supporting me in all my decisions, for standing by me no matter what, and for helping me to the finish line that is this project. Love you guys.

Resumo

As técnicas de odometria visual (VO) e localização e mapeamento simultâneos baseados em visão (SLAM) dependem principalmente de características reconhecíveis que podem ser rastreadas e detectadas em várias imagens. Assim, os métodos modernos utilizam características que são invariantes à escala, observáveis de vários ângulos e tolerantes a alterações na iluminação. No entanto, neblina, nevoeiro ou névoa podem reduzir a visibilidade dos pontos característicos. O desempenho das técnicas SLAM/VO baseadas em visão é afetado por fenómenos atmosféricos que mudam no decorrer do dia. Para investigar os efeitos do nevoeiro, o sistema ORB-SLAM foi o método escolhido como o principal desta dissertação. Entre as diversas versões de ORB-SLAM existentes, esta dissertação teve como foco o ORB-SLAM2 e o ORB-SLAM3, comparando o seu desempenho. Notavelmente, ORB-SLAM3 oferece dois modos operacionais: só localização (o que consiste em odometria visual) e SLAM, garantindo assim uma análise extensiva. As métricas de desempenho abrangeram vários parâmetros, entre os quais o número de características detectadas, a percentagem de estimativas válidas em relação ao total calculado e a média quadrática quer no eixo vertical quer no eixo horizontal. Esta rigorosa estrutura de avaliação permite comparações significativas entre as diversas implementações dos sistemas, possibilitando assim uma análise direta entre os dois sistemas mencionados e consequentemente tirar conclusões mais corretas e verosímeis.

Abstract

Visual Odometry (VO) and vision-based Simultaneous Localization and Mapping (SLAM) techniques mainly rely on recognizable characteristics that can be tracked and detected over several frames. Such modern methods make use of characteristics that are scale-invariant, observable from various angles, and tolerant of alterations in light. However, haze, mist, or fog can reduce the visibility of the feature points. The performances of vision-based SLAM and VO techniques are effectively affected by atmospheric phenomena that change during the day. To investigate the effects of fog, ORB-SLAM was chosen as a case study in this dissertation. Among the various versions of ORB-SLAM, this work focused on ORB-SLAM2 and ORB-SLAM3, comparing their performance. Notably, ORB-SLAM3 offers two operational modes: localization only (VO) and SLAM, warranting a comprehensive analysis. Performance metrics encompassed the number of detected features, the percentage of valid estimates relative to the total computed, and Root Mean Square Errors (RMSE) error in both vertical and horizontal axes. This rigorous framework of evaluation enables meaningful comparisons across the various implementations of the systems allowing for a direct analysis between the two mentioned systems and consequently draw more accurate and truthful conclusions.

"I think it's very important to get more women into computing. My slogan is: Computing is too important to be left to men."

— Karen Spärck Jones

Contents

List of Acronyms

BA	Bundle Adjustment
BoW	Bag of Words
BRIEF	Binary Robust Independent Elementary Features
CAD	Computer-aided Design
DCP	Dark Channel Prior
EKF-SLAM	Extended Kalman Filter SLAM
FAST	Features from Accelerated Segment Test
FoV	Field of View
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
g2o	General Graph Optimization
GPU	Graphical Processing Units
IMU	Inertial Measurement Unit
KF	Keyframes
LiDAR	Light Detection and Ranging
LBD	Line Binary Descriptor
ORB	Oriented FAST and Rotated BRIEF
PL-SLAM	Point and Line SLAM
RGB	Red, Green and Blue

RMS	Root Mean Square
RMSE	Root Mean Square Error
RTK-GNSS	Real Time Kinematic Global Navigation Satellite System
SLAM	Simultaneous Localization and Mapping
UGV	Unmanned Ground Vehicle
VO	Visual Odometry
3D	Three Dimensional

List of Figures

List of Tables

1 Introduction

Nowadays autonomous vehicles are seen as a solution to several problems, such as traffic congestion, car accidents or even improving accessibility to people with disabilities. According to Kuutti et al. [5] an autonomous vehicle is expected to contain five algorithmic building blocks, namely: localization, perception, planning, control, and system management for proper operation. A rough estimation of localization can be achieved using Global Navigation Satellite System (GNSS), and this is already a widespread localization mode on the roads, helping drivers find their ways using navigation applications. However, particularly in urban areas with high rise buildings as well as in tunnels, availability and accuracy of GNSS suffers significantly, failing to provide high quality data to allow accurate and safe control of the vehicles [6]. As a result, autonomous driving requires complementary solutions relying on local sensing, using sensors such as sonar, Light Detection and Ranging (LiDAR), radar and cameras [7, 8]. These sensors may also completely eliminate the need for GNSS in a Simultaneous Localization and Mapping (SLAM) configuration [9]. Using SLAM an autonomous vehicle can create and update its map as it moves.

In this context, cameras can be used both for SLAM and also for semantic scene understanding, such as recognizing traffic signs or pedestrians, eventually triggering proper behaviour in a more natural manner [7]. Cameras, instead of being part of a SLAM algorithm, can also be used as part of a Visual Odometry (VO) system, in which the egomotion of an agent is estimated exclusively from the input of one or more cameras that are connected to it [10]. VO is similar to wheel odometry, because it incrementally estimates the position of the vehicle by analyzing the changes motion causes to the data captured by its onboard cameras. In VO the main concern is the local consistency of the map, and the local map is utilized to produce a more precise estimate of the local path. However in SLAM the primary concern is the global consistency of the map. Hence, both approaches are closely related, and a full SLAM algorithm may be build using VO.

Both VO and Visual-SLAM rely on detecting and matching invariant features in images.

A feature point is taken to be invariant when its signature stays consistent as illumination, field of view, and its distance to the cameras vary. To achieve VO and Visual-SLAM, methods based on point features such as FAST [11], Shi-Tomasi [12] or ORB [1,3] and methods based on line features have been studied [13–15], among others. However, fog and haze can reduce visibility significantly, possibly hindering feature detection.

Fog is made of suspension of water droplets in the atmosphere, which can reduce visibility down to few meters. Haze is made of high levels of air pollutants, such as dust, smoke, and other particles that scatter and absorb sunlight diminishing visibility as well [16]. Both environmental phenomena work as obstruction to light and therefore worsen the quality of the images seen and processed by the algorithms, resulting in poorer results. This can have damaging consequences for instance, resulting in road accidents. Although image processing continues to be developed, and obstructed views or bad image quality is not an obstacle to image processing, the same cannot be said for implementations that aim to identify objects, match features and track them [17]. This means that fog removal is a prominent problem to be tackled. In order to solve this problem, various image enhancement approaches have been developed [18–27]. However, to the best of our knowledge, nobody has studied neither the effect of defogging algorithms on SLAM, nor the effect of fog induced low visibility on SLAM. In this work we study the effect of fog on the performance of visual SLAM, and show that defogging algorithms can help partially alleviate the problem.

1.1 Objectives

The objective of this work is to study and analyse the quality of the final path produced by SLAM and VO as well as the variations in the amount and quality of the features with different levels of fog.

1.2 Contributions

To the best of the our knowledge, this dissertation presents the first work to present an analyse the effects of foggy images when used with ORB-SLAM and a proposed solution to increase its performance in these conditions. It is also the first work to introduce realistic fog on a real dataset using a stereo system.

Some of the work in this dissertation was submitted to the 6th Iberian Robotics Conference (ROBOT2023) [28].

This work was also partially done in the scope of the project Intelligent Logistic Autonomous Fleet (ILAF) with reference POCI-01-0247-FEDER-072534.

1.3 Document Structure

The following describes the organization of this dissertation and the content of each chapter:

- **Chapter 2** presents some crucial concepts mainly regarding how the environment is represented in robotics and some researched approaches that were used in this work as well as literature surveys on various Visual SLAM and VO approaches, and on defogging techniques.
- **Chapter 3** presents a more thorough theoretical exposition of the methods used in this proposal.
- **Chapter 4** outlines the developing and execution of the suggested strategy for Performance Analysis of Visual Odometry and Visual SLAM in Foggy Environments. Therefore it covers an overview of the system, its way of working and a complete clarification of each stage.
- **Chapter 5** incorporates experimental findings that were attained. This chapter also includes a discussion of these findings and their significance;
- **Chapter 6** considers the general outcomes of the research and makes some suggestions for refinements to be made in subsequent efforts.

2 Related Work

In this chapter a theoretical background for the basic concepts used in the scope of this dissertations are presented. The reader is presented with a background of different implementations of point feature based , line based and consequently point and line based visual SLAM approaches. Additionally, the reader is also presented with the current framework in defogging approaches, a main topic of discussion in this dissertation.

2.1 Environment Representations in Robotics

The key issue with SLAM is determining if a mobile robot can be placed in an unknown region and gradually build an accurate map of that environment while also determining its position within that map. Because of this, a recursive answer would be ideal. This problem can be expressed in a probabilistic way [9], where the joint posterior density of the landmark locations is calculated for all times. A method addressing the SLAM problem depends on two important models: the observation model and the motion model. The observation model represents the likelihood of making an observation when the position of the vehicle and landmarks are known. The motion model explains that the state transition is a stochastic model (Markov process) in which the subsequent state is independent of the observations and the map and only depends on the previous state and the applied control. There are several computational solutions to SLAM, namely Extended Kalman Filter SLAM (EKF-SLAM) and SLAM using Rao-Blackwellized particle filters (FastSLAM). EKF-SLAM [29] is used when the problem is represented in a state-space model with additive Gaussian noise. On the other hand, FastSLAM [30] is used when the vehicle motion model is described with a set of samples of non-Gaussian probability distribution. Both these methods are explained more thoroughly in [9].

2.2 Point Feature Based Visual SLAM Approaches

Point feature visual SLAM approaches use as their main source of information, points that when of interest are considered keypoints. These keypoints are identified over several frames and consequently matched comparing their descriptors which allows for the information regarding the estimation of the camera pose. The feature then, is constituted by the descriptor and the keypoint, [31].

ORB-SLAM [1–3], a state of the art method for Visual SLAM, is built on top of four main blocks - map, tracking, local mapping and loop closure. The map block stores a set of Keyframes (KF) with observed stereo features and their descriptors and information of the Three Dimensional (3D) camera pose, covisibility graph, where each node represents a KF and edges between them allowing for real-time Bundle Adjustment (BA) of the local map, and a spanning tree, the minimum connected representation of a graph that has all the KFs. The feature tracking block represents the algorithm that tracks image features from a sequence of stereo frames and computes their 3D position. Afterwards the 3D landmarks are projected onto the new camera pose and the projection errors are minimized in order to obtain camera pose increment. This is repeated for every frame, until a new KF is inserted. Once inserted two different threads are run in parallel (local mapping and loop closure). The local mapping thread searches for new feature correspondences between the new KF, the last one and the ones connected to the last one through the covisibility graph. This allows for the local map of the current KF to be built. This local map has all the KFs that share at least a predefined number of landmark observations with the current one as well as all the landmarks observed by them. The last step in this thread is the optimization of all the elements within the local map. Finally the loop closure thread works by extracting a visual descriptor for each image, based on a Bag of Words (BoW) approach. The visual characteristics of the collected KFs while the camera is moving are saved in a database, which is then used to discover comparable frames to the present one. Only if the local sequence around this KF is comparable will the best match be deemed a loop-closure candidate.

2.3 Line Based Visual SLAM Approaches

Line based Visual SLAM approaches mainly differ from point feature based ones by identifying features with lines instead of points. They were firstly introduced in the filtering stage of SLAM [32, 33], then in the optimization with BA stage [13, 34, 35] and finally in the loop

closure phase of visual SLAM [36].

Ma et al. [37] presented JunctionSLAM. This method uses coplanar junction detection to detect the lines. It first uses the Douglas-Peucker algorithm [38] to detect them and afterwards it builds a putative coplanar junction. Thereafter, junction matching is done using a multi-scale rotated BRIEF descriptor and subsequently the line matching is executed by extracting two line matches after obtaining a junction match between two frames. After, the junctions and the BRIEF descriptors, are used in the line tracking, mapping and loop closure threads. It was also designed a cost function to minimize both the reprojection error of line segments and the alignment error of the vanishing points. As the other state of the art SLAM methods, this one has three threads: motion estimation, local mapping and loop closing. In the motion estimation phase, for each new stereo frame, the algorithm detects the lines segments in order to do the line matches. It also uses the vanishing point extraction method [39] in order to cluster lines in the left image and then compute the coordinates of vanishing points in the image plane. The motion is estimated using trifocal tensor geometry. Concerning motion estimation, at least 3 line matches between the left image of the current frame and the last keyframe are needed. Regarding the local mapping phase, when a new keyframe is added to the pose graph, the 3D coordinates of line segments in this new keyframe are reconstructed by intersecting two planes. The camera poses and line segments that belong to active keyframes are refined by local BA. After this process, a culling strategy is used to remove outliers from line matches or vanishing points clusters. Finally in the loop closure detection phase, the procedure is similar to ORB-SLAM but this method uses junctions as feature points. BoW are also used as representation to do loop closure detection and relocalization. Concerning experimental results, on synthetic data, the rotation error and translation error are reduced by involving vanishing point in the BA. On the other hand, in real data, the proposed JunctionSLAM works well although in conclusion it is better suited for line-rich environments only. Whereas in texture-rich ones, there might be not enough line segment features and therefore point-feature SLAM approaches are more suitable.

2.4 Line and Point Feature Based Visual SLAM Approaches

Line and point feature approaches combine the methods mentioned in the two previous sections.

Gomez-Ojeda et al. [13] proposed PL-SLAM, a stereo approach that is heavily based on

ORB-SLAM so it also uses the map, feature tracking, local mapping and loop closure blocks. However it takes into account lines as well, aiming for regions particularly poor in features. Respecting the map block, it has a set of KF that contain the observed stereo features, their descriptors and information of the 3D camera pose, detected 3D landmarks - list of observations and for each its corresponding more representative descriptor. It also stores the estimated 3D position of points and direction and estimated 3D coordinates for lines. In addition it creates a covisibility graph where each node represents a KF, and the edges between them their relation, allowing for real-time BA of the local map. Finally it creates a spanning tree which represents the minimum connected representation of a graph that has all the KFs. Concerning the feature tracking block, the algorithm tracks image features from a sequence of stereo frames and computes their 3D position and associated uncertainty. Afterwards the 3D landmarks are projected onto the new camera pose and projection errors are minimized in order to obtain both camera pose increment and the associated covariance. This process is repeated for every frame until a new KF is inserted. Once inserted, two different threads are run in parallel - local mapping an loop closure. Regarding the local mapping block, it searches for new feature correspondences between the new KF, the last one and the ones connected to the last one through the covisibility graph. This allows the local map of the current KF to be built. This local map has all the KFs that share at least 20 landmark observations with the current one as well as all the landmarks observed by them. The last step in this thread is the optimization of all the elements within the local map. At last, the loop closure block extracts a visual descriptor for each image, based on a BoW approach. The visual characteristics of the collected KFs while the camera is moving are saved in a database, which is then used to discover comparable frames to the present one. Only if the local sequence around this KF is comparable will the best match be deemed a loop-closure candidate. In terms of experimental validation, this algorithm was tested in the EuRoC MAV and KITTI dataset and a low-textured scenario captured by the authors. In the EuRoC MAV dataset, PL-SLAM is successfull in all sequences. In the KITTI dataset, PL-SLAM also completed successfully the trajectory estimation for all sequences. Finally, in the low-textured scenario, the line based approach is capable of robustly estimate the camera path in all sequences and also have a good performance in terms of accuracy.

Lim et al. [15] extends [12] using vanishing points obtained from line features, as well as the lines themselves comparing them using Line Binary Descriptor (LBD) [40]. The point features are extracted using Shi-Tomasi algorithm [41] and are tracked by KLT. Also, the IMU measurement model is defined by the pre-integration method. Finally, the optimization-

based method employs a two-way marginalization with Schur complement [42]. The line features are extracted with LSD and tracked with LBD. To represent lines in 3D, they use Plücker coordinates and orthonormal representation. The algorithm functions as follows. First vanishing point hypotheses with random sampling for all extracted lines is created. Then similar lines through comparison between the hypotheses are merged and afterwards the vanishing points are possible to be calculated. Since the algorithm detects vanishing points through J-linkage, this method allows to find all vanishing points through the hypotheses. Due to the ability of the line features being fully observable using the vanishing point measurements, this algorithm achieves better results than other state of the art algorithms,

2.5 Defogging Approaches

Image dehazing methods can be divided into four categories: image enhancement, image restoration, fusion based and deep learning based. Image enhancement type of techniques focus on improving the human perception of the image usually by enhancing contrast, such as using histogram equalization [25]. These type of methods exhibit simplicity and efficiency, however they lack adaptation and perform poorly in complex scenarios. Image restoration based approaches, in contrast to the previous family of approaches, take into account the physical process that causes image degradation, and tries to improve the images by taking into account this process. For this several different approaches have been implemented based on atmospheric scattering model [24], a color attenuation prior [26] or a haze-line prior [27]. The third family of approaches uses multiple real or derived images of the same scene [22,23], sometimes taken at infrared frequencies, extracts different features from the different images, combines those features and performs deblurring. This type of method is known for its high robustness, good anti-cluttering effectiveness, and it is typically faster than deep learning-based methods. Lastly, in recent years, deep learning-based haze removal methods have become the conventional research. Some of learning based methods take into account the atmospheric scattering model [20, 21]. They derive a dispersion map using a deep learning network and use the map to extract sharp images from an atmospheric scattering model. With the development of deep learning techniques, more and more methods such as attention, generated adversarial networks and knowledge transfer are being applied to the field of image dehazing. These methods tend to be end-to-end without the help of atmospheric scattering models, use realistic fog-free image pairs for training, and show better performance [18, 19].

Deep learning-based methods have proven to be superior to conventional methods in terms of robustness and effectiveness, and generally perform better in scenes with dense and uneven haze. However, they require significant computational power and Graphical Processing Units (GPU).

He et al. [24] presented in 2009 a straightforward and efficient soft matting Dark Channel Prior (DCP) based single-image haze reduction technique, which is heavily used by later researchers as part of their method. DCP basis is the statistics of outdoor images that don't have any haze. It is also stated that the majority of the local regions that don't include the sky (dark pixels), have a very low intensity in at least one of the RGB channels. This is a direct result from airlight. Using this information, the transmission of haze may be accurately estimated from the dark pixels. Afterwards a haze-free image and a depth map can be reconstructed by using a haze imaging model and a soft matting interpolation technique.

3 Methods

This chapter presents a detailed introduction to ORB-SLAM, showing its time evolution, detailing the different parts, and highlighting the differences between the different versions. This is followed by a description of the methods used to calculate ORB features, which are the backbone of ORB-SLAM. Afterwards it also presents an introduction to both the fog and defogging model that are used in this project.

3.1 ORB SLAM

ORB-SLAM was first proposed by Mur-Artal et al. in [1]. In this, the authors propose a new monocular SLAM system that offers several contributions such as: using the same features for all tasks (tracking, mapping, relocalization and loop closing) and also using ORB [43] features (explained more thoroughly in 3.4). It also offers a real-time procedure for vast environments. This is done by using a covisibility graph, which allows the tracking and mapping threads to be able to focus on a local covisible region. Besides, it uses an Essential Graph which is built from a spanning tree sustained by the system, loop closure links and strong edges derived from the previously mentioned covisibility graph. This allows for real time loop closing. In addition it provides recovery from tracking failure and allows for map reuse by having camera relocalization in real-time with notable invariance to illumination and viewpoint. Moreover, an initialization method based on model selection that grants the possibility of creating an initial map of planar and nonplanar scenes is also proposed. Finally, regarding the problem of keyframe selection, a survival of the fittest approach is applied in order to upgrade the robustness of the system by rejecting redundant keyframes.

In Figure 3.1 the system overview of ORB SLAM and therefore its main components and blocks can be seen. Tracking (Section 3.1.3), local mapping (Section 3.1.4) and loop closing (Section 3.1.5) threads are run in parallel. In the following sections, all of these will be explained more comprehensively.

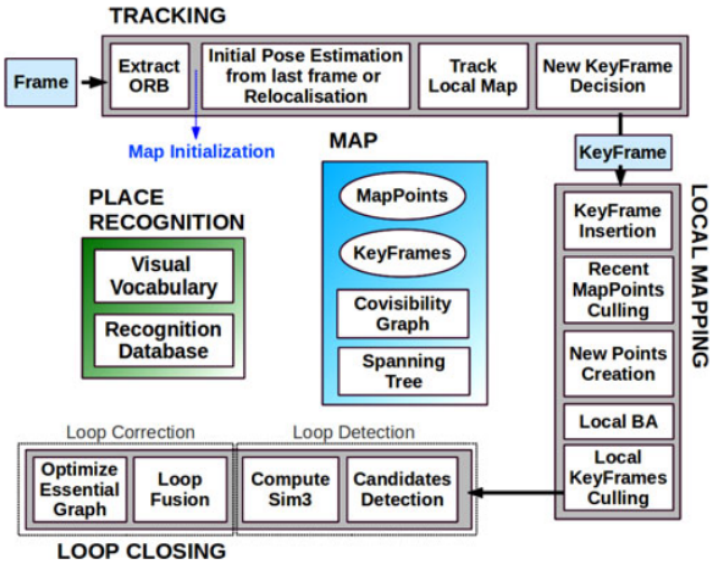


Figure 3.1: Overview of the ORB SLAM system with all the main blocks represented [1].

3.1.1 Place Recognition

This block, while not being a thread that runs in parallel with the other threads, it allows for the system to have a BoW based on [44], which in turn allows for loop detection and relocalization. The vocabulary is generated offline from a set of images. The system creates a database that has the information concerning each visual word and the associated keyframe. Whenever a keyframe is deleted during the culling process, the database is updated accordingly. The usage of BoW also enables a quicker search when comparing different sets of ORB features since it permits the matching of only the features that are related in the same node of the vocabulary tree.

3.1.2 Map

The map block contains information about map points, keyframes, the covisibility graph and the spanning tree.

Each map point stores the corresponding 3D position in the world coordinate system, the viewing direction, the ORB descriptor and the maximum and minimum distances where the point can be observed. In turn, each keyframe stores the camera pose - defined as a rigid body transformation that allows the conversion between world and camera coordinate system, the camera intrinsics and every ORB feature extracted in the correspondent frame.

The covisibility graph is a weighted graph where each node is a keyframe and an edge between two nodes represents the information common to both keyframes. The weight of

the edge is represented by a variable θ and it is the same as the number of common map points. In order to allow for a more robust loop correction, an essential graph is created, where every node, and therefore keyframe, is represented but not all edges. Nevertheless all prominent information is presented. Stemming from these last two concepts, a spanning tree is also created, which is built incrementally since the first keyframe, providing a subgraph of the covisibility graph but with less edges. Every time a keyframe is updated, either inserted or removed, the tree and its links are also updated. When inserted, the keyframe is linked to the edge that shares the most observations regarding points, and when removed, the system removes all links previously connected to that specific keyframe. This means that the spanning tree is within the essential graph

3.1.3 Tracking

This thread is responsible for localizing the camera within every frame and afterwards deciding when a frame is worthy of being considered as a new keyframe. Firstly, the system extracts the ORB features from the frame by extracting FAST corners in eight-scale levels, with the number of the corners to extract depending on the resolution of the frame. Afterwards, each scale is divided in to a grid so that in each cell of the grid, corners are detected. Here, the number of corners to be extracted is adjusted, if necessary. Following this, orientation and ORB descriptor are calculated onto the retrieved FAST corners. The ORB descriptor is later used for feature matching.

Later, the system checks if tracking (the whole procedure of this thread) was successful for the last frame. If yes, then a constant velocity model is used to foresee the camera pose and search for map points in the last frame. Depending on the number of correspondences, the search is widened, or not, and then the pose is optimized taking into account the correspondences.

Thereafter, if tracking is lost, the proposed solution is the implementation of a place recognition module. This module is used in these situations and allows for global relocalization. Again, if tracking is lost, the frame is converted into BoW and the database is searched for a potential keyframe correspondence, allowing then for global relocalization. The camera pose is only optimized if enough inliers are found, allowing for the tracking thread to continue its functioning.

After all these steps are executed, the method has an initial estimation of the camera pose and an initial set of feature matches. This allows for the creation of the local map.

The local map contains a set of keyframes that shares map points with the current frame. It also contains a set of keyframes that are neighboring of the first set of keyframes in the covisilibility graph. Adding to this, it also has a reference keyframe which is a keyframe that (d)share the majority of the map points with the current frame. Finally, each map point that is detected in both sets of keyframes is searched in the current frame so that the camera pose can be at last optimized with all map points found in the frame.

Lastly, the last phase in this thread is how to decide if the current frame is deemed a keyframe. As a way to make this procedure faster, new keyframes are inserted rather quickly, since there is a step in the next thread - local mapping - to remove redundant keyframes. So, for a frame to be considered a keyframe, the following criteria must be satisfied. First, there has to be at least 20 frames since the last global relocalization. Then, either the local mapping thread is busy or there has been more than 20 frames since the last keyframe insertion. Then, the current frame has to have at least 50 tracked points. Lastly the current frame has to have less than 90% of the tracked points in the reference keyframe.

3.1.4 Local Mapping

As seen in Figure 3.1 the local mapping thread has several phases: keyframe insertion, recent map points culling, new points creation, local BA and local keyframes culling.

In the keyframe insertion phase the covisibility graph is updated by adding a new node for the current (new) keyframe and consequently updating the edges. The spanning tree is also updated by linking the current keyframe with the one that it has the most points in common. Afterwards the BoW is updated respectively with the keyframe.

Then, in the recent map points culling phase, the map points that were previously detected, matched and stored, are only kept if they satisfy the following conditions throughout the next three keyframes: the same point has to be found in more than 25% of the predicted frames and it must be seen from at least three keyframes if more than one keyframe has passed since the formation of the map point. This means that a map point can only be erased if isn't found in at least three keyframes. This allows for the map to not contain a lot of outliers.

In the new map point creation phase, new map points are generated by using the information provided in the covisibility graph and triangulating ORB information from the connected keyframes. Triangulation is performed to each ORB pair and they are considered as new points only after some parameters are checked, such as parallax, reprojection error,

scale consistency and depth in both cameras. In the case of an unmatched ORB, a matched is examined with other unmatched point in another keyframe.

Local BA is used in this approach in order to optimize the currently processed keyframe, every single keyframe linked to it in the covisibility graph and consequently all map points marked in those same keyframes. In this stage it is also where any outliers are removed.

Finally, the local mapping thread finds surplus keyframes and removes them. This allows the robustness of the system to continue. These keyframes are only removed if at least, 90% of map their points have been marked more than three keyframes.

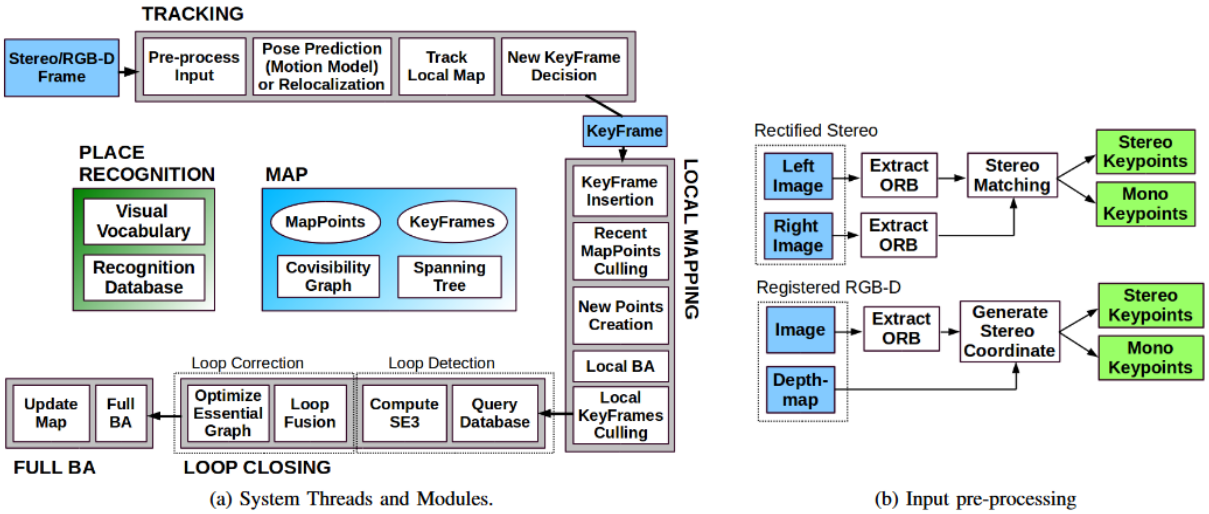
3.1.5 Loop Closing

While the last two sections deal with several frames, the current thread only uses the last keyframe handled in the local mapping block. The current block detects and closes loops. First it needs to analyze which loops are required to close and which are not. To compute this the correspondence between the BoW of the current frame and the the BoW of its neighbors represented in the covisibility graph is used. Then, only the lowest score is stored. If the score is higher than the lowest score, when searching the database, those keyframes are removed. Afterwards, for the loop to be deemed a candidate to loop closing, the thread has to detect at least three consistent loop candidates. This happens if they are connected in the covisibility graph.

In order for a loop to be able to be closed, the similarity transformation between the current keyframe and the loop keyframe needs to be calculated. This process is done by computing the correspondences between ORB in the current keyframe and in the loop candidate. If a similarity with enough inliers are found, then it is optimized.

For a loop to be corrected, the covisibility graph needs to be updated by adding new edges, which allows for the loop to be closed and the map points that are repeated need to be fused. With the similarity transform that was calculated in the last step, the current keyframe pose is updated and therefore all neighbors are also updated. The map points that are matching and the ones that are considered inliers in the computation of the similarity transformation are fused. This means that all keyframes that are used in this process update their edges in the covisibility graph.

At last, an optimization of the pose graph represented by the Essential Graph is done, allowing for an effective loop closure. This represents the propagation of the correction of the error onwards in the graph.



(a) System Threads and Modules. (b) Input pre-processing

Figure 3.2: Overview of the ORB SLAM2 system with all the main blocks represented [2].

3.2 ORB SLAM2

After presenting ORB SLAM, Mur-Artal et al. presented ORB SLAM2 [2] which is built upon the original ORB SLAM but whereas the original was created for monocular camera systems, the improved one can be used by stereo and RGB-D cameras. In Figure 3.2 it can be seen that the overall view and main threads of the system are fairly similar to Figure 3.1, but it has some additions, mainly the preprocessing input block and the full BA thread.

Since this system can either use stereo or RGB-D frames, the preprocessing was created so that the relevant keypoints can be stored and the images dropped and therefore for the rest of the ORB SLAM code the type of sensor that was used is irrelevant. This allows the system to handle both monocular (explained in section 3.1) and stereo keypoints.

Stereo keypoints are defined by the three coordinates: the coordinates in left image and horizontal coordinate in the right image. In the case of stereo cameras ORB features are obtained from both images and then for each feature in the left image, a matching feature is found in right image. In the case of RGB-D cameras, the ORB features are extracted in the left frame and then a virtual right coordinate is attained by transforming its depth value. This allows for the keypoints obtained from RGB-D cameras to still be handled as stereo keypoint. A stereo keypoint can be considered as a close or as a far keypoint depending on its depth. If the depth is more than baseline times 40, then the keypoint is a far point, otherwise it is a close point. While points that are far yield more information regarding rotation, translation and scale, close keypoints allow for triangulation from one frame if depth is precisely computed.

On the other hand, monocular keypoints are described with two coordinates from the left image. Because these points coincide with all ORB features that don't have a match, or in the RGB-D case, don't have enough depth, these points only supply information concerning rotation and translation.

Another change from the original ORB SLAM proposal, is the use of BA in various blocks. In the tracking thread motion-only BA is used. In the local mapping thread only local BA is used, and in the loop closing thread full BA is used. Motion-only BA only optimizes the camera position and orientation between 3D points in world coordinates, and stereo, or monocular, keypoints. Whereas local BA optimizes several covisible keyframes and all points seen within. Lastly, full BA is the same as the local BA but now all keyframes and their points are optimized.

Regarding, loop closing, the novelty in this system is that the optimization is not based on similarities but instead on rigid body transformations.

At last, the final innovation is a localization mode. This module deactivates both the local mapping and the loop closing threads. The meaning of this is that the tracking thread uses visual odometry by matching ORB features between the present frame and the 3D points generated in previous frame.

3.3 ORB-SLAM3

ORB-SLAM3 was presented by Campos et al. [3] and it is based on previous iterations of ORB SLAM, [1, 2] and on ORB SLAM visual-inertial [45]. One of the main novelties in this system is the use of multimap data association, since the systems used as basis use short, medium and long term data association and here all of those benefits are put to use. Multimap data association grants the opportunity to use in BA features detected in other maps from other instances where the same environment was mapped. A system overview can be seen in Figure 3.3 The authors also comprise the contributions of this system by enumerating several new features.

The first is that the system relies on maximum a posteriori estimation for the IMU initialization. The second contribution is a renewed way to solve place recognition. Instead of using just the DBoW2 library [44] an algorithm is proposed in which frames classified as potential keyframes are first tested for geometrical consistency and only afterwards for local consistency, with three covisible keyframes; whereas in DBoW2 first temporal consistency and then geometric consistency are checked. This allows for more data association, although

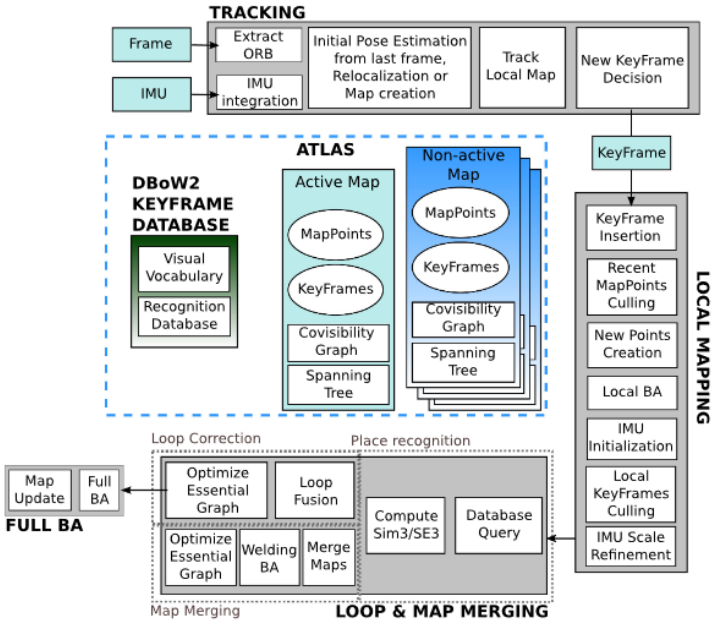


Figure 3.3: Overview of the ORB-SLAM3 system with the main blocks represented [3].

at a higher computational cost. ORB SLAM 3 also introduced an Atlas, which manages several disconnected maps (nonactive maps) and use the mapping processes on them. The active map is the one used by the local mapping thread (3.1.4. Then, a DBWo2 database of keyframes is generated and subsequently used in the relocalization, loop closing and, because the maps are disconnected, map merging tasks. Finally, an abstract camera representation is also introduced. This means that the code does not depend on the model of the camera but instead on its projection, unprojection and Jacobian features.

In the tracking thread, the reprojection error regarding matched features is reduced by calculating the the position of the current frame in relation to the active map at the time. If the system is in visual-inertial mode, the velocity and IMU are computed taking into account the inertial residuals at the time of optimization. If the tracking is set to lost, this thread uses maps of the Atlas to relocalize. If this task is successful, then the tracking thread is allowed to continue its work. It is also the job of this thread to classify the current map as a nonactive map and to switch active maps if needed. The local mapping thread is also tasked with adding keyframes, pointing them to the active map, removing redundant keyframes and using BA to cleanse the map. Furthermore, if IMU is being used, then its parameters are initialized and refined in this thread also. Lastly, in the loop and map merging thread, the maps stored in the Atlas are taken into account and one of its tasks is to detect shared regions between the active map and the Atlas, for every keyframe. This means that in the eventuality of that shared area belonging to the active map, loop correction is performed. If

not, then both the active map and the map where the region belongs, are merged into one, which then evolves into the active map. The last task is performing full BA to the map, although that process is independent in order to not impact the performance of the system.

3.4 ORB-Features

Oriented FAST and Rotated BRIEF (ORB) features [43] are described as binary descriptors based on BRIEF [46] but it mainly consists of using an oriented FAST detector and a rotated BRIEF descriptor [43] to detect the intended features.

The FAST algorithm was first proposed by Rosten et. al, [47], and it is described in Algorithm 1.

Algorithm 1: FAST algorithm

Input: Image

Output: Corner

Choose a pixel p in the image that will be used to determine whether it is an interest point or not. Let I_p be the correspondent intensity. ;

Select suitable threshold, t.;

Assume a 16 pixel circle surrounding the pixel being tested.;

p is a corner if there is a set of n adjacent pixels in the circle that are brighter than $I_p + t$ or darker than $I_p - t$.;

A high speed test is performed in order to exclude non-corners. The test is performed by, in the case of a circle with 16 pixels, looking into pixels 1, 9 and if they are too bright or too dark, it looks into pixels 5 and 13. If p is a corner then at least three of the mentioned pixels must be brighter than $I_p + t$ or darker than $I_p - t$. If this doesn't happen, then p is not a corner. This is done iteratively to all pixels.

Although this algorithm is known and used due to its high performance, it also has some weaknesses. Some of them are the following:

- The selection of pixels is not ideal since its effectiveness depends on the distribution of corner occurrences and the sequence of the questions
- High-speed test results are discarded.
- There are several features located adjacently.

- When $n < 12$, the high-speed test does not generalize well.
- FAST features do not have an orientation component.

To solve problems due to the lack of orientation and corner distribution in FAST features, Rublee et. al [43] proposed using the Harris corner measure [48] to sort the FAST keypoints. Afterwards the intensity centroid is used to assign the corner orientation [49]. This method assumes that the intensity of a corner is offset from its center and therefore this might be used to assign orientation. This all results in oFAST, a FAST keypoint orientation.

On the other hand, BRIEF descriptors function by using binary strings as an efficient feature point descriptor [46]. Calonder et. al [46], building on previous works [50, 51], purely create a bit vector from the test responses done on the smoothed image patch. The flow of the computation is represented by algorithm 2. BRIEF offers advantages when compared to other descriptors such as faster computation time and high recognition rates but on the other hand it is only a feature descriptor so it does not present any method to actually find the features. Furthermore BRIEF descriptors do not perform well when faced with rotations, so Rublee et. al [43] also presented a solution to this problem.

The solution encountered was to steer BRIEF according to the orientation of the keypoints. This results in a loss of variance. So, in order to tackle this issue and to minimize association with binary tests, a learning method was developed to choose a suitable subset of binary tests. This is done in the following way

1. Compare each test to all training patches.
2. The vector T is formed by ranking the tests according to how far away they are from a mean of 0.5.
3. Do a greedy search:
   (a) Move the first test from T and into the result vector R.
   (b) Compare the subsequent test from T to each test in R. Add it to R if its absolute correlation is below a certain level otherwise reject it.
   (c) Repeat the former step until there are 256 tests in R. If there are less than 256, increase the threshold and try again.

This results in rBRIEF - rotated BRIEF.

Algorithm 2: BRIEF algorithm

Input: Smoothed Image Patch

Output: Feature descriptor calculation and matching

In the smoothed image patch, establishes a test τ on a patch p of size $S \times S$ in the way that is described in equation 3.1, where p(x) is the pixel intensity in a smoothed version of p at x = $(u, v)^T$;

A set of n_d(x,y) location pairs exclusively defines a set of binary tests;

The BRIEF descriptor 3.2 is set as the n_d-dimensional bitstring;

$$\tau(p; x, y) := \begin{cases} 1 & \text{if p(x)} < \text{p(y)} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$f_{n_d}(p) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(p : x_i, y_i) \tag{3.2}$$

The combination of oFAST and rBRIEF is what results in an ORB feature.

3.5 Fog Model

In computer vision fog has been usually modeled using an atmospheric scattering model proposed by Koschmieder [17, 24, 52, 53]. This model is given by

$$I(z) = I_0(z)t(z) + A(1 - t(z)) \tag{3.3}$$

where $t(z)$, the parameters that expresses the transmission parameter - the amount of light that travels straight to the observer without being dispersed by the medium, is given by

$$t(z) = e^{-kd(x,y)} \tag{3.4}$$

$d(x, y)$ is the distance of the object at pixel (x, y) and k is the fog extinction coefficient given by

$$k = \frac{C}{R_m} \tag{3.5}$$

$I(z)$ represents the image with fog, $I_0(z)$ represents the image radiance which corresponds to the fog-free image. A is the skylight and finally R_m is the value of the visibility in meters. In the literature $C = 3.912$ is a common choice.

3.6 Defogging

Fog removal is ideally expressed as

$$I_0(z) = \frac{I(z) - A}{t(z)} + A \tag{3.6}$$

which is the inverse equation of equation (3.3). However, during fog removal only I(z) is known, and other variables have to be estimated. In this work we follow the method proposed by [17] to remove fog. The method consists of estimating a DCP, a one channel image corresponding to the darkest channel of the RGB image in a moving window; estimation of $t(z)$, the transmission parameter which is associated to the transmission map or fog thickness; improvement of $t(z)$ by image matting; and then eventually estimating the fog-free image. Hence the first estimate of $t(z)$ is given by

$$t(z) = 1 - \min_{c \in RGB} \left(\frac{I(z)}{A} \right) \tag{3.7}$$

which is improved using

$$t = \hat{t}(L + \lambda U) \tag{3.8}$$

with L being the Laplacian matrix used for matting, λ is the regularization parameter - a small value set as 0.0001 - and U is an identity matrix the same size of L. L is specified as

$$L(i,j) = \sum_{k|(i,j) \in w_k} \left(\delta_{ij} - \frac{1}{|w_k|} \left(1 + (I_i - \mu_k)^T \left(\sum_k + \frac{\epsilon}{|w_k|} U \right)^{-1} (I_j - \mu_k) \right) \right) \tag{3.9}$$

where δ_{ij} is the Kronecker delta, w_k is a window with $|w_k|$ as the number of pixels within, μ_k is a 3×1 mean vector, I_i and I_j are the colours in the window, ϵ is the regularization parameter assumed to be 10^{-6} and finally $\sum_k$ is a 3×3 covariance matrix.

Since $t(z)$ may have close to zero values for some pixels, the final image is subjected to noise. Therefore, equation 3.6 is modified to

$$I_0(z) = \frac{I(z) - A}{max\ \{\hat{t},\ t_0\}} + A \tag{3.10}$$

giving the estimated fog-free image.

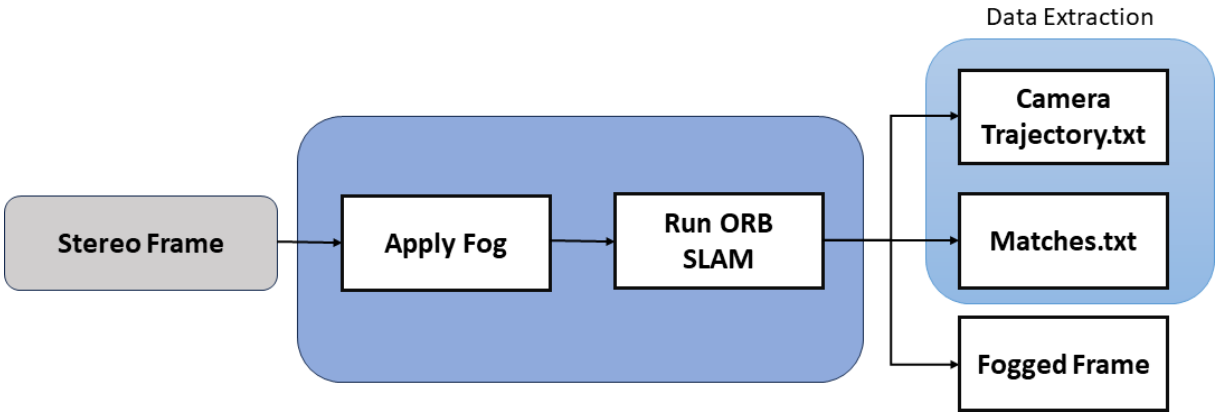
4 Proposed Approach

The methodology in this work consists of three main procedures. The first is generation of artificial fog, which is calculated using a well known fog model, and added to images of a fog-free dataset. The second procedure is removing the fog, assuming nothing about the fog generation process is known. This would also correspond to the realistic case of collecting a dataset in foggy weather, and then removing the fog from the collected images. The third procedure is application of ORB-SLAM to detect ORB features in the images and eventually run SLAM to create a path of the vehicle that collected the dataset. This final part is run on the original fog free images, fogged images, and defogged images in order to compare the performance.

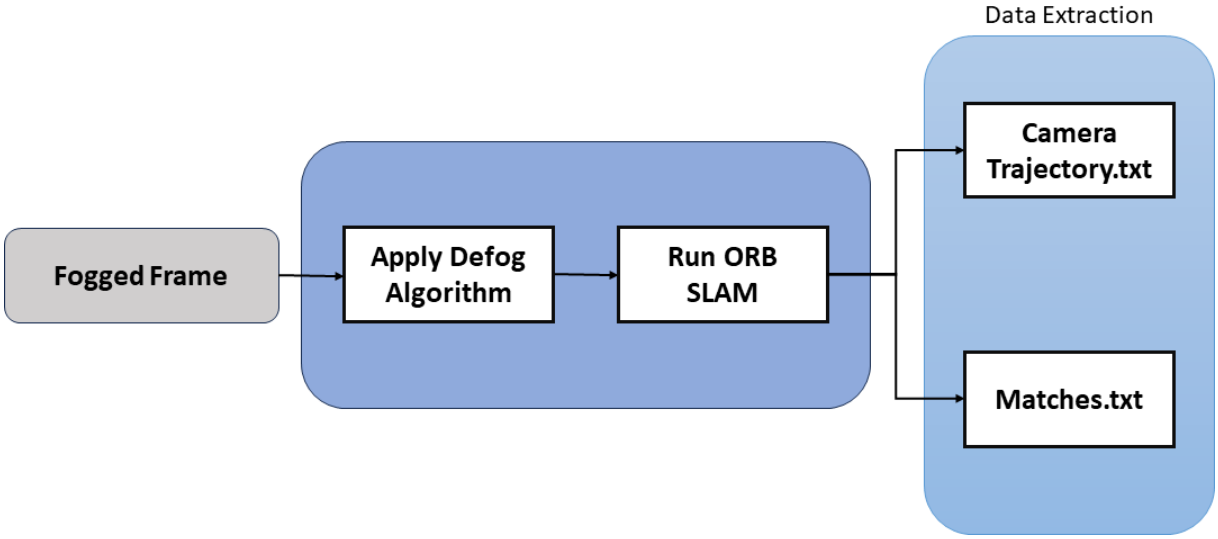
4.1 System Overview

The used approach connects all the previously mentioned methods of chapter 3. Figure 4.1 shows the two main tasks that are performed on a dataset. This way the stages of all the process can be distinguished as follows

- Apply Fog Algorithm
- Run ORB-SLAM
- Data extraction
- Apply Defog Algorithm



(a)



(b)

Figure 4.1: Block diagrams of the used approach. (a) Process with fogged dataset. (b) Process with defogged dataset.

4.2 Apply Fog Algorithm

Section 3.5 described a commonly used fog model. However, this model is not directly usable to simulate fog on a real dataset because it requires depth ($d(z)$) of the image features. In order to solve this problem, in this thesis we propose estimating the depth from the stereo images, and using this depth to simulate fog on the images of the dataset.

For $d(z)$, in this work the stereo disparity is used. The Red, Green and Blue (RGB) stereo image pair is first converted to gray scale, and then using known camera parameters, the disparity map is calculated. The missing pixels of the disparity map are filled in using cubic polynomial interpolation. Eventually $d(z)$ is found as

$$d = \frac{b \times f}{M} \tag{4.1}$$

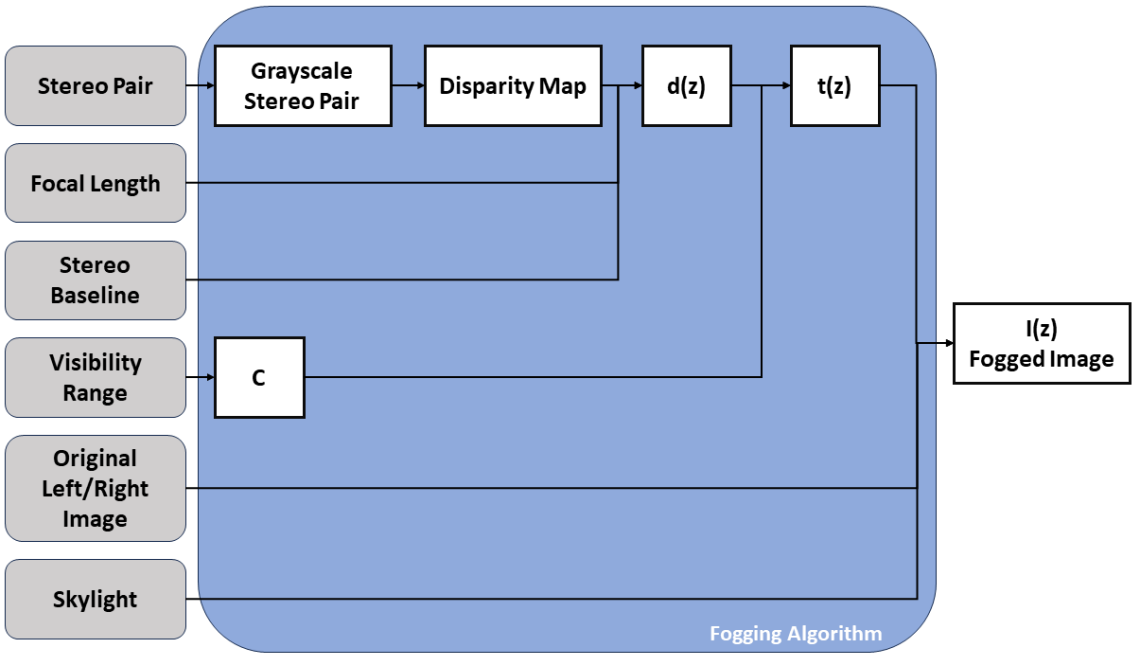


Figure 4.2: Fogging Algorithm

where b is the stereo baseline, f is the focal length, and M is the smoothed disparity map. Finally, in order to generate the image with fog, equation 3.3 is applied to each channel of the image (R, G and B).

The fog adding process was implemented as a MATLAB script. There, the images from their respective folders (from the left and right cameras) are loaded and converted to grayscale. The visibility range, R_m is chosen here also, since the density of fog depends on it. The values for baseline and the focal length of the cameras used to collect the dataset also have to be defined here. Afterwards, all these parameters are given to a function that applies the actual fog to the images. In this function, a disparity map is calculated for each pair of images. A disparity map gives the apparent motion between two images and does so by doing a correspondence between each pixel in the left and in the right image and then calculating the distance between them. The final result is a grayscale map that represents the distance between the pixels with different colours with colder colours being lower distances and warmer colours being bigger distances.

Once the disparity map is obtained, it is smoothed using a filter. It is possible now to use equation 4.1 and k can also be calculated. Once both of these calculations are done, $t(z)$ can be obtained. Now it is possible to apply equation 3.3 to the pair of images that were provided in the beginning. Figure 4.2 illustrates this process.

4.3 Run ORB-SLAM

4.3.1 ORB-SLAM2

ORB-SLAM2 was used in a MATLAB distribution[1] due to the lack of good and usable results from the open source C implementation. On MATLAB, some changes were required such as allowing customizable thresholds of detected features needed to consider the current frame a keyframe. Variables were added to allow tracking the estimated pose of the cameras directly in MATLAB, rather than in the original viewer (Pointcloud viewer), which sometimes would not properly deliver the path. More code was also added to calculate the displacement errors between successive frames in each axis. However, a major change to the code was adding the resilience to the code, to allow it start a new track when localization is lost. The original code used to crash consistently when localization failed.

4.3.2 ORB-SLAM3

In order to run ORB-SLAM3 on the datasets, some packages - Pangolin, OpenCV, Eigen3 - were required. After installing the dependencies, running the ORB-SLAM3 [2] package however was not straightforward. It required some changes to the source code, so that we could compile and run it with ROS support. Since the dataset is a stereo one, only this node was used.

Afterwards, some further changes were made to the source code in order to obtain more results and to allow the possibility of analyzing the results with different values of visibility range and consequently comparing results with ORB-SLAM2. The first change that was made was the creation of a function that would deliver the values of the transformation matrix $P = [R|T]$ that allows to plot the trajectory. Another change was also the creation of a function to extract the value of various variables that relate to the number of matches, features and matched features in each pair of stereo frame. The command line of the software package was also changed to allow a more user friendly usage across different datasets and system parameters, such as allowing the user to choose the operating mode it wants to run the dataset with like localization or SLAM.

[1] https://www.mathworks.com/help/vision/ug/stereo-visual-simultaneous-localization-mapping.html
[2] https://github.com/UZ-SLAMLab/ORB_SLAM3

4.4 Data Extraction

As mentioned in the previous section, there was a need to extract relevant information in order to analyze the performance of the system. This was done by creating two files, CameraTrajectory.txt and Matches.txt. Both of these files are extracted in both localization and SLAM mode, and for datasets with and without fog.

The first file stores the information of all twelve values that constitute the P matrix that contains the rotation (R) and translation (T) of the camera pose in global coordinates. This information allows plotting the trajectory made in such conditions and consequently evaluate the differences and effects of fog in the path, as it is aimed to be done in this work.

Then, the file Matches.txt has the content of the variables that represent the matches detected in the frames in both modes, localization and SLAM, and all of the tracked points in the frame. All of the analyzes and conclusions retrieved from this information is detailed in chapter 5.

4.5 Apply Defog Algorithm

The task of defogging is done in a similar way as the fogging algorithm. It also receives as input the stereo pair but in this case, the images contain fog. This way, and following equation 3.6, both $t(z)$ and A are needed. A is calculated the same as in the fogging algorithm. However in order to apply equation 3.7, the darkest channel needs to be found and that is done by finding the minimal value of each color channel in the image within a certain moving window. In the source code, this moving window is chosen by the user and this result is subsequently used in equation 3.6 As explained in section 3.6, this estimation is improved by applying the Laplacian, expressed with 3.9. Finally, the fog free image can be attained with equation 3.10. Figure 4.3 illustrates this process.

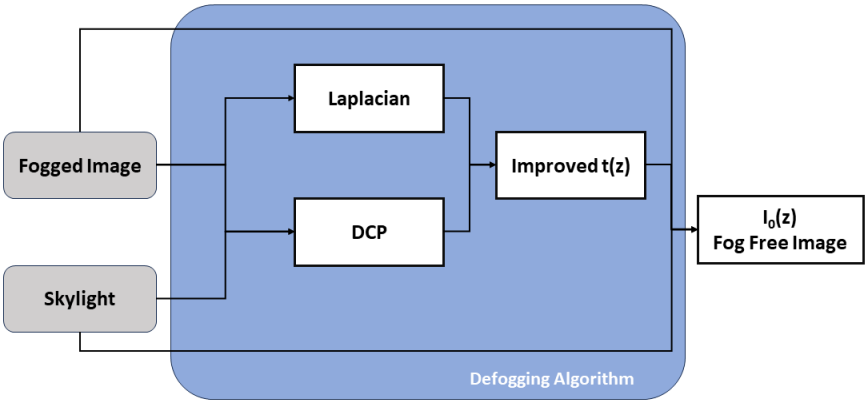


Figure 4.3: Defogging Algorithm

5 Experimental Work

In this chapter, the experimental work performed in this dissertation is presented as well as the setup and datasets used. An analysis of the results is also presented and several parameters are compared in order to portray an accurate analysis of the performance of both ORB-SLAM2 and ORB-SLAM3 systems.

5.1 Experimental Setup

5.1.1 Husky Robot Platform

In this thesis a skid-steered robot, Clearpath Husky UGV [1], was used (Figure 5.1). The robot is equipped with an RTK-GNSS receiver in a dual antenna configuration, a WiFi antenna, an IMU, a 3D LiDAR, two cameras, a battery and a Septentrio GNSS system. Figure 5.2 shows the communication infrastructure of the listed sensors. The 3D LiDAR sensor is a Velodyne VLP-16 and it has 16 vertical beams, 100 m range and 360° horizontal FoV. The two cameras are Point Grey FL-3-GE-13S2C-CS and they have 1.3 MP resolution, a global shutter and they provide color images at 120 fps. The cameras were used to collect several datasets on the Polo II campus of the university for use in the Visual SLAM. The IMU is an Xsens MTi 300. It also has a very low bias stability (10°/h) and it provides acceleration and orientation data that was planned to be used to complement visual data for SLAM. RTK-GNSS in general allows more accurate localization than an ordinary GNSS system [54], and the system in use on Husky excels on this allowing sub-centimeter localization performance.

5.1.2 KITTI Platform

The platform used in the KITTI dataset to retrieve data was a Volkswagen car that carried four video cameras, two Point Grey Flea2 grayscale cameras with global shutter and two

[1] See `https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/`

Figure 5.1: Customized Husky robot used in the implementation.

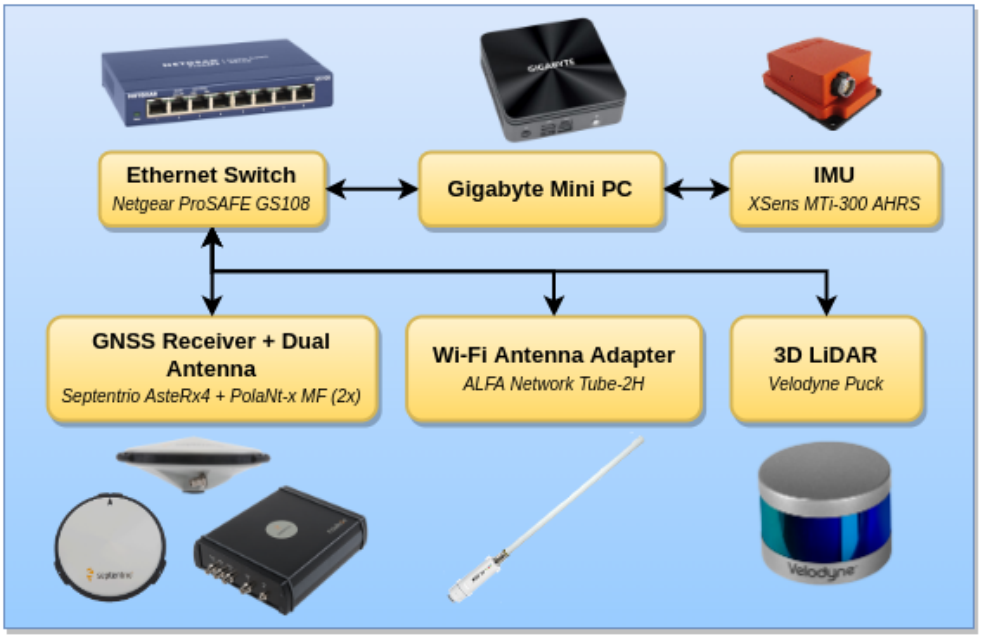


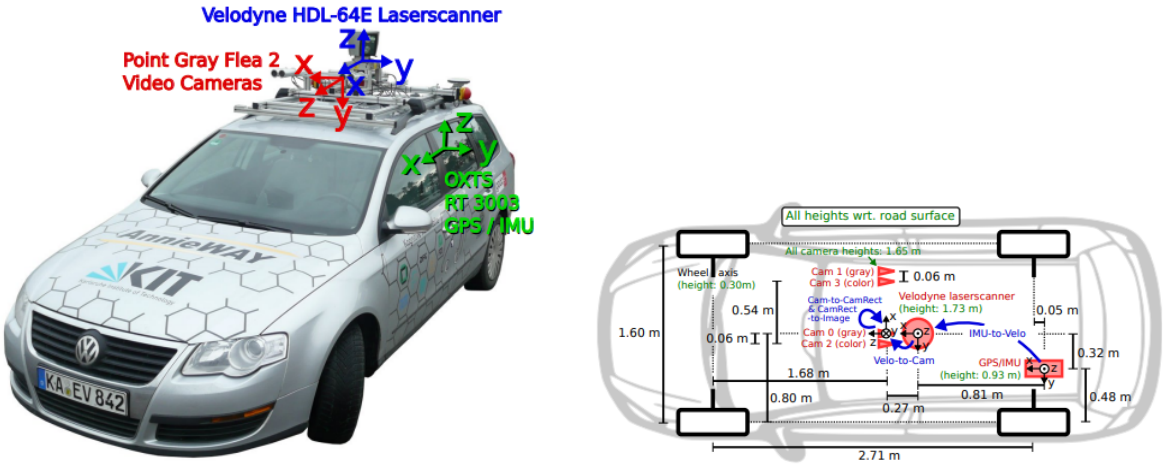
Figure 5.2: Schematic of the hardware communication infrastructure.

Point Grey Flea2 color cameras also with global shutter. In addition, it had four Edmund Optics lenses with an opening angle of 90° and a vertical opening angle of the region of interest of 35°. Furthermore it contained a Velodyne HDL-64E rotating 3D laser scanner with 64 beams, 120 m of range and a 360° horizontal and 26.8° vertical FoV. Finally, it also had an OXTS RT3003 inertial and GPS navigation system with RTK, 6 axis, a frequency of 100 Hz, and a resolution 0.02 m for each 0.1°

5.1.3 Datasets Used

The KITTI dataset [4] is a prominent benchmark used in several computer vision applications, specially in tasks related to autonomous driving and robotics. This dataset is wildly used because it contains a large range of data captured from a moving car supplied with a variety of sensors such as a velodyne LiDAR, cameras and GNSS/IMU sensors. In this work a sequence covering 440m with 570 stereo frames was used. The corresponding image resolution is 1392×512, and the dataset is dated 26/09/2011. The setup used to create this dataset is described in Figure 5.3b.

ORB-SLAM was run on the original images, which had no fog, on images with artificial fog, and then on images from which artificial fog was removed. Artificial fog was introduced using the procedure described in section 4.2. Then it was removed following the procedure described in section 4.5. Different datasets were generated, varying the maximum visibility of the foggy environment from 15 m to 100 m. Then ORB-SLAM was run recording the features, their locations and the whole path along, eventually comparing the results to the ground truth provided as GNSS coordinates data in the dataset. Since the current focus was on feature detection and tracking performance, IMU readings of the dataset were not used in the process.



(a) (b)

Figure 5.3: (a) Vehicle used as a recording platform to retrieve KITTI dataset. (b) Sensor setup on recording platform. Taken from [4].

5.2 Software Packages

The first experiments of this thesis work were done using ORB-SLAM2, utilizing mainly the open source C implementation [2]. Due to the lack of good and usable results, and difficulties in changing the code, later a MATLAB implementation of ORB-SLAM2 [2] was used.

Later, the open source implementation of ORB-SLAM3[3] provided by the proposers of ORB-SLAM3 [3] was used. The ORB-SLAM3 package was installed on an Ubuntu 20.04 operating system. Preliminary experiments were run using the ROS version of the ORB-SLAM, with the aim of running all the system on Husky, the skid-steered robot of the Field Robotics Laboratory. Then for further tests using public datasets, the non-ROS version was used. This package makes use of other support packages such as DBoW2 [4], Sophus [5], g2o [6], Pangolin[7] and OpenCV[8].

DBoW2 is a renewed implementation of the first, DBoW. It is an open source library which uses C++ language to index and consequently convert images into a bag-of-words representation. It works by applying a hierarchical tree to the nearest neighbors in the frame

[2] https://www.mathworks.com/help/vision/ug/stereo-visual-simultaneous-localization-mapping.html

[3] https://github.com/UZ-SLAMLab/ORB_SLAM3

[4] https://github.com/dorian3d/DBoW2

[5] https://github.com/strasdat/Sophus

[6] https://github.com/RainerKuemmerle/g2o

[7] https://github.com/stevenlovegrove/Pangolin

[8] https://opencv.org/

and then generating a visual vocabulary associated to it. Additionally, it employs an image database containing inverted and direct files to index images which allows quick searches of the database. Since this is an improved version of the original DBoW library, there is some differences between them. Firstly, DBoW2 allows for any type of descriptor to be used. In relation to ORB-SLAM specifically, it is very useful since it can work automatically with ORB and BRIEF descriptors. One of the main differences is that the original library uses binary format and this one uses an OpenCV storage. This permits the use of vocabularies to be stored in YAML format, and therefore allows for less compatibility issues.

General Graph Optimization (g2o) [55] is also a C++ open source framework that is used for optimizing graph-based nonlinear error functions and specially solving problems derived from SLAM and BA.

Pangolin is also used by ORB-SLAM3 for visualization and user interface. Its main advantages are the compatibility with several Operating Systems, substantial video wrappers for several cameras and media formats, which allows easy usage.

OpenCV, another major package used by ORB-SLAM, is an open source computer vision library that provides easy access to computer vision tools to its users. It includes traditional algorithms and tools as well as state-of-the-art ones. Since it also allows easy changes to code, all of this presents an advantage over other computer vision libraries.

Finally, Sophus is also implemented in C++ and its main goal is to solve 2D and 3D geometric problems, which in the context of this work helps to solve the rotations and translations and therefore represent the necessary rigid body transformations.

5.3 Tests and Results

Fig. 5.4 shows several examples of the process of adding and removing artificial fog, with Fig. 5.4a showing the original image, Fig. 5.4b showing fog with maximum visibility (R_m) of 70 m, Fig. 5.4c showing fog with R_m of 25 m, and Fig. 5.4d showing fog removed from Fig. 5.4c. Added fog can be seen to be realistic. However, defogging, although is able to clean fog in close range particularly on the left and right sides of the image, which are occupied by relatively darker objects, fails to remove fog properly around the central region of the image, loosing possibility of detecting features along that part.

(a) (b)

(c) (d)

Figure 5.4: (a) Original image from the KITTI dataset with no fog. (b) Image (a) with fog of max visibility of 70 m. (c) Image (a) with fog of max visibility of 25 m. (d) Image (c) after fog removal.

5.3.1 ORB-SLAM2

Fig. 5.5 presents four of the histograms of the distance of the features to the camera, and Table 5.1 presents a summary of the feature counts. With clear images (Fig. 5.5a) ORB-SLAM2 can be seen to be detecting the most features along the whole trajectory, with more than 106 k in total, mostly focusing on close range. However, it is able to detect two thousand features at even 90 m. Adding fog with R_m of 25 m reduces the detected features along the whole trajectory down to just 2.1 k, mostly being from points closer than 20 m (Fig. 5.5b). Removing the fog on the other hand, increases the number of detected features by almost 15− fold, and the detected features can be seen to extend to distances of up to 100 m. Introducing fog with R_m of 50 m can be seen to reduce the number of detected features still significantly, however down to 17.4 k only (Fig. 5.5c). Defogging the dataset increases the feature count by almost 2 fold, up to 30.5 k, lifting the histogram up and also increasing the number of detected features in the long range. Setting R_m to 75 m, in contrast to 50 m, reduces the features less, only down to 31.3 k. However, removing fog from these images, does not improve the number of features, but reduces them particularly at close range, which can be seen as a side effect of processing the images. In this case the histogram looses some of the closer features, but gains slightly at the far range (Fig. 5.5d).

The effect of changing the number of detected features on the interpolated overall localiza-

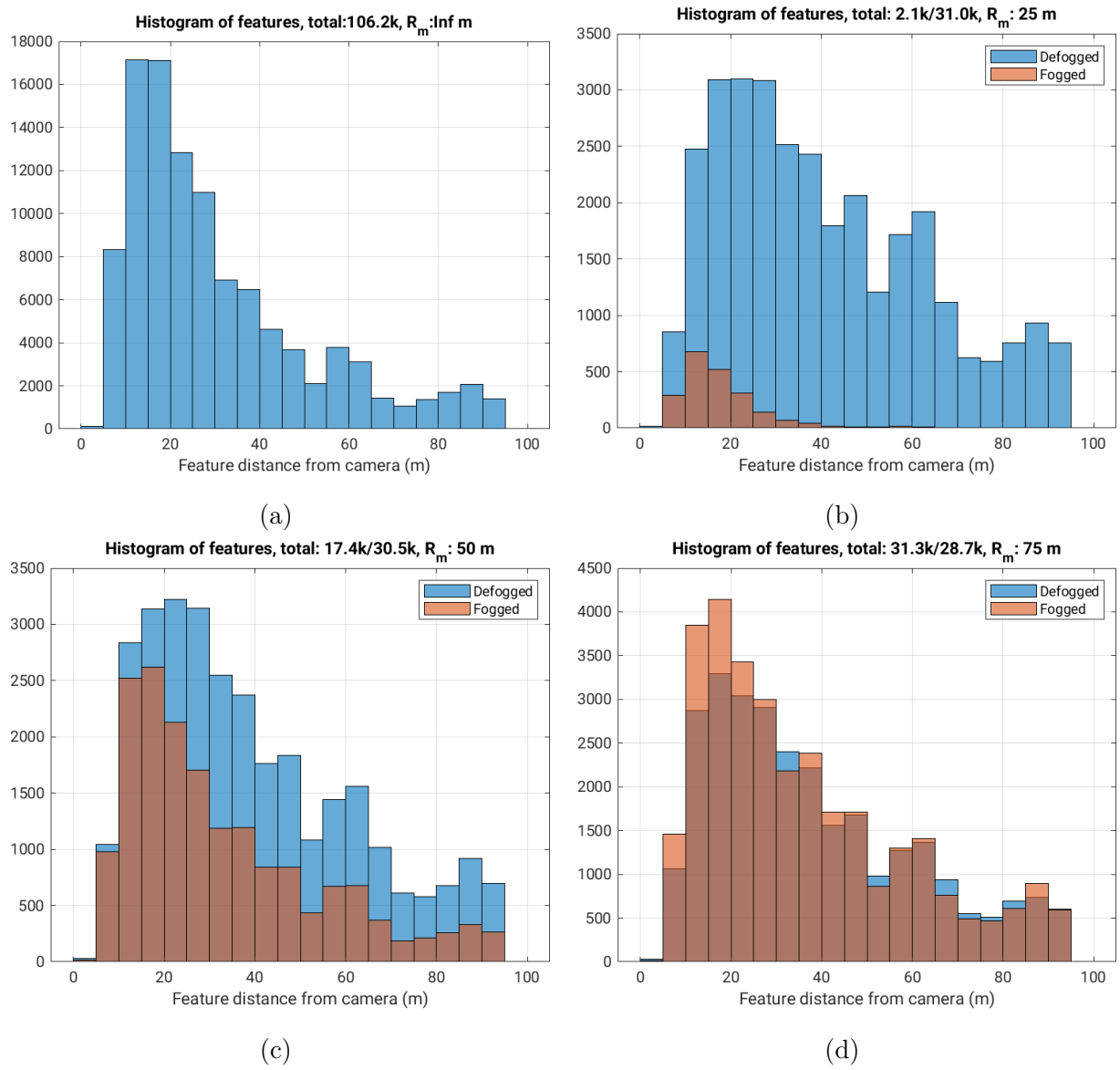


Figure 5.5: Histograms of the distance of the features to the camera for scenarios with different maximum visibility, (a) Infinite, (b) 25 m, (c) 50 m, (d) 75 m.

tion performance can be seen in Fig. 5.6 and Table 5.1, where the errors in localization along the horizontal plane and the vertical axis are presented. ORB-SLAM with original images has an error less than 1 m, oscillating mainly around 0.2 m, with Root Mean Square (RMS) value 0f 0.2 m. Adding fog to the images increases the errors considerably, reaching RMS values of 1.53 m and 1.09 m for R_m of 50 m and 75 m respectively. Defogging the images reduces the error down to mostly less than 1 m, particularly for the R_m of 75 m case (Fig. 5.6c Fig. 5.6e), and the RMS values down to 0.82 m and 0.85 m. However, intermittent large errors are still observed.

Table 5.1 summarizes the number of detected features, ratio of valid displacement estimates which happen only when ORB-SLAM returns successfully a motion estimate, as well

Table 5.1: Performance summary - Low Resolution Images (ORB-SLAM2)

R_m (m)	Number of Features		Valid Estimates (%)		RMSE (m) horizontal		RMSE (m) vertical	
	Fog	Defog	Fog	Defog	Fog	Defog	Fog	Defog
15	86	10.0k	-	47	-	-	-	-
20	678	24.0k	3	80	-	3.00	-	2.15
25	2.1k	31.0k	12	96	-	1.59	-	1.73
30	4.4k	32.4k	22	95	-	4.31	-	3.74
35	7.4k	31.9k	38	98	-	1.32	-	1.13
40	10.9k	31.2k	54	100	-	1.00	-	0.70
45	14.2k	30.8k	68	100	1.40	1.02	1.66	0.56
50	17.4k	30.5k	67	100	1.53	0.82	1.00	0.61
55	20.6k	29.9k	77	100	1.85	0.92	0.98	0.23
60	23.7k	29.6k	87	100	1.44	0.82	1.18	0.09
65	26.4k	29.2k	89	100	1.76	0.78	1.97	0.27
70	29.0k	28.8k	98	100	1.90	0.83	3.36	0.31
75	31.3k	28.7k	96	98	1.09	0.85	2.65	0.26
80	33.1k	28.3k	99	99	3.36	0.81	1.61	0.15
85	34.9k	28.2k	100	99	1.47	0.83	1.47	0.30
90	36.2k	28.1k	100	100	3.44	0.82	1.86	0.29
95	37.2k	27.8k	100	99	1.58	0.71	0.80	0.13
100	38.2k	27.7k	100	100	0.82	0.84	0.80	0.26
∞	106.2k	-	100	-	0.19	-	0.05	-

as the corresponding RMS errors. RMS errors are reported only when the ratio of valid estimates is more than 60 %. Adding fog, even if just a little bit with R_m of 100 m, reduces the number of features to almost one third. This table represents the values when the frames provided have only 40 % of their original full resolution. Adding more fog reduces the number of features even further. As the number of features over all the trajectory becomes less than 14 k, ORB-SLAM fails to estimate motion along half of the path. Defogging helps improve the detected features considerably when moderate to significant fog is added. With a fog of R_m of 20 m, the valid estimates increase from 3 % to 80 %. However, improvements to the number of features fade after R_m reaches 70 m, despite its better localization performance

Table 5.2: Performance summary - Full Resolution Images (ORB-SLAM2)

R_m (m)	Number of Features		Valid Estimates (%)		RMSE (m) horizontal		RMSE (m) vertical	
	Fog	Defog	Fog	Defog	Fog	Defog	Fog	Defog
15	-	-	-	-	-	-	-	-
20	-	26.4k	-	91	-	1.18	-	0.94
25	-	25.1k	-	95	-	2.94	-	2.41
30	-	21.9k	-	93	-	1.59	-	1.07
35	-	19.7k	-	93	-	1.08	-	1.34
40	38.9k	18.2k	83	94	9.05	1.46	2.21	1.30
45	43.2k	17.2k	90	92	1.14	1.31	1.44	0.72
50	45.3k	16.5k	96	97	1.13	1.28	1.39	1.12
55	45.9k	15.7k	96	88	1.30	1.30	1.32	0.48
60	45.4k	15.5k	96	97	2.98	0.87	1.75	0.82
65	44.3k	15.0k	97	88	0.99	1.22	1.38	0.53
70	43.1k	14.7k	99	95	1.11	1.19	1.33	1.01
75	41.5k	14.5k	100	94	1.12	0.84	1.20	0.38
80	40.2k	14.2k	98	95	1.07	0.81	1.30	0.31
85	39.2k	14.3k	100	93	1.21	0.90	1.14	0.36
90	38.2k	14.0k	98	96	1.45	0.84	1.01	0.30
95	37.6k	14.0k	100	100	1.10	0.82	0.99	0.52
100	36.8k	13.7k	100	98	1.09	0.79	0.72	0.22
∞	67.1k	-	100	-	0.19	-	0.04	-

in terms of the RMS error. RMS error along both the horizontal plane and the vertical axis can also be seen to improve with defogging.

Table 5.2 presents the same information as Table 5.1 but now with the images in full resolution. Although now there are more values of R_m for which ORB-SLAM2 fails to produce sufficient features, and hence fails localization, R_m for which features are detected have many more features and a higher percentage of valid estimates. On the contrary, the values for the Root Mean Square Error (RMSE) are in some cases worse, for example for R_m of 40 m and 60 m reaching 9 m and 3 m.

Fig. 5.7 shows three of the resultant paths and the ground truth for ORB-SLAM2. The

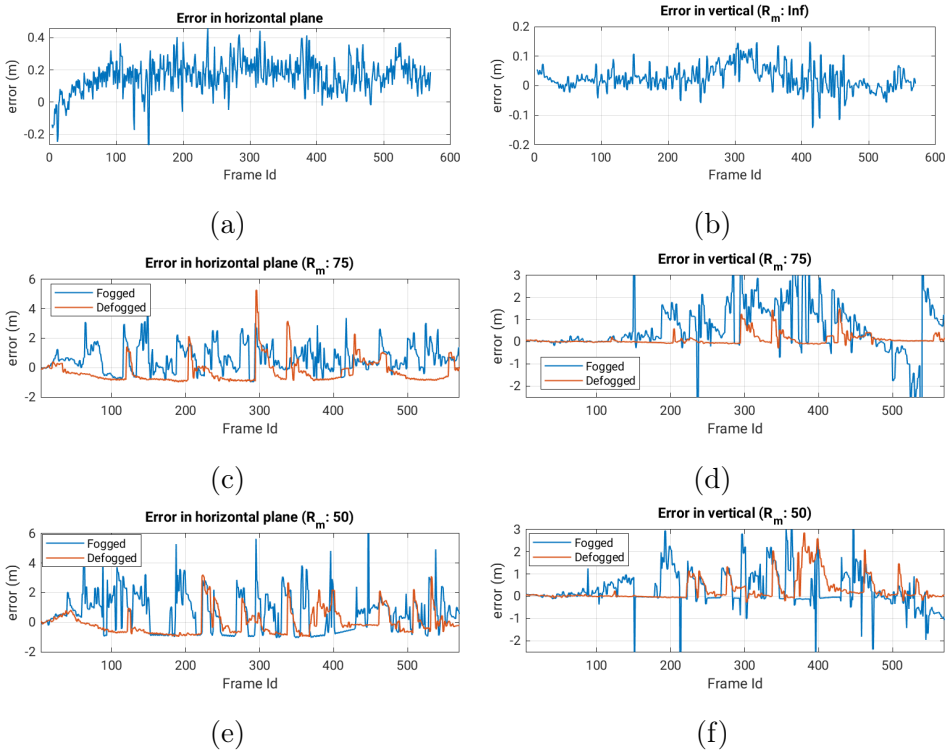


Figure 5.6: Error in displacement estimates along the horizontal plane and the vertical axis between successive frames. (a, b) using images with no fog (c,d) using images with artificial fog of maximum visibility of 75 m (e, f) and maximum visibility of 50 m

.

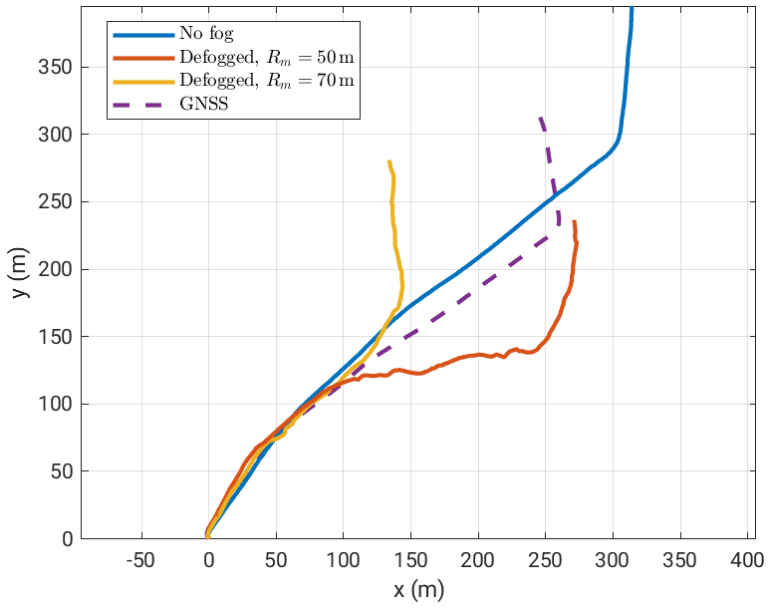


Figure 5.7: Vehicle pose estimated using fog-free images and defogged images with maximum visibility of 50 m and 70 m.

path due to the original images follows the ground truth roughly, though it leads, caused mainly by the drift due to the RMS error of 0.2 m. This drift is partially caused by the

Table 5.3: Performance summary - Localization Mode - Full Resolution (ORB-sLAM3)

R_m (m)	Number of Features		Valid Estimates (%)		RMSE (m) horizontal		RMSE (m) vertical	
	Fog	Defog	Fog	Defog	Fog	Defog	Fog	Defog
15	-	22.5k	-	97	0.80	0.52	0.06	0.73
20	14.9k	30k	95	97	3.10	0.53	3.12	0.52
25	30k	31.6k	97	97	0.49	0.52	0.77	0.60
30	36.3k	31.8k	97	98	0.59	0.43	0.72	0.82
35	39.3k	30.3k	97	97	0.51	0.47	0.57	0.71
40	42.8k	30k	97	97	0.47	0.43	0.67	0.77
45	44.9k	29.7k	97	97	0.48	0.43	0.69	0.85
50	45.8k	28.1k	97	97	0.52	0.50	0.65	0.73
55	46.2k	27.5k	97	97	0.44	0.53	0.70	0.73
60	47k	26.5k	97	97	0.46	0.77	0.65	1.03
65	46k	27.1k	97	97	0.42	0.49	0.70	0.72
70	45.7k	27.4k	97	97	0.40	0.52	0.75	0.67
75	45k	26.6k	97	97	0.37	0.48	0.82	0.71
80	46.2k	26.5k	97	98	0.38	0.51	0.79	0.72
85	45.2k	25.8k	97	97	0.39	0.53	0.78	0.68
90	43.8k	26k	97	97	0.41	0.55	0.72	0.67
95	44.3k	26k	97	97	0.39	0.50	0.80	0.73
100	43.8k	25.5k	98	97	0.35	0.52	0.83	0.65
∞	141k	-	100	-	0.16	-	0.34	-

limited resolution of the images. The images with R_m of 70 m result in a path that keeps the form, but with a much larger error. Finally, the path of R_m of 50 m fails in form.

In these experiments, defogging can be seen to help improve the performance of ORB-SLAM when estimating motion locally. However, the improvements are limited when the images are saturated, which happens more often when the objects are further away and have a lighter color. This is expected to be a limiting factor on the usability of image defogging for SLAM.

Table 5.4: Performance summary - Localization Mode - Low Resolution (ORB-SLAM3)

R_m (m)	Number of Features		Valid Estimates (%)		RMSE (m) horizontal		RMSE (m) vertical	
	Fog	Defog	Fog	Defog	Fog	Defog	Fog	Defog
15	-	6.2k	-	94	-	4.01	-	3.65
20	-	13.k4	-	95	-	3.65	-	2.43
25	-	14.1k	-	95	-	3.51	-	2.19
30	-	14.6k	-	96	-	3.52	-	2.96
35	-	17.3k	-	96	0.79	2.83	0.06	2.75
40	17.9k	16.7k	94	95	3.89	3.28	1.95	3.57
45	18.2k	16.5k	94	95	3.85	3.06	2.17	2.34
50	18.5k	11.7k	94	95	3.77	2.30	2.31	3.86
55	18.8k	13.4k	94	94	3.86	3.24	1.87	2.72
60	16.2k	10.8k	95	93	3.82	2.61	1.31	2.83
65	17.4k	15.9k	95	96	3.52	2.98	1.98	2.63
70	5.4k	15.6k	95	95	2.83	2.76	3.39	2.78
75	14.5k	16.3k	92	96	2.47	2.74	3.29	2.72
80	6.3k	5.4k	72	96	3.94	4.18	2.93	4.17
85	16.9k	5k	95	88	3.47	4.16	2.29	4.64
90	10.3k	18.8k	95	96	3.38	3.30	3.72	2.46
95	13.8k	5.6k	95	95	3.90	2.95	3.37	2.25
100	6.8k	14.7k	91	94	4.81	4.61	3.83	4.46
∞		-		-		-		-

5.3.2 ORB-SLAM3

Since ORB-SLAM3, also provides the ability to use localization mode, it was found relevant to compare the usage of the system in the two modes: localization and SLAM. An overview of the performance is presented in tables 5.3, 5.4, 5.5 and 5.6. These tables provide information regarding the number of features, the percentage of valid estimates and the RMSE, in meters, along the horizontal and vertical axis, all of them for the fogged and defogged datasets.

Table 5.5: Performance summary - SLAM Mode - Full Resolution (ORB-SLAM3)

R_m (m)	Number of Features		Valid Estimates (%)		RMSE (m) horizontal		RMSE (m) vertical	
	Fog	Defog	Fog	Defog	Fog	Defog	Fog	Defog
15	5.4k	150k	6	97	0.25	0.64	1.17	1.55
20	42.1k	153.7k	39	98	0.61	0.58	1.59	1.19
25	90.7k	153.5k	72	99	0.60	0.49	1.79	1.30
30	128k	150.9k	90	99	0.68	0.40	1.40	1.04
35	149.8k	150k	95	99	0.71	0.54	1.37	1.14
40	163.1k	146.4k	99	99	0.54	0.54	1.25	1.11
45	169k	147.8k	99	99	0.50	0.54	1.17	1.12
50	171k	149.2k	99	99	0.40	0.47	1.04	0.97
55	172k	146.3k	99	99	0.39	0.65	1.05	0.87
60	174.3k	143.1k	99	99	0.44	0.47	1.10	1.05
65	174.1k	143.1k	99	99	0.51	0.61	1.14	0.98
70	173.8k	146.6k	100	99	0.39	0.70	0.92	1.12
75	174.5k	141.2k	100	99	0.40	0.61	1.00	0.94
80	176.9k	138.9k	100	99	0.39	0.60	0.83	0.99
85	176.4k	139.8k	100	99	0.41	0.59	0.96	0.99
90	176.1k	138.6k	100	99	0.43	0.62	0.79	0.68
95	175.3k	141.3k	99	99	0.41	0.67	1.00	0.76
100	175.9k	146.8k	99	99	0.45	0.59	1.04	0.68
∞	259.9k	-	100	-	0.16	-	0.34	-

Comparison with ORB-SLAM

The characteristic that stands out the most, when compared to 5.1 is the increase in the ratio of valid estimates. In ORB-SLAM2, in fogged conditions, only from $R_m = 70\,\text{m}$ on wards the valid estimates reach 98 % whereas in ORB-SLAM3 both localization and SLAM mode reach close to that value at $R_m = 20\,\text{m}$ (95 %) and $R_m = 30\,\text{m}$ (90 %), respectively.

Once again it can be observed that defogging helps track more features, as seen for example in Table 5.3, where at $R_m = 15\,\text{m}$ the number of tracked features increase from 0 to 22.5 k. In Table 5.5, also with $R_m = 15\,\text{m}$ the number of features increase 30− fold, going from 5.4 k features to 150 k. This indicates that not only ORB-SLAM3 delivers better results

Table 5.6: Performance summary - SLAM Mode - Low Resolution (ORB-SLAM3)

R_m (m)	Number of Features		Valid Estimates (%)		RMSE (m) horizontal		RMSE (m) vertical	
	Fog	Defog	Fog	Defog	Fog	Defog	Fog	Defog
15	-	29.7k	-	39	-	-	-	-
20	-	59.4k	-	76	-	3.57	-	1.22
25	-	75.9k	-	91	-	3.80	-	1.37
30	-	78.3k	-	95	-	3.80	-	1.12
35	3.7k	78.8k	6	97	-	3.84	-	1.33
40	8.1k	76k	12	98	-	3.64	-	0.77
45	13.3k	78.2k	20	98	2.67	3.53	0.84	1.14
50	21k	79k	29	98	3.04	3.31	1.14	1.06
55	26.9k	75.9k	39	98	3.42	3.36	1.24	0.78
60	36.7k	75.3k	51	98	3.74	3.35	1.41	0.75
65	41.4k	76.3k	56	98	3.70	3.00	1.36	0.78
70	48.9k	76.2k	65	99	3.87	2.99	1.38	0.84
75	57.8k	76.3k	79	98	4.02	3.15	1.33	0.86
80	62k	78.1k	84	98	4.03	3.06	1.42	0.87
85	67k	76.1k	88	98	3.96	3.00	1.25	0.98
90	69.5k	75.3k	91	99	3.86	2.42	1.18	0.89
95	73.5k	-	92	98	3.62	2.76	1.17	0.81
100	72.1k	75.3k	93	99	3.84	3.21	1.25	0.87
∞		-		-		-		-

than ORB-SLAM2 but that SLAM mode also produces better results than localization mode, regarding the number of tracked features. On the other hand these better results stop being relevant when R_m reaches 30 m in localization mode and 40 m in SLAM mode.

An RMSE value is better the closer it is to zero. This means that, again ORB-SLAM3 presents better results for RMSE values especially in the horizontal axis. This is confirmed when analyzing, for instance, the values concerning the horizontal RMSE, in defogged conditions, with $R_m = 20$ m where RMSE is 3 and the corresponding error in localization and SLAM mode is 0.53 (decrease of 82.33%) and 0.58 (decrease of 80.67%), respectively. This decrease in the RMSE value represents an improvement in the tracked features, where it

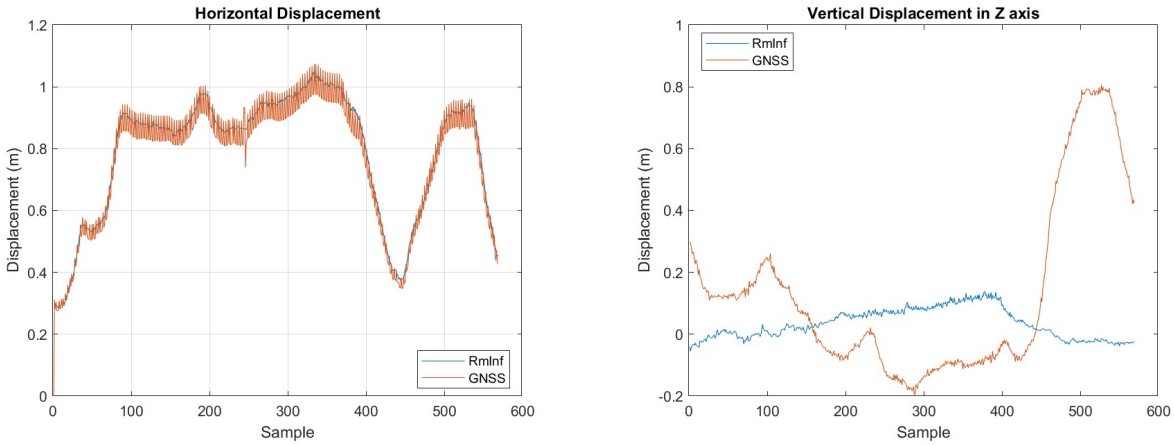
goes from 24 k in ORB-SLAM2, to 30 k and 153.7 k in ORB-SLAM3. In situations where fog is present, or in context of this thesis, added, this phenomenon is also seen when $R_m = 70$ offers an RMSE (again in the horizontal axis) value of 1.90 m in ORB-SLAM2 whilst in ORB-SLAM3, those values are 0.40 m and 0.30 m, respectively in localization and SLAM mode.

Finally, with regard to the values of RMSE in the vertical axis, a similar analysis is presented. While in localization mode the results are very good, since not a single one reaches zero; both with and without fog, in SLAM mode this is not case. In fact, it is the opposite as only 33 % of results, in foggy and not foggy conditions, are below 1 m. Although at first glance this seems that is in accordance with the results obtained with ORB-SLAM2, actually in that case only the first results, therefore with lower values of R_m, have higher values of RMSE. This permits to infer that localization mode in ORB-SLAM3 is the one that produces better results regarding the displacement in the vertical axis.

Comparison between Localization and SLAM mode

It is also possible to draw conclusions regarding the performance between the two modes of ORB-SLAM3, localization and SLAM. Firstly, regarding the number of features, localization mode tracks less features, regardless of the conditions and the improvement that defogging gives to feature tracking declines at $R_m = 30$ m whereas in SLAM mode it declines at $R_m = 40$ m. In terms of valid estimates, the values are similar in both modes, the only difference being that the localization mode never reaches 100 % of valid estimates, with the maximum value being 98 % whereas SLAM mode reaches 100 %, when R_m is 70, 75, 80, 85 and 90, all in foggy conditions. In the horizontal RMSE columns, the values are once again similar in both modes. However, in the vertical RMSE columns, as it was stated in the previous section, the values are essentially the opposite between tables with the results from SLAM mode being similar to the ones from ORB-SLAM2.

Considering that ORB-SLAM2 was used two times with different resolutions, it was only fair that each mode of ORB-SLAM3 was used in the same way. That comparison can be made by analyzing tables 5.3 and 5.4 for localization mode and 5.5 and 5.6 for SLAM mode. Regarding localization mode, with lower resolution, it only starts to detect features with $R_m = 40$ m whereas with full resolution it only does not detect within the first scenario of $R_m = 15$ m. In the matter of the valid estimates, although the lowest value is still a good one, 72 % with $R_m = 80$ m, it only has enough feature to do this calculation, again



(a) (b)

Figure 5.8: Displacement estimates along the horizontal plane and the vertical axis between successive frames. (a, b) using images with no fog

.

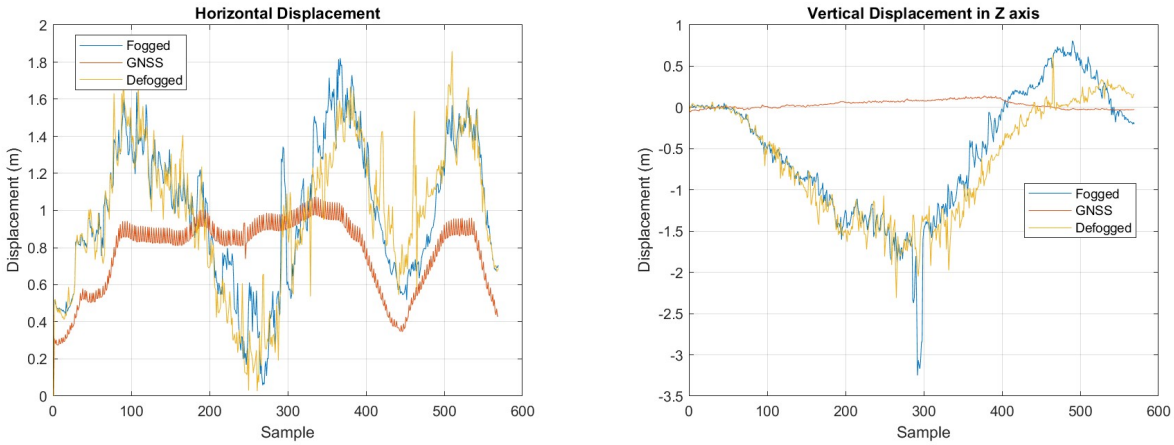
when $R_m = 40\,\text{m}$. Moreover in SLAM mode, what stands out the most is of course the difference between the number of detected features as with lower resolution the system can't even detect any in foggy conditions until R_m reaches 35 m. Although with defogged images there is detection from the lowest value of R_m, the number of features is a lot smaller. Also, regarding the percentage of valid estimates, it reaches values as low as 6 % with low resolution, whereas with full resolution the same value is reached but in the first scenario, with $R_m = 15\,\text{m}$.

Displacements Estimates Comparison

Figures 5.8, 5.9 and 5.10 represent the displacements in the horizontal axis when there is no fog present in the images, in localization mode and in SLAM mode, respectively. In the last two, the values of R_m that were used are the same as in figure 5.6 so that an easier comparison can be done.

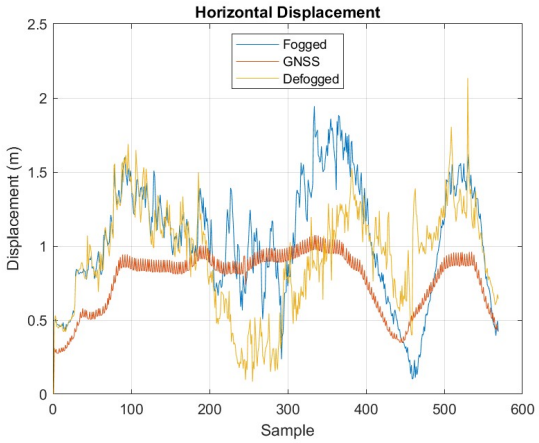
Regarding the situation with no fog, in the horizontal plane the difference is minimal, although in the vertical axis it is more noticeable, when examining the values, the difference is less than 1 m. Since the dataset was recorded without fog, it makes sense that in the horizontal plane there is no difference.

Analyzing the values for the displacements in Figure 5.9, it is clear that between fogged and defogged conditions the differences are smaller than when compared to the GNSS data, which also happens when running ORB-SLAM2 (Figure 5.6). Although there is difference

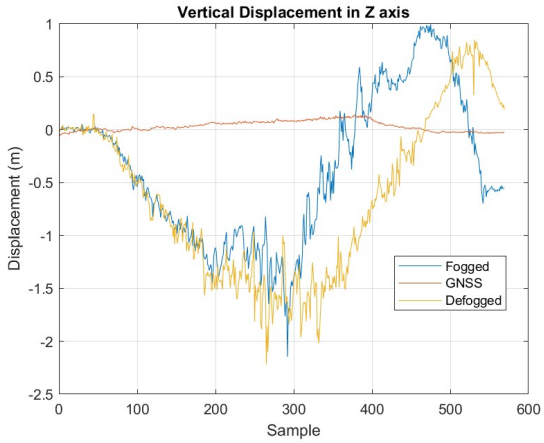


(a)

(b)



(c)



(d)

Figure 5.9: Displacement estimates, in localization mode, along the horizontal plane and the vertical axis between successive frames. (a, b) using images with artificial fog of maximum visibility of 50 m (c,d) using images with artificial fog of maximum visibility of 75 m.

between the artificial conditions implemented and the GNSS data, in the horizontal plane the trend is the same. In the vertical axis, the difference is bigger but still is never bigger than 1 m. Again, defogging the images helps in reducing the error, especially in the vertical axis. However, it is common to all situations that in the later frames the displacement is more distinct, due to the cumulative error.

In SLAM mode (Figure 5.10) the first observation that can be made is the presence of a few frames that have a very big displacement value. When running the experiments, it was observed that during the turns, ORB-SLAM would perform worse hence the presence of these values. In this mode, fogging actually has less displacement errors than defogging. The reason for this is that SLAM mode has more difficulty in keeping track of features and therefore in storing the information regarding the trajectory. Even then, in the vertical

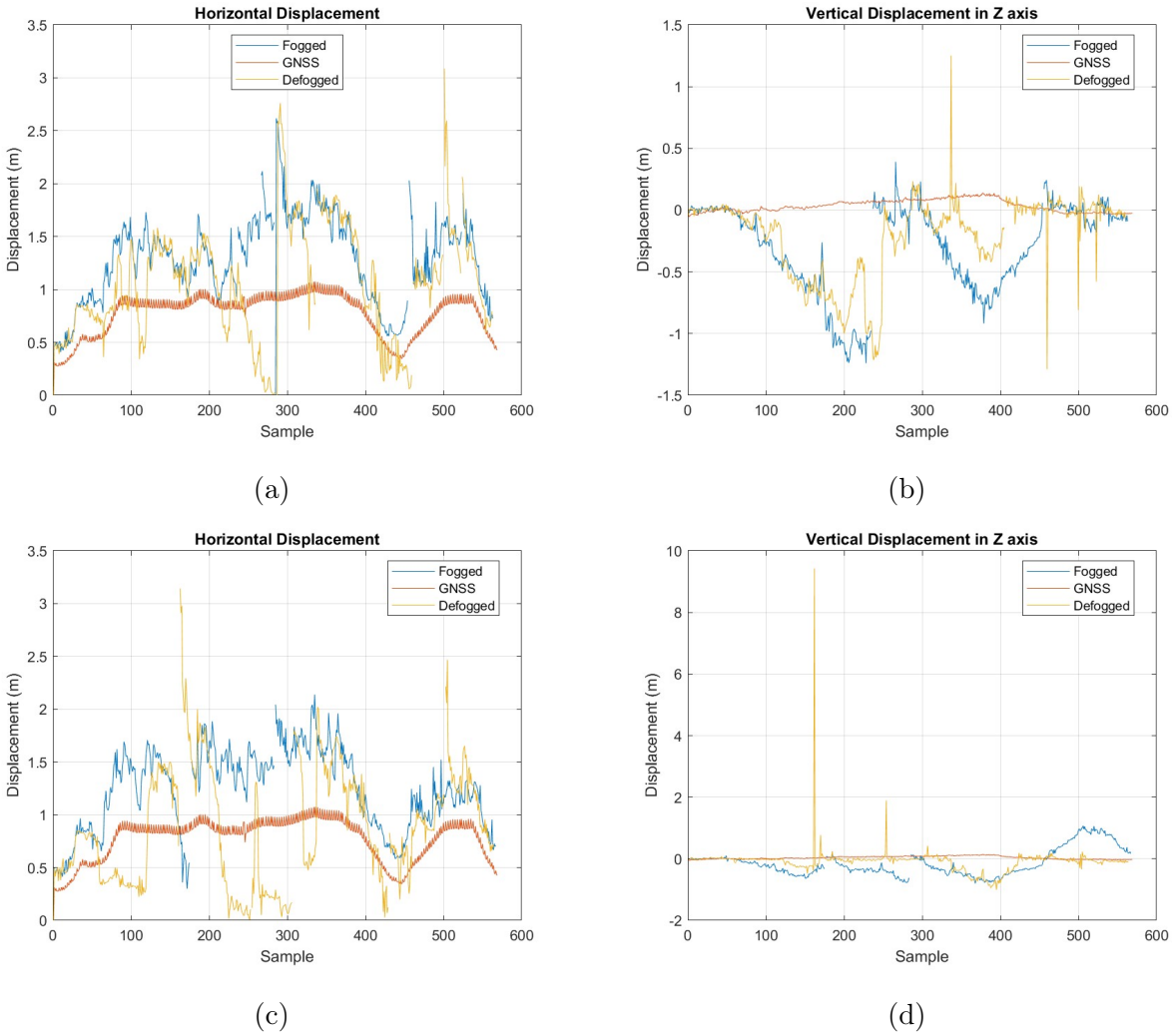


Figure 5.10: Displacement estimates,in SLAM mode, along the horizontal plane and the vertical axis between successive frames. (a, b) using images with artificial fog of maximum visibility of 50 m (c,d) using images with artificial fog of maximum visibility of 75 m.

displacement, it is clear the then bigger the value of R_m, which subsequently has less fog, the smaller the displacement errors.

Path comparison

Figure 5.11 presents the estimated vehicle poses in the two modes that ORB-SLAM3 allows, localization and SLAM. All the paths generated include the trajectories with R_m being 50 m and 70 m since these are the values that were used when studying the performance of ORB-SLAM2, which allows for a direct comparison to the performance of both systems.

The first most noticeable difference among the different graphs is that in SLAM mode, the trajectories are not continuous, resulting in several "branches". This happens because the design of the system requires that when the tracking is lost, after a certain amount of

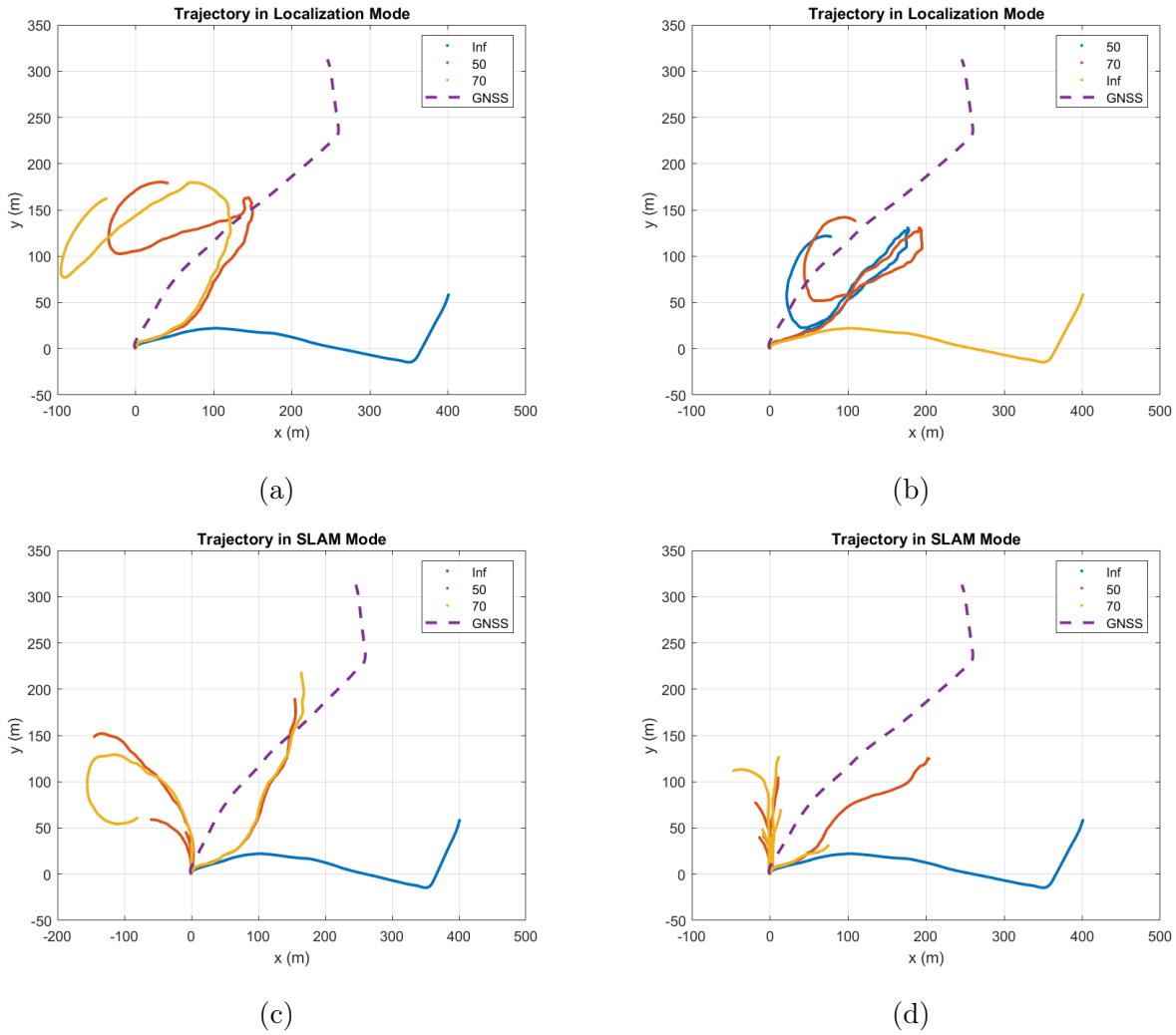


Figure 5.11: Vehicle pose estimated using fog free images, fogged and defogged images with maximum visibility of 50m and 70m. (a) and (b) in localization mode with (a) being the fogged images and (b) the defogged ones. (c) and (d) are in SLAM mode with (c) being with fogged images and (d) with the defogged ones.

frames without any features detected, the tracking returns to its origin. As explained in section 3.3, there are several maps in the Atlas of the system. This means that although the estimated pose returns to the origin, they are indeed different maps.

Upon direct comparison of Figures 5.7, 5.11b and 5.11d, the one that represents better results is indeed the one from ORB-SLAM2 (Figure 5.7).

6 Conclusion and Future Work

Integration of autonomous vehicles into our transportation systems holds the promise of addressing critical issues such as traffic congestion, accidents, and enhanced accessibility for individuals with disabilities. Cameras play a crucial role not only in SLAM but also in semantic scene understanding, allowing vehicles to recognize essential elements like traffic signs and pedestrians. VO and Visual-SLAM are reliant on detecting invariant features in images, and they offer valuable techniques for vehicle localization and environment mapping. However, environmental factors like fog and haze can severely hold up feature detection, potentially leading to adverse consequences such as road accidents. While various image enhancement methods have been developed to address this issue, their impact on SLAM performance remains an unexplored area. This dissertation sheds light on the significant influence of fog on visual SLAM and shows the potential of defogging algorithms in mitigating these challenges, marking a crucial step forward in the advancement of autonomous driving technology.

In order to study the effects of fog, ORB-SLAM was selected as the case study in this thesis. ORB-SLAM has several versions, and in this work ORB-SLAM2 and ORB-SLAM3 were used, and their performance was compared. Since ORB-SLAM3 has two modes of working, namely localization only (VO) and SLAM, these two modes were also compared in order to attain an in depth analysis. The number of features detected, the percentage of valid estimates out of the total computed and the RMS errors both in the vertical and horizontal axis were used as performance measures. This way, all different implementations of the systems can be meaningfully compared.

This work has presented and discussed localization results both in terms of relative displacement and also absolute position with respect to a global origin. However, absolute localization fails frequently, and since the dataset does not contain proper loops, which would trigger place recognition after some frames, ORB-SLAM is not able to recover properly in SLAM mode. In such cases, Visual SLAM clearly needs input from a reliable global

localization system such as GNSS, or external landmarks whose position is well known.

6.1 Future Work

This work initially aimed to compare performance of line based approaches as well. However, available open source implementations of line based VO or Visual SLAM approaches are out dated, causing at times serious syntax, dependency and binary compatibility issues. Therefore, line based analysis was left out. Porting existing line based approaches to work with modern tools would allow a comparison of point based and line based approaches in haze conditions. Despite the results provided by the KITTI dataset, an up to date way of testing the system would be using a dataset recorded using the state of the art sensors of the Field Robotics Lab on the university campus. This would allow another confirmation of the performance of the used systems.

7 References

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, pp. 1147–1163, October 2015.

[2] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-d cameras," *IEEE Transactions on Robotics*, vol. 33, pp. 1255–1262, October 2017.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, pp. 1874–1890, December 2021.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[5] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, pp. 829–846, April 2018.

[6] M. Aldibaja, N. Suganuma, K. Yoneda, and R. Yanase, "Challenging environments for precise mapping using GNSS/INS-RTK systems: Reasons and analysis," *Remote Sensing*, vol. 14, p. 4058, January 2022.

[7] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," *Journal of Modern Transportation*, vol. 24, pp. 284–303, December 2016.

[8] I. Yaqoob, L. U. Khan, S. M. A. Kazmi, M. Imran, N. Guizani, and C. S. Hong, "Autonomous driving cars in smart cities: Recent advances, requirements, and challenges," *IEEE Network*, vol. 34, pp. 174–181, January 2020.

[9] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 99–110, June 2006.

[10] D. Scaramuzza and F. Fraundorfer, "Visual odometry, part i: The first 30 years and fundamentals," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[11] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters*, vol. 5, pp. 422–429, April 2020.

[12] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, pp. 1004–1020, August 2018.

[13] R. Gomez-Ojeda, F.-A. Moreno, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo SLAM system through the combination of points and line segments," *IEEE Transactions on Robotics*, vol. 35, pp. 734–746, June 2019.

[14] J. Lu, Z. Fang, Y. Gao, and J. Chen, "Line-based visual odometry using local gradient fitting," *Journal of Visual Communication and Image Representation*, vol. 77, p. 103071, May 2021.

[15] H. Lim, J. Jeon, and H. Myung, "UV-SLAM: Unconstrained line-based SLAM using vanishing points for structural mapping," *IEEE Robotics and Automation Letters*, vol. 7, pp. 1518–1525, April 2022.

[16] G. Harish Babu and N. Venkatram, "A survey on analysis and implementation of state-of-the-art haze removal techniques," *Journal of Visual Communication and Image Representation*, vol. 72, p. 102912, October 2020.

[17] M. I. Anwar and A. Khosla, "Vision enhancement through single image fog removal," *Engineering Science and Technology, an International Journal*, vol. 20, pp. 1075–1083, June 2017.

[18] S. Yin, Y. Wang, and Y.-H. Yang, "Attentive U-recurrent encoder-decoder network for image dehazing," *Neurocomputing*, vol. 437, pp. 143–156, May 2021.

[19] D. Zhao, L. Xu, L. Ma, J. Li, and Y. Yan, "Pyramid Global Context Network for Image Dehazing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, pp. 3037–3050, August 2021.

[20] J. Zhang and D. Tao, "FAMED-Net: A Fast and Accurate Multi-Scale End-to-End Dehazing Network," *IEEE Transactions on Image Processing*, vol. 29, pp. 72–84, 2020.

[21] W.-T. Chen, H.-Y. Fang, J.-J. Ding, and S.-Y. Kuo, "PMHLD: Patch Map-Based Hybrid Learning DehazeNet for Single Image Haze Removal," *IEEE Transactions on Image Processing*, vol. 29, pp. 6773–6788, 2020.

[22] L. Schaul, C. Fredembach, and S. Süsstrunk, "Color image dehazing using the near-infrared," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 1629–1632, November 2009.

[23] C. O. Ancuti and C. Ancuti, "Single Image Dehazing by Multi-Scale Fusion," *IEEE Transactions on Image Processing*, vol. 22, pp. 3271–3282, August 2013.

[24] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1956–1963, June 2009.

[25] M. F. Al-Sammaraie, "Contrast enhancement of roads images with foggy scenes based on histogram equalization," in *2015 10th International Conference on Computer Science & Education (ICCSE)*, pp. 95–101, July 2015.

[26] Q. Zhu, J. Mai, and L. Shao, "A Fast Single Image Haze Removal Algorithm Using Color Attenuation Prior," *IEEE Transactions on Image Processing*, vol. 24, pp. 3522–3533, November 2015.

[27] D. Berman, T. Treibitz, and S. Avidan, "Non-local Image Dehazing," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1674–1682, IEEE, June 2016.

[28] R. Singéis, S. Dogru, and L. Marques, "Performance analysis of orb-slam in foggy environments," in *ROBOT2023: Sixth Iberian Robotics Conference*, 2023. Submitted.

[29] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 229–241, June 2001.

[30] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the National Conference on Artificial Intelligence*, November 2002.

[31] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel, "A comprehensive survey of visual SLAM algorithms," *Robotics*, vol. 11, p. 24, February 2022.

[32] P. Smith, I. D. Reid, and A. J. Davison, "Real-time monocular SLAM with straight lines," in *26*, BMVA, January 2006.

[33] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "StructSLAM: Visual SLAM with building structure lines," *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 1364–1375, April 2015.

[34] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4503–4508, May 2017.

[35] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust visual SLAM with point and line features," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1775–1782, September 2017.

[36] G. Zhang, D. H. Kang, and I. H. Suh, "Loop closure through vanishing points in a line-based monocular SLAM," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4565–4570, May 2012.

[37] J. Ma, X. Wang, Y. He, X. Mei, and J. Zhao, "Line-based stereo SLAM by junction matching and vanishing point alignment," *IEEE Access*, vol. 7, pp. 181800–181811, 2019.

[38] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, pp. 112–122, December 1973.

[39] H. Chang and F. Tsai, "Vanishing point extraction and refinement for robust camera calibration," *Sensors*, vol. 18, p. 63, January 2018.

[40] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, pp. 794–805, October 2013.

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

[42] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *Journal of Field Robotics*, vol. 27, no. 5, pp. 587–608, 2010.

[43] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[44] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, pp. 1188–1197, October 2012.

[45] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, pp. 796–803, April 2017.

[46] M. Calonder, V. Lepetit, C. Strecha, P. Fua, K. Daniilidis, P. Maragos, and N. Paragios, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*, pp. 778–792, 2010.

[47] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), Lecture Notes in Computer Science, pp. 430–443, Springer, 2006.

[48] C. Harris and M. Stephens, "A combined corner and edge detector," in *Procedings of the Alvey Vision Conference 1988*, pp. 23.1–23.6, Alvey Vision Club, 1988.

[49] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, pp. 291–307, February 1999.

[50] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 448–461, March 2010.

[51] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1465–1479, September 2006.

[52] W. Middleton, "Vision through the atmosphere in geophysik ii/geophysics ii," *Springer*, vol. 14, pp. 254–287, 1957.

[53] J.-P. Tarel, N. Hautiere, L. Caraffa, A. Cord, H. Halmaoui, and D. Gruyer, "Vision enhancement in homogeneous and heterogeneous fog," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, no. 2, pp. 6–20, 2012.

[54] C. Rizos, "Network RTK Research and Implementation: A Geodetic Perspective," *Journal of Global Positioning Systems*, vol. 1, no. 2, pp. 144–150, 2002.

[55] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, May 2011.