UNIVERSITÀ
DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN TECNOLOGIE DELL'INFORMAZIONE

CICLO XXXI

Multiple View Odometry for Autonomous Driving

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutore:

Chiar.mo Prof. Pietro Cerri

Dottorando: *Alessandro Cionini*

Anni 2015/2018

Man muss noch Chaos in sich haben,
um einen tanzenden Stern gebären zu können.

Abstract

This thesis presents a real-time visual odometry system which makes use of multiple views. The motion estimation system has been designed for a concrete implementation on a computer vision processor specifically intended for autonomous driving. An autonomous vehicle has to work in different environments characterized by peculiar issues and, of course, it has to provide the same level of safety in any condition, therefore accuracy and reliability are essential requirements for such a system. Moreover, the hardware-related constraints led to conceive a lightweight method with small memory requirements that does not make use of maps.

The proposed approach uses feature correspondences from consecutive frames. The keypoints are extracted from the image using the Harris corner detector and then a BRIEF-like descriptor is used for the comparison operation. A fundamental characteristic of the proposed method is the robustness against outliers, which is provided by a motion estimator specifically designed to reject them through the use of a weight function.

A multiple view approach has been implemented, in order to increase the accuracy and the reliability of the system. The multiple view paradigm is exploited both in time and space, indeed the motion estimation is computed using one or more cameras and then it is further optimized considering a set of the last acquired frames through Sparse Bundle Adjustment. The system is therefore quite modular and it can be tailored to the particular setup of the autonomous vehicle.

An open dataset widely used in literature has been used to evaluate the performance of the proposed method in terms of accuracy and to compare it with the top-ranked state of the art methods, showing comparable results. The system has been tested in different configurations on a dataset specifically acquired for its validation, in order to highlight the contributions of the multiple view approach proposed. The results underlined how the use of multiple cameras significantly reduces the overall translation error and how the use of multiple frames considerably decreases the rotation error.

Contents

List of Figures

List of Tables

Introduction

Computer vision gained tremendous relevance in the last years, becoming one of the most lively field of research in Information Technology. The recent spread of connected intelligent devices, a phenomenon known as the Internet of Things (IoT) made possible by the ongoing electronic miniaturization, as well as the massive production of valuable data in the form of images, as part of the Big Data revolution which is taking place since last decade, boosted the research in a way that was barely imaginable very few years ago.

Deep learning and, more specifically, the Neural Network recent comeback in Artificial Intelligence played an undeniable and prevailing role in the astounding development of modern computer vision. However, while deep learning techniques represent nowadays the most effective solutions for image classification, the study of geometric techniques and mathematical models cannot be disregarded when dealing with tasks that address accurate scene reconstruction through visual perception (see Figure 1).

Computer vision naturally spreads to numerous application: from automatic inspection and process control in manufacturing to navigation for autonomous agent and image classification of the huge amount of data generated on the Internet. The availability of high-resolution camera systems, capable of considerable computational power, combined with the overwhelming improvement in computer vision techniques made autonomous driving one of the hottest technology in both public and private research (see Figure 2), starting a gold rush for automobile manufacturers all over the world.

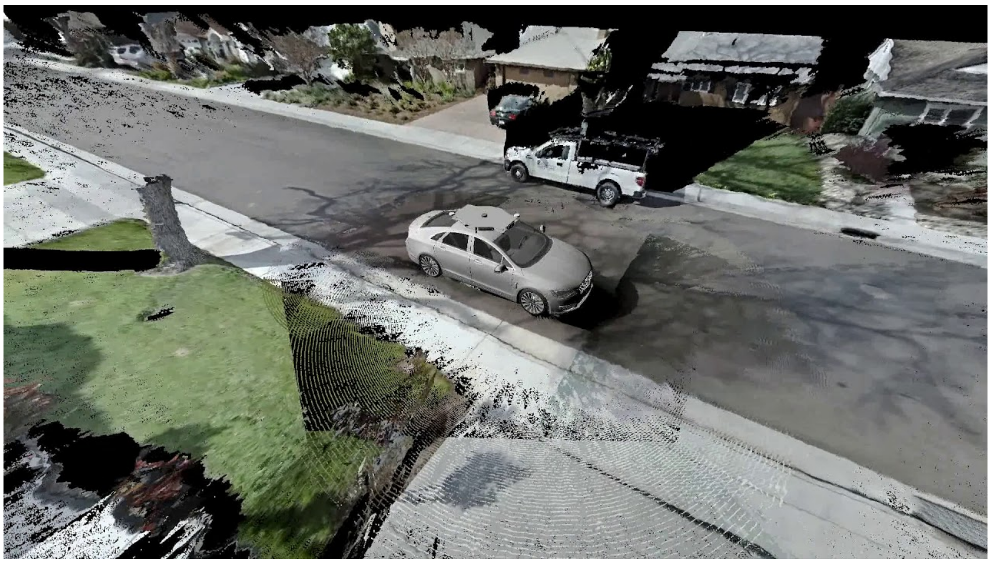


Figure 1: Full 360° 3D reconstruction based on the perception of a complete set of stereo cameras.

A self-driving car is a vast and complex system, involving many modules that tackle different challenging task, working all together in synergy in order to perceive the world and navigate. Self localization is a fundamental function for an intelligent vehicle, since it allows to establish position and orientation in a given reference system, thus allowing navigation and improving sensor fusion. A solid and accurate self localization system cannot rely on the Global Positioning System (GPS) alone, which is affected by intrinsic problems and low accuracy; for that reason it usually exploits also odometry, the process which uses data from sensors to estimate change in position and orientation over time. Traditional odometry techniques are performed using devices such as rotary encoders to measure wheel rotations, an approach inapplicable in case of non-standard locomotion methods such as drones. Visual Odometry (VO) represents a valid alternative for a system with a camera, determining equivalent odometry information through use of sequential images, regardless of the actual locomotion method.

Figure 2: Examples of fully autonomous vehicles: (a) Waymo by Google; (b) Advanced Technologies Center (ATC) by Uber; (c) Embedded Vehicle Autonomy (EVA) by Ambarella.

VO can be categorized as monocular VO when using a single camera, or stereo VO when using a stereo camera. A stereo camera has the advantage of providing information about depth and scale, thanks to the two points of view and the relative calibration, while producing the same information through a monocular camera is non trivial. VO is typically computed by feature-based methods, extracting and tracking image feature points in two consecutive images that are then used to determine the rigid transformation, therefore the model, which explains their movement in the scene.

A way to improve egomotion estimation with accurate information about scale is to extend the processing to multiple frames, in order to reduce noise and to better constrain the model.

A real time multiple view method for stereo VO is described in this dissertation, which introduces a Iteratively Reweighted Estimator robust to outliers that can be used with multiple cameras and a multiple frame optimization based on a local Sparse Bundle Adjustment. The following pages are structured as follows: chapter 1 introduces the VO problem, common approaches and major achievements in the last decades; chapter 2 introduces the geometrical concepts adopted in this work, basically a preparatory chapter for chapter 3 which describes the proposed method in all its aspects; chapter 4 shows the results obtained both with open datasets and sequences specifically acquired using a vehicle designed for algorithm validation; finally, chapter 5 discusses the contribution of this work.

Chapter 1

A Brief History of Visual Odometry

Visual Odometry is the process of estimating the egomotion of an agent using only the input of one or more cameras attached to it. The term VO was coined by Nister in [1], inspired by wheel odometry, although the problem of estimating a vehicle's egomotion from visual input alone is way older. Indeed, this problem was first introduced in the early 1980s and was described by Moravec in [2], a stereo-like approach for feature-based motion estimation which definitely represents a milestone. Traditional odometry incrementally estimates the motion of a vehicle by integrating the number of turns of its wheels over time, thus suffering of typical estimation errors due to wheel slip. A problem that does not affect VO, which uses the same approach by incrementally estimating the pose through the analysis of changes induced by motion on the images acquired by the onboard camera. VO is thus a robust and reliable supplement to the localization system, especially in GPS-denied environments.

VO has been gaining increasing popularity over the last decade, as evidenced by the large number of publications on the topic [3] as well as the release of open datasets made available for objective performance comparison [4, 5, 6]. In literature, work estimating motion from image sequences can be divided in two major categories: optical flow algorithms, which can be considered as dense methods, and feature-based algorithms, which can be considered as sparse methods [7]. Dense methods are less accurate and computationally more expensive than sparse techniques, that only require the ability of robustly track features across multiple frames. Feature-based techniques are indeed extensively more used, although there is some recent work recovering motion from optical flow [8]. In this chapter, a more detailed formulation of the VO problem is presented, as well as an overview of the state of the art regarding VO techniques, mostly for what concerns feature-based methods both for Stereo and Monocular approaches.

1.1 The Visual Odometry Problem

The VO problem can be outlined considering a simple setting: a camera, either monocular or stereo, is moving through an environment acquiring images at every time instant k. In a monocular system, the set of acquired images is denoted by $I_{0..n} = \{I_0, ..., I_n\}$, while in a stereo system, the two sets of acquired images are denoted by $I_{L,0..n} = \{I_{L,0}, ..., I_{L,n}\}$ and $I_{R,0..n} = \{I_{R,0}, ..., I_{R,n}\}$.

Two adjacent camera positions at time instants $k-1$ and k are related by a rigid body transformation ${}_kT^{k-1} \in \mathbb{R}^{4\times4}$ (see Appendix A) of the following form:

$$ {}_kT^{k-1} = \begin{bmatrix} {}_kR^{k-1} & {}^kt_{k\to k-1} \\ 0 & 1 \end{bmatrix} \tag{1.1}$$

where ${}_kR^{k-1} \in SO(3)$ is the rotation matrix and ${}^kt_{k\to k-1} \in \mathbb{R}^{3\times1}$ the translation vector. Therefore, the set $T_{1:n} = {}_1T^0, ..., {}_nT^{n-1}$ contains all the relative transformations corresponding to subsequent motions and the set of camera poses $P_{0:n} = P_0, ..., P_n$ contains the transformations of the camera with respect to the initial coordinate frame $k = 0$. The k^{th} camera pose $P_k =_0 T^k$ can be computed by inverting all the previous transformations and concatenating all of them as follows:

$$ P_k = \prod_{i=1}^{k} {}_{i-1}T^i \tag{1.2}$$

or iteratively by:

$$ P_k = P_{k-1}\ {}_{k-1}T^k \tag{1.3}$$

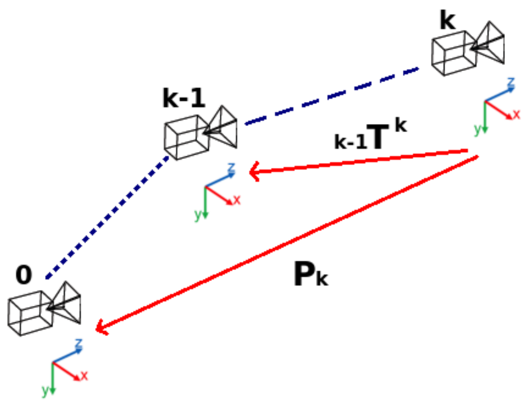


Figure 1.1: The VO problem consists in estimating the relative transformations between the frames and the poses with respect to the first frame.

with P_0 being the camera pose at the instant $k = 0$, which can be set for simplicity to the identity matrix I_4.

The task of VO is to compute at each camera position k the relative transformation ${}_kT^{k-1}$ between the images I_{k-1} and I_k and eventually composing it with the previous pose P_{k-1} in order to recover the full trajectory of the camera (Figure 1.1). The position and orientation of the camera is therefore computed incrementally.

1.1.1 Drift: Public Enemy Number One

The major intrinsic issue in VO is the incremental pose estimation, which makes errors introduced by each relative transformation accumulate over time. The accumulated error engenders a drift of the estimated trajectory from the real path. While some applications, such as vehicle control, only necessitate accurate relative pose estimation, for others, such as self localization in GPS denied environments, is of utmost importance to keep drift as small as possible.

Typical techniques for reducing the drift require the use of a map, either precomputed or incrementally produced and updated as in Simultaneous Localization And Mapping (SLAM) approaches [9, 10, 11, 12, 13], which allows to occasionally reset the drift by localizing the agent onto it. The goal of SLAM in general is to obtain a global and consistent estimate of the agent path. This implies keeping a map of the environment in order to realize when the vehicle returns to a previously visited area, a condition called loop closure, allowing drift reduction in the estimated path and improving the map itself as well. While SLAM techniques have been usually limited to indoor workspaces, where the approach takes advantage of the map that is repeatedly covered without the drawback of unrestrained map expansion, some work have been designed for large scale areas [14, 15, 16, 17] a crucial designing target for autonomous driving applications.

In general, VO can be used as a building block for a complete SLAM algorithm, which also need a way to detect loop closing and a global optimization technique to obtain a consistent map. Although a complete SLAM algorithm can be used to provide VO with potentially high precision, it is evidently more complex and computationally expensive, requiring also a considerable amount of memory in case large scale application. Therefore VO still represents a preferable solution for real-time applications and low requirements systems.

Another way of tackling the drift problem is local optimization over the last m camera poses, an approach called windowed bundle adjustment that has been used in some works, as in [18, 19, 20, 21].

Combining VO with other sensors, such as GPS, laser or IMU, is clearly a simple approach for avoiding drift in pose estimation [19, 22, 23].

1.2 Feature-based Visual Odometry

VO techniques relying on inter-frame point correspondences typically use feature detectors and descriptors. Features are characteristic points in the image, usually each one associated with a descriptor which can be used to match the feature to a keypoint extracted in a different image. Given two images, to find feature matches across them means finding the couple of image projections produced by the same distinctive world point (Figure 1.2). Extracting a considerable number of features from two images and finding matches between elements of the two lists, allows to gather valuable information for relative pose estimation. Computer vision literature

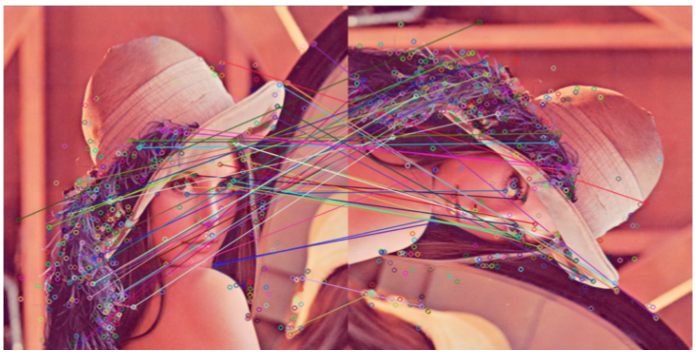


Figure 1.2: Example of feature extraction and matching.

abound in methods for extracting meaningful points from the image, since image keypoints can be used in a considerable variety of computer vision applications. While some VO algorithms use custom designed feature detectors, such as [24], a non-comprehensive list of widespread feature detectors and descriptors is presented below.

HARRIS Harris corner detector, introduced by Harris and Stephens in 1988 [25], represents the most popular corner detection operator in computer vision and it is invariant to rotation, scale and illumination. Harris improved upon Moravec's corner detector [2] by considering differential of the corner score with respect to direction directly, computing the locally averaged moment matrix from the image gradients and then combining the Eigenvalues of the moment matrix to compute measure, from which maximum values indicate corners position. The applications of Harris corner detector are uncountable, [26, 1, 27] represent some relevant examples for this dissertation.

FAST Features from Accelerated Segment Test (FAST), proposed by Rosten and Drummond [28], is a corner detection method. FAST uses a circle of 16 pixels to classify whether a candidate point is actually a corner, looking for contiguous pixels belonging to the circle that are brighter or darker than the candidate by a defined threshold. The advantage of FAST is its computational efficiency in term of time and resources, which made it suitable for real-time applications such as [29].

BRIEF Binary Robust Independent Elementary Features (BRIEF), introduced by Calonder [30], is a feature descriptor which can be used to describe previously extracted keypoints. Given a feature, BRIEF takes a smoothened image patch around it, selects a set of location pairs and computes a bit value for each pair by comparing the pixel intensities to those positions. The advantages are the small memory requirements and the ease of matching through Hamming distance.

SIFT Scale-Invariant Feature Transform (SIFT), proposed by Lowe [31, 32], is an algorithm to detect and describe features, invariant to variation in scale, rotation, illumination and viewpoint. The SIFT algorithm has 4 basic steps: first is to estimate the scale space extrema using the Difference of Gaussian (DoG); secondly, a key point localization where the key point candidates are localized and refined by eliminating the low contrast points; thirdly, a key point orientation assignment based on local image gradient; lastly, a descriptor generator to compute the local image descriptor for each key point based on image gradient magnitude and orientation. Examples of SIFT application to 3D reconstruction are [33, 34, 27].

SURF The Speeded Up Robust Features (SURF), proposed by Bay [35], is partly inspired by SIFT. SURF approximates the DoG with box filters and detect interest points using an integer approximation of the determinant of Hessian blob detector. The feature descriptor is based on the sum of the Haar wavelet response around the point of interest. SURF is several times faster that SIFT and still robust against different image transformations. An application to VO can be found in [36].

Choosing how features are extracted and matched between consecutive frames is critical because, as already seen, every method implies a different computational cost and consequently performances in terms of robustness, quality and speed. Anyway, whichever technique is adopted, the outcome is a list of matched keypoints that are

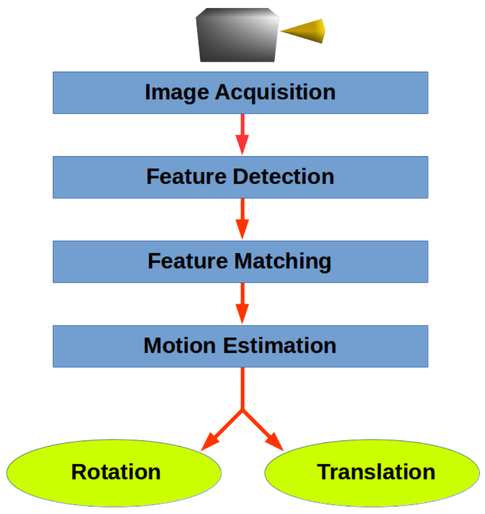


Figure 1.3: Full pipeline of a feature-based VO algorithm.

then used to estimate the relative transformation which explains how those keypoints moved through the two subsequent images. Figure 1.3 shows the full pipeline of a feature-based VO algorithm.

1.3 Stereo Visual Odometry

The use of a stereo pair of cameras, as opposed to a single monocular camera, greatly simplifies the motion estimation problem because the known transformation between the cameras allows the metric structure of the scene to be estimated using a single stereo image pair. In other words, a stereo camera allows to directly measure, by means of triangulation, the relative 3D position of the tracked features at every agent location and use them to derive the relative motion.

As Moravec put his milestone in feature-based motion estimation [2], many successive works tried to move forward by improving the estimation accuracy. For many years all the proposed methods shared the same minimization approach by solving the relative motion as a 3D-to-3D point registration problem, hence triangulating the 3D points at every stereo pair [37, 38, 39]. Nister et al. in [1] besides coining the term VO, proposed a completely different approach based on 2D visual features detected using Harris corner detector [25], which computes the relative motion as a 3D-to-2D camera pose estimation problem and performs outlier rejection using Random Sample Consensus (RANSAC) [40]. Comport et al. [41] introduced another technique which relies on the quadrifocal tensor, hence solving a 2D-to-2D motion estimation problem without having to triangulate 3D points in the stereo images.

Methods based on 3D-to-3D pose estimation generally suffer of less accurate motion computation caused by the triangulation procedure, which strictly depends on the camera calibration accuracy and intrinsically tends to introduce a triangulation error. A more detailed analysis of the camera projection model and a comparison of motion estimation methods is presented in section 1.5.

Among recent works, Cvišić et al. proposed a solid approach in [42] and further improved in [13], based on accurate features selection using a stereo camera. Buczko proposed a stereo method that is based on a flow-decoupled normalized reprojection error, which estimates rotation and translation independently, separating and compensating the components of feature flow due to rotation [43]. Badino et al. proposed a method for reducing error in stereo motion estimation by incorporating the history of a feature in its image coordinates [44]. Finally, Zhu proposed an interesting approach based on a dual Jacobian optimization that is fused into a multi-scale pyramid scheme, also introducing a gradient-based feature representation robust to illumination changes [45].

1.4 Monocular Visual Odometry

Solving the egomotion estimation by means of a single camera is intriguing, although very hard to settle completely. The interest in monocular methods lies in the fact that stereo VO can degenerate to the monocular case when the distance to the scene is much larger than the stereo baseline. The disadvantage is that motion can only be recovered up to a scale factor [46, 47, 48, 16], thus metric odometry which uses a monocular camera is only feasible through the use of some other measurement of scale, such as an inertial measurement unit [49, 23] or wheel odometry. Since the absolute scale is unknown, monocular methods determine the relative scale with respect to an arbitrary initial scale, usually set equal to the distance between the first two frames.

Considering feature-based methods, the first real-time large scale monocular VO method has been presented in [1], which used RANSAC for outlier rejection and a novel five-point minimal solver [48] to compute the motion hypotheses in RANSAC. The five points RANSAC is also used in several other works, such as [50] and [51], where the estimation of translation and rotation was decoupled. Mouragnon et al. [52] used a fast and local bundle adjustment to recover both the motion and the 3D map, evaluating speed and robustness on real data.

1.5 Motion Estimation Methods

Motion estimation methods can be categorized depending on whether the feature correspondences are specified in two or three dimensions. Considering two frames $k-1$ and k and the corresponding feature lists f_{k-1} and f_k :

2D-to-2D In this case, both features lists f_{k-1} and f_k are specified in 2D image coordinates.

3D-to-2D In this case, features in f_{k-1} are specified in 3D by triangulation while features in f_k are specified in 2D image coordinates.

3D-to-3D In this case, both features lists f_{k-1} and f_k are specified in the corresponding 3D camera reference system. If using a stereo camera, every feature is triangulated using (2.9).

The 2D-to-2D methods usually aim to estimate the Essential Matrix that describes the geometric relations between two views [46] of a calibrated camera. The Essential Matrix contains the camera motion parameters up to an unknown scale factor for the translation, that has to be recovered at a later stage. The rotation matrix ${}_kR^{k-1}$ and the translation vector ${}^{k}t_{k\rightarrow k-1}$ are extracted from the Essential Matrix by decomposition, an efficient approach is proposed in [48].

In 3D-to-3D methods, the transformation ${}_{k}T^{k-1}$ can be computed by determining the aligning transformation of the two 3D feature lists. The estimation consists in finding the ${}_{k}T^{k-1}$ that minimizes the L_2 distance between the two 3D feature lists:

$$\arg\min_{{}_{k}T^{k-1}} \sum_{i} \left\| \bar{P}_{k}^{i} - {}_{k}T^{k-1} \bar{P}_{k-1}^{i} \right\|_{2} \tag{1.4}$$

where $\bar{P}$ are the homogeneous coordinates of the 3D points and i denotes the i-th feature. To weight the 3D points in the estimation using their measurement uncertainties, if known, is an improvement proposed in [53]. The computed transformation is characterized by an absolute scale, hence it is ready to be used for computing incremental poses by concatenation.

3D-to-3D methods have been adopted in many works in the past, including the first pioneering approaches of VO. However, according to Nister et al. [50] the error in 3D-to-3D motion estimation is way greater compared to other methods, because it minimizes the 3D position error contrary to those methods, which instead minimize the image reprojection error. As Matthies et al. stated in [37], the uncertainty in triangulation is not a scalar value growing with distance, which can be represented by a spherical covariance, but rather it has a diamond shape caused by the quantization error of features's image coordinates, since the uncertainty of the 3D point is also skewed and oriented in the space depending on the distance and on the point of view. The 3D points can be weighted using a 3D Gaussian, which makes it possible to represent an ellipsoidal covariance, but the distant points present longer tails in the true error distribution making the 3D position error minimization a weak approach (Figure 1.4).

In 3D-to-2D methods, the transformation ${}_{k}T^{k-1}$ is computed from the 3D-to-2D correspondences P_{k-1} and p_k, where the last ones are the f_k in image coordinates. The estimation consists in finding the ${}_{k}T^{k-1}$ that minimizes the image reprojection error, a problem known as *perspective from n points* (PnP):

$$\arg\min_{{}_{k}T^{k-1}} \sum_{i} \left| p_{k}^{i} - \tilde{p}_{k-1}^{i} \right|^{2} \tag{1.5}$$

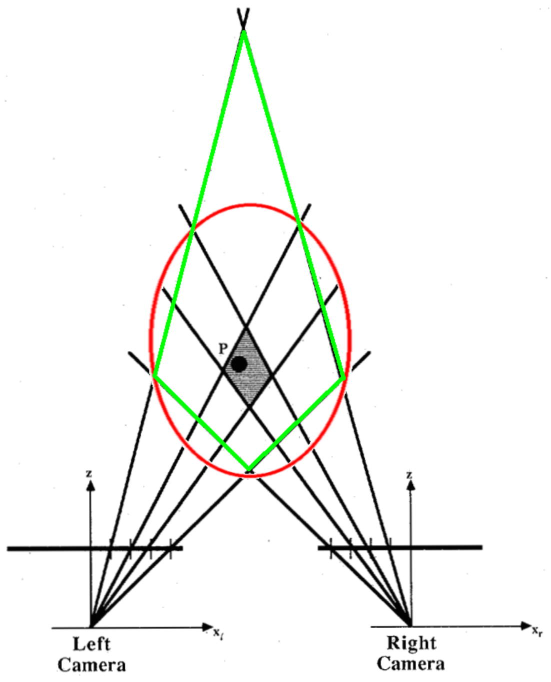


Figure 1.4: The uncertainty in triangulation. The real diamond shaped distribution is represented in green, while the 3D Gaussian approximation in red.

where $\tilde{p}^i_{k-1}$ is the projection over the k-th frame of the i-th feature point P^i_{k-1}:

$$\tilde{p}^i_{k-1} = \Pi({}_kT^{k-1}, P^i_{k-1}) \tag{1.6}$$

The PnP problem has different solutions in literature [54, 46], with the minimum solution requiring only three correspondences [40, 55, 56]. In the case of a stereo camera, the 3D point P_{k-1} is obtained by triangulation from the previous image pair, while in the case of a monocular camera, the 3D point P_{k-1} is obtained by triangulation using at least the two previous frames.

1.6 Bundle Adjustment

Bundle Adjustment is the problem of refining a visual reconstruction to produce a jointly optimal 3D structure and viewing parameter estimation, intended as camera pose and calibration. In BA the parameters are estimated by minimizing some cost function that quantifies the model fitting error, trying to achieve an optimal solution with respect to both structure and camera variations [57]. BA can be considered as a large sparse geometric parameter estimation problem, the parameters being the 3D feature coordinates, the camera poses and the camera calibration. The minimization is usually achieved using non-linear least-squares algorithms, such as Levenberg-Marquardt.

The mathematical definition is straightforward since BA just minimizes a reprojection error formulated as:

$$\min_{c_j, f_i} \sum_{i=1}^{n} \sum_{j=1}^{m} b_{ij} \left| \Pi(c_j, f_i), p_{ij} \right|^2 \tag{1.7}$$

where c_j denotes the camera pose and calibration at frame j, f_i denotes the i-th 3D feature point, $\Pi(c_j, f_i)$ denotes the image projection of f_i at frame j using the corresponding camera parameters c_j, p_{ij} denotes the image observation of feature f_i at frame j and b_{ij} is a binary variable which assumes value 1 if the feature point f_i is visible at frame j. The formula highlights the main drawback of BA, that is, the growth of computation time with the number of frames [58].

Sparse Bundle Adjustment (SBA) introduced in [59] is an optimized version which takes advantage of the lack of interaction among parameters for different 3D points and camera parameters in multiple view reconstruction, which results in the underlying normal equations exhibiting a sparse block structure; SBA has been used in various works, such as [18, 19, 21].

BA-based VO improvements usually adopt a constrained BA, which results in better performances [20], or a windowed BA, which just optimizes the last observed frames [52], in order to bound the required computation time to a maximum determined by the window size, regardless of the trajectory length and the number of frames.

Chapter 2

3D Reconstruction Geometry

The geometry involved in 3D reconstruction is a tremendously vast topic. Even considering only the models and concepts used for this work, providing a comprehensive description is infeasible, therefore in this chapter just a brief overview is provided. Firstly, an outline of the adopted camera model is presented. Secondly, the extension of that model to the stereo camera is delineated, showing some fundamental concepts of stereo vision. Lastly, the generic triangulation method for monocular camera exploited in this work is described.

2.1 Pinhole Camera Model

The projection model used for computing camera to world projection is the pinhole projection model [46]. In the pinhole projection system, the image is formed by the intersection of the light rays from the world through the center of the lens, called pinhole, with the focal plane (Figure 2.1). Given a 3D world point $P = [x, y, z]^T$ in the camera reference system and $\hat{p} = [\hat{u}, \hat{v}]^T$ being its projection on the camera sensor, the operation that allows to map the world point to the corresponding sensor point is given by the following equation:

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \frac{f}{z} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.1}$$

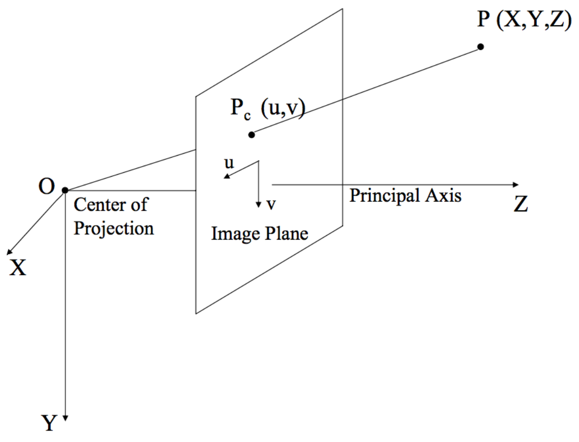


Figure 2.1: The pinhole projection model.

where f is the focal length, which is the distance between the pinhole and the sensor. The sensor point $\hat{p}$ is not the image point $p = [u, v]^T$, which is indeed obtained by applying another transformation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} D_u \, \hat{u} \\ D_v \, \hat{v} \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2.2}$$

where the coordinates (u_0, v_0), which represent the projection of the perspective center over the image, called principal point, compensate the offset with the origin of the image coordinate space, while D_u and D_v are factors that convert from meters to pixels, usually identical for modern digital sensors. The parameters f, D_u and D_v are typically incorporated in the factors $k_u = f D_u$ and $k_v = f D_v$ that are indeed the focal lengths expressed in pixel. Eventually, adopting homogeneous coordinates, the mapping from the 3D camera point to the 2D image point can be performed by the perspective projection equation:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KP = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.3}$$

where $\lambda = z$ representing the depth factor is implied and K is called intrinsic matrix, since it contains all the intrinsic parameters that describe the geometric property of the camera. The inverse of the intrinsic matrix:

$$K^{-1} = \begin{bmatrix} \frac{1}{k_u} & 0 & -\frac{u_0}{k_u} \\ 0 & \frac{1}{k_v} & -\frac{v_0}{k_v} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

allows to compute the inverse projection up to a scale factor, from 2D image point to 3D camera point. The intrinsic parameters are estimated performing camera calibration through a planar checkerboard-like pattern [60], a process that allows to correct radial lens distortion.

2.2 Triangulation

Triangulation is the operation of reconstructing the 3D world coordinates of a point observed from multiple viewpoints. Triangulating points, seen in 2 or more views, enables the reconstruction of the observed scene, but some information regarding the relative position and orientation of the views is required. Stereo vision is a specific case of multiple view geometry, which allows to significantly simplify the triangulation operation.

2.2.1 Stereo Vision

Given a stereo camera characterized by a solid structure, the easiest way to perform stereo vision is to calibrate the two sensors in order to obtain the relative pose between them and apply a warp operation to the image pair called rectification [46]. Rectification makes homologous points from the two images to lay on the same line, that is, to have the same v coordinate, and also makes the two cameras ideally perfectly aligned on the same reference system. Given a 3D point $P = [x, y, z]^T$ and the position of the left camera $T_L = [x_L, y_L, z_L]$, both in the right camera reference system, which is considered as master without loss of generality, the left $p_L = [u_L, v_L]^T$ and right

$p_R = [u_R, v_R]^T$ projections can be computed as:

$$\begin{bmatrix} u_R \\ v_R \end{bmatrix} = \begin{bmatrix} k_u \frac{x}{z} + u_0 \\ k_v \frac{y}{z} + v_0 \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} u_L \\ v_L \end{bmatrix} = \begin{bmatrix} k_u \frac{x - x_L}{z} + u_0 \\ k_v \frac{y - y_L}{z - z_L} + v_0 \end{bmatrix} \tag{2.6}$$

but camera alignment constraints forces $x_L = -b$, $y_L = 0$ and $z_L = 0$ with b, called baseline, being the offset between the cameras over the axis along which they lay. Therefore, the problem of 3D reconstruction is definitely simplified, since it is possible to introduce a new mathematical concept called disparity. Considering the projections of P over the rectified images, v_L and v_R are identical by construction and the disparity is defined as the difference between the horizontal coordinates:

$$d = u_L - u_R \tag{2.7}$$

Introducing those concepts in equation (2.5) and (2.6), the relation between disparity and P emerges:

$$d = u_L - u_R = \left(k_u \frac{x + b}{z} + u_0\right) - \left(k_u \frac{x}{z} + u_0\right) = k_u \frac{b}{z} \tag{2.8}$$

and applying some inversions and substitutions in (2.5) and (2.8):

$$\begin{aligned} x &= (u_R - u_0)\tfrac{b}{d} \\ y &= (v_R - v_0)\tfrac{k_u}{k_v}\tfrac{b}{d} \\ z &= k_u \tfrac{b}{d} \end{aligned} \tag{2.9}$$

The intrinsic matrix K and its inverse defined in equation (2.3) and (2.4) can be extended to the stereo case introducing the reprojection matrix Q [61] and its inverse:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & u_0 & 0 \\ 0 & k_v & v_0 & 0 \\ 0 & 0 & 0 & k_u b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = Q \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{k_u} & 0 & 0 & -\frac{u_0}{k_u} \\ 0 & \frac{1}{k_v} & 0 & -\frac{v_0}{k_v} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{k_u b} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = Q^{-1} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.11}$$

2.2.2 Generic Triangulation

In case of a monocular camera, the triangulation requires few more steps. In this case, the position and orientation between two frames is not fixed and known, hence the triangulation requires first to estimate them. The geometric relationship between the two images I_k and I_{k-1} of a calibrated monocular camera is described by the essential matrix E, which contains the camera motion parameters up to an unknown scale factor for the translation:

$$E_k \simeq \left[{}^{k}t_{k\to k-1}\right]_{\text{x}} \, {}_{k}R^{k-1} \tag{2.12}$$

where the symbol $\simeq$ denotes that the equivalence is valid up to a scale factor and $\left[{}^{k}t_{k\to k-1}\right]_{\text{x}}$ is the skew symmetric matrix of ${}^{k}t_{k\to k-1}$:

$$\left[{}^{k}t_{k\to k-1}\right]_{\text{x}} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{2.13}$$

The essential matrix defines a fundamental linear relationship between homologous points from the two viewpoints:

$$p_k^{\mathrm{T}} \, E_k \, p_{k-1} = 0 \tag{2.14}$$

where $p_k = [x_k, y_k, 1]^{\mathrm{T}}$ and $p_{k-1} = [x_{k-1}, y_{k-1}, 1]^{\mathrm{T}}$ are the homologous points defined in the camera reference system in homogeneous coordinates. Given a set of corresponding image points, it is possible to estimate the essential matrix by enforcing E to satisfy the epipolar constraints defined in (2.14), an estimation problem which usually represents the main step of a monocular VO algorithm. Essential matrix estimation methods are not analyzed in this dissertation, while the following important result is used in this work.

Assuming to know the relative transformation between two images I_k and I_{k-1}, therefore knowing the essential matrix E_k and the corresponding rotation matrix ${}_kR^{k-1}$ and translation vector ${}^kt_{k\to k-1}$, it is possible to triangulate homologous points. As Kanazawa and Kanatani stated in [62], given two optimal homologous camera points $\dot{p}_k$ and $\dot{p}_{k-1}$ from two images I_k and I_{k-1} and knowing the relative pose between them, the *depth* D of the corresponding 3D point with respect to the camera reference system at image I_k can be computed as follows:

$$\begin{aligned} z &= \dot{p}_k \times {}_kR^{k-1}\,\dot{p}_{k-1} \\ D &= \frac{z^{\mathrm{T}}\,E_k\,\dot{p}_{k-1}}{z^{\mathrm{T}}\,z} \end{aligned} \tag{2.15}$$

and therefore the 3D point P_k is obtained as:

$$P_k = D\,\dot{p}_k \tag{2.16}$$

Unfortunately, camera points are always affected by noise, which results in the camera points not satisfying the epipolar constraints (2.14). Methods for correcting the camera points p_k and p_{k-1} in order to get the optimal camera points $\dot{p}_k$ and $\dot{p}_{k-1}$ have been introduced by Kanatani et al. in [62] and [63], however Lindstrom proposed in [64] a simple and efficient method for the two-view camera points optimization and triangulation problem, which can converge to the optimal solution in exactly two iterations.

The results from [64] enable the optimal camera points to be retrieved in few steps. First, the following matrix is defined:

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.17}$$

which allows to define the upper left 2×2 submatrix of E_k:

$$\tilde{E}_k = S\,E_k\,S^{\mathrm{T}} \tag{2.18}$$

then the algorithm is outlined in Algorithm 1.

Algorithm 1 Two-view camera points optimization

```
$n_k \leftarrow S\, E_k\, p_{k-1}$
$n_{k-1} \leftarrow S\, E_k^{\mathrm{T}}\, p_k$
$a \leftarrow n_k^{\mathrm{T}}\, \tilde{E}_k\, n_{k-1}$
$b \leftarrow \frac{1}{2}\left(n_k^{\mathrm{T}}\, n_k + n_{k-1}^{\mathrm{T}}\, n_{k-1}\right)$
$c \leftarrow p_k^{\mathrm{T}}\, E_k\, p_{k-1}$
$d \leftarrow \sqrt{b^2 - ac}$
$\lambda \leftarrow \frac{c}{b+d}$
$\Delta_k \leftarrow \lambda\, n_k$
$\Delta_{k-1} \leftarrow \lambda\, n_{k-1}$
$\dot{p}_k \leftarrow p_k - S^{\mathrm{T}}\, \Delta_k$
$\dot{p}_{k-1} \leftarrow p_{k-1} - S^{\mathrm{T}}\, \Delta_{k-1}$
```

Chapter 3

Robust Multiple View Odometry

The method proposed in this dissertation is a real-time stereo feature-based Visual Odometry algorithm for embedded platforms, optimized using a Sparse Bundle Adjustment over a sliding window. The approach is designed to be automatically integrated with other cameras, whether stereo or monocular, improving robustness and accuracy. A main requirement for this research was the low impact in terms of memory usage, an essential feature for an embedded implementation, which is to be executed concurrently with many other modules in a fully autonomous vehicle. For that reason, every VO methods relying either on any kind of map or a long history for observed features has been rejected by default. In the same way, considering the risky impact of delays in autonomous driving applications, a real-time response is deemed to be essential; thus, the requirement was set to 30 frames per second using 1920×1080 images on a relatively low-power embedded system. The requirements have guided the design of the algorithm outlined in the following sections, which has been implemented in a real autonomous vehicle equipped with multiple cameras and it has been tested using sequences acquired by a vehicle specifically designed for algorithm validation, whose results are discussed in chapter 4.

In this chapter, the feature extraction and matching module is outlined first; then, the motion estimation algorithm is described considering different available configurations of perception systems; finally, the multiple frame optimization procedure is

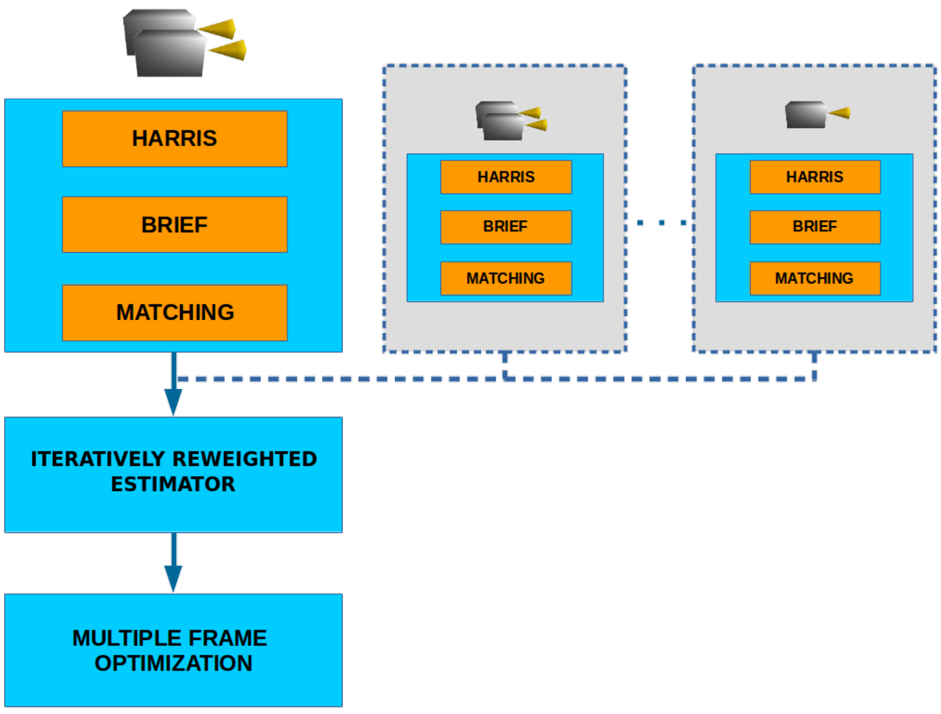


Figure 3.1: Full pipeline of the algorithm. Dotted lines represent optional components.

introduced. The full pipeline of the VO system is shown in Figure 3.1, where the three cyan squares represent the main blocks described in the following sections.

3.1 Feature Extraction and Matching

3.1.1 HARRIS Corner Detector

The feature extractor adopted in this work is Harris Corder Detector (HCD) [25]. Among the many feature extractor techniques in literature, some of them discussed in section 1.2, HCD represents a simple yet robust feature extractor, invariant to rotation, scale and illumination. The feature extractor has to be robust in any possible environment, providing a considerable amount of keypoints at each acquired frame.

HCD requires the extraction of the horizontal I_x and vertical I_y gradient images from the input image. This operation can be carried out exploiting different levels of parallelism: for instance, a naive SIMD algorithm implementing a 3×3 Sobel filter requires 16 additions/subtractions per pixel and multiple pixels can be processed in parallel.

$$\begin{aligned} I_x(0,0) = & -1 \cdot I(-1,-1) - 2 \cdot I(-1,0) - 1 \cdot I(-1,+1) \\ & +1 \cdot I(+1,-1) + 2 \cdot I(+1,0) + 1 \cdot I(+1,+1) \\ I_y(0,0) = & -1 \cdot I(-1,-1) - 2 \cdot I(0,-1) - 1 \cdot I(+1,-1) \\ & +1 \cdot I(-1,+1) + 2 \cdot I(0,+1) + 1 \cdot I(+1,+1) \end{aligned} \tag{3.1}$$

The most generic HCD algorithm then requires to compute the C matrix for each single point of the image:

$$C(x,y) = \begin{bmatrix} \sum_{\delta \in \Omega} I_x^2(\delta)\omega(\delta) & \sum_{\delta \in \Omega} I_x(\delta)I_y(\delta)\omega(\delta) \\ \sum_{\delta \in \Omega} I_x(\delta)I_y(\delta)\omega(\delta) & \sum_{\delta \in \Omega} I_y^2(\delta)\omega(\delta) \end{bmatrix} \tag{3.2}$$

with I_x and I_y being the above mentioned image gradients, considering an area Ω around (x, y) using a weight function $w(\cdot)$. The Harris cost at each point is then computed as:

$$H(x,y) = \det(C) - \kappa \operatorname{Tr}(C)^2 = C_{1,1}C_{2,2} - C_{2,1}C_{1,2} - \kappa(C_{1,1} + C_{2,2}) \tag{3.3}$$

with κ being an empirical factor. Finally, a non-maxima suppression step is done in order to keep alive only local maxima, which are reported as keypoints. The full HCD algorithm is shown in Figure 3.2.

Although the HCD response across the image is filtered using a non-maxima suppression, in the developed version the number of corners is further reduced in order to obtain a prefixed maximum number of keypoints, an essential factor for a low level hardware implementation in order to have fixed memory requirements; the maximum number of features adopted for the final implementation is 2048. Of course, corners decimation has to take into account that the resulting keypoints should be uniformly distributed across the image and also that the features with the lowest score should be discarded first. The keypoints are indeed filtered considering a grid structure of size 8×8, each bucket containing at most 32 features, providing a fair distribution of the

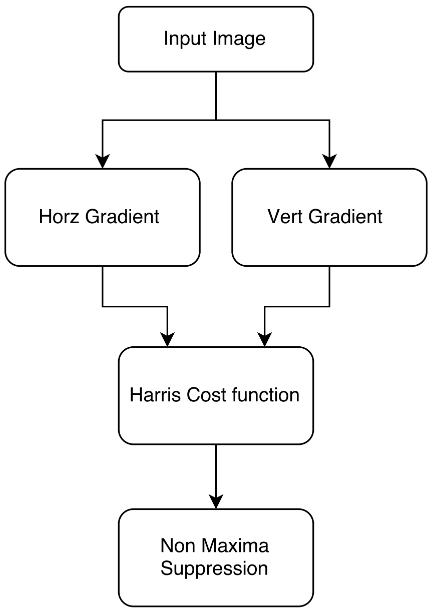


Figure 3.2: Basic flowchart of the Harris Corner Detector.

extracted corners across the image. During the decimation procedure, if more than 32 keypoints belong to a given bucket, the bucket list is sorted considering the score, and the lowest score features are removed.

3.1.2 BRIEF

HCD is a keypoint extractor, hence it does not provide a description of the feature. The kind of feature descriptor adopted in this work is BRIEF [30], which characterizes a given point as a binary representation of intensity differences, computed between predetermined pairs of points inside the patch. The main advantage of BRIEF is that the comparison operation, which is the Hamming distance, can be easily implemented at low-level using an embedded platform, thus achieving great performance in terms of speed and furthermore requiring a small amount of memory. BRIEF is not scale invariant, but that should not be a problem considering the 30 frames per second execution that should make the scale variation between consecutive frames negligible.

A BRIEF descriptor is defined by a list of segments inside the ideal considered patch. For every extracted keypoint, the image patch around it is considered and the binary string is generated comparing the value of the two pixels at the end of each segment in the image patch, which has usually been preprocessed by a low-pass filter. Therefore, the descriptor is characterized by the following elements:

- The size of patch.
- The number of comparisons, which define the descriptor size.
- The descriptor shape, defined by the comparison segments
- The kind of low-pass filter.

Experiments have been done at the very first phase of this work, highlighting that there are no major improvements by using a patch larger than 31×31 pixels. Small improvements have been observed using a descriptor larger than 256 bit, while different low-pass filters (e.g. 5×5 average, Gaussian $\sigma = 1$, Gaussian $\sigma = 2$) provide comparable results.

The shape of the BRIEF-like descriptor has been trained on several sequences using a genetic approach, evaluating the quality of the resulting optical flow. Considering the previous statement, the final implementation is a 256 bit BRIEF descriptor extracted from a 31×31 image patch filtered using a 5×5 average filter. Although the number of comparison is 256, the actual number of pixels involved is 64. The shape of the BRIEF-like descriptor is shown in Figure 3.3.

3.1.3 Matching

The VO approach proposed is stereo based, hence the features are used to compute the corresponding 3D points used in the estimation step. The triangulation method for stereo camera used, introduced in subsection 2.2.1, requires the disparity value for a given keypoint. While global methods for computing disparity exist [65], the approach here implemented consists in performing the same feature extraction procedure over the slave camera - let's say the left one without loss of generality - and match keypoints

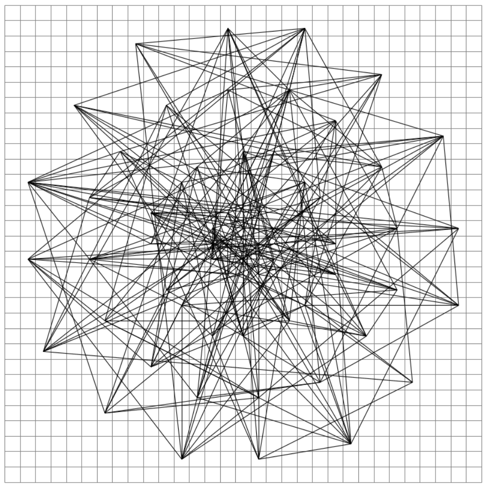


Figure 3.3: The designed 256 bit BRIEF descriptor which uses 64 elements from a 31×31 image patch.

in the stereo view. Feature matching is therefore computed between keypoints extracted at the same time in the stereo camera for feature triangulation and also between subsequent frames for motion estimation.

Considering the bucket subdivision of the extracted features and defining the search range (d_L, d_R, d_U, d_D) which establishes the distance between keypoints for the 4 directions, the matching operation can be optimized by comparing features belonging to buckets, from the other images, at a consistent distance. Of course, within the valid buckets the matching operation is done only for features that lay inside the defined ranges (Figure 3.4). For the stereo match, usually $d_L = 0$ and d_R are set to the maximum disparity expected, while d_U and d_D are configured to allow a small vertical tolerance, considering that, for rectified images, corresponding keypoints should lay over the same image row. For the flow match, d_L, d_R, d_U, d_D can all be set to the same value, supporting a squared search range. The bucket feature matching procedure is outlined in Algorithm 2.

The matching of feature binary descriptors is done by computing the Hamming distance, whose performance is determined by the presence of the POPCOUNT

Algorithm 2 Bucket feature matching

```
Let $l$ be the feature array
Let $f$ be the current feature $\in l$
$x_{start} = f.x - d_L$
$x_{end} = f.x + d_R$
$y_{start} = f.y - d_U$
$y_{end} = f.y + d_D$
for $b_h = b_{h\_start}$ to $b_{h\_end}$ do
  for $b_v = b_{v\_start}$ to $b_{v\_end}$ do
    for $count = 1$ to $32$ do
      if $l[b_h][b_v][count].x \geq x_{start}$ and $l[b_h][b_v][count].x \leq x_{end}$ and
      $l[b_h][b_v][count].y \geq y_{start}$ and $l[b_h][b_v][count].y \leq y_{end}$ then
        $Compare(f.descriptor, l[b_h][b_v][count].descriptor)$
      end if
    end for
  end for
end for
```

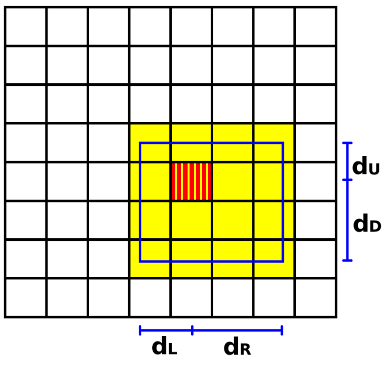


Figure 3.4: Bucket feature matching procedure: in red the bucket containing the current feature; in blue the ranges defined; in yellow the buckets (from the other image) considered for the comparison. A feature in the red bucket is compared with all the features belonging to the yellow buckets and contained inside the blue rectangle.

instruction on the hardware processor used. A correspondence is valid only if the cost of the best match is below a threshold th and better than the cost of the second best match, considering a uniqueness factor $u \in (0, 1)$ (see Algorithm 3).

Algorithm 3 Match validation

```
Let $u$ be the uniqueness value
Let $th$ be the match threshold
Let $f$ be the current feature
if $f.min\_cost < th$ and $f.min\_cost < f.second\_min\_cost \cdot u$ then
    $f.match$ valid
else
    $f.match$ invalid
end if
```

The matching operation for the stereo images is intrinsically robust, since the correspondent keypoint is searched over a narrow range, thanks to the rectification constraints, while the flow match is generally more prone to error because of noise.

For that reason, the feature list produced from the right image is firstly matched against the feature list produced from the left one, generating an ideal stereo feature list which contains only the right keypoints properly matched and for each of them the corresponding disparity value computed using the left keypoint, each stereo feature defined indeed by 3 coordinates (u, v, d). The right features with no correspondence are thus invalidated and the new stereo feature list is matched against the previous frame stereo feature list. Performing the flow match by only considering features survived to the stereo match, increase significantly the stability of tracked keypoints, since stereo feature are stronger, and results in better accuracy and robustness of the overall algorithm. The full matching procedure is shown in Figure 3.5.

3.2 Multiple Camera Motion Estimation

The motion estimation is handled as a 3D-to-2D minimization problem, with the features extracted at the previous frame triangulated and projected to the current frame, where the reprojection error with respect to the observed current feature is minimized. While at a very first stage the implemented method was a 2D-to-2D, a 3D-to-2D turns out to be more general allowing to easily integrate multiple cameras in the estimation.

In the single camera case, a 3D point can be simply defined in the reference system of the stereo camera and the motion estimated is indeed the rotation and translation of the stereo camera itself. The features are therefore triangulated using (2.9) and the estimation consists in minimizing the image reprojection error, which is computed also on the third value of the stereo feature, that is, the disparity value and (1.6) becomes:

$$\tilde{p}_{k-1} = Q^{-1} \; {}_{k}T^{k-1} \; \bar{P}_{k-1} \tag{3.4}$$

where the projection is computed using the inverse reprojection matrix Q^{-1} from (2.10).

In the multiple camera case a 3D point can be defined in the reference system of the vehicle, assuming to have the relative position and orientation of every camera with respect to the body of the vehicle, and then projected back to the original camera

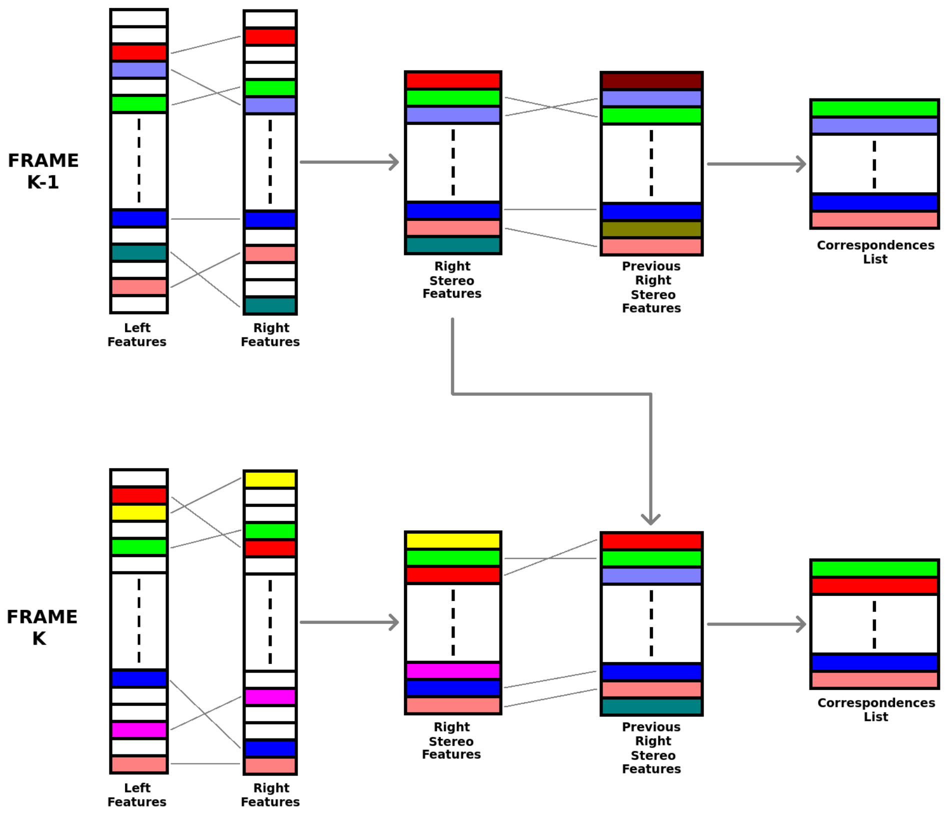


Figure 3.5: The stereo and flow feature matching: the stereo match is computed first, then the resulting subset of stereo features is used for flow matching considering the subset of stereo features obtained at the previous frame.

when computing the reprojection error, directly estimating the motion of the vehicle. The features are triangulated using (2.10) in the vehicle reference frame as follows:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} {}_{B}R^{C_j} & {}^{B}t_{B\rightarrow C_j} \\ 0 & 1 \end{bmatrix} Q_j \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{3.5}$$

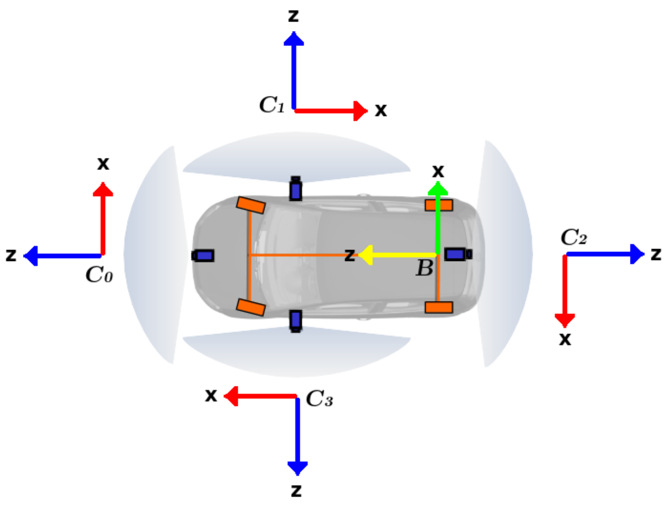


Figure 3.6: A possible autonomous vehicle setup: multiple cameras looking at different scenes. Calibrating camera position and orientation with respect to the vehicle's body allows to reconstruct the perceived world in the same reference system.

where ${}_{B}R^{C_j}$ and ${}^{B}t_{B\to C_j}$ are the rotation matrix and translation vector that allow to transform the triangulated point from the j-th camera reference system C_j to the vehicle's body reference system B, and Q_j is the reprojection matrix for the j-th camera. As shown in Figure 3.6, the autonomous vehicle may have multiple cameras with different reference systems, but the triangulated points get defined in the vehicle's body reference system.

The triangulated features are all defined in the same reference system, so (1.5) and (1.6) have to change accordingly. The image reprojection error to be minimized in order to estimate the motion ${}_{k}T^{k-1}$ is the following:

$$\arg\min_{{}_{k}T^{k-1}} \sum_{j} \sum_{i}^{N_j} \left| {}_{j}p_k^i -{}_{j} \tilde{p}_{k-1}^i \right|^2 \tag{3.6}$$

where k identifies the current frame, j identifies the camera, N_j defines the number of features for the j-th camera, ${}_{j}\tilde{p}_{k-1}^i$ is the projection over the k-th frame of the i-th feature point ${}_{j}P_{k-1}^i$ observed in the j-th camera. In the stereo camera case, the projection is again computed using Q^{-1} from (2.10) as follows:

$$ {}_j\tilde{p}_{k-1} = Q^{-1}\ {}_{C_j}T^B\ {}_kT^{k-1}\ {}_j\bar{P}_{k-1} \tag{3.7}$$

where ${}_{C_j}T^B$ is the transformation matrix that allows to transform a 3D point from the vehicle's body reference system B to the j-th camera reference system C_j. In the moncular camera case, the projection is computed using the K matrix from (2.3) as follows:

$$ {}_j\tilde{p}_{k-1} = \begin{bmatrix} K_j & 0 \\ 0 & 1 \end{bmatrix}\ {}_{C_j}T^B\ {}_kT^{k-1}\ {}_j\bar{P}_{k-1} \tag{3.8}$$

As (3.7) and (3.8) highlights, the motion ${}_kT^{k-1}$ to be used in the reprojection formula is the same for every feature, regardless the camera. The advantage is that multiple cameras can be used in order to obtain a better accuracy and robustness, without having any drawback. Multiple cameras look at difference scenes, making the full system more resistant to camera occlusion and to potential lack of information due to a single point of view, for instance when a truck is in front of the vehicle. Moreover, in a multiple camera configuration, the additional camera can be monocular, without incurring in the scale estimation problem typical of monocular VO algorithm, since the features are triangulated from the 2 previous frames using the last estimated motion which contains the scale information (see subsection 2.2.2).

3.2.1 Iteratively Reweighted Estimation

The set of feature correspondences often contains outliers as a consequence of false matches or moving objects, which present a motion not consistent with that of the vehicle. In order to cope with the presence of outliers, an optimization approach that iteratively rejects them has been applied.

The optimization is indeed performed by an iterative estimation, where, in order to reduce the impact of outliers, the minimization problem defined in (3.6) has been modified incorporating a weight factor that changes at every iteration:

$$ \arg\min_{{}_kT^{k-1}} \sum_j \sum_i^{N_j} w_{j,i} \left| {}_jp_k^i -_j \tilde{p}_{k-1}^i \right|^2 \tag{3.9}$$

and therefore the optimization consists in solving, at each iteration, a problem with the following form:

$$ {}_kT^{k-1}(t+1) = \arg\min_{{}_kT^{k-1}} \sum_j \sum_i^{N_j} w_{j,i}(t) \left| {}_jp_k^i -_j \tilde{p}_{k-1}^i \right|^2 \tag{3.10} $$

where the weight factor $w_{j,i}(t)$ for the i-th feature observed in the j-th camera is computed and updated using the previous solution as:

$$ w_{j,i}(t) = \frac{a}{\max(r_{i,j,k}(t), \lambda)} \tag{3.11} $$

where a is the age of the tracked feature, defined as the number of frame couples for which it has been matched, and defining the residual

$$ r_{i,j,k}(t) = \left| {}_jp_k^i -_j \tilde{p}_{k-1}^i \right|^2 \tag{3.12} $$

with λ being a constant which prevents the division by zero and implicitly defines the maximum value of the weight function. The minimization problem defined in (3.10) has the form of a non-linear least squares problem, hence it can be optimized using Gauss-Newton algorithm. The robustness of the outliers rejection scheme is further improved by removing the outliers from the correspondence list before starting a new iteration, classifying as outliers the stereo features having a residual value $r_{i,j,k}(t)$ greater than a predefined threshold th_{res}. Assuming that the outliers are bounded, after the first iteration they will have a larger residual cost than the inliers and this provides a simple method for outliers rejection, reducing their contribution in the new cost function.

The proposed method finds justification in the Iteratively Reweighted Least Squares (IRLS), which is an optimization method that can be used to solve problems characterized by a p-norm based objective function:

$$ \arg\min_{\beta} \sum_{i=1}^{n} |y_i - f_i(\beta)|^p \tag{3.13} $$

by iteratively solving a weighted least squares problem of the form:

$$ \beta^{t+1} = \arg\min_{\beta} \sum_{i=1}^{n} \omega_i \left(\beta^{(t)} \right) |y_i - f_i(\beta)|^2 \tag{3.14} $$

where the weight function $\omega_i\left(\beta^{(t)}\right)$, which weights every observations according to the previous minimization solution, is defined as:

$$\omega_i\left(\beta^{(t)}\right) = |y_i - f_i(\beta)|^{p-2} \tag{3.15}$$

The weight function defined in (3.11) depends on the inverse of the residual, therefore, considering (3.13), (3.14) and (3.15), it looks like the optimization problem introduced in (3.10) is equivalent to a problem with a $0 - norm$ objective function of the following form:

$$\arg\min_{{}_kT^{k-1}} \sum_j \sum_i^{N_j} \left|{}_j p_k^i -_j \tilde{p}_{k-1}^i\right|^0 \tag{3.16}$$

where the $0-norm$ is defined as the number of non-zero elements in a vector, therefore it returns 0 when a residual value is 0 and 1 otherwise. Optimizing such a problem is equivalent to finding the solution which satisfies the larger number of "perfect" inliers, which is of course unfeasible in the real world and an estimator of that kind would be unstable. The λ factor prevents that instability and makes the estimator robust to outliers, therefore what (3.10) and (3.11) do, can be seen as minimizing a problem where keypoints having a residual value greater than r_{min}, i.e. the outliers, are evaluated through the $0 - norm$ in the original problem (3.16) and keypoints having a smaller residual, i.e. the inliers, get a constant weight forcing them to be evaluated as in an usual least squares problem.

A general non-linear least squares problem, which is a special case of p-norm objective function based problem with $p = 2$, is defined as follows:

$$\arg\min_{\beta} \sum_{i=1}^{n} |y_i - f_i(\beta)|^2 \tag{3.17}$$

can be solved by solving the normal equations:

$$\left(\mathrm{J^T J}\right)\Delta\beta = \mathrm{J^T}\Delta y \tag{3.18}$$

with respect to $\Delta\beta$, which is the shift vector that updates the previous solution β. Equation (3.18) represents the basis for the Gauss-Newton algorithm for a non-linear least squares problem, which is iteratively solved as:

$$\beta^{(s+1)} = \beta^{s} - \left(\mathrm{J}_{\mathrm{f}}^{\mathrm{T}}\,\mathrm{J}_{\mathrm{f}}\right)^{-1}\,\mathrm{J}_{\mathrm{f}}^{\mathrm{T}}\,\mathrm{r}\left(\beta^{s}\right) \tag{3.19}$$

where $r(\cdot)$ is the residual function $r(\beta^{s}) = y_i - f_i(\beta^{s})$. However, the concept of weight function introduced in (3.10) requires to modify (3.18), which becomes:

$$\left(\mathrm{J}^{\mathrm{T}}\,\mathrm{W}\,\mathrm{J}\right)\Delta\beta = \mathrm{J}^{\mathrm{T}}\,\mathrm{W}\,\Delta y \tag{3.20}$$

(3.19) consequently becomes:

$$\beta^{(s+1)} = \beta^{s} - \left(\mathrm{J}_{\mathrm{f}}^{\mathrm{T}}\,\mathrm{W}\,\mathrm{J}_{\mathrm{f}}\right)^{-1}\,\mathrm{J}_{\mathrm{f}}^{\mathrm{T}}\,\mathrm{W}\,\mathrm{r}\left(\beta^{s}\right) \tag{3.21}$$

Therefore, Equation (3.10) defines the problem and (3.21) shows how to iteratively solve it using the Gauss-Newton algorithm. Finally, the last refinement is represented by an outliers rejection procedure which is done after each iteration, that simply consists in removing remarkable outliers from the minimization pool and then continuing the optimization process until convergence. Although the developed estimator is fairly robust to outliers, removing the most clear of them after each iteration makes the minimization more efficient, since Jacobian and residual computation remain onerous operations which is reasonable to avoid if possible.

3.3 Multiple Frame Optimization

The motion estimation provided by the Iteratively Reweighted Estimator is further improved through a SBA performed over a sliding window. The goal of SBA is to produce a jointly optimal 3D points and camera poses estimation, in this case within a fixed-size window which bounds the computational cost and time. In this application, the 3D points optimization is not the final target of SBA, but it helps to improve the overall quality of camera poses estimation.

The developed SBA works over a sliding window of 45 frames, but it does not actually perform the optimization over the full window, since it only considers one every five frames therefore optimizing an actual problem of 9 frames. Within

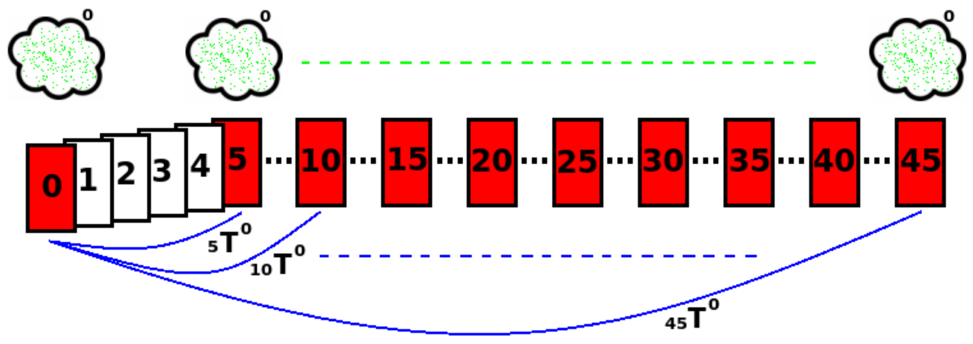


Figure 3.7: The sliding window based SBA considers a window of 45 frames, where only one frame every five is optimized (in red). The camera poses and 3D points are defined with respect to the first frame within the window.

the window, all the camera poses are defined with respect to the first frame of the considered window and, consequently, all the 3D points are defined in the reference system of the same first frame (Figure 3.7).

However, although just 9 frames out of 45 are considered for SBA optimization, the information provided by the other frames is taken into account in order to supply a better initial guess of the optimization: even though a new feature may have been observed only three times in the 9 frames considered for SBA, it has certainly been observed in many more frames considering the full window and those observations are used to provide the initial 3D point estimation when the SBA problem is defined; likewise, the SBA expects an initial guess for the camera poses of the 9 frames and such an estimate is simply provided by concatenating the relative transformations of the frames within the windows obtained by the Iteratively Reweighted Estimator (Figure 3.8).

The advantage of this approach is the quality of the initial guess for every parameter to be optimized, resulting in way less operation in the SBA step, which requires just few iteration to converge. However, regarding 3D point estimation, the described technique is applied only to features that have not yet been optimized in the windowed SBA and, regarding motion estimation, it is applied only to the last camera pose, the

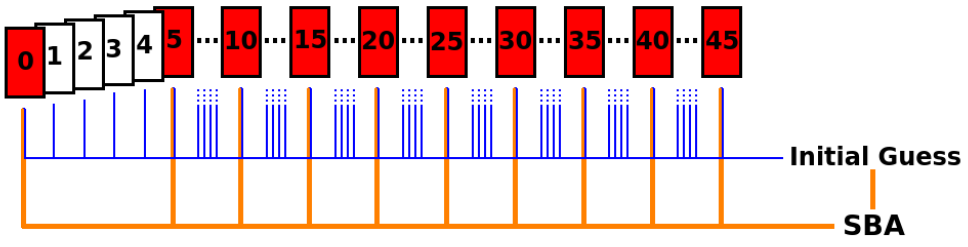


Figure 3.8: Within the SBA window, all the frames are used to compute the initial guess both for new camera poses and new 3D points, while only one every five frames, in red, are actually optimized in the SBA.

only one that was not in the previous SBA window. In other words, the initial guess procedure is applied to bootstrap and to new features and to the new camera pose as well, while in other cases the starting values for estimated parameters come from the previous windowed SBA for the sake of efficiency, requiring just few operations to adapt the previous solutions to the new problem.

Considering Equation (1.7), every 3D point ${}^{0}P_i$ associated to the i-th feature f_i is defined in the reference system of frame 0. At the next windowed SBA computation, the window is shifted of 5 frames and every previously optimized 3D point still alive in the new window is updated by simply using the previously optimized ${}_{5}T^{0}$ as follows:

$$ {}^{5}\bar{P}_i = {}_{5}T^{0}\,{}^{0}\bar{P}_i \tag{3.22}$$

where frame 5 is going to represent the new frame 0 inside the window. In the same way, all the previous camera poses ${}_{i}T^{0}$ have to be updated consistently using the previously optimized ${}_{5}T^{0}$ as follows:

$$ {}_{i}T^{5} = {}_{i}T^{0}\left({}_{5}T^{0}\right)^{\mathrm{T}} \tag{3.23}$$

The number of 3D points optimized is bounded to a maximum in order to limit the computation time, adding first to the problem the 3D points that have been observed more in the frames considered for SBA optimization since those are the one providing more information to the problem solution.

The windowed SBA provides an optimized camera pose every 5 frames, therefore generating a VO estimation with a lower fixed frequency than the Iteratively Reweighted Estimator, somehow bounding the VO drift to the interval between SBA optimized camera poses.

Chapter 4

Results

The full VO system proposed has been tested at a first stage using open datasets and then using sequences acquired by a vehicle specifically designed for algorithm validation. Since VO provides a motion estimation, the sensor that better fits for ground truth generation is of course GPS. Unfortunately, commercial GPS modules have poor accuracy and low frequency which provide a localization error in the order of meters and thus make them unsuitable for ground truth generation. Therefore, a vehicle designed for VO validation has to be equipped with a high accuracy GPS module integrated with IMU, in order to generate a valuable ground truth able to highlight the algorithm performances. Once such ground truth data is provided, a metric to evaluate the algorithm has to be defined.

4.1 Metric

Geiger et Al. in [4] extended [66] and introduced a metric for evaluating VO algorithms commonly adopted in the research community, using GPS+IMU based ground truth camera poses. Basically, the main problem in ground truth generation is that GPS provides an absolute position, while a VO algorithm provides a relative motion between subsequent frames, making the comparison non trivial in order to avoid effects caused by data belonging to different domains. The solution is to evaluate the VO result by generating prefixed-length paths through concatenation of relative transformations and compare them with the correspondent paths generated using GPS+IMU data.

Considering a sequence composed by n frames, a list of n camera poses has to be computed from the GPS+IMU data available as ground truth and from the VO raw data as well. The generated k-th camera pose has as source reference system the one of the k-th frame and as destination reference system a common one which can coincide with the reference system at frame 0 for the sake of simplicity. Therefore, once the k-th ground truth pose is defined as $P_k^{gt} = \left[{}_0R^k|{}^0t_{k\to 0}\right]^{gt}$, the list of n poses $P_{0:n}^{gt} = \left\{P_0^{gt}, ..., P_n^{gt}\right\}$ has to be generated from ground truth data, where the first camera pose could be defined as $[I|0]$ since source and destination coincide in this case.

For each frame, the rotation and translation errors for a list of subsequences of different lengths have to be evaluated. Considering the following list of lengths for the evaluation:

$$L = \{100m, 200m, 300m, 400m, 500m, 600m, 700m, 800m\} \tag{4.1}$$

a given frame k can be associated with 8 paths, each one starting from the frame pose and ending after the corresponding distance in L has been covered. In order to do that, for each frame k the corresponding distance from frame 0 is computed using the following recursive formula:

$$dist_k = dist_{k-1} + \left\|{}^0t_{k-1\to 0}^{gt} - {}^0t_{k\to 0}^{gt}\right\|_2 \tag{4.2}$$

Then, for each frame k and for each i-th length L_i, it is possible to identify the path $p_{k\to j}^i$ where j is the first next frame for which that length is covered, thus the first j that satisfies:

$$dist_j > dist_k + L_i \tag{4.3}$$

Consequently, for every frame k 8 pose error matrices can be computed, each one representing the delta pose between the ground truth pose and the estimated VO pose after covering a particular distance L_i , applying the following:

$$P_{k,L_i}^{err} = \left[R_{k,L_i}^{err}|t_{k,L_i}^{err}\right] = \left[{}_jR^k|{}^jt_{k\to j}\right]^{vo} \left[{}_kR^j|{}^kt_{j\to k}\right]^{gt} \tag{4.4}$$

Defined the pose error matrix P^{err}_{k,L_i}, a translation error function $t_{err}(k, L_i)$ is introduced as

$$t_{err}(k, L_i) = \left\| t^{err}_{k,L_i} \right\|_2 \tag{4.5}$$

and a rotation error function $r_{err}(k, L_i)$ as

$$r_{err}(k, L_i) = \arccos\left(\max\left(\min\left(\frac{\mathrm{Tr}\left(R^{err}_{k,L_i}\right) - 1}{2}, 1\right), -1\right)\right) \tag{4.6}$$

The $t_{err}(k, L_i)$ and $r_{err}(k, L_i)$ error functions produce 16 scalar values for every frame of a sequence, that can be used to generate a graphical representation of the errors considering different covered distances. After those error values have been computed, a reasonable summary can be produced by the introduction of the following performance parameters:

$$t_{avg\_err}(L_i) = \frac{\sum_{k=0}^{n} t_{err}(k, L_i)}{n\, L_i} 100 \tag{4.7}$$

which is the average translation error percentage considering length L_i,

$$t_{err} = \frac{\sum_{i=0}^{7} t_{avg\_err}(L_i)}{7} \tag{4.8}$$

which is the global average translation error percentage,

$$r_{avg\_err}(L_i) = \frac{\sum_{k=0}^{n} r_{err}(k, L_i)}{n\, L_i} 100 \tag{4.9}$$

which is the average rotation error over the length L_i in $deg/100m$,

$$r_{err} = \frac{\sum_{i=0}^{7} r_{avg\_err}(L_i)}{7} \tag{4.10}$$

which is the global average rotation error in $deg/100m$.

4.2 Open Dataset Results

The open dataset used for evaluating the algorithm performance is the KITTI Vision Benchmark Suite [4], which has 11 sequences acquired by a vehicle equipped with a 1.4 Megapixels grayscale stereo camera Point Grey Flea 2 (FL2-14S3M-C), all provided with ground truth trajectories generated using a inertial navigation system OXTS RT 3003.

The VO algorithm has been tested in two configurations: the first one, called IRE, which only uses the Iteratively Reweighted Estimator without the further improvement of the windowed SBA and the second one, called IRE+WSBA, which uses the whole pipeline, in order to also evaluate the improvement of the multiple frame optimization phase. Unfortunately, the KITTI vehicle only has a frontal stereo camera, therefore it is not possible to benefit of the multiple camera feature of the algorithm for this test.

A qualitative evaluation of the algorithm and a comparison between the two configurations is provided in Figure 4.1, 4.2 and 4.3, where the IRE+WSBA configuration clearly outperforms the IRE configuration, with the windowed SBA improving the estimated path alignment with the ground truth trajectory which may corresponds to a smaller rotation error.

A quantitative evaluation, based on the metric provided, is presented in Figure 4.4, which represents the average rotation error as defined in Equation (4.9), and in Figure 4.5, which provides the average translation error percentage as defined in Equation (4.7); the data are displayed in Table 4.1. Table 4.2 shows a comparison between the results obtained from the full system and the public results of some of the top ranked VO and SLAM algorithms [45, 42, 13, 12] present in literature, where the translation and rotation errors are reported for every sequence. Although the comparison with SLAM-based algorithms is unfair, it is interesting to evaluate the current top performances using vision-only algorithms. The overall performance of the proposed method is good, considering the design choices that are derived from the embedded implementation requirements, in particular for rotation error which is definitely comparable to the state of the art; the translation error is usually slightly worse than the competitors' translation error, probably because of the inferior feature extraction and description method adopted, which is somehow dated with respect to the more complex techniques used by the other works in the list. This probably causes a lack of accuracy for image feature coordinates and therefore a worse estimation of the translation component.

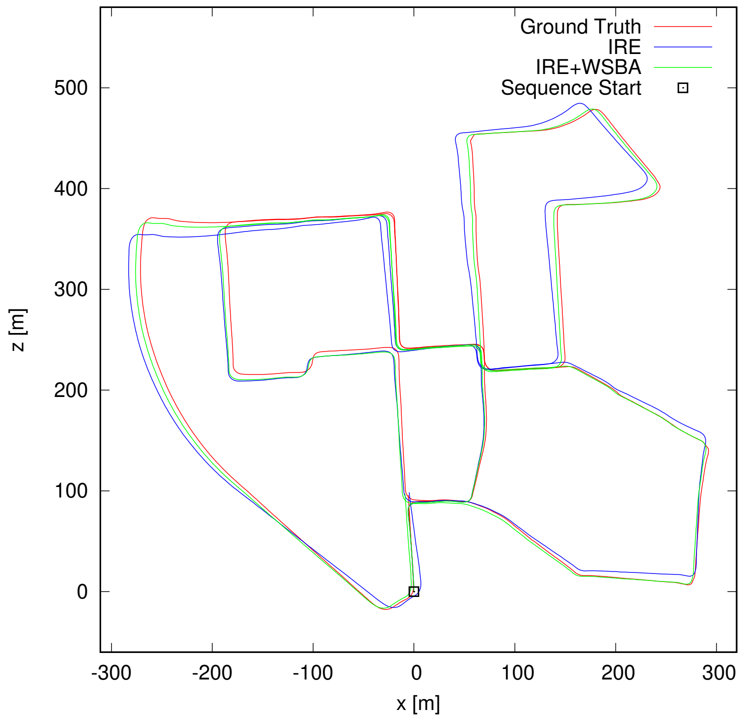


Figure 4.1: The result for sequence 00 of the KITTI dataset. The ground truth trajectory is represented in red, the IRE configuration trajectory in blue, the IRE+WSBA configuration trajectory in green.

	IRE	**IRE + WSBA**
t_{err}	1.16	0.92
r_{err}	0.32	0.25

Table 4.1: Average translation error [%] and rotation error [deg/100m] considering the whole dataset.

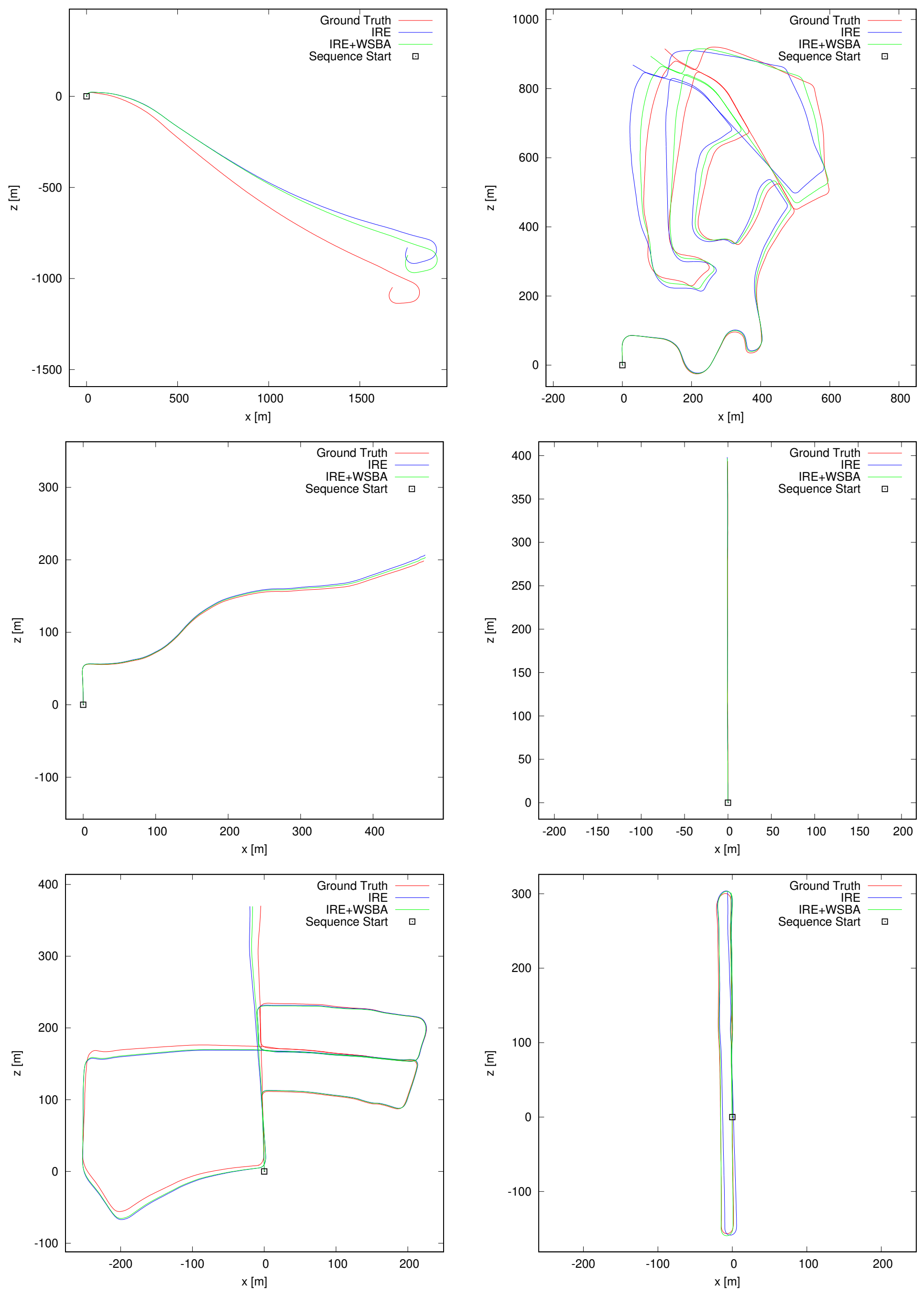


Figure 4.2: The result for sequences 01, 02, 03, 04, 05 and 06 of the KITTI dataset.

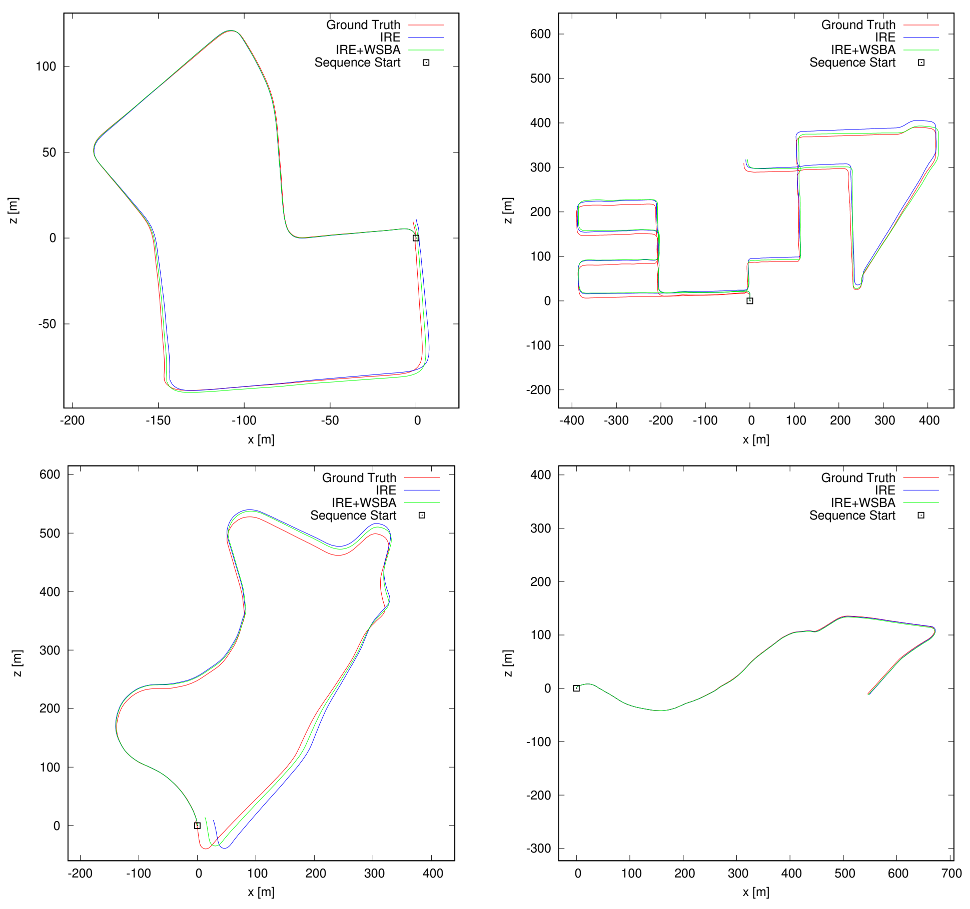


Figure 4.3: The result for sequences 07, 08, 09 and 10 of the KITTI dataset.

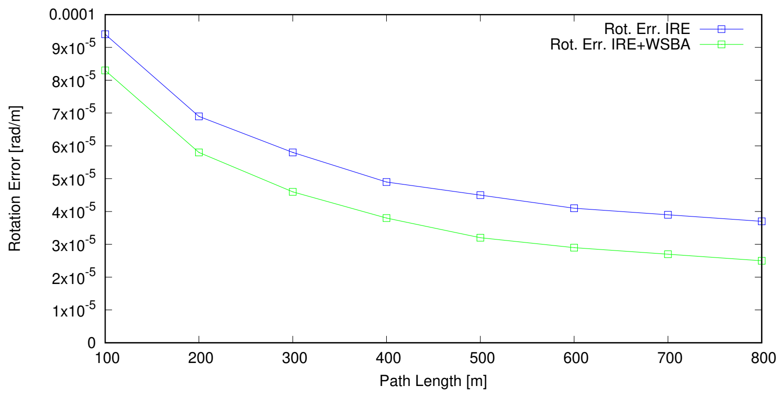


Figure 4.4: The average rotation error. IRE configuration trajectory is represented in blue, IRE+WSBA configuration trajectory in green.

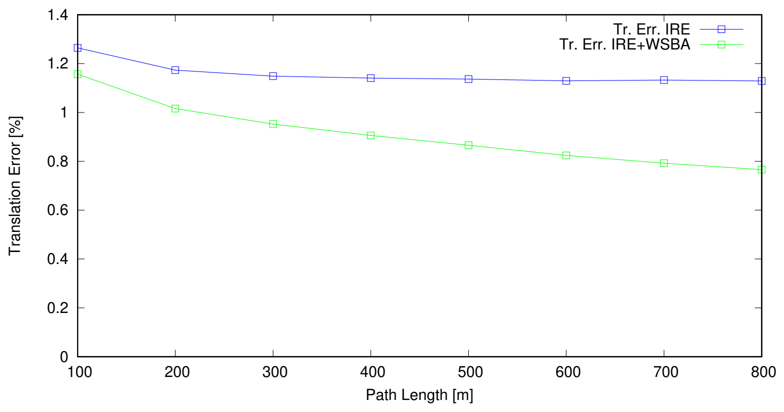


Figure 4.5: The average translation error. IRE configuration trajectory is represented in blue, IRE+WSBA configuration trajectory in green.

	IRE+WSBA		**GDVO**		**SOFT-VO**		**SOFT-SLAM**		**ORB-SLAM2**	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
00	0.74	0.26	0.71	0.53	0.63	0.24	0.66	0.22	0.70	0.25
01	1.81	0.16	1.00	0.65	0.96	0.18	0.96	0.18	1.39	0.21
02	0.97	0.28	0.70	0.41	0.74	0.20	1.36	0.23	0.76	0.23
03	0.98	0.27	0.75	0.37	0.70	0.23	0.70	0.23	0.71	0.18
04	0.95	0.14	0.42	0.64	0.50	0.18	0.50	0.18	0.48	0.13
05	0.80	0.25	0.47	0.38	0.47	0.20	0.43	0.17	0.40	0.16
06	0.92	0.21	0.41	0.40	0.38	0.18	0.41	0.14	0.51	0.15
07	0.84	0.42	0.40	0.50	0.36	0.23	0.36	0.24	0.50	0.28
08	0.92	0.24	0.88	0.39	0.78	0.21	0.78	0.21	1.05	0.32
09	1.13	0.24	0.77	0.35	0.74	0.17	0.59	0.18	0.87	0.27
10	0.54	0.25	0.63	0.41	0.68	0.26	0.68	0.26	0.60	0.27

Table 4.2: Comparison with top ranked state of the art VO and SLAM algorithms. For each sequence, the average rotation error [deg/100m] and the average translation error [%] is reported.

4.3 Validation Vehicle Results

The vehicle used for validation is equipped with two 4K stereo cameras both having a baseline of $0.3m$, one pointed to the front direction of the vehicle and the other to the back, and a Applanix inertial navigation system for providing ground truth data. The objective of the acquired sequences was driven by the need of evaluating the robustness of the algorithm in particular situations, such as in high-speed turns or when driving through roundabouts and speed bumps. Those kind of situations are not faced in the KITTI dataset, but they resulted to be critical cases for odometry estimation. All the results presented in this section have been obtained running the VO algorithm over 1920×1080 images, downscaled from the acquired 4K images.

First of all, some examples of the feature extraction and matching modules are shown in Figure 4.6, where the good stability of the feature extracted is clear, since false matches represent a very small amount of the matched features.

Table 4.3 reports the results for the IRE and IRE+WSBA configurations both in single camera mode and in multi-camera mode; the multi-camera versions of the IRE and IRE+WSBA configurations have been called (M)IRE and (M)IRE+WSBA. The multi-camera mode improves significantly the translation error, thanks to the contribution of the additional viewpoint. On the other hand, the rotation error increases in the multi-camera IRE configuration - without WSBA - probably because the use of multiple cameras makes the algorithm more sensible to the relative calibration of the cameras with respect to the body of the vehicle. The (M)IRE+WSBA configuration does not suffer from this rotation error, because the relative calibration of both cameras is optimized together with the motion, hence the miscalibration is implicitly corrected by the SBA paradigm.

A qualitative evaluation of the algorithm and a comparison is provided by Figure 4.7, 4.8 and 4.9. As expected, the (M)IRE+WSBA configuration is the most performing one, since it significantly reduces the translation error with respect to the single camera IRE+WSBA, while the rotation error is comparable.

In general, the results for the single camera configurations are worse than the results obtained for the KITTI dataset. A reason for this has to be found in the presence of roundabouts, which usually tend to increase the rotation error, while another reason could be that the cameras used for acquiring the sequences are equipped with Rolling Shutter (RS) sensors and not with to the better Global Shutter (GS) sensors used for the KITTI dataset. A RS sensor acquires the image sequentially, thus every line is acquired at a different time, producing a warped image and consequently an estimation error; methods for compensating RS effects exist [67], but in this work those techniques have not been implemented.

	IRE		**IRE+WSBA**		**(M)IRE**		**(M)IRE+WSBA**	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
v_00	1.94	0.43	1.92	0.13	N/A	N/A	N/A	N/A
v_01	1.13	0.40	1.05	0.21	0.93	0.59	0.93	0.25
v_02	1.23	0.36	1.18	0.20	1.10	0.59	0.73	0.16
v_03	1.27	0.54	1.23	0.40	0.60	0.53	0.62	0.25
v_04	1.61	0.78	1.61	0.77	1.47	1.08	1.02	0.78
v_05	1.81	0.93	1.58	0.62	1.04	0.84	1.08	0.62
v_06	1.32	0.45	1.20	0.26	1.06	0.68	0.74	0.19
v_07	1.29	1.51	1.28	1.26	1.41	1.49	1.10	0.93
v_08	1.07	0.33	1.16	0.11	1.07	1.28	0.41	0.46
AVG	1.41	0.64	1.36	0.44	1.09	0.89	0.83	0.46

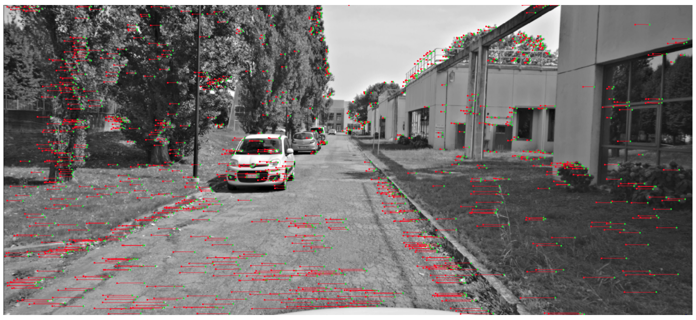
Table 4.3: Average translation error [%] and rotation error [deg/100m] obtained for every sequence.

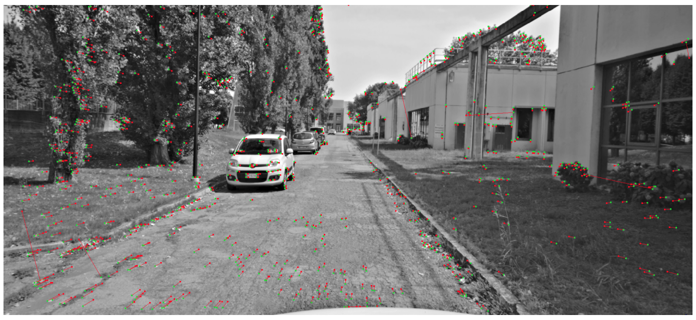
4.4 Computation Time

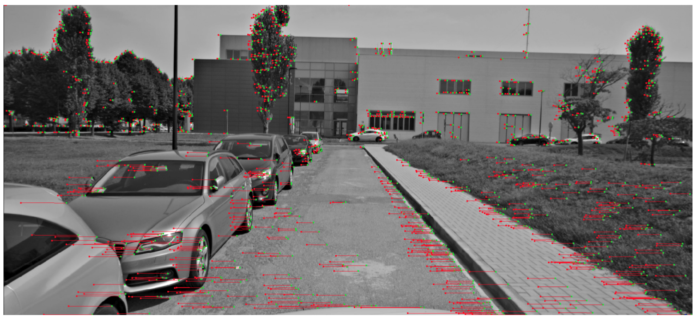
The computation time of the different modules has been measured running on the embedded platform, which presents dedicated units for handling feature extraction and matching, and an ARM® CORTEX™ A53 for general purpose tasks. The results for the different components of the system have been collected through testing of the full system on the validation vehicle; the average computation time and the corresponding average standard deviation for each module are the followings:

- The feature extraction and description module requires 7.0ms for a single camera, considering 2048 features extracted from each view, hence this time it is almost halved in the monocular case. Thanks to the dedicated hardware module it is very stable and its average standard deviation is 0.2ms.
- The feature matching module requires a total of 3.0ms to match features from the left and right view and then to match the obtained stereo features with the previous stereo features, hence this time is almost halved in the monocular case. Thanks to the dedicated hardware module it is very stable and its average standard deviation is 0.1ms.
- The IRE estimation module always requires 2.1ms since the computation time is directly proportional to the number of features and, in case of multiple cameras, the maximum number of features is the same and the amount is equally divided between the available views. In this case the execution is performed by the ARM® processor, which determines a more variable computation time characterized by an average standard deviation of 0.3 ms.
- The WSBA optimization module requires 30.2ms every 5 frames, hence an equivalent amount of 6.0ms for each frame. The time required does not depend on the number of frames, since the maximum number of features is fixed and equally divided between the available views. The total average standard deviation is 4ms.

The total computation time required for a single camera stereo configuration is 18.1ms through simple addition of the times mentioned above. For the two stereo camera case, with a hardware module for feature extraction and a different hardware module for feature matching - which is indeed the case of the hardware platform used for the embedded implementation - the resulting pipeline structure limits the overall computation time to 25 ms (see Figure 4.10).


(a)


(b)


(c)

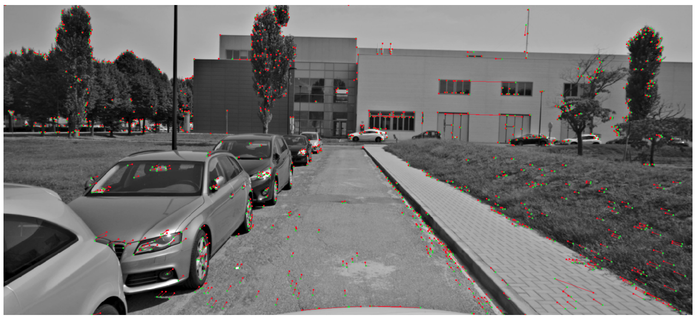
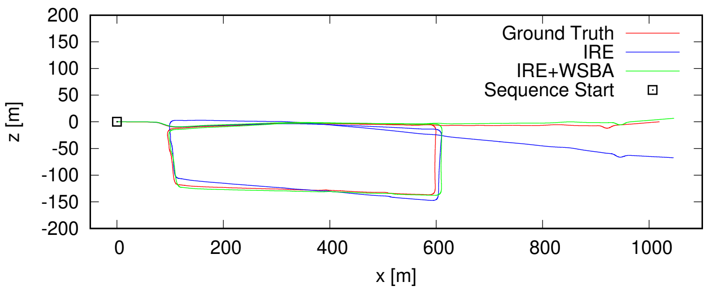
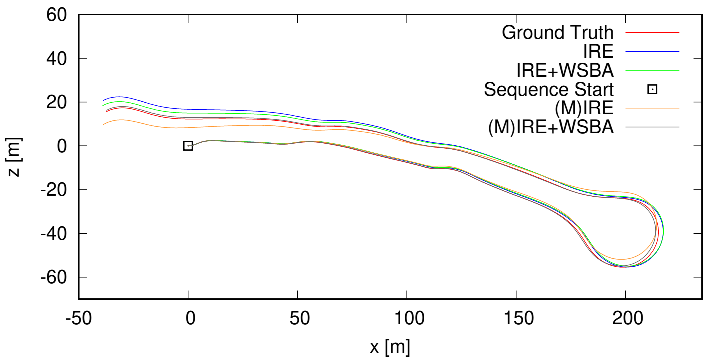

(d)

Figure 4.6: Examples of feature extraction and matching: the stereo match result in (a) and (c); the flow match result in (b) and (d).





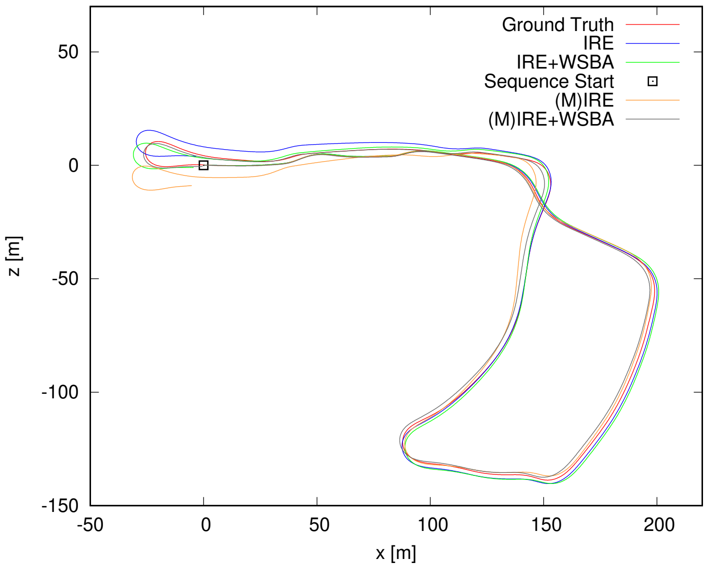
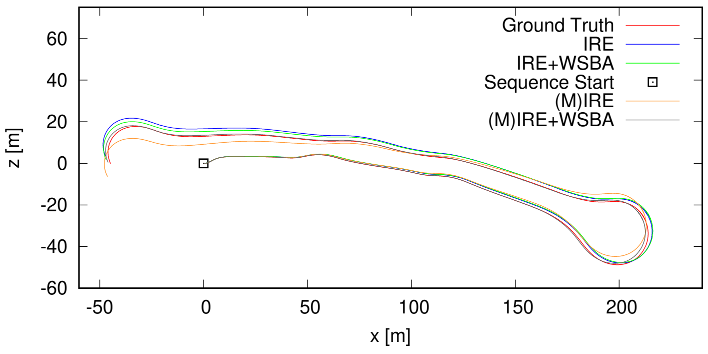
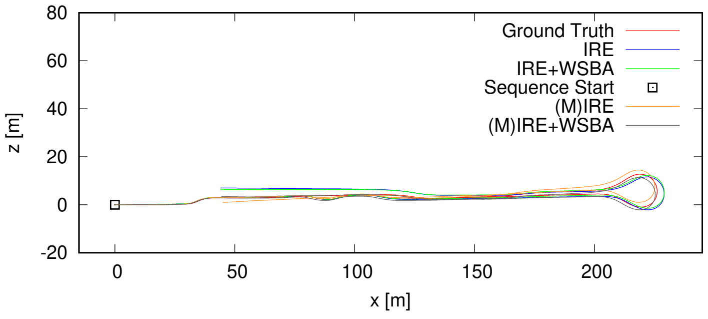


Figure 4.7: The result for sequences v_00, v_01 and v_04 of the dataset.





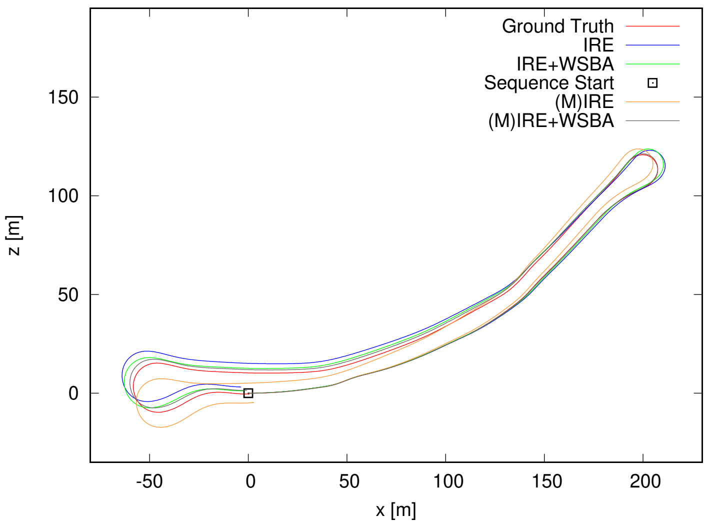
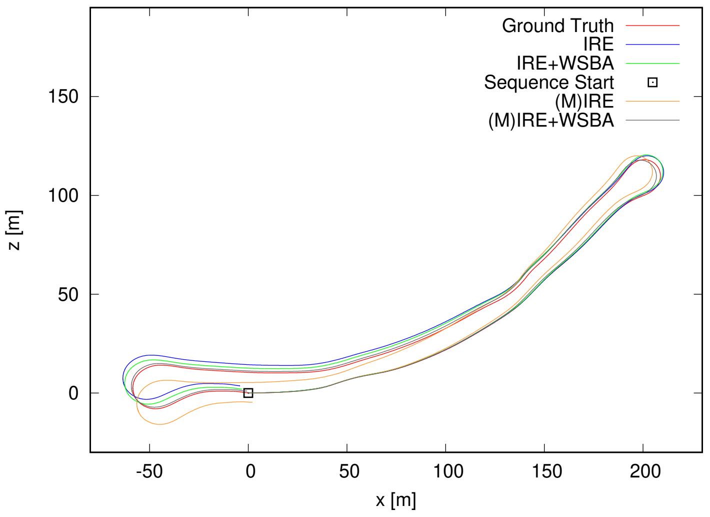
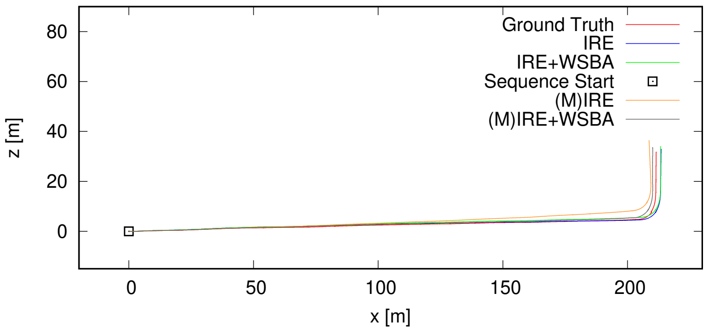


Figure 4.8: The result for sequences v_02, v_03 and v_05 of the dataset.





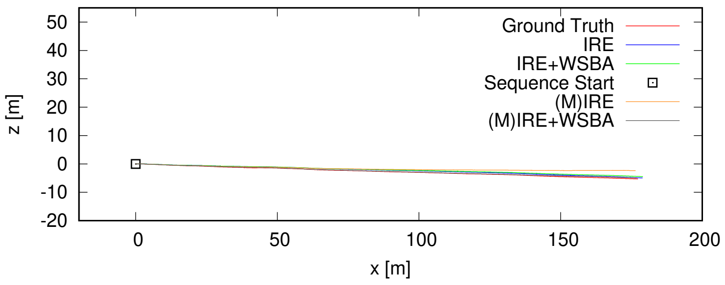


Figure 4.9: The result for sequences v_06, v_07 and v_08 of the dataset.

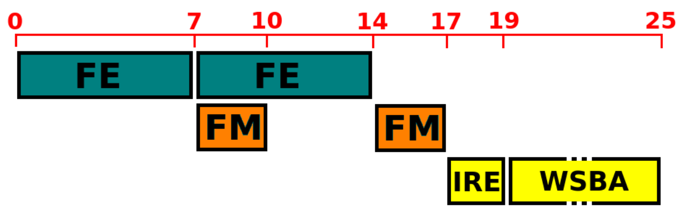


Figure 4.10: The computation time required for a configuration with two stereo cameras.

Chapter 5

Conclusions

The aim of this thesis was to outline a modular system for real-time visual odometry estimation. The developed method has been designed to be implemented on an embedded system, composed by one or multiple cameras equipped with an on-board computer vision processor. The proposed system is therefore adaptable to different configurations, since it works using a single stereo camera or a stereo camera plus other cameras, either monocular or stereo. The main requirements were the real-time execution at 30 frames per second using 1920×1080 images and a limited cost in terms of memory. The motion estimator is robust to outliers and converges in few iterations, thanks to an iterative reweighting scheme. A further multiple frame optimization technique has been developed in order to improve the overall quality of the estimated motion, exploiting a Sparse Bundle Adjustment over a sliding window.

The proposed system has been evaluated using a well-known open dataset and acquiring new sequences through the use of a vehicle specifically designed for image acquisition and ground truth generation. The metric adopted for evaluation, which is very common in literature, provides a rotation error and a translation error that represent a simple and fast way to compare different methods. For both the open and the acquired datasets, the proposed method has been tested also having the multiple frame optimization disabled, in order to evaluate the contribution of the multiple frame optimization.

The results obtained on the open dataset have been compared to some of the most performing VO and SLAM algorithms in literature, although the dataset only allows to test the proposed method in the single-camera configuration. The comparison shows that the full system provides a motion estimation characterized by a rotation error comparable to that achieved by top ranked works, 0.25 deg/100m on average, while the translation error is slightly worse although it is close to 0.9% on average, which is definitely a good result. The most probable reason for this lies in the weaker feature extraction and description method adopted, with respect to the advanced techniques used by other works in the comparison list, since it probably provides weaker image features and causes a lack of accuracy in the estimation of the translation component. For what concerns the multiple frame optimization, the results show that it definitely improves the motion estimation, reducing the rotation and translation errors of about 20%.

The dataset acquired using the validation vehicle has been structured to put to test the algorithm with some difficult scenarios that open datasets miss, such as performing high-speed turns or going through roundabouts and speed bumps, situations that are usually encountered when driving in many European roads. Most of the acquired sequences have been recorded using two stereo cameras, one looking to the front and the other to the back of the vehicle, which have made it possible to evaluate the full system in the multiple camera configuration. Therefore, the acquired dataset has been used to propose a comparison between the single-camera and the multi-camera configurations, in order to highlight the contribution of the multi-camera option. The results revealed that the multi-camera approach is able to significantly improve the translation estimation, reducing on average the translation error of about 39% with respect to the single-camera configuration. Considering the whole acquired dataset, the proposed method provided an average translation error of 0.83% and an average rotation error of 0.46 deg/100m.

The proposed algorithm fulfills the initial requirements, since it does not exploit any kind of map, thus it requires a limited amount of memory, and it takes 18 ms to run in the single-camera configuration on the low-power embedded platform adopted, and 25 ms in the configuration that uses two stereo cameras.

This thesis has enlightened the importance of exploiting multiple views in order to achieve accurate motion estimation and 3D reconstruction. The use of multiple views in the time domain, which is what a Bundle Adjustment technique does, significantly improves the accuracy in the estimation of the rotation component of motion. In the same way, the use of multiple views in the space domain, enable by a multi-camera configuration, considerably enhances the accuracy in the estimation of the translation component of motion. The combination of these concepts provides a reliable and accurate motion estimation method, moreover the system results robust to temporary occlusions thanks to the use of multiple viewpoints.

An obvious improvement that has to be done is enabling the VO estimation when using a single monocular camera. Of course, the information about scale cannot be easily retrieved using just a monocular camera, but integrating the scale information from a different kind of sensor is a straightforward solution, that does not present drawbacks. For what concerns feature extraction, a more performing method should be implemented in order to improve feature accuracy.

Besides the improvements obtained here exploiting multiple views, redundancy and sensors integration is the only way to achieve autonomous driving. The real world is something that cannot be completely predicted. While research is making tremendous improvements in computer vision, car accidents - which sadly occurred in some recent experiments of unsupervised autonomous driving - pointed out how such systems cannot rely on a single or few sensors. A method can be trained and tested over millions of different situations - and to be honest this is not the case yet, even for already commercialized systems - but the unpredictable, the black swan, will always come out and, at that point, it won't be just a drop in the results of a paper.

In order to honor what inspired the first milestones in computer vision applied to autonomous driving, that is the development of a safer and sustainable transportation, the keyword must be $reliability$. An autonomous car has to rely on multiple sensors that must be able to cooperate in order to work better, whichever the task is, but that can also be able to provide a fair result in case of temporary fault of some of them, allowing the vehicle to safely and properly put itself in a safe condition in case of need. In autonomous driving the winning solution has to be the fusion and integration of different sensors, because if a failure - which may result in the loss of a life - can be avoided by integrating a different sensor in the system, that is the way to go. It may not be the minimal solution and neither the cheapest, but it definitely is the only one acceptable. The recent run of car makers in autonomous driving wonderfully boosted the research all over the world, but everyone should keep in mind that if people feel autonomous driving unsafe, this crazy rush will rapidly end as it started.

Appendix A

Notation

The notation adopted for defining rigid transformations and vectors can be outlined with the following points:

- A transformation matrix is denoted by a capitol letter, for instance a relative transformation is usually denoted by T.
- A transformation matrix is composed by a rotation matrix R and a translation vector t.
- A transformation matrix that transforms from the source reference system s to the destination reference system d is denoted by ${}_dT^s$.
- The rotation matrix which defines the transformation matrix ${}_dT^s$ is denoted by ${}_dR^s$, where s is the source reference system and d is the destination reference system.
- The translation vector which defines the transformation matrix ${}_dT^s$ is denoted by ${}^dt_{d\rightarrow s}$, where the left superscript denotes that the vector is defined in the destination reference system and right subscript denotes the sense of the vector.
- A transformation matrix ${}_dT^s$ can be applied by applying the rotation matrix ${}_dR^s$ first and then by adding the translation vector ${}^dt_{d\rightarrow s}$.

Given those points, a transformation matrix ${}_dT^s$ has the following form:

$${}_dT^s = \begin{bmatrix} {}_dR^s & {}^dt_{d\to s} \\ 0 & 1 \end{bmatrix}$$

Bibliography

[1] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. Ieee, 2004.

[2] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980.

[3] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[5] Martin Peris, Sara Martull, Atsuto Maki, Yasuhiro Ohkawa, and Kazuhiro Fukui. Towards a simulation driven stereo vision system. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1038–1042. IEEE, 2012.

[6] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[7] JK Aggarwal and N Nandhakumar. On the computation of motion from sequences of images-a review. *Proceedings of the IEEE*, 76(8):917–935, 1988.

[8] Peter Corke, Dennis Strelow, and Sanjiv Singh. Omnidirectional visual odometry for a planetary rover. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 4007–4012. IEEE, 2004.

[9] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[10] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[11] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[12] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[13] I Cvisic, J Cesic, I Markovic, and I Petrovic. Soft-slam: Computationally efficient stereo visual slam for autonomous uavs. *Journal of Field Robotics*, 2017.

[14] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[15] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International journal of computer vision*, 94(2):198–214, 2011.

[16] Hauke Strasdat, J Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2, 2010.

[17] Carlos Estrada, José Neira, and Juan D Tardós. Hierarchical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[18] Niko Sünderhauf, Kurt Konolige, Simon Lacroix, and Peter Protzel. Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In *Autonome Mobile Systeme 2005*, pages 157–163. Springer, 2006.

[19] Kurt Konolige, Motilal Agrawal, and Joan Sola. Large-scale visual odometry for rough terrain. In *Robotics research*, pages 201–212. Springer, 2010.

[20] Friedrich Fraundorfer, Davide Scaramuzza, and Marc Pollefeys. A constricted bundle adjustment parameterization for relative scale estimation in visual odometry. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1899–1904. IEEE, 2010.

[21] Jean-Philippe Tardif, Michael George, Michel Laverne, Alonzo Kelly, and Anthony Stentz. A new approach to vision-aided inertial navigation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4161–4168. IEEE, 2010.

[22] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and automation, 2007 IEEE international conference on*, pages 3565–3572. IEEE, 2007.

[23] Eagle S Jones and Stefano Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430, 2011.

[24] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. IEEE, 2011.

[25] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[26] Michael Kaess, Kai Ni, and Frank Dellaert. Flow separation for fast and robust stereo odometry. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3539–3544. IEEE, 2009.

[27] Keju Peng, Xin Chen, Dongxiang Zhou, and Yunhui Liu. 3d reconstruction based on sift and harris feature points. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 960–964. IEEE, 2009.

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

[29] Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3946–3952. IEEE, 2008.

[30] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[31] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[32] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[33] Matthew Brown and David G Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, pages 56–63. IEEE, 2005.

[34] Chris Beall, Brian J Lawrence, Viorela Ila, and Frank Dellaert. 3d reconstruction of underwater structures. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4418–4423. IEEE, 2010.

[35] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[36] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 486–492. IEEE, 2010.

[37] Larry Matthies and STEVENA Shafer. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, 1987.

[38] Clark F Olson, Larry H Matthies, Marcel Schoppers, and Mark W Maimone. Rover navigation using stereo ego-motion. *Robotics and Autonomous Systems*, 43(4):215–229, 2003.

[39] Yang Cheng, Mark W Maimone, and Larry Matthies. Visual odometry on the mars exploration rovers. *IEEE Robotics and Automation magazine*, 13(2):54, 2006.

[40] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[41] Andrew I Comport, Ezio Malis, and Patrick Rives. Accurate quadrifocal tracking for robust 3d visual odometry. Citeseer.

[42] Igor Cvišić and Ivan Petrović. Stereo odometry based on careful feature selection and tracking. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–6. IEEE, 2015.

[43] Martin Buczko and Volker Willert. Flow-decoupled normalized reprojection error for visual odometry. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 1161–1167. IEEE, 2016.

[44] Hernán Badino, Akihiro Yamamoto, and Takeo Kanade. Visual odometry by multi-frame feature integration. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 222–229, 2013.

[45] Jianke Zhu. Image gradient-based joint direct visual odometry for stereo camera. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4558–4564, 2017.

[46] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[47] Andrew J Davison and David W Murray. Simultaneous localization and map-building using active vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):865–880, 2002.

[48] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[49] Dennis Strelow and Sanjiv Singh. Motion estimation from image and inertial measurements. *The International Journal of Robotics Research*, 23(12):1157–1195, 2004.

[50] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[51] Jean-Philippe Tardif, Yanis Pavlidis, and Kostas Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2531–2538. IEEE, 2008.

[52] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 363–370. IEEE, 2006.

[53] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[54] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Accurate non-iterative o (n) solution to the pnp problem. In *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*, pages 1–8. IEEE, 2007.

[55] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. 2011.

[56] David Nistér and Henrik Stewénius. A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision*, 27(1):67–79, 2007.

[57] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[58] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, et al. Building rome on a cloudless day. In *European Conference on Computer Vision*, pages 368–381. Springer, 2010.

[59] Manolis Lourakis and Antonis Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical report.

[60] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[61] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[62] Yasushi Kanazawa and Kenichi Kanatani. Reliability of 3-d reconstruction by stereo vision. *IEICE TRANSACTIONS on Information and Systems*, 78(10):1301–1306, 1995.

[63] Kenichi Kanatani, Yasuyuki Sugaya, and Hirotaka Niitsuma. Triangulation from two views revisited: Hartley-sturm vs. optimal correction. *In practice*, 4:5, 2008.

[64] Peter Lindstrom. Triangulation made easy. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1554–1561. IEEE, 2010.

[65] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[66] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387, 2009.

[67] Alberto Broggi, Alessandro Cionini, Francesca Ghidini, and Paolo Zani. Handling rolling shutter effects on semi-global matching in automotive scenarios. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 1134–1139. IEEE, 2017.

So Long, and Thanks for All the Fish