Sparse Stereo Visual Odometry with Local Non-Linear Least-Squares Optimization for Navigation of Autonomous Vehicles

by

Edgar Fabian Aguilar Calzadillas

B.Sc. in Mechatronics Engineering

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Aerospace Engineering

Carleton University
Ottawa, Ontario

Abstract

In this thesis, the author presents a Sparse Stereo Visual Odometry system for navigation of autonomous vehicles. The proposed system has the capability to estimate the camera's pose based on its surrounding environment. In contrast to other Visual Odometry systems with Bundle Adjustment optimization, the system proposed in here differs in four main aspects: (1) it utilizes both stereo frames to track features between frames; (2) it does not require a bootstrap step to initialize the algorithm; (3) it performs a local optimization at every increment frame instead of perform a windowed optimization; and (4) it consider the both stereo images inside the optimization instead of just one side of the stereo system. The system was tested on the Karlsruhe Institute of Technology (KITTI) Vision Benchmark Suit, as well as with a set of video sequences recorded with commercial stereo cameras on the roads of the city of Ottawa, Ontario.

Acknowledgements

I would like to thank my supervisor Professor Jurek Z. Sasiadek for giving me the opportunity to join the Aerospace Engineering program at Carleton University, and persuading me to achieve more and reach my goals in this thesis project. There are not enough words to describe my gratitude to his advice and help during this years of research.

To all my friends who were far and worried about how I was doing far away from home during the completion of my master's degree, and to those who were close encouraging me to keep on working as hard as I could to finish this project.

To my family, who were concerned about me leaving my professional career behind to come back to university and persuade my dream of getting my master's degree. I know that for my family this project must have seemed like a strange idea. However, they were very supportive during all this time.

To Melisa, thank you for being there all those long nights when I was working on this project and for supporting me and motivating me to keep on trying once and once again.

Table of Contents

List of Tables

List of Figures

List of Appendices

Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AI	Artificial Intelligence
AV	Autonomous Vehicle
BA	Bundle Adjustment
C++	Programing language C++
DD	Direct Disparity method
DLT	Direct Linear Transformation
DoF	Degrees of Freedom
DoG	Difference of Gaussians
FHD	Full High Definition
FLANN	Fast Library for Approximate Nearest Neighbor
GNSS	Global Navigation Satellite System
GPU	Graphic Processing Unit
GT	Ground Truth
HD	High Definition
IMU	Inertial Measurement Unit
KIT	Karlsruhe Institute of Technology
KITTI	Karlsruhe Institute of Technology and Toyota Institute
LDP	Location Determination Problem
LiDAR	Light Detection and Ranging
MATLAB	Matrix Laboratory software
NASA	National Aeronautics and Space Administration
NCC	Normalized Cross Correlation
OpenCV	Open Source Computer Vision
P3P	Perspective-three-Point
P6P	Perspective-six-Point
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
RGB	Red Blue Green
RGBA	Red Blue Green Alpha
RMS	Root Mean Square
SDD	Sum of Squared Differences
SDK	Software Development Kit
SfM	Structure from Motion
SGM	Semi-Global Matching
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speed Up Robust Features
SVD	Singular Value Decomposition
SVO	Stereo Visual Odometry
SSVO	Sparse Stereo Visual Odometry

TTI-C	Toyota Technological Institute at Chicago
VO	Visual Odometry
VSLAM	Simultaneous Localization and Mapping
UV	Unmanned Vehicle
ZED	Commercial stereo camera manufactured by Stereolabs
ZED-Mini	Commercial stereo camera manufactured by Stereolabs

Nomenclature

Greek Letters

γ	Skew factor
θ	Roll angle
κ	Radial distortion coefficients
λ	Eigenvalues
ρ	Tangential distortion coefficients
σ	RMS-distance
ϕ	Pitch angle
ψ	Yaw angle
Δ	Difference
$\prod_L$	Image plane left
$\prod_R$	Image plane right
Σ_k	Covariance matrix for error propagation

Roman Letters

$\vec{a}$	Set of descriptors corresponding to the keypoints over the left image plane
$\vec{b}$	Set of descriptors corresponding to the keypoints over the right image plane
b	Baseline
c	Center of the image plane
C_k	Camera pose at a current instant of time
C_{k-1}	Camera pose at a previous instant of time
d	Disparity
e	Epipole
e_L	Epipole corresponding to the left image plane
e_R	Epipole corresponding to the right image plane
$\boldsymbol{E}$	Essential matrix
f	Focal length
$f_{L,k}$	Set of features in the left image plane at current instant of time
${}^{t}f_{L,k-1,k}$	Set of left tracked left features between two consecutive frames
$f_{R,k}$	Set of features in the right image plane at current instant of time
${}^{t}f_{R,k-1,k}$	Set of left tracked right features between two consecutive frames
f_x	Focal length in x-component
f_y	Focal length in y-component
$\boldsymbol{F}$	Fundamental matrix
g	Function of 3D points reprojected into 2D features at a certain camera pose

GT	Sub index that denotates Ground Truth
I	Image matrix
I_L	Left image matrix
I_R	Right image matrix
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{J}$	Jacobian matrix
$\boldsymbol{K}$	Intrinsic matrix
$\boldsymbol{K}_L$	Left intrinsic matrix
$\boldsymbol{K}_R$	Right intrinsic matrix
k	Sub index which denotates time
l_L	Epipolar line over the left image plane
l_R	Epipolar line over the right image plane
M	Structure tensor
$\boldsymbol{M}$	Camera projection matrix
$\boldsymbol{M}_L$	Left camera projection matrix
$\boldsymbol{M}_R$	Right camera projection matrix
p	Set of keypoints
p_L	World point projected into the left image plane
$\check{p}_L$	Normalized points in the left image plane
$\hat{p}^i_{L,k-1}$	Reprojected keypoints from X^i_{k-1}into left image plane
p_R	World point projected into the right image plane
$\check{p}_R$	Normalized points in the right image plane
$\hat{p}^i_{R,k-1}$	Reprojected keypoints from X^i_{k-1}into right image plane
p_x	Horizontal component of a set of keypoints
p_y	Vertical component of a set of keypoints
${}^{w}\boldsymbol{P}$	World point
P	World point in homogeneous form
$Q_{k-1,k}$	Vector with parameters to optimize on local optimization algorithm
$\boldsymbol{R}$	Rotation matrix
t	Translation vector
T	Normalizing transformation
T_k	Transformation at a current instant of time
$T_{k-1,k}$	Transformation at a current instant to time with respect to a previous one
u	Pixel coordinates in the vertical axis
u_0	Vertical component of the center image plane
u_L	Vertical pixel coordinate of a projected world point
v	Pixel coordinates in the horizontal axis
v_0	Horizontal component of the center image plane
v_R	Horizontal pixel coordinate of a projected world point
VO	Sub index that denotates Visual Odometry
W	Window patch
x_L	X component left camera coordinate system
x_R	X component right camera coordinate system
X	X component world coordinate system
X_k	Point cloud of triangulated features at current instant of time

y_L	Y component left camera coordinate system
y_R	Y component right camera coordinate system
Y	Y component world coordinate system
z_L	Z component left camera coordinate system
z_R	Z component right camera coordinate system
Z	Z component world coordinate system

Chapter 1

Introduction

The main goal of *Computer Vision Systems* is to develop applications that improve the robotic field, specifically, the field of unmanned vehicles (UV), by providing visual tools for control and artificial intelligence (AI) of the UVs. It provides cameras which serve as an ocular sensor for machines, and it involves the parametrization and geometrization of the environment where the UV is performing.

One of the most well-known problem is the so-called *Structure from Motion* (SfM) problem. This problem aims to recover the relative pose of a camera mounted in an agent and to build a three-dimensional (3D) map in six degrees of freedom (6-DOF) of the environment from a set of camera images. SfM has the capability of recovering the structure from sequentially ordered or unordered image sets.

In the literature, derivations of the SfM can be found. Among them, it is found the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it, which is known as *Simultaneous Localization and Mapping* (SLAM). SLAM is a way for a robot to localize itself in an unknown environment while incrementally constructing a map of its surroundings. Its consequent derivation to camera vision systems is known as *Visual Simultaneous Localization and Mapping* (VSLAM). In contrast to SfM, VSLAM focuses on globally consistent estimation, while keeping track of image points (keypoints) in two-

dimensions (2D) and landmarks in three-dimensions (3D) to build a three-dimensional map of the surroundings where the agent is navigating.

Visual Odometry (VO) is a particular case of SfM and is indeed meritorious of special attention in the machine vision community. It focuses on the estimation of the motion in 6-DOF of the camera (mounted in a mobile agent) from images ordered sequentially only; for example, as a new frame is captured by the camera system, a new motion is estimated. In the literature, it is common to find that the *Structure from Motion* problem is frequently referred to as a synonym of Visual Odometry. In comparison to VSLAM, however, Visual Odometry might or might not keep track of all previous history information of the camera.

The wide majority of *Visual Odometry* systems are real-time applications and they have been introduced in different scientific and industrial fields, including robotics, augmented and virtual reality and automotive applications, and wearable computing. In contrast to other localization and tracking systems, VO has its own irreplaceable advantages.

Firstly, compared to wheel odometry, Visual Odometry is not affected by wheel slippage and the trajectories are better estimated. Secondly, another indisputable advantage of VO systems is relative lower costs, especially in comparison to other sensors such as the Light Detection and Ranging (LiDAR). The latter is not only more expensive in economic terms, but the interpretation of the LiDAR output (a point cloud) itself is computationally costly. Moreover, miscalculations in the LiDAR might produce erroneous commands to the control system of an unmanned vehicle. Additionally, the flexibility provided by VO systems allows them to adapt them to any kind of environment, making it easier to optimize

them so as to produce results and trajectories to the extent that even high-precision inertial measurement units (IMUs) cannot achieve.

Visual Odometry systems are often divided in three categories known as monocular, binocular, and trifocal tensors. Most of the monocular applications require a bootstrap initialization, which is solved by the mere translation of the first two captured image frames. Monocular Vision Odometry systems are the simplest and less sophisticated, and for this reason, they require being complemented and assisted by other sensors, namely LiDAR or IMU, in order to solve its inherent translation and scene reconstruction scale ambiguity. For the aforementioned reason, monocular applications are not often found in industrial applications. In other hand, binocular and trifocal tensors do not present the problem of scale ambiguity due to their more sophisticated and complex capability to estimate depth over the scene and its surroundings, which is used to build high-precision maps as well as scene reconstructions.

The platform chosen to implement this work was the software MATLAB with a wrapper interface to OpenCV libraries. Even thought, OpenCV is a library and collection of algorithms implemented in C++ programing language, which are being used by a large number of researchers, MATLAB has advantages over C++, such as: (1) the powerful matrix library, since MATLAB language is based on linear algebra routines; (2) the large collection of toolboxes related to the field of computer vision and optimization; (3) visualization of results and code debugging is easier to perform in MATLAB than in C++; and finally (4) MATLAB is a software that is well documented in every single toolbox function, whereas OpenCV is not due its open source nature.

In addition to the Visual Odometry system presented in this thesis, this work has been tested and evaluated using an academic visual odometry dataset. This dataset is the *KITTI vision benchmark suit*, which was developed by the *Karlsruhe Institute of Technology* (KIT) at Karlsruhe, Germany, in collaboration with and the *Toyota Technological Institute* (TTI-C) at Chicago, USA. The dataset provides stereo images captured by a stereo vision system mounted on a mobile vehicle. Also, it provides 'ground truth' camera poses built from the fusion of a *Velodyne Light Detection and Ranging* (LiDAR) and a *Global Navigation Satellite System* (GNSS) system.

Finally, the author has built a dataset similar to the KITTI one in the city of Ottawa, ON, using two commercial stereo cameras. The ZED and ZED-Mini are two low cost stereo cameras manufactured by Stereolabs. Essentially, both cameras have the same technical specifications with the difference in their lens separation (baseline), the ZED camera has a baseline of 120 mm, meanwhile the ZED-Mini has a baseline of 63 mm.

In this thesis, the author presents an optimized algorithm for *Stereo Visual Odometry* (SVO) systems based on two-dimensional feature tracking and processing motion estimation by implementing the well-known *Perspective-three-Point* (P3P) problem, which aims to determine the position and orientation of the camera in the world reference frame from three 2D-3D point correspondences. Finally, the position and orientation computed by the P3P solver is then optimized with a *Local Non-Linear Least-Squares Optimization* at every incremental frame or instant of time.

In comparison to the classical Visual Odometry pipeline, which is optimized with a least-squares approximation called *windowed Bundle Adjustment* (BA), the method presented in this work is advantageous because it provides good results (trajectory

estimation) without needing to keep tracking a multiplicity of previous frames to be used to optimize the final results, that is, the estimation of the trajectory of a mobile vehicle.

1.1 Thesis Contributions

The present work makes valuable contribution to the field of *Robotics and Computer Vision Systems* in a variety of ways, where the specific meaningful, novel and original contribution of this work is the modification performed by the author to the classic Visual Odometry algorithm. As it will be discussed in detail in the following Chapters, this thesis provides a significant and useful alternative to the Windowed Bundle Adjustment optimization algorithm which is typically used in the aerospace industry, automotive industry, and the robotics field to estimate and optimize the trajectory of an agent into an unknown environment.

The author of this work presents a Visual Odometry system with a Local Non-Linear Least-Squares Optimization algorithm. In contrast to Windowed Bundle Adjustment optimization, which is done every determinate window of image frames, this optimization process is done at every time a frame is processed. The final results or estimated trajectories obtained from the proposed method were closer to the ground truth data provided in the KITTI dataset, compared to the results obtained from the VO system with windowed BA.

The classic Visual Odometry algorithm only keeps track of features on the left or right side of the stereo system, which is the main reason for drifted trajectories. In contrast to the classic VO algorithm, the algorithm proposed in this thesis tracks features in both, left and right sides of the stereo system, which then are processed simultaneously to reduce

the amount of miscalculated image features. This modification allows the system to improve stereo feature correspondences.

The Local Non-Linear Least-Squares Optimization proposed in this thesis work is solved by the Levenberg-Marquardt algorithm. The optimized parameters in this algorithm are the camera location and orientation at every instant of time, where in comparison to the classic visual odometry algorithm, in this approach the orientation is expressed in quaternions and not in Euler angles to avoid the *'gimbal lock'* that Euler angles are prompt to get.

As such, this work provides a simple but effective alternative to windowed bundle adjustment optimization by performing a local non-linear least-squares optimization of the camera pose frame by frame.

Consequently, it also allows for a re-evaluation of the effectiveness and efficiency of different features and descriptors. Also, the algorithms for stereo feature matching and feature tracking are evaluated, as well as the methods for outlier removal on feature sets.

Furthermore, the author presents a study to assess the depth perception based on image resolution and change of baseline (distance between stereo camera lenses) of commercial stereo camera systems. This study presents a methodology to set a threshold in terms of depth perception for Visual Odometry systems by the comparison and discussion of three different depth estimation algorithms.

1.2 Thesis Summary

This thesis is structured in nine Chapters. Chapter 1 provided a concise introduction. Chapter 2 outlines and reviews the state of art and prior work in regards to monocular and

stereo vision odometry systems. Chapter 3 provides an overview of the stereo visual system as well as the fundamental concepts of epipolar geometry, which in turn sets the basis of photometry. Chapter 4 discusses in detail how to extract information from the image frame via features and descriptors; it also explains how to transform the aforementioned information into a 3D Euclidean space so as to recover structure and scene reconstruction. Also, an evaluation of the depth estimation for commercial applications is discussed. Chapter 5 introduces the methodology adopted in this work to perform motion estimation, error propagation, and pose optimization. Also, it presents the general Visual Odometry pipeline algorithms. Chapter 6 introduces the modified Sparse Stereo Visual Odometry with Local Non-Linear Least-Squares Optimization Algorithm proposed by the author. Chapter 7 presents the experimental evaluation of the algorithm with small and large motions. Chapter 8 provides directions and guidance for future work. Finally, Chapter 9 concludes this work by reviewing its main findings and highlighting its contributions.

Chapter 2

Relevant Prior Work

Visual Odometry (VO) is the process of estimating the ego-motion of a vehicle at a known or unknown environment, using only as input the information of a single or multiple cameras attached to it. The term *visual odometry* was first introduced in 2004 by Nistér et al. [1]. The name given to the process was selected because of its similarity to wheel odometry, which incrementally estimates the distance traveled by a mobile vehicle. In the same fashion, Visual Odometry estimates the pose (translation and rotation with respect to a local or global reference coordinate frame) of the vehicle through the analysis of the input images of its surroundings, and their correspondent changes in time which are induced by the vehicle movement.

The problem of ego-motion estimation of a vehicle by observing a sequences of images was first studied and analyzed in 1980s by Moravec [2] at Stanford University. Moravec employed a monocular single camera mounted on a vehicle which was sliding on a rail. By doing so, he emulated a stereo vision system. The vehicle moved in a go and stop fashion, where it would slide in a perpendicular way with respect to the world's scene. The vehicle then would stop to take an image shot, and then repeat the motion until a total of nine images were captured.

Moravec then extracted the corners (keypoints) from the first image and matched them with the other eight images. To match the corners over the sequence of images, he used the Normalized Cross Correlation (NCC) algorithm [3], to subsequently perform 3D

scene reconstruction. Furthermore, the motion of the camera was then calculated by aligning the landmarks (3D keypoints) from the scene reconstruction observed at different locations.

Later on, in 1987, Matthies and Shaffer [4] recovered Moravec's work and extended it by deriving a motion error model using 3D Gaussian distributions as opposed to the scalar model that Moravec presented in his work.

It is interesting to observe that the vast majority of the early research in the field of visual odometry was motivated by the NASA Mars exploration program, in the effort to provide exploration roves with the capability to estimate their own pose in 6 degrees-of-freedom (DoF) in the unknown environment of the Martian surface.

In 1999, Lacroix et al. [5] introduced a method for self-localization of rovers in unknown outdoors environments that relied on raw images (considering lenses distortion) instead of geometric constrained (rectified images) data for motion estimation. In their work, they used dense stereo matching processing, to then select candidate key points using Forster detector and analyzing the correlation function around their peaks.

A few years later, in 2006, Cheng et al. [6], [7] recovered Lacroix's work and used it in their final Visual Odometry implementation on board of the Mars rovers. In their work, they used the Harris corner detector [8] algorithm to extract key points from the image planes. Then, they used the correlation function around the keypoint – as presented by Lacroix et al. –. Finally, as proposed by Nistér et al. [1], they implemented the Random Sample Consensus (RANSAC) [9] in the motion estimation step for outlier keypoint rejection.

In recent decades, a diversity of algorithms have been developed[10]–[13]. Those can be divided into two main categories based on the platform of implementation. On the one hand, there are monocular visual odometry systems that use a single camera [14]. On the other, we can find binocular camera systems, notably stereoscopic or stereo camera [10].

The monocular system has a disadvantage over stereo systems because the single camera pipelines can only recover the motion of the vehicle up to an unknown scale factor. The scale factor can be determined by the integrations or fusion with other sensors, namely the Inertial Measurement Unit (IMU) [15]. In other hand, stereo visual odometry has the capability to compute depth by performing triangulations of correspondence keypoints over the stereo image pairs.

One of the most important contributions to visual odometry is Scaramuzza's work[16]–[19]. He summarizes the visual odometry algorithms depending on the platform of implementation. Scaramuzza categorized visual odometry in three different types depending on the information that is being processed:

- *2D-to-2D:* Feature correspondences motion estimation;
- *3D-to-3D:* 3D structure correspondences motion estimation; and
- *3D-to-2D:* 3D structure to feature 2D correspondences motion estimation.

Each of the types listed above can be sub-divided into three different categories, as follows:

- *Dense VO* is focused on getting the relative pose estimation of the camera considering most of the image information, e.g. it considers +300,000 or more pixels [20];
- *Semi-Dense VO,* which in contrast to Dense VO, considers less information from the image, e.g. it considers only corners and/or edges ~10,000 pixels [21]; and

- *Sparse VO* considers only *keypoints*, which in most of the cases are corners or points of interest with high values on their gray scale gradients and evenly distributed over the whole image plane, e.g. it considers ~2,000 pixels [22], [23].

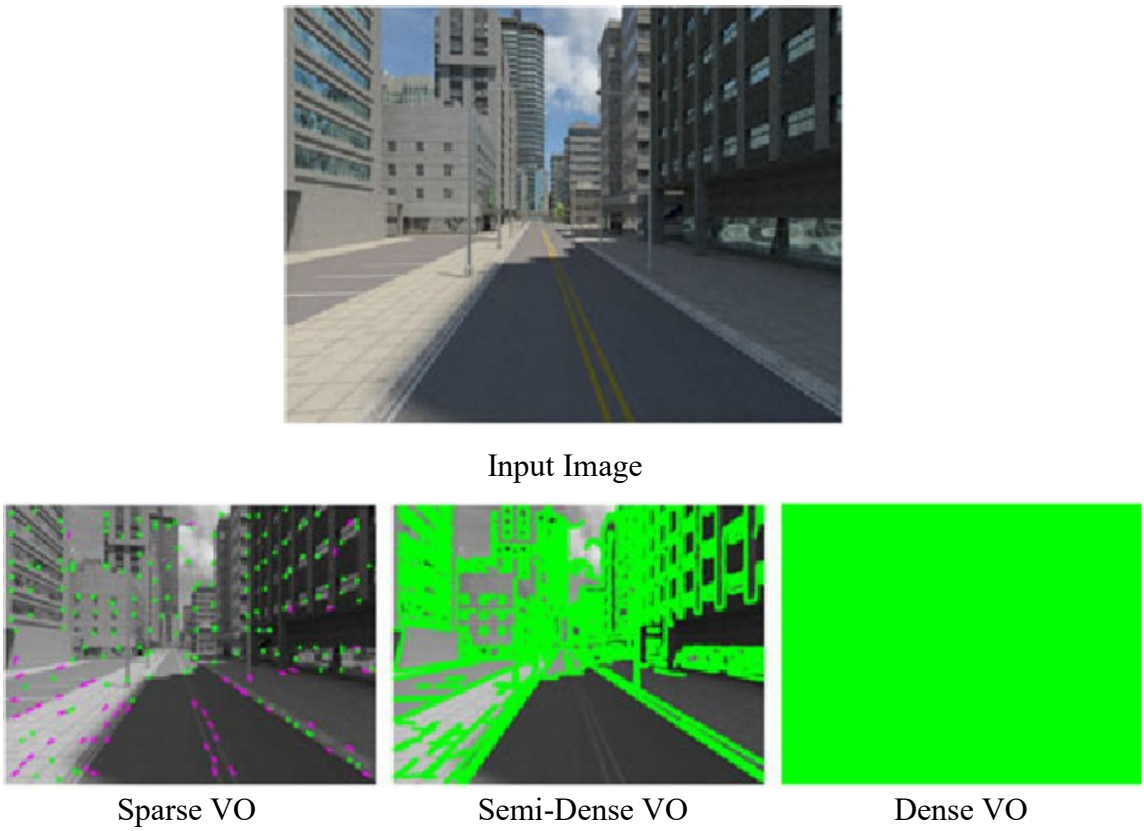


Figure 1. Visual Odometry Systems Categories [24].

Chapter 3

System Overview

In similarity to Sparse Visual Odometry pipelines, the system presented in this thesis consists of four main components, namely feature detection, scene reconstruction, motion estimation, and motion optimization.

In its first stage, this system performs feature detection and feature descriptor extraction. After processing the input images, the set of features are analyzed through the *Epipolar Constraint* using the *five-point* algorithm in addition to *Random Sample Consensus* (RANSAC) to remove outliers and potential erroneous matches. Once the sample of matching features is attained, scene reconstruction is obtained via *Direct Linear Transformation* (DLT), *Semi-Global Matching* to generate a *Depth Map,* or *Direct Disparity,* to achieve scene reconstruction (in the form of a 3D point cloud) from the features extracted. Motion estimation is then computed by solving the *Perspective-three-Point* (P3P) problem, which provides an initial guess of the motion of the camera through the space in 6 Degrees-of-Freedom (DoF). Finally, the system performs a motion optimization via *Local Non-Linear Least-Squares Optimization* which minimizes the reprojection error of 3D structure features into 2D features over the image plane.

In this Chapter, the author introduces the basis of the stereo camera system setup and the camera calibration algorithm, as well as stereo image rectification, and the epipolar geometry constraint, which are fundamentals blocks for every visual odometry pipeline.

3.1 The Stereo Camera Setup

The stereo camera setup consists in a set of two cameras colinear to each other and separated by the *baseline* or distance between lenses, perpendicular to the world scene. These systems, in contrast to the monocular systems are scale-ambiguity-free, which allow them to produce scene reconstruction in the form of point clouds up to a known scale (metric).

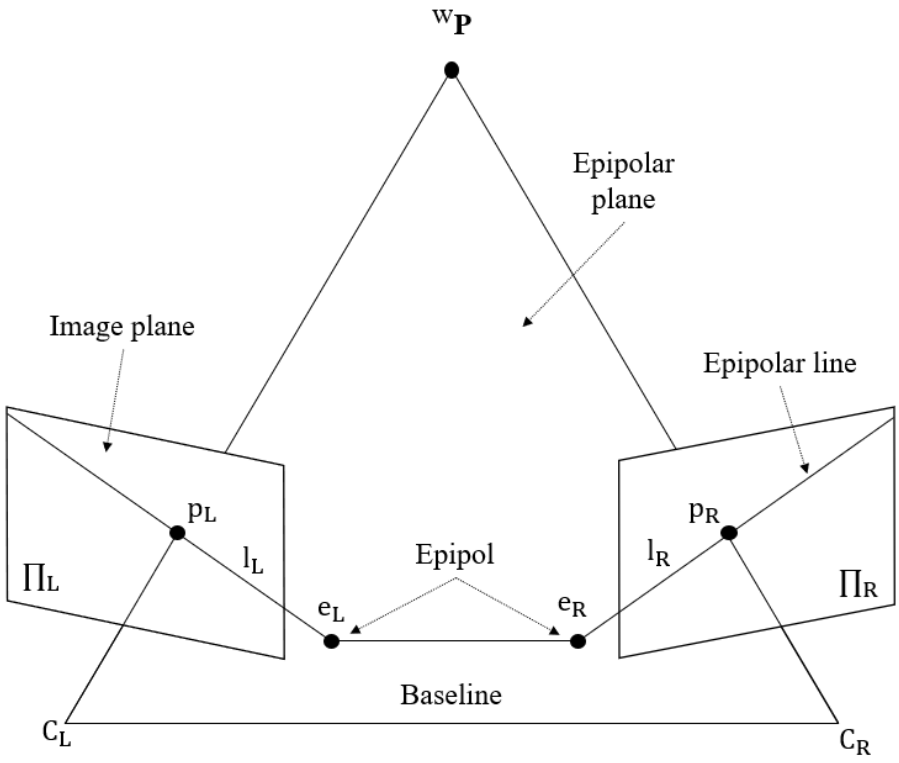


Figure 2. Epipolar Geometry Constraint.

In general, a stereo system consists of two identical cameras that capture the left and right video or sequences of images of a determinate world scene. The conventional configuration of the stereo vision system is shown in Figure 2. A point in the world

described in three-dimensional (3D) Euclidean space is projected into the image plane of both cameras. Thus, two-dimensional (2D) points on image planes of the 3D point are projected [25].

In order to obtain the depth information of the point in the world, we must first solve one of the inherent stereo vision problems, known as the *correspondence problem*. Such that, the system has the capability to find and exact corresponding interest points on both image planes, left and right. Solving the *correspondence problem* requires high computational power because the disparity (distance between the projected points) of every image point must be computed in order to find its respective depth within the image plane.

Thus, to reduce the computational complexity of the *correspondence problem,* the *Epipolar Geometry* of the stereo vision system must be imposed so that correspondences of 3D points projected into 2D image planes are restricted to certain areas of interest.

In turn, in order to reduce and constraint the problem, a procedure called *camera calibration* must be performed. In the following section of this work, the author introduces the basis of the general *camera calibration* procedure and its application on the reduction of the mentioned *correspondence problem*.

3.2 Camera Calibration

For any machine vision application, the most important task is to determine the camera parameters. Every single camera includes radial and tangential distortions induced by the lenses located in front of the camera sensors, which makes the projection of the light into the camera sensor to be captured in a distorted manner. To remove the sources of error over

the images (radial and tangential distortions), a process called *camera calibration* must be done, such that the *intrinsic* and *extrinsic* parameters of the camera system are computed.

The *intrinsic* parameters describe the information related to the internal components of the camera. These parameters include the *focal length* (f) in x and y direction, *center of the image plane* (c), *radial* (κ) and *tangential* (ρ) *distortion coefficients*, and a *skew factor* (γ), which is usually set to a zero-scale value, except in the circumstance where the system works under very specific circumstances, such as when the input is comprised by a series of images taken from another scene in images.

The *extrinsic* parameters describe the information related to the external components of the camera system. The parameters express mathematically the relation of the lenses that the camera system has in terms of location and orientation; e.g. for a stereo camera each of the lenses has its own coordinate system expressed in 6 DoF, where depending on the system configuration, one of the lenses can be set as the origin. The remaining lens is then referenced with respect to that first one, and the separation of the lenses is established to be the *baseline* of the stereo system.

In order to achieve the calibration of the system, it is necessary to stablish a known coordinate system as reference, which in the literature is known as the *world coordinate system*. The *world coordinate system* allows the stereo camera to be referenced to a known coordinate system and to establish a known metric (‘millimeters, ‘inches’), which is then used to determine the camera’s own local coordinate system.

The best well-known procedure for camera calibration was proposed by Zhang [26] and it consists in taking several pictures of a known pattern (usually, a chess board) with known square dimensions, as shown in Figure 3. These images are then processed to look

for the corners of the pattern. Since the dimensions of the squares are known, it is easier to compute the intrinsic and extrinsic parameters by following Zhang's work, described in detail in [27].

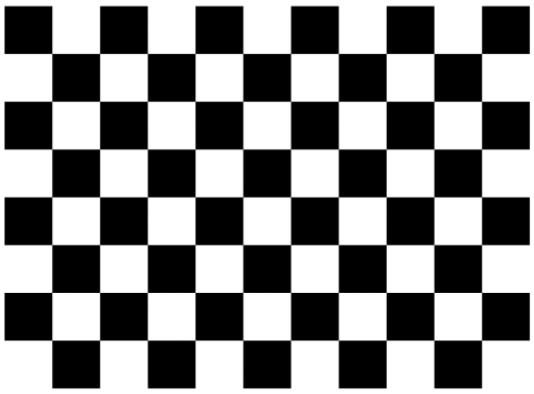


Figure 3. Typical Chess Board Pattern Used for Camera Calibration.

The parameters of the camera system can be expressed in matrix form, such that the problem can be solved by linear algebra. Equation 3.1 shows the *intrinsic matrix* $\boldsymbol{K}$, which contains all the internal parameters in a homogeneous form, thus:

$$\boldsymbol{K} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where, f_x and f_y are the focal length expressed in x and y directions, u_0 and v_0 are the *principal point* or center of the image plane in pixel coordinates, and γ is the skew factor.

The extrinsic parameters are then expressed into the *projection matrix* $\boldsymbol{M}$ as shown in equation 3.2, such that:

$$\boldsymbol{M}_L = \boldsymbol{K}_L[\boldsymbol{I} \mid \boldsymbol{0}] \, , \boldsymbol{M}_R = \boldsymbol{K}_R[\boldsymbol{R} \mid -\boldsymbol{R}t] \tag{3.2}$$

where, $\boldsymbol{M}_L$ and $\boldsymbol{M}_R$ are 3×4 matrices known as *camera projection matrices* of left and right cameras respectively, $\boldsymbol{K}_L$ and $\boldsymbol{K}_R$ are the intrinsic camera matrices of the left and right

cameras, $\boldsymbol{I}$ denotates the 3×3 identity matrix representing the rotation of the left camera in degrees, $\boldsymbol{R}$ is a 3×3 matrix which expresses the rotation of right lens with respect to left, and t is a 3×1 vector which represents the translation (baseline) of camera right with respect to camera left.

The *intrinsic* and *extrinsic* parameters of a stereo system are shown in a graphic format in Figure 4, where it is observed that the calibration patter coordinate system is used as the *world coordinate system* of reference to calibrate the stereo camera. Both lenses have their own coordinate system where one of them (typically the right one) is referenced to the other one, e.g. the right lens is *translated* (t) and *rotated* $(\boldsymbol{R})$ with respect to the left one.

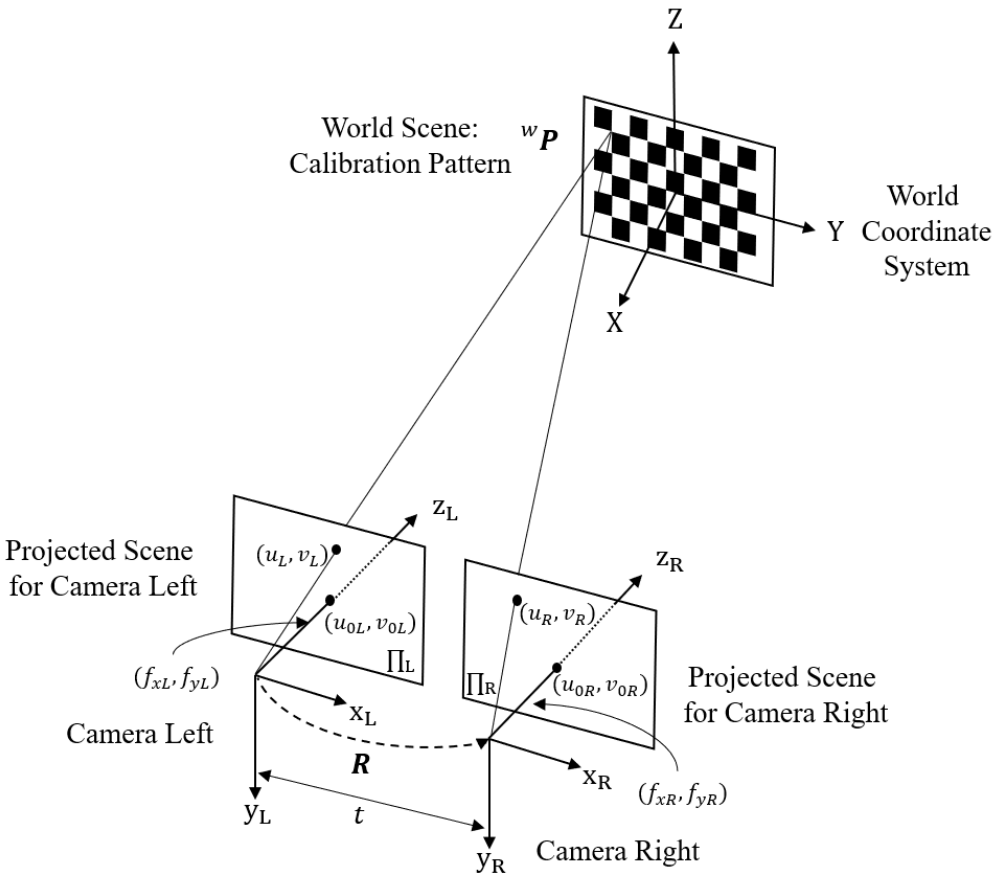


Figure 4. Stereo Vision Coordinate System.

3.3 ZED and ZED-Mini Camera Calibration

For the purposes of this work, two commercial stereo cameras have been used. The ZED and ZED-Mini stereo cameras are shown in Figure 5. These stereo cameras are manufactured by Stereolabs [28]. Even though both cameras come with a Software Development Kit (SDK) application for depth perception, in this thesis the used of this application has been minimized to only extract raw image data.

Figure 5. ZED and ZED-Mini Stereo Cameras [28]**.**

The ZED and the ZED-Mini stereo cameras have the same technical specifications; the main difference between them is the change in the baseline or sensors separation. The ZED stereo camera has a baseline of $120\,mm$, whereas the ZED-Mini comes with a baseline of $63\,mm$. A full description of the technical specifications for both stereo cameras can be found in Appendix A.

In this work, both stereo camera have been calibrated as shown in [26], [29], [30]. The dataset for camera calibration consist of more than 300 pictures of a chessboard pattern of 7×11 as shown in Figure 3. The pictures were taken from an approximated distance of one meter. The software used for stereo camera calibration was MATLAB Stereo

Calibration Toolbox included on the Computer Vision Toolbox [31]. In Figure 6, a sample of the images utilized over the camera calibration process is shown.

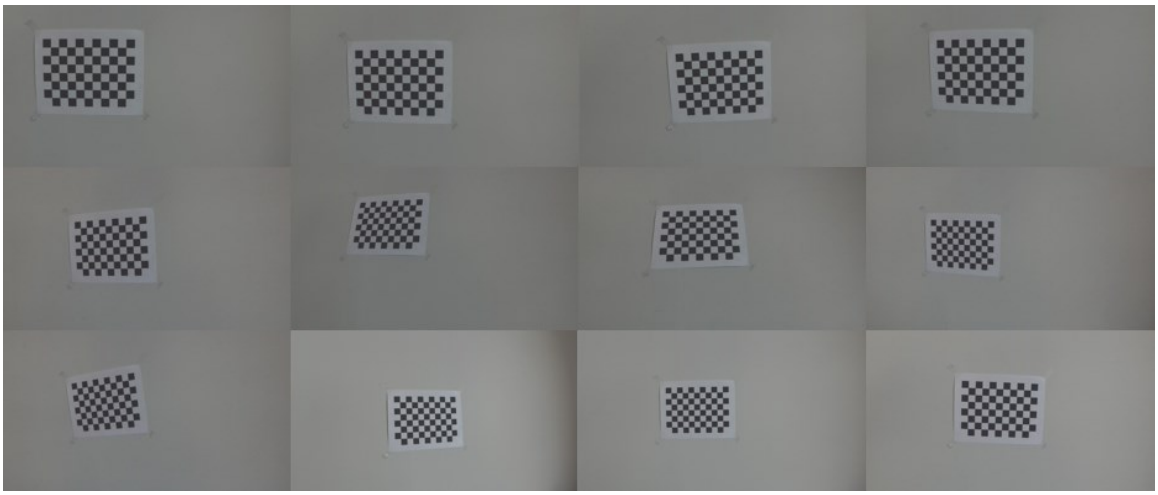


Figure 6. Camera Calibration Input Images.

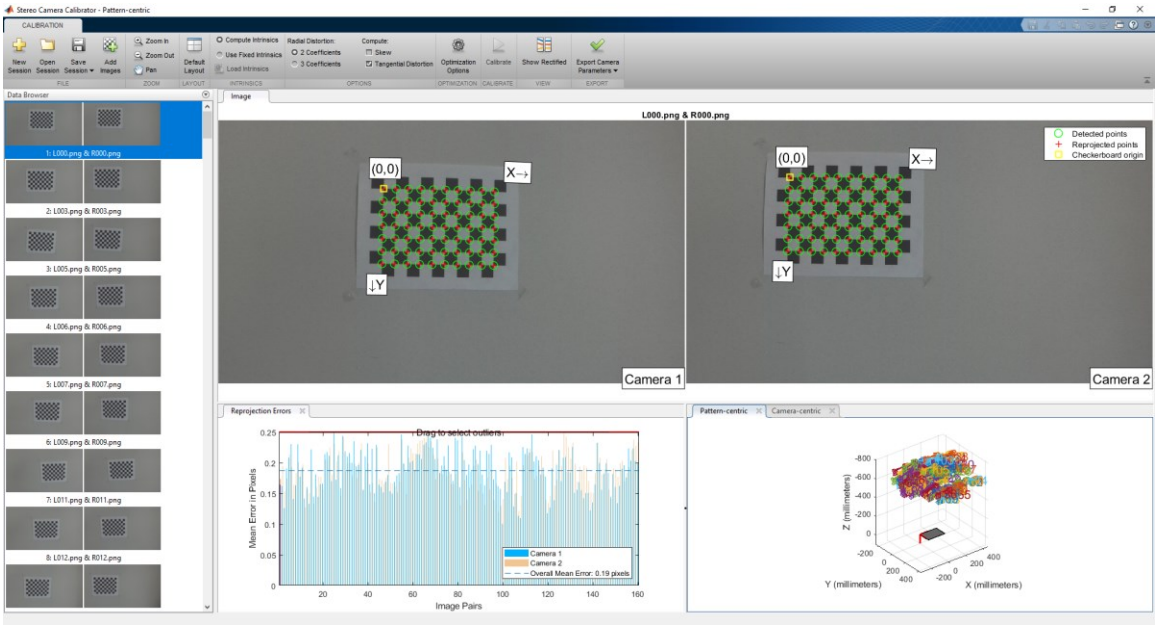


Figure 7. MATLAB Stereo Camera Calibration Toolbox.

The images of the calibration pattern are then processed over the calibration toolbox, where it yields as result the camera parameters, expressed in *intrinsic* and *extrinsic* format, as can be seen in Figure 7. The toolbox has the capability to remove or add images that are either correct or erroneous, and also shows the accuracy of the calibration in terms of reprojection error.

In a similar fashion, the *extrinsic* parameters can be observed. As it is shown in Figures 8 and 9, both cameras ZED (Figure 8) and ZED-Mini (Figure 9), are posed with respect to the world reference frame or the calibration pattern.

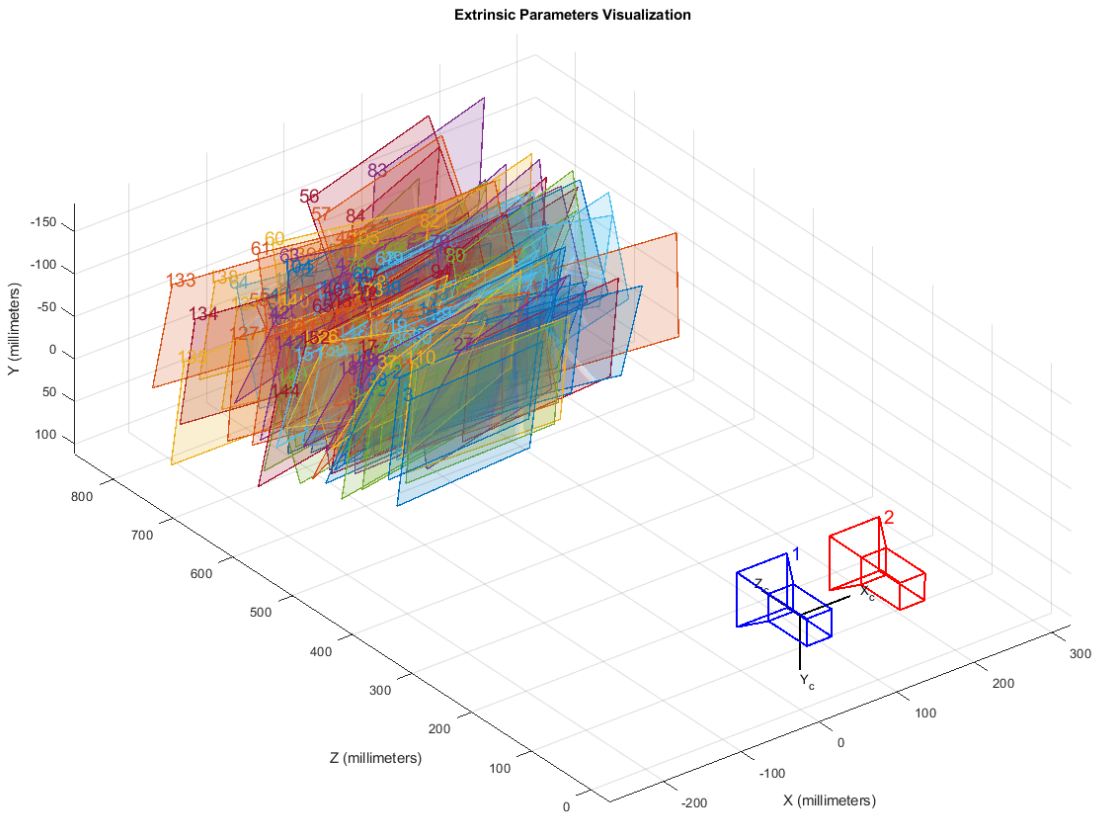


Figure 8. ZED Camera Extrinsic Parameters Visualization.

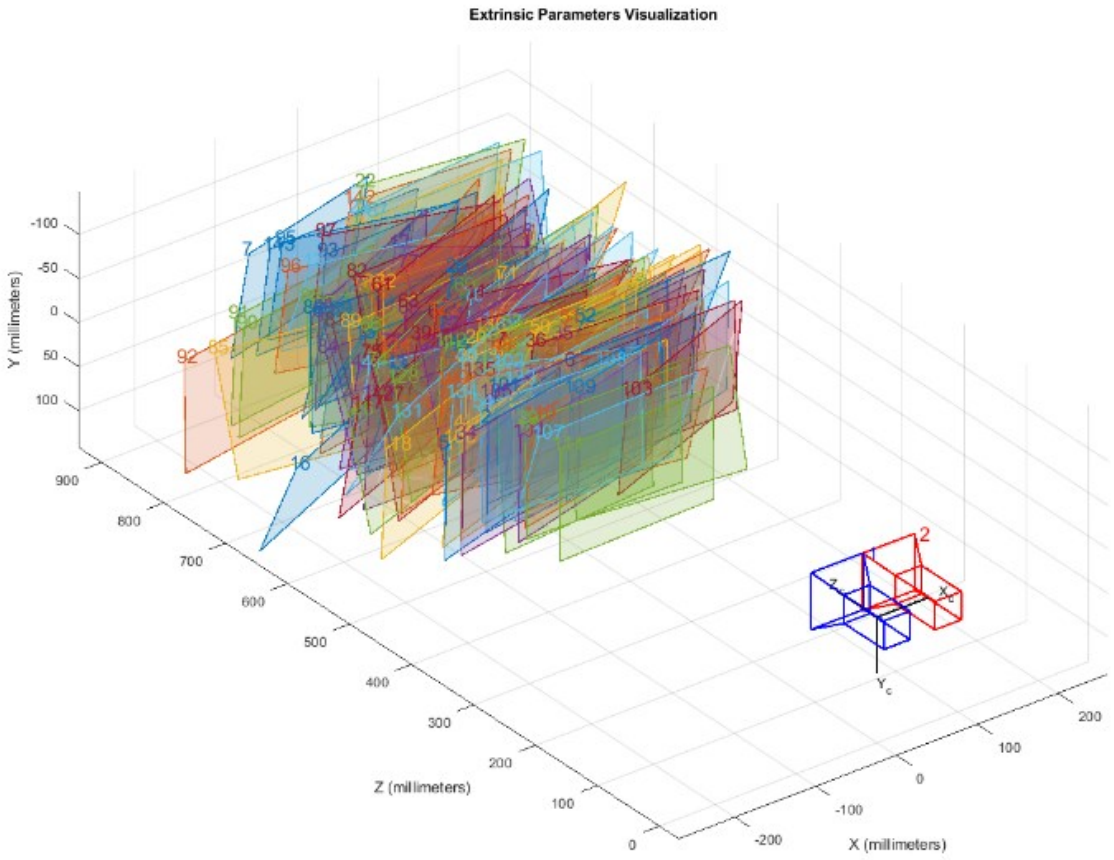


Figure 9. ZED-Mini Camera Extrinsic Parameters Visualization.

Finally, a summary of the calibration parameters for the ZED and ZED-Mini stereo cameras can be found in Table 1 to 4, where the calibration has been done for two different resolutions, Full High Definition (FHD) resolution of 1080×1920 pixels and High Definition (HD) resolution of 720×1280 pixels.

For the purposes of this work, camera calibration has been done only for the ZED and ZED-Mini stereo cameras, because the KITTI visual benchmark suit provides the camera *projection matrices* ($\boldsymbol{M}_L$ and $\boldsymbol{M}_R$).

Table 1. ZED Stereo Camera Parameters – FHD

ZED Stereo Camera Parameters – FHD (1080 x 1920)		
Intrinsic Parameters		
Parameter:	**Left Sensor**	**Right Sensor**
Image Size $[pixels]$	[1080, 1920]	[1080, 1920]
Radial Distortion	[-0.1713, 0.0235]	[-0.1616, 0.0098]
Tangential Distortion	[6.1917e-04, -7.4807e-04]	[3.2921e-04, -5.0884-04]
Focal Length $[pixel/mm]$	[1389.8916, 1395.7836]	[1387.8748, 1395.8984]
Principal Point $[pixels]$	[940.1720, 581.8396]	[950.5980, 513.7520]
World Units	'millimeters'	'millimeters'
Extrinsic Parameters		
Rotation of R to L $[deg]$	[-0.0107, 0.0085, -0.0019]	
Translation of R to L $[mm]$	[-120.2315, 0.2819, -0.3282]	

Table 2. ZED Stereo Camera Parameters – HD

ZED Stereo Camera Parameters – HD (720 x 1280)		
Intrinsic Parameters		
Parameter:	**Left Sensor**	**Right Sensor**
Image Size $[pixels]$	[720, 1280]	[720, 1280]
Radial Distortion	[-0.1716, -0.0029]	[-0.1642, 0.0115]
Tangential Distortion	[4.0899e-04, -29342e-04]	[1.2349e-04, 2.9108e-04]
Focal Length $[pixel/mm]$	[698.3691, 700.7696]	[698.1083, 701.2001]
Principal Point $[pixels]$	[634.4744, 381.2755]	[641.4908, 346.6402]
World Units	'millimeters'	'millimeters'
Extrinsic Parameters		
Rotation of R to L $[deg]$	[-0.0115, 0.0064, -0.0020]	
Translation of R to L $[mm]$	[-120.7001, 0.2924, -0.6832]	

Table 3. ZED-Mini Stereo Camera Parameters – FHD

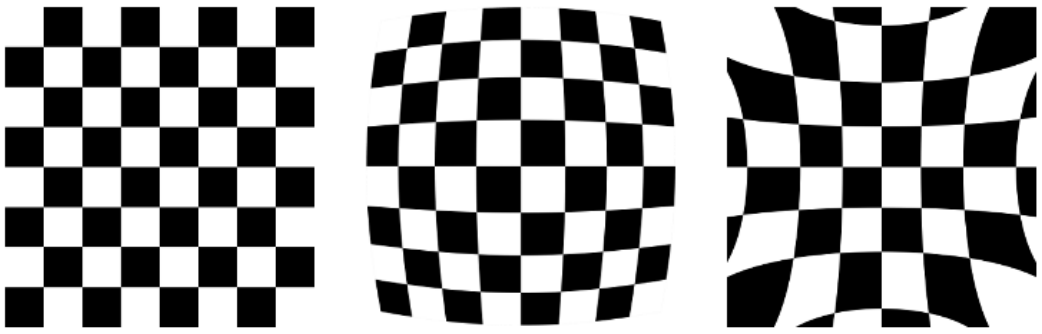
ZED-Mini Stereo Camera Parameters – FHD (1080 x 1920)		
Intrinsic Parameters		
Parameter:	**Left Sensor**	**Right Sensor**
Image Size $[pixels]$	[1080, 1920]	[1080, 1920]
Radial Distortion	[-0.1629, -0.0015]	[-0.1715, 0.0116]
Tangential Distortion	[4.7756e-04, -0.0016]	[4.1845e-04, -0.0019]
Focal Length $[pixel/mm]$	[1387.3801, 1394.2296]	[1385.4878,1392.8351]
Principal Point $[pixels]$	[977.2785, 545.7594]	[993.5550, 525.8243]
World Units	'millimeters'	'millimeters'
Extrinsic Parameters		
Rotation of R to L $[deg]$	[0.0032, -0.0048, -0.0007]	
Translation of R to L $[mm]$	[-63.6606, -0.0516, -0.2000]	

Table 4. ZED-Mini Stereo Camera Parameters – HD

ZED-Mini Stereo Camera Parameters – FHD HD (720 x 1280)		
Intrinsic Parameters		
Parameter:	**Left Sensor**	**Right Sensor**
Image Size $[pixels]$	[720, 1280]	[720, 1280]
Radial Distortion	[-0.1620, -.0122]	[-0.1693, -.0020]
Tangential Distortion	[1.4635e-04, -6.8850e-04]	[8.3469e-05, -0.0013]
Focal Length $[pixel/mm]$	[696.9387, 699.8648]	[696.4279, 699.3971]
Principal Point $[pixels]$	[652.9635, 364.9504]	[660.3474, 355.1498]
World Units	'millimeters'	'millimeters'
Extrinsic Parameters		
Rotation of R to L $[deg]$	[0.0028, -0.0035, -0.0007]	
Translation of R to L $[mm]$	[-63.6153, -0.0845, -0.2534]	

3.4 Stereo Image Rectification and Epipolar Geometry

Stereo Image Rectification, or *Image Rectification*, is the process that removes the radial and tangential distortion of the stereo images by performing 2D transformations over the images using the calibration parameters such that the stereo image pair is the aligned into the horizontal image axis.



Rectified Image Barrel distortion Pincushion distortion

Figure 10. Rectified and Non-Rectified Images Examples [32]**.**

The rectified images can be considered as new images because *image rectification* not only produces the alignment of the left and right images into the horizontal axis, but also it might change the size of the input images to ensure that the centers of the images are at the same high with respect to each other. Figure 10 shows examples of image distortions and the desired rectified image.

Figure 11 shows an example of the left lens of a stereo camera. The picture shows the radial and tangential distortion induced by the lenses. By careful observation, one can also note that the system falls within the category of *barrel distortion.* Note the distortion of the buildings at the rear and right side of the image.

Figure 11. Left Lens Input Image Stereo Camera.

The process of *image rectification* has the advantage of allowing for convenient ways to solve the *correspondence problem* between stereo images, and also for stereo feature

matching. Indeed, it permits to minimize the aforementioned problems from 2D problems into 1D problems [33].

As it is shown in Figure 12, the desired rectified stereo images will be then the ones where all the epipolar lines are aligned to the horizontal axis and meet the *epipoles* at infinity, giving as result epipolar lines which are straight and horizontal. The radial and tangential distortion are removed from the input stereo image pairs and the center of the images are stablished to be at the same high, considerably simplifying the *correspondence problem*.

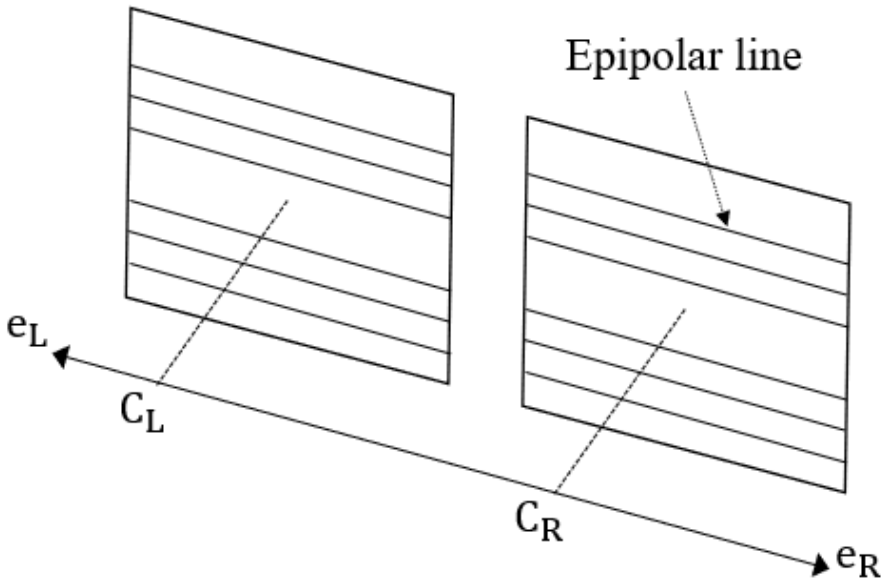


Figure 12. Rectified Stereo Configuration.

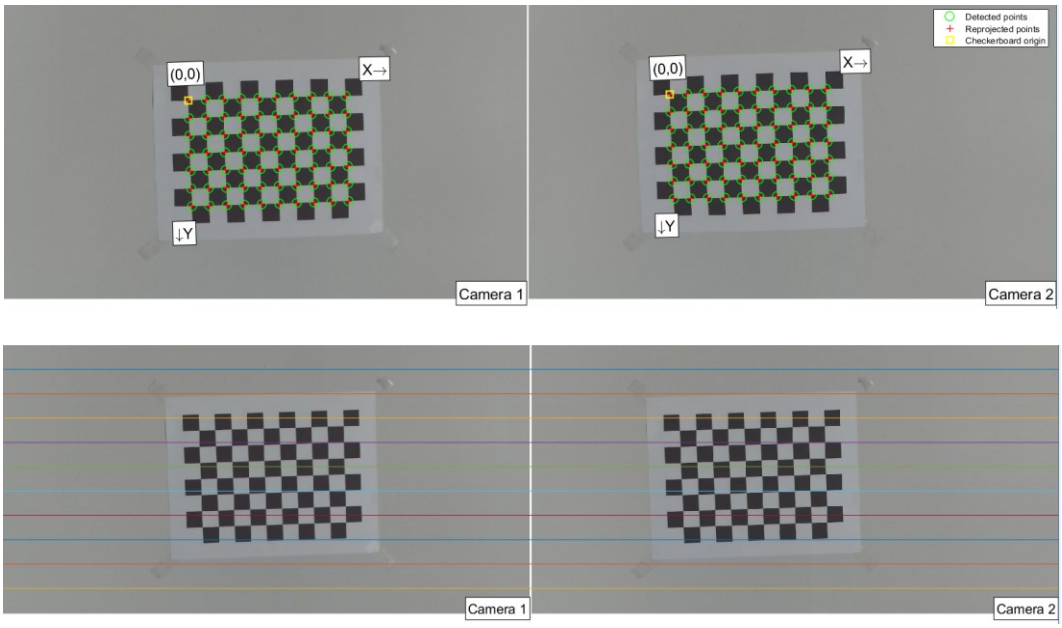


Figure 13. Before (Top) and After (Bottom) Image Rectification with Epipolar Lines.

Once the rectification of the images is completed, the *epipolar geometry constraint* is then imposed over the stereo input images. As shown in Figure 2, in a pair of rectified images, the reprojection of a world point into the *image planes* are the two corresponding points in the left and right images and they always lie over the *epipolar lines*. Figure 2 shows that both projected points are over the epipolar lines, which is an important and strong constraint for finding correspondences between the rectified images. For example, in the left image plane, p_L is the projection of the world point P and its conjugate point in the right image plane lie over the epipolar line l_R. Therefore, it is possible to search the matching or corresponding point only over the line l_R, restricting the search to a one-dimensional problem rather than a two-dimensional problem.

Figure 14. Left Image after Image Rectification.

Figure 14 shows the rectified output image from the image presented in Figure 11, where one can observe how the lenses distortion (radial and tangential distortions) have been removed from the input image. Note how the buildings are not distorted anymore, showing geometric patterns.

It is important to remember that due to the calibration of the camera, the output rectified images might change in size with respect to the input image, as it is shown in Figures 11 and 14. It is easy to observe then that the sizes of the images have changed in order to compensate the distortion of the lenses, from the raw input size of 1080×1920 pixels to 1045×2070 pixels.

3.4.1 The Fundamental Matrix and Essential Matrix

The *Fundamental* and *Essential* matrices mathematically express what is shown in Figure 2 and 12: point – to – epipolar line of a stereo system set up. In this section, the

mathematical model of the *Epipolar Geometry* is introduced, which in the literature it expressed by the *Fundamental and Essential* matrices [33]–[36].

Let us assume that the world point with 3D coordinates is being imaged in both cameras with 2D pixel coordinates, such that its coordinates in homogeneous form would be:

$$^{w}P = [X\ Y\ Z\ 1]^{T}$$

$$p_L = [u_L\ v_L\ 1]^{T}\ , p_R = [u_R\ v_R\ 1]^{T}$$

where: p_L and p_R are the corresponding points on the image planes of the projected world point ^{w}P.

The fundamental matrix $\boldsymbol{F}$ is a 3×3 matrix; it allows us to map a point p_R over the image plane $\prod_L$ and compute its epipolar line l_L. In a similar way, it allows us to map a point p_L over the image plane $\prod_R$ and to compute its epipolar line l_R. The linear mapping representation of the corresponding epipolar lines in each image planes is defined by:

$$l_R = \boldsymbol{F}p_L\ , l_R = p_R\boldsymbol{F} \tag{3.3}$$

If the epipolar line is computed, then the epipolar constraint condition is satisfied, such that: $l_R p_R = 0$, and $l_L p_L = 0$, which yields to:

$$p_R\boldsymbol{F}p_L = 0 \tag{3.4}$$

Similarly, the essential matrix $\boldsymbol{E}$ is a 3×3 matrix that expresses that same relationship as the fundamental matrix, such that:

$$p_R\boldsymbol{E}p_L = 0 \tag{3.5}$$

The explicit form of the fundamental and essential matrices is stated as follows:

$$\boldsymbol{E} = \boldsymbol{R}_{R/L}{}^{T}\,\boldsymbol{S}_{R/L} \tag{3.6}$$

$$\boldsymbol{F} = \boldsymbol{K}_R{}^{-T}\, \boldsymbol{R}_{R/L}{}^{T}\, \boldsymbol{S}_{R/L}\, \boldsymbol{K}_L{}^{-1} = \boldsymbol{K}_L{}^{-T}\, \boldsymbol{E}\, \boldsymbol{K}_R{}^{-1} \tag{3.7}$$

where:

$$\boldsymbol{S}_{R/L} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{3.8}$$

and T_x, T_y, T_z are the components of the translation vector $\mathbf{t}$ of camera right relative to camera left.

3.4.2 The Normalized 8-Point Algorithm

In this section, a methodology to compute the *Fundamental* matrix is introduced. This algorithm is known as the *Normalized 8-Piont Algorithm* suggested by Zisserman in his book [34]. The aforementioned algorithm is an optimization of the algorithm introduced by Louguent-Higgins in 1981 [37]. A summary of the algorithm is shown in Table 3 and it is described as follows.

Consider a set of a minimum of eight 2D points over the left image plane in its homogeneous form as $p_L = [u_L\ v_L\ 1]^T$ and their correspondence set of eight 2D points over the right image plane $p_R = [u_R\ v_R\ 1]^T$, such that:

$$p_R^T \boldsymbol{F} p_L = 0 \tag{3.9}$$

Then it is possible to apply a normalization of the image points such that the centroid of the reference points is at the origin coordinates and the root mean square (RMS) distance of the points from the origin is equal to $\sqrt{2}$. Thus, the translation of the 2D points is defined by:

$$T = \begin{bmatrix} \sigma & 0 & -\sigma\, mean(p_x) \\ 0 & \sigma & -\sigma\, mean(p_y) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

where: σ denotates the RMS-distance of the points from the origin. The normalized set of points are then defined as:

$$\hat{p}_L = Tp_L \,,\, \hat{p}_R = Tp_R \tag{3.11}$$

Substituting equation 3.11 into equation 3.9, it can be rewritten as:

$$[u_R\ v_R\ 1] \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{32} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} = 0 \tag{3.12}$$

Since it is stated that the Fundamental matrix is a singular matrix that has a $rank = 2$ and 7 degrees of freedom, one can force this by setting $F_{33} = 1$, such that equation 3.12 can be rewritten as:

$$\begin{bmatrix} u_{L1}u_{R1} & u_{L1}v_{R1} & u_{L1} & v_{L1}u_{R1} & v_{L1} & u_{R1} & v_{R1} \\ u_{L2}u_{R2} & u_{L2}v_{R2} & u_{L2} & v_{L2}u_{R2} & v_{L2} & u_{R2} & v_{R2} \\ u_{L3}u_{R3} & u_{L3}v_{R3} & u_{L3} & v_{L3}u_{R3} & v_{L3} & u_{R3} & v_{R3} \\ u_{L4}u_{R4} & u_{L4}v_{R4} & u_{L4} & v_{L4}u_{R4} & v_{L4} & u_{R4} & v_{R4} \\ u_{L5}u_{R5} & u_{L5}v_{R5} & u_{L5} & v_{L5}u_{R5} & v_{L5} & u_{R5} & v_{R5} \\ u_{L6}u_{R6} & u_{L6}v_{R6} & u_{L6} & v_{L6}u_{R6} & v_{L6} & u_{R6} & v_{R6} \\ u_{L7}u_{R7} & u_{L7}v_{R7} & u_{L7} & v_{L7}u_{R7} & v_{L7} & u_{R7} & v_{R7} \\ u_{L8}u_{R8} & u_{L8}v_{R8} & u_{L8} & v_{L8}u_{R8} & v_{L8} & u_{R8} & v_{R8} \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \end{bmatrix} = -\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.13}$$

Equation 3.13 has the form of a linear least-squares problem $Ax = b$, which can be easily solved by linear algebra or by applying *Singular Value Decomposition* (SVD) to A, thus:

$$svd(A) = [U\, \Sigma\, V^T] \tag{3.14}$$

where the solution of $\hat{F}$ lies over the null space (the last column) of the matrix V^T. Now one can enforce the $\det(\hat{F}) = 0$ by applying once again the SVD of $\hat{F}$. Giving as result $\widehat{F'} = \hat{S}\, \hat{\Sigma}\, \hat{V}^T$, then the last step is to denormalize the $\widehat{F'}$ by applying the inverse transformation of the input set of points, thus:

$$\boldsymbol{F} = T_R^T \, \widehat{F}' \, T_L \tag{3.15}$$

Table 5. Algorithm 1. The Normalized 8-Point Algorithm

Algorithm 1: The Normalized 8-Point Algorithm	
Given a set of $n \geq 8$ stereo correspondence image points $\{p_L \leftrightarrow p_R\}$, we can determine the Fundamental matrix $\boldsymbol{F}$, such that it meets the epipolar constraint denotated by $p_R^T \boldsymbol{F} p_L = 0$.	
i.	*Do Normalization:* Compute the transformation of the set of points such that: $\hat{p}_L = T_L p_L$ and $\hat{p}_R = T_R p_R$, where T_L and T_R denotate the normalizing transformation of its corresponding input points.
ii.	*Compute the Fundamental Matrix* $\hat{F}$: Find the solution of equation the linear system $\mathrm{Ax} = \mathrm{b}$, by applying SVD to A.
iii.	*Constraint:* force the $\det(\hat{F}) = 0$, then apply again SVD thus, $\widehat{F}' = \hat{S} \, \hat{\Sigma} \, \hat{V}^T$.
iv.	*Do Denormalization:* Compute $\boldsymbol{F} = T_R^T \, \widehat{F}' \, T_L$, fiving as result the Fundamental Matrix corresponding to the original input stereo correspondences $\{p_L \leftrightarrow p_R\}$.

3.4.3 The 5-Point Algorithm

As it was described in the previous section, the Fundamental Matrix encodes the epipolar constraint and it is useful to map a point extracted from the right image plane of the stereo system into the left image plane. Furthermore, the Fundamental Matrix can be computed without knowledge of the calibrated cameras $\boldsymbol{K}_L$ and $\boldsymbol{K}_R$, which constraints the system up to an unknown scale factor. However, if the system is calibrated and the calibration matrices are known, the epipolar constraint can be simplified to equation 3.5, which contains the named *Essential Matrix*, and as previously mentioned, also encodes the epipolar constraint of a stereo system.

The direct form to compute the Essential Matrix was already discussed in section 3.4.1, where the computation of the Essential Matrix is straightforward provided the Fundamental Matrix is known.

In 2004, Nistér proposed the *5-Point Algorithm* [35]. In the same fashion as the 8-Point Algorithm, the goal of the *5-Point Algorithm* is to compute the Essential Matrix. The main advantage of computing the Essential Matrix directly, instead of computing the Fundamental Matrix, is saving on computational time and it is described as follow.

Consider a set of five stereo corresponding points denoted as p_L and p_R. Each of these correspondences produce a constraint in the form:

$$p_R^T \boldsymbol{E} p_L = 0 \tag{3.16}$$

which can be rewritten as:

$$p^T \boldsymbol{E} = 0 \tag{3.17}$$

where:

$$p \equiv [p_{L1}p_{R1}\ p_{L2}p_{R1}\ p_{L3}p_{R1}\ p_{L1}p_{R2}\ p_{L2}p_{R2}\ p_{L3}p_{R2}\ p_{L1}p_{R3}\ p_{L2}p_{R3}\ p_{L3}p_{R3}]^T \tag{3.18}$$

$$\hat{E} \equiv [E_{11}\ E_{12}\ E_{13}\ E_{21}\ E_{22}\ E_{23}\ E_{31}\ E_{32}\ E_{33}] \tag{3.19}$$

By substituting equations 3.18 and 3.19 into 3.17, it yields a 5×9 matrix which just as equation 3.13 lies into a linear system of the form $Ax = 0$. The solution for $\hat{E}$ can then be easily computed by SVD, giving as result the Essential Matrix $\boldsymbol{E}$ that satisfies equation 3.16.

3.5 Summary

In this Chapter, the author has introduced an overview to the stereo system and its foundations. It also explained, the theory behind the Epipolar Geometry and the Epipolar

Constraint, which can easily be imposed by performing Image Rectification and calculating either the Fundamental and/or Essential Matrices. Camera calibration is the most important procedure to perform, since every single image that is being captured by the camera has to be rectified. In the following Chapter, the concept of scene reconstruction will be presented.

Chapter 4

Image Frame and Scene Reconstruction

As discussed in the previous Chapter, the *correspondence problem* is easily solved thanks to the constraint imposed by the Epipolar Geometry. In this Chapter, the author discusses how to find *features* over the image plane, also known as *corners* or *keypoints,* and how to build a *feature descriptor* so as to find its corresponding point into a new image frame from the same scene world taken from another position.

4.1 Features and Descriptors

The concept of *corners* is based on the belief that it would prove to be useful in myriad applications in its ability to represent an object of invariant form into an image, or several images of the same scene [38].

A *corner* is a small patch of an image that contains local information and it is likely to be recognized into another image.

A *keypoint* is an extension of the concept of corner. It contains information from a small local patch of an image and makes it highly recognizable into other images of the same scene.

The *descriptor* summarizes the information about a keypoint such that it makes it easier to recognize that patch of information into another image of the scene.

4.1.1 Harris Corner Detector

In recent decades, many different algorithms to detect and select points of interest in an image have been proposed. However, the Harris corner detector [8] introduced more than 30 years ago remains as the most popular algorithm to perform this task. Cited more than 16,000 times in the literature, the Harris corner detector appears in publications and applications that are used in myriad research fields and in a wide variety of industrial applications.

In a nutshell, the Harris corner detector algorithm responds to the energy function calculated for a target point (x, y) in the image plane, which considers average changes of image gradient intensity that results from shifting a small window, generally of 3×3 pixels, in all directions. It yields as results three cases to take into consideration, namely:

- *flat region*: where there is no change of gradient intensity in any direction;
- *edge region*: where there is no change over the edge direction; and
- *corner region*: where gradient intensity significantly changes in all directions.

In mathematical terms, we can express the specifications of the above by introducing the *energy function* as follows. Consider a grey scale image I with a small window patch defined as $(x, y) \in W$ which is being shifted $(\Delta u,\ \Delta v)$. The *sum of squared differences* (SDD) between the image and the window patch is given by the following equation:

$$E_{(\Delta u, \Delta v)} = \sum_{(x,y) \in W} (I(x, y) - I(x + \Delta u\ ,\ y\ + \Delta v))^2 \tag{4.1}$$

one can then take the right side of equation 4.1 and apply first-order Taylor series. Consider I_x and I_y as partial derivatives of I, then we produce the following approximation:

$$E_{(\Delta u, \Delta v)} \approx \sum_{(x,y) \in W} (I_x(x, y)\Delta u - I_y(x, y)\Delta v)^2 \tag{4.2}$$

equation 4.2 then can be expressed in matrix form as:

$$E_{(\Delta u,\Delta v)} = [\Delta u\ \Delta v]\ M\ [\Delta u\ \Delta v]^T \tag{4.3}$$

where M is called the *structure tensor,* defined as a 2×2 symmetric matrix, thus:

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{4.4}$$

Once the structure tensor has been built, it is necessary to calculate the eigenvalues λ_1 and λ_2 of M, thus:

- if both λ_1 and λ_2 are small, then $E_{(\Delta u,\Delta v)}$ is almost constant in all directions, thus the target point (x, y) is in a constant-intensity gradient region;
- if $\lambda_1 \gg \lambda_2$ or $\lambda_1 \ll \lambda_2$, then $E_{(\Delta u,\Delta v)}$ increases only in one direction, thus the target point (x, y) is on an edge region;
- if both λ_1 and λ_2 are high, then $E_{(\Delta u,\Delta v)}$ increases in all directions, thus the target point (x, y) lies in a corner region.

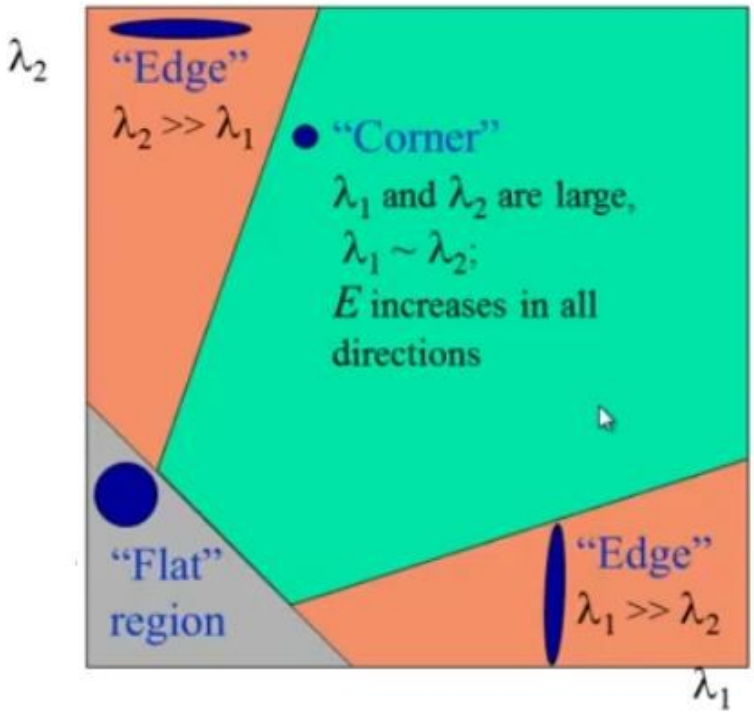


Figure 15. Classification of Harris corner detector [39].

There is another way to determine if the target point (x, y) lies in a corner region. This is done by building the *response function* of the system, which is defined as follows:

$$R = \det(M) - k\big(trace(M)\big)^2 \tag{4.5}$$

where k is an empirically determined constant, which recommended values lie in the range $[0.04, 0.06]$ and $trance(M) = m_{11} + m_{22}$. Since equations 4.4 and 4.5 are directly related, one can express equation 4.5 as:

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{4.6}$$

Once the response function R is calculated for each target point (x, y) over the image plane, one will have as results a cluster of *corner points*, which are then passed through a threshold and non-maximum suppression algorithm to ensure the best corner points. In addition, image patches around the corners are used as descriptors.

Even though the Harris corner detector is not computationally expensive, it has the advantage that it is rotation-invariant, but its disadvantage against other algorithms is that it is non-scale-invariant. This means that if we apply a rotation to the image, the corner detected will still be recognized in another image, but if we apply a scale transformation to the image, the corner might not be recognized.

Figure 16 shows the Harris corner features detected over a pair of stereo images. In the top, the stereo image pair are shown, where one can observe the chess board calibration pattern. The area of interest for corners search has been specified to be the area where the pattern is found. One can note how most of the features detected are over the corners of the calibration pattern, in both left and right images. In the bottom, a maximization of both left and right images is shown.

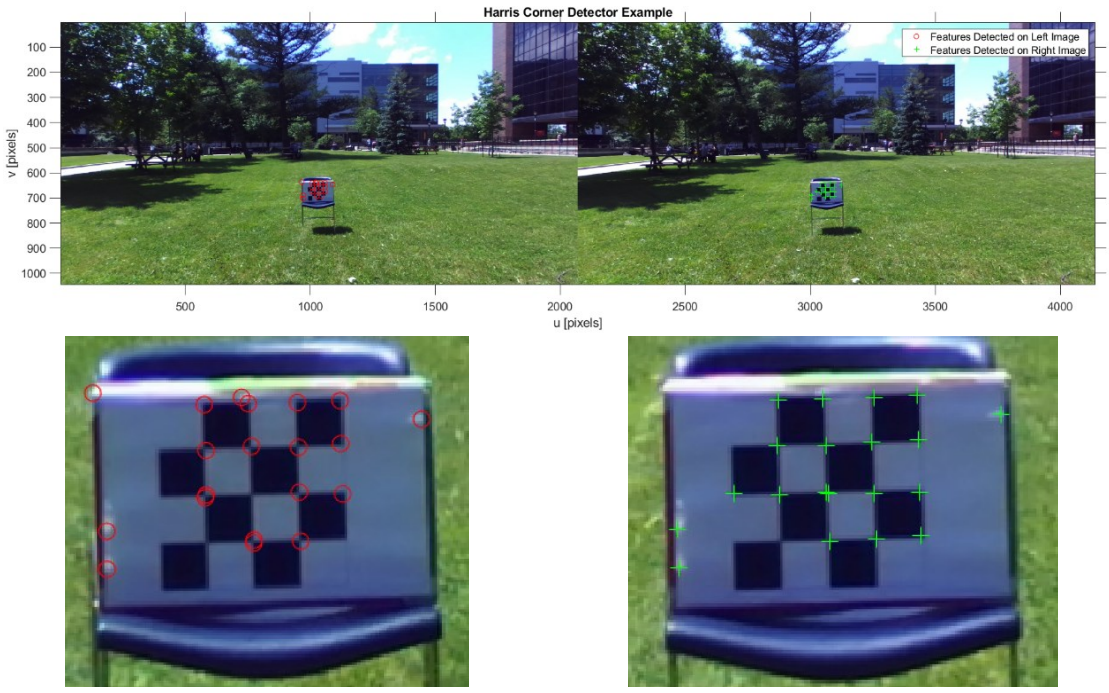


Figure 16. Features Detected Using Harris Corner Detector.

4.1.2 SURF Features and Descriptors

As it was mentioned in the previous section, the Harris corner detector is non-scale invariant. In 2004, Lowe proposed an algorithm to overcome the non-scale-invariant problem, known as the *Scale-Invariant Feature Transform* (SIFT) [40] that applied a *Difference of Gaussians* (DoG) filter across the discretized scale space. Once DoG are found, one pixel in the image is compared to its 8-pixel neighbors and the 9-pixel in the previous and next scale levels so as to find a local maxima gradient value. If a local maxima gradient value is found, the pixel is considered as a potential keypoint.

The location of the keypoints is then found by a refinement of the potential keypoints. To do so, Lowe proposed a Taylor series expansion over the scale space to get more good locations of the local maxima gradient, where if the value of the gradient is less than a

certain threshold, the pixel is rejected. Since DoG has a high response to edges, SIFT uses a similar method than the Harris corner detector to reject them. A 2×2 Hessian matrix (H) is used to compute its curvature and if the ratio is greater than a certain threshold, the pixel is discarded. Finally, the descriptor of the keypoints is defined; for each keypoint, a 16×16 window of pixels around the point of interest is taken and then divided into 16 sub-blocks, where for each sub-block, an 8-bin orientation histogram is created. As a result, 128-bin values represented as a vector form keypoint descriptor is created.

Even though SIFT has advantages such as being illumination-invariant, contrast-invariant, scale-invariant, and rotation-invariant, the algorithm is computationally expensive, which impacts directly on the image processing for applications like *Visual Odometry*.

For this reason, in 2006, Bay, Tuytelaars and Van Gool proposed an optimized algorithm called *Speed Up Robust Features* (SURF) [41], which, as its name claims, offers a faster version of SIFT.

In contrast to SIFT – which finds potential keypoints by applying a DoG filter over the scale-space of the image plane –, SURF introduced a box filter to find the local maxima gradient over the scale-space, which has the advantage that it can be done in parallel for different scales.

For most applications, the orientation of the potential keypoints that SIFT provides as extra information is not required. SURF provides this functionality, since the potential keypoints can be set as "upright" keypoints, speeding-up the process. The descriptor of the keypoints is calculated from a window of 20×20 pixels around the point of interest;

it is then divided into 4×4 subregions. SURF provides two different dimensions for descriptors 64 and 128; the lower the dimension, the higher the speed of computation.

SURF provides a robust set of points of interest around the image plane that are of high importance in the processing time for applications in *Visual Odometry*. In contrast to the Harris corner detector, SURF has the same advantages that SIFT features provide. However, accordingly to Bay, Tuytelaars and Van Gool, SURF performs up to 4 times faster than SIFT to compute the same amount of keypoints and descriptors.

Figure 17 shows an example of how SURF works. In the figure, one can observe the same pair of stereo images over the same window of search area. Note how the features detected are not exactly the corners of the calibration pattern, because of the information included on the descriptors for each image.

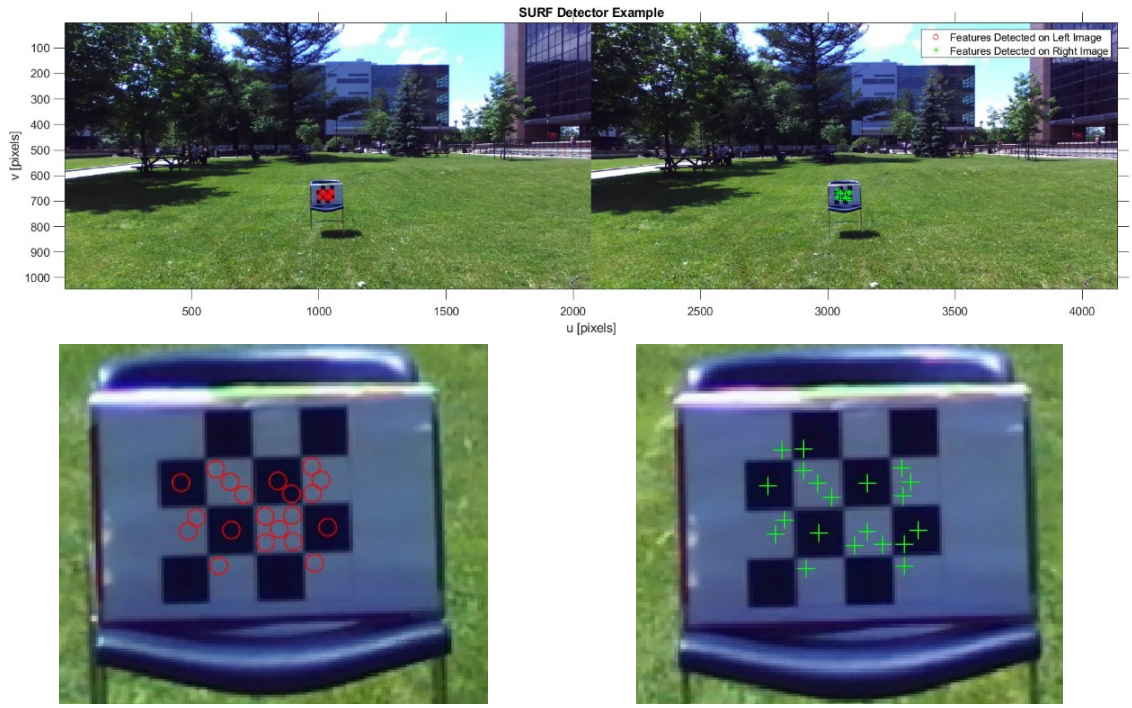


Figure 17. Features Detected Using SURF.

4.2 Matching Keypoints on Stereo Images

In the previous section, the author has discussed methodologies to extract *keypoints* and their *descriptor* over the image plane. In this section, methodologies to match those features are introduced; these are of high importance for *scene reconstruction* and further developed in section 4.4.

In the literature, there are two main methodologies to match features, the most popular one being the matching of the extracted keypoints utilizing their descriptor information, and the alternative methodology being the tracking of the keypoints over a second image. It is of high importance to take into consideration the epipolar geometry and image rectification, which were previously discussed in sections 3.1 and 3.4.

4.2.1 Matching Features

For the purposes and scope of this thesis, the word '*matching*' will refer to the task of searching and finding correspondences of keypoints over the left and right image planes of the stereo system at a given instant of time. Furthermore, it will be assumed that either the *Fundamental* or *Essential* matrices are known to remove *outliers*, as it will be discussed in more detail in the following section.

Additionally, the baseline of the stereo system is considered to be relatively small, and the orientation relative to both cameras is assumed to be approximately the same. The stereo system is assumed to be calibrated and the input process images are rectified. These assumptions are required in order to facilitate the matching features process.

Table 6. Distance Metrics for the Brute Force Algorithm Matcher.

Metric Distance	***Function Equation***
NORM_L2	$dist(\vec{a}, \vec{b}) = \sqrt{\sum_i (a_i - b_i)^2}$
NORM_L2SQR	$dist(\vec{a}, \vec{b}) = \sum_i (a_i - b_i)^2$
NORM_L1	$dist(\vec{a}, \vec{b}) = \sum_i abs(a_i - b_i)$
NORM_HAMMING	$dist(\vec{a}, \vec{b}) = \sum_i (a_i == b_i)?\,1:0$
NORM_HAMMING2	$dist(\vec{a}, \vec{b}) = \sum_i [(a_i == b_i)\&\&(a_{i+1} == b_{i+1})]\,?\,1:0$

The first and most common algorithm for matching features is the one called *Brute Force* algorithm. The *Brute Force* matcher algorithm implements a straightforward solution to match the extracted keypoints. It takes the descriptor of keypoints in the left image $\vec{a}$ and attempts to match it with the descriptor of keypoints from the right image $\vec{b}$ based on their norm metric distance, which depending on the type of feature and descriptor (e.g. Harris, SIFT, SURF), can be specified from one of the listed in Table 6.

Figure 18 and 19 show examples of features detected and matched by using the *Brute Force* algorithm. In Figure 18 shows the matching points detected using the Harris corner detector showed previously in Figure 16. Since corners are built with a binary descriptor, the *Brute Force* algorithm performs the $norm\ L_1$ to match the keypoints. It is observed that most of the keypoints detected and matched remain over the epipolar constraint, but also keypoints outside the epipolar constraint are present.

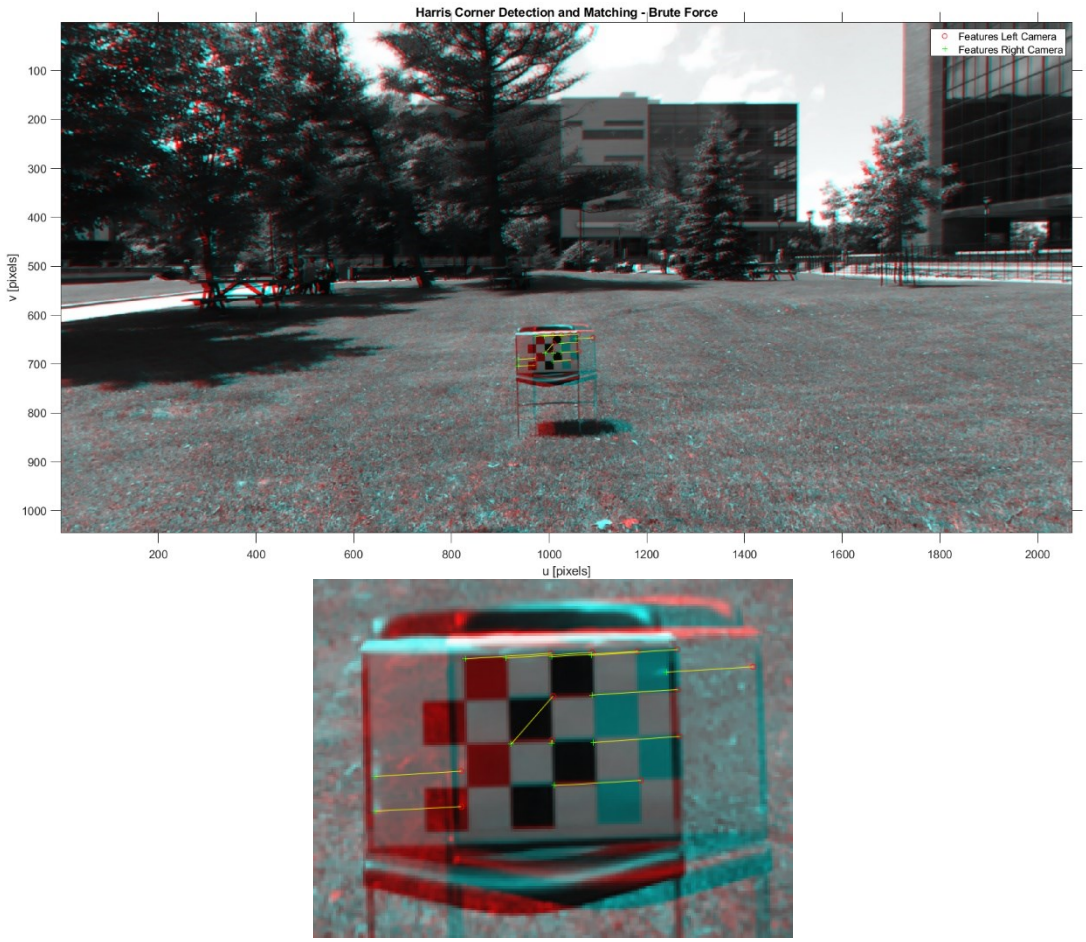


Figure 18. Harris Corner Detection and Matching by Brute Force.

On the other hand, Figure 19 shows the matched SURF keypoints detected previously (Figure 17). When SURF keypoints are used, the *Brute Force* algorithm performs the $norm\ L_2$ to match the descriptors of the keypoints available. In the same way as the matched Harris keypoints, the presence of keypoints outside the epipolar constraint is observed. The keypoints that remain outside the epipolar constraint can be removed to avoid miscalculations as it is going to be discussed in Section 4.3.

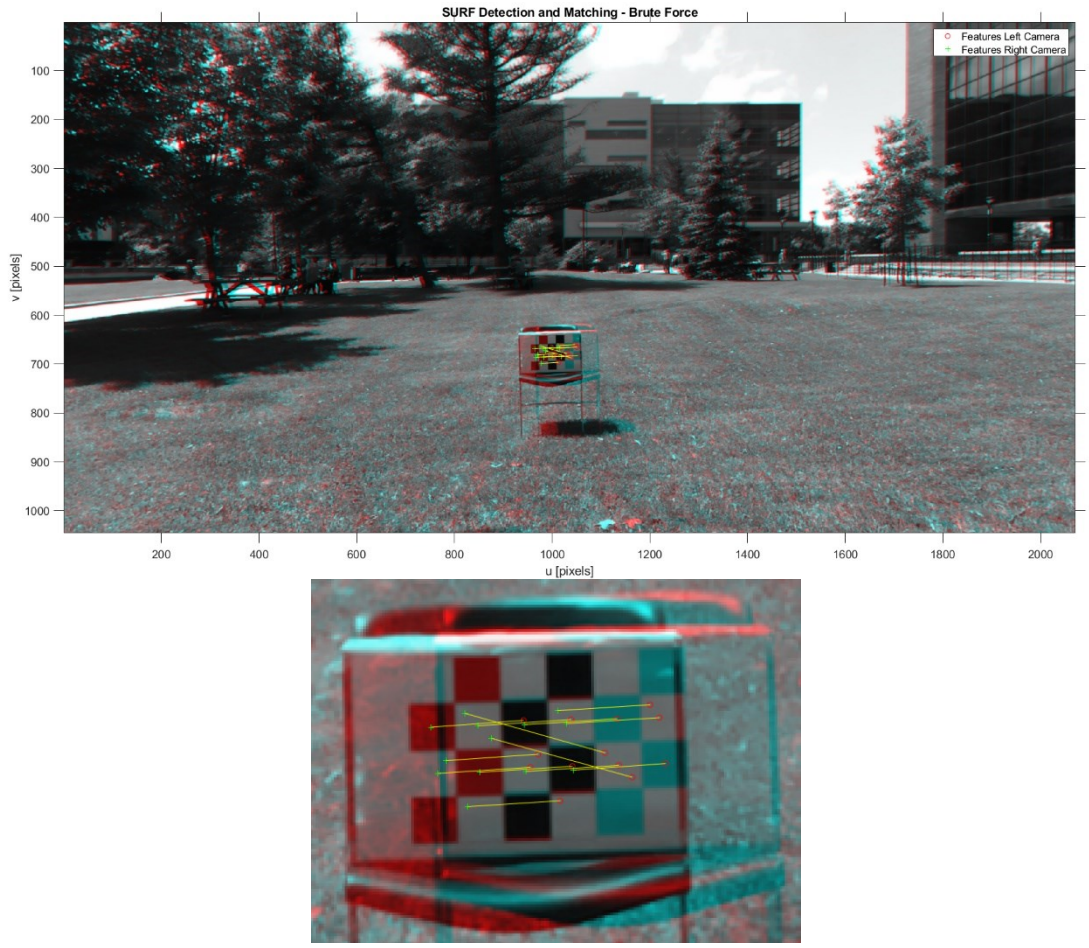


Figure 19. SURF Detection and Matching by Brute Force.

The second algorithm is the so-called *Fast Library for Approximate Nearest Neighbor* (FLANN) computation, which was introduced by Muja and Lowe in 2009 [42]. FLANN implements an optimized library of solvers for the *Nearest Neighbor* problem algorithms first proposed by Friedman and Bentley in 1977 [43].

In similarity to *Brute Force,* FLANN considers the descriptor of keypoints of image left and the descriptor of keypoints of image right by searching for the best match around the nearest neighbor set of pixels around the point of interest through a *k-d tree*, which is a binary multilevel tree where each note represents a subfile of the query descriptors. Then,

the Euclidean distance between the descriptor vector is calculated. Finally, a good match is detected if it meets the condition proposed by Lowe, expressed in equation 4.7.

$$Lowe's\ Condition = \begin{cases} if\ dist(\vec{a}, \vec{b}) \leq \vec{a} - 0.7\vec{b} & Good\ Match \\ if\ dist(\vec{a}, \vec{b}) > \vec{a} - 0.7\vec{b} & Reject \end{cases} \quad (4.7)$$

The Brute Force algorithm works well for small sets of keypoints, but it has a clear tendency to be slower when the quantity of features increases. However, FLANN performs faster than Brute Force even with a large set of points of interest.

In a few words, the process of matching features on stereo image pairs can be summarized as follows:

1. Identify keypoints over both left and right image planes.
2. Build the descriptor for the identified keypoints.
3. Use FLANN matcher algorithm.
4. Reject mismatches using equation 4.7.

Figure 20 and 21 show examples of features matched using the FLANN algorithm. Figure 20 shows the matching features from the same keypoints presented in Figure 16 (the *Brute Force* algorithm). Notice how the presence of keypoints which are outside the epipolar constraint are present once again.

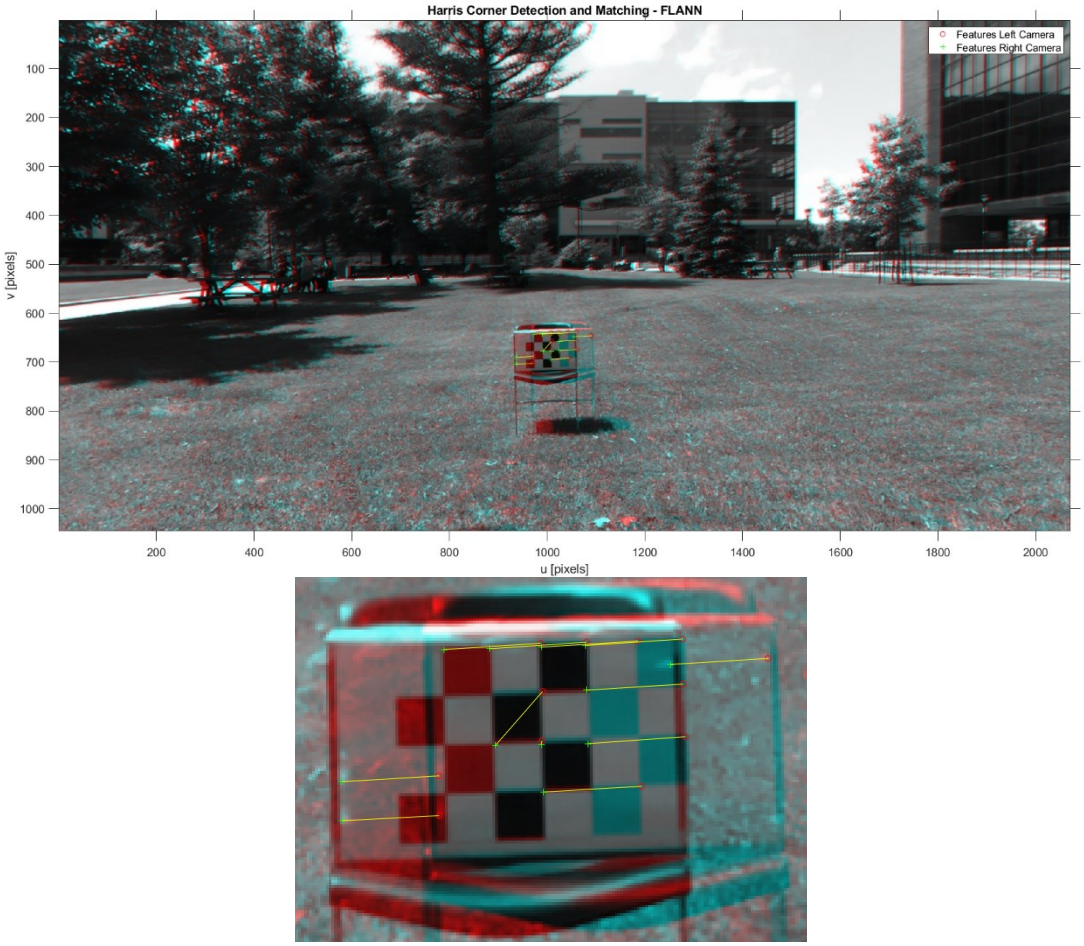


Figure 20. Harris Corner Detection and Matching by FLANN.

On the other hand, Figure 21 shows the matched SURF keypoints presented in Figure 17. Similarly, to the matched Harris keypoints, notice the presence of keypoints outside the epipolar constraint.

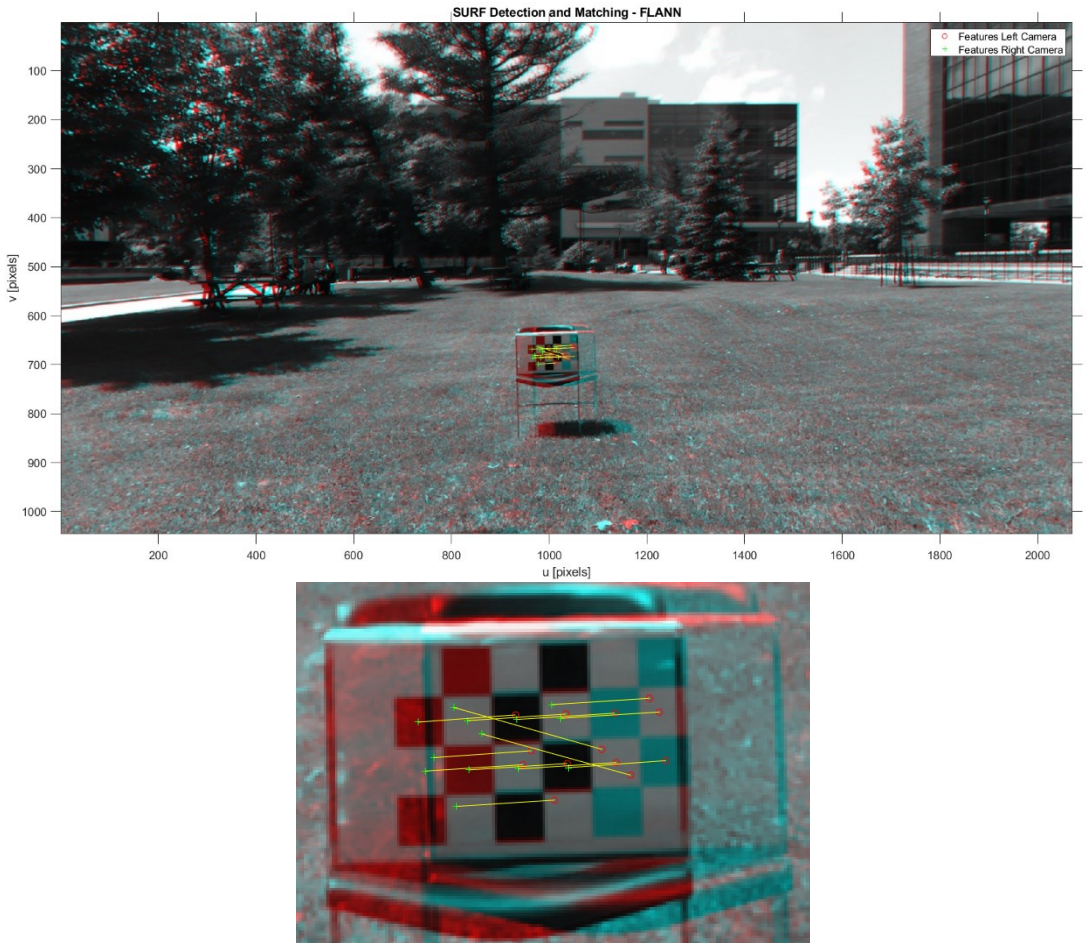


Figure 21. SURF Detection and Matching by FLANN.

4.2.2 Tracking Features

As mentioned at the beginning of this Chapter, the second methodology to match features over a stereo image pair is to track them. Whereas *matching features* utilizes the descriptor information from the keypoints extracted from the stereo images, the method of *tracking features* considers the gradient intensity of the features found in an image and looks for its correspondence feature with the same gradient intensity in a second image.

The tracking algorithm used in this thesis was introduced by Lucas and Kanade in 1981 [44], where they introduced the concept of *optical flow.* In their work, the authors

associate a movement vector (u, v) to every interesting pixel in the scene obtained by comparing two consecutive images: as for the purposes of this work, stereo image pairs.

The *Lucas-Kanade algorithm* considers the following assumptions:

- The two input images are separate by a small-time increment Δt, in such a way that the objects have not a significant displacement.
- The input images describe a scene of objects of contrasting textures in different shades of grays (gray gradient scales) which changes smoothly over time.

In contrast to the methods previously discussed (Brute Force and FLANN), the *Lucas-Kanade algorithm* does not scan the whole second input image to look for the best matching gradient intensity that corresponds to the first input gradient. It takes the gradient to match from the first input image and guesses in which direction the keypoint has moved, such that the local changes in gradient intensity can be extracted over a small window of pixels, generally in the range [15,31] pixels.

The algorithm can be explained as follows [45]. Assume that there are two input images from the same scene taken at a relative small instant of time and that one can consider the intensity of the gray scale of a point of interest (x, y) over the first input image denotated by a, such that in the second image the intensity of the pixels has increased to b.

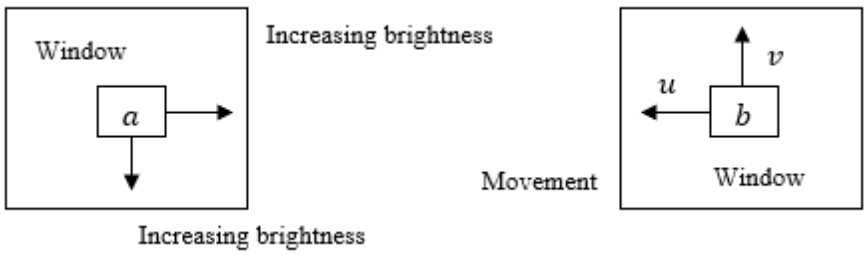


Figure 22. Change of Brightness Intensity of an Interest Point.

If it is assumed that the increase in brightness per pixel at the point of interest pixel (x, y) is $I_x(x, y)$ in the x-direction, and the increase of brightness per pixel in the y-direction is $I_y(x, y)$, as shown in figure 22, one can notice a total increase in brightness after the movement of u pixels in the x-direction and v pixels in the y-direction, thus:

$$I_x(x, y) \cdot u + I_y(x, y) \cdot v$$

which matches the local difference in intensity $(b - a)$ defined as $I_t(x, y)$, such that:

$$I_x(x, y) \cdot u + I_y(x, y) \cdot v = -I_t(x, y) \tag{4.8}$$

Since a simple pixel usually does not contain enough information for matching with another pixel, the algorithm utilizes a window of 3×3 neighborhood pixels around the point of interest (x, y), such that we can set 9 linear equations of the form:

$$I_x(x + \Delta x, y + \Delta y) \cdot u + I_y(x + \Delta x, y + \Delta y) \cdot v = -I_t(x + \Delta x, y + \Delta y)$$

for $\Delta x = -1, 0, 1$ and $\Delta y = -1, 0, 1$, and the equations can be summarized in a matrix form, thus:

$$S \begin{pmatrix} u \\ v \end{pmatrix} = \vec{t} \tag{4.9}$$

where S is a 9×2 matrix containing the rows $\left(I_x(x + \Delta x, y + \Delta y),\ I_y(x + \Delta x, y + \Delta y)\right)$ and $\vec{t}$ is a vector which contains the 9 terms $-I_t(x + \Delta x, y + \Delta y)$.

The exact solution for equation 4.9 can be found by a Least-Square approximation and its multiplication by S^T:

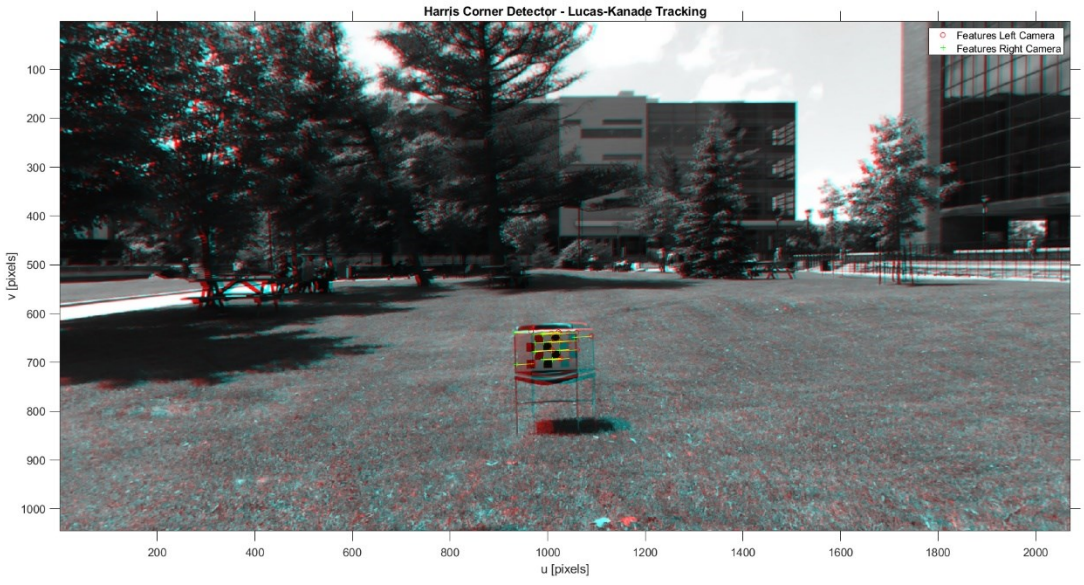
$$S^T S \begin{pmatrix} u \\ v \end{pmatrix} = S^T \vec{t} \tag{4.10}$$

and leaving the point of interest coordinate pixel alone, equation 4.10 is written as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = (S^T S)^{-1} S^T \vec{t} \tag{4.11}$$

The solution of equation 4.11 is the final guess or the corresponding point to the input point of interest. The disadvantage of this algorithm is that in order to get a correspondence point, there must be slightly the same intensity in the second image. On the other hand, the main advantage is that for a neighborhood of a fixed size, which is the case in most of the applications, the number of operations needed to compute equation 4.11 is constant, and therefore, the algorithm is linear in the number of pixels being examined.

In a nutshell, the *Lucas-Kanade algorithm* tracks a template of keypoints detected in a given image and looks for the best match over a second image of the same scene. In Figure 23 shows how the Harris corners detected in Figure 16 are tracked over the right image. In Figure 24 one can observe the same methodology but using the SURF features presented in Figure 17.



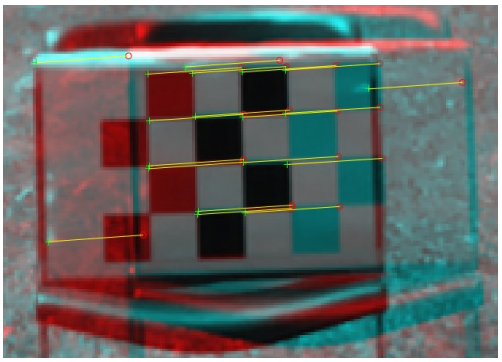


Figure 23. Lucas-Kanade Tracker using Harris Corner Keypoints.

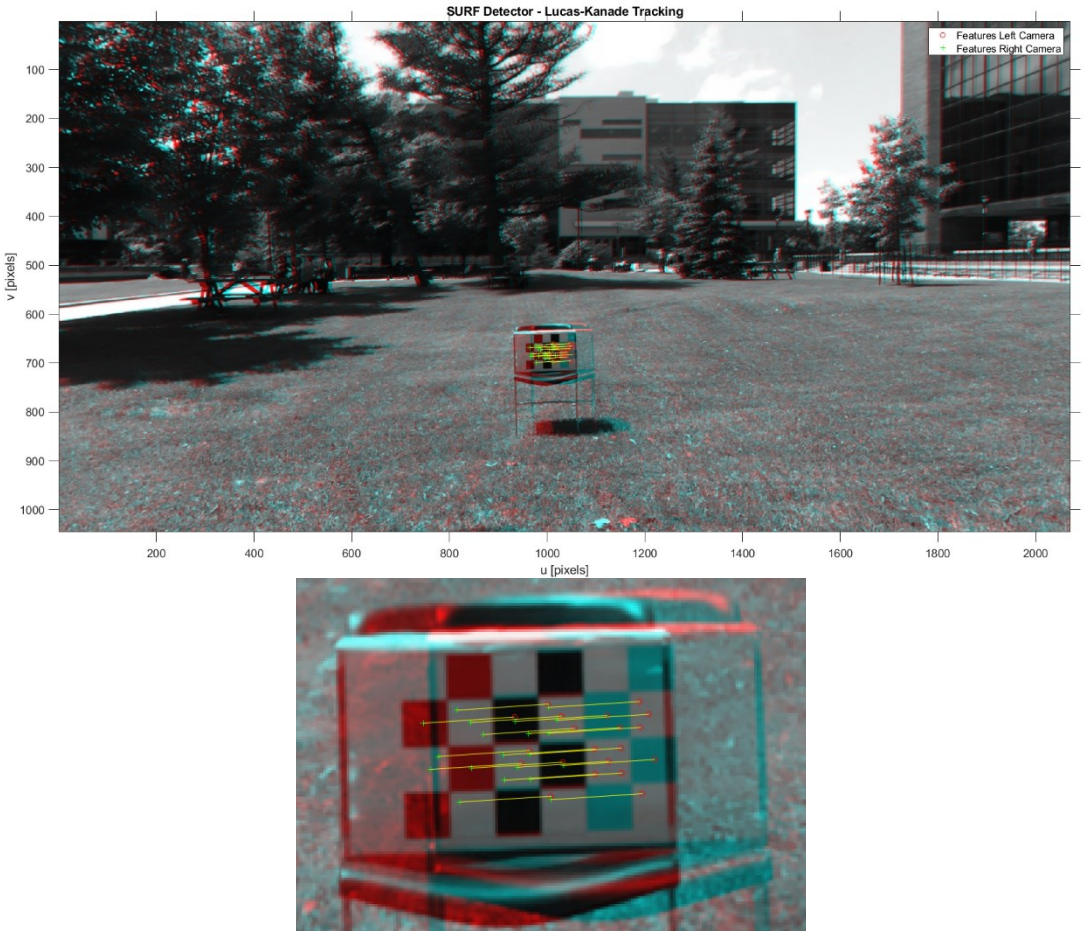


Figure 24. Lucas-Kanade Tracker using SURF Keypoints.

4.3 Removing Outliers

As it has been discussed in sections 4.1 and 4.2, visual odometry systems rely on keypoints of high confidence over the image plane. However, even though techniques such as *matching features* and *tracking features* work over the constraint imposed by the Epipolar Geometry, in most of the cases, not all the matching keypoints might lie over the aforementioned constraint.

The keypoints which lie outside the epipolar constraint are called *outliers* and it is of high importance to remove them from the set of keypoints. Even though the matching keypoints might not be that far from their corresponding epipolar line, outliers can cause drift and deviations on the output of the general Visual Odometry system.

In 1981, Fishcler and Bolles [9] introduced a method to solve the Location Determination Problem (LDP). The method is known as *Random Sample And Consensus* (RANSAC). The goal of RANSAC is to estimate the parameters of a model (in this case, the *Epipolar Constraint)* starting from a set of data contaminated by large amounts of *outliers*, considered as the set of matching points extracted from one of the techniques previously discussed.

During past years there have been modifications that have attempted to optimize RANSAC [36]. However, the main algorithm is composed by two steps that repeat in an iterative fashion, namely the *hypothesize* and *test framework* steps.

The *hypothesize* step computes the model from a randomly selected sample from the input data. Once the model is built, the algorithm checks the whole input dataset against the built model in the *test* step. The set of elements that fit the model are known as *consensus set,* and the set of elements that do not fit the model are called *outliers*.

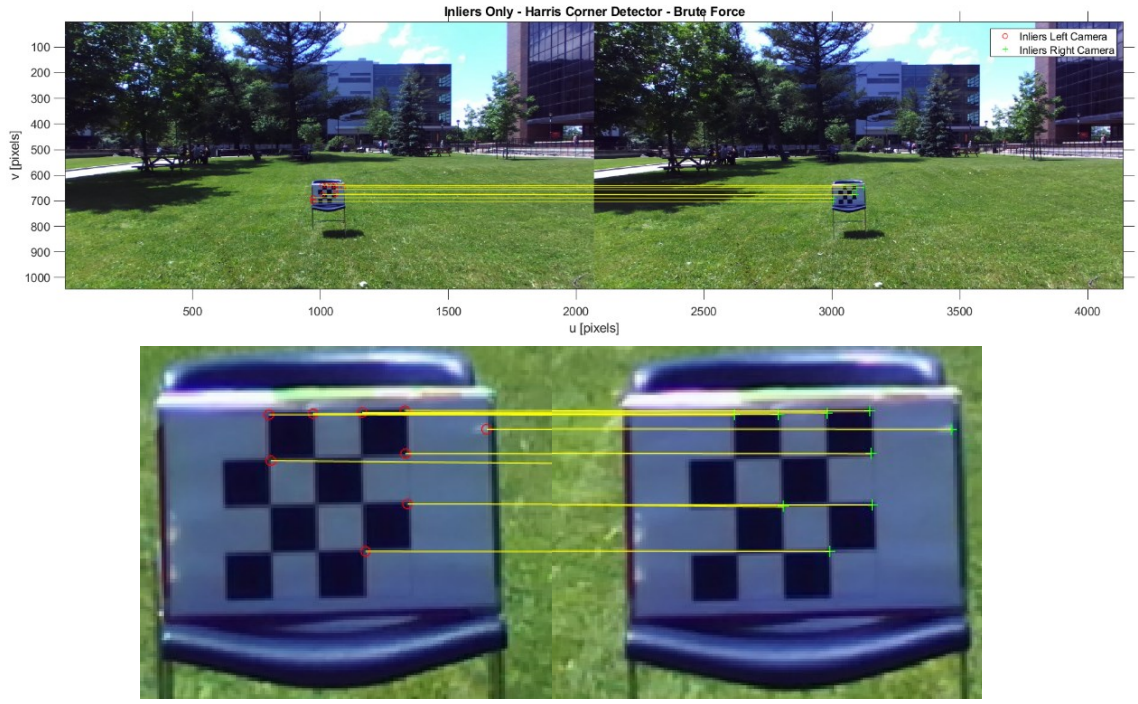
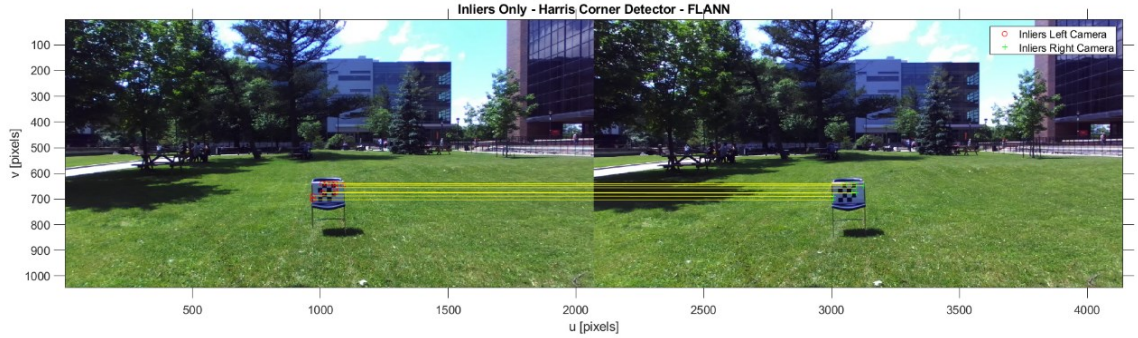


Figure 25. Inliers on a Stereo Image Pair - Harris Corners Matched with Brute Force.

Figure 25 shows how the RANSAC algorithm in combination with the 5-Point algorithm has removed the erroneous mismatched keypoints observed in Figure 18, leaving only *inlier* keypoints over the epipolar lines. In the same fashion, Figure 26 shows the *inlier* Harris corners presented in Figure 20 when the corners were matched using FLANN.



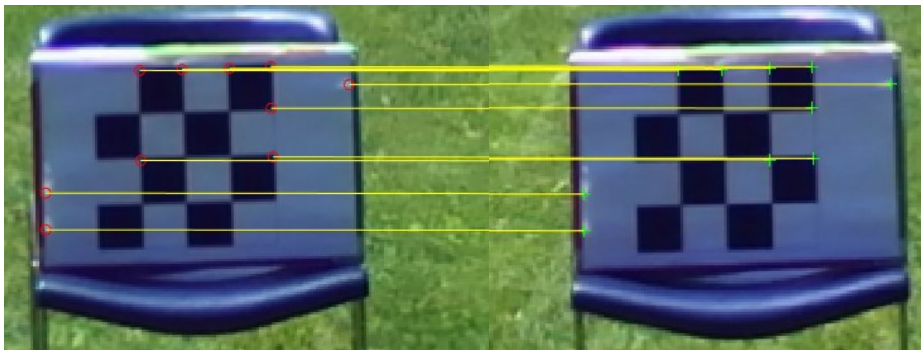


Figure 26. Inliers on a Stereo Image Pair - Harris Corners Matched with FLANN.

Figure 27 and 28 show the *inlier* SURF keypoints that were matched using *Brute Force* algorithm (Figure 19) and FLANN (Figure 21), respectively.

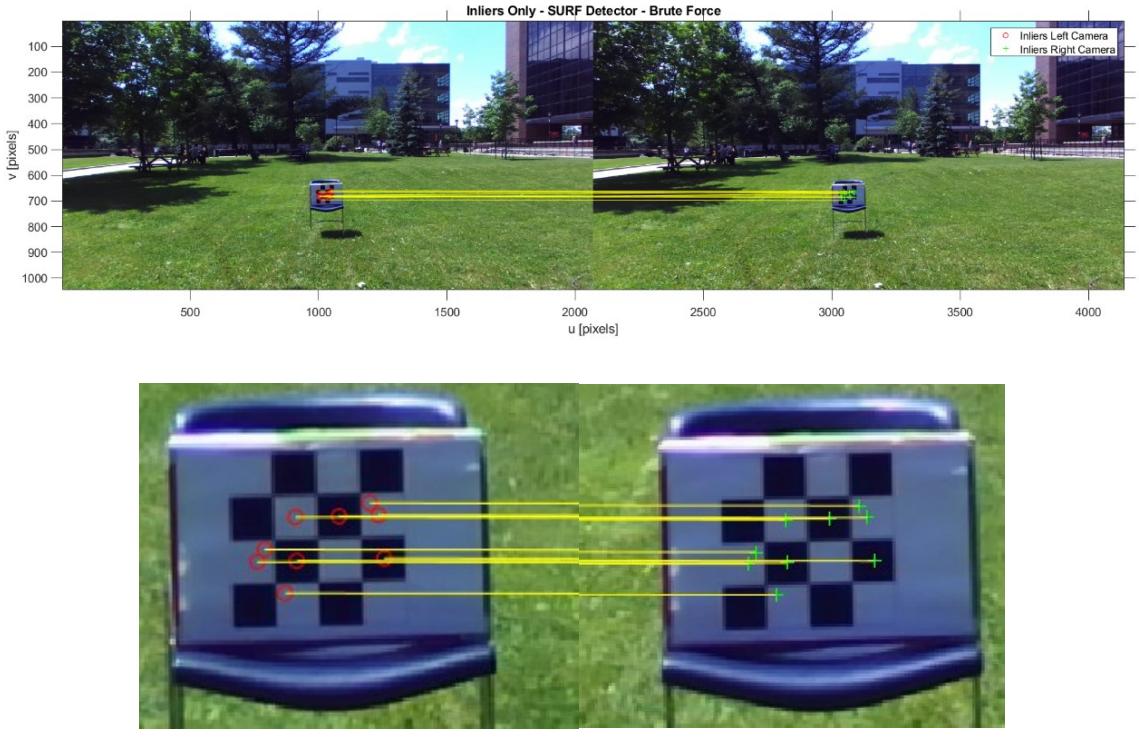


Figure 27. Inliers on a Stereo Image Pair – SURF Keypoints Matched with Brute Force.

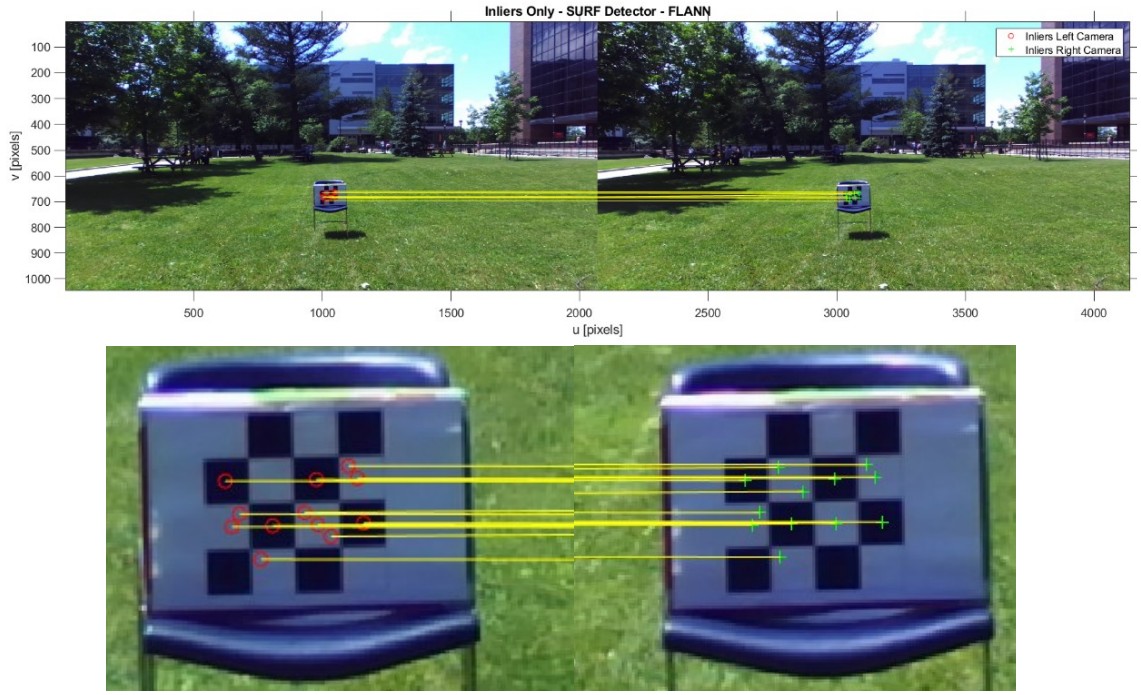


Figure 28. Inliers on a Stereo Image Pair – SURF Keypoints Matched with FLANN.

Notice how the *inliers* show a straight horizontal line so that the keypoints lie over their corresponding *epipolar lines.* This ensures that the set of keypoints extracted from the image plane indeed are outliers free.

4.4 Scene Reconstruction

In the literature, the concept of *scene reconstruction* refers to the process of converting information from the image plane described in 2D features into a 3D structure. To do so, 2D keypoints are selected from the input image, then matched with other set of 2D keypoints, and finally transformed into a 3D structure, whereby depth estimation with respect to the camera system can be extracted as shown in Figure 29. An example of dense scene reconstruction can de observed.

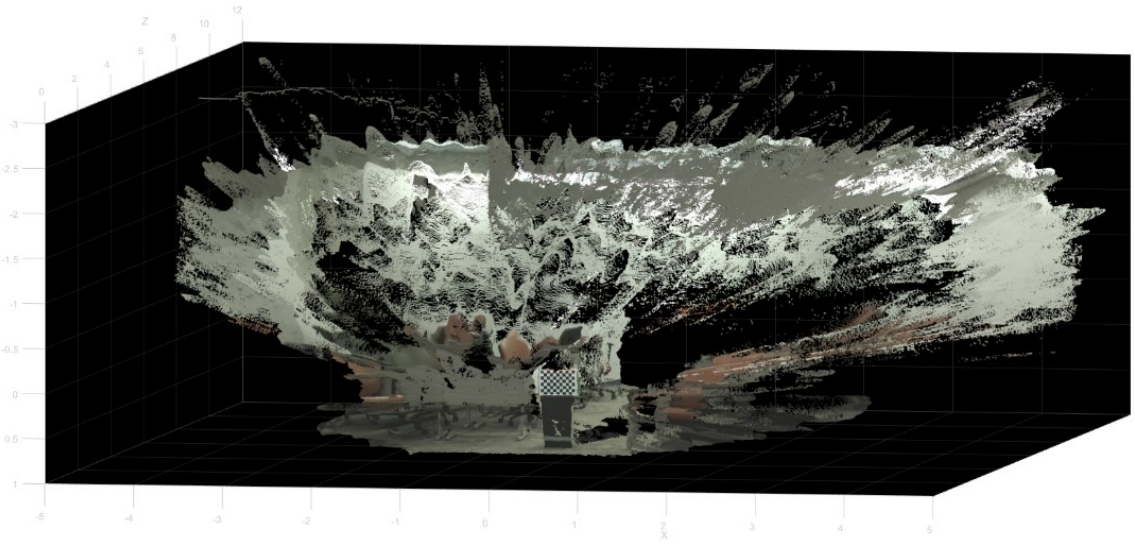


Figure 29. Dense Scene Reconstruction.

In this section, three methodologies to estimate depth from a pair of stereo images are discussed. The first method is the so-called *Direct Linear Transformation* (DLT); the second one is the *Semi-Global Matching* method; and the third method is the *Direct Disparity* one, which is a simplification of the Semi-Global Matching method.

4.4.1 Direct Linear Transformation

The *Direct Linear Transformation* (DLT) method [34] is an algorithm that solves a group of variables from a set of similar relations and it is described as follows.

Consider a set of keypoints over the image plane extracted utilizing one of the techniques described in section 4.1, such that p_L and p_R represent a set of corresponding keypoints in the left and the right images of the stereo image pairs.

Recalling the Epipolar Geometry described in Figure 2, and using the set of correspondence keypoints, we can express the correspondence image points as:

$$p_L = \boldsymbol{M}_L\, P\ ,\ p_R = \boldsymbol{M}_R\, P \tag{4.12}$$

where the matrices $\boldsymbol{M}_L$ and $\boldsymbol{M}_R$ are the projection matrices from equation 3.2 and P is the projected keypoint in a 3D Euclidean space. One can combine the two equations and express them into the form of $A\ P = 0$, which is an equation linear in P and can be easily solved by *Singular Value Decomposition* (SVD).

To do so, correspondence set of keypoints p_L and p_R must be expressed. Also, their corresponding 3D projection P into *homogeneous* form can be determined, thus:

$$\tilde{p}_L = [u\ v\ 1]^T\ ,\ \tilde{p}_R = [u\ v\ 1]^T\ , P = [X\ Y\ Z\ 1]^T \tag{4.13}$$

where the homogeneous scale factor can be easily eliminated by the cross product of the set of keypoints and the product of the projection matrices by the projected points, in the form $\tilde{p}_L \times (\boldsymbol{M}_L\, P) = 0$ and $\tilde{p}_R \times (\boldsymbol{M_R}\, P) = 0$, giving as result three equations for each keypoint, where two of those equations are linearly independent, thus:

$$\begin{gathered} u(\tilde{m}^{3T}\, P) - (\tilde{m}^{1T}\, P) = 0 \\ v(\tilde{m}^{3T}\, P) - (\tilde{m}^{2T}\, P) = 0 \\ u(\tilde{m}^{1T}\, P) - v(\tilde{m}^{2T}\, P) = 0 \end{gathered} \tag{4.14}$$

where $\tilde{m}^{iT}$ represent the rows of the projection matrices $\boldsymbol{M}_L$ and $\boldsymbol{M}_R$. We can observe that these equations are *linear* in the components of P. Then, we can compose a linear system of the form $A\ P = 0$, such that:

$$A = \begin{bmatrix} u\tilde{p}_L^{3T} & - & \tilde{p}_L^{1T} \\ v\tilde{p}_L^{3T} & - & \tilde{p}_L^{2T} \\ u\tilde{p}_R^{3T} & - & \tilde{p}_R^{1T} \\ v\tilde{p}_R^{3T} & - & \tilde{p}_R^{2T} \end{bmatrix} \tag{4.15}$$

where two equations have been included from each image, giving a total of four equations and four homogeneous unknowns. Note that equation 4.15 is redundant, since the solution is up to scale.

By applying SVD to equation 4.15, the matrix A can be decomposed into the form of $A = [U\,\Sigma\,V^T]$, and by extracting from the null-space (the last column) of matrix V^T, the vector of the form $[v_1\ v_2\ v_3\ v_4]^T$ can be considered as the solution, where the aforementioned scale factor is the last element v_4. It is just a matter of normalizing the vector dividing it by its last element, which yields to the 3D coordinates of P with respect to the camera frame.

4.4.2 Semi-Global Matching

In the previous section, the *Direct Linear Transformation* (DLT) method has been discussed, which is able to project a set of corresponding keypoints into a 3D Euclidean space, where each point is calculated separately. In order to generate full scene reconstruction (dense scene reconstruction), more sophisticated algorithms must be employed.

The *Semi-Global Matching* (SGM) algorithm introduced by Hirschmüller in 2008 [46] tackles this problem. The algorithm aims to build a *Depth Map* or *Disparity Map* based on two stereo images from the scene by looking for pixel gradient intensities along the epipolar lines. As such, it considers as input a pixel in the left image I_L inside a small window and looks over its correspondent epipolar line on the right image I_R, until the right match is found. Figure 30 shows how the algorithm takes the information inside the

window on the left image and searches for the correspondent corner over its correspondence epipolar line on the right image.

In order to perform the SGM algorithm, some assumptions have to be made. The input image pair must be already rectified, and the intrinsic and extrinsic parameters of the system must be known. The input images must have the same size, and both must be processed in gray scale [47].

Even when the window in the left image might contain more information than the one found in the right image (as shown in Figure 30) the algorithm can still match the right corner. If the pixel intensity is found in both images, it yields the *Disparity* value (distance between the two pixels) corresponding to the query pixels; otherwise, the disparity is set as either indeterminate or infinity.

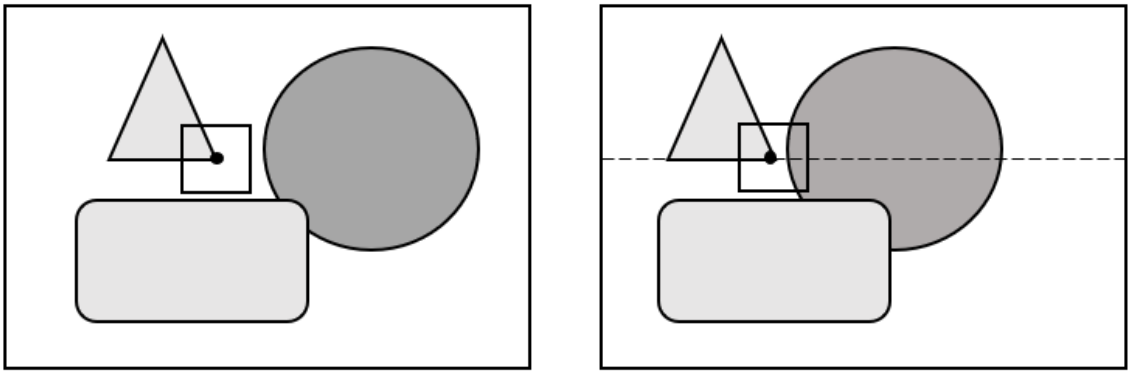


Figure 30. Correlation Based Stereo Matching.

In the previous years, there have been numerus contributions to the dense correspondence problem. Some of the most important contributions come from the University of Middlebury. Figure 31 shows an example of one of their datasets, where the

image shows the disparity map (right image) corresponding to the image in the left side. The disparity map is presented in a form of an RGB color map, where Red denotates proximity to the camera and Blue represents objects and point far away from the camera.

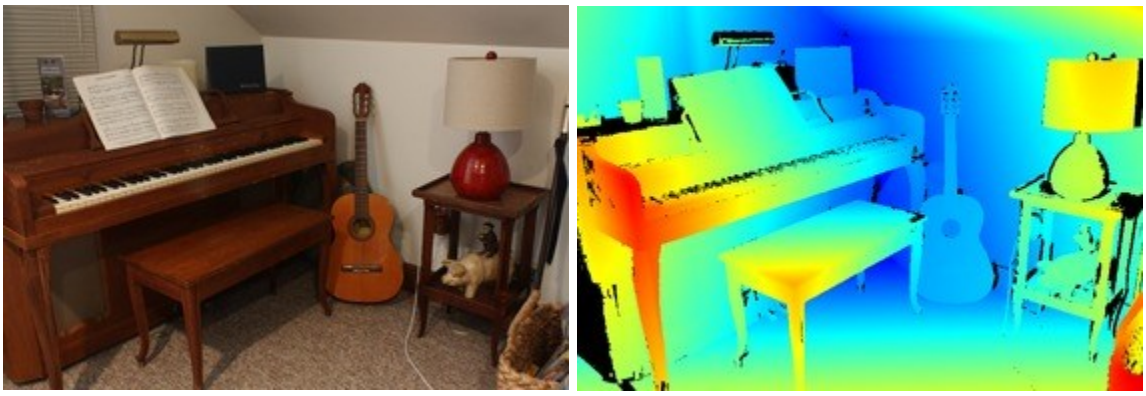


Figure 31. Left: Source Piano Image. Right: Disparity Map for Piano Image [48]**.**

After computing the *Disparity Map* as shown in Figure 32, where the *Red* color on the color bar represents the closets points to the camera and *Blue* denotates the farthest points from the camera, the rectified stereo geometry or epipolar constraint can be used to compute the coordinates in 3D Euclidean space of a set of interest points.

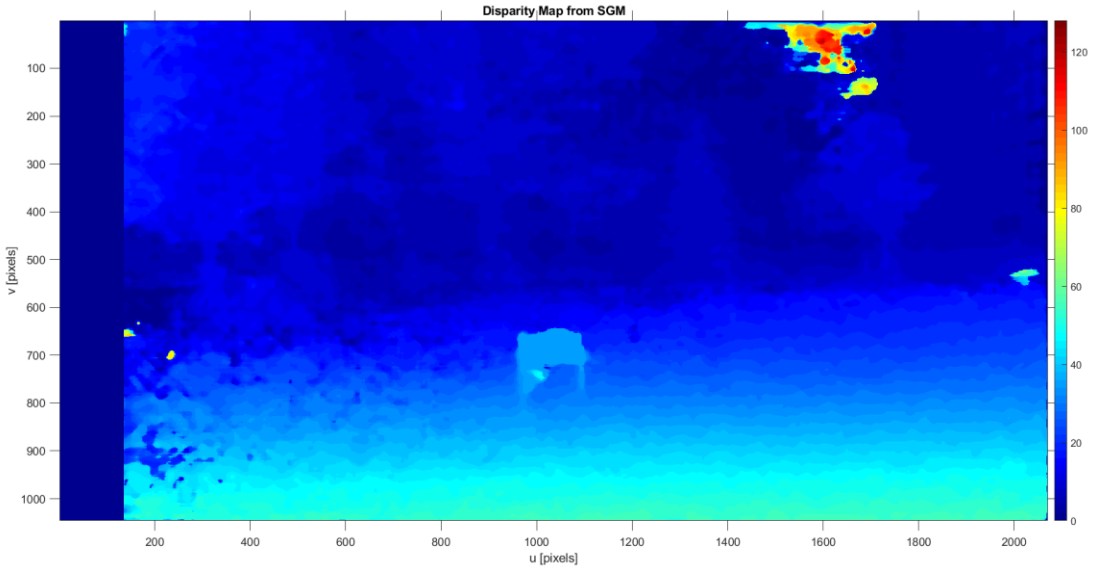


Figure 32. Disparity Map Corresponding to a Stereo Image Pair.

Figure 33 shows the top view of two cameras with parallel optical axes separated by its corresponding baseline b. The two images provide the view of the world point $P = [X, Y, Z]^T$. The projection over the image plane left is expressed as $p_L = [u_L, v_L]^T$ and the right plane as $p_R = [u_R\ , v_R]^T$.

The diagram shows the z-axis which represents the distance from the cameras (where $z = 0$ is located at the center point of both cameras), the x-axis represents the 'horizontal' distances, and the y-axis pointing downwards (across the page). Considering that the images are 'perfectly rectified', the *disparity* or distance between the corresponding keypoints p_L and p_R can be computed by the measurement over the x-axis of the coordinate system, where $v_L = v_R$ (the rows over the image plane), such that the *disparity* between u_L and u_R is the result of the difference of the camera positions or $d = u_L - u_R\ , d < 0$. We can apply simple elementary geometry to deduce the depth Z coordinate of the world point P.

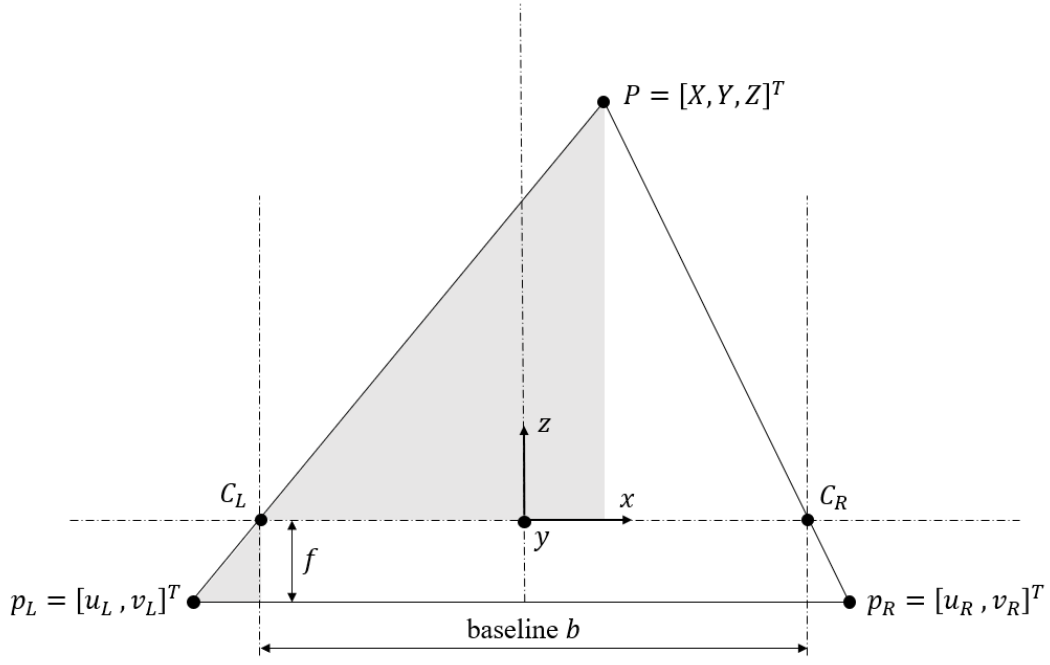


Figure 33. Stereo Geometry in a Rectified Configuration.

Notice that in the diagram, u_L, C_L and C_L, X are the hypotenuses of similar right-angled triangles (shown in gray). The focal length f and z are positive numbers and x, u_L, u_R are coordinates that may be positive or negative. Having this in mind, we can write the following expressions:

$$\frac{u_L}{f} = -\frac{\frac{b}{2} + X}{Z}, \quad \frac{u_R}{f} = \frac{\frac{b}{2} - X}{Z} \tag{4.16}$$

In equation 4.16 we can eliminate X by solving both equations in terms of X and substituting them into each other, which yields to $Z(u_R - u) = bf$ and, for instance, we can obtain:

$$Z = \frac{bf}{u_R - u_L} = \frac{bf}{d} \tag{4.17}$$

It is worth to mention that equation 4.17 depends directly on the value of the disparity $d = u_R - u_L$ of the world point P projected into the image frames. If $(u_R - u_L) \to 0$, then the value of the depth Z tends to infinity ($Z \to \infty$), where zero disparity indicates that the world point is at an infinite distance from the camera system. We can then deduce that '*small disparities'* yield relative errors in the measurement, whereas a *'wide disparity'* reduces the relative error in Z.

Then, the remaining X and Y coordinates of the world point P can be easily calculated as follows:

$$X = \frac{-b(u_L + u_R)}{2d}, Y = \frac{bv}{d} \tag{4.18}$$

where $v = v_L = v_R$ considering the assumption that the images are perfectly rectified.

4.4.3 Direct Disparity Method

The Semi-Global Matching algorithm is a powerful algorithm that provides the values of disparity corresponding to the set of keypoints p_L and p_R to compute the 3D coordinates of their corresponding world points P. The disadvantage of this method in comparison with the DLT algorithm is that it is computationally expensive, and it demands to be performed over a Graphic Processing Unit (GPU).

In order to avoid time consumption and computing power, one can just calculate the disparity directly from a set of corresponding keypoints p_L and p_R. Since the images processed are assumed to be rectified, it is easy calculate the *disparity* (difference or distance) of a corresponding set of keypoints. By following the stereo geometry shown in Figure 33, we can just calculate the *disparity* thus:

$$d = u_L - u_R \, , d < 0 \tag{4.19}$$

Then, by following equations 4.16, 4.17 and 4.18, it is easy to calculate the 3D coordinates of the projected set of keypoints p_L and p_R. This provides a good calculation of the coordinates in Euclidean space of the world point P, without the need of process a Disparity Map to obtain the disparity values of a set of corresponding stereo matching features.

4.5 Case of Study: ZED and ZED-Mini Depth Estimation Evaluation

One of the most important tasks for any Visual Odometry system is *Depth Estimation*. Depth estimation in stereo vision systems is susceptible to deviations because of different sources of errors, such as lighting, texture on the scene, blur, among others. In this section,

the author presents a case of study to evaluate and assess the depth estimation measurements obtained from the ZED and ZED-Mini stereo cameras.

The ZED and ZED-Mini have their own SDK application that allows the user to capture images and videos. These data can be retrieved in two different formats; images and videos can be exported as *rectified,* which means that the software outputs the data without lens distortions; or as *unrectified* or *raw* data, which contains the radial and tangential distortions induced by the lenses.

The problem with the *rectified* data provided by the Stereolabs system is that the size of the output image or video is the same as the input, e.g. if the user captures an image at 1080×1920 pixels (FHD) resolution, the rectified image output would be the same image size of 1080×1920 pixels. This presents two problems: 1) it is clear that some portion of the output image has been interpolated in order to match the same size of the input image; and 2) since the number of pixels added or removed by the interpolation to the output image is unknown, it is not possible to establish where the center of the image is, thus, the *principal point* or *center of the image* is unknown.

In this section, a case of study is introduced to assess the results of the depth estimation measurements obtained with the ZED and ZED-Mini stereo cameras, where the main goal is to compare the *rectified* and *unrectified* outputs of the Stereolabs SDK system and evaluate their impact in depth estimation. Also, this case of study aims to asses each of the methodologies presented in the aforementioned Section 4.4 by performing a comparison of all the methods, in both the *rectified* and the *unrectified* outputs, in terms of depth estimation.

The experiment consisted in taking images of a calibration chess board pattern located in front of the camera starting at 5 meters of distance with respect to the fixed location of the stereo camera. Then, the distance of the pattern with respect to the camera was increased by 5 meters at each image shot, until a distance of 30 meters was reached following the same methodology as in [49], and as it is shown in Figure 34. The experiment has been performed in two different scenarios, *Indoors* and *Outdoors*, in order to assess the consistency of the system in different environments.

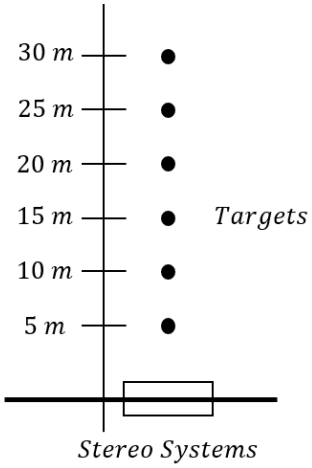


Figure 34. Depth Estimation Experimental Evaluation Layout

The pictures were obtained from both stereo cameras, ZED and ZED-Mini. They were considered as in *rectified* (i.e. radial and tangential distortion were removed from the dataset using the Stereolabs SDK), and in *unrectified* (raw) format. The resolutions selected for the purposes of this evaluation were FHD (1080 pixels) and HD (720 pixels) for both cameras.

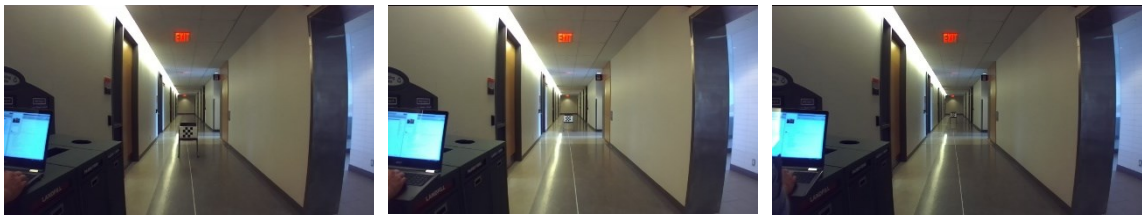
Then, the *unrectified* images were rectified using the camera parameters obtained in Section 3.3 and summarized in Table 1 to 4. The *unrectified* images obtained from the ZED Stereo Camera (baseline of 120 mm) changed in sizes from the raw resolution of

1080×1920 pixels to 1045×2070 pixels, and 720×1280 pixels to 719×1514 pixels, respectively. In other hand, the rectified images from the ZED-Mini Stereo Camera (baseline of 63 mm) changed in sizes from the raw resolution of 1080×1920 pixels to 1080×2095 pixels, and 720×1280 pixels to 741×1532 pixels, respectively.

During the experimental process, it was observed that as the distance of the chess board pattern was increasing in distance with respect to the stereo cameras, and for this reason the stereo matching keypoint algorithm, specifically the ones performed with the SURF feature – descriptor, yield into erroneous matching correspondences. This was because the size of the chess board pattern decreases in terms of pixels: the longest the distance of the pattern with respect to the camera, the lower its size in terms of pixels. For this reason, for the purposes of this evaluation, keypoints selection was performed using the Harris corner detector instead of SURF.

Also, as it was shown in Sections 4.2 and 4.3, the combination of FLANN matching algorithm with 5-Point RANSAC outlier removal presented more reasonable results than using the Brute Force matching algorithm. Because of these reasons, the evaluation was performed using the FLANN as keypoint matcher and the 5-Point RANSAC algorithm for keypoint outlier removal.

Figure 35 shows a sample of the *unrectified* images that were taken for the purposes of this evaluation. On the top, one can see the *Indoors* environment, and at the bottom of the image, the *Outdoors* environment, where the distances are displayed in meters with respect to the stereo camera.

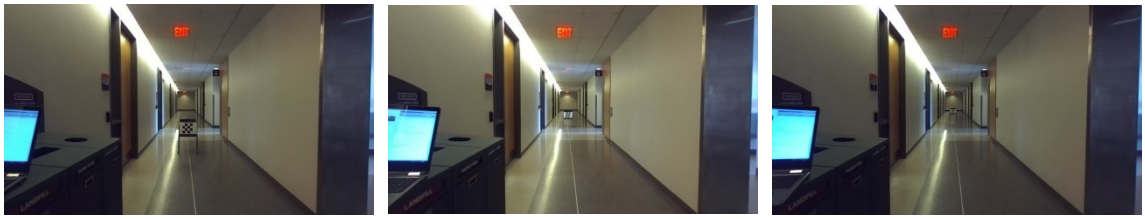


a) 5 $meters$ b) 10 $meters$ c) 15 $meters$

d) 5 $meters$ e) 10 $meters$ f) 15 $meters$

Figure 35. Depth Estimation Experimental Evaluation – Unrectified Samples.

In the same fashion, Figure 36 shows a sample of the *rectified* images that were extracted from the Stereolabs SDK for this evaluation. On the top, one can observe the *Indoors* environment, and at the bottom, the *Outdoors* environment.



a) 5 $meters$ b) 10 $meters$ c) 15 $meters$

d) 5 $meters$ e) 10 $meters$ f) 15 $meters$

Figure 36. Depth Estimation Experimental Evaluation – Rectified Samples

In the following tables and figures, the results of the evaluation are presented. The depth estimation measurements obtained from the three different algorithms for scene reconstruction are presented in meters units. The tables show the results of the *Indoors* and Outdoors environments, meanwhile the figures show the results in graphic representation for to facilitate the explanation.

For an easy interpretation of the results, both tables and figures show the results based on the *distance* of the chess board pattern with reference to the left lens of the stereo cameras. In regards to the tables, two sections specify the type data, 'Rectified' or 'Unrectified'. Also, from left to right the results are listed as Direct Disparity (DD); Direct Linear Triangulation (DLT); and Semi-Global Matching (SGM).

Table 7. Depth Estimation Evaluation Results – ZED at 1080 pixels - Indoors.

Indoors Depth Estimation Evaluation – ZED – 1080p						
Distance $[m]$	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD $[m]$	**DLT** $[m]$	**SMG** $[m]$	**DD** $[m]$	**DLT** $[m]$	**SMG** $[m]$
5	4.94	6.82	4.80	4.96	2.96	4.94
10	9.31	3.98	9.51	9.53	4.00	9.49
15	14.49	4.70	13.83	14.42	4.67	14.66
20	17.77	5.00	17.61	19.09	5.08	19.93
25	24.02	5.40	21.65	22.64	5.30	24.90
30	26.50	5.51	24.41	27.33	5.51	30.78

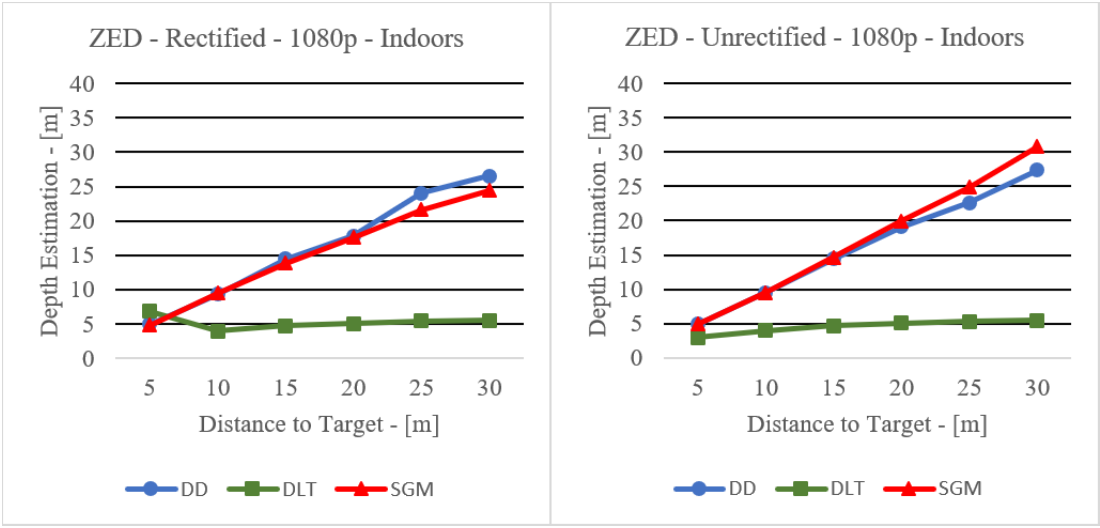


Figure 37. Depth Estimation Evaluation Results – ZED at 1080 pixels - Indoors.

Table 8. Depth Estimation Evaluation Results – ZED at 1080 pixels - Outdoors.

Outdoors Depth Estimation Evaluation – ZED – 1080p						
Distance [m]	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD [m]	**DLT [m]**	**SMG [m]**	**DD [m]**	**DLT [m]**	**SMG [m]**
5	4.92	2.87	4.66	4.81	6.72	4.83
10	9.20	3.94	8.92	9.29	5.36	9.29
15	13.29	4.55	13.18	13.79	4.58	14.78
20	16.51	4.89	17.09	17.02	4.90	17.93
25	19.74	5.13	21.09	22.99	5.29	21.70
30	28.57	5.59	23.77	24.16	10.67	39.03

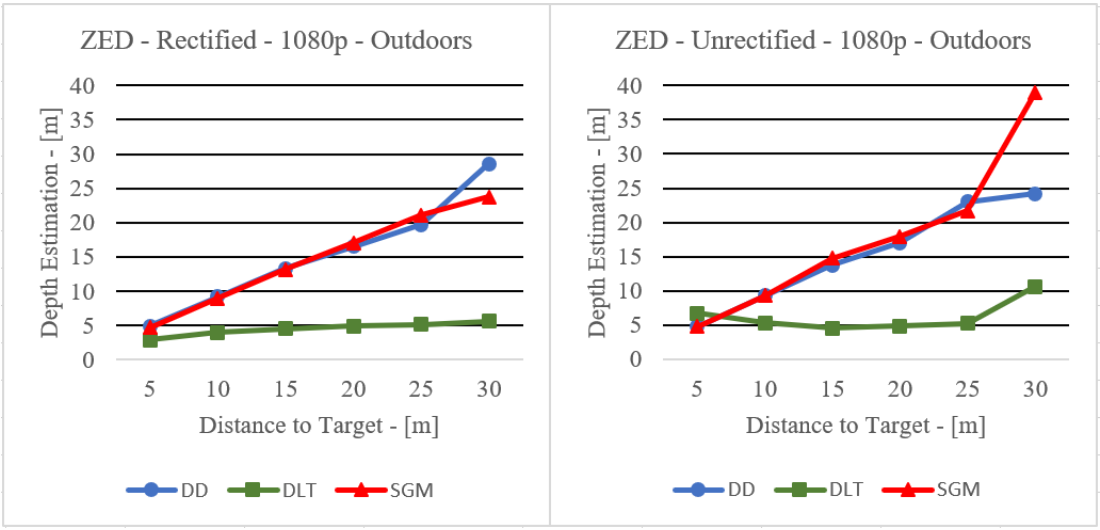


Figure 38. Depth Estimation Evaluation Results – ZED at 1080 pixels - Outdoors.

Table 9. Depth Estimation Evaluation Results – ZED-Mini at 1080 pixels - Indoors.

Indoors Depth Estimation Evaluation – ZED-Mini – 1080p						
Distance [m]	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD [m]	**DLT [m]**	**SMG [m]**	**DD [m]**	**DLT [m]**	**SMG [m]**
5	5.30	3.53	5.19	5.11	3.41	5.12
10	9.53	5.15	10.59	10.10	5.06	10.13
15	15.85	6.26	16.35	14.98	6.30	15.23
20	20.18	6.95	21.30	21.01	6.84	20.89
25	24.75	7.46	26.90	24.28	7.19	29.01
30	29.88	9.22	30.48	31.61	8.58	30.20

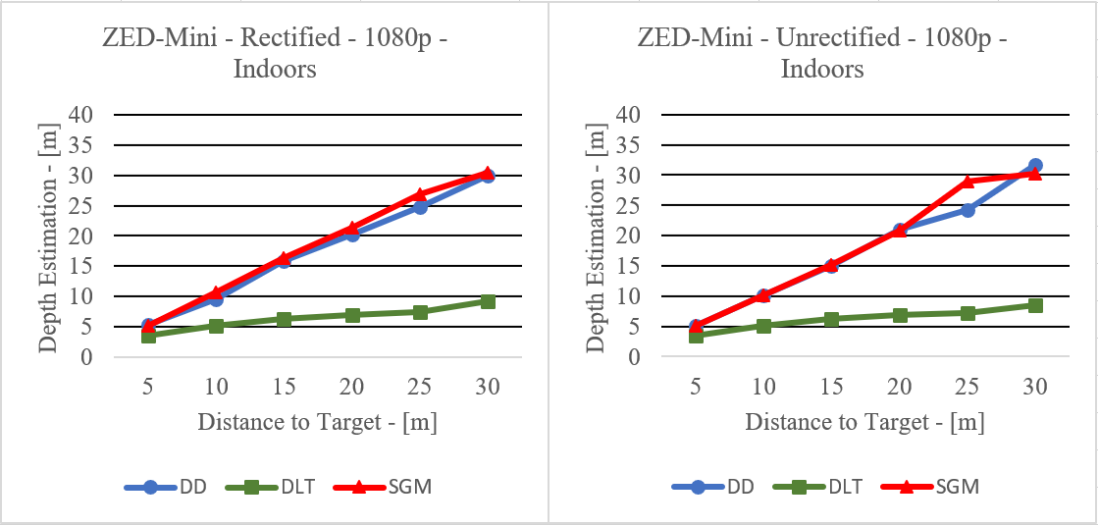


Figure 39. Depth Estimation Evaluation Results – ZED-Mini at 1080 pixels - Indoors.

Table 10. Depth Estimation Evaluation Results – ZED-Mini at 1080 pixels - Outdoors.

Outdoors Depth Estimation Evaluation – ZED-Mini – 1080p						
Distance [m]	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD [m]	**DLT [m]**	**SMG [m]**	**DD [m]**	**DLT [m]**	**SMG [m]**
5	5.03	3.39	4.98	4.98	3.31	4.97
10	9.96	5.05	9.79	10.01	5.02	10.05
15	15.17	6.32	14.87	14.86	6.21	14.90
20	20.28	6.89	20.95	19.97	6.92	19.96
25	25.02	10.32	28.41	24.18	8.49	30.93
30	29.09	8.27	30.79	31.14	8.27	32.68

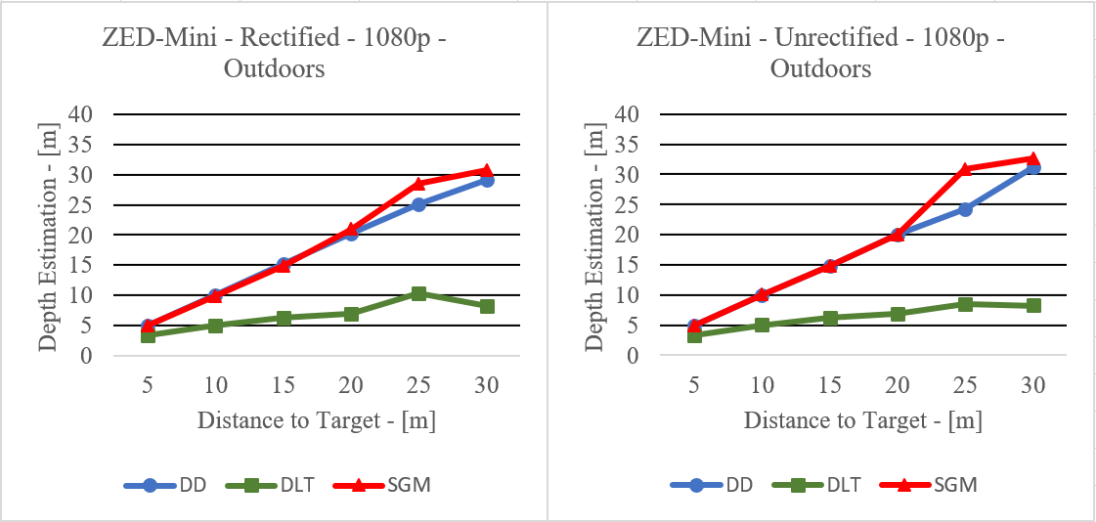


Figure 40. Depth Estimation Evaluation Results at 1080 pixels - Outdoors.

From the results, one can notice the tendency of *unrectified* data to be closer to the real distance than those from the *rectified* one by the Stereolabs SDK. The main reason why the results from the *rectified* data diverged from the real distance might be due to the fact that the center of the image was unknown, and thus, the calculations yielded erroneous results.

The Direct Disparity method was the method with less computation steps because once the keypoints were correctly matched and outliers were removed, the algorithm just had to measure the distance (in pixels) of the corresponding matching keypoints. Also, it was observed that this method was also the one that produced results closest to the real distance of the calibration pattern with respect to the fixed location of the cameras.

On other hand, the Semi-Global Matching algorithm showed good results but required more computation steps. This was because this method had to build the disparity map corresponding to the stereo image pair. After the disparity map was computed, the process of depth estimation was performed.

Finally, it can be noticed that the Direct Linear Transformation method did not perform as well as the two other aforementioned algorithms. One of the reasons why this method did not yield as good results might be due to the fact that it relied directly on the *Projection Matrices*, which are completely dependent on the camera calibration parameters. Therefore, it is recommended to perform a high precision camera calibration as the one proposed by Geiger et al. [50] in their work for the KITTI Vision Benchmark Suit.

It was also observed that the greater the distance between the pattern and the cameras, the depth estimation started yield to erroneous estimation values. This helps to set a threshold in terms of depth estimation. The results showed that depth estimation at approximately 25 meters of distance and beyond presented values that did not corresponded to the real distances between the pattern and the cameras.

Based on the aforementioned observations and findings, the *unrectified* output data from the ZED and ZED-Mini stereo cameras for the Visual Odometry algorithm were used. Also, the Direct Disparity method was selected to perform scene reconstruction, setting a threshold of 25 meters for depth estimation. This helps the performance of the Visual Odometry system because the points that lie over the radial threshold could be discarded.

Finally, the results for HD resolution (720 pixels) can be found on appendix C.

4.6 Summary

In this Chapter, two different methodologies for feature detection over the image plane were introduced: the Harris corner detector and the Speed Up Robust Features (SURF), as

well as techniques to match the detected keypoints in stereo pair images. Furthermore, the Random Sample Consensus algorithm to remove outliers was presented.

Three different algorithms to perform scene reconstruction and depth have been discussed, specifically, Direct Linear Transformation (DLT), Semi-Global Matching (SGM) and Direct Disparity.

A case of study was introduced and discussed, where the main goal was to assess the depth estimation measurements of the ZED and ZED-Mini stereo cameras. It was observed that the rectified output provided by the Stereolabs SDK was not as good as in the case when the process of Image Rectification was performed by using unrectified or raw data. In the following Chapter, the methodology to perform Motion Estimation of the camera in the world will be developed.

Chapter 5

Motion Estimation

Motion estimation is, computationally speaking, at the core of every Visual Odometry system. In a nutshell, the process estimated the motion of the camera between two consecutive frames, e.g. the previous image frame and the current image frame at a given instant of time. The estimated trajectory or path that the camera has traveled can be recovered by concatenating every single motion estimation. For example, as a new motion is estimated, the latest estimation is concatenated with the previous one until the last frame in a sequence of images or video is being reached.

5.1 Visual Odometry Pipeline

The machine vision community that focuses on Visual Odometry systems explains that *Visual Odometry Pipeline* are the algorithms that perform the estimation of trajectories based on camera systems. Depending on the kind of camera (monocular, binocular, or trifocal) the algorithms differ according to the requirements of each system.

In this section, the different pipeline algorithms and their adaptations to different vision systems are discussed. Each of the Visual Odometry Systems considers several algorithms to be performed either in real-time or in post-processing mode. The most typical assumptions include – but are not limited to – the following considerations:

- camera calibration must be done beforehand;

- the Epipolar Geometry works as a constraint on the system in all cases;
- image rectification can be performed during the process (in real-time); or beforehand in the case when the system performs in post-processing mode;
- if the system works in real time, image rectification is performed as part of the algorithm; however, if the system performs in post-processing mode, image rectification can be done beforehand; and
- the environment must present satisfactory illumination to be able to detect 'good' texture over the world scene to allow for the computation of apparent motion.

The steps that every variant of Visual Odometry System perform during processing can be summarized as follows:

1. extract sequence of images or video;
2. feature detection and extraction of keypoints;
3. scene reconstruction;
4. motion estimation; and
5. concatenation of camera transformations and optimization.

Needless to add, this general procedure might vary depending on the system so as to meet the specific system's requirements and are mostly dependent on how motion estimation is performed.

In this section, the author discusses how the correspondence matching keypoints between two consecutive frames is performed, as well as how the relative camera motion

is calculated. Also, three methodologies to process camera motion estimation are introduced: 2D-to-2D, 3D-to-3D, and 3D-to-2D. For the purposes of this thesis, from now onward, the suffix *k* will be used to represent instant of time.

5.1.1 2D-to-2D: Feature Correspondences Motion Estimation

This was the first approach that the scientific community suggested to solve the VO algorithms and its main property is the epipolar constraint. It is based on the feature correspondences f_{k-1}, f_k in 2D image coordinates between the two consecutive frames I_{k-1}, I_k. For simplicity, it assumes that the camera has been calibrated and the feature correspondences (set of keypoints) f_{k-1}, f_k are normalized. In other words, the inverse of the intrinsic camera matrix K^{-1} has been applied.

Given a set of correspondence features in 2D image coordinates of two consecutive frames, we can compute the essential matrix E by applying the *5-point algorithm* discussed in section 3.3.3. It is well-known that E contains the information of camera motion between two image frames, that is, rotation and translation up to an unknown scale factor. By solving the 5-point algorithm one can estimate E, but in order to compute a robust E , an *outlier removal* must be introduced.

RANSAC can be incorporated to the 5-point algorithm, such that in an iterative process, we compute E considering a randomized sample from the set of keypoints, and then test it against the entire set of features. One can perform this iterative process until reaching a certain ratio of *inliers* or a certain iteration threshold.

From the robust estimation of E, the rotation and the translation can be extracted. In general, there exist four possible solutions for rotation R and translation t. A detailed

decomposition of E into R and t can be found in [35], which, in a few words, is the application of SVD to E, such that:

$$\text{svd}(E) = USV^T \tag{5.1}$$

where $diag(S) = \{1, 1, 0\}$ and the four solution are:

$$\begin{aligned} R &= U(\pm W^T)V^T \,, \\ t &= U(\pm W)SU^T \end{aligned} \tag{5.2}$$

and:

$$W^T = \begin{bmatrix} 0 & \mp 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

Once the four solutions for R and t have been computed, the correct solution by triangulation of a single point can be estimated. The right solution is the one where the triangulated point lies in front of both cameras.

The visual odometry algorithm with the 2D-to2D correspondences is summarized as in *Algorithm 2,* in Table 11.

In algorithm 2, one can notice that the translation between two frames is up to an unknown scale factor. However, it is possible to compute relative scales for the subsequent transformations. One can triangulate the features from two consecutive frames f_{k-1}, f_k, to get a 3D point cloud as X_{k-1} and X_k. The relative distance between any combinations of two points in the computed point clouds can be calculated to determine the *distance ratio* between them thus:

$$r = \frac{\|X_{k-1,i} - X_{k-1,j}\|}{\|X_{k,i} - X_{k,j}\|} \tag{5.4}$$

In order to obtain major robustness, the scale ratios for many *inlier* points are computed and the mean of the ratio is used. Then, the translation vector t_k is rescaled with the distance ratio.

Table 11. Algorithm 2. 2D-to-2D feature correspondences.

Algorithm 2: 2D-to-2D: Feature Correspondences
i. Compute the first image frame I_k
ii. Extract features f_k
iii. Set initial camera pose C_k
iv. Store information from first frame as I_{k-1}, f_{k-1} and C_{k-1}, respectively
While Exist a new image frame
v. Compute a new frame I_k
vi. Extract features f_k
vii. Compute essential matrix for the image pair I_{k-1} and I_k
viii. Decompose essential matrix into R_k and t_k to form transformation T_k
ix. Compute relative scale between I_{k-1} and I_k and rescale t_k
x. Concatenate transformation by $C_k = C_{k-1}\, T_k$
If Optimization required
xi. Do windowed bundle adjustment to refine C_k
End if
xii. Store information from current frame as I_{k-1}, f_{k-1} and C_{k-1}, respectively
End While

5.1.2 3D-to-3D: 3D Structure Correspondences Motion Estimation

In comparison to 2D-to-2D feature correspondences algorithm (which can be performed by any visual camera system), the 3D-to-3D structure correspondences algorithm can only be performed by binocular or trifocal tensors. The reason for this is that the aforementioned method is up to scale by its nature of 2D features. In other hand, the 3D-to-3D can also

estimate depth, and for instance, the scale for the translation vector t can be directly estimated.

In this case, it is necessary to triangulate the 2D features extracted from two consecutive frames f_{k-1} and f_k at each instant of time. This method only applies to stereo vision systems. The camera motion T_k can be computed by determining the aligning transformation between the two 3D point clouds, which are calculated through triangulation algorithms.

The solution consists in finding the T_k that minimized the metric distance L_2, thus:

$$\arg\min_{T_k} \sum_i \left\| \tilde{X}_k^i - T_k \, \tilde{X}_{k-1}^i \right\| \tag{5.5}$$

where the subscript i denotates the ith feature, and $\tilde{X}_k^i$, $\tilde{X}_{k-1}^i$ are the homogeneous coordinates of the triangulated features f_{k-1} , f_k, i.e. $\tilde{X} = [x, y, z, 1]^T$.

In order to solve T_k, at least three non-collinear point correspondences between the triangulated features (point clouds) $\tilde{X}_k^i$ and $\tilde{X}_{k-1}^i$, are needed. In 1987, Arun et al. [51] proposed a methodology to solve this problem. In their work, they proposed that one possible solution is to compute the rotation R by utilizing SVD, and the translation t_k by calculating the difference of the centroid (mean value) of the triangulated features. By doing this, the translation can be expressed as:

$$t_k = \overline{X_k} - R\overline{X_{k-1}} \tag{5.6}$$

where the accent $\bar{\cdot}$ represents the arithmetic mean value of the set of point clouds.

Meanwhile, the rotation can be easily computed by applying SVD to the difference of the point clouds as $USV^T = \text{svd}((X_{k-1} - \overline{X_{k-1}})(X_k - \overline{X_k}))$, which yields:

$$R = VU^T \tag{5.7}$$

Since the computed transformation T_k has absolute scale, the trajectory of the camera motion can be easily computed by conducting a direct concatenation of the transformations of the sequence.

Some examples of applications similar to this approach can be found in [52], [53] The visual odometry algorithm for 3D-to-3D structure correspondences is summarized in *Algorithm 3,* as follows:

Table 12. Algorithm 3. 3D-to-3D: Structure Correspondences.

Algorithm 3: 3D-to-3D: Structure correspondences
i. Compute the first stereo image frames $I_{L,k}$ and $I_{R,k}$
ii. Extract and match stereo features $f_{L,k}$ and $f_{R,k}$
iii. Triangulate features to build the point cloud X_k
iv. Set initial camera pose C_k
v. Store information from first frame as $I_{L,k-1}$, $I_{R,k-1}$, $f_{L,k-1}$, $f_{R,k-1}$, X_{k-1} and C_{k-1}, respectively
While Exist a new image frame
vi. Compute a new stereo image pair $I_{L,k}$ and $I_{R,k}$
vii. Extract and match stereo features $f_{L,k}$ and $f_{R,k}$
viii. Triangulate features to build the point cloud X_k
ix. Compute the L_2 distance from the centroid of the point clouds X_{k-1} and X_k
x. Compute $T_k = [R\|t]$
xi. Concatenate transformation by $C_k = C_{k-1}\, T_k$
If Optimization required
xii. Do windowed bundle adjustment to refine C_k
End if
xiii. Store information from first frame as $I_{L,k-1}$, $I_{R,k-1}$, $f_{L,k-1}$, $f_{R,k-1}$, X_{k-1} and C_{k-1}, respectively
End While

5.1.3 3D-to-2D: 3D Structure to Feature 2D Correspondences Motion Estimation

Accordingly to Nistér [1], camera motion estimation using 3D-to-2D correspondences is more accurate than 3D-to-3D structure correspondences. The reason is that in 3D-to-3D we minimize the $norm\ L_2$ distance between the two point clouds [53]; whereas in 3D-to-2D, we attempt to minimize the reprojection error [54]. In this section, the author describes in detail the 3D-to-2D motion estimation, which is the one implemented for the purposes of this thesis and is illustrated in Figure 41.

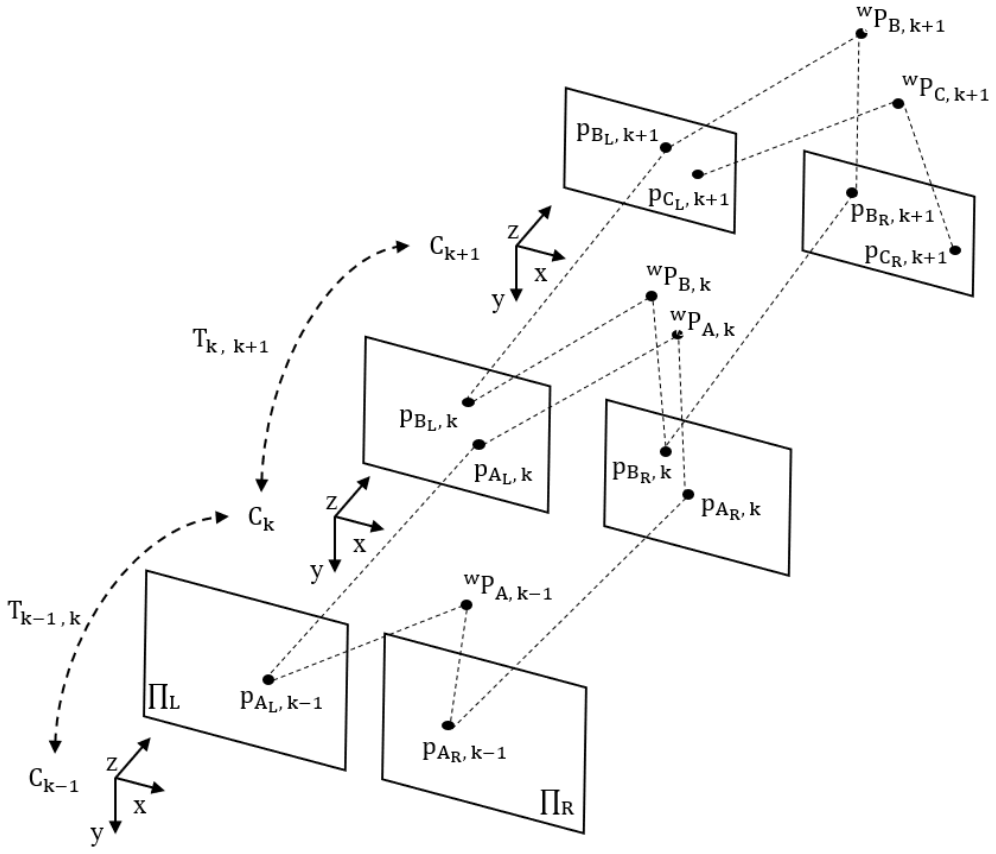


Figure 41. Relative Camera Pose and Concatenation of Transformations.

Consider a set of stereo features f_{k-1} and f_k which corresponded to the image frames I_{k-1} and I_k, respectively. Then, for the triangulated structure for the frame at $k-1$ is computed as X_{k-1}. The transformation T_k can be obtained by the correspondences of X_{k-1} and their reprojection into the 2D set of keypoints $p_{L,k}$ in the left image frame at I_k.

In general, the formulation of this case is to find the transformation T_k that minimizes the image reprojection error:

$$\arg\min_{T_k} \sum_i \left\| p_k^i - \hat{p}_{k-1}^i \right\|^2 \tag{5.8}$$

where $\hat{p}_{k-1}^i$ is the reprojection of the 3D point cloud X_{k-1}^i into the image frame I_k.

In the literature, this problem is known as *perspective-n-point* (PnP) problem. It was introduced in 1981 by Fischler and Bolles [55] and since then, there have been myriad approaches to solve it [56], [57]. The PnP originates from the camera calibration problem and it aims to determine the relative camera pose between two image frames. As it is shown in Fischler and Bolles' work, the minimal solution case involves three correspondences between the 3D structure point cloud and the 2D features set of keypoints.

The aforementioned minimal case is known as *perspective from three points* (P3P) [19], and just as the essential matrix E decomposition into rotation R and translation t, it returns four possible solutions as well. The P3P problem involves the law of cosines and a large number of elimination steps, which are beyond the scope of this thesis. Instead, the conceptual form for solving the algorithm is developed in this work.

Consider a sample of $n \geq 6$ from 3D-to-2D structure to image features correspondences. For simplicity, it is here assumed that the camera has been calibrated and feature correspondences (set of keypoints) f_{k-1}, f_k are normalized, in other words, that the inverse of the intrinsic camera matrix K^{-1} has been applied. By stacking the constraints

of six point-correspondences, a liner system of equations in the form $AT = 0$ is constructed, which can be solved via the direct linear transformation method (DTL) discussed in section 4.4.1, thus:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -x_1 & -y_1 & -z_1 & -1 & x_1v_1 & y_1v_1 & z_1v_1 & v_1 \\ x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 & -z1u_1 & u_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & -x_6 & -y_6 & -z_6 & 1 & x_6v_6 & y_6v_6 & z_6v_6 & v_6 \\ x_6 & y_6 & z_6 & 1 & 0 & 0 & 0 & 1 & -x_6u_6 & -y_6u_6 & -z_6u_6 & u_6 \end{bmatrix} \begin{bmatrix} T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{21} \\ T_{22} \\ T_{23} \\ T_{24} \\ T_{31} \\ T_{32} \\ T_{33} \\ T_{34} \end{bmatrix} = 0 \qquad (5.9)$$

where the vector T_{ij} represents the transformation between frames, $T_k = [R|t]$, x_i, y_i, z_i are the coordinates of the 3D point cloud X_{k-1}, and u_i, v_i represent the 2D image coordinates of the keypoints.

The entries of the vector T can be obtained from the null vector of A, which can be easily computed using SVD. Then, the rotation R and translation t can be extracted from $T_k = [R|t]$. It is important to consider that the rotation matrix obtained from the last step is not necessarily orthonormal. The Gram-Schmidt process [58] can be used to convert R into an orthogonal matrix.

In theory, this method could be computed for every single camera system. However, it is highly recommended to be used in binocular and trifocal systems. The motion estimation via the 3D-to-2D correspondences motion estimation assumes that the 2D points only come from one camera, which in most of the visual odometry systems implementations is the left lens of the stereo camera. Then, the system computes the P3P

problem to estimate de motion of the camera from one frame to another, by minimizing equation 5.7, giving as result the transformation (translation and rotation) in 6 DoF of the camera with respect to a previous state or instant of time.

There exists two variations of this algorithm depending in which monocular system is being used. For simplicity, the visual odometry algorithm for 3D-to-2D structure to feature correspondences is summarized in *Algorithm 4,* and it is presented in Table 13.

Table 13. Algorithm 4. 3D-to-2D: Structure to feature correspondences

Algorithm 4: 3D-to-2D: Structure to feature correspondences
i. Compute the first stereo image frames $I_{L,k}$ and $I_{R,k}$
ii. Extract and match stereo features $f_{L,k}$ and $f_{R,k}$
iii. Triangulate features to build the point cloud X_k
iv. Set initial camera pose C_k
v. Store information from first frame as $I_{L,k-1}$, $I_{R,k-1}$, $f_{L,k-1}$, $f_{R,k-1}$, X_{k-1} and C_{k-1}, respectively
While Exist a new image frame
vi. Compute a new stereo image pair $I_{L,k}$ and $I_{R,k}$
vii. Extract and match stereo features $f_{L,k}$ and $f_{R,k}$
viii. Triangulate features to build the point cloud X_k
ix. Track 2D features $f_{L,k-1}$ at $I_{L,k-1}$ to $f_{L,k}$ at $I_{,k}$; thus: ${}^{t}f_{k-1,k}$
x. Compute correspondences for the tracked features ${}^{t}f_{k-1,k}$ and X_{k-1}
xi. Compute camera pose estimation (P3P), thus: $T = [R\|t]$
xii. Concatenate transformation by $C_k = C_{k-1}\,T_k$
If Optimization required
xiii. Do windowed bundle adjustment to refine C_k
End if
xiv. Store information from first frame as $I_{L,k-1}$, $I_{R,k-1}$, $f_{L,k-1}$, $f_{R,k-1}$, X_{k-1} and C_{k-1}, respectively
End While

In the case of monocular cameras, the algorithm has the following variations:

i. Compute the first two frames I_{k-2} and I_{k-1}

ii. Extract and match features between first two frames f_{k-2} and f_{k-1}

iii. Triangulate features to build the point cloud $X_{k-2,k-1}$

5.2 Camera Motion Error Propagation

In visual odometry pipelines, the camera poses are computed by concatenating the transformations, in most of the cases, from two subsequent views (frames) at times k and $k-1$. However, each of these $T_{k-1,k}$ transformations has an uncertainty, and the uncertainty of the camera pose C_k depends on the uncertainty of the past transformations.

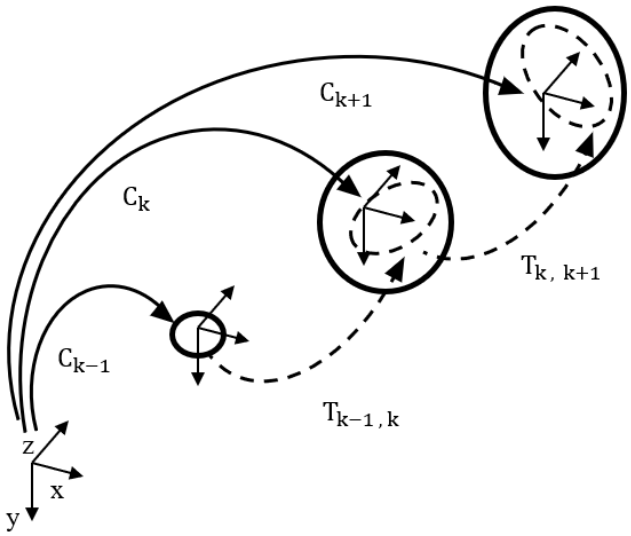


Figure 42. Uncertainty Propagation.

This is illustrated in Figure 42, where the uncertainty of $T_{k,k+1}$ computed by the VO system depends directly on the geometry induced by the camera calibration and the image features. The uncertainty of the camera pose at C_k is a combination of the uncertainty at a

previous instant of time C_{k-1} shown in solid ellipse, and the uncertainty induced by the transformation between the two consecutive frames $T_{k-1,k}$, shown in dashed ellipse. The uncertainty propagation can be for stereo systems can be reviewed in detail in [4].

Each camera pose C_k and each transformation $T_{k-1,k}$ can be represented by a six-element vector which contains the position and orientation (in Euler angles ϕ, θ, ψ) in Euclidean space. These element vectors are denoted as $\vec{T}_{k-1,k}$ and $\vec{C}_k$ respectively, e.g. $\vec{C}_k = [x, y, z, \phi, \theta, \psi]^T \in \mathbb{R}^6$. Each transformation $\vec{T}_{k-1,k}$ is represented by its mean and covariance $\Sigma_{k-1,k}$, which is a 6×6 matrix as is shown in [18]. The camera pose with respect to a previous frame is written as $\vec{C}_k = f(\vec{C}_{k-1}, \vec{T}_{k-1,k})$ with their covariances Σ_{k-1} and $\Sigma_{k-1,k}$. Then, $\vec{C}_k$ can be computed by using *error propagation law* [59], which utilizes a first-order Taylor series approximation, thus:

$$\Sigma_k = J \begin{bmatrix} \Sigma_{k-1} & 0 \\ 0 & \Sigma_{k-1,k} \end{bmatrix} J^T \tag{5.10}$$

and solving equation 5.10, we obtain:

$$\Sigma_k = J_{\vec{C}_k} \Sigma_{k-1} J_{\vec{C}_k}^T + J_{\vec{T}_{k-1,k}} \Sigma_{k,k-1} J_{\vec{T}_{k-1,k}}^T \tag{5.11}$$

where $J_{\vec{C}_k}$ and $J_{\vec{T}_{k-1,k}}$ are the Jocobians of the function $f(\vec{C}_{k-1}, \vec{T}_{k-1,k})$, respectively.

From equation 5.11, we can observe that the camera pose uncertainty is always increasing when the transformations frame to frame are being concatenating. For this reason, it is important to keep the uncertainties of each individual transformation as minimal as possible to reduce or avoid *drift.*

5.3 Windowed Bundle Adjustment

As it was previously discussed, error is propagated from the current camera motion to a further motion due to the uncertainty and inherent depth misestimations of the camera system. In order to reduce the error propagated in time, a camera pose optimization must be done.

The *windowed Bundle Adjustment* (BA) is an iterative least-squares approach optimization procedure. This method aims to optimize the parameters of the camera and the 3D structure landmarks tracked over time, i.e. tracked for more than two consecutive frames. Furthermore, it is more than a 3D reconstruction technique, since it simultaneously performs extrinsic calibration of the cameras and scene reconstruction.

Windowed bundle adjustment considers a determined window of n image frames and then performs a parameter optimization of the camera poses and the 3D landmarks for the set of image frames inside the window n. This algorithm aims to minimize the error function and thus minimize the reprojection error, such that:

$$\arg\min_{X^i, C_k} \sum_{i,k} \left\| p_k^i - g\left(X^i,\ C_k\right) \right\|^2 \tag{5.12}$$

where p_k^i is the ith 2D set of keypoints corresponding to the set of 3D landmarks X^i measured at the k^{th} frame and $g\left(X^i,\ C_k\right)$ is the reprojection of the 3D landmarks into their corresponding image frame with respect to the current camera pose C_k inside the window.

The algorithm requires an initialization that is close to the minimum, usually considering the camera pose between the first two poses inside the window. The Jacobian for this has a specific structure and can be reviewed for efficient computation in [34], [60] [61].

The problem with Windowed Bundle Adjustment is that can be hard to automate and set for a Visual Odometry application, since it is very sensitive to false correspondence mismatches. When it converges 'correctly', windowed bundle adjustment yields an optimal solution of camera pose estimation and landmark correction, i.e. the 3D configuration that best suits the observations in all views inside the window. The main assumption here is that the whole set of selected 2D keypoints and their corresponding 3D landmarks inside the window are correct.

5.4 Summary

In this Chapter, the camera motion estimation theory for Visual Odometry systems has been discussed, and Scaramuzza's categorization of the different types of motion estimation was developed. Furthermore, it explained that motion estimation is susceptible to error due to miscalculations induced by the camera system. The principal causes of error propagation in visual odometry systems which yield to a *drift* in the trajectory estimation was reviewed. Finally, the most popular method for motion optimization, namely the windowed bundle adjustment was introduced as a tool to refine the motion of the camera by correcting the 3D landmark viewed over a window (period of time) and their corresponding camera pose at a given instant of time.

Chapter 6

SVO with Local Optimization

In this Chapter, the author's proposed algorithm and its variations with respect to the ones discussed in the previous Chapter will be introduced. In this implementation, it has been assumed that the camera calibration procedure has been done beforehand, but with no limitations regarding the stereo image rectification of frames, e.g. the rectification can be done within the process of the algorithm.

6.1 Sparse Stereo Visual Odometry with Local Non-Linear Least-Squares Optimization

The algorithm proposed in this work performs in a similar fashion to the 3D-to-2D motion estimation algorithm and follows the same assumptions than the ones presents in Section 5.1 in regards to the minimal needs of illumination, camera calibration and epipolar constraint.

The algorithm starts by computing the first stereo image pair $I_{L,k}$, $I_{R,k}$ and performs image rectification if the frames are specified as *unrectified*, so as to enforce the epipolar geometry of the system. The algorithm then processes the rectified pair of frames to extract features and descriptors specified as $f_{L,k}$, $f_{R,k}$.

The features extracted are then processed through a modified five-point-RANSAC algorithm, where the possible mismatches and outliers are removed. From the features extracted with inliers only, scene reconstruction (triangulation) is conducted, yielding the point cloud X_k. It is assumed that the stereo visual odometry system starts from a zero location $t = (x, y, z)^T$ and orientation $R = (\phi, \theta, \psi)^T$ represented by the Euler angles, such that $T_k = [R|t]$ and $C_k = T_k = (x, y, z, \phi, \theta, \psi)^T \in \mathbb{R}^6$. The initial pose is then expressed as:

$$C_k = T_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.1}$$

Once the first camera pose is set, the current frame is processed and set as a previous frame. Thus, the current data to be processed against the next one as $I_{L,k-1}$, $I_{R,k-1}$, $f_{L,k-1}$, $f_{R,k-1}$, X_{k-1} and C_{k-1}, respectively.

The second stereo image pair is then computed following the same steps described above, thus $I_{L,k}$, $I_{R,k}$, $f_{L,k}$, $f_{R,k}$, X_k are computed. The Lucas-Kanade algorithm is used to track the inliers features only from previous frame $f_{L,k-1}$, $f_{R,k-1}$ to find 2D image correspondences from $I_{L,k-1}$, $I_{R,k-1}$ to $I_{L,k}$, $I_{R,k}$ which yields to ${}^t f_{L,k-1,k}$ and ${}^t f_{R,k-1,k}$. The tracked features in left and right current frames $I_{L,k}$, $I_{R,k}$ are then processed through the modified five-point-RANSAC algorithm to remove possible outliers.

Motion estimation is then computed using the perspective-three-points (P3P) problem and RANSAC, by using only the inliers set of tracked points in the left image frames ${}^t f_{L,k-1,k}$ and their corresponding structure 3D points from the previous image frame X_{k-1}, so the transformation of the frame at k with respect to $k-1$ is expressed as:

$$T_{k-1,k} = (x, y, z, \phi, \theta, \psi)^T \tag{6.2}$$

In here, the non-linear least-squares optimization approach is introduced by taking advantage of the P3P-RANSAC procedure, which not only gives as results the transformation (translation and rotation of the camera in 6 DOF), but also removes the outliers of the input set of 3D-to-2D correspondences.

This algorithm aims to minimize the reprojection error from the 3D structure points projected into the image frame in 2D features expressed in equation 5.8, but the right side of the stereo image pairs was also considered, e.g., left side and right side are processed over an iterative non-linear least-squares Levenberg-Marquardt algorithm [61], [62].

To do so, the reprojection error equation 5.7 has been modified to include the right side of the image, thus:

$$\arg\min_{T_k} mean \sqrt{\sum_i \left\| p_{L,k}^i - \hat{p}_{L,k-1}^i \right\|^2 + \sum_i \left\| p_{R,k}^i - \hat{p}_{R,k-1}^i \right\|^2} \tag{6.3}$$

where $\hat{p}_{L,k-1}^i$ and $\hat{p}_{R,k-1}^i$ are the reprojection of the 3D point cloud X_{k-1}^i into the image frames $I_{L,k}$ and $I_{R,k}$, and $p_{L,k}^i$ and $p_{R,k}^i$, are the inliers only set of tracked points from the frames at $k-1$ to the frames at k.

In this approach, the Euler angles for the orientation (ϕ, θ, ψ) obtained from the P3P-RANSAC algorithm are then expressed as *quaternions*, due to their lack of impact from the '*gimbal lock*' [64], [65], which is a phenomena where the Euler angles become coplanar. If during the concatenation of camera transformations and camera poses the Euler angles are coplanar the VO system would yield to erroneous trajectories. For this reason, the Euler angles are then expressed in quaternion form as:

$$\boldsymbol{q_k} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \begin{pmatrix} \cos(0.5\,\theta) \\ \phi \sin(0.5\,\theta) \\ \theta \sin(0.5\,\theta) \\ \psi \sin(0.5\,\theta) \end{pmatrix} \tag{6.4}$$

and substituting equation 6.5 into 6.3, the transformation of the current frame with respect to the previous frame $T_{k-1,k}$ can be rewritten as:

$$Q_{k-1,k} = (x, y, z, q_1, q_2, q_3, q_4)^T \tag{6.5}$$

where $Q_{k-1,k}$ represents the parameters of the transformation between frames to be optimized.

Then, the reprojected points $\hat{p}^i_{L,k-1}$ and $\hat{p}^i_{R,k-1}$ into 2D image frame from X^i_{k-1} are calculated at every iterative incremental step by the following equation:

$$\hat{p}^i_{L,k-1} = \boldsymbol{M}_L \, X^i_{k-1} \; ; \hat{p}^i_{R,k-1} = \boldsymbol{M}_R \, X^i_{k-1} \tag{6.6}$$

where $\boldsymbol{M}_L$ and $\boldsymbol{M}_R$ are the projection camera matrices of the left camera and the right camera, respectively.

The iterative process stops when the reprojection error, i.e. equation 6.4, reach a specific threshold. Then, $Q_{k-1,k}$ is normalized again into $T_{k-1,k}$, thus:

$$T_{k-1,k} = \left(x_f, y_f, z_f, \phi_f, \theta_f, \psi_f\right)^T \tag{6.7}$$

where the sub-index $\cdot_f$ denotates the final optimized parameters.

The concatenation of the transformation obtained from equation 6.8 and the first camera pose in equation 6.1 is then performed. Finally, the variables in the current frame are set to be processed into the next instant of time $k+1$. The algorithm repeats these steps until the last frame in the image sequence is reached.

$$C_k = C_{k-1} \, T_{k-1,k} \, , k = \{1,2,3, \dots n\} \tag{6.8}$$

Table 14 summarizes the algorithm proposed by the author.

Table 14. Algorithm 5: SVO with Local Optimization

Algorithm 5: SVO with Local Optimization

i. Compute the first stereo image frames $I_{L,k}$ and $I_{R,k}$
ii. Extract and match stereo features $f_{L,k}$ and $f_{R,k}$
iii. Triangulate features to build the point cloud X_k
iv. Set initial camera pose C_k
v. Store information from first frame as $I_{L,k-1}$, $I_{R,k-1}$, $f_{L,k-1}$, $f_{R,k-1}$, X_{k-1} and C_{k-1}, respectively

While Exist a new image frame

vi. Compute a new stereo image pair $I_{L,k}$ and $I_{R,k}$
vii. Extract and match stereo features $f_{L,k}$ and $f_{R,k}$
viii. Triangulate features to build the point cloud X_k
ix. Track 2D features $f_{L,k-1}$, $f_{R,k-1}$ at $I_{L,k-1}$, $I_{R,k-1}$ to $f_{L,k}$, $f_{R,k}$ at $I_{L,k}$, $I_{R,k}$; thus: ${}^{t}f_{L,k-1,k}$ and ${}^{t}f_{R,k-1,k}$
x. Compute correspondences for the tracked features ${}^{t}f_{L,k-1,k}$, ${}^{t}f_{R,k-1,k}$ and X_{k-1}
xi. Compute camera pose estimation (P3P) using only ${}^{t}f_{L,k-1,k}$ and X_{k-1} correspondences, thus: $T_{k-1,k} = [R|t]$
xii. Prepare $T_{k-1,k}$ for optimization, thus: R is express in quaternion $\boldsymbol{q}_k$
xiii. Build parameters to optimize, $Q_{k-1,k} = (x, y, z, q_1, q_2, q_3, q_4)^T$
xiv. Perform non-linear least-squares optimization of $Q_{k-1,k}$, minimizing equation 6.4
xv. Convert quaternion back to Euler angles: $T_{k-1,k} = \left(x_f, y_f, z_f, \phi_f, \theta_f, \psi_f\right)^T$
xvi. Concatenate transformation by $C_k = C_{k-1}\, T_{k-1,k}$
xvii. Store information from first frame as $I_{L,k-1}$, $I_{R,k-1}$, $f_{L,k-1}$, $f_{R,k-1}$, X_{k-1} and C_{k-1}, respectively

End While

6.2 Summary

The Sparse Stereo Visual Odometry with Local Non-Linear Least-Squares Optimization Algorithm has been discussed in this Chapter. In the following Chapter, an evaluation of the classic VO algorithm and the one discussed in this Chapter are presented.

Chapter 7

Experimental Evaluation

In order to evaluate the robustness of the Stereo Visual Odometry (SVO) system with Non-Linear Least-Squares Local Optimization, a set of data which provides ground truth camera pose information has been selected to evaluate the output results. Nevertheless, the author has also performed data acquisition with stereo systems and tested these algorithms. Both sets of data are focused on outdoor environments, specifically in automotive applications.

In this Chapter, firstly, the experiment that involves evaluating the capabilities of a stereo system in terms of calibration, keypoints selection, and depth estimation is discussed. Secondly, the results of the visual odometry system proposed by the author is evaluated against a set of data with ground truth information. Finally, the results from the data acquisition performed by the author are presented.

The system was implemented in the platform MATLAB [66], utilizing the Computer Vision [31], Image Processing [67], and the Optimization [68] toolboxes. Also, an interface to the C++ libraries of OpenCV [69] was implemented over the MATLAB environment. The acquired data from the ZED and ZED-Mini stereo cameras was exported in raw format from the Stereolabs SDK [70].

7.1 KITTI Vision Benchmark Suit

The first visual odometry dataset is the one provided by the *Karlsruhe Institute of Technology* (KIT) located in Karlsruhe, Germany, and the *Toyota Technological Institute* (TTI-C) located in Chicago, USA, known as the *KITTI vision benchmark suit* [71]–[73]. The benchmark suit is a compilation of data acquisition sequences of sensors like LiDAR, GNSS, and two stereo systems, RGBA and gray scale systems. The sensors were mounted in the autonomous driving platform Annieway [74] as shown in Figure 43. In their benchmark, they include datasets of *scene flow, depth estimation, tracking,* and the main focus of this work: *visual odometry.*

The dataset is provided in raw and rectified formats, and it includes the camera projection matrices of the stereo system as well as camera poses which are consider as ground truth data. The ground truth was built by the fusing of the *Velodyne Light Detection and Ranging* (LiDAR) and the *Global Navigation Satellite System* (GNSS) systems.

The dataset provides 22 stereo sequences of videos which were captured by driving around the city of Karlsruhe, Germany. The dataset provides 11 sequences with ground truth for training purposes and 11 without ground truth for the purposes of evaluation [75].

The KITTI Vision Benchmark Suit has been used for the purposes of development and testing of the stereo visual odometry algorithm presented in this thesis. The classical visual odometry pipeline (3D-to-2D: Correspondences Algorithm) with and without windowed Bundle Adjustment has been tested in order to compare the results with the method proposed in this work.

Figure 43. Autonomous Driving Platform AnnieWAY [72], [74]**.**

The sequences are provided in image format rectified gray scale and the dataset only provide the camera projection matrices of the corresponding left and right cameras as shown in equations 7.1 and 7.2. Since the KITTI benchmark only provides the projection matrices $\boldsymbol{M} = \boldsymbol{K}[R \mid -Rt\,]$, the only way to estimate depth on the sequences is by using Direct Linear Transformation method.

$$\boldsymbol{M}_L = \begin{bmatrix} 718.856 & 0 & 607.1928 & 0 \\ 0 & 718.856 & 185.2157 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{7.1}$$

$$\boldsymbol{M}_R = \begin{bmatrix} 718.856 & 0 & 607.1928 & -386.1448 \\ 0 & 718.856 & 185.2157 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{7.2}$$

The parameters of the image sequences provided by the KITTI dataset are listed in Table 15 and a sample of the image pair can be observed in Figure 44 and 45.

Table 15. KITTI Rectified Image Parameters.

KITTI Vision Odometry Benchmark Suit Parameters		
Parameter	***Description***	***Units / Format***
Rectified Image Format	Loss-Less	.png
Rectified Image Resolution	370x1226	pixels
Rectified Image Channels	1 – Gray Scale	N/A

Figure 44. KITTI Image Sequence Sample.

Figure 45. KITTI Stereo Image Pair Sample.

7.1.1 Methodology of Evaluation

For the purposes of evaluation, the system presented in this thesis (*Sparse Stereo Visual Odometry with Local Non-Linear Least-Squares Optimization* (Section 6.1)), was compared to the classic Visual Odometry with Windowed Bundle Adjustment algorithm

(Section 5.1.3 and 5.3). The systems were compared and evaluated to assess the robustness of the algorithm presented in this work.

The evaluation was performed in 5 of the 11 sequences provided with ground truth in the KITTI dataset. All the evaluations were performed with the same parameters for every algorithm involved. The whole list of parameters are shown in Table 16.

Table 16. KITTI Evaluation Parameters.

KITTI Evaluation Parameters		
Parameter	**Classic VO + BA**	**Ours**
	Algorithm / Description	**Algorithm / Description**
2D Features and Descriptors	SURF	SURF
Feature Matcher	FLANN	FLANN
Feature Tracker	Lucas-Kanade	Lucas-Kanade
Dual Tracking	N/A	Yes
Outlier Removal	5-Point-RANSAC	5-Point-RANSAC
Outlier Threshold	0.8 pixels	0.8 pixels
Depth Estimator	DLT	DLT
Reprojection Error Threshold	0.5 pixels	0.5 pixels
Max Depth Threshold	30 meters	30 meters
Motion Estimation	P3P-RANSAC	P3P-RANSAC
P3P Reprojection Threshold	0.8 pixels	0.8 pixels
Windowed BA	10 frames	N/A
Local Optimization	N/A	Yes

The system proposed in this thesis is capable of achieving results close to the ground truth data, especially in sequences involving short (less than 500 meters) to medium (around 1000 meters) range of travel distances. In the following sections, the results for the selected sequences 03, 05, 07, 09 and 10 are presented. The figures present the system's performance compared to the KITTI ground truth and to the classic visual odometry with windowed bundle adjustment algorithm. The KITTI ground truth is presented in *red color*

with the legend ‘KITTI - GT’; the classic Visual Odometry with Bundle Adjustment algorithm is shown in *green color* with the legend ‘VO + BA’; and finally, the system presented by the author is introduced in *blue color* with the legend ‘SVO + LO (author’s method).

All the results are presented in meters unless otherwise specified. The graphs for each sequence are presented in the aforementioned order and in the following categories of information: 1) Top view of the trajectory; 2) Camera trajectory in a 3D space; 3) RMS Error in X, Y and Z axis; 4) Translational RMS Error; 5) Rotational RMS Error and; 6) Estimated velocity.

The RMS errors in X, Y and Z axis refers to the drift that the Visual Odometry system is prompt to get due to the uncertainties of the transformations between cameras poses, and it was calculated thus:

$$X_{RMS} = \sqrt{(X_{GT} - X_{VO})^2} \tag{7.3}$$

$$Y_{RMS} = \sqrt{(Y_{GT} - Y_{VO})^2} \tag{7.4}$$

$$Z_{RMS} = \sqrt{(Z_{GT} - Z_{VO})^2} \tag{7.5}$$

where, X_{RMS}, Y_{RMS} and Z_{RMS} are the RMS error of the root mean squared difference of the camera ground truth location represented by X_{GT}, Y_{GT} and Z_{GT}, and the estimated camera translation calculated by the VO systems (VO + LO and VO + BA) represented by X_{VO}, Y_{VO} and Z_{VO}.

The translational and rotational RMS errors were computed comparing the estimated transformations of the camera at every instant of time and the camera locations provided in the ground truth data, such that:

$$Tanslational_{RMS} = \sqrt{(X_{GT} - X_{VO})^2 + (Y_{GT} - Y_{VO})^2 + (Z_{GT} - Z_{VO})^2} \quad (7.6)$$

$$Rotational_{RMS} = \sqrt{(\phi_{GT} - \phi_{VO})^2 + (\theta_{GT} - \theta_{VO})^2 + (\psi_{GT} - \psi_{VO})^2} \quad (7.7)$$

where ϕ_{GT}, θ_{GT}, ψ_{GT} are the Euler angles at every camera pose provided on the ground truth data and ϕ_{VO}, θ_{VO}, ψ_{VO} are the estimated Euler angles obtained from the VO systems (VO + LO and VO + BA) evaluated in this work.

The main reason why the system was evaluated only in five of the eleven sequences provided in the KITTI dataset was that neither the system proposed in this thesis nor the VO with windowed BA were capable to mitigate the drift in trajectories at high velocities. Another reason is that some of the sequences presented traveled distances over the threshold that is being evaluated in this thesis work (1.5 kilometer).

7.1.2 KITTI Dataset – Sequence 03 Results

Sequence 03 is a hybrid environment sequence where the vehicle starts at the limits of a residential area and moves across a landscape that ends up in the forest area. Table 17 shows a summary of the results presented in Figures 46 to 51.

Table 17. KITTI Sequence 03 Results Summary.

Parameter:	Units	KITTI - GT	SVO+LO	VO + BA
Total Displacement	$[m]$	472.44	478.14	484.58
Mean Velocity	$[km/h]$	21.26	21.51	21.8
Max Velocity	$[km/h]$	34.03	34.95	35.26
Mean Translational RMS Error	$[m]$	N/A	10.6846	23.8728
Mean Rotational RMS Error	$[deg]$	N/A	0.0695	0.1334

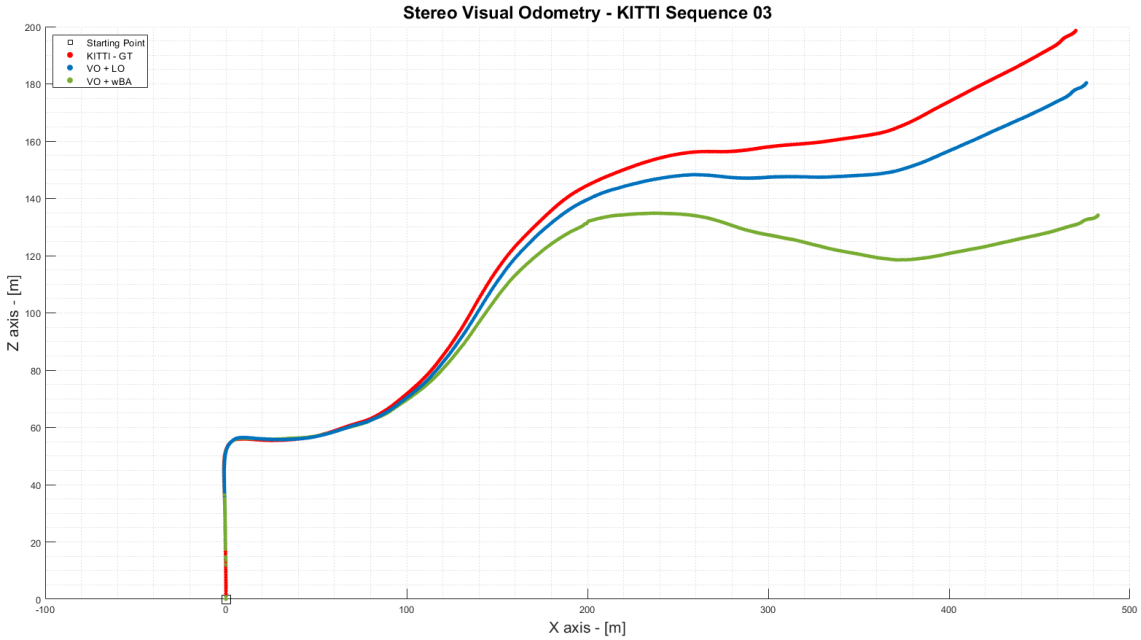


Figure 46. KITTI Sequence 03 – Top View Trajectory.

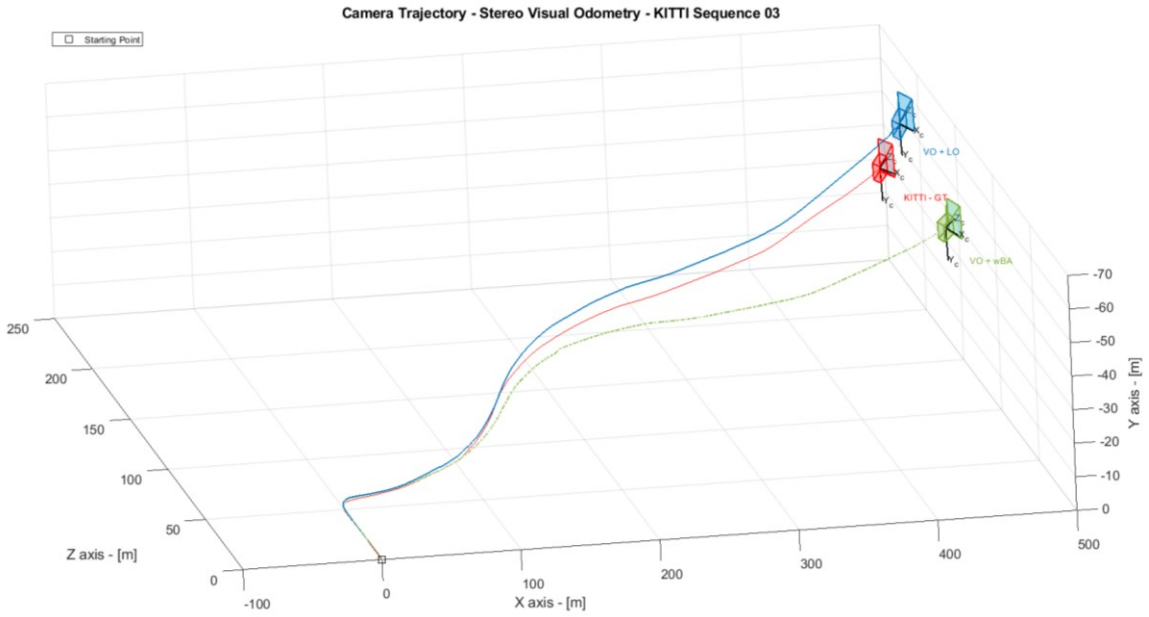


Figure 47. KITTI Sequence 03 – Camera Trajectory 3D Space.

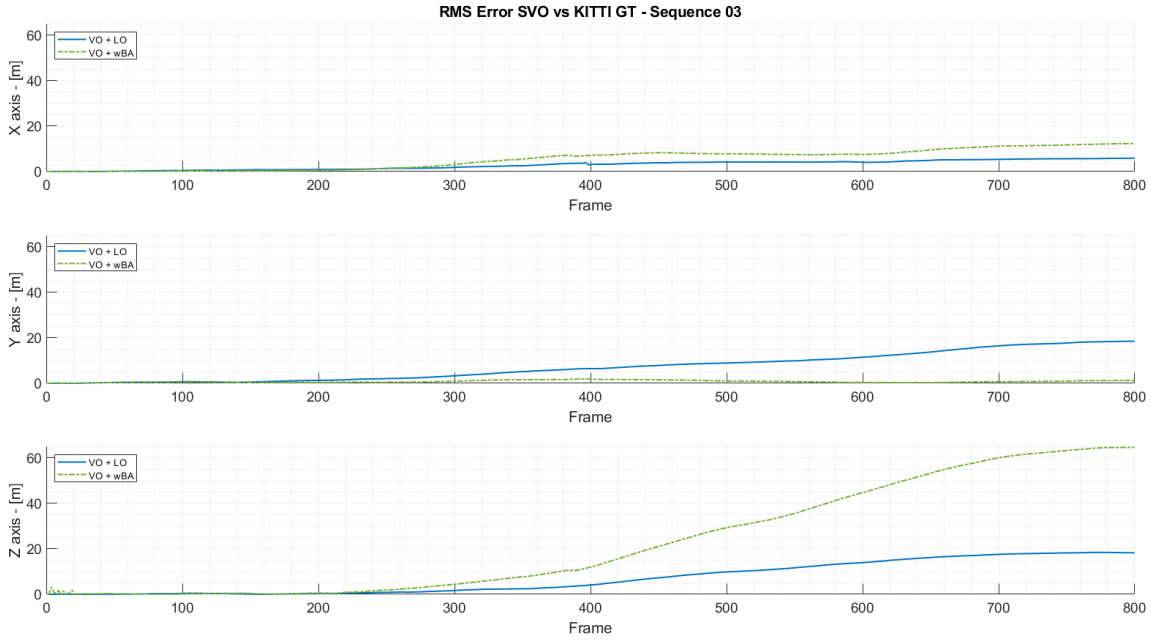


Figure 48. KITTI Sequence 03 – RMS Error in XYZ axis.

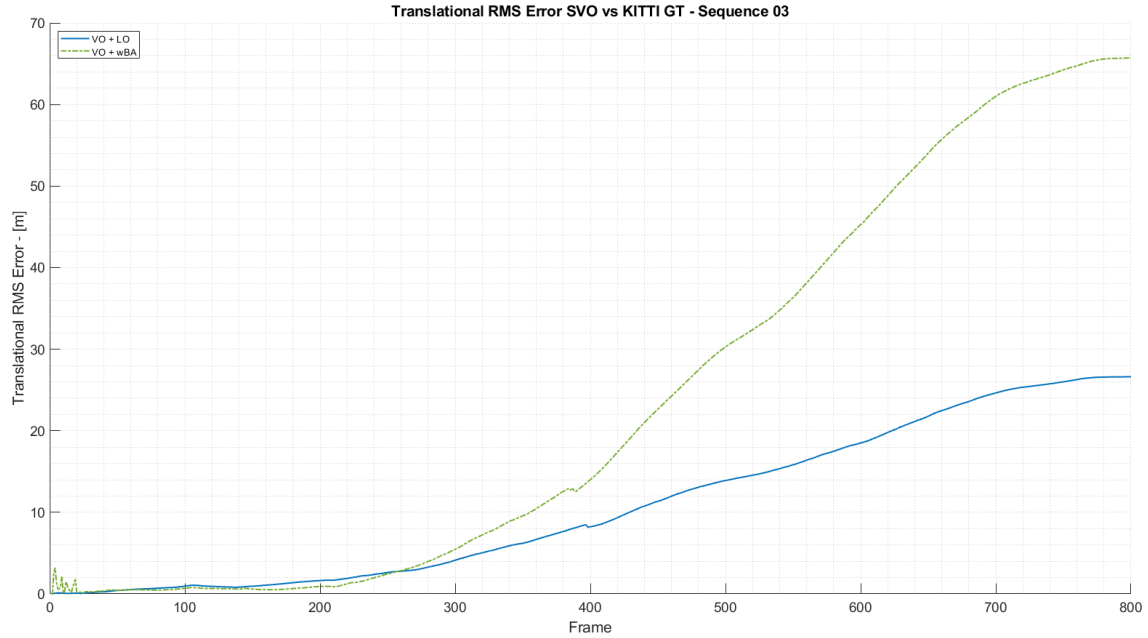


Figure 49. KITTI Sequence 03 – Translational RMS Error.

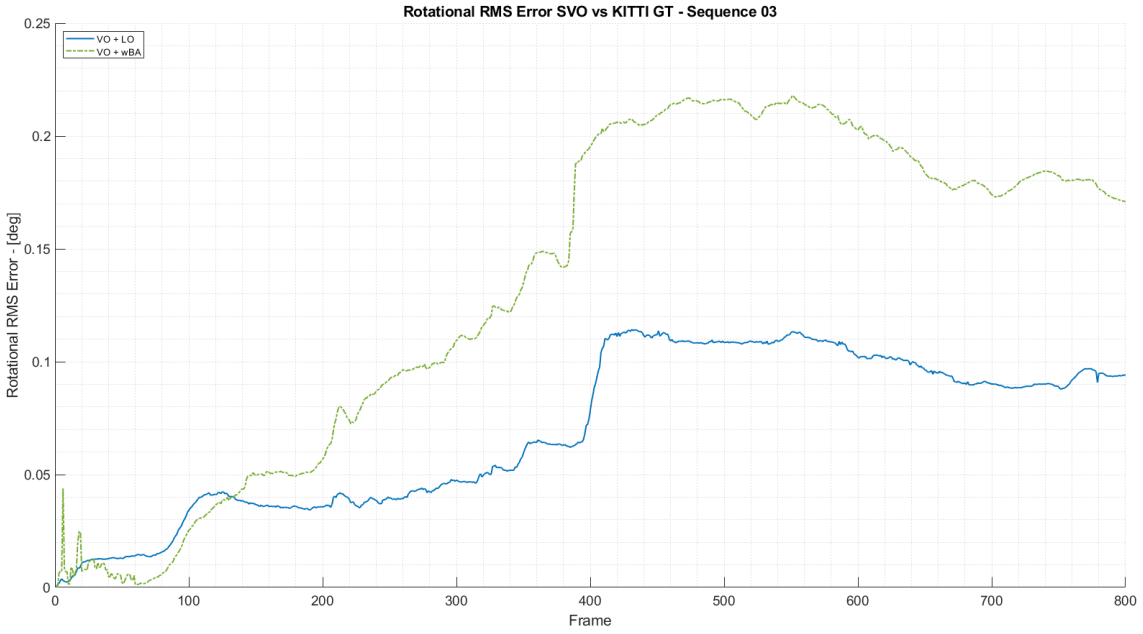


Figure 50. KITTI Sequence 03 – Rotational RMS Error.

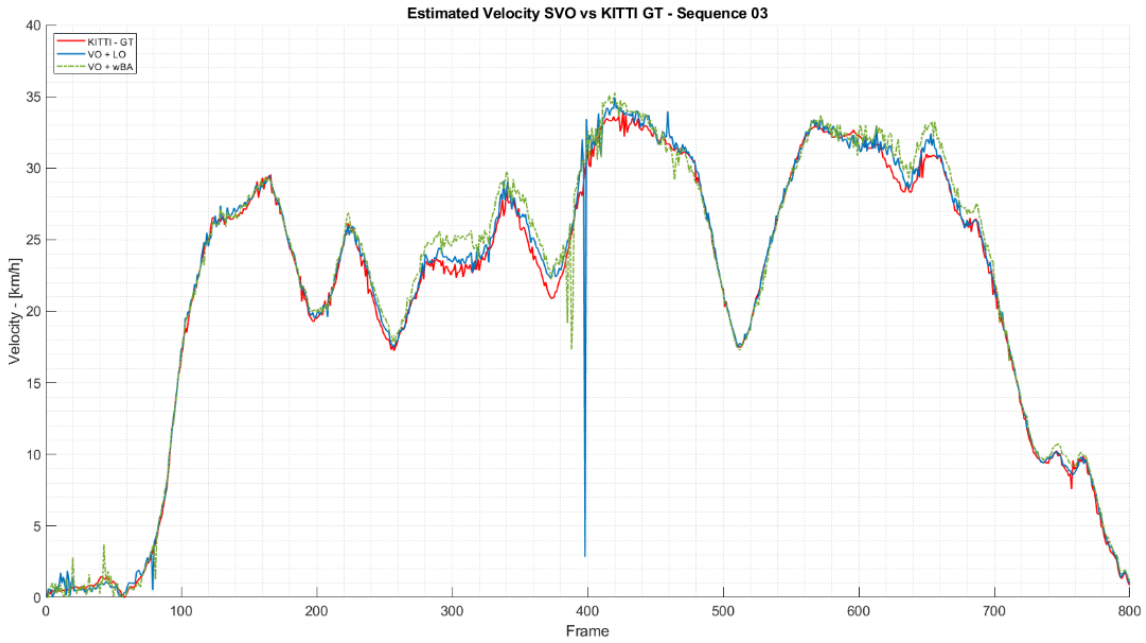


Figure 51. KITTI Sequence 03 – Estimated Velocity.

7.1.3 KITTI Dataset – Sequence 05 Results

Sequence 05 was recorded in a residential neighborhood of the city of Karlsruhe. The vehicle drove an approximated distance of 1.5 km with several turns, where car traffic was presented. Table 18 shows a summary of the results presented in Figures 52 to 57.

Table 18. KITTI Sequence 05 Results Summary.

Parameter:	Units	KITTI - GT	SVO + LO	VO + BA
Total Displacement	$[m]$	1414.1	1477.0	1489.2
Mean Velocity	$[km/h]$	18.44	19.26	19.42
Max Velocity	$[km/h]$	42.07	43.43	43.34
Mean Translational RMS Error	$[m]$	N/A	20.5350	22.2435
Mean Rotational RMS Error	$[deg]$	N/A	0.3478	0.3654

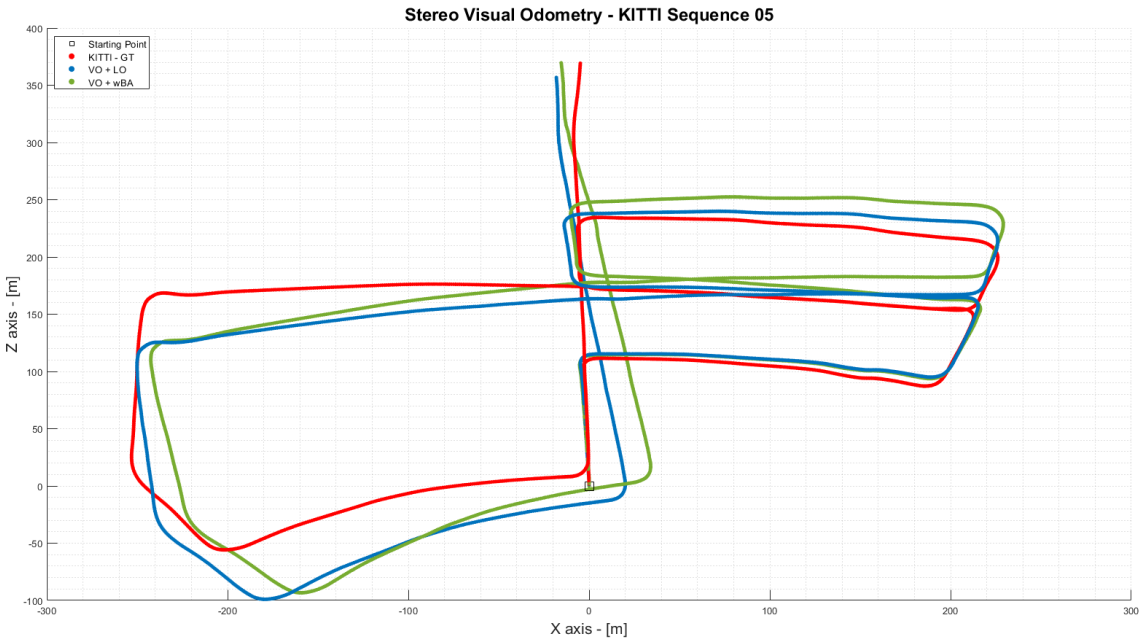


Figure 52. KITTI Sequence 05 – Top View Trajectory.

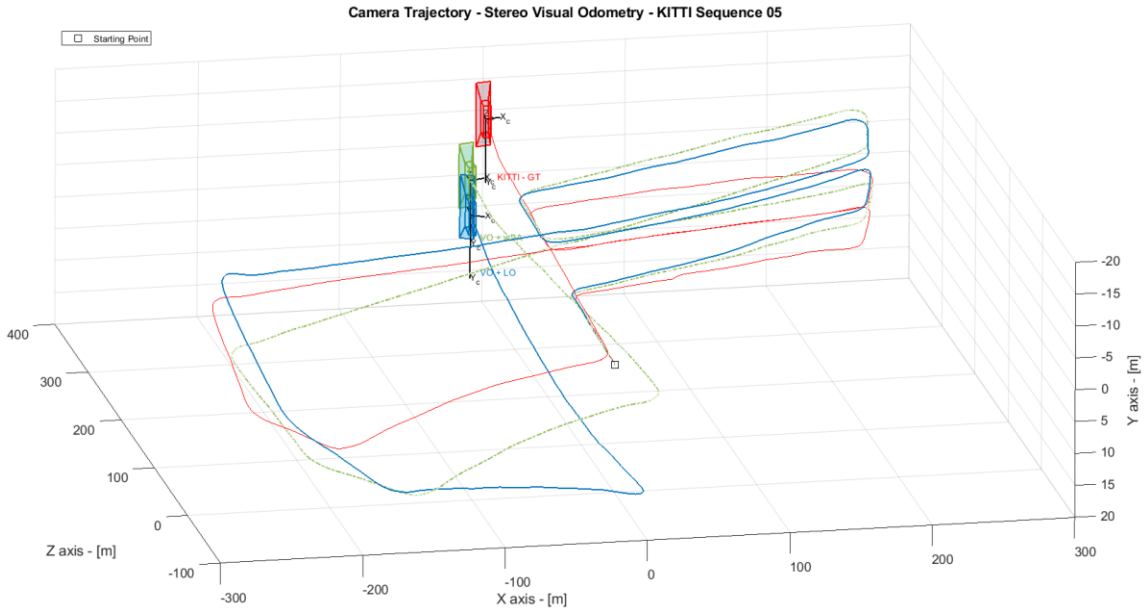


Figure 53. KITTI Sequence 05 – Camera Trajectory 3D Space.

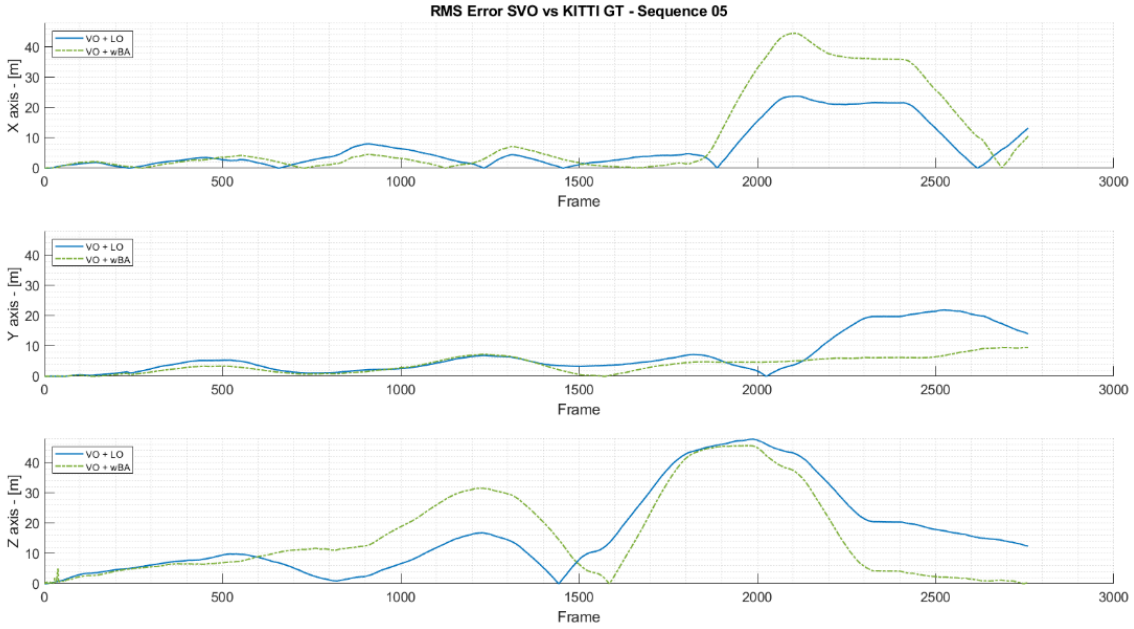


Figure 54. KITTI Sequence 05 – RMS Error in XYZ axis.

Figure 55. KITTI Sequence 05 – Translational RMS Error.

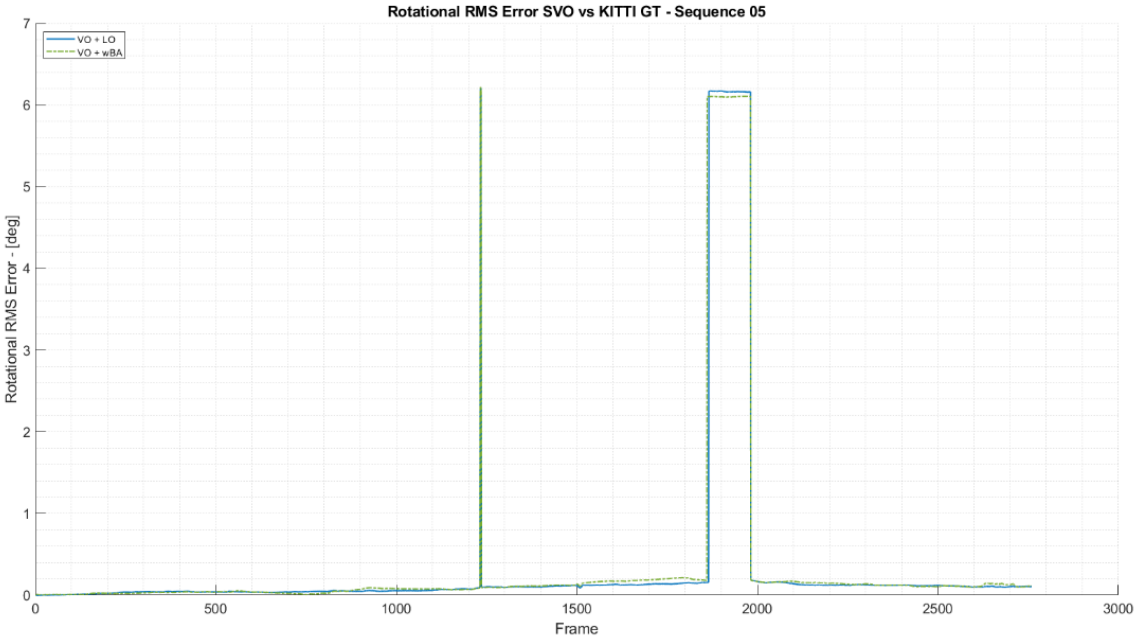


Figure 56. KITTI Sequence 05 – Rotational RMS Error.

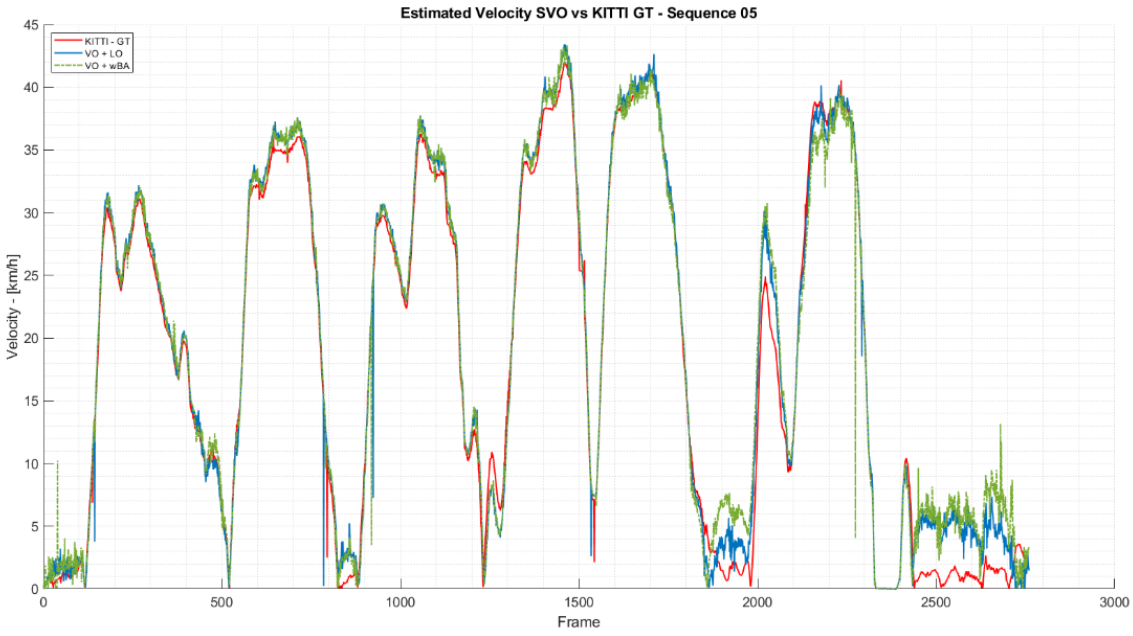


Figure 57. KITTI Sequence 05 – Estimated Velocity.

7.1.4 KITTI Dataset – Sequence 07 Results

Sequence 07 is one whereby the vehicle drives around a residential area. The sequence shows a busy environment such as the usual environment that an autonomous vehicle must be able to manage. Table 19 shows a summary of the results presented in Figures 58 to 63.

Table 19. KITTI Sequence 07 Results Summary.

Parameter:	Units	KITTI - GT	SVO + LO	VO + BA
Total Displacement	$[m]$	385.14	395.64	398.91
Mean Velocity	$[km/h]$	12.60	12.94	13.05
Max Velocity	$[km/h]$	43.43	44.58	45.01
Mean Translational RMS Error	$[m]$	N/A	4.7251	5.0872
Mean Rotational RMS Error	$[deg]$	N/A	0.0075	0.1116

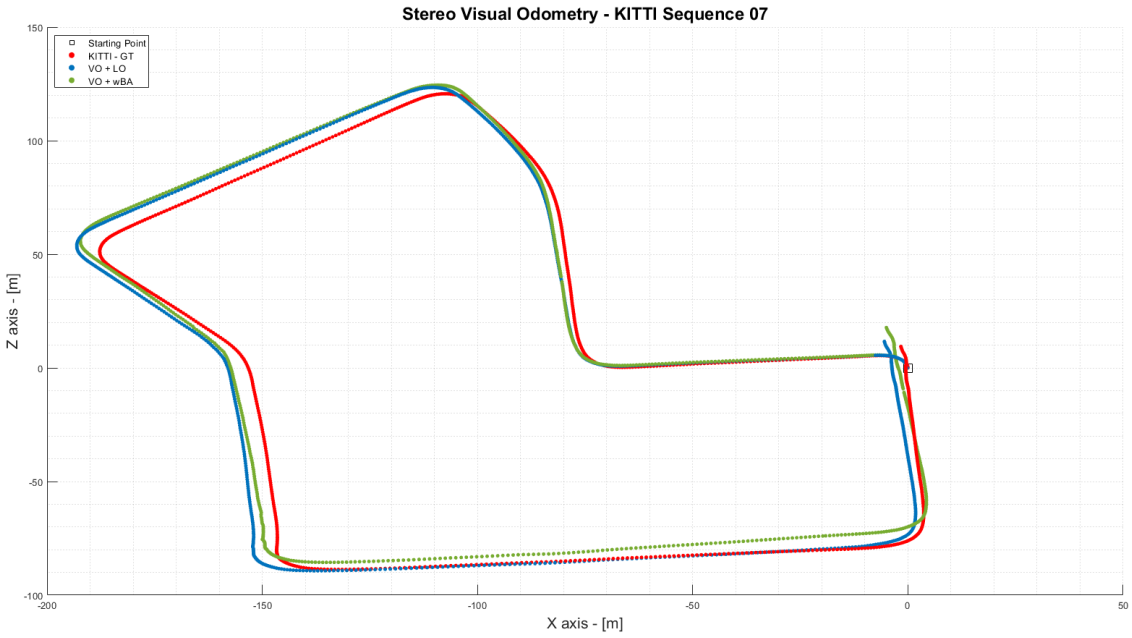


Figure 58. KITTI Sequence 07 – Top View Trajectory.

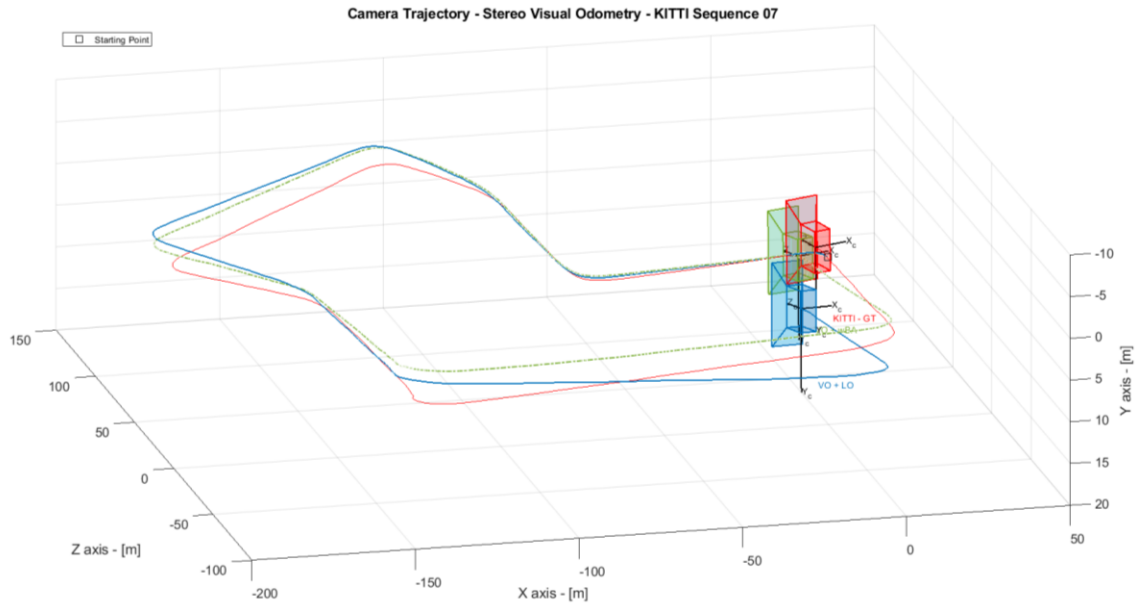


Figure 59. KITTI Sequence 07 – Camera Trajectory 3D Space.

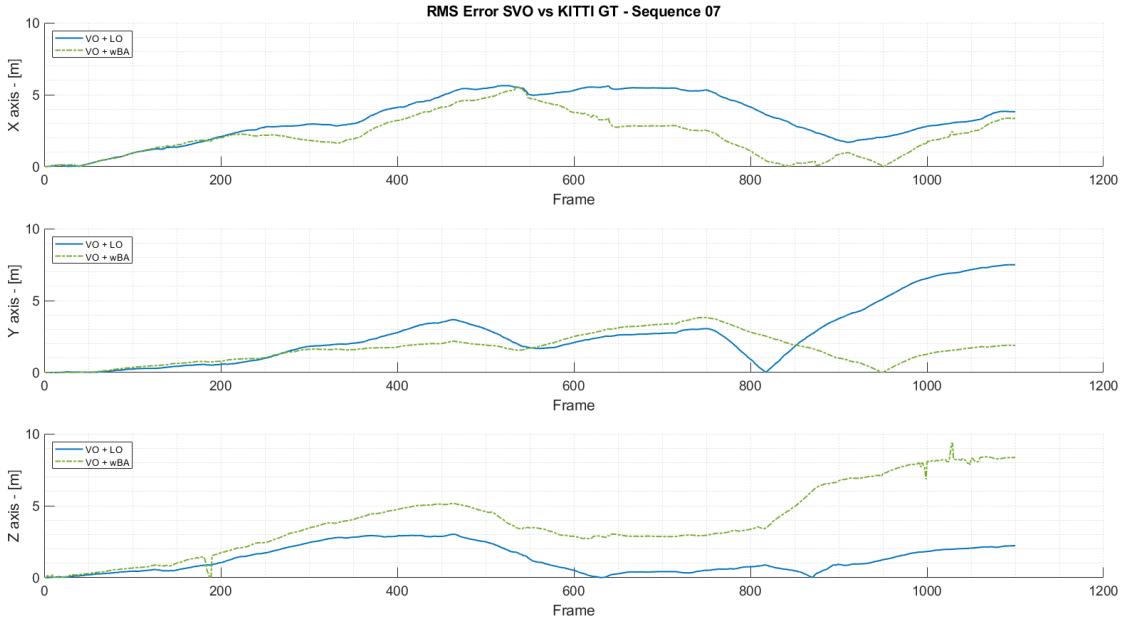


Figure 60. KITTI Sequence 07 – RMS Error in XYZ axis.

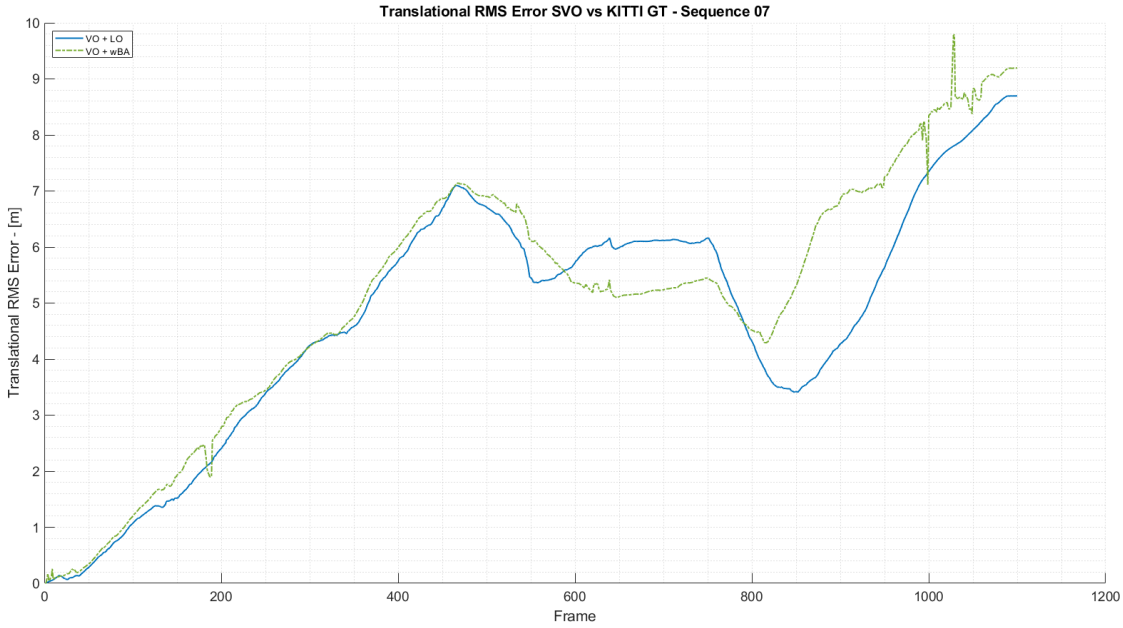


Figure 61. KITTI Sequence 07 – Translational RMS Error.

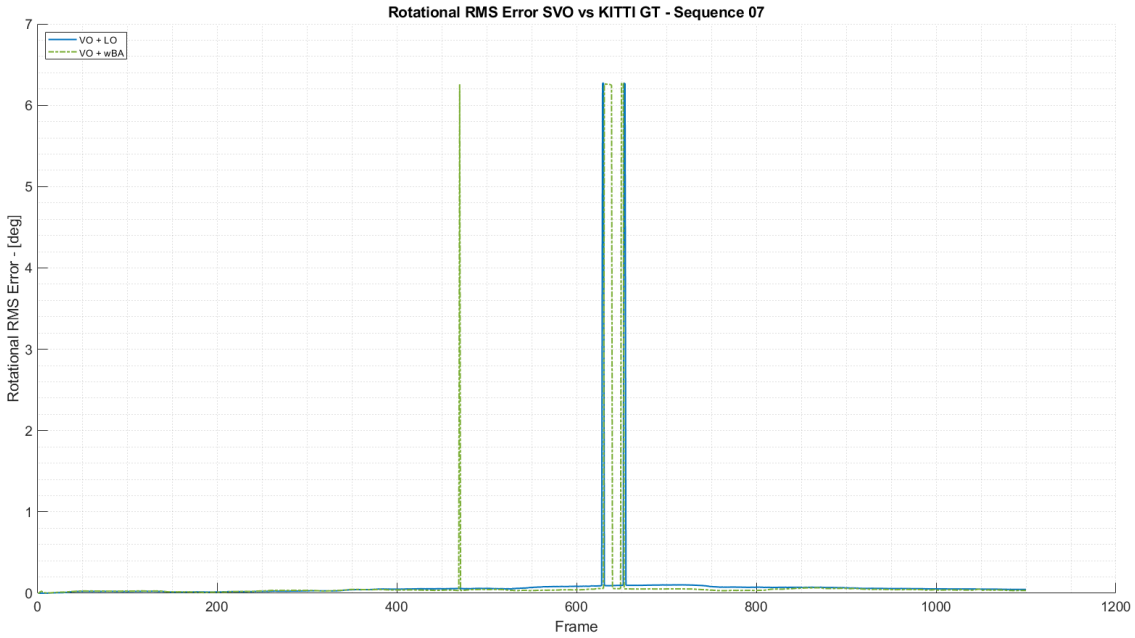


Figure 62. KITTI Sequence 07 – Rotational RMS Error.

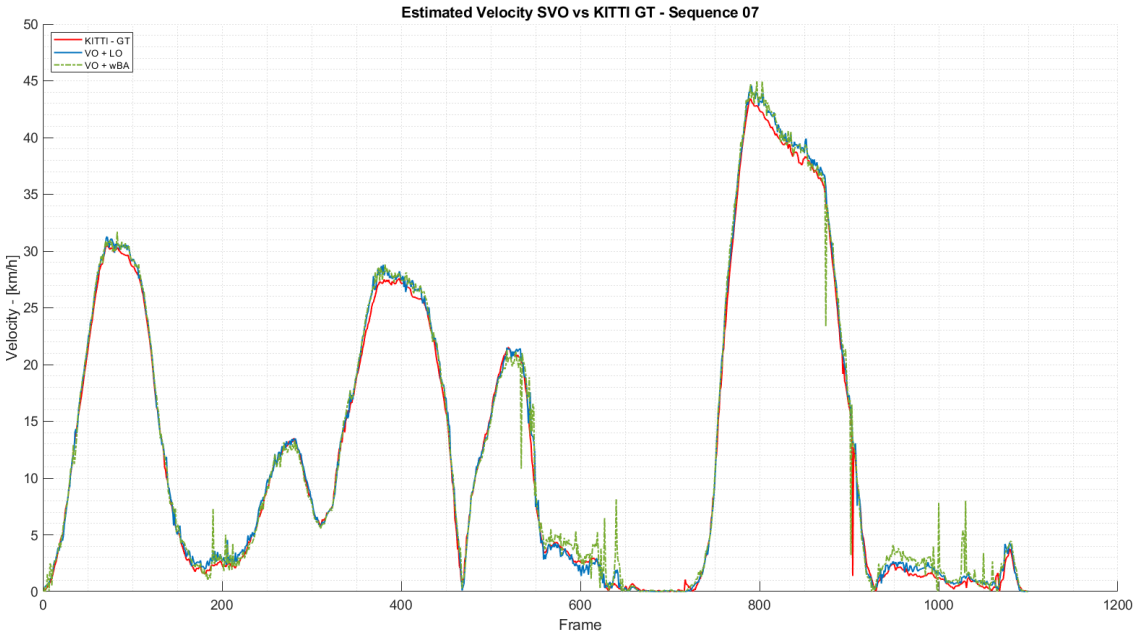


Figure 63. KITTI Sequence 07 – Estimated Velocity.

7.1.5 KITTI Dataset – Sequence 09 Results

Sequence 09 is a close loop sequence and hybrid environment in a forest area and a suburb residential area in the city of Karlsruhe. Table 20 shows a summary of the results presented in Figures 64 to 69.

Table 20. KITTI Sequence 09 Results Summary.

Parameter:	Units	KITTI - GT	SVO + LO	VO + BA
Total Displacement	$[m]$	1007.75	1039.00	1118.67
Mean Velocity	$[km/h]$	22.81	23.52	25.32
Max Velocity	$[km/h]$	45.59	51.12	54.63
Mean Translational RMS Error	$[m]$	N/A	14.3826	89.5589
Mean Rotational RMS Error	$[deg]$	N/A	0.0935	0.4903

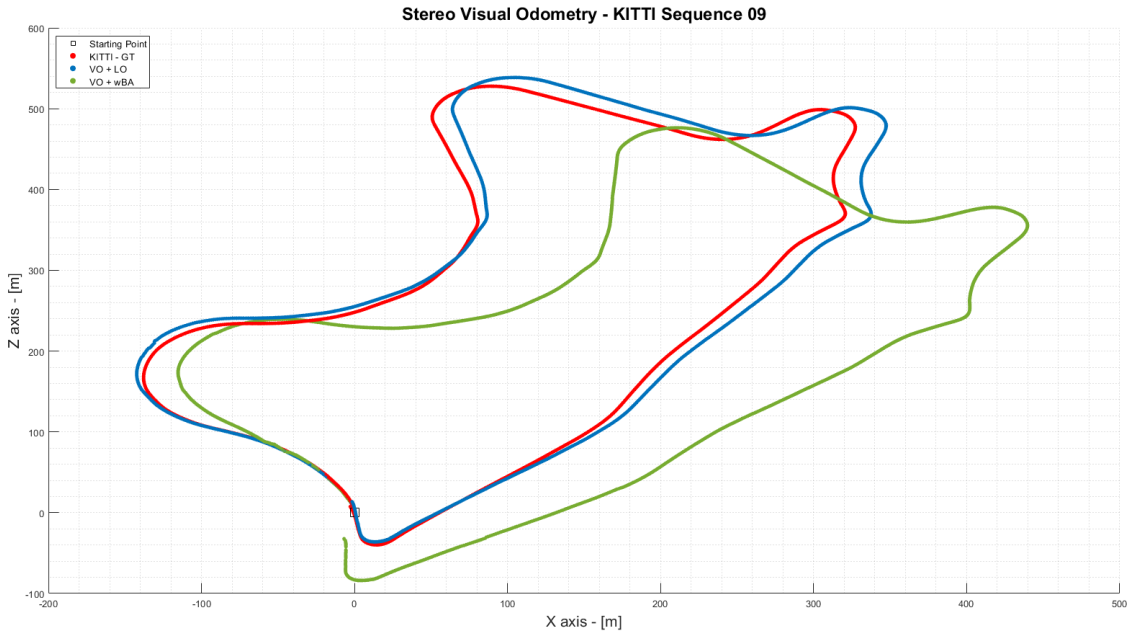


Figure 64. KITTI Sequence 09 – Top View Trajectory.

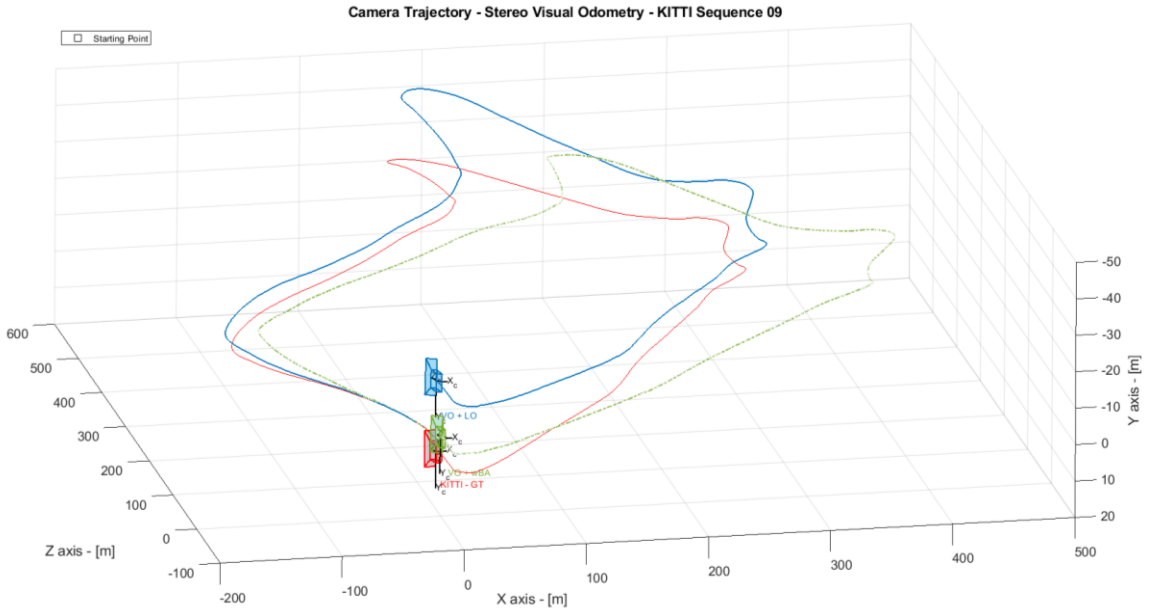


Figure 65. KITTI Sequence 09 – Camera Trajectory 3D Space.

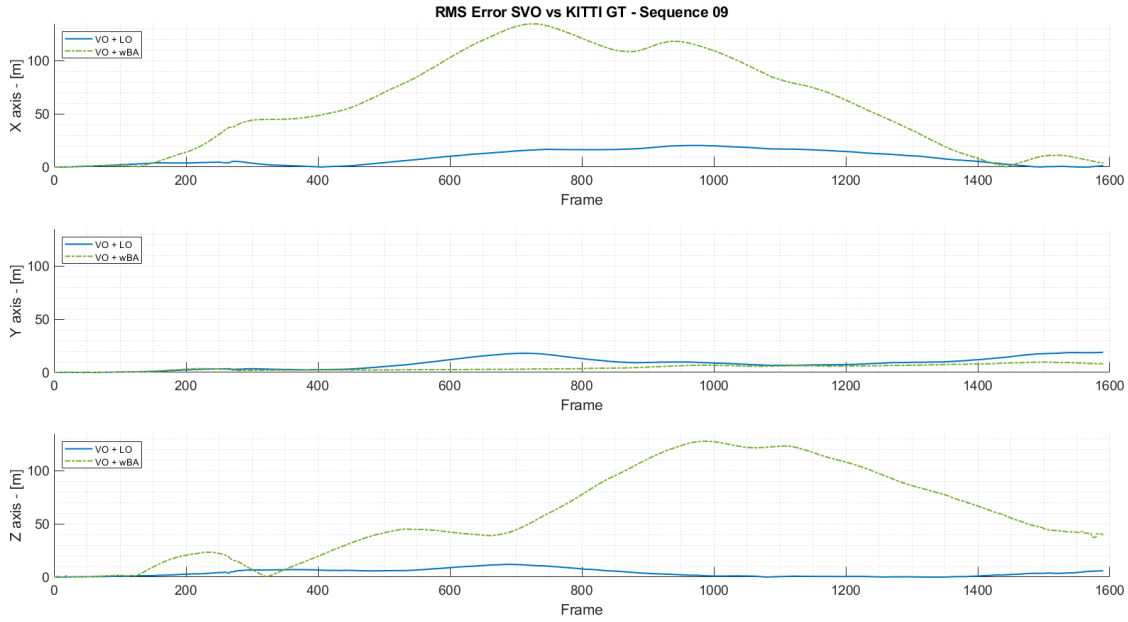


Figure 66. KITTI Sequence 09 – RMS Error in XYZ axis.

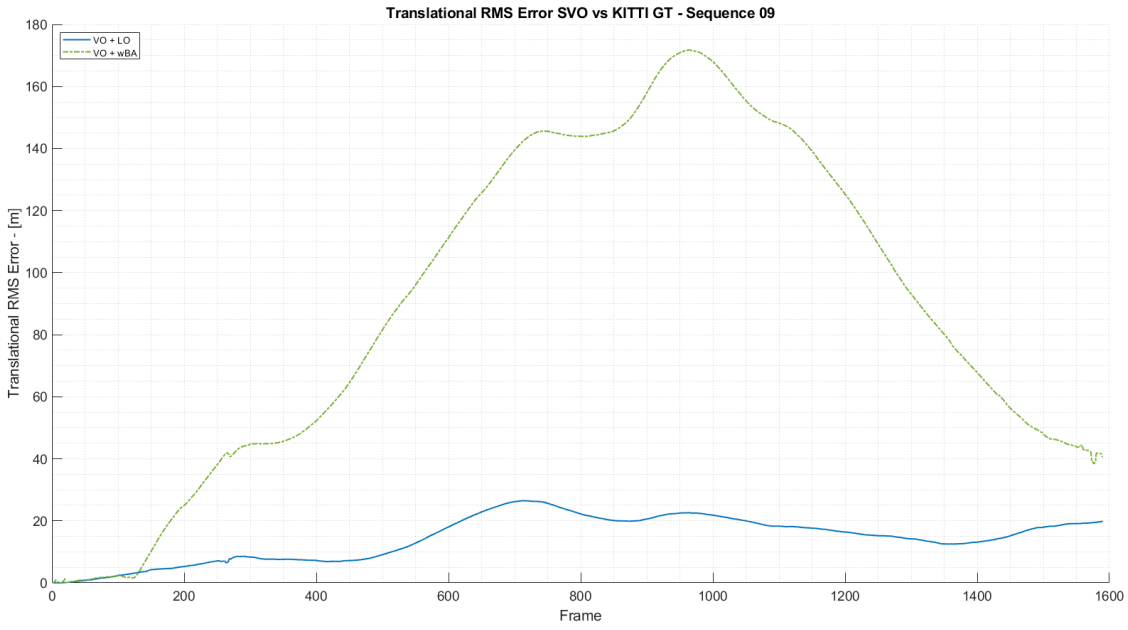


Figure 67. KITTI Sequence 09 – Translational RMS Error.

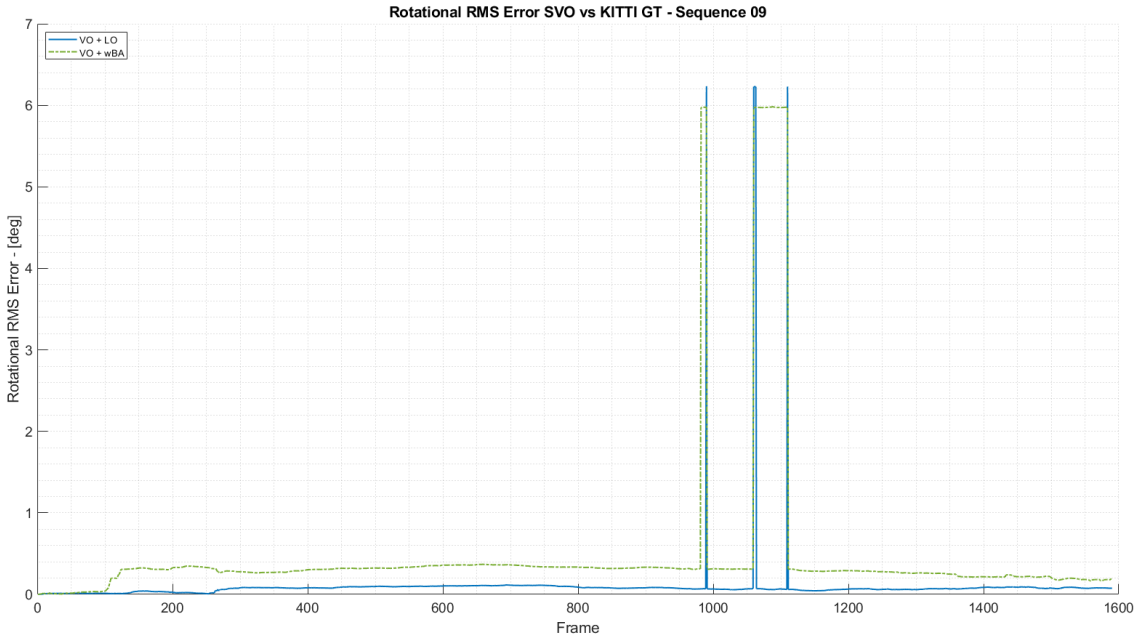


Figure 68. KITTI Sequence 09 – Rotational RMS Error.

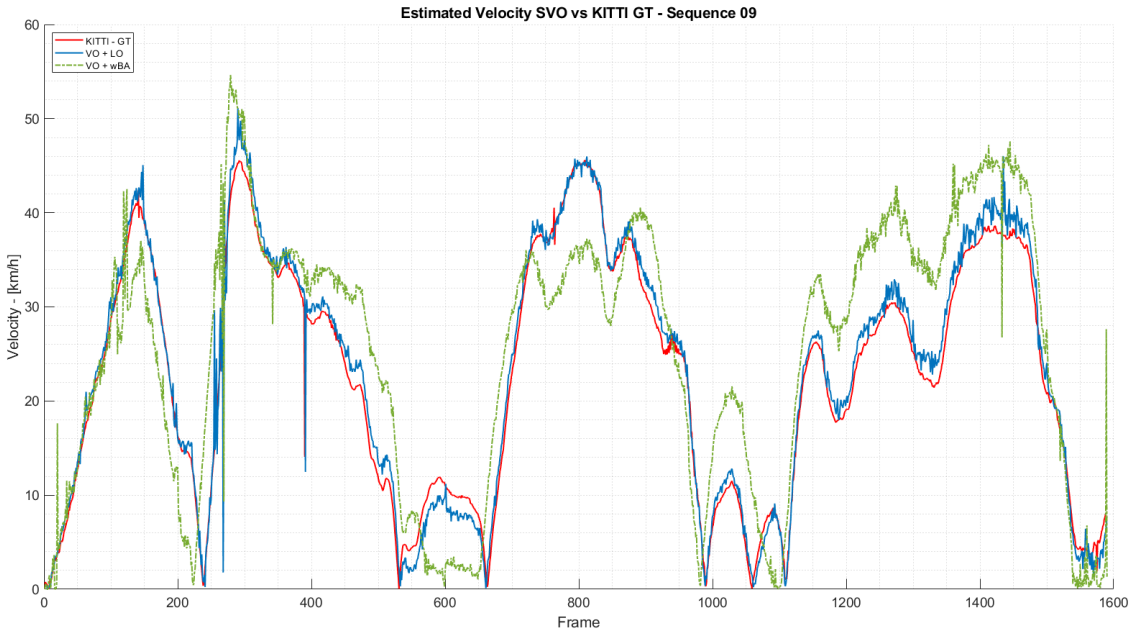


Figure 69. KITTI Sequence 09 – Estimated Velocity.

7.1.6 KITTI Dataset – Sequence 10 Results

Sequence 10 was recorded in the same a forest area and a suburb residential area in the city of Karlsruhe as sequence 09, but this sequence is not a close loop circuit. Table 21 shows a summary of the results presented in Figures 70 to 75.

Table 21. KITTI Sequence 10 Results Summary.

Parameter:	Units	KITTI - GT	SVO + LO	VO + BA
Total Displacement	$[m]$	796.46	816.21	811.89
Mean Velocity	$[km/h]$	23.89	24.48	24.35
Max Velocity	$[km/h]$	54.02	54.76	53.64
Mean Translational RMS Error	$[m]$	N/A	17.13	30.8816
Mean Rotational RMS Error	$[deg]$	N/A	0.0802	0.1452

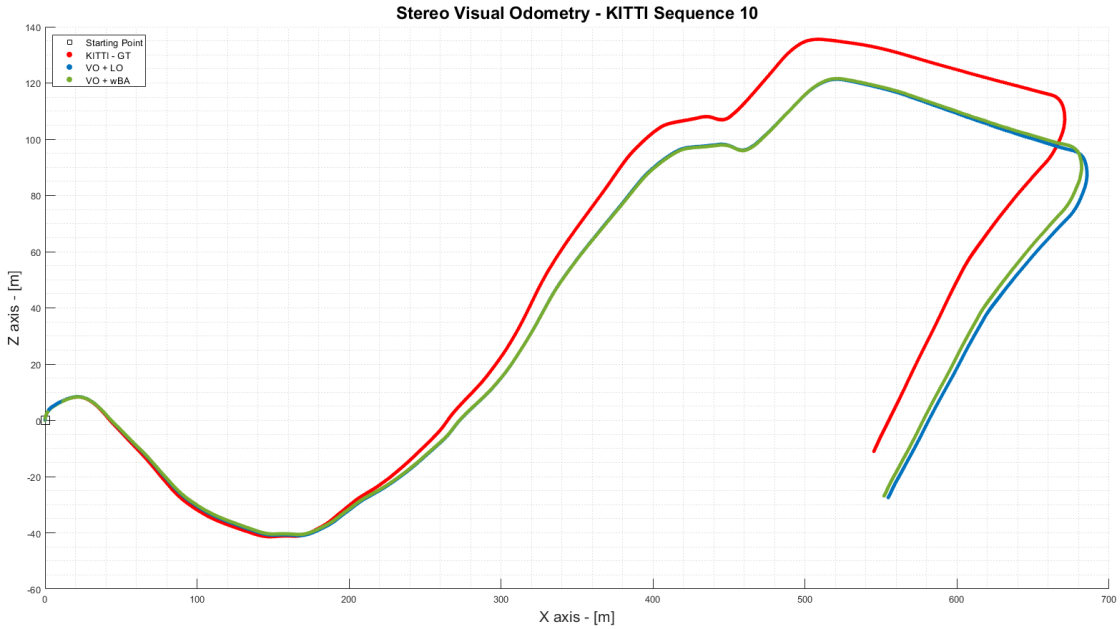


Figure 70. KITTI Sequence 10 – Top View Trajectory.

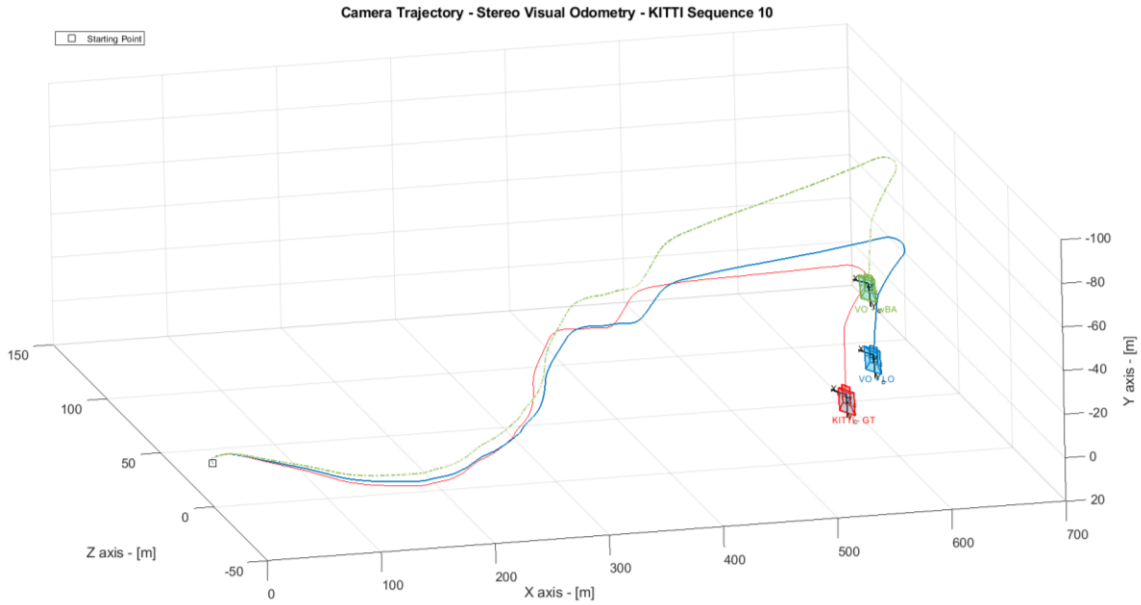


Figure 71. KITTI Sequence 10 – Camera Trajectory 3D Space.

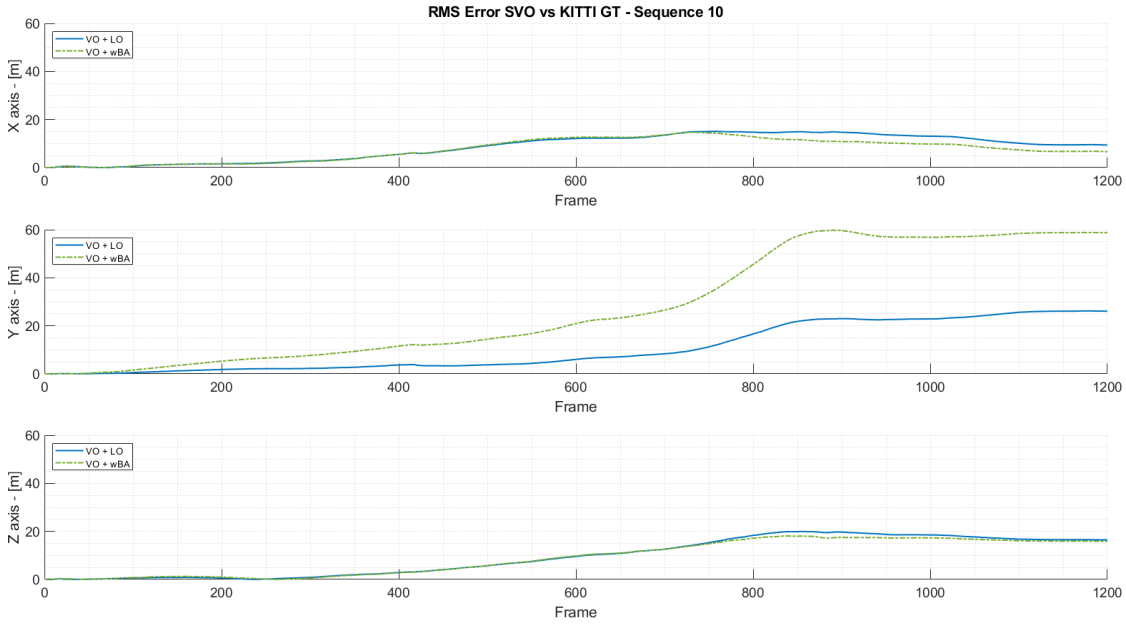


Figure 72. KITTI Sequence 10 – RMS Error in XYZ axis.

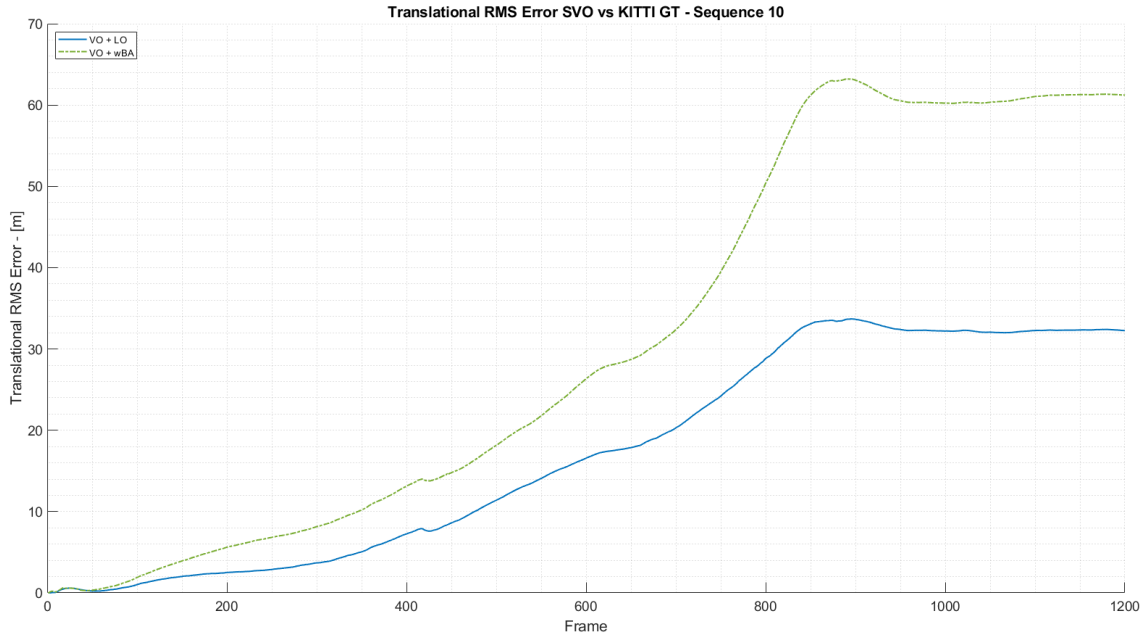


Figure 73. KITTI Sequence 10 – Translational RMS Error.

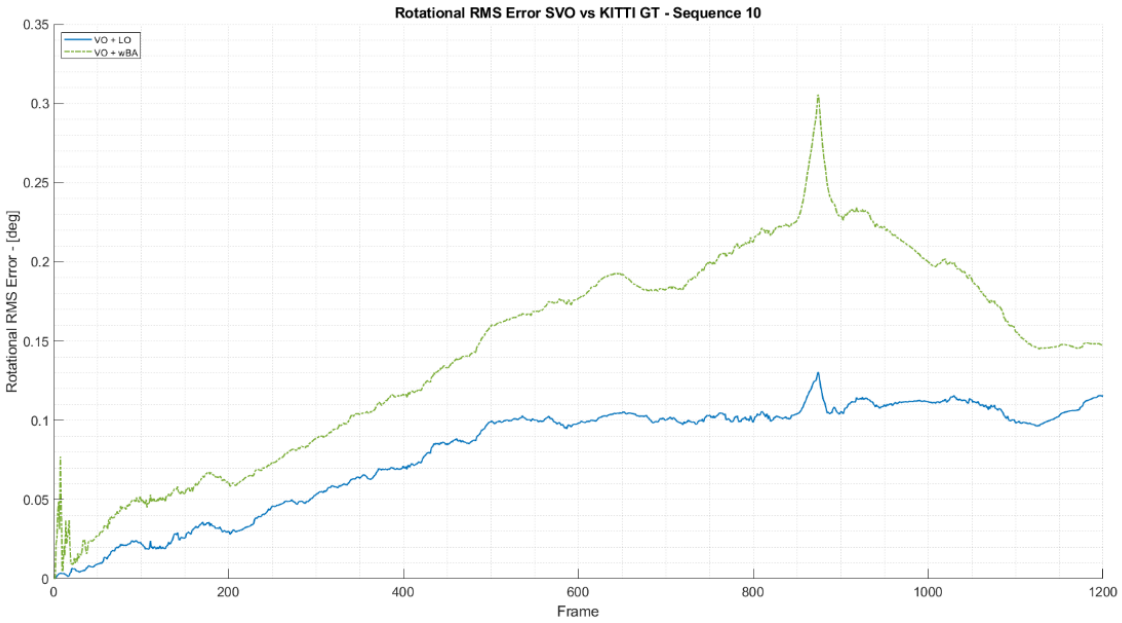


Figure 74. KITTI Sequence 10 – Rotational RMS Error.

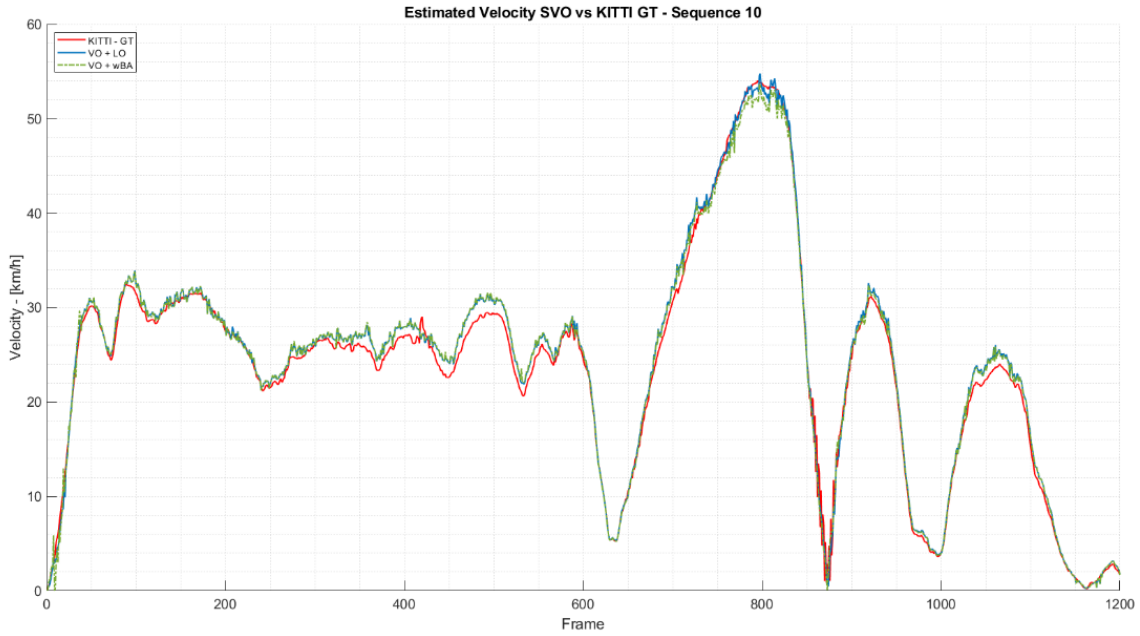


Figure 75. KITTI Sequence 10 – Estimated Velocity.

7.2 ZED and ZED-Mini Visual Odometry Dataset

The KITTI vision odometry benchmark suit allowed for the assessment and testing of the system, which has returned good results in short (less than 500 meters) and medium (around 1500 meters) travelled distances. However, the depth estimation study has constrained the system to rely only on landmarks below a specific threshold, which for the purposes of this work 25 meters in order to ensure the attainment of good depth estimation measurements.

In this section, the author's dataset results are presented. The sequences of videos recorded in the city of Ottawa, Ontario, followed the same methodology than the KITTI dataset. The ZED and the ZED-Mini stereo cameras were mounted on a vehicle, as shown in Figure 62, which was driven around a residential area in the city of Ottawa. In contrast to the KITTI dataset, there was no GNSS and LiDAR assistance; for this reason, the results presented in this thesis are only the sequences of videos and their respective visual odometry results and trajectory estimation.

Figure 76. Data Acquisition with ZED and ZED-Mini.

7.2.1 Methodology of Evaluation

The ZED and ZED-Mini Dataset were evaluated in the same fashion as the methodology presented for the KITTI Vision Benchmark Dataset. The results presented in the following sections were obtained under the same parameters than the ones used for the KITTI dataset with a few modifications in terms of the maximum depth threshold, and the algorithm to obtain depth estimation.

Depth estimation was performed using the Direct Disparity method, instead of using the DLT. This change was made based on the experimental depth estimation evaluation for the ZED and ZED-Mini stereo cameras discussed in Section 4.5.

The dataset contemplates indoors and outdoors environments, where the indoors sequence takes place in a living room, and the outdoors sequences are performed on roads just as the ones provided by the KITTI dataset.

The outdoors sequences were recorded at an average speed of $25\ km/h$ and the frequency rate for the reading of each frame was set to $6\ Hz$. The parameters for both types of sequences (indoors and outdoors) are listed in Table 22.

The process of features extraction from the stereo images was conducted by the SURF algorithm to then be matched by using FLANN. The 5-Point RANSAC algorithm was used in two different stages in order to remove outliers: 1) at the matching keypoint over the stereo images stage; and 2) after the tracking features with the Lucas-Kanade algorithm from a previous frame to the current one.

For the purposes of this evaluation, the Sparse Stereo Vision Odometry with Local Non-Linear Least-Squares Optimization algorithm is the only algorithm performed.

Table 22. ZED and ZED-Mini Dataset Evaluation Parameters

Parameter	Algorithm / Description
2D Features and Descriptors	SURF
Feature Matcher	FLANN
Feature Tracker	Lucas-Kanade
Dual Tracking	Yes
Outlier Removal	5-Point-RANSAC
Outlier Threshold	0.8 pixels
Depth Estimator	DD
Reprojection Error Threshold	0.5 pixels
Max Depth Threshold	25 meters
Motion Estimation	P3P-RANSAC
P3P Reprojection Threshold	0.8 pixels
Windowed BA	N/A
Local Optimization	Yes

7.2.2 ZED Dataset – Sequence 01 Results

The path chosen for the Sequence 01 of the dataset was located in a residential area in the neighborhood of Overbrook, Ottawa, as shown in Figure 77. The path describes a close loop squared trajectory starting at the circled area and follows the directions shown in the diagram in red arrows.

Figure 77. ZED and ZED-Mini Dataset Sequence 01 Path.

This sequence was recorded using both cameras (ZED and ZED-Mini) and the videos were recorded in two different resolutions, FHD (1080 pixels) and HD (720 pixels), respectively.

In Figure 78 and 79, a sample of the image of the ZED dataset sequence 01 is observed. Figure 78 shows the rectified image corresponding to the left lens of the stereo camera, and Figure 79 corresponds to the stereo image pair captured by the stereo system.

Figure 78. Left Lens View – ZED and ZED-Mini Dataset Sequence 01 Sample.

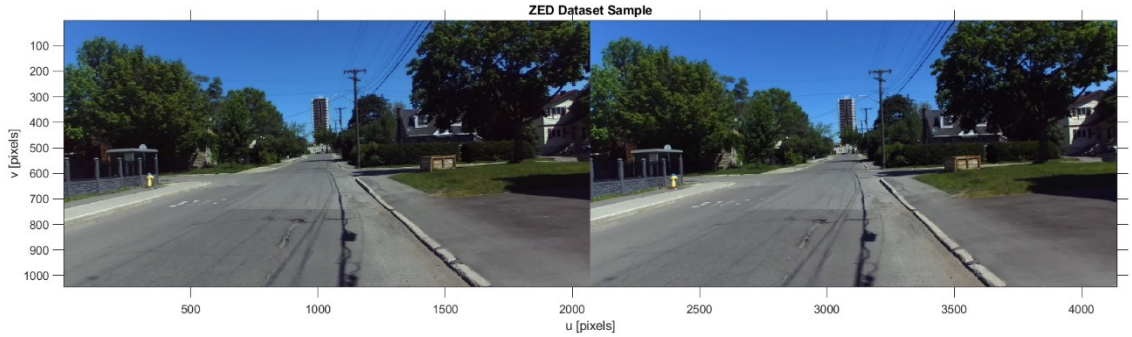


Figure 79. Stereo Image Pair View – ZED and ZED-Mini Dataset Sequence 01 Sample.

In the same fashion, Figure 80 shows a sample of the FLANN matching keypoint algorithm over the stereo image pair, which was used in this evaluation.

Figure 80. SURF Keypoints Matched over the Stereo Image Pair using FLANN – ZED and ZED-Mini Dataset Sequence 01.

Table 23 shows a summary of the results of the ZED and ZED-Mini Visual Odometry evaluation.

Table 23. ZED and ZED-Mini Sequence 01 Results Summary.

Parameter:	**Units**	**ZED 120 mm 1080p**	**ZED 120 mm 720p**	**ZED-Mini 63 mm 1080p**
Total Displacement	$[m]$	291.29	347.44	284.18
Mean Velocity	$[km/h]$	9.72	7.28	8.27
Max Velocity	$[km/h]$	29.66	18.43	34.02

All the figures are presented in meters unless otherwise specified. The graphs are presented in the following order: 1) Top view of the trajectory; 2) Camera trajectory in a 3D space; and 3) Estimated Velocity.

The results presented in the following figures have the following parameters:

- ZED camera (baseline of 120 mm) at FHD (1080 pixels) resolution.
- ZED camera (baseline of 120 mm) at HD (720 pixels) resolution.
- ZED-Mini camera (baseline of 63 mm) at FHD (1080 pixels) resolution.

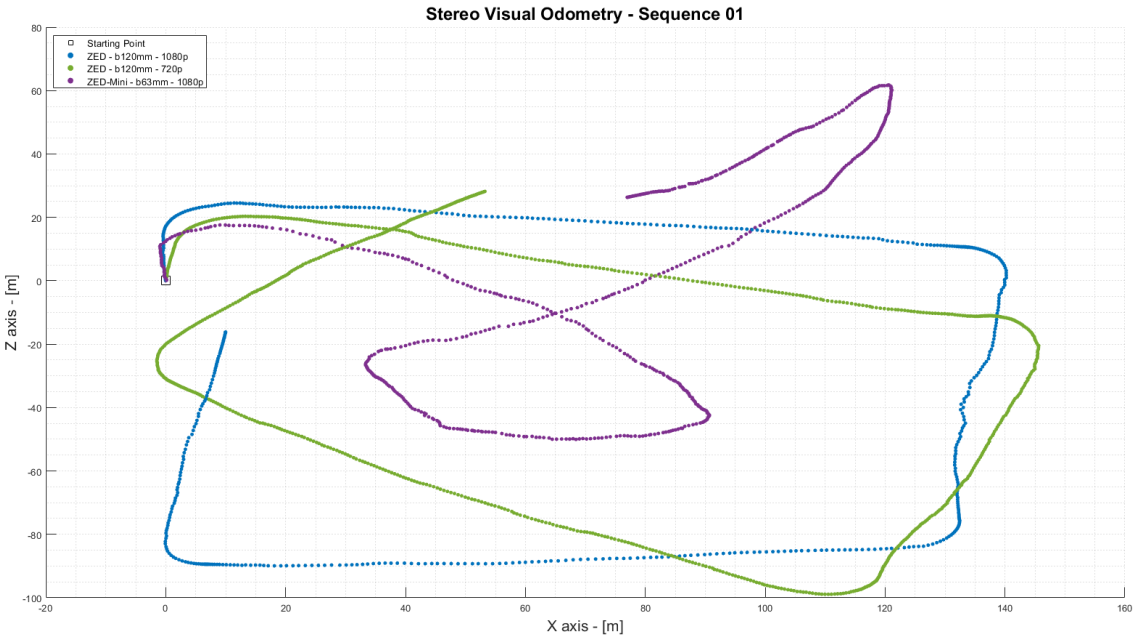


Figure 81. Top View Trajectory – ZED and ZED-Mini Dataset Sequence 01.

The results presented in Figure 81 and 82 show that the only sequence that actually followed the real trajectory was the sequence recorded at 1080p with the ZED stereo camera with baseline of 120 millimeters, where the estimated trajectory was close to the real path traveled by the vehicle. However, the system still drifted in the y-axis.

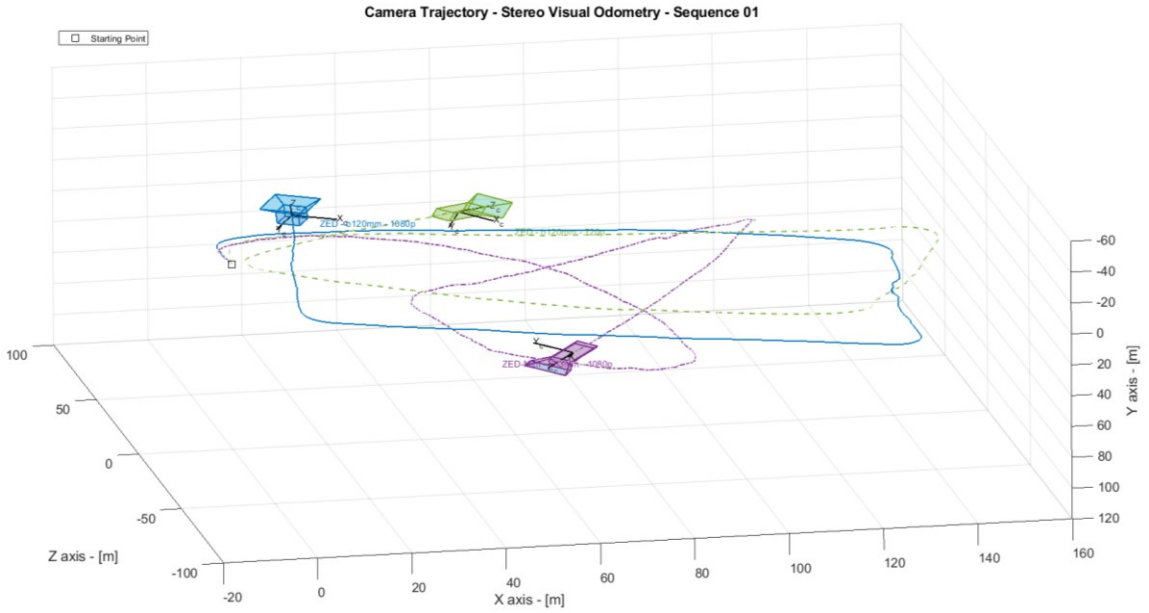


Figure 82. Camera Trajectory – Isometric View – ZED and ZED-Mini Dataset Sequence 01

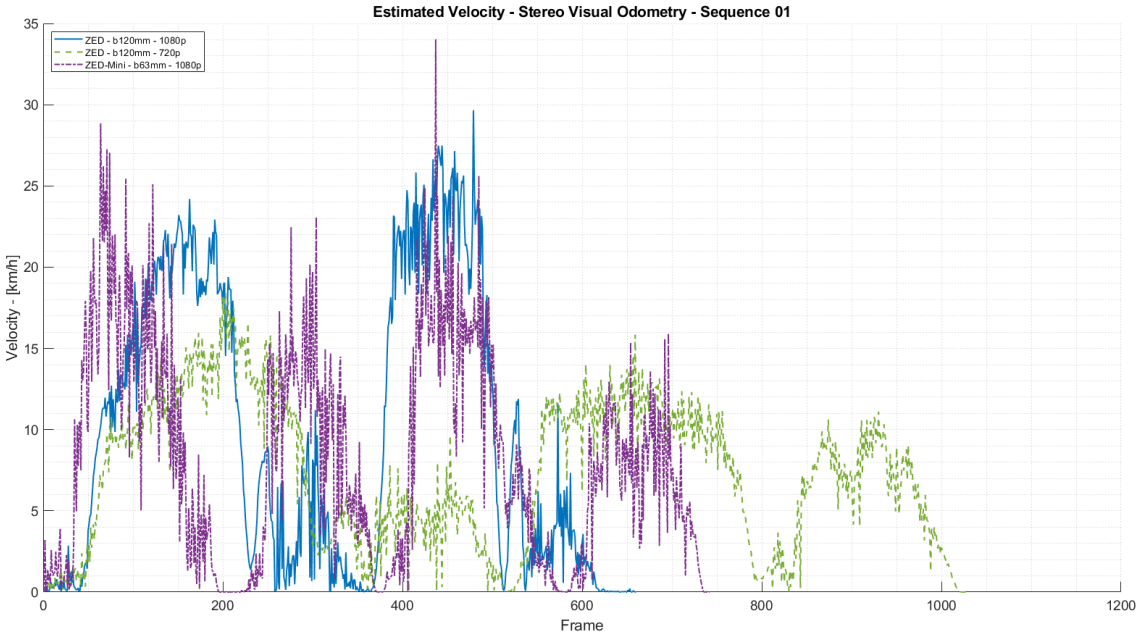


Figure 83. Estimated Velocity – ZED and ZED-Mini Dataset Sequence 01.

On the other hand, the sequence recorded with the same ZED camera but at a lower resolution of 720p drifts from the real path traveled by the vehicle. This might have happened because of the reduction in terms of resolution from 1080p to 720p. The reduction in resolution is directly proportional to the information available for feature extraction over the image plane. Thus, the less pixels available for feature extraction, the less the points of interest available for feature detection, and therefore, motion estimation got impacted due to the lower number of feature samples.

Finally, the ZED-Mini results showed that the estimated trajectory drifts considerably with respect to the real path traveled by the vehicle. The reason for this might be the reduction of baseline from 120 mm to 63 mm, which reduces the capability to compute a good depth estimation for keypoints located far away from the camera, which in turn directly impacts over the motion estimation algorithm.

Nevertheless, the results from the ZED at FHD resolution are close to the real trajectory, even though it still drifts over time. Since the system does not have its own GNSS and LiDAR sensors acquisition data available, it is hard to assess the actual error of the system.

Finally, the following figures show the estimated trajectory and the sparse 3D scene reconstruction (point cloud) that the system was able to reconstruct from the computed landmarks used in the motion estimation algorithm used in the video sequence ZED – b120mm – 1080p.

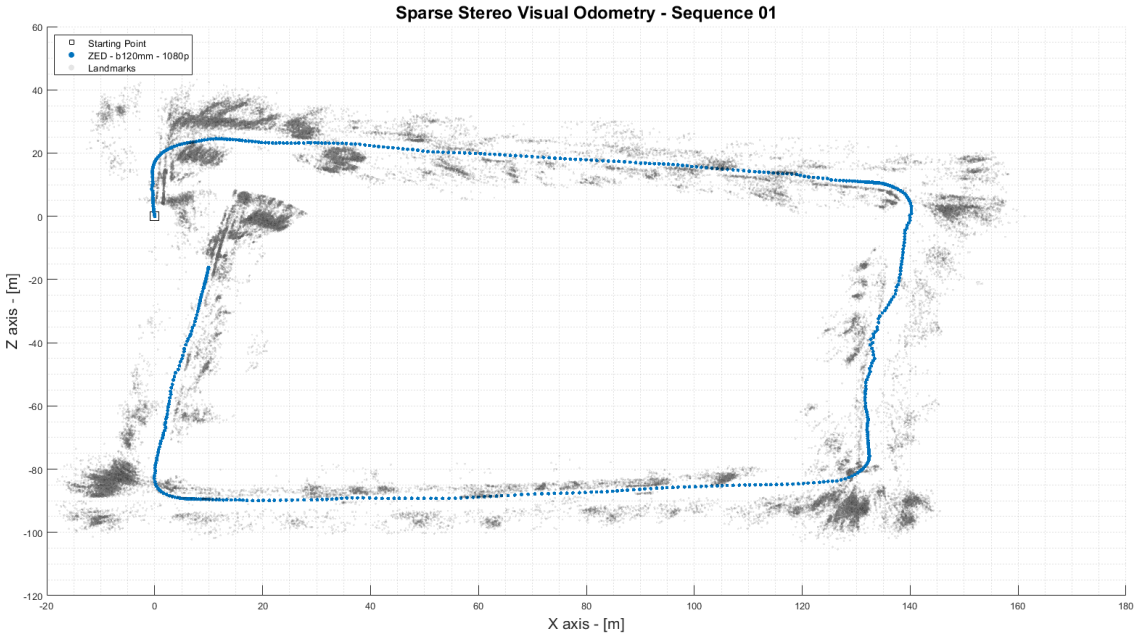


Figure 84. Visual Odometry and Sparse Map – Top View – ZED and ZED-Mini Dataset Sequence 01.

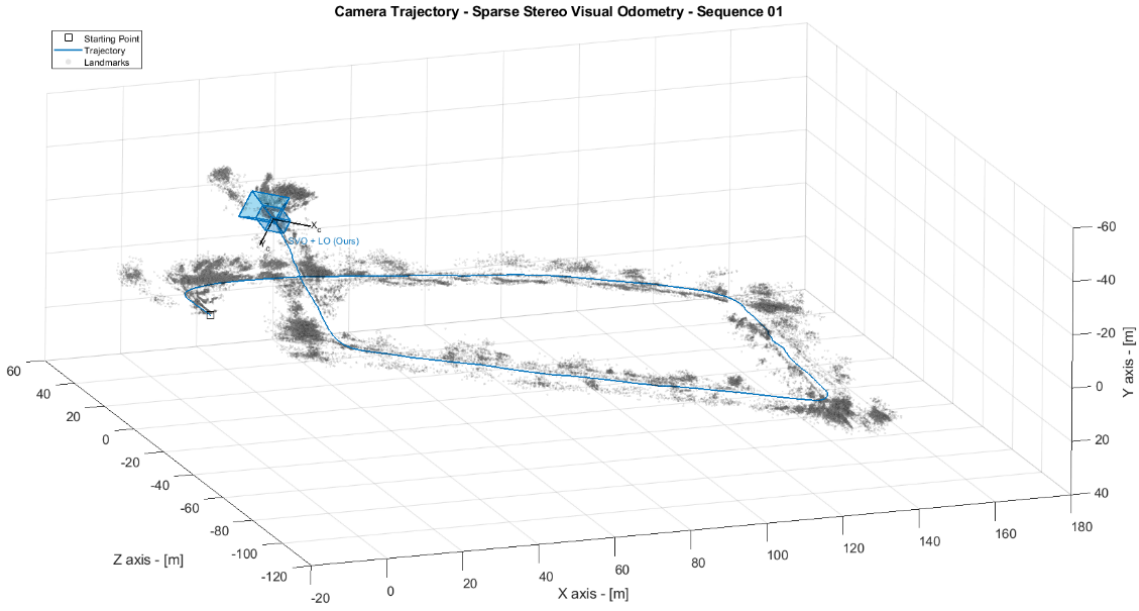


Figure 85. Visual Odometry and Sparse Map – Isometric View – ZED and ZED-Mini Dataset Sequence 01.

7.2.3 ZED Dataset – Sequence 02 Results

The sequence 02 of the ZED and ZED-Mini dataset was recorded indoors. In this sequence, only the ZED-Mini dataset was used, and the video was recorded with an FHD (1080 pixels) resolution.

The sequence shows a living room and its main purpose is to build a sparse map of the camera environment surroundings. Figure 86 shows a sample of the environment, and Figure 87 shows the stereo image pair captured by the ZED-Mini camera.

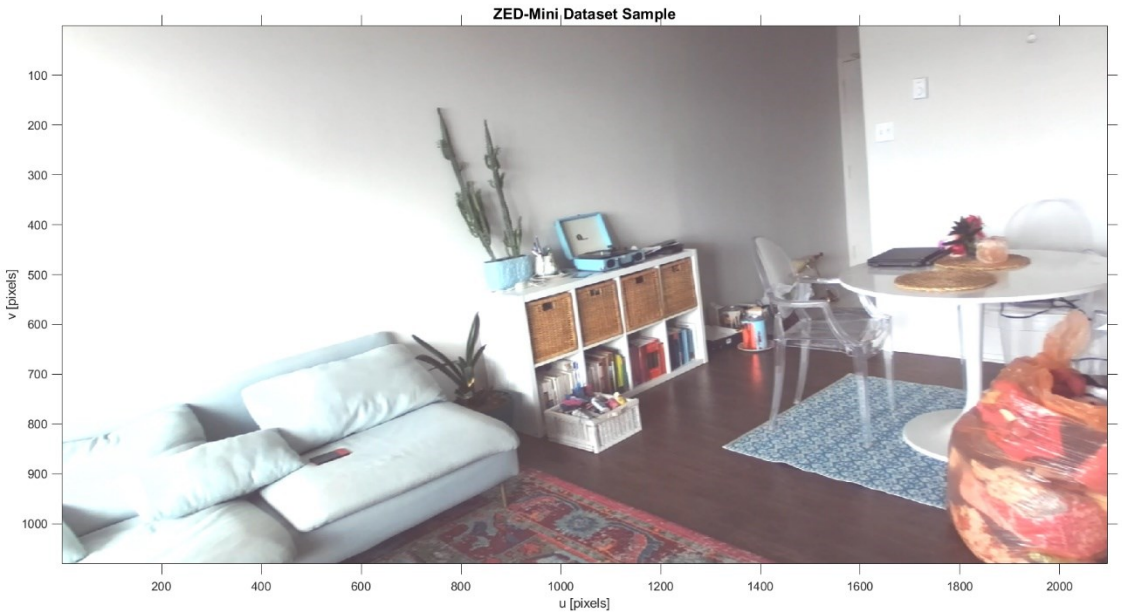


Figure 86. Left Lens View – ZED-Mini Dataset Sequence 02 Sample.

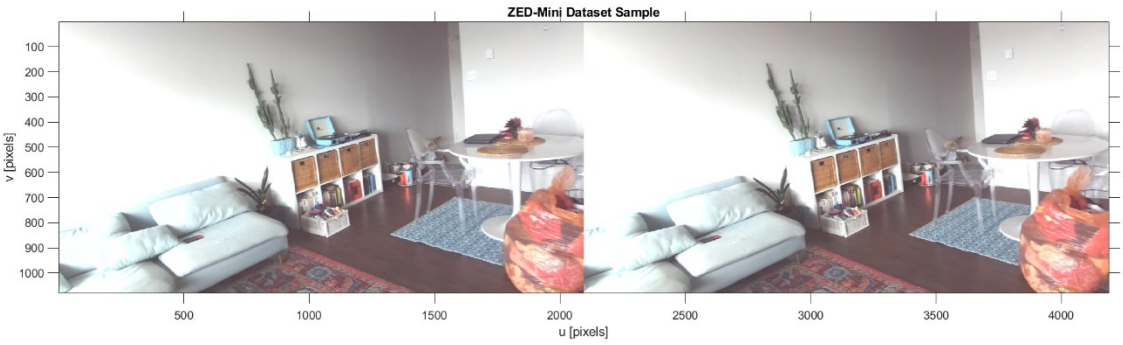


Figure 87. Stereo Image Pair View – ZED-Mini Dataset Sequence 02 Sample.

In the same fashion, Figure 88 shows a sample of the FLANN matching keypoint algorithm over the stereo image pair, which was used in this evaluation. Notice how the keypoints are concentrated over the drawers and over the carpets on the floor. This is due do the high gradient contrast on the surfaces of those objects, which make them a perfect scenario for SURF detector. In the other hand, the walls, which are white, show minimal to no keypoints.

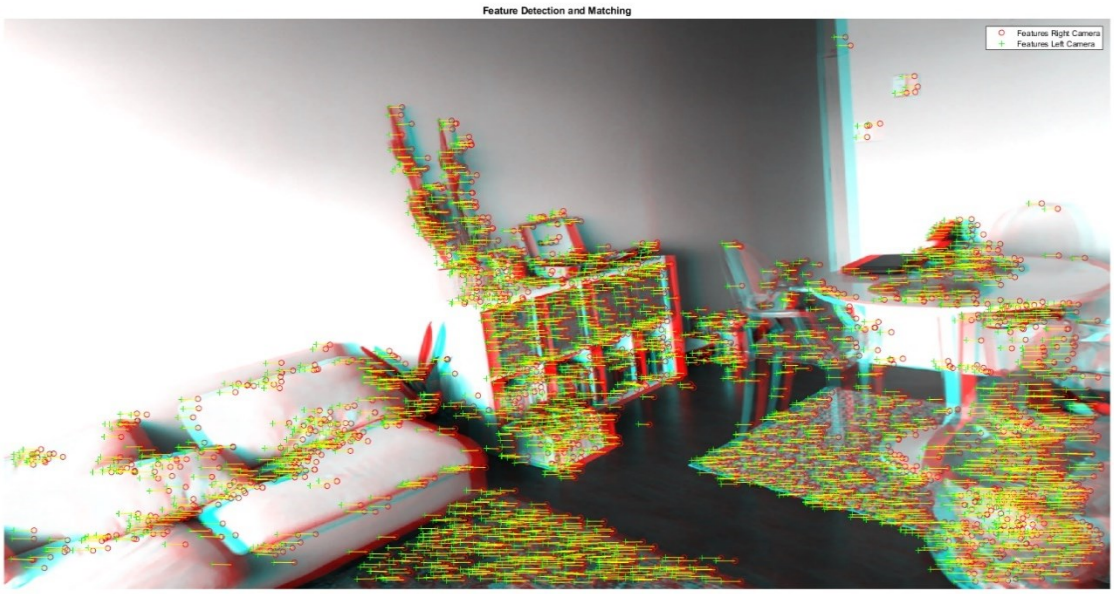


Figure 88. SURF Keypoints Matched over the Stereo Image Pair using FLANN – ZED Dataset Sequence 02.

Figure 89 shows the trajectory traveled by the ZED-Mini during the video recording. It is hard to assess if the estimated trajectory actually followed the real trajectory of the camera, because there is no other sensor attached to the camera and thus, there is no comparison in terms of traveled distance. On the other hand, the trajectory seems quite close to the one that it can be appreciated on the video sequence.

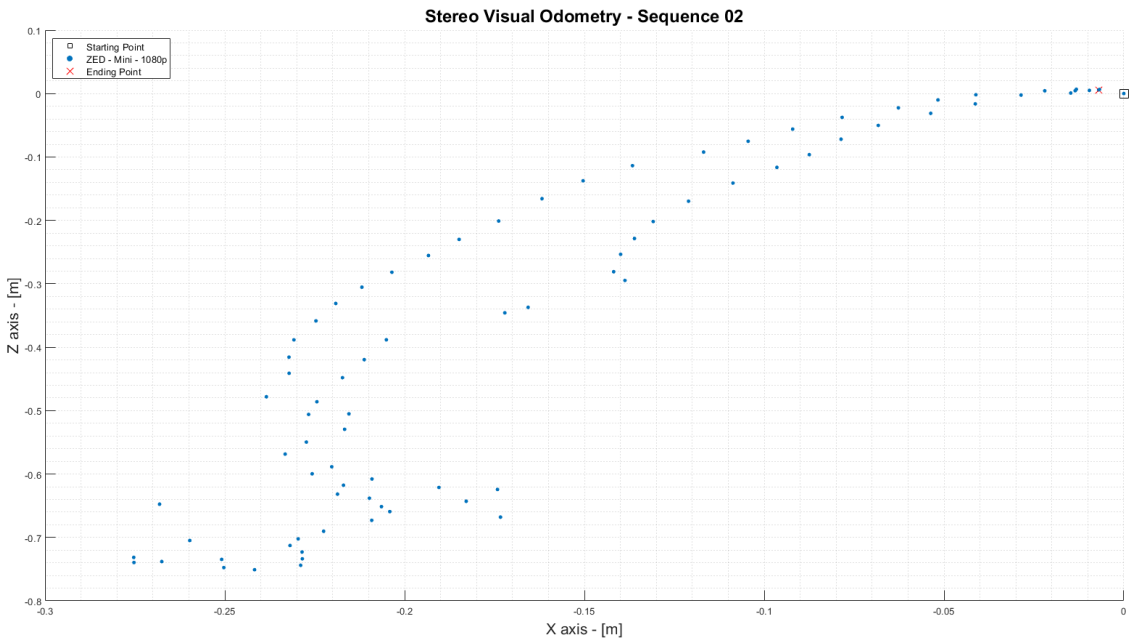


Figure 89. Visual Odometry – ZED and ZED-Mini Dataset Sequence 02.

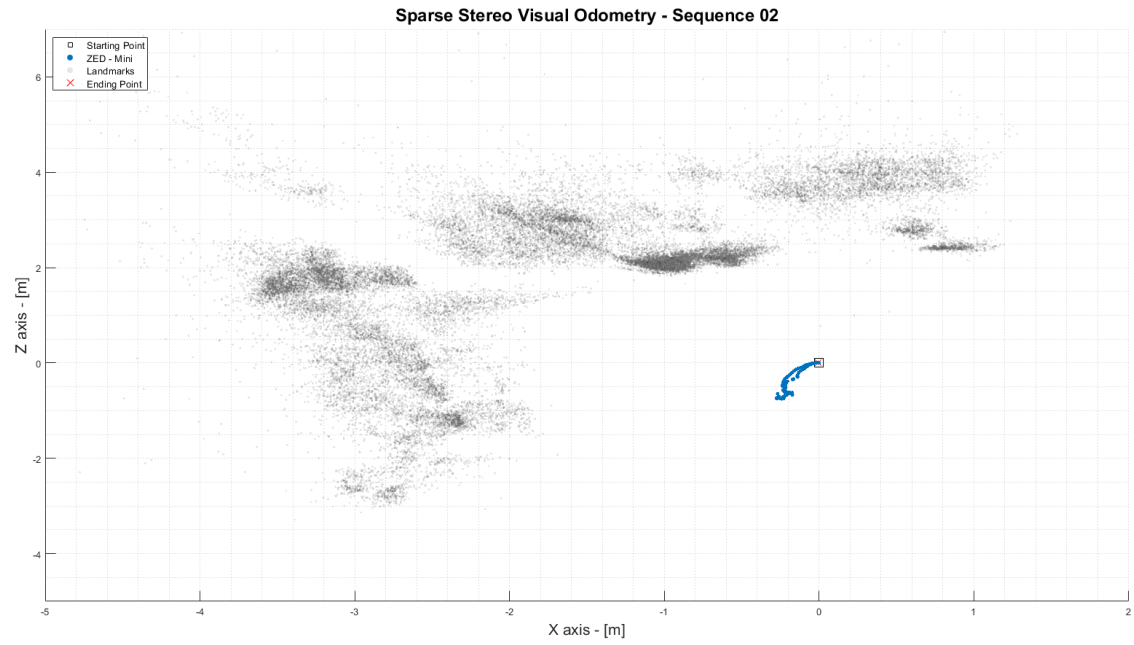


Figure 90. Visual Odometry and Sparse Map – Top View –

ZED and ZED-Mini Dataset Sequence 02.

Figure 90 and 91 show the path of the camera and the sparse map built out of the landmarks considered over the processing of the sequence. Figure 90 shows a top view of the scene, meanwhile Figure 91 presents an isometric view of the scenario.

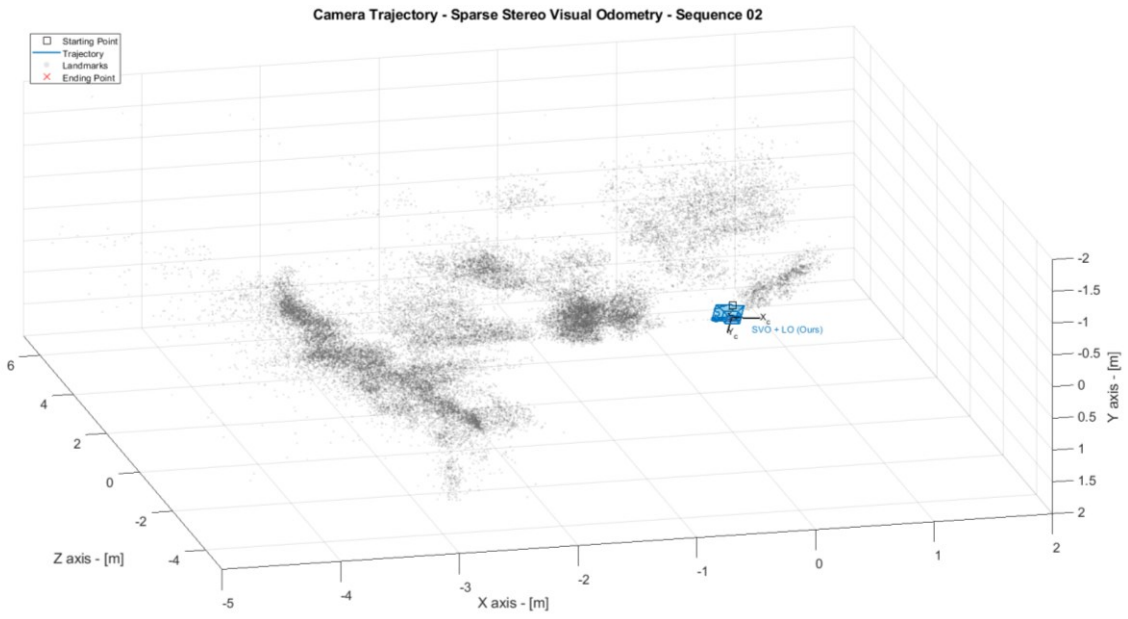


Figure 91. ZED-Mini Visual Odometry and Sparse Map – Isometric View – ZED and ZED-Mini Dataset Sequence 02.

7.2.4 ZED Dataset – Sequence 03 Results

The sequence 03 in the ZED and ZED-Mini dataset was recorded outdoors during the Winter of the year 2019 in the city of Ottawa, ON. This sequence was recorded only with the ZED camera at FHD (1080 pixels) resolution. The path chosen for this sequence was outside Carleton University campus, specifically Colonel By Drive, as shown in Figure 92. The sequence starts on the red circled area and travels in the direction marked with the red arrows until reaching the end point, marked as well in a red circle area.

Figure 92. ZED Dataset – Sequence 03 Path.

Figure 93 shows a sample of the left lens captured on the video sequence recorded in a snow environment, and Figure 94 shows the stereo image pair sample.

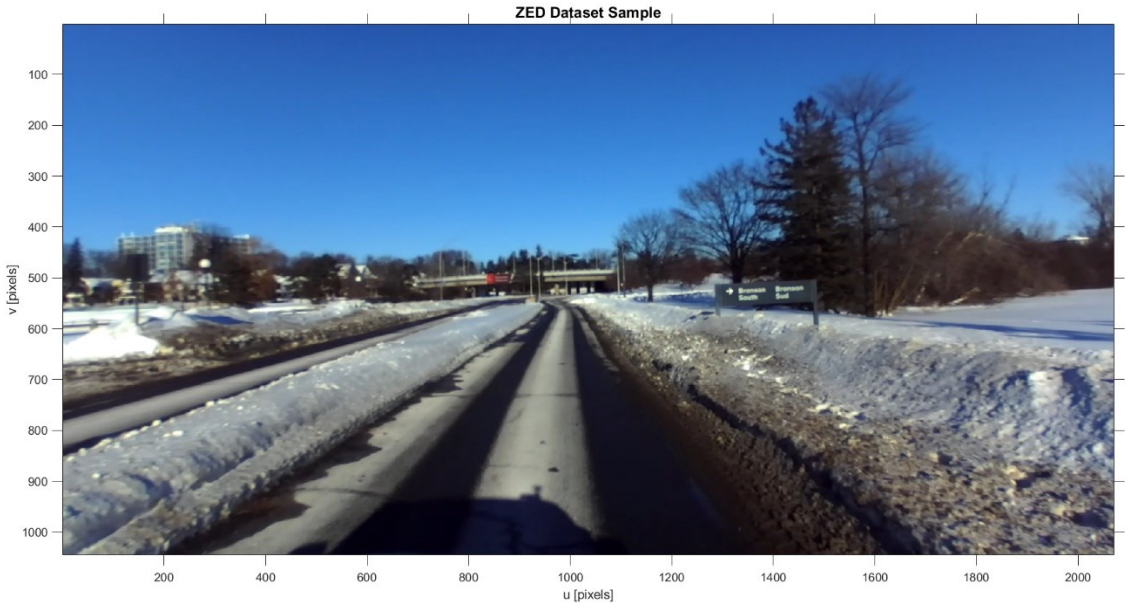


Figure 93. Left Lens View - ZED Dataset Sequence 03 Sample.

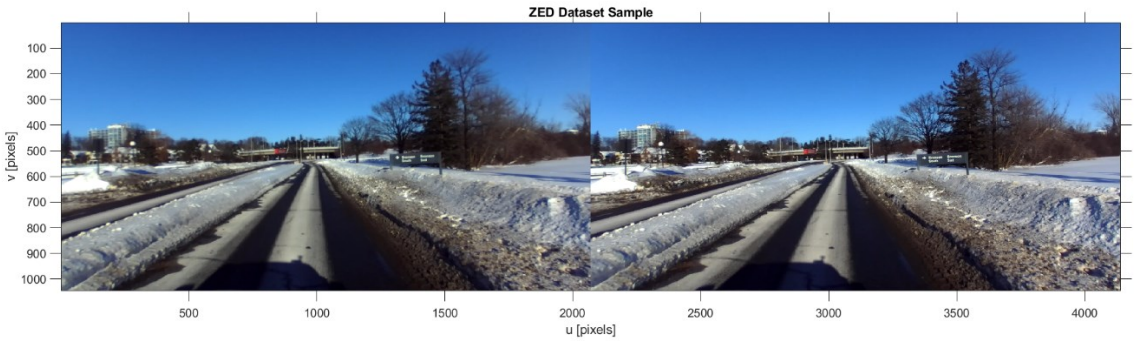


Figure 94. Stereo Image Pair View - ZED and ZED-Mini Dataset Sequence 03 Sample.

In the same fashion described above, Figure 95 shows the SURF keypoints detected and matched using FLANN over the stereo image pair. Notice how the snow which was removed and piled out of the road works well for features detection, but the wet road is not a good scenario for SURF detector. This is due to the reflection of the sunlight over the asphalt, and also the predominant dark color which does not provide gradient changes on the gray scale images.

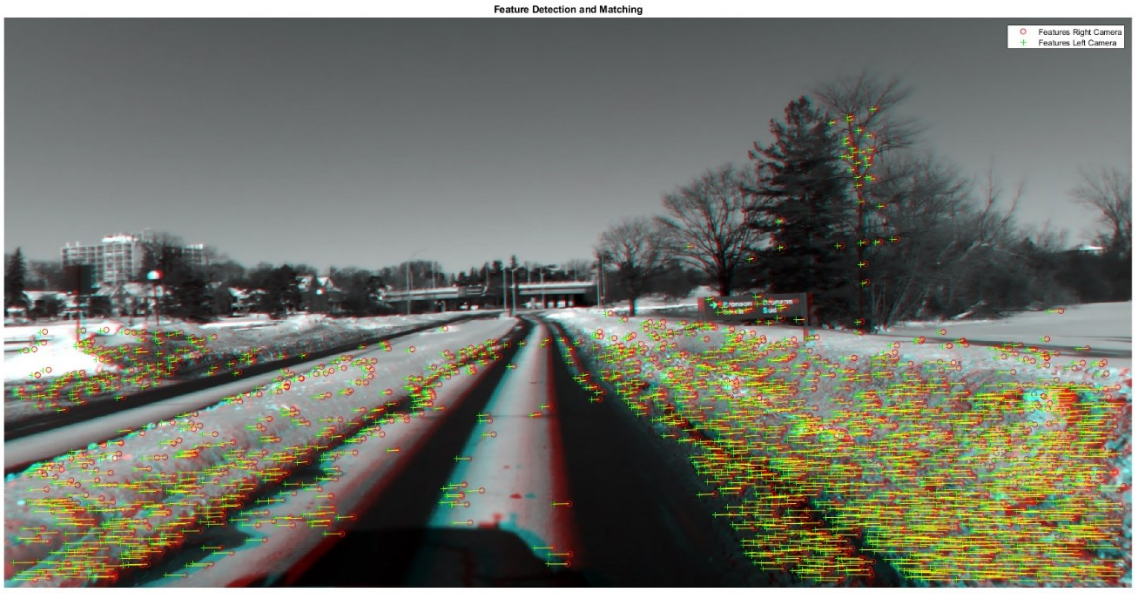


Figure 95. SURF Keypoints Matched over the Stereo Image Pair using FLANN – ZED and ZED-Mini Dataset Sequence 03.

Table 24 summarizes the odometry results for the sequence 03 of the ZED and ZED-Mini dataset evaluation.

Table 24. ZED Odometry Sequence 03 Results Summary.

Parameter:	Units	ZED - b120 mm 1080p
Total Displacement	$[m]$	497.49
Mean Velocity	$[km/h]$	10.18
Max Velocity	$[km/h]$	26.5

Figure 96 shows the visual odometry results for sequence 03, where it is possible to observe that the camera follows the trajectory presented in Figure 92, but at the end of the trajectory, the results drifted from the real path. One reason why this might have happened is the lack of sunlight at the end of the video sequence, since it is crucial to keep good illumination in the surrounding environment.

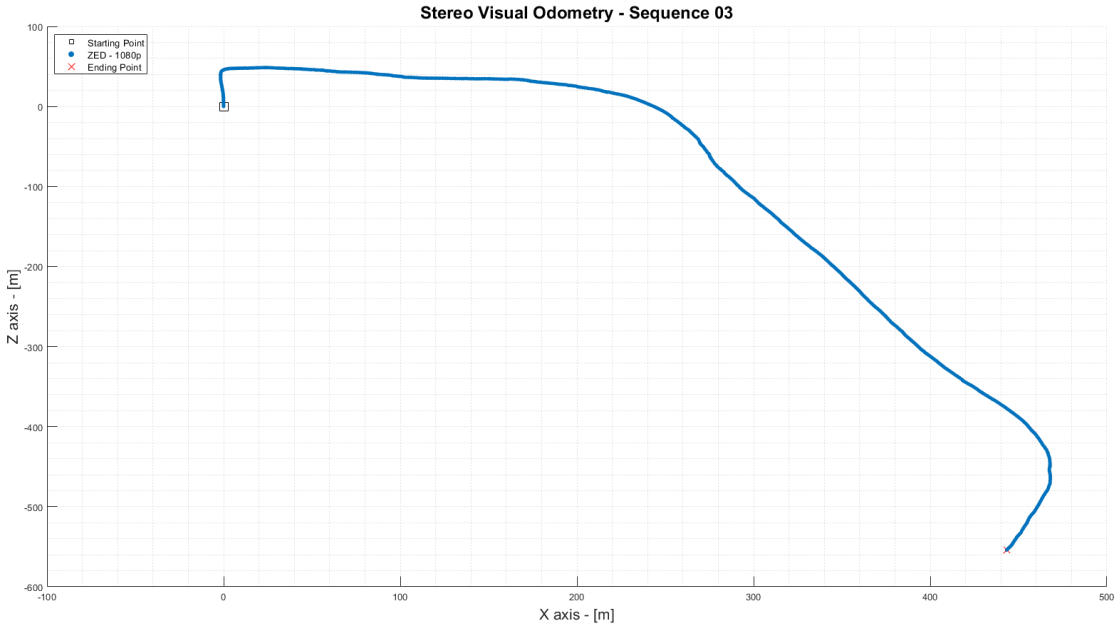


Figure 96. Visual Odometry – ZED and ZED-Mini Dataset Sequence 03.

Finally, Figure 97 shows the sparse map built from the landmarks considered during the processing of this video sequence.

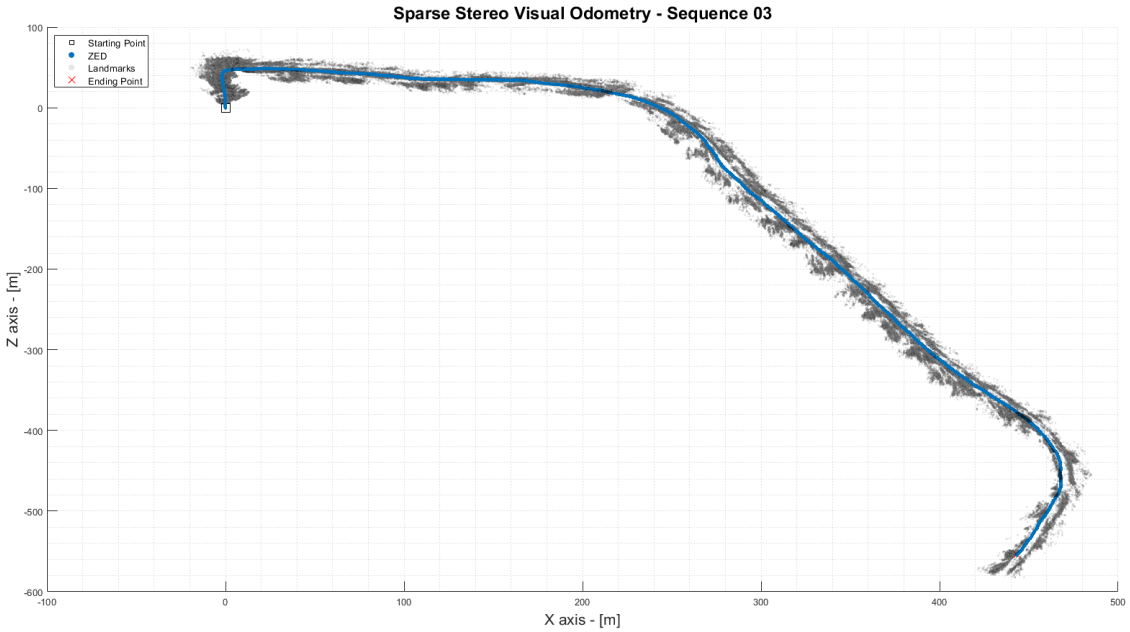


Figure 97. Visual Odometry and Sparse Map – Top View – ZED and ZED-Mini Dataset Sequence 03.

Chapter 8

Future Work

In the Stereo Visual Odometry with Local Non-Linear Least-Squares Optimization System presented in this thesis, the only input data are the stereo video sequences that rely on the camera parameters extracted from the camera calibration procedure. However, plenty of other Visual Odometry (VO) applications integrate the fusion of multiple sensors.

The most important task is to improve the camera calibration algorithm to achieve millimetric precision in measurements from the camera system. Also, the integration of an Inertial Measurement Unit (IMU), as well as a wheel odometry sensor, will improve the results of the system in terms of translation and rotation. Even though the sensors previously mentioned might improve the results of the system, it would be ideal to also integrate a Global Navigation Satellite System (GNSS) and a Light Detection and Ranging (LiDAR) to achieve the full autonomy of the vehicle.

Since in this thesis work, the system was tested only during the day and with sufficient sun light illumination, it would be interesting to test the system into night or low illumination conditions. The main point to address here would be to assess if the system is capable to yield to reasonable results with an artificial source of light placed in front of the vehicle in such a way that helps the camera to perceive the world in low illumination scenarios.

Finally, this system was not tested in real-time. Most of the VO applications are required to perform in real time; for this reason, the next step is to optimize the software and hardware so it can be able to run it in real time.

Chapter 9

Conclusions

In this thesis, a fully functional Stereo Visual Odometry (SVO) system with a Local Non-Linear Least-Squares Optimization for navigation of autonomous vehicles (AVs) was presented. This work has introduced a simple but innovative alternative to the classic Visual Odometry (VO) algorithm with Windowed Bundle Adjustment (BA) optimization approach.

The system presented in this work was coded and implemented in the software MATLAB 2019a (at its latest version) with a wrapper interface to OpenCV (C++) libraries. It works in the same fashion as the Sparse Visual Odometry algorithm, its main contributions being that: (1) the system carefully track features (keypoints) in both left and right lenses so as to assess that the set of set of keypoints and set tracked features indeed are outlier free; (2) motion estimation or camera pose (translation and rotation) is computed via the Perspective-three-Point (P3P) algorithm, to later on be optimized by expressing the motion orientation (Euler's angels) output from the P3P solver into quaternions so as to avoid the *'gimbal lock'* problem and; (3) in contrast to the windowed bundle adjustment (that keeps track of key-frames and the information in terms of keypoints to perform the optimization of the camera motion in the mentioned window of frames), this system performs a local optimization frame by frame minimizing the reprojection error of three-dimensional (3D) landmarks into two-dimensional (2D) image features into both stereo

image frames, e.g. every instant of time that a frame is captured, a new camera motion is estimated and that new camera pose is optimized.

This thesis work was evaluated and successfully tested by using the Karlsruhe Institute of Technology and Toyota Institute (KITTI) Vision Benchmark Suit dataset. The robustness of the system was compared and tested against the ground truth (camera poses) data provided by the KITTI dataset, which is the trajectory of the vehicle obtained by the fusion of a Velodyne Light Detection and Ranging (LiDAR) and a Global Navigation Satellite System (GNSS). The system results were also compared against the classic visual odometry algorithm with windowed bundle adjustment (at 10 windowed frames).

As a result of the validation of the experiments conducted with the KITTI dataset, it was observed that the proposed VO system with local optimization presented results closer to the ground truth data than the ones extracted from the VO with windowed BA. Also, the estimated traveled distances and the estimated velocity were similar to the trajectories provided on the KITTI dataset.

In terms of translation and orientation optimization, the results provided by the system proposed in this thesis (VO with local optimization) showed that the translation and orientation RMS errors are smaller compared to the RMS errors from the results from VO with windowed BA.

In short, the results were evaluated by calculating the RMS error of the system's output against the ground truth camera's poses provided on the KITTI dataset, giving as result good performance on short distances (less than 500 meters) to medium distances (in the range of 500 meters to 1500 meters). For distances above 1.5 kilometers, the system drifts from the ground truth trajectory because of the inherent error propagation imposed

by the uncertainty of the transformations between two frames (instants of time) and the concatenation with previous camera locations.

This thesis has also contributed by providing a study about how the lack of high precision camera parameters and the change of baseline of the stereo system impact over the depth perception measurements and their performance on applications for visual odometry.

Furthermore, in this thesis was demonstrated that for targets, potential keypoints, or points of interest that are far away (25 meters or beyond) from the stereo system, the output depth estimation yielded to erroneous values that did not match the reality, where the farthest the points of interest are the less realistic the results are. It is of high importance for visual odometry systems to determine the depth estimation threshold in order to avoid miscalculations on camera motion, which impact directly on the estimated motion of the system and over the estimated traveled trajectory.

In doing so, this thesis work presented a fieldwork study to evaluate the depth estimation output of two commercial stereo cameras, namely the ZED and the ZED-Mini stereo cameras, both manufactured by Stereolabs. In this study, it was observed that for points of interest beyond 25 meters, depth estimation differs from realistic values. In the other hand, points of interest that are relatively close (25 meters or less) to the stereo system yield to more realistic results. This was the case for both cameras, even though their baselines changed in a ratio of 0.525; ZED camera has a baseline of 120 millimeters, whereas ZED-Mini has a baseline of 63 millimeters.

The piece also reviewed and analyzed the advantages of the Harris corner detector over the Speed Up Robust Features (SURF) detector in terms of key-point extraction for

static applications, e.g. the camera is in a fixed position with respect to the world scene. Even though the Harris corner detector provides a good feature extraction for interest key points that are far away from the camera system, its disadvantage of being non-scale invariant makes non a good feature detector for applications such as visual odometry. On the other hand, SURF has the advantage of being scale invariant and also rotation invariant, which makes it an ideal feature extractor for visual odometry systems.

As result of this study, it was determined that the maximum depth estimation threshold for the ZED and ZED-Mini stereo cameras to be used in visual odometry application, must be less than 25 meters in order to yield reasonably good results.

Finally, the Stereo Visual Odometry system proposed in this work was successfully tested on a dataset acquired in the roads of the city of Ottawa, Ontario, by replicating the methodology utilized on the KITTI dataset and recording video sequences with the two aforementioned commercial stereo cameras, the ZED and the ZED-Mini stereo cameras. The sequences were recorded by driving in a neighborhood of the city at noon time in a sunny day in order to avoid the sun occlusion, which is known for inducing noise in camera vision systems. One of the sequences was recorded indoors to assess the functionality of the system in terms of sparse mapping. And the final sequence was recorded in an uncommon snow weather condition to validate if the system was capable to still yield to good results.

Over the results of the evaluation of the system over the video sequences recorded in Ottawa was observed that this system is able to perform well with the ZED camera in a full high definition (FHD) resolution of 1080×1920 pixels. Even when the system drifted in

the y-axis, the estimated trajectory was close to the real path traveled by the vehicle (ZED Dataset - Sequence 01).

On the other hand, the sequence recorded with the same ZED camera but a lower resolution of 720×1280 pixels or high definition (HD), showed that the system drifts from the real path traveled by the vehicle. The reason why this happened might be explained by the reduction in terms of pixels available. The less pixels available for feature extraction, the less the information available for depth estimation, which impacts directly on the motion estimation algorithm.

In regards to the ZED-Mini, which sequence was recorded in FHD, the results showed that the estimated trajectory deviates with respect to the real path traveled by the vehicle. The reason for this might be because the reduction of baseline from 120 mm to 63 mm reduces the capability to compute good depth estimation for keypoints that are located far away from the camera.

In short, it is ideal to use the ZED stereo camera at a full high definition (FHD, 1080×1920 pixels) for applications in visual odometry systems, incorporating SURF feature extractor due to its scale invariance. The motion local non-linear least-squares optimization presented in this this work resulted into an alternative to the windowed bundle adjustment optimization.

Appendices

Appendix A ZED and ZED-Mini Technical Specifications

ZED and ZED-Mini technical specifications [76], [77]

Table A.1. ZED and ZED-Mini Camera Specifications.

Camera Specifications	*ZED*	*ZED-Mini*
Output Resolution Side by Side	2x (2208x1242) @ 15fps	2x (2208x1242) @ 15fps
	2x (1920x1080) @ 30fps	2x (1920x1080) @ 30fps
	2x (1280x720) @ 60fps	2x (1280x720) @ 60fps
	2x (640x480) @100fps	2x (640x480) @100fps
Output Format	YUV 4:2:2	YUV 4:2:2
Field of View	Max. 110° (D)	Max. 110° (D)
Depth Range	0.5 m to 15 m	0.15 m to 12 m
Baseline	120 mm	63 mm
Interface	USB 3.0	USB 3.0 Type-C port

Table A.2. ZED and ZED-Mini Electronic Specifications.

Electronic Specifications	*ZED*	*ZED-Mini*
Sensor Type	1/3 in backside illumination sensors with high low-light sensitivity	1/3 in backside illumination sensors with high low-light sensitivity
Active Array Size	4M pixels per sensor	4M pixels per sensor
Focal Length	2.8 mm – f/2.0	2.8 mm – f/2.0
Shutter	Electronic synchronized rolling shutter	Electronic synchronized rolling shutter
Pixel Size	2µm	2µm

Table A.3. ZED and ZED-Mini General Specifications.

General Specifications	*ZED*	*ZED-Mini*
Dimensions	175x30x33 mm	124.5x30.5x26.5 mm
Weight	159 g	62.9 g
Power	380 mA / 5V USB	380 mA / 5V USB
Operating Temperature	0 °C to +45°C	0 °C to +45°C

Appendix B Singular Value Decomposition

Singular Value Decomposition (SVD) is a powerful technique for dealing with sets of equations that are either singular, or numerically close to singular. SVD is the method of choice to solve most of the *linear least-squares* problems [78]. This powerful tool is based on the theorem that every matrix $\boldsymbol{A}$ of size $m \times n$ square or rectangular, can be rewritten as the product of a matrix $\boldsymbol{U}$ of size $m \times n$, a matrix $\boldsymbol{\Sigma}$ of size $n \times n$ with positive or zero elements, and the transpose of a matrix $\boldsymbol{V}$ of size $n \times n$, with the following properties:

- $\boldsymbol{U}$ is column-orthonormal, where the column vectors are all mutually orthogonal unit vectors.
- $\boldsymbol{\Sigma}$ is a diagonal matrix in which the diagonal elements are the singular values σ_i of $\boldsymbol{A}$. The singular values are positive values arranged in descending order lower bounded by zero, and can be computed by solving the characteristic equation of $\boldsymbol{A}$, thus:

$$(\boldsymbol{A}^T\boldsymbol{A})x = \lambda x \tag{B.1}$$

Rewriting equation B.1 such that equal to zero:

$$(\boldsymbol{A}^T\boldsymbol{A} - \lambda\boldsymbol{I})x = 0 \tag{B.2}$$

The characteristic equation of $\boldsymbol{A}$ is then expressed as:

$$\boldsymbol{A}^T\boldsymbol{A} - \lambda\boldsymbol{I} = 0 \tag{B.3}$$

Finally, solving the system of equations for all values of λ yields to the singular values of $\boldsymbol{A}$, thus:

$$\sigma_i = \sqrt{\lambda_i} \tag{B.4}$$

- $\boldsymbol{V}$ is a square orthogonal matrix, where its columns and rows are orthonormal, such that:

$$\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I} \tag{B.5}$$

By applying SVD to a given matrix $\boldsymbol{A}$, it yields:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{B.6}$$

Appendix C ZED and ZED-Mini – Depth Estimation Evaluation and Assessment

The results presented in this appendix are part of the results presented on the Section 4.5 Case of Study: ZED and ZED-Mini – Depth Estimation Evaluation and Assessment.

Table C.1. Depth Estimation Evaluation Results – ZED at 720 pixels - Indoors.

Indoors Depth Estimation Evaluation – ZED – 720p						
Distance $[m]$	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD $[m]$	**DLT $[m]$**	**SMG $[m]$**	**DD $[m]$**	**DLT $[m]$**	**SMG $[m]$**
5	4.99	3.01	4.99	4.99	2.85	5.00
10	9.85	4.07	9.94	9.52	3.91	9.99
15	14.78	4.73	14.92	14.82	4.61	14.89
20	18.13	5.03	19.74	17.08	4.82	18.95
25	20.63	6.01	21.32	19.48	4.99	21.03
30	23.67	6.04	22.05	24.95	5.28	22.25

Table C.2. Depth Estimation Evaluation Results – ZED at 720 pixels - Outdoors.

Outdoors Depth Estimation Evaluation – ZED – 720p						
Distance $[m]$	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD $[m]$	**DLT $[m]$**	**SMG $[m]$**	**DD $[m]$**	**DLT $[m]$**	**SMG $[m]$**
5	4.99	2.94	4.96	4.94	2.82	4.92
10	9.69	4.03	9.49	9.63	3.92	9.93
15	13.59	4.59	14.93	13.84	4.46	14.97
20	18.02	5.01	17.98	20.89	5.04	19.33
25	24.37	5.91	25.78	24.67	5.41	22.91
30	25.35	5.46	28.27	22.26	5.79	29.37

Table C.3. Depth Estimation Evaluation Results – ZED-Mini at 720 pixels - Indoors.

Indoors Depth Estimation Evaluation – ZED-Mini – 720p						
Distance $[m]$	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD $[m]$	**DLT $[m]$**	**SMG $[m]$**	**DD $[m]$**	**DLT $[m]$**	**SMG $[m]$**
5	5.07	3.53	5.44	5.08	3.51	5.02
10	10.38	5.24	9.92	10.64	5.23	10.61
15	14.78	7.73	15.41	14.84	6.59	15.71
20	19.99	7.72	20.81	19.10	7.30	22.65
25	31.54	8.21	30.96	31.58	8.17	28.60
30	32.80	9.10	30.32	28.64	8.18	27.28

Table C.4. Depth Estimation Evaluation Results – ZED at 720 pixels - Outdoors.

Outdoors Depth Estimation Evaluation – ZED – 720p						
Distance $[m]$	**Rectified by ZED SDK**			**Unrectified (raw format)**		
	DD $[m]$	**DLT $[m]$**	**SMG $[m]$**	**DD $[m]$**	**DLT $[m]$**	**SMG $[m]$**
5	5.04	5.53	5.03	5.00	3.50	5.00
10	10.00	5.23	10.63	10.35	5.20	10.29
15	14.96	6.46	15.25	15.01	6.55	14.85
20	19.43	8.20	21.88	18.88	7.16	21.75
25	26.33	9.62	34.74	22.21	10.59	37.12
30	43.12	7.93	48.51	31.02	8.06	44.53

Bibliography

[1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, 2004, pp. 652–659.

[2] H. P. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Ph.D. Dissertation, Stanford University, Stanford, CA, 1980.

[3] J. P. Lewis, "Fast Normalized Cross-Correlation," 2008. [Online]. Available: http://scribblethink.org/Work/nvisionInterface/nip.pdf. [Accessed: 09-May-2019].

[4] L. Matthies and S. A. Shafer, "Error Modeling in Stereo Vision," *IEEE J. Robot. Autom.*, vol. 3, no. 3, pp. 239–248, 1987.

[5] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo, "Rover Self Localization in Planetary-Like Environments," in *Artificial Intelligence, Robotics and Automation in Space*, 1999, pp. 433–440.

[6] Y. Cheng, M. W. Maimone, and L. H. Matthies, "Visual Odometry on the Mars Exploration Rovers," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 54–62, 2006.

[7] M. Malmore, Y. Cheng, and L. H. Matthies, "Two Years of Visual Odometry on the Mars Exploration Rovers," *J. F. Robot.*, vol. 24, no. 3, pp. 245–267, 2007.

[8] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Proceedings of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[9] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] A. Howard, "Real-Time Stereo Visual Odometry for Autonomous Ground

Vehicles," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2008, pp. 3946–3952.

[11] W. Mou, H. Wang, and G. Seet, "Efficient Visual Odometry Estimation Using Stereo Camera," in *IEEE International Conference on Control and Automation, ICCA*, 2014, pp. 1399–1403.

[12] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," in *2015 European Conference on Mobile Robots, ECMR 2015 - Proceedings*, 2015.

[13] K. U. Pavan, M. P. V. Sahul, and B. T. V. Murthy, "Implementation of Stereo Visual Odometry Estimation for Ground Vehicles," in *RTEICT 2017 - 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, Proceedings*, 2017, vol. 01, pp. 1173–1177.

[14] J. Engel, J. Sturm, and D. Cremers, "Semi-Dense Visual Odometry for a Monocular Camera," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1449–1456.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, 2017.

[16] D. Scaramuzza, "Ominidirectional Vision: From Calibration to Robot Motion Estimation," Doctoral Thesis, Eidgenössische Technische Hochschule Zürich, 2008.

[17] D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, 2011.

[18] D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part II - Matching,

Robustness, and Applications," *IEEE Robot. Autom. Mag.*, vol. 19, no. 2, pp. 78–90, 2012.

[19] L. Kneip, D. Scaramuzza, and R. Siegwart, "A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2969–2976.

[20] C. Kerl, J. Stuckler, and D. Cremers, "Dense Continuous-Time Tracking and Mapping with Rolling Shutter RGB-D Cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, vol. 2015 Inter, pp. 2264–2272.

[21] T. Schops, J. Enge, and D. Cremers, "Semi-Dense Visual Odometry for AR on a Smartphone," in *ISMAR 2014 - IEEE International Symposium on Mixed and Augmented Reality - Science and Technology 2014, Proceedings*, 2014, pp. 145–150.

[22] W. Gong *et al.*, "Human Pose Estimation from Monocular Images: A Comprehensive Survey," *Sensors (Basel).*, vol. 16, no. 12, pp. 1–40, 2016.

[23] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2018.

[24] C. Foster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, 2017.

[25] S. Zhang, *Handbook of 3D Machine Vision*, 1st Ed. Taylor & Francis Group, 2013.

[26] Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown

Orientations," *Proc. Seventh IEEE Int. Conf. Comput. Vis.*, pp. 666–673 vol.1, 1999.

[27] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

[28] Stereolabs, "Stereolabs - Capture the World in 3D," *stereolabs.com*, 2019. [Online]. Available: https://www.stereolabs.com/. [Accessed: 18-Jan-2019].

[29] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, 2008.

[30] H. Liu, Y. Dong, and F. Wang, "Study on Camera Calibration for Binocular Stereovision Based on Matlab," vol. 404, pp. 201–209, 2016.

[31] Matlab, "Computer Vision Toolbox," *Release 2019a*. The MathWorks Inc, 2019.

[32] "Camera Calibration and 3D Reconstruction — OpenCV 2.4.13.7 Documentation," *docs.opencv.org*, 2019. [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. [Accessed: 07-Mar-2019].

[33] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machin Vision*, 3rd Editio. Thomson, 2008.

[34] R. Hartley and A. Zisserman, *Multiple Geometry in Conmputer Vision*, 2nd ed. Cambridge University Press, 2004.

[35] D. Nistér, "An Efficient Solution to the Five-Point Relative Pose Problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–770, 2004.

[36] P. H. S. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry," *Comput. Vis. Image Underst.*, vol. 78, no. 1, pp. 138–156, 2000.

[37] H. C. Longuet-Higgins, “A Computer Algorithm for Reconstructing a Scene from Two Projections,” *Nature*, vol. 293, no. September, pp. 133–135, 1981.

[38] G. Bradski and A. Kahler, *Learning OpenCV 3*, 1st ed. O’Reilly Media, Incorporated, 2016.

[39] A. Mordvintsev and A. Rahmank, “Harris Corner Detention — OpenCV-Python Tutorials 1 Documentation,” *opencv-python-tutroals.readthedocs.io*, 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html. [Accessed: 04-Jan-2019].

[40] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *Int. J. Comput. Vision.*, vol. 60, no. 2, pp. 91–110, 2004.

[41] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded Up Robust Features,” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[42] M. Mujia and D. G. Lowe, “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration,” *Int. Conf. Comput. Vis. Theory Appl. VISAPP.*, 2009.

[43] J. H. Freidman, J. L. Bentley, and R. A. Finkel, “An Algorithm for Finding Best Matches in Logarithmic Expected Time,” *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, 1977.

[44] B. D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision,” *Proc. 7th Int. Jt. Conf. Artif. Intell.*, pp. 674–679, 1981.

[45] R. Rojas, “Lucas-Kanade in a Nutshell,” *Freie Univ. Berlinn, Dept. Comput. Sci. Tech. Rep*.

[46] H. Hirschmüller, “Stereo Processing by Semiglobal Matching and Mutual Information,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, 2008.

[47] H. Hirschmüller, “Semi-Global Matching - Motivation, Developments and Applications,” *Photogramm. Week 11*, no. Figure 1, pp. 173–184, 2011.

[48] D. Scharstein, “2014 Stereo Datasets,” *vision.middlebury.edu*, 2015. [Online]. Available: http://vision.middlebury.edu/stereo/data/scenes2014/. [Accessed: 16-Dec-2018].

[49] W. Boonsuk, “Investigating Effects of Stereo Baseline Distance on Accuracy of 3D Projection for Industrial Robotic Applications,” in *Proceedings of The 2016 IAJC-ISAM International Conference*, 2016, pp. 94–98.

[50] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, “Automatic Camera and Range Sensor Calibration Using a Single Shot,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012, pp. 3936–3943.

[51] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-Squares Fitting of Two 3D Point Sets,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[52] Y. Jiang, H. Chen, G. Xiong, and D. Scaramuzza, “ICP stereo visual odometry for wheeled vehicles based on a 1DOF motion prior,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014, pp. 585–592.

[53] P. V. Bustamante and J. M. I. Zannatha, “Visual Odometry Based on Trilateration Method with Stereo Images,” *2017 Int. Conf. Electron. Commun. Comput. CONIELECOMP 2017*, pp. 1–6, 2017.

[54] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, “Robust Stereo Ego-motion for Long Distance Navigation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000, vol. 2, pp. 453–458.

[55] M. a Fischler and R. C. Bolles, “Random Sample Paradigm for Model Consensus: A Apphcatlons to Image Fitting with Analysis and Automated Cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[56] Y. Wu, “PnP Problem Revisited,” *J. Math. Imaging Vis.*, vol. 24, no. 1, pp. 131–141, 2006.

[57] F. Moreno-Noguer, V. Lepetit, and P. Fua, “Accurate non-iterative O(n) solution to the PnP problem,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[58] M. I. of T. (MIT), “Orthogonal Matrices and Gram-Schmidt,” *Linear Algebra Fall 2011*, 2011. [Online]. Available: https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/least-squares-determinants-and-eigenvalues/orthogonal-matrices-and-gram-schmidt/MIT18_06SCF11_Ses2.4sum.pdf. [Accessed: 19-Nov-2018].

[59] R. C. Smith and P. Cheeseman, “On the Representation and Estimation of Spatial Uncertainty,” *Int. J. Rob. Res.*, vol. 5, no. 4, pp. 56–68, 1986.

[60] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, “Bundle Adjustment — A Modern Synthesis,” in *International Workshop on Vision Algorithms*, 2000, vol. 26, no. 2, pp. 298–372.

[61] M. I. A. Lourakis and A. A. Argyros, “SBA: A software package for generic sparse

bundle adjustment," *ACM Transactions on Mathematical Software*, vol. 36, no. 1, 2009.

[62] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*, Fourth Ed. New Jersey: John Wiley & Sons, Inc., 2013.

[63] J. J. Moré, "The Levenberg-Marquardt Algorithm: Implementation and Theory," in *Watson, G.A. Numerical Analysis: Lecture Notes in Mathematics*, vol. 630, Berlin Heidelberg: Springer, 1978, pp. 105–116.

[64] R. Dawson, "Orbits and Locked Gimbals," *Math. Intell.*, vol. 35, no. 1, pp. 67–70, 2013.

[65] E. G. Hemingway and O. M. O'Reilly, "Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments," *Multibody Syst. Dyn.*, vol. 44, no. 1, pp. 31–56, 2018.

[66] Matlab, "MATLAB," *Release 2019a*. The MathWorks Inc, 2019.

[67] Matlab, "Image Processing Toolbox," *Release 2019a*. The MathWorks Inc, 2019.

[68] Matlab, "Optimization Toolbox," *Release 2019a*. The MathWorks Inc, 2019.

[69] OpenCV, "OpenCV 3.4.1," *Release 3.4.1*. OpenCV Team, 2018.

[70] Stereolabs, "ZED SDK," *Release 2.8*. Stereolabs, 2019.

[71] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous Driving? The KITTI Vision Benchmark Suite," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3354–3361, 2012.

[72] A. Geiger, "The KITTI Vision Benchmark Suit," *cvlibs.net*, 2019. [Online]. Available: http://www.cvlibs.net/datasets/kitti/. [Accessed: 16-Dec-2018].

[73] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI

Dataset," *Int. J. Robot. Res.*, 2013.

[74] W. Paal, "MRT Institut für Mess- und Regelungstechnik," *mrt.kit.edu*, 2019. [Online]. Available: https://www.mrt.kit.edu/annieway/. [Accessed: 25-Jun-2019].

[75] A. Geiger, "Visual Odometry / SLAM Evaluation 2012," *cvlibs.net*, 2019. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_odometry.php. [Accessed: 16-Dec-2018].

[76] Stereolabs, "ZED Stereo Camera - Stereolabs," *stereolabs.com*, 2019. [Online]. Available: https://www.stereolabs.com/zed/. [Accessed: 20-Mar-2018].

[77] Stereolabs, "ZED Mini Stereo Camera - Stereolabs," *stereolabs.com*, 2019. [Online]. Available: https://www.stereolabs.com/zed-mini/. [Accessed: 12-Mar-2019].

[78] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. New York City: Cambridge University Press, 1992.