Universidade Estadual de Campinas
Instituto de Computação

Ronnypetson Souza da Silva

Monocular Visual Odometry Based on Epipolar Geometry

Odometria Visual Monocular Baseada na Geometria Epipolar

CAMPINAS
2021

Ronnypetson Souza da Silva

Monocular Visual Odometry Based on Epipolar Geometry

Odometria Visual Monocular Baseada na Geometria Epipolar

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientadora: Profa. Dra. Esther Luna Colombini

Este exemplar corresponde à versão final da Dissertação defendida por Ronnypetson Souza da Silva e orientada pela Profa. Dra. Esther Luna Colombini.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Si38m

Silva, Ronnypetson Souza da, 1994-
Monocular visual odometry based on epipolar geometry / Ronnypetson Souza da Silva. – Campinas, SP : [s.n.], 2021.

Orientador: Esther Luna Colombini.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Visão por computador. 2. Inteligência artificial. 3. Robótica. 4. Odometria visual. 5. Processamento de imagens. 6. Processamento de imagens - Técnicas digitais. I. Colombini, Esther Luna, 1980-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Odometria visual monocular baseada na geometria epipolar
Palavras-chave em inglês:
Computer vision
Artificial intelligence
Robotics
Visual odometry
Image processing
Image processing - Digital techniques
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Esther Luna Colombini [Orientador]
Hélio Pedrini
Luiz Marcos Garcia Gonçalves
Data de defesa: 24-02-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-7955-238X
- Currículo Lattes do autor: http://lattes.cnpq.br/9589225919316577

Universidade Estadual de Campinas
Instituto de Computação

Ronnypetson Souza da Silva

Monocular Visual Odometry Based on Epipolar Geometry

Odometria Visual Monocular Baseada na Geometria Epipolar

Banca Examinadora:

- Profa. Dra. Esther Luna Colombini
  IC/UNICAMP
- Prof. Dr. Hélio Pedrini
  IC/UNICAMP
- Prof. Dr. Luiz Marcos Garcia Gonçalves
  DCA/UFRN

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 24 de fevereiro de 2021

Dedication

To my mother Maria Helena, who not just fostered me and my siblings with uttermost dedication, but loved us and fought for us bravely. She was a woman of immense hope and faith and moved mountains with her will. Death did not stop her, as her will moves inside the ones who love her and will go on until the end of times.

Acknowledgements

I would like to thank my mother Maria Helena and my father José Romildo for being lovely and supportive parents from the beginning. I cannot pay back what they gave me, for it is so much and beyond the reach of money and power.

I also would like to thank my brother Rafael and my sister Renata. They are unconditional lovers and I was happy growing up with them.

I thank my academic advisor, Prof. PhD Esther Luna Colombini, who received me in this program, supported me, and had admirable patience throughout the process.

I also thank my colleagues and friends at UNICAMP and CPQD, as those places have provided me with valuable experience in both industry and academy.

The present work was carried out with the support of CNPq, *Conselho Nacional de Desenvolvimento Científico e Tecnológico* - Brazil.

Resumo

Na Odometria Visual Monocular, existem várias abordagens relacionadas à otimização de pose e estrutura. Algumas permitem o uso de observações em granularidade fina (ex.: pontos de interesse), mas requerem um número grande de parâmetros. Isso limita a densidade da reconstrução de estrutura e o número de poses que podem ser otimizadas em tempo real. Nós investigamos uma caracterização com poucos parâmetros da Odometria Visual que consiste apenas de poses e das posições de epipolos em pares de imagens. Fazemos isso formulando um tipo indireto de *Bundle Adjustment*, onde parâmetros de estrutura como profundidade inversa de pontos são substituídos por epipolos da geometria de dois pontos de vista. Nossa formulação permite a reconstrução da cena com a mesma densidade que qualquer conjunto de associações de pontos-chave. Além disso, tal reconstrução não aumenta o número de parâmetros quando o número de pontos aumenta e possui custo computacional de tempo linear no número de pontos. Para verificar se a formulação proposta reflete dados do mundo real, construímos uma *pipeline* de Odometria Visual baseada em características (indireta) que minimiza o erro da reprojeção com relação às poses e epipolos. Nós usamos gradientes de segunda ordem para otimizar as poses na álgebra do SE(3) e os epipolos em seu espaço original. Daí executamos a odometria e a reconstrução de cena na base de dados KITTI. Os experimentos mostram que a abordagem proposta de fato se encaixa com observações reais e tem resultados de odometria compatíveis com os da literatura, ao mesmo tempo que exerce controle sobre a estrutura sem usar parâmetros de pontos.

Abstract

In Visual Odometry, there are many approaches related to the optimization of pose and structure. Some allow for fine-grained use of the observations, like feature points, but require many parameters. This limits the density of structure reconstruction and the number of poses that can be optimized in real-time. In this work, we propose a low-parameter characterization of Visual Odometry that consists solely of poses and the locations of eipoles in pairs of images. We do that by formulating an indirect type of Bundle Adjustment that consists of replacing structure parameters like inverse point depths with epipoles from two-view geometry. Our formulation allows us to reconstruct the scene with the same density as any set of point associations provided. Furthermore, this reconstruction does not increase the number of parameters when the number of points increases, and it has a linear computational time cost on the number of points. To check if the proposed formulation fits real-world data, we build a feature-based (indirect) VO pipeline that minimizes the reprojection error concerning poses and epipoles. We employ second-order gradients to optimize the poses in the algebra of SE(3) and the epipoles in their original space. Then, we perform odometry and structure reconstruction in the KITTI dataset. Experiments show that the proposed approach indeed fits real observations and has odometry results compatible with the literature while allowing for some control over structure without point parameters.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The term "odometry" is derived from "odometer", a sensor of wheeled vehicles that measures the amount of wheel rotation. This type of measurement was one of the first used to estimate ego-motion on the ground. Nevertheless, the word "odometry" has been used lately to describe motion computation in the general sense, including other sensors (e.g., camera, IMU) and movement not restricted to the ground (e.g., underwater, aerial). In this context, *Visual Odometry* (VO) is the odometry based solely on apparent movement in a sequence of images from one or more cameras attached to a vehicle (Nister et al. [42]).

Monocular Visual Odometry differs from *Stereo Visual Odometry* on the type of camera view: in the first, there is only a single camera not provided with any stereo rig (e.g., the camera is static concerning the vehicle), while the second usually involves two or more cameras. If a single camera slides over a stereo rig to take multiple images when the vehicle is in the same position, it is also considered stereo, such as in the work of Moravec [39].

VO is an essential component in mobile robotics, as it continuously provides the system with information about the current location and orientation inside a map of the environment. In turn, this allows for the execution of tasks inside that environment. Moreover, VO does not depend on the robot model or dynamics and can be developed independently of other components, although information about the robot model might be useful in some cases. VO also has applications in *Virtual Reality* (VR). For instance, the combination of virtual objects and characters in real scenes fits the VO approach well.

In VO and Visual SLAM (VSLAM), we want to estimate ego-motion from a time-sequence of images in real time and also create a map of the environment. In the ideal case, if we know the first, we can reconstruct the second from geometric constraints. But in practice, there are problems in measurements like noise and wrong associations of scene primitives (e.g. points, lines). Many of the works in VO/VSLAM treat the data association problem with outlier filtering using robust methods like RANSAC (Fischler and Bolles [16]). The noise in measurements is supposed to vanish away as we observe more (filtered) points in the scene and minimize the reprojection error. Some works in VO/VSLAM do minimize the reprojection error in a windowed Bundle Adjustment (BA) setup with parameters related to point depth, where inverse point depth is a popular choice (Engel et al. [12]). Some of the works that do not include point parameters also do

not minimize the reprojection error in BA (Engel et al. [10, 11]), but optimize pose with pose-graph optimization, which does not depend on point measurements, but adjusts the initialization values of poses. There are also methods that do not use pose-graph optimization nor BA with point parameters and they work like "concatenative" systems that optimize only in small windows and will not have features like loop-closure (Pereira et al. [46], Wang et al. [58]) but might have some sense of global optimization if using data-driven techniques like Deep Learning (Parisotto et al. [44]). An interesting case is a subroutine of ORB-SLAM (Mur-Artal et al. [40]), which has point parameters but the point depths are frozen at their current value and BA is performed optimizing only poses. Nothing restricts the use of both BA and pose-graph optimization, and it is done in ORB-SLAM. Regardless of the choice of pose optimization method, the scene can be reconstructed by triangulation of the associated points, although it can be polluted if the point association only happens in short intervals of time but the points are observed in longer periods. Therefore, joint optimization of pose and structure in BA have the possibility of better map construction, although the optimization gets harder as more parameters enter the system.

Although there are many aspects of VO that can be subject to optimization, like that of scene primitives, in this work we are only interested in parametrization to optimize pose and structure, which are the main goals of VO and VSLAM. In this work, we propose a low-parameter characterization of Visual Odometry that consists solely of poses and the locations of epipoles in pairs of images. We do that by formulating an indirect type of BA that consists of replacing structure parameters like inverse point depths with epipoles from two-view geometry. The proposed approach reduces the number of parameters drastically and still allows for some control over structure optimization.

More specifically, we propose to minimize the reprojection error in BA without attributing directly one parameter for each key-point. We do so by writing the point depth as a function of the pose, epipole, and the point association related to the epipole. This form of depth is easy to manipulate for analytical purposes and allows for Jacobians' derivation of the reprojection error concerning poses and epipoles without the need for point parameters. Hence, efficient second-order optimization routines like Levenberg-Marquardt can be employed to allow real-time applications. However, we leave the investigation of the practical use of our formulation for future work.

As contributions of this work, we can list the following. First, this work observes a geometrically sound connection between camera pose (orientation and position), second epipole from two-view geometry, and point depth. It also considers the generalization for sequences with more than two camera frames to be compatible with windowed optimization. Then, we propose a VO pipeline that incorporates the parametrization to validate our formulation.

1.1 Objective

In this work, our goal is to propose a characterization of VO that relies on a small number of parameters and can still adjust pose and geometry to some extent. Although there are

many aspects of VO that can be subject to optimization, like scene primitives, in this work, we are only interested in parametrization to optimize pose and structure, which are VO's primary goals. Furthermore, we want to implement a prototype feature-based monocular VO system based on such parametrization and test it on a well-established VO benchmark.

We want to find such a few-parameter VO based on two-view geometry, which regards the relationship between points in a 3D scene and its images in two *pinhole cameras* (not to be confused with physical cameras).

1.2 Contributions

As contributions of this work, we can list the following. First, this work finds a geometrically sound connection between camera pose (orientation and position), second epipole from two-view geometry, and point depth. It generalizes the relation for sequences with more than two camera frames to be compatible with windowed optimization. Then, we propose a VO pipeline that incorporates the parametrization and implement an open-source prototype of the algorithm[1]. Then, we analyze the algorithm's results on a set of sequences from an outdoor dataset. That analysis gives intuition about what can be modified to improve the algorithm in terms of quality and speed, making running on more complex configurations possible. That, in turn, would allow for a more in-depth investigation of the parametrization that we propose.

Second, the above contributions employed an open-source automatic second-order optimization framework from two other existing frameworks[2]. Although it is a straightforward way to implement the VO algorithm's optimization, it is only useful for prototyping and not real-world applications. Hence, we provide an efficient C++ implementation of the optimization[3].

1.3 Research Questions

Considering the following requirements for a VO system: **(i)** the use of windowed optimization of poses and geometry; **(ii)** a small number of optimization parameters that do not depend on the number of key-points; **(iii)** it models the key-point depths, even if indirectly, we make the following questions:

- Is there a formulation of the reprojection error in terms of pose parameters plus a constant number (per frame) of parameters that also model the depth of key-points in the 3D scene?
- If so, is it compatible with windowed optimization Bundle Adjustment?
- Can it be implemented in a VO pipeline?

[1] `https://github.com/Ronnypetson/tfoe_vo`
[2] `https://github.com/Ronnypetson/sotorch`
[3] `https://github.com/Ronnypetson/epivo`

1.4 Outline

The remaining of this work is organized as follows. In Chapter 2 we explain many important concepts concerning the VO pipeline, including camera projection geometry and the close view of two cameras to a mathematical optimization apparatus, selection, and matching of image points informative ego-motion. Then, in Chapter 3, we summarize some representative work from classic SLAM/VO based on regular Bundle Adjustment and optimization of the reprojection error. We also summarize application-oriented work and applications of learning-based methods to the SLAM/VO pipeline. In Chapter 4, we introduce the pose plus epipole parametrization and describe the Epipolar VO algorithm in detail, including the algorithm's parametrization role. Then Chapters 5 and 6 describe the methodology of the experiments and give a detailed analysis of the results from Epipolar VO in 11 outdoors sequences. Chapter 7 contains general conclusions about this work and gives some directions to investigate better the parametrization we present.

Chapter 2

Elements of Visual Odometry

This chapter covers some basic principles that support Visual Odometry algorithms, from basic geometric concepts, going through camera pose representations and two-view geometry to specific algorithms for key-point extraction and matching. It also supports the introduction of the Epipolar VO algorithm proposed in Chapter 4.

2.1 3D Euclidean Space and the Pinhole Camera Model

To address the problem of estimating and reconstructing the 3D geometry of a pose, it is necessary to have in mind some basic geometric concepts. First, consider points in space as represented as elements of the three-dimensional Euclidean space $\mathbb{R}^3$. In this sense, given a Cartesian coordinate system of reference (Figure 2.1), every point has its location pinpointed by three real numbers, and every triple of real numbers represents only one point in space. Therefore, this characterizes the fitting of points in space inside the mentioned 3D Euclidean space without ambiguity. The *origin* of such coordinate system is where the axis meet. By definition, the point $(0, 0, 0)$ means the point at position 0 in all three axes. The angle between any two of the three axes is always 90 degrees.

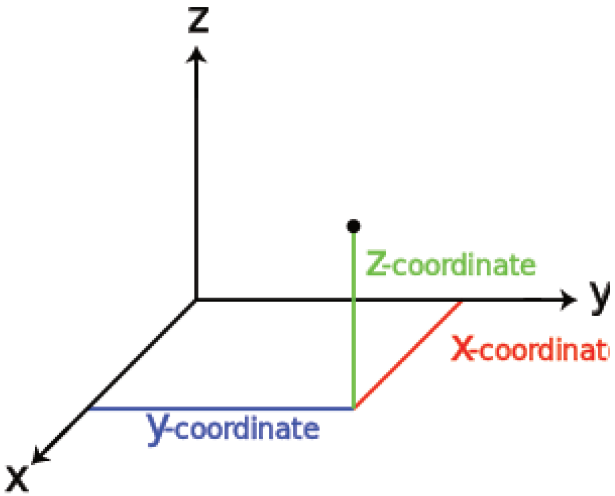


Figure 2.1: Cartesian coordinate system.

In a geometric sense and for most practical purposes, a camera is a mapping from

points in a three-dimensional scene inside the 3D Euclidean space to a plane, which has two dimensions and is equivalent to the 2D Euclidean space. Such mapping is a non-linear transformation, nevertheless it can be implemented by using a multiplication by a matrix followed by a perspective division.

The simplest camera model is known as the *pinhole* camera (Hartley and Zisserman [24]) (Figure 2.2). In this model, the projection of a point X in the scene is the intersection $\mathbf{x}$ of the line connecting X and the center of projection (or camera center) C with the image plane $Z = f$, considering a Euclidean coordinate system with the origin at C and its Z axis being the "forward-looking" direction of the camera. f is known as the *focal distance*.

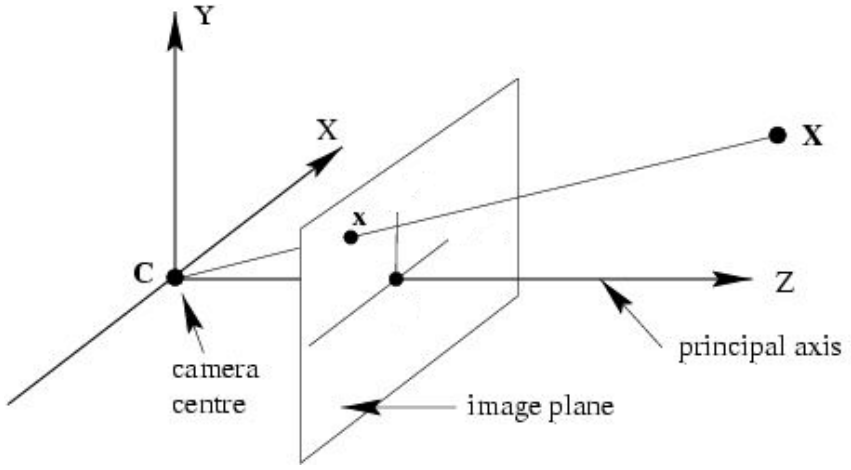


Figure 2.2: Projection of a point accordingly to the pinhole model. Image adapted from Hartley and Zisserman [24].

It follows that the point of intersection $\mathbf{x}$ is $(fx/z, fy/z, f)$, which translates into $(fx/z, fy/z)$ inside the image plane. In terms of homogeneous coordinates (e.g., considering the equivalence class (kx, ky, k) as referring to the same point (x, y)), a point $X = (x, y, z, 1)$ in space is mapped as follows

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} \equiv \begin{bmatrix} fx/z \\ fy/z \end{bmatrix}, \tag{2.1}$$

such that

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

is known as the *camera projection matrix*. Thus, P is a convenient way to represent the camera, and it is sometimes called *the camera* interchangeably.

A more general form of P for the pinhole model includes *offsets* in the x and y coordinates of the image plane, as this is required in many practical situations (e.g., the

camera and the image have different coordinate systems). This translates into

$$P = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.3}$$

so that point (x, y, z) is mapped to $(fx/z + c_x, fy/z + c_y)$.

The matrix P is usually factored into

$$P = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix} = K \begin{bmatrix} I_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix}, \tag{2.4}$$

so that K is intrinsic to the camera (e.g., does not depend on external factors) and is known as the *camera calibration matrix*. The right multiplier represents the pose transformation of the camera concerning the world coordinate frame. Because we assumed the camera coordinate system to be the world (inertial) coordinate system, the transformation is the identity (e.g., no translation and no rotation). That does not need always to be the case, and the transformation of the coordinate system can be any composition of translation followed by a rotation, or *rigid body* transformation. Section 2.2 will give more details on that.

In practice, the individual light receptors of each pixel in a camera may not be a perfect square. In such a case, the sensor does not preserve the proportions between the horizontal and vertical dimensions. This is the case of CCD cameras (Hartley and Zisserman [24]), for which the focal distance is replaced by two new values, α_x, and α_y. These values incorporate both the focal distance and the stretching, s_x and s_y, in the X and Y axis so that

$$\begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix} = f \begin{bmatrix} s_x \\ s_y \end{bmatrix} \tag{2.5}$$

and

$$P = \begin{bmatrix} \alpha_x & 0 & c_x \\ 0 & \alpha_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix} = K \begin{bmatrix} I_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix}. \tag{2.6}$$

2.2 Camera Pose and its Representations

When working with more than one coordinate system, it is common to establish a *world frame*, which is the coordinate system from where all the others will be expressed (see Figure 2.3). Therefore, a coordinate system (X', Y', Z') is characterized by a translation followed by a rotation from the world frame (X, Y, Z). Such transformation is known as the *pose* of the frame. Both rotation and translation in 3D Euclidean space have 3 degrees of freedom each, totaling 6 degrees of freedom for the frame pose. In practice, poses can be used to pinpoint the location and orientation of a moving camera in the world.

Then, if X is a point with (x, y, z) coordinates in the world frame, and the camera

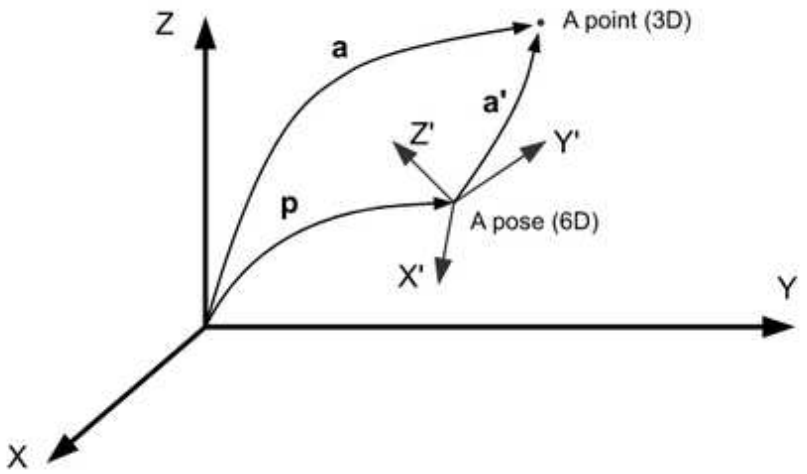


Figure 2.3: Illustration of a pose **p** with 6 degrees of freedom in a reference coordinate frame (X, Y, Z), a 3D point **a** and its relative coordinates **a'** in **p**. Image obtained from Blanco [3].

pose in the world frame is equivalent to the affine transformation $[R|\mathbf{t}]$, the coordinates of the same point in the camera frame will be $R^{-1}X - R\mathbf{t}$. In homogeneous coordinates, it is

$$X_{cam} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} X. \tag{2.7}$$

Pose estimation between pairs of consecutive camera frames is at VO's heart, as it is generally used as initialization for further refinement with windowed bundle adjustment or pose-graph optimization, which will be detailed in Sections 2.5 and 2.6, respectively. Nothing restricts the use of pure pairwise pose estimation to compute the global estimates of poses, as the global poses (e.g., in the world frame) consist of the "concatenation" of the relatively smaller increments. Then, if

$$T_{i+1,i} = \begin{bmatrix} R_{i+1,i} & \mathbf{t}_{i+1,i} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.8}$$

is the relative pose displacement of frame $i+1$ w.r.t frame i, the global pose or the pose concerning the first frame of the entire sequence is

$$T_{i+1} = T_i T_{i+1,i}. \tag{2.9}$$

In the above representation, T is a 4x4 *homogeneous transformation matrix*, while $R \in SO(3)$ (see Section 2.2.3) is its rotation component, and $\mathbf{t} \in \mathbb{R}^3$ is its translation component. It has 12 parameters, although general motion only has 6 degrees of freedom.

Other forms of representation include:

- Euler angles plus translation $[\theta_x, \theta_y, \theta_z, tx, ty, tz]$. It has the minimum number of parameters possible but suffers from *gimbal lock*.

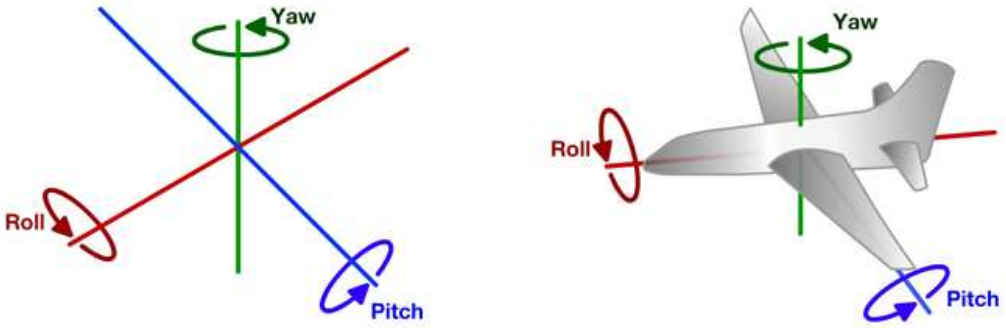


Figure 2.4: Illustration of yaw, pitch, and roll angles.

- Quaternions plus translation $[qw, qx, qy, qz, tx, ty, tz]$, with one parameter more than the minimum necessary but, is free of singularities.
- $\mathfrak{se}(3)$ representation (Section 2.2.3). It results from the invertible *logarithmic map* from the homogeneous form in SE(3) to the $\mathfrak{se}(3)$ algebra form. It only contains the minimum of 6 parameters (3 for rotation and 3 for translation) and is free of singularities. This is the form of the pose parameters in the algorithm introduced in Chapter 4.

2.2.1 Euler Angles and Translation

One of the most used pose representations is the translation plus the *Euler angles* yaw θ_z, pitch θ_y, and roll θ_x (see Figure 2.4). The translation values are the same as in the homogeneous transformation. The meaning of the three angles are three consecutive rotations along each axis individually: first, a rotation of θ_z around the Z axis, followed by a rotation of θ_y around the *new* modified Y axis, and then a rotation of θ_x around the *new* modified X axis. As long as the pitch angle is not $\pm 90°$, there is always a correspondence free of ambiguity between each 3D rotation and a triplet of yaw, pitch, and roll angles. The special case when the pitch angle is $\pm 90°$ is known as *gimbal lock*, and it results in the ambiguity between the roll and yaw angles.

2.2.2 Quaternions and Translation

Another common pose representation is the translation plus unit-length quaternion. In this representation, the translation part is the same as in the homogeneous transformation. The values (q_w, q_x, q_y, q_z) in the quaternion are related to the 3D rotation by an angle of θ around the vector

$$\frac{1}{\sin\frac{\theta}{2}} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}, \tag{2.10}$$

such that $q_w = \cos\frac{\theta}{2}$ and $\left\|(q_w, q_x, q_y, q_z)\right\| = 1$. Even though there are 4 parameters for rotation in this representation, it has just 3 degrees of freedom because of the unit-length constraint. Different from the Euler angles plus translation representation, this one does not have degenerate cases.

2.2.3 SE(3) Lie Group and its Associated Algebra

As mentioned previously in this Section 2.2, the homogeneous form of poses or rigid body transformations is

$$T = \begin{bmatrix} R_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \tag{2.11}$$

where $\mathbf{t} \in \mathbb{R}^3$ is its translation component and $R \in SO(3)$ is its rotation matrix. The *Special Orthogonal group*, or $SO(3)$ for short, is the Lie group formed by 3×3 orthogonal matrices in $\mathbb{R}^{3\times3}$ with determinant equal to 1 and its product being the usual matrix product. This means that the product of any two of such matrices also has those properties (e.g., the composition of two rotations is also a rotation, which preserves proportions and scale).

The 4×4 homogeneous transformation itself is also a Lie group, known as the *Special Euclidean group* or $SE(3)$ for short and the usual matrix product as its operation. The fact that $SE(3)$ is a Lie group is useful because of the existence of the associated Lie algebra $\mathfrak{se}(3)$, which has the minimal representation of 6 parameters, is free of degenerate cases, and has been preferred for gradient-based optimization in the last years (Blanco [3]).

As is going to be detailed next, the transformation of elements from $SE(3)$ to $\mathfrak{se}(3)$ is invertible and differentiable, which guarantees that optimizing in $\mathfrak{se}(3)$ using gradients and interpreting the result in $SE(3)$ is possible without ambiguity. This is what happens in the algorithm introduced in Chapter 4.

The $\mathfrak{so}(3)$ Lie Algebra

The elements of $\mathfrak{so}(3)$ are the linear combinations of the derivatives of the rotations along the three axes at infinitesimally small angles. Those derivatives form the *basis* of $\mathfrak{so}(3)$ and are

$$B_0 := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{2.12}$$

$$B_1 := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \tag{2.13}$$

and

$$B_2 := \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.14}$$

Therefore, the three degrees of freedom correspond to the three real scalars of the combination, so that if

$$\hat{\omega} = \alpha B_0 + \beta B_1 + \gamma B_2 \tag{2.15}$$

then $\hat{\omega}$ is uniquely defined by $\omega = (\alpha, \beta, \gamma)$.

The $\mathfrak{se}(3)$ Lie Algebra

Similarly to $\mathfrak{so}(3)$, the elements of $\mathfrak{se}(3)$ are generated by a basis that is formed by the derivatives of the individual displacements (rotations and translations) at values approaching 0. For the rotation part, it is isomorphic to the $\mathfrak{so}(3)$ basis $\{B_0, B_1, B_2\}$, so that only the translation generators remain to be defined:

$$B_3 := \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.16}$$

$$B_4 := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.17}$$

and

$$B_5 := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.18}$$

So, if

$$\hat{\zeta} = \alpha B_0 + \beta B_1 + \gamma B_2 + x B_3 + y B_4 + z B_5 \tag{2.19}$$

then $\hat{\zeta}$ is uniquely defined by $\zeta = (\alpha, \beta, \gamma, x, y, z)$ and therefore has six degrees of freedom.

Logarithmic and Exponential Maps

The $SE(3)$ group and its $\mathfrak{se}(3)$ algebra are connected by the logarithmic and exponential maps. The logarithmic map takes one element of the group and returns one element of the algebra:

$$\mathbf{log} : SE(3) \longrightarrow \mathfrak{se}(3), \tag{2.20}$$

while the exponential map

$$\mathbf{exp} : \mathfrak{se}(3) \longrightarrow SE(3) \tag{2.21}$$

is its inverse:

$$\mathbf{exp} \circ \mathbf{log} = I_{SE(3)} \tag{2.22}$$

and vice-versa:

$$\mathbf{log} \circ \mathbf{exp} = I_{\mathfrak{se}(3)}. \tag{2.23}$$

If $\zeta = (\alpha, \beta, \gamma, x, y, z) = (\omega, \mathbf{t})$ represents $\hat{\zeta} \in \mathfrak{se}(3)$, then

$$\mathbf{exp}(\zeta) := \begin{bmatrix} e^{\hat{\omega}} & U\mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{2.24}$$

where $e^{\hat{\omega}}$ is the matrix exponential and

$$U = I_{3\times 3} + \frac{1-\cos\phi}{\phi^2}\hat{\omega} + \frac{\phi - \sin\phi}{\phi^3}\hat{\omega}^\top\hat{\omega} \tag{2.25}$$

and $\phi = \left\|\omega\right\|_2$.

On the other hand, if $T \in SE(3)$ has rotation component R and translation $\mathbf{t}$, then

$$\mathbf{log}(T) := (\omega, U^{-1}\mathbf{t}), \tag{2.26}$$

where ω is the 3-vector form of $\log(R) = \dfrac{\phi}{2\sin\phi}(R - R^\top)$ (from the Rodrigues' formula) with $\phi = \left\|\omega\right\|_2$.

Blanco (Blanco [3]) gives details about gradient-based optimization on $\mathfrak{se}(3)$.

2.3 Formulation of Monocular Visual Odometry

VSLAM and VO have a probabilistic formulation in which parameters X are optimized in order to maximize the probability $P(Y|X)$, where Y is either the raw observations (e.g., pixel intensities) or the observations in terms of intermediary features (e.g., the position of key points). When Y is the set of raw pixel intensities, the method is **direct**, while it is **indirect** (feature-based) when Y is a set of features that represent the frames. Figure 2.5 illustrates some basic SLAM concepts such as landmark (feature) mapping, robot localization, and drift.

In an indirect model, the positions of the points which it keeps are, at least approximately, the projection of the associated 3D point, that is $p_i^{true} = \Pi_i(X)$, where Π_i is the projection of the point indexed by i. For direct models the assumption is that the pixel intensities are the intensities at the corresponding projected points, that is $I_{ref}(p_i) = I^{true}(\Pi_i(X))$, where I_{ref} and I^{true} are the reference and true images, respectively. An even more basic assumption that holds for almost all implementations is that the chosen points are points over rigid bodies, points over bodies that cannot be distorted.

In direct methods, uncertainty about a pixel's intensity (photometric noise) is usually modeled as noise that obey a Gaussian distribution ($I^{obs}(p_i) = I^{true}(p_i) + n_{photo}$, where

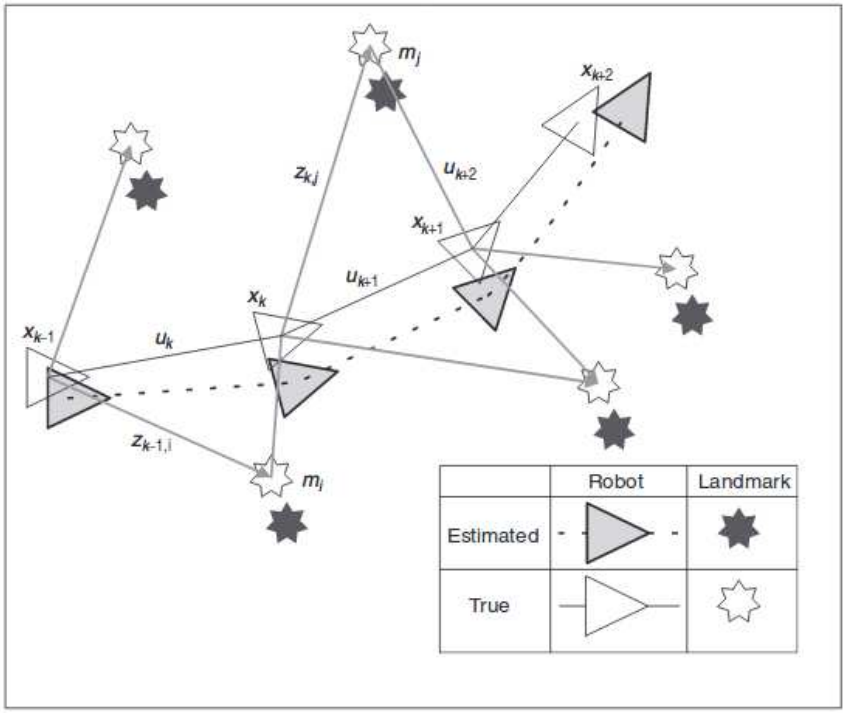


Figure 2.5: A feature-based SLAM and the divergence between real and predicted location, and divergence in keypoint mapping. Figure obtained from Jha and Raman [26].

$n_{photo} \sim \mathcal{N}(0, \sigma^2)$), while indirect methods model the position of key points with uncertainty as Gaussian noise ($p_i^{obs} = p_i^{true} + n_{geom}$, where n_{geom} has distribution $\mathcal{N}(0_2, \sigma_{2\times2}^2)$) for both pixel coordinates (geometric noise). Generally, the parameters X are compound by the camera parameters (calibration values). In formulations with other sensors, it can have inertial IMU (Inertial Measurement Unit) parameters, filtering parameters (Bain and Crisan [2]) (when applicable), and the geometry of pixels or key points, depending on the method.

Optimization of the parameters in these algorithms' formulation requires minimizing a negative log-likelihood or energy objective function. The energy function is the sum of the squares of residuals defined as the difference between a measurement and a prediction. For direct methods, this difference is

$$r_i(X) := I^{obs}(\Pi_i(X)) - I_{ref}(p_i) \tag{2.27}$$

and the corresponding energy function is

$$E(X) := \sum_{p_i \in I_{ref}} r_i(X)^2 \tag{2.28}$$

For indirect methods, the residuals are

$$\mathbf{r}_i(X) := \Pi_i(X) - p_i^{obs} \tag{2.29}$$

and the energy

$$E(X) := \sum_{p_i \in I_{ref}} \left\| \mathbf{r}_i(X) \right\|_2^2 . \tag{2.30}$$

2.4 Epipolar Geometry and Visual Odometry

The two views' geometry is related to the projection of 3D points into different cameras and the relative positions and orientations of the camera views. Such relationships allow the derivation of epipole properties between any pair of frames given the epipoles of adjacent frames. In turn, that can define an optimization problem for Visual Odometry and 3D reconstruction with a low amount of parameters.

2.4.1 Epipolar Geometry

The geometry involving two points of view, each one consisting of a projective camera (e.g., the pinhole camera) and the correspondence between the projections of a scene point X in the two images is known as *epipolar geometry*. Figure 2.6 illustrates the main elements involved: the camera centers behind their respective image planes, the image $\mathbf{x}$ of 3D point X in the left camera and also implicitly its correspondent $\mathbf{x}'$ in the right camera, the *baseline*, which is the line segment connecting the camera centers, the *epipoles*, which are the intersections of the baseline with each image plane, the *epipolar plane*, which is the plane determined by point X and the camera centers, and finally the *epipolar lines*, which are the lines of intersection between the epipolar plane and the image planes.

Notice that, because all epipolar planes contain the baseline, all the epipolar lines intersect at the corresponding epipole. Also, given that every point $\mathbf{x}$ in one image is the projection of some 3D point in the line connecting the camera center and the point itself, there is a one-to-one correspondence between a point $\mathbf{x}$ in one image to an epipolar line in the other image. This is one of the most essential properties in epipolar geometry, as it restricts the match of $\mathbf{x}$ in the other image, $\mathbf{x}'$, to be over that epipolar line. The mapping from point to epipolar line is a linear map F, and the epipolar constraint just mentioned can be expressed as

$$\mathbf{x}'^{\top}(F\mathbf{x}) = 0. \tag{2.31}$$

The term $(F\mathbf{x})$ is the epipolar line of $\mathbf{x}$ in the other image. The product being zero indicates that $\mathbf{x}'$ should lie on that line. F is known as the *fundamental matrix*. The *essential matrix* E is analogous to F, but for $\mathbf{x}$ and $\mathbf{x}'$ in pixel coordinates. That means

$$E = K'^{\top} F K, \tag{2.32}$$

where K and K' are the calibration matrices of the first and second cameras, respectively, when working with monocular visual odometry, $K = K'$ because the two cameras correspond to the same physical camera in different instants in time. Working with E or F is a matter of which coordinates one prefers the points to be on, but one can be recovered from the other if the calibration matrices are known.

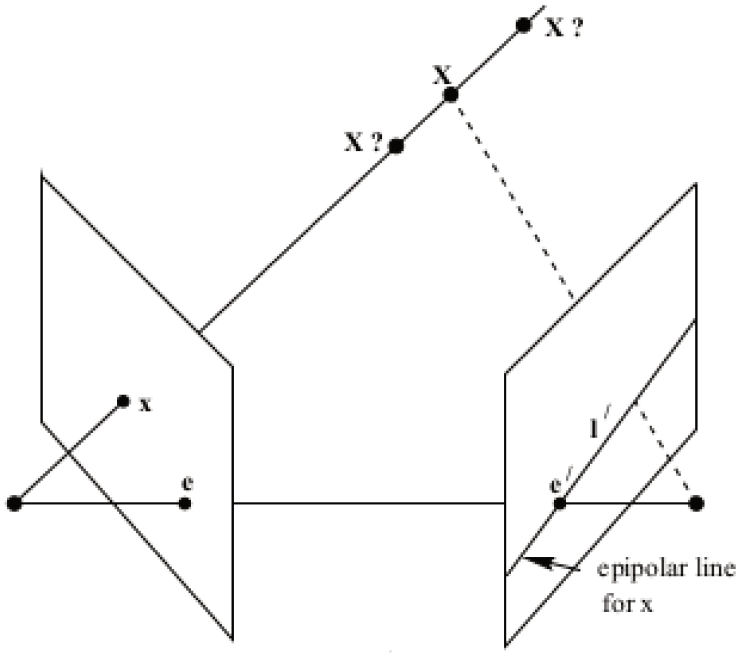


Figure 2.6: Illustration of the elements that define the geometry of two projective views. Adapted from Hartley and Zisserman [24].

On the other hand, if the matches $(\mathbf{x}, \mathbf{x}')$ and the relative pose between the cameras are known, point X can be recovered given that the lines of reverse projection of the points must intersect at that point in space. *Triangulation* algorithms do precisely that and can be used for 3D scene reconstruction.

2.4.2 The Essential Matrix and Motion

When working with a camera model like the pinhole camera, it is possible to find relative poses from the *essential matrix* E without treating the pose as parameters. E arises from the *epipolar constraint*: for every correspondence $(\mathbf{x}, \mathbf{x}')$ of points in pixel coordinates, it holds that

$$\mathbf{x}'^{\top} E \mathbf{x} = 0. \tag{2.33}$$

The geometric meaning is that $\mathbf{x}'$ must lie in its counterpart's epipolar line in the first image, $E\mathbf{x}$. Also, it is true that

$$E = [\mathbf{t}]_{\times} R, \tag{2.34}$$

which tells why E encodes the relative motion between the two views. The two main ways of finding E, namely Nister's 5-point algorithm (Nistér [41]) and Longuet-Higgins' 8-point algorithm (Longuet-Higgins [33]), consist of solving a linear system of equations inside *Random Sampling and Consensus* (RANSAC) (Fischler and Bolles [16]) iterations. RANSAC is essential for pose estimation, as the presence of outlier correspondences severely affects the resulting poses, and the concatenation of those make the errors even more salient.

Notice that the motion is not the set of parameters itself, but the values of E notice that, from the optimization point of view. Once E is estimated, then R and t are extracted from it. To do so, the *Singular Value Decomposition* (SVD) is applied so that if

$$USV^\top = E \tag{2.35}$$

and

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.36}$$

then there are four candidate poses:

$$R = U(\pm W^\top)V^\top \tag{2.37}$$

$$\mathbf{t} = U(\pm W)(S/s)U^\top \tag{2.38}$$

where $\mathbf{t}$ is determined up to scale and

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.39}$$

There is only one correct pose from the candidates, and it is the one that satisfies the positive depth constraint of a point triangulated from its locations in the two images (Hartley and Zisserman [24]).

When using RANSAC, the number of minimal sets necessary to achieve a certain probability of finding one with only inliers grows exponentially with the minimal set size. Therefore, it is desirable to need as few points as possible. The 5 points needed for the computation of E are the minimum considering the general motion, which has 6 degrees of freedom. However, in constrained motion, like planar or car motion, there are fewer degrees of freedom, and it should be possible to use fewer points. Scaramuzza [49] shows that locally planar car movement can be estimated using only one point correspondence and RANSAC or other outlier removal method known as *histogram voting*. In such case, the rotation and translation (up to scale) are functions of a single parameter, that is, the angle at the *Instantaneous Center of Rotation* and between the positions of the car in two subsequent instants.

2.5 Windowed Bundle Adjustment

Bundle Adjustment consists of estimating camera pose and scene geometry in the general sense, without the requirements of SLAM or VO, by minimizing the *reprojection error* of a large number of feature points. The underlying optimization problem results from some probabilistic assumptions about the distribution of the measurement errors and the measurements' independence. It depends heavily on the correct *data association* between

the feature points and the scene points. The minimization of the reprojection error consists of solving a large linear system. In order for it to be practical, the sparsity of the associated matrix must be explored.

As new frames and key-points are incorporated into the system, the number of parameters increases rapidly. When the optimization involves second-order derivatives, the computation time is quadratic or cubic in the number of parameters, especially if there is too much correlation between pairs of parameters. The strategies to keep the computation time practical include parameter marginalization with Schur complement and key-frame selection. Both strategies are used in Engel et al. [12]. A simpler strategy is to work with limited local maps, as the feature points are the most responsible for the big number of parameters in the system. This is adopted in Aladem and Rawashdeh [1]. In this work, we formulate an "indirect" type of bundle adjustment by replacing the structure parameters with epipoles from two-view geometry to drastically reduce the number of parameters and still have some control over the structure's optimization.

Geometric versus Photometric Reprojection error

In order to find suitable estimates of motion (poses) and structure (point depth), it is common to minimize the *geometric reprojection error*. For a single point correspondence $(\mathbf{x}, \mathbf{x}')$ in pixel coordinates, camera matrix K, pose $T = [R|\mathbf{t}]$, and depth d it is defined as

$$E(R, \mathbf{t}, d)_{(\mathbf{x},\mathbf{x}')} = \left\| K\Pi(T^{-1}d[K^{-1}\mathbf{x}|1]^\top) - \mathbf{x}' \right\|_\gamma , \tag{2.40}$$

where Π is the projection from the scene to the second frame and $\left\|.\right\|_\gamma$ is some norm, usually L2, L1, or Huber. The error involving all reprojected points is the weighted sum over the point matches M

$$E(R, \mathbf{t}, d) = \sum_{m \in M} w_m E(R, \mathbf{t}, d)_m. \tag{2.41}$$

The *photometric error*, used in direct methods, is based on the pixel brightness difference around the estimated reprojection (at the target image) and around the point in the source image. For the photometric error to be differentiable, the image derivatives for pixel positions must be defined, which is not much of a problem because nowadays, even automatic differentiation frameworks implement that.

Reprojection graph and factor graph

One way to characterize the relationship between poses that "see" a lot of same features or "reproject" into one another is by a directed graph, where each node is a pose and edges indicate that features can be reprojected from source to target node. Then we can define a global pose error that involves the pairwise reprojections and might have redundant information that helps to adjust the motion and structure estimates in a broader locality. Even though there can be many reprojections, the number of parameters in the proposed parametrization scales linearly with the number of frames.

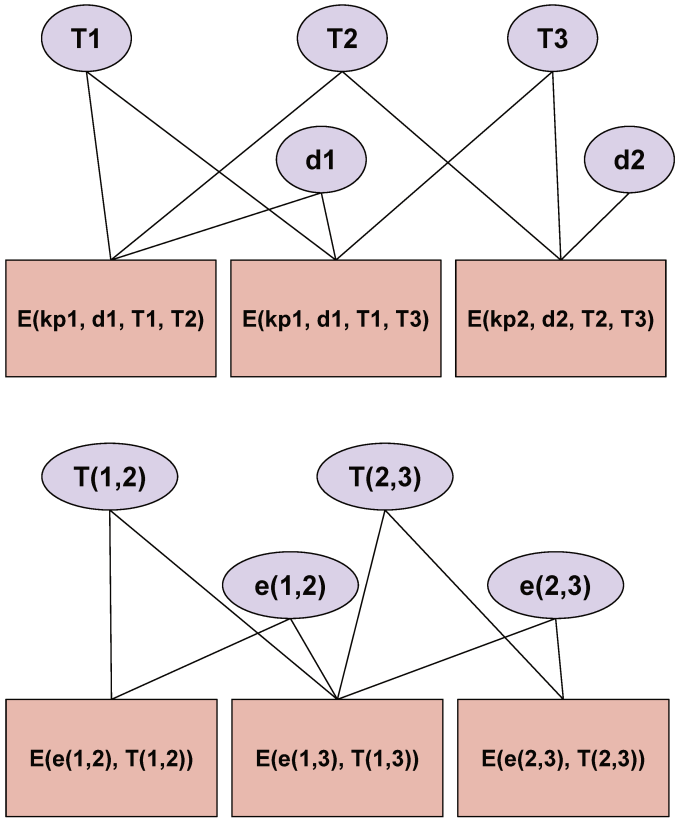


Figure 2.7: Comparison of factor graphs of BA (top) where each energy term depends on a key-point depth and global poses from the start and target frames (e.g., d_1, T_1, and T_2, respectively), and the proposed windowed optimization (bottom). Notice that in BA, a key-point can be identified in more than one pair of frames. In that context, *data association* is the process of identifying a feature in different places, and it is important for the minimization of the energy. On the other hand, in our optimization, the pose parameters are the local displacements between consecutive key-frames, and the epipoles are also between consecutive key-frames. Due to this approach, a reprojection from key-frame 1 to 3 depends on all poses and epipoles on the way. It does not mean that all those parameters need to be optimized at once in every iteration, however.

When it comes to implementing the global error, if the pairwise error is already implemented, it is sufficient to sum or average the constituent local errors if the respective pose and second epipole are properly expressed in the original parameters.

Also, the *factor graph* is a standard tool for the visualization of the dependencies of the energy terms in optimization. Figure 2.7 top is the factor graph of a proper Bundle Adjustment problem, while the bottom is the factor graph in our parametrization.

2.6 Pose-graph Optimization

It is possible to refine the estimates of poses (nodes) in a windowed fashion with "constraints" (directed edges) that are prior estimates of poses in two instants of trajectory. This is known as *pose graph optimization* and does not necessarily use structure parame-

ters. In such a case, the poses are initialized and then refined by optimizing the problem defined in the pose-graph. In this context, **g2o** (Kümmerle et al. [29]) is a powerful tool used often.

One example of a successful pose-graph algorithm is the one by Olson et al. [43], as they introduce a fast algorithm for the refinement of noisy initial poses. It is based on SGD (Kiefer and Wolfowitz [27]) and on a special representation of the state space being optimized. Latif et al. [31] equip pose-graph SLAM with robust place recognition, so that it performs loop-closures. One example of optimization with a pose-graph that can have its constraints altered *during* optimization is the one introduced in Sünderhauf and Protzel [56]. As a result, the loop-closure part of the algorithm becomes more robust.

Pure pose-graph optimization without structure parameters belongs to the "few parameter" side of Visual Odometry, as in the minimal case, there are just 6 parameters per key-frame. Its downside is that it does not make full use of the information from point measurements, and in turn, the quality of the pose estimates might be inferior. Furthermore, if we perform triangulation using point matches and the resulting poses, the pose estimates' inferior quality will lead to poor structure reconstruction. We try to alleviate this problem without adding too many parameters replacing the structure-related parameters from bundle adjustment (e.g., inverse depth of points) by epipoles (see Section 4.1).

2.7 On-manifold Second-Order Optimization

Suppose we want to minimize an *energy* function E defined as

$$E(\mathbf{x}) \coloneqq \sum_{i=1}^{N} r_i(\mathbf{x})^2, \tag{2.42}$$

where r_i is called a *residual* and $\mathbf{x}$ are the values we want to find. If we have a current estimate $\mathbf{x}_0$ for $\mathbf{x}$ that is close enough to the global optimum, we might refine the estimate iteratively with displacements ϵ, so that if E is flat around $\mathbf{x}_0$ then $\mathbf{x}_0 + \epsilon$ is closer to the optimum. Methods like Gauss-Newton and Levenberg-Marquardt do this by using a first-order Taylor approximation of E:

$$\begin{aligned} E(\mathbf{x}_0 + \epsilon) &= \left\|\mathbf{r}(\mathbf{x}_0 + \epsilon)\right\|_2^2 \\ &\approx \left\|\mathbf{r}(\mathbf{x}_0) + \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\mathbf{x_0})\right\|_2^2 \\ &= \mathbf{r}_0^\top \mathbf{r}_0 + 2\epsilon^\top \mathbf{J}_0^\top \mathbf{r}_0 + \epsilon^\top \mathbf{J}_0^\top \mathbf{J}_0 \epsilon \end{aligned} , \tag{2.43}$$

where

$$\begin{aligned} \mathbf{J}_0 &= \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\mathbf{x_0}) \\ \mathbf{r}_0 &= \mathbf{r}(\mathbf{x}_0) \\ \mathbf{r} &\coloneqq \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix}. \end{aligned} \tag{2.44}$$

Then if we take $H \coloneqq \mathbf{J}_0^\top \mathbf{J}_0$ and $\mathbf{b} \coloneqq \mathbf{J}_0^\top \mathbf{r}_0$, it follows that

$$\epsilon = -H^{-1}\mathbf{b}, \tag{2.45}$$

and the new value of $\mathbf{x}_0$ is

$$\mathbf{x}_0 \leftarrow \mathbf{x}_0 + \epsilon. \tag{2.46}$$

The difference of Levenberg-Marquardt from Gauss-Newton lies on the matrix H, which is also known as approximate Hessian (under the assumption of normally distributed errors):

$$H_{LM} \coloneqq (H + \lambda diag(H)), \tag{2.47}$$

as this has a regularization effect that makes the optimization behave as gradient descent for high λ (good if the current estimate is too far from the optimum) or Gauss-Newton for small λ (if the current estimate is close to the optimum). λ should always be non-negative.

2.8 Extraction and Correspondence of Key-points

Visual Odometry methods that depend on a feature-matching step usually rely on three things: the extraction of points of interest, the matching of those points between adjacent frames, and the computation of the relative pose between the adjacent frames. For the first, feature descriptors and corner detectors are useful in many applications because they take the high-dimensional input images and return a reduced set of points that are informative enough for the task at hand. Those points can then be matched in a brute-force way (e.g., checking among all the point pairs that give the better matches), but it can also involve some optimization, as in the case of the Lucas-Kanade optical flow. Finally, the resulting set of point correspondences can be used to estimate movement parameters (e.g., find the Essential Matrix with RANSAC and then recover the relative pose from it).

2.8.1 FAST Corner Detector

The main requisites for a feature detector in a Visual Odometry algorithm are robustness to noise and the correctness of the points' position. Also, the point detection step's speed

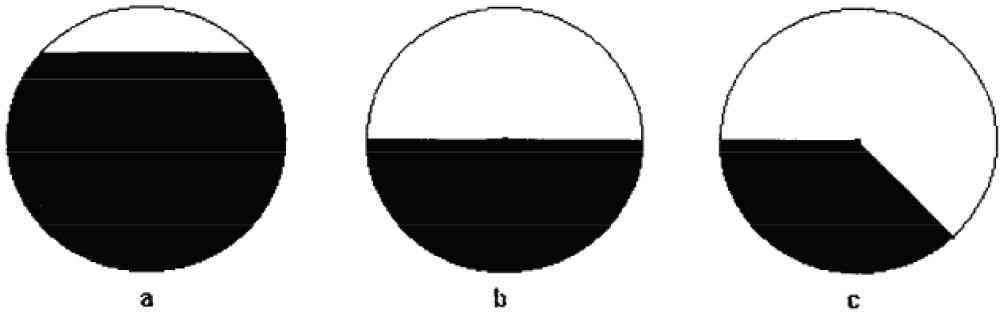


Figure 2.8: The three cases for corner detection considered by the FAST detector. In (a) the center of the window is inside a uniform region with many lines with uniform intensity going through the center. In (b), there is one direction that passes through the center and is uniform. There is no such line in (c), and thus, the point at the center is considered a corner. The CRF should be able to distinguish (c) from (a) and from (b). Image adapted from Trajkovic and Hedley [57].

is fundamental if there is a real-time requirement (Trajkovic and Hedley [57]). The last is FAST corner detection's main advantage, as there are options nowadays that include other types of a feature than corners and are invariant to some types of transformation like scale and rotation. Some examples of well-known robust descriptors are SIFT (Lowe [35]), BRIEF (Calonder et al. [7]), and ORB (Rublee et al. [48]).

Moravec [38] defines a corner in an image as a point with a high difference in intensity in all directions (referent to the horizontal, the vertical, and the diagonals in between). After corners are extracted from an image, the points can be used in a further matching step that will ideally associate every corner in one image to its occurrence in another. In this case, it is good to avoid corner patterns in textured regions, as there are many locations with the same vicinity appearance (e.g., a wall of rectangular bricks).

The FAST algorithm's main components are its Corner Response Function (CRF) and an efficient multigrid search that suppresses corners with a significant potential of mismatch in textured regions. FAST is a modified version of the Harris detector (Harris and Stephens [22]). The difference is that FAST only checks points with an image gradient higher than some threshold. This reduces the total time significantly, as the algorithm only applies the CRF in a much smaller set of points.

The CRF from FAST can distinguish corners from edges and uniform regions (see Figure 2.8) by attributing high values to corners and low values to the other cases. For a particular direction consisting of two points at the window edge A and B passing through the center C, it is defined as

$$CRF(A, B, C) := (I_A - I_C)^2 + (I_B - I_C)^2, \tag{2.48}$$

where I is the grid of image intensities after the original image is convolved with a Gaussian smoothing filter, it accepts a point as a corner if the CRF minimum along a set of directions is greater than some threshold.

The multigrid strategy consists of taking the CRF of points in two steps: first in small resolution and then in a higher resolution so that texture corners, which are not invariant

to scale transformation, are removed. True physical corners of objects in a scene, on the other hand, are more robust to scale transformation and give correct matches in two-view vision most of the time.

2.8.2 Lucas-Kanade Optical Flow

Optical flow is the apparent 2D displacement of objects in a sequence of images. It has applications in Structure from Motion and is based on two assumptions: the pixel intensities in the image do not change with the time approaching zero, and close neighboring pixels tend to have the same displacement or *disparity*. If $I(x_0, y_0, t_0)$ is the intensity of pixel (x_0, y_0) at time t_0, then the first assumption is expressed by

$$I(x_0, y_0, t_0) = I(x_0 + dx, y_0 + dy, t_0 + dt). \tag{2.49}$$

By taking the linear approximation around (x_0, y_0, t_0) it holds that

$$\begin{aligned} I(x_0, y_0, t_0) &\approx I(x_0, y_0, t_0) + \frac{dI}{dt}(t_0)dt \\ &= I(x_0, y_0, t_0) + \left(\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\frac{dt}{dt}\right)(t_0)dt \end{aligned} \tag{2.50}$$

so that

$$\left(I_x u + I_y v + I_t\right)(t_0)dt \approx 0, \tag{2.51}$$

where $u = \dfrac{dx}{dt}$ and $v = \dfrac{dy}{dt}$ are the components of the displacement. The term dt and t_0 can be omitted if the computation involves just two discrete image frames. In this case the *optical flow equation* is

$$I_x u + I_y v + I_t = 0. \tag{2.52}$$

The Lucas-Kanade optical flow (Lucas and Kanade [36]) solves Equation 2.52 for a small window (e.g. 3×3) around a given point. It finds (u, v) by solving an equivalent linear system formed by several equations equal to the number of points in the window, thus being over-determined whenever the window has more than 2 points. In its original formulation, it uses iterative weighted Newton-Raphson. Also, it applies the multi-scale strategy first with a small resolution and then with the original resolution. This helps the algorithm be more robust to large displacements because in the small resolution, the bigger displacements "fit" inside the window while the original resolution naturally works for the smaller ones.

2.8.3 Random Sampling and Consensus

Random Sampling and Consensus (RANSAC) (Fischler and Bolles [16]) is a model fitting paradigm robust to outliers. Unlike non-robust methods, it does not assume that errors in data points are just noise that vanishes by the use of many data points. On the contrary, it is based on model fitting with the minimum number of data points required. Algorithm

Algorithm 1: Given a set of data points P, a minimum number of points M for fitting a model, a minimum number of points K for the final consensus set, an error tolerance ϵ for a point to be considered part of a consensus, and the probability p of an arbitrary point to be an outlier, for every $q \in (0, 1)$ there is an integer $N > 0$ such that N or more iterations inside RANSAC will give a consensus set with at least M points (none of those outliers) with probability q.

Result: If found, the fitted model parameters Θ and its consensus set C. Otherwise $\{\}, \{\}$.

$i \longleftarrow 0$
while $i < N$ **do**
  $S \longleftarrow sample\_points(P, M)$
  $\Theta \longleftarrow fit\_parameters(S)$
  $C \longleftarrow find\_consensus\_set(P, \epsilon, \Theta)$
  if $K \leq |C|$ **then**
    $\Theta \longleftarrow fit\_parameters(C)$
    return Θ, C
  end
  $i \longleftarrow i + 1$
end
return {}, {}

1 is a high-level form of RANSAC. The *sample_ points* subroutine samples a set S of M points from the input data points P. The subset S is known as a *minimal set* for fitting the model. Then *find_ consensus_ set* creates a set C consisting of all points in P that have error less than ϵ under the current model Θ returned by *fit_ paramters(S)*. If the number of elements in the current consensus set C is greater than the minimum required K to have a satisfactory consensus, the algorithm stops and returns Θ and C. If it did not find a suitable consensus after all the iterations, the algorithm returns nothing.

The relationship between the rate of outliers p in the data and the number of iterations needed to find a satisfactory set with probability q is

$$N = \frac{\log(1 - q)}{\log(1 - (1 - p)^M)}. \tag{2.53}$$

For instance, if q is fixed at 99% and p is 40%, a minimal set of $M = 4$ points requires around 34 iterations, while a minimal set of $M = 6$ points requires 97 iterations, and 272 iterations for $M = 8$. This growth is exponential. Also, depending on the outliers' rate, RANSAC may take a very low number of iterations, although it might not find a suitable set if that rate is much higher than half the points. In Structure-from-Motion, RANSAC is widely applied in estimating the Essential Matrix, usually with 5 points for the minimal set.

The presence of just a few outliers can divert the estimated model from the ideal one if the fitting strategy is not robust (see Figure 2.9). In linear regression with one input variable, there are only 2 parameters, and the minimum number of points required to fit the line is 2.

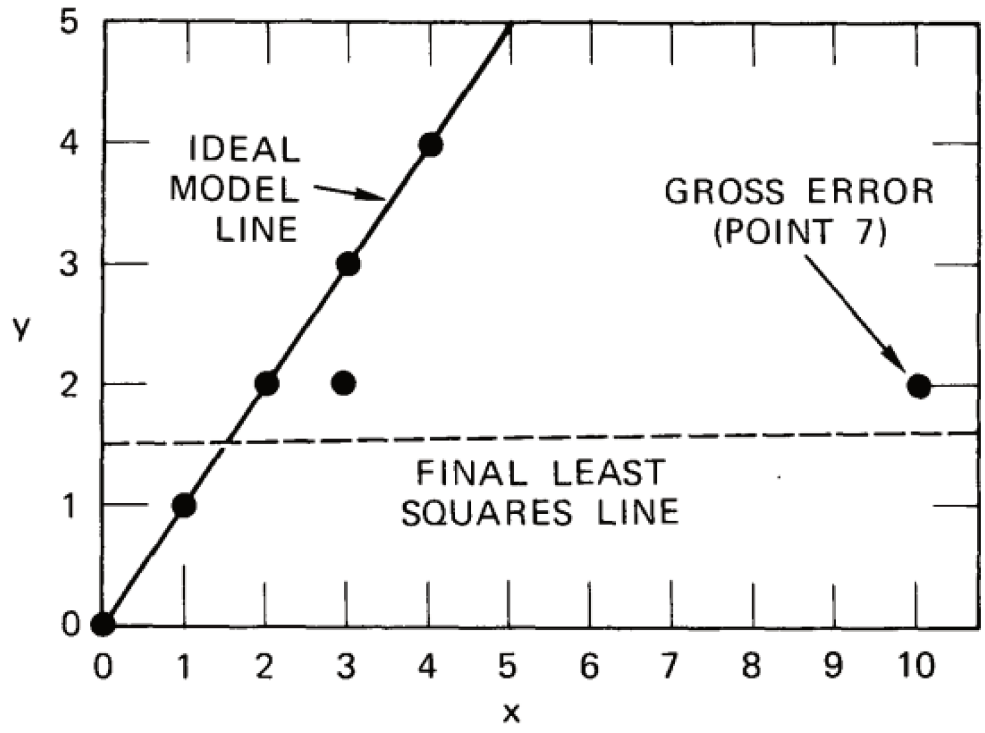


Figure 2.9: Illustration of a linear regression model fitted in the presence of one outlier in comparison with the ideal model. Image adapted from Fischler and Bolles [16].

Chapter 3

Related Work

This chapter reviews part of the Visual SLAM/Visual Odometry literature. It includes classic works, application/niche works, and applications of data-based learning to Visual Odometry. Works related to VO that have a low amount of parameters usually do not use windowed optimization, like Scaramuzza [49], or if they do, it is just pose-graph optimization without geometry parameters, as Kiefer and Wolfowitz [27], Latif et al. [31], Olson et al. [43], Sünderhauf and Protzel [56].

3.1 Classic Works on SLAM/VO

Among the works on monocular SLAM, Davison et al. [9] came up with an efficient real-time algorithm, which runs on a conventional CPU with a frame rate of 30 Hz. Their work is one of the first in the "pure" vision domain (they also use information about the vehicle's acceleration and angular acceleration) to achieve drift-free results. The approach defines a sparse set of features consisting of small patches, such as 11x11 rectangles from the images, such that the search for the optimal locations of these features is controlled by Gaussian ellipsoids generated around the expected feature location, which significantly limits that search space. Besides the probabilistic approach to mapping the image features, they also introduce feature initialization and feature orientation estimation solutions. Despite being called "MonoSlam," their system focuses on the 3D trajectory recovery more than on geometry reconstruction and can thus be classified as an indirect visual odometry algorithm.

ORB-SLAM (Mur-Artal et al. [40]) combines the efficient and robust ORB features (Rublee et al. [48]), which are invariant concerning rotation, scale, and also are resistant to noise, with a new initialization procedure that suggests initial maps for the planar and non-planar scene possibilities. Whereas the planar case aims at finding a proper *homography matrix*, the non-planar case searches for a good *fundamental matrix*. A selection is then performed to select the best of the two. Furthermore, a *survival of the fittest* approach guides the selection of map points and keyframes, helping to discard redundant frames and keep only a set of frames and points diverse enough for all the algorithm tasks, namely tracking, mapping, re-localization, and loop closing. The reuse of the same features for the algorithms tasks helps to make it efficient and straightforward.

ORB-SLAM is tested on three known benchmarks: NewCollege (Smith et al. [53]), TUM RGB-D (Sturm et al. [55]), and KITTI (Geiger et al. [20]). The first is a large robot outdoors sequence of 2.2km. TUM RGB-D is a set of 16 hand-held indoors sequences useful for evaluation of general indoors performance. KITTI is a 11 car outdoors sequence and is very challenging for monocular SLAM/VO because of fast car speed and rotations. Contrary to previous work tested on the NewCollege dataset, ORB-SLAM can handle the 20fps sequence while closing global loops. The absolute trajectory errors (ATE) on the TUM RGB-D dataset on ground-truth are compared to those of LSD-SLAM and PTAM (Klein and Murray [28]). While ORB-SLAM can process all sequences entirely, except one, PTAM and LSD-SLAM fail on much more. ORB-SLAM and PTAM are equally accurate on the open sequences and better than LSD-SLAM, while ORB-SLAM is superior in detecting large loops. ORB-SLAM also shows much better trajectory accuracy when compared to the state-of-the-art at the time on the KITTI dataset.

Direct Sparse Odometry (DSO) (Engel et al. [12]) is a monocular direct method, where a kind of photometric error should be minimized concerning the parameters, which include camera poses, depth maps, and the intrinsic values of the camera. To understand how this error is computed, we can look at its most fundamental part: the photometric error around a pixel. First, we take two frames: a reference frame and the current frame. Then we choose one of the pixels in the selected sparse subset of pixels in the reference image. By obtaining the pixels' reprojection around the selected pixel into the current image (given the parameters' values at the moment), we compute the sum of the differences of the pixels' intensities, taking into account the exposure times and the brightness transfer parameters. The amount gives the total error over all frames, overall points in the frames, and those from which the reference frame's point is visible. Unlike the usual direct methods, there is no smoothness prior, but pixels from regions with gradient are sampled uniformly from the images instead.

DSO is evaluated in the following datasets: TUM monoVO (Engel et al. [13]), EuRoC MAV (Burri et al. [5]), and ICL NUIM (Handa et al. [21]). The first has 50 photometrically calibrated sequences from outdoors and indoors environments, but it only has ground-truth data for loop-closure, making the authors use alignment error instead of absolute trajectory error. The second dataset has 11 stereo-inertial sequences from indoor environments but lacks photometric calibration. ICL NUIM is a set of ray-traced sequences from indoor environments and does not require calibration. The error metric for both EuRoC and ICL NUIM is absolute trajectory error. Compared to ORB-SLAM, the method outperforms it in a big part of the sequences in all datasets.

Large-Scale Direct Monocular SLAM (LSD SLAM) (Engel et al. [10]) is a stereo direct method that provides accurate maps and poses estimates in big environments, like street blocks, while the previous direct techniques were limited to smaller scales. On the algorithm"s full-scale, the scene's map consists of a graph with the pose of the keyframes and the frames' depth maps as vertices and the photometric constraint between consecutive keyframes as edges. Therefore, the optimization over the global map is a graph optimization problem, for which a highly recommended global optimizer like **g2o** (Kümmerle et al. [29]) could be used. However, the authors preferred to use their implementation of a Gauss-Newton based optimization for the underlying least-squares problem. On a

small scale between frames, a new image is to be considered the new reference frame if the camera got too far from the previous keyframe, in which case a frame alignment is performed for the new frame, resulting in its pose and depth map. Otherwise, the image is used to refine the reference frame's current depth-map through filtering, a probabilistic model for updating a system's states by merging new observations with the current estimate. When it replaces a keyframe, it adds to the global map through optimization.

3.2 Applications of Learning-Based Methods

Deep Learning can be used for both frame preprocessing and end-to-end pose estimation in visual SLAM/Odometry. In the first case, images provided by a camera need to be undistorted, as in the rolling shutter effect, before being passed as input. This process has the advantage of detachment from the SLAM system, although such preprocessing may turn the whole process too time-consuming and therefore not feasible for real-time applications. For end-to-end pose estimation, the poses are estimated directly from the frames by a deep neural network. Even though the training phase can consume too much time, given that it requires a significant and varied amount of images, it has the potential of modeling a variety of behaviors in different environments that would otherwise be manually crafted by an expert or just ignored. For instance, new state-of-the-art visual SLAM/Odometry systems hardcode effects like lens vignetting, time of exposure, camera intrinsic reprojection geometry, and photometry. There is also the advantage of using a pre-trained network for feature extraction and training a separate model with the resulting representations as input.

A Deep Learning architecture is used in Laina et al. [30] to directly estimate in real-time the depth map of all pixels of a given RGB image. It is a fully convolutional neural network formed of a pre-trained ResNet 50 (He et al. [25]) network followed by a set of new efficient up-scaling layers introduced by the authors. The model is trained with a reverse Huber loss, and its ground-truth data comes from the NYU depth V2 dataset (Silberman et al. [51]). Their architecture, combined with the reverse Huber loss, shows better accuracy than the previous state of art Deep Learning works and uses fewer parameters. One noticeable advantage of their work is that the model is trained end-to-end without processing its output again. Their experiments on two known benchmarks of indoors and outdoors scenes compare their results against a handful of state-of-the-art works for scene reconstruction. They outperform the results of the other works in seemingly all the metrics used before their work.

Rengarajan et al. [47] used a CNN (LeCun et al. [32]) with horizontal and vertical rectangular filters to predict parameters of the camera motion for each row of a given single frame. With the estimated motion values at hand, which are a simplified version of camera motion in their work, the input frame is undistorted. The data used in their work is synthesized by people purposely shaking a camera while taking the pictures and some artificially distorted images. The camera motion model for each row y is limited to rotation around the optical axis and translation on the x axis, but the motion on every row is not independent of each other because they are given by polynomial interpolation

of degree 3. It considerably reduces the number of values to be estimated and may lead to oversimplification of the possible distortions. Using a preprocessing step that relies on a deep neural network may be impractical in some scenarios, as the time and processing power required for generating the estimates of frame correction can be too high. Still, if a centralized server can provide the correction step for many clients in real-time, the overall efficiency could be worth it if the distortion removal is good enough for the motion algorithm's structure receiving the corrected images.

Their quantitative analysis regards PSNR (Peak Signal-to-Noise Ratio) and RMSE for translational and rotational errors between ground truth and estimated displacement. The experiments on a synthetic dataset show less error if considering all possible combinations of shutter effect (rotation only, translation only, and rotation plus translation) among methods that do not use references in the pictures, but still worse in most metric methods that use references. The same is true on a dataset of real urban scenes and human faces.

DeepVO (Wang et al. [58]) uses feature representation from a deep network like VGGNet (Simonyan and Zisserman [52]) as input to a recurrent neural network that estimates poses directly from sequences of such observations. They trained the system from the mean squared error between ground truth poses and estimated poses. Instead, if in a direct formulation with mapping, the objective function could be independent of ground-truth data, as the photometric error from reprojection could be computed from poses and depth values. In their experimental results on the KITTI VO benchmark (Geiger et al. [20]), the authors show that their method outperforms monocular VISO2 (Geiger et al. [19]) regarding the KITTI metrics like the RMSEs of translational and rotational errors, although this is not true when compared to the stereo version of VISO2.

In Parisotto et al. [44], the authors present a complete visual SLAM solution employing end-to-end differentiable Convolutional and Recurrent Neural Network modules. There are three main parts: a local pose estimator that is a convolutional model and acts on pairs of subsequent frames, a pose aggregator composed of temporal convolutional layers, and a neural graph optimizer formed by Recurrent Neural Network (RNN) cells and an attention mechanism. This last module is responsible for accumulating information over an entire trajectory and minimize drift.

Experiments are executed on 2D and 3D game maze environments. Compared with DeepVO, their method is superior regarding translational and rotational pose errors if the global estimation is allowed, but it draws with DeepVO if the only local estimation is permitted.

3.3 Applications of VSLAM/VO for Aerial and Underwater Vehicles

Given that a vehicle can navigate in a wide variety of conditions, like indoors or outdoors, in places with high or low texture, and with high or low speed, there are different requirements for good navigation for each case. For aerial vehicles and robots, the navigation can suffer severely because of sudden motion (drones and robots), even in well-behaved indoors, and therefore it needs special attention.

The indoors navigation strategy presented in Celik et al. [8] runs the monocular SLAM FastSLAM (Montemerlo et al. [37]) on a micro aerial vehicle (MAV) with all the necessary processing hardware onboard and a consumer-level USB camera. Their SLAM is based on corner and line features and explores the lines' orthogonality in a closed environment to estimate distance. The experiments show that the systems suffer from position drift in the order of 2m in a loop with more than 100m, but has good feature tracking and is capable of closing loops.

One work for underwater applications with a low-cost camera is Ferrera et al. [15]. This work uses optical flow features in a keyframe setting and graph optimization for the resulting local tracking objective. They show how optical flow is considerably better than descriptors for this type of environment and test their algorithm on simulated and real underwater datasets against current state-of-the-art methods, namely ORB-SLAM, LSD-SLAM, and SVO (Forster et al. [17]). These experiments in different water turbidity levels show that their algorithm has less translation drift in more than half of the sequences.

Table 3.1 summarizes related works developed for a range of different applications, from aerial and terrestrial to underwater. It also classifies them by the type of method (direct or indirect, sparse or dense, monocular or stereo), by the kind of sequence in which it was tested, indoor, outdoor, or both it used Deep Learning.

Paper	**BA?**	**PG?**	**FP?**	**Dataset**
Davison et al. [9]	Yes	No	Yes	Humanoid robot
Mur-Artal et al. [40]	Yes	Yes	Yes	KITTI, NC, TUM-RGBD
Engel et al. [12]	Yes	No	Yes	TUM Mono, Euroc, ICL
Engel et al. [10]	No	Yes	No	TUM-RGBD
Engel et al. [11]	No	Yes	No	KITTI, Euroc
Wang et al. [58]	No	No	No	KITTI
Ferrera et al. [15]	Yes	No	Yes	private
Celik et al. [8]	No	No	No	private
Gao et al. [18]	Yes	Yes	Yes	TUM-Mono, Euroc, KITTI
Parisotto et al. [44]	No	No	No	2D maze, ViZ-Doom
Lovegrove et al. [34]	Yes	Yes	Yes	Simulated
Schubert et al. [50]	Yes	No	Yes	ICL, TUM-RGBD

Table 3.1: Visual SLAM/Odometry related works spanning different scenarios. BA stands for Bundle Adjustment, PG stands for Pose-graph, and FP stands for feature parameters.

Chapter 4

The Epipolar VO Algorithm

This chapter presents a high-level view of our algorithm, the Epipolar VO pipeline (Figure 4.2). Epipolar VO is a Monocular VO algorithm based on the connection between two-view geometry, relative poses of frames, and scene points' depth. It incorporates the parametrization introduced in Section 4.1.

4.1 The Relationship Between Epipoles, Poses, and Depth

The following properties involving poses, epipoles, and point depths are the basis for a parametrization of Visual Odometry consisting of poses (6-parameter general motion) and second-epipoles (3 parameters instead of 2 for derivation, implementation, and interpretation convenience) from pairs of images. This parametrization will be used in the Monocular VO algorithm introduced in Chapter 4. It is also possible to include relative scales (1 parameter per frame), but in this work, the scale is treated separately in Section 4.5. Thus, such parametrization contains only $9n$ parameters, where n is the number of image frames in the sequence. It can be used to retrieve depth maps with the desired density at the time wanted, online or offline. Also, it is compatible with both direct and indirect methods.

In the next sections, epipoles can be refered to as pairs (x, y) in pixel coordinates or their equivalence classes (kx, ky, k), with $k \neq 0$, in order to simplify the derivation and explanation of their properties. Points $\mathbf{x}$ will be 2-D projection coordinates (x, y) or 3-D $(x, y, 1)$ depending on the context.

4.1.1 Second Epipole and Point Depth

In pure translation (Figure 4.1), one might assume that the depth of point P with respect to the first (left) image is inversely proportional to the magnitude of the apparent displacement $\mathbf{x}' - \mathbf{x}$. Furthermore, the closer P is to a point Q, the smaller is the displacement $\mathbf{x}' - \mathbf{x}$, being 0 if $P = Q$. So, to the depth of the points over the circumference be constant, some quantity must compensate that decrease. Assume that multiplying the inverse of $\left\|\mathbf{x}' - \mathbf{x}\right\|$ by the distance between $\mathbf{x}'$ and $\mathbf{e}'$ in the second (right) image

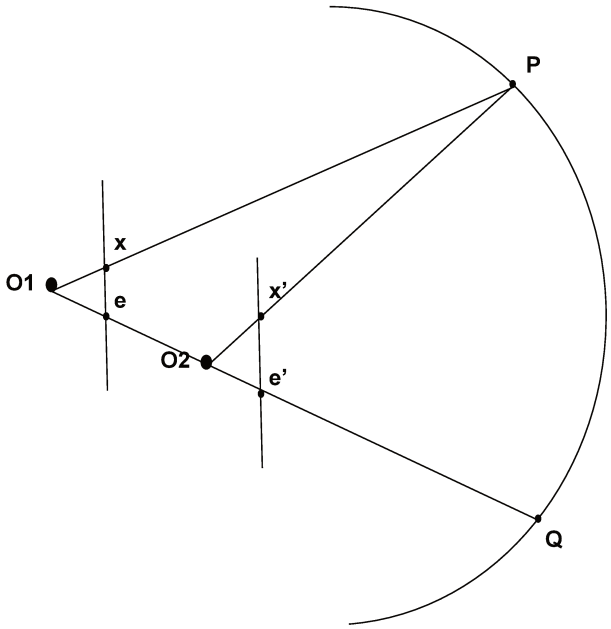


Figure 4.1: Example of epipolar geometry. $\mathbf{x}$ and $\mathbf{x}'$ are the images of the scene point P. $\mathbf{e}$ and $\mathbf{e}'$ are the first and second epipoles, respectively. The pose of the camera in the second view is a pure translation from the first view.

constrains the depth (in the local scale) of the points in the circumference of center $\mathbf{O}_1$ and radius $\mathbf{P} - \mathbf{O}_1$ to be equal. Then set the scene depth d of point $\mathbf{x}$ to be

$$d = \frac{\left\|\mathbf{x}' - \mathbf{e}'\right\|}{\left\|\mathbf{x}' - \mathbf{x}\right\|}. \tag{4.1}$$

When there is rotation, the points at $\mathbf{e}'$ and $\mathbf{e}$ have non-zero displacement from one image to the other. That means the "focus of expansion" is no longer at those points.

Next comes a demonstration that this intuition about the depth of a point given its apparent displacement and distance to the epipole is somewhat correct. From Longuet-Higgins [33] it holds that (assuming that the camera matrix is disregarded, or $K = I$) if $[R^\top|\mathbf{t}]$ is the relative pose of the second camera concerning the first, then

$$d = \frac{(\mathbf{r}_1 - x_1^{'}\mathbf{r}_3)\mathbf{t}}{(\mathbf{r}_1 - x_1^{'}\mathbf{r}_3)\mathbf{x}} = \frac{(\mathbf{r}_2 - x_2^{'}\mathbf{r}_3)\mathbf{t}}{(\mathbf{r}_2 - x_2^{'}\mathbf{r}_3)\mathbf{x}} \tag{4.2}$$

from where

$$d = \frac{\begin{bmatrix}1 & 0 & -x_1^{'}\end{bmatrix} R\mathbf{t}}{\begin{bmatrix}1 & 0 & -x_1^{'}\end{bmatrix} R\mathbf{x}} = \frac{\begin{bmatrix}0 & 1 & -x_2^{'}\end{bmatrix} R\mathbf{t}}{\begin{bmatrix}0 & 1 & -x_2^{'}\end{bmatrix} R\mathbf{x}} \tag{4.3}$$

and then

$$d = \left\|d\right\| = \frac{\left\|\begin{bmatrix}1 & 0 & -x_1^{'} \\ 0 & 1 & -x_2^{'}\end{bmatrix} R\mathbf{t}\right\|}{\left\|\begin{bmatrix}1 & 0 & -x_1^{'} \\ 0 & 1 & -x_2^{'}\end{bmatrix} R\mathbf{x}\right\|}. \tag{4.4}$$

If $R = I$ and $t_z \neq 0$ then

$$d = \frac{\left\|\mathbf{t} - t_z\mathbf{x}'\right\|}{\left\|\mathbf{x} - \mathbf{x}'\right\|} = \left\|t_z\right\| \frac{\left\|\mathbf{x}' - \mathbf{e}'\right\|}{\left\|\mathbf{x}' - \mathbf{x}\right\|}, \tag{4.5}$$

that is the same as Equation 4.1 up to $\left\|t_z\right\|$ if $t_z \neq 0$. Section 4.1.3 will show how to use $\mathbf{e}'$ instead of $\mathbf{t}$ so that Equation 4.4 is the "general form" of Equation 4.1. It is also possible to infer from Equation 4.4 that d may get bad values as the denominator approaches zero because noise becomes more relevant. Then, one possibility is to set w_m in Equation 2.41 to a non-decreasing function of that denominator.

In order to get to Equation 4.4 use the property that if $A, B, C, D \in \mathbb{R}$, $B \neq 0$, $D \neq 0$, and

$$\frac{A}{B} = \frac{C}{D}, \tag{4.6}$$

with

$$C = Ak, D = Bk, k \in \mathbb{R}, \tag{4.7}$$

then

$$\frac{\left\|\begin{bmatrix} A \\ C \end{bmatrix}\right\|}{\left\|\begin{bmatrix} B \\ D \end{bmatrix}\right\|} = \frac{\left\|\begin{bmatrix} A \\ Ak \end{bmatrix}\right\|}{\left\|\begin{bmatrix} B \\ Bk \end{bmatrix}\right\|} = \frac{\sqrt{A^2(1+k^2)}}{\sqrt{B^2(1+k^2)}} = \frac{\left\|A\right\|}{\left\|B\right\|}. \tag{4.8}$$

4.1.2 Uncertainty of Measurements

One downside of getting rid of feature parameters - like inverse depth - is that one loses the ability to model the uncertainty of measurements at the point-level granularity (e.g., assign low importance to the errors of points with high uncertainty). In a second-order optimization setup (e.g., with Levenberg-Marquardt), this lack of feature parameters translates into a much more compact and "less informative" Hessian structure. One possibility is to create many "candidate epipoles" per-frame where each can be initialized at different locations or fit to a subset of the points. Then one epipole would be derived or selected from the candidates. That is left for further investigations.

4.1.3 Properties of Epipoles

Suppose the second epipole in the image is modeled as a parameter, and it is desired to avoid redundancy of those parameters in an optimization window for bundle adjustment. In that case, one option is to take advantage of some relationship between the epipoles of adjacent frames. In particular, from two-view geometry (Hartley and Zisserman [24])

$$\mathbf{e}' = K\mathbf{t}_{21} \tag{4.9}$$

and

$$\mathbf{e} = KR_{21}^{\top}\mathbf{t}_{21}, \tag{4.10}$$

where $\mathbf{e}'$ is the second epipole in the second image concerning the first, $\mathbf{e}$ is the corresponding first epipole - which is also the second epipole of the first image for the second, and K is the intrinsic camera matrix (monocular case). R_{ij} and $\mathbf{t}_{ij}$ are the pose components of frame i with respect to frame j. Also, from Equations 4.9 and 4.10

$$\mathbf{e} = KR_{21}^{\top}K^{-1}\mathbf{e}'. \tag{4.11}$$

This implies that it is unnecessary to create separate parameters for both epipoles if one also wants to minimize some reprojection error from the second image into the first. It also implies that in pure translation, the two epipoles are equal.

It also would be interesting to have some "inductive" property of second epipoles, so that one could define the second epipole in a third image concerning the first ($\mathbf{e}'_{31}$) given the second epipoles in the second w.r.t the first ($\mathbf{e}'_{21}$) and given the third w.r.t the second ($\mathbf{e}'_{32}$). Thus, if

$$\mathbf{e}'_{31} = K[R_{31}|\mathbf{t}_{31}][\mathbf{0}|1]^{\top} = K\mathbf{t}_{31} \tag{4.12}$$

and

$$[R_{31}|\mathbf{t}_{31}] = [R_{32}|\mathbf{t}_{32}][R_{21}|\mathbf{t}_{21}] \tag{4.13}$$

then

$$\mathbf{e}'_{31} = KR_{32}\mathbf{t}_{21} + K\mathbf{t}_{32} \tag{4.14}$$

$$\mathbf{e}'_{31} = KR_{32}K^{-1}\mathbf{e}'_{21} + \mathbf{e}'_{32}. \tag{4.15}$$

Notice that the composition of second epipoles is equivalent to the aggregation of translations between consecutive frames, as expected, given the relationship in Equation 4.9. We treat the translation and the second epipole as separated parameters because we want more freedom to fit geometry, which is represented by $\mathbf{e}'$. Otherwise, if there were no epipole, the optimization would be equivalent to pose-graph optimization.

Also, it might be useful to define a "second epipole" function relating second epipoles from every pair of images:

$$E : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^2, \tag{4.16}$$

where its arguments are the indexes of the second and first image frames in the sequence, respectively. This function could be defined inductively as

$$\begin{cases} E(i,i) \coloneqq K \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\top} & i \in \mathbb{N} \\ E(i,j) \coloneqq KR_{i,j+1}K^{-1}E(j+1,j) + E(i,j+1) & i > j \\ E(i,j) \coloneqq KR_{i,j}^{\top}K^{-1}E(j,i) & j > i \end{cases}. \tag{4.17}$$

It is going to be well-defined if the values, $E(i+1,i)$, K, and $R_{i+1,i}$ are given. In an optimization setup, the optimizer would try to find the $E(i+1,i)$ that best fits the definition of E, and the number of parameters could be linear in the number of image frames.

Simplifying the definition of E above leads to

$$E(i,j) = KR_i \sum_{\tau=j+1}^{i} R_\tau^{-1} K^{-1} E(\tau, \tau-1), \tag{4.18}$$

where R_i and R_τ are rotations with respect to world coordinates. Now it is possible to define a continuous version of E that could be useful for parametrizations with an interpolation:

$$E(t_1,t_0) = KR_{t_1} \int_{t_0}^{t_1} R_\tau^{-1} K^{-1} \xi(\tau) d\tau \tag{4.19}$$

where

$$\xi(t) \coloneqq \lim_{h\to 0} E(t, t-h). \tag{4.20}$$

Even though $E(t,t) = 0$, the limit above can assume other values, like with pure translation with some x or y component. The limit is the constant value of the epipoles across the interval, except at t. In the continuous formulation, t can be interpreted as being time.

If the motion is planar, then the second epipole has only one degree of freedom corresponding to its position along the x axis. In such a case, its y coordinate is constant at the center of height.

4.2 Jacobian of Geometric Reprojection Error

The matrix $\mathbf{J}_0 = \dfrac{\partial \mathbf{r}}{\partial \mathbf{x}}(\mathbf{x}_0)$ is known as the *Jacobian* around $\mathbf{x}_0$. The present notation has N rows, which is the number of residuals, and D columns, which is the number of parameters to be optimized (the dimension of column vector $\mathbf{x}$). Thus, H is a $D \times D$ matrix and $\mathbf{b}$ is a D-dimensional column vector. For the Epipolar VO algorithm, finding the Jacobian of the reprojection error as in Section 2.7 might be useful. In that case, $D = 6 + 3 = 9$ is the total number of parameters of pose and epipole from a pair of consecutive frames.

Depending on whether we compute the reprojection error in pixel space or projection plane, the part of the Jacobian concerning the epipole may be multiplied by the calibration matrix K or not. Because we assume that K is already optimal, we work on the projection plane without needing to include K.

First, we show the Jacobian of reprojection residuals concerning the pose part of parameters in the algebra of $SE(3)$, $\zeta = (\mathbf{t}, \omega)$. Let the reprojection residual of a single point pair $(\mathbf{p}, \mathbf{p}')$ in the projection plane coordinates be

$$r = \left\| \Pi(T_0 e^{\zeta}(d\mathbf{p})) - \mathbf{p}' \right\|_2^2, \tag{4.21}$$

with

$$d = \|d\| = \frac{\left\| \begin{bmatrix} 1 & 0 & -p_1' \\ 0 & 1 & -p_2' \end{bmatrix} R_0 e^{\omega} \mathbf{e}' \right\|}{\left\| \begin{bmatrix} 1 & 0 & -p_1' \\ 0 & 1 & -p_2' \end{bmatrix} R_0 e^{\omega} \mathbf{p} \right\|} \tag{4.22}$$

as in Equation 4.4, with e^{ζ} being the exponential map of ζ (see Section 2.2.3) in the homogeneous transformation form, and e^{ω} being the rotation part of that form. $\mathbf{e}'$ is the second epipole in the projection plane (not in pixel coordinates).

The Jacobian $\left.\frac{\partial r}{\partial \zeta}\right|_{\zeta=0}$ follows from successive applications of the chain rule, product rule, and division rule. The most important "sub-Jacobians" are going to be $\left.\frac{\partial d}{\partial \zeta}\right|_{\zeta=0}$, $\left.\frac{\partial e^{\zeta}}{\partial \zeta}\right|_{\zeta=0}$, and $\left.\frac{\partial e^{\omega}}{\partial \zeta}\right|_{\zeta=0}$. The last two we take from Blanco [3]:

$$\left.\frac{\partial e^{\zeta}}{\partial \zeta}\right|_{\zeta=0} = \begin{bmatrix} \mathbf{0}_{3\times3} & -B_0 \\ \mathbf{0}_{3\times3} & -B_1 \\ \mathbf{0}_{3\times3} & -B_2 \\ I_3 & \mathbf{0}_{3\times3} \end{bmatrix}_{12\times6} \tag{4.23}$$

$$\left.\frac{\partial e^{\omega}}{\partial \zeta}\right|_{\zeta=0} = \begin{bmatrix} \mathbf{0}_{3\times3} & -B_0 \\ \mathbf{0}_{3\times3} & -B_1 \\ \mathbf{0}_{3\times3} & -B_2 \end{bmatrix}_{9\times6}, \tag{4.24}$$

where B_i are the generators of the algebra of $SO(3)$ introduced in Section 2.2.3. The Jacobian columns are arranged to correspond to the "denominator" stacked column by column in one long column vector. Then let

$$A = \begin{bmatrix} 1 & 0 & -p_1' \\ 0 & 1 & -p_2' \end{bmatrix} R_0 e^{\omega} \mathbf{e}' \tag{4.25}$$

and

$$B = \begin{bmatrix} 1 & 0 & -p_1' \\ 0 & 1 & -p_2' \end{bmatrix} R_0 e^{\omega} \mathbf{p}, \tag{4.26}$$

then

$$d = \frac{(A^{\top}A)^{1/2}}{(B^{\top}B)^{1/2}} \tag{4.27}$$

and

$$\frac{\partial d}{\partial \zeta} = \frac{1}{B^{\top}B}\left((A^{\top}A)^{-1/2}(B^{\top}B)^{1/2}A^{\top}\frac{\partial A}{\partial \zeta} - (B^{\top}B)^{-1/2}(A^{\top}A)^{1/2}B^{\top}\frac{\partial B}{\partial \zeta} \right), \tag{4.28}$$

so that we need $A|_{\zeta=0}$, $B|_{\zeta=0}$, $\left.\frac{\partial A}{\partial \zeta}\right|_{\zeta=0}$, and $\left.\frac{\partial B}{\partial \zeta}\right|_{\zeta=0}$ in order to find $\left.\frac{\partial d}{\partial \zeta}\right|_{\zeta=0}$. The first two are easy, as we just replace e^{ω} by I_3. The other two may be tricky to visualize because they are 3-D tensors, but we can separate the last dimension of the Jacobian corresponding to

the parameters ζ and find the individual expressions:

$$\left.\frac{\partial A}{\partial \zeta_i}\right|_{\zeta=0} = \begin{bmatrix} 1 & 0 & -p_1^{'} \\ 0 & 1 & -p_2^{'} \end{bmatrix} R_0 \left.\frac{\partial e^{\omega}}{\partial \zeta_i}\right|_{\zeta=0} \mathbf{e}' \tag{4.29}$$

$$\left.\frac{\partial B}{\partial \zeta_i}\right|_{\zeta=0} = \begin{bmatrix} 1 & 0 & -p_1^{'} \\ 0 & 1 & -p_2^{'} \end{bmatrix} R_0 \left.\frac{\partial e^{\omega}}{\partial \zeta_i}\right|_{\zeta=0} \mathbf{p}. \tag{4.30}$$

Likewise, we get $\left.\frac{\partial d}{\partial \zeta_i}\right|_{\zeta_i=0}$ by plugging in the corresponding values in Equation 4.28. The result will be a 1×6 row matrix. This strategy works whenever there is a Jacobian multiplying a vector or matrix on the right.

Then we can proceed by applying the chain and product rules procedurally, as the final nonreducible form may be too convoluted and not be worth the risk of error from an implementation point of view:

$$\left.\frac{\partial r}{\partial \zeta}\right|_{\zeta=0} = 2(\Pi(T_0(d_0\mathbf{p})) - \mathbf{p}')^{\top} \left.\frac{\partial \Pi}{\partial T_0 e^{\zeta}(d\mathbf{p})}\right|_{\zeta=0} R_0 \left(\left.\frac{\partial e^{\zeta}}{\partial \zeta}\right|_{\zeta=0} (d_0\mathbf{p}) + \mathbf{p} \left.\frac{\partial d}{\partial \zeta}\right|_{\zeta=0} \right), \tag{4.31}$$

where d_0 is d at $\zeta = 0$, and

$$\Pi(X) = \Pi([x, y, z]^{\top}) = [x/z, y/z, 1]^{\top} \tag{4.32}$$

and

$$\frac{\partial \Pi}{\partial X} = \frac{1}{z} \begin{bmatrix} 1 & 0 & -x/z \\ 0 & 1 & -y/z \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.33}$$

assuming $z > 0$. A C++ implementation of the Jacobian of this reprojection error is available[1].

The Jacobian with respect to the epipole $\left.\frac{\partial r}{\partial \mathbf{e}'}\right|_{\mathbf{e}'=\mathbf{e}'_0,\zeta=0}$ follows from $\left.\frac{\partial d}{\partial \mathbf{e}'}\right|_{\mathbf{e}'=\mathbf{e}'_0,\zeta=0}$, which is

$$\left.\frac{\partial d}{\partial \mathbf{e}'}\right|_{\mathbf{e}'=\mathbf{e}'_0,\zeta=0} = \frac{1}{(B^{\top}B)^{1/2}} \left((A^{\top}A)^{-1/2} A^{\top} \frac{\partial A}{\partial \mathbf{e}'} \right) \Bigg|_{\mathbf{e}'=\mathbf{e}'_0,\zeta=0}, \tag{4.34}$$

where

$$\left.\frac{\partial A}{\partial \mathbf{e}'}\right|_{\mathbf{e}'=\mathbf{e}'_0,\zeta=0} = \begin{bmatrix} 1 & 0 & -p_1^{'} \\ 0 & 1 & -p_2^{'} \end{bmatrix} R_0. \tag{4.35}$$

The appearance of $\mathbf{e}'_0$ is because we want to discern the current value of the epipole from the parameter epipole being optimized. Now suppose that the reprojection error is not a function that involves just R_0, t_0, and $\mathbf{e}'_0$ because the reprojection is not between consecutive frames, but distant frames t_0 and t_1 ($t_1 > t_0$). If the parameters are the poses and epipoles between consecutive frames in the forward order, then $T_0 e^{\zeta}$ in Equation 4.21

[1]Refer to `https://github.com/Ronnypetson/epivo/blob/master/test_jac.cpp`

must be replaced with the product (on the left)

$$\prod_{t=t_0}^{t_1-1} T_0^{(t)} e^{\zeta^{(t)}}, \tag{4.36}$$

while $R_0 e^{\omega}$ must be replaced with (also a product of successive multiplications on the left)

$$\prod_{t=t_0}^{t_1-1} R_0^{(t)} e^{\omega^{(t)}} \tag{4.37}$$

in Equation 4.22 and $\mathbf{e}'$ must be replaced by

$$E(t_1, t_0) = E^{(t_1-1)} + \sum_{t=t_0+1}^{t_1-1} \left(\left(\prod_{k=t}^{t_1-1} R_0^{(k)} e^{\omega^{(k)}} \right) E^{(t-1)} \right), \tag{4.38}$$

which is the same as Equation 4.18, but in terms of local rotations. $E^{(x)} \coloneqq E(x+1, x)$ is the generalization of epipole introduced in Section 4.1. Now the Jacobians we need are

$$\left. \frac{\partial E(t_1, t_0)}{\partial E^{(k)}} \right|_{E^{(k)}=E_0^{(k)}, \zeta=0} = \prod_{t=k+1}^{t_1-1} R_0^{(t)}, \tag{4.39}$$

$$\left. \frac{\partial \prod_{t=t_0}^{t_1} R_0^{(t)} e^{\omega^{(t)}}}{\partial \zeta_i^{(k)}} \right|_{\zeta=0} = \left(\prod_{t=k+1}^{t_1-1} R_0^{(t)} \right) \left. \frac{\partial e^{\omega}}{\partial \zeta_i} \right|_{\zeta=0} \left(\prod_{t=t_0}^{k} R_0^{(t)} \right), \tag{4.40}$$

and

$$\left. \frac{\partial E(t_1, t_0)}{\partial \zeta_i^{(k)}} \right|_{\zeta=0} = \sum_{t=t_0+1}^{k} \left(\left(\prod_{j=k+1}^{t_1-1} R_0^{(j)} \right) \left. \frac{\partial e^{\omega}}{\partial \zeta_i} \right|_{\zeta=0} \left(\prod_{j=t}^{k} R_0^{(j)} \right) E^{(t)} \right). \tag{4.41}$$

If there are backward reprojections (from a frame to a previous), which is the case for loop closure, first write the inverse of the reprojection transformation and then derive the Jacobian analogously.

4.3 Feature Extraction and Matching

Figure 4.2 depicts the pipeline proposed for our VO algorithm. The first step in the pipeline consists of taking pairs of subsequent images of a sequence and then search for points of interest in both. Then it stores the matches for the subsequent frames (indexes i and $i+1$) in a hash-table so that those points are going to be filtered in the key-point selection stage.

The same procedure as above applies for any pair of images (not necessarily in forward order or adjacent) specified in the **window of reprojections** (see Figure 4.3). For instance, to optimize reprojection errors between frames i and j inside a window of time

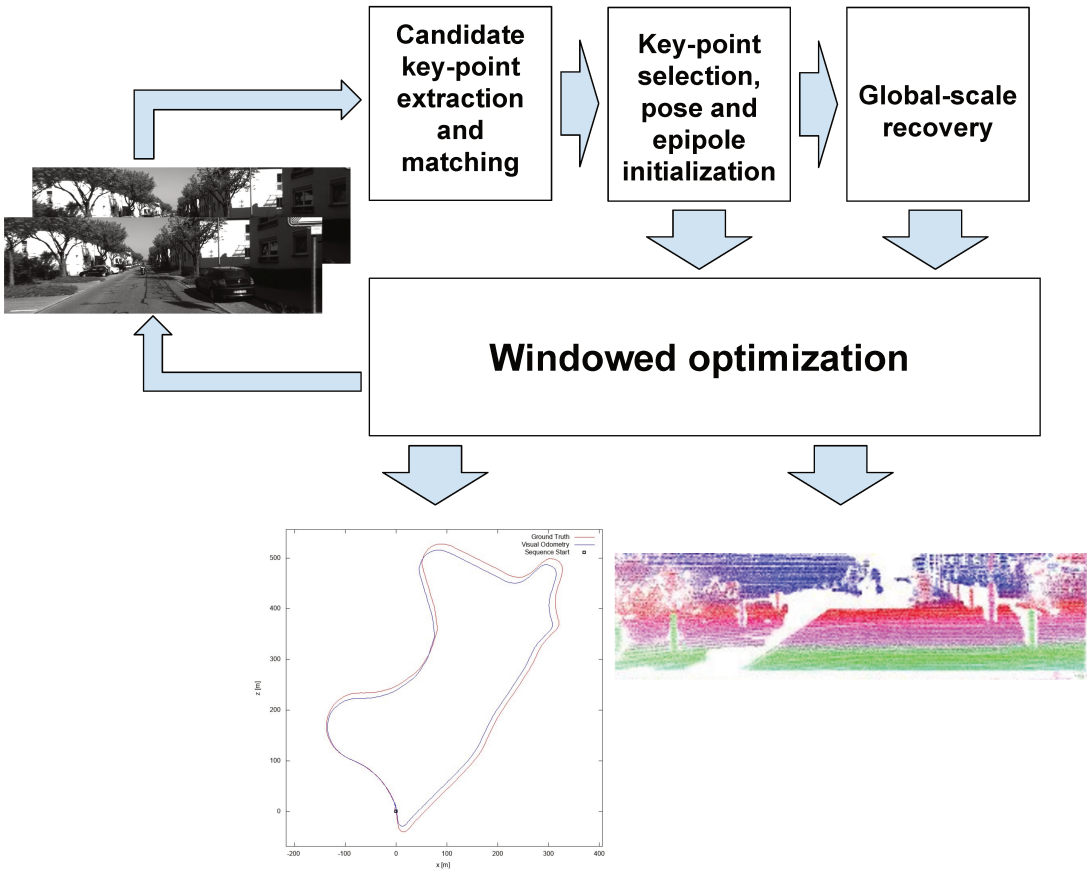


Figure 4.2: High-level view of the Epipolar VO pipeline.

steps (at frame frequency), the feature extraction and matching procedure takes images i and j. The points and matches will also be stored in a hash table that maps from the pair of indices (i, j) to the point matches from i to j. The key-point selection will also apply to the points in those matches. More details about the implementation of point matching are available in Chapter 5.

At this stage, there can be thousands of point matches for each pair of images. Then, after the key-point selection procedure described in Section 4.4, this is reduced to hundreds of points or less.

4.4 Initialization and Key-point Selection

After the extraction and matching steps in each reprojection pair in a window of frames, the next step is to apply RANSAC in order to get the essential matrix (see Section 2.4.2) and selecting a subset of the points for each pair of images in the reprojection window. Those resulting key (not key-points yet) are the consensus set resulting from RANSAC, such that they agree upon the essential matrix up to a margin of error $\epsilon < 0.1$ pixel.

Then, the pipeline extracts the relative pose of each pair of frames - pose of the second concerning the first - by **triangulation** of the points selected in the previous step, such that a new subset of points agrees on the recovered pose. This last subset of points is used

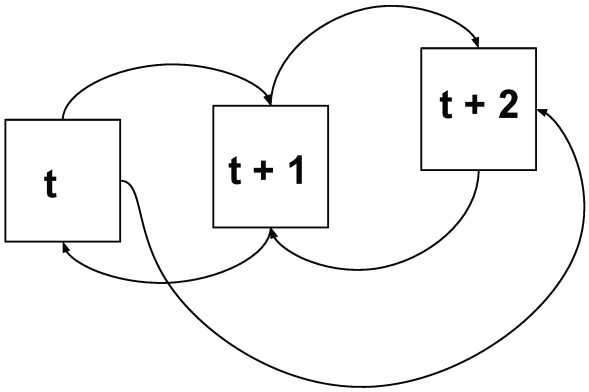


Figure 4.3: Example of reprojection graph with window size $w = 3$ and strides of reprojection $s = \{+1, -1, +2\}$. The arrows indicate that the key-points are matched from the incoming image to the target image.

as the key-points for the given pair of frames. The resulting poses do not have the correct scale so that the rotation part is usually close to the ground truth, but the translation part always has magnitude 1, even though it points in the right direction. The algorithm recovers the global scale afterward (see Section 4.5).

If a pose's rotation deviates too much from the identity, it is set to identity and the translation part is set to $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$. Otherwise, if the translation has all components with magnitude less than 0.8, it is set to $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$.

Also, if the median magnitude of the key-points' parallax is less than 1e−2 pixels or the number of key-points is less than 64, the rotation part receives the identity value, and the epipole initial value is the center of the camera. Otherwise, the pose remains as it is, and the epipole initial value is the translation part of the relative pose projected into the image plane.

The initial value of the parameters to be optimized - poses and epipoles - is the relative poses of the consecutive frames and the respective epipoles (Section 4.6). Algorithm 2 summarizes this process.

4.5 Global Scale Computation

In order to recover the real scale of the pose transformation ($T_{i+1,i}^{L,L}$) between subsequent left frames at instants i and $i+1$ (just one pair in the reprojection window), it is sufficient to have the ground-truth relative pose ($T_{i,i}^{L,R}$) between the two cameras of the stereo rig at instant i (left and right) and, assuming that the left image is the one used for monocular odometry, the left image at frame $i+1$. Because $T_{i,i}^{L,R}$ has the correct scale, in order to recover the scale of $T_{i+1,i}^{L,L}$, it is sufficient to find values s_0, s_1 such that

$$\begin{bmatrix} R_{i,i+1} & s_1\mathbf{t}_{i,i+1} \\ \mathbf{0} & 1 \end{bmatrix}^{L,R} \begin{bmatrix} R_{i+1,i} & s_0\mathbf{t}_{i+1,i} \\ \mathbf{0} & 1 \end{bmatrix}^{L,L} = \begin{bmatrix} R_{i,i} & \mathbf{t}_{i,i} \\ \mathbf{0} & 1 \end{bmatrix}^{L,R} \tag{4.42}$$

Algorithm 2: Initialization and key-point selection. S and T are lists of 2-D point coordinates from consecutive images. A point $S[i]$ is the match of $T[i]$ and vice-versa.

Result: Initial values for poses (R, t) and epipoles (e); selected feature points and their matches (source, target) (S'', T'') and the reprojection weight s.

$E, S', T' \longleftarrow RANSAC\_Essential\_Matrix(S, T)$
$R, t, S'', T'' \longleftarrow Triangulate(E, S', T')$
$s \longleftarrow 1$
if $Tr(R) < 2.4$ **then**
  $R \longleftarrow I_3$
  $t \longleftarrow \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$
  $s \longleftarrow 1\text{e}{-14}$
else
  if $max(|t|) < 0.8$ **then**
    $t \longleftarrow \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$
    $s \longleftarrow 1\text{e}{-14}$
  end
end
if $median(abs(S'' - T'')) < 1\text{e}{-2}$ OR $|S''| < 64$ **then**
  $R \longleftarrow I_3$
  $e \longleftarrow \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$
else
  $e \longleftarrow t$
end

and then take s_0 as the desired scale factor (see Figure 4.4). This is equivalent to solving the linear system below (Equation 4.43). This step of the algorithm finds the solution by least-squares to ignore problems with ill-conditioned matrices.

$$\begin{bmatrix} R_{i,i+1}^{L,R}\mathbf{t}_{i+1,i}^{L,L} & \mathbf{t}_{i,i+1}^{L,R} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} = \mathbf{t}_{i,i}^{L,R}. \tag{4.43}$$

Even though the scale can be included as a parameter and be optimized jointly with other parameters, this possibility is left for further investigations.

4.6 Windowed (Indirect) Bundle Adjustment

Given the initial values of the relative poses between consecutive frames and those of the respective epipoles, the algorithm solves many optimizations, one for each position of the sliding window. There is no overlap between the positions of the sliding window. For a particular position in the sequence, the algorithm minimizes a cost function that depends on the window's reprojection errors.

Each sliding window position uses the same reprojection pattern, a graph with the arrows representing the directed pairs of frames to be considered (see Figure 4.3). A

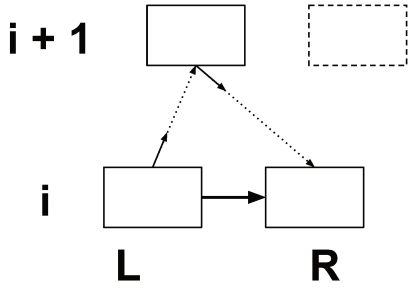


Figure 4.4: The relation between the real displacement of the stereo pair and the displacement in time determines the global scale.

simple way to aggregate the reprojection errors (Equation 2.41) corresponding to the arrows is to take the average as the window's cost function, although it might be good to filter the reprojections of bad initializations.

When the reprojection involves two frames that are not consecutive *or* are not in the forward order, derive the relative poses and epipoles from the parameters, which are the consecutive displacements and epipoles along the forward path that connect the two frames. When the target frame has an index smaller than the source frame, invert the poses. Follow Equation 4.18 to derive the epipole.

Consider the following two ways of aggregating the reprojection errors inside a window. Both attribute weights to the reprojections so that reprojections identified as bad during initialization are assigned with very low weight (1e−14), while good initializations have weight 1. The difference between the two is whether the key-points have weights or are treated equally. In the first case, the weights of the key-points are proportional to the denominator in Equation 4.4, so that points with less parallax or with depth estimates supposedly more contaminated with noise have less importance. The algorithm implements the first version of error aggregation, although further implementations might use some version similar to the second.

The epipole parametrization that we present is agnostic about the type of error to be minimized, but we minimize the geometric error because we use point-like feature matching in the experiments. The steps explained above are summarized in Algorithm 3.

4.7 Global Poses and Scene Structure

As the parameters that go through optimization are the relative displacements between adjacent frames in the forward order, to get the poses concerning the world coordinate, it is necessary to concatenate the poses. The global poses follow from Equation 2.9.

Regarding scene geometry, given the pose and epipole corresponding to any two frames, for any given set of point associations from the first frame to the second, Epipolar VO recovers the depth map of those points by applying Equation 4.4 for each point.

Algorithm 3: High-level view of the algorithm. It is the optimization of poses given reprojection graph G with window size w as a list of edges ($x = (i, j)$, where i and j are frame indices), and key-point (source, target) matches (S, T) for every $x \in G$. L is the total length of the sequence. The *Initialize* routine is the same as in Algorithm 2. The *Compute_scale* and *Extract_and_match* routines are described in Sections 4.5 and 4.3, respectively.

Result: Poses (R, t) and epipoles (e) for the entire sequence after minimization of reprojection errors; global-scale factors (z).

$start \longleftarrow 0$
while $start < L - w$ **do**
  $G' \longleftarrow Shift(G, start)$
  $params \longleftarrow \{\}$
  $scales \longleftarrow \{\}$
  for $x \in G'$ **do**
    $u, v \longleftarrow x$
    $S, T \longleftarrow Extract\_and\_match(I_u, I_v)$
    $R_x, t_x, e_x, S_x, T_x, s_x \longleftarrow Initialize(S, T)$
    if $v = u + 1$ **then**
      $params \longleftarrow Insert(params, [R_x, t_x, e_x])$
      $z \longleftarrow Compute\_scale(I_u, I_v)$
      $scales \longleftarrow Insert(scales, z)$
    end
    while *Not converge* **do**
      $cost \longleftarrow 0$
      for $x \in G'$ **do**
        $u, v \longleftarrow x$
        $R_{u,v}, t_{u,v}, e_{u,v} \longleftarrow Recover(params, u, v)$
        $T'_x \longleftarrow Reproject(S_x, R_{u,v}, t_{u,v}, e_{u,v}, K)$
        $error \longleftarrow \left\| T'_x - T_x \right\|_\gamma$
        $cost \longleftarrow cost + s_x error$
      end
      $cost \longleftarrow cost / |G'|$
      $J \longleftarrow Jacobian(cost, params)$
      $params \longleftarrow Update(params, J)$
    end
  end
  $start \longleftarrow start + w$
end

Chapter 5

Experimental Setup

This chapter describes the experiments analyzed in Chapter 6 on what concerns the data, the evaluation, and some implementation details of the Epipolar VO algorithm, introduced in Chapter 4.

5.1 Dataset

In this work, we used the camera sequences 00 to 10 from KITTI dataset (Geiger et al. [20]), as their ground-truth poses are available, differently from sequences 11 to 22. All sequences are outdoors and from car movement. The dataset contains two calibrated stereo pairs of cameras. One pair is gray-scale, and the other is RGB. It also contains LiDAR points, which are used as a reference in evaluating the scene reconstruction.

5.1.1 Camera Data

In all our experiments, the left image is the default for simulating the monocular setup. Our experiments only use the gray-scale stereo pair of images, which are 376 by 1241 pixels, when recovering the global scale (see Section 4.5).

5.1.2 Laser Points

The Velodyne laser points serve as the standard for checking the dense depth maps generated by Epipolar VO. Every frame of every sequence has an associated Velodyne set of points that can be projected into the image plane. Thus, it contains the depth information for a relatively dense set of points in the images, although the visible ones are mostly in the lower half of the images.

5.2 Evaluation Metrics

The experiments' main pose metrics are adapted from the tool available at the KITTI odometry benchmark, which computes average translational (%) and average rotational (rad/m) errors for sub-sequences of 100, 200, 300, ..., 800 meters. Because the publicly

available ground-truth poses are for sequences 00 - 10, the evaluation code needs adaptation to computes the errors against those poses instead of poses from sequences 11 - 21. The overall error of a sequence is the average of the errors of the subsequences. Also, we change the rotation error unity from rad/m to deg/100m.

For a given sequence from KITTI, the computation of the rotational and translational errors for a length l subsequence starting at position p is, provided that

$$E^{p,l} := (\hat{\mathbf{T}}_p^{-1}\hat{\mathbf{T}}_{p+l})^{-1}(\mathbf{T}_p^{-1}\mathbf{T}_{p+l}), \tag{5.1}$$

- Translational error:

$$E_{\mathbf{t}}^{p,l} := \frac{\left\|\mathbf{t}^{p,l}\right\|_2}{l}, \tag{5.2}$$

where $\mathbf{t}^{p,l}$ is the translation part of $E^{p,l}$;

- Rotational error:

$$E_R^{p,l} := \frac{\arccos\max(\min(d,1),-1)}{l}, \tag{5.3}$$

where $R^{p,l}$ is the rotation part of $E^{p,l}$ and $d = \dfrac{Tr(R^{p,l})-1}{2}$.

The error associated with a subsequence of length l is the average of all errors from the start $0, 10, 20, \ldots$ multiples of 10 until it fits in the original sequence. The original sequence's final error is the average of the errors of the subsequences of lengths $100, 200, 300, 400, 500, 600, 700, 800$ that fit inside it.

When comparing the general formulation of depth with the limited pure translation case, we use the Relative Pose Error (RPE) metric. This metric is defined as follows

- Relative Pose Error (RPE):

$$RPE := \sqrt{\frac{1}{k}\sum_{i=1}^{n-k}\left\|\mathbf{E}_i\right\|_F} \tag{5.4}$$

where $\left\|.\right\|_F$ is the Frobenius norm and $\mathbf{E}_i = (\hat{\mathbf{T}}_i^{-1}\hat{\mathbf{T}}_{i+k})^{-1}(\mathbf{T}_i^{-1}\mathbf{T}_{i+k})$ and k is chosen depending on how local the accuracy should be. A small k tells more about local drift and large k about global drift (Sturm et al. [55]).

5.3 Implementation

The current implementation is not optimized for frame-rate, as the optimizer L-BFGS-B (Byrd et al. [6]) receives, at every iteration, Jacobians computed automatically with PyTorch's autograd (Paszke et al. [45]) and that requires expensive operations of type conversion between PyTorch's tensors and Numpy arrays (Harris et al. [23]). However, we have implemented an optimized version in C++ for the final algorithm. The project repository is available[1].

[1]Refer to `https://github.com/Ronnypetson/tfoe_vo`

The scene primitives that the algorithm matches are FAST corners (Trajkovic and Hedley [57]). The matches (from one image to another) come from Lucas-Kanade optical flow (Lucas and Kanade [36]). The implementation uses OpenCV (Bradski [4]) for both the point detection and matching steps.

5.4 Experiments

To approach distinct features of our Epipolar VO algorithm, we run three sets of experiments, as described in the following sections.

5.4.1 Parametrization Evaluation

To evaluate the effect of employing different estimations of keypoints' depth, we perform a set of experiments (Section 6.1) where with estimate depth using the particular case (Equation 4.1) and the general case (Equation 4.4) formulations. We run the experiments in two dataset sequences that have high rotations and low rotations.

5.4.2 Odometry Experiments

To evaluate our proposed approach, we run five different configurations of Epipolar VO with different window-stride configurations and keypoint weighting, denoted **I** to **V**, for the sequences 00 to 10 from KITTI. For these experiments, we measure the rotational and translational errors detailed in Section 5.2. We evaluate the error statistics concerning sub-sequence length and velocity and the qualitative aspect of the final Visual Odometry trajectory compared to the ground-truth. These experiments are presented in Section 6.2. The norm used in the reprojection error is the Huber norm.

5.4.3 Structure Experiments

To evaluate the depth maps of some pairs of frames provided by our parametrization, we run this set of experiments. The depth maps are generated by combining a dense Farneback (Farnebäck [14]) optical flow with the frame pair is optimized epipole. The evaluation consists of a visual comparison with the Velodyne scan's depth map corresponding to the frame pair. For each map (Velodyne and ours), we normalize the point depths so that the furthest point has a depth equal to 1. Results from this experiments are presented in Section 6.3.

Chapter 6

Results and Analysis

This section analyzes the suitability of the depth formulation, the effect of indirect bundle adjustment with small windows in the odometry, and the quality of reconstruction after optimization, having as standard the Velodyne points from the data.

6.1 Depth in General Motion vs Pure Translation

To evaluate the effect of employing different estimations of keypoints' depth, we perform this set of experiments. We show that, even when the relative difference in orientation is locally small (e.g., between consecutive frames), the keypoints' depth approximation proposed by Equation 4.1 results in inferior quality compared to the one proposed by Equation 4.4. Experiments performed with sequences 04 (very little rotation) and 03 (considerable rotation) of the KITTI dataset demonstrate that the general form of depth described in Equation 4.4 should be employed as a default estimator. In Figure 6.2 and Figure 6.1, the box-plots on the left are from the "pure translation" depth (Equation 4.1), while its "general" counterpart (Equation 4.4) is the other. Due to this confirmation, we employ the general formulation from now onwards.

6.2 Indirect Bundle Adjustment

We present qualitative and quantitative assessments of our Epipolar VO with different BA configurations through ATE and RPE metrics and trajectories in the following sections. We also break the trajectory errors by length and by speed for a more in-depth analysis. All the trajectory figures, as well as the error by length/speed figures in this section, are from configuration **IV** only, as we prioritize the analysis of the effect of the biggest bundle adjustment window. We compare our results with the results from Pereira et al. [46] and Song et al. [54]. The complete quantitative comparison is depicted in Table 6.2 and the overall error comparison (sequences 00-10) are presented in Table 6.1.

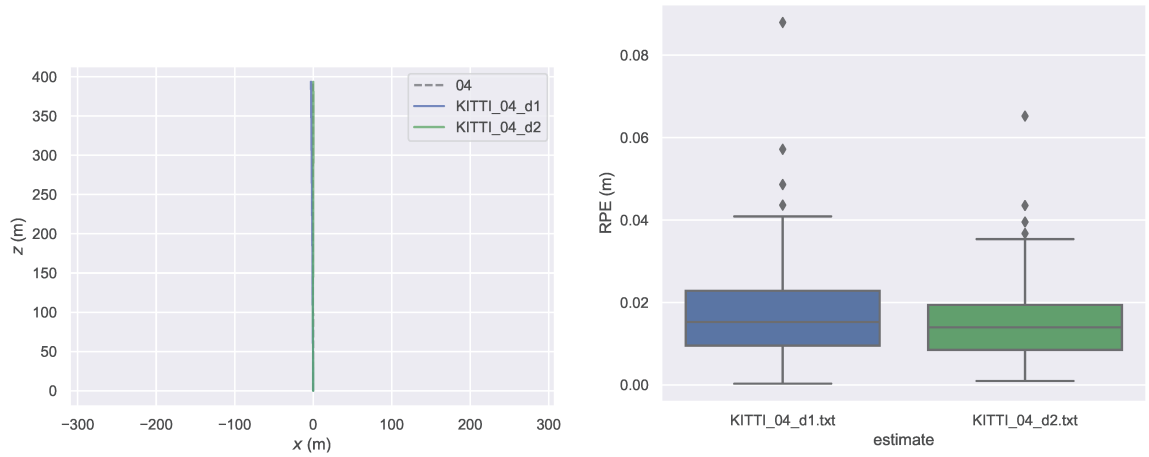


Figure 6.1: Trajectory and statistics of Relative Pose Error for sequence 04 from KITTI.

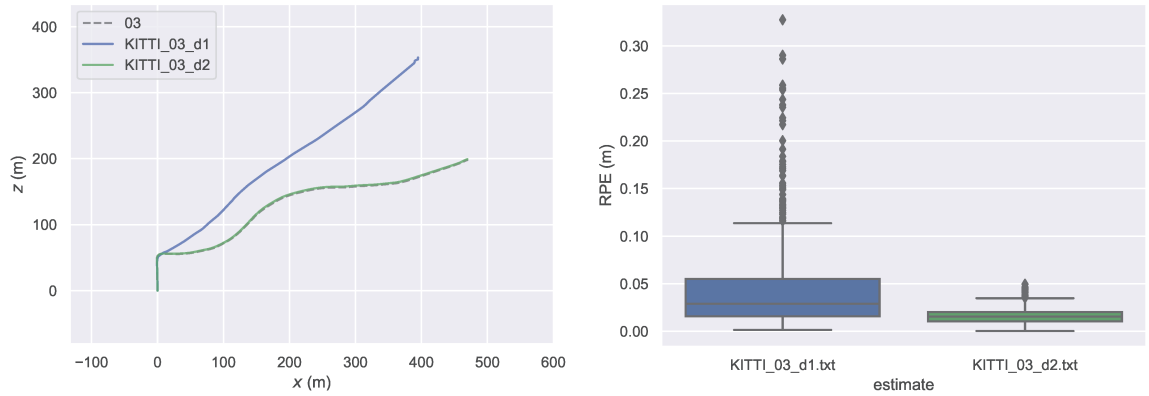


Figure 6.2: Trajectory and statistics of Relative Pose Error for sequence 03 from KITTI.

Table 6.1: Overall error comparison (sequences 00-10). s stands for stride of reprojection and w is sliding window size.

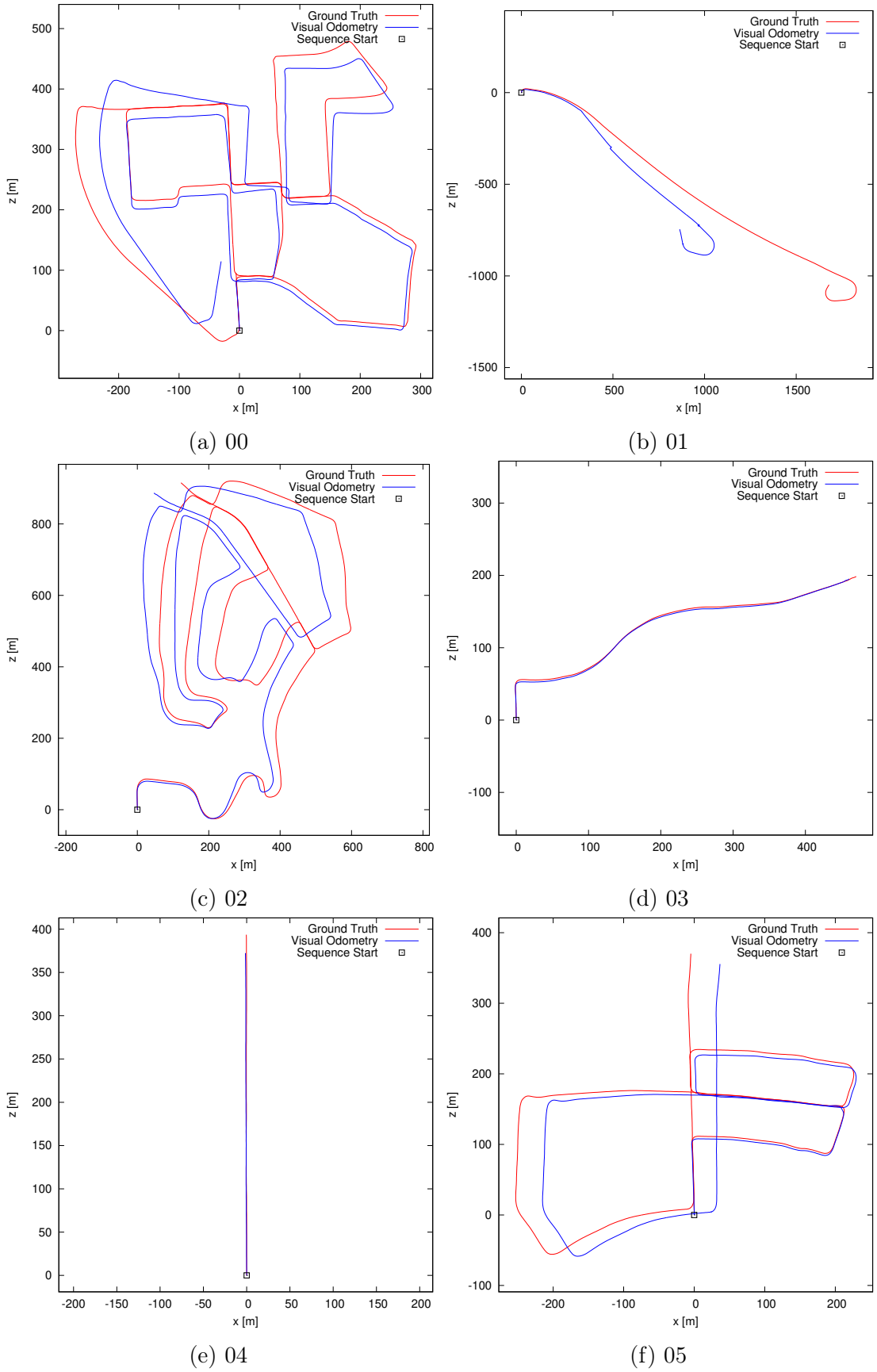
Config.	**Trans. (%)**	**Rot. (deg/100m)**
$s = \{\pm 1, \pm 2\}$, $w = 3$, key-point weighting **(I)**	4.95	1.00
$s = \{\pm 1\}$, $w = 2$ **(II)**	4.35	0.54
$s = \{\pm 1, \pm 2\}$, $w = 3$ **(III)**	4.94	0.81
$s = \{\pm 1, \pm 2\}$, $w = 4$ **(IV)**	4.55	0.65
Initialization only **(V)**	4.03	**0.31**
Pereira et al. [46]	**1.10**	**0.31**
Song et al. [54]	2.10	0.47

6.2.1 Sequence 00

Including our configurations and the results from Pereira et al. [46] and Song et al. [54], sequence 00 has medium to high difficulty in both rotation and translation. It is the second-longest sequence, making it more prone to difficult regions' appearance (e.g., bad texture, high movement of scene objects). Figure 6.3a shows that even though the odometry is good at many hard turns, some locations with severe error in rotation affect the trajectory's overall appearance. There is also a slight error in scale, reflecting regions with bad initialization of translation, like the one near the first turn.

For sequence 00, Figure 6.5 shows that rotation error statistics decrease with the increase of the subsequence length. This may be explained by the fact that, for the most part, the subsequences will have straight trajectories, and thus, the rotation error in turns are smoothed out. Translation error by length, on the other hand, seems to decrease much more softly. This may be because translation errors happen more uniformly, not depending so much on the trajectory aspect.

The fact that subsequences with higher speed have lower rotational error is counter-intuitive (Figure 6.6) but may be explained by the fact that the vehicle is at the lower speeds when it is making a turn, and that is precisely where the rotational error happen more. On the other hand, when the vehicle is going on a straight line, it increases its velocity, and the rotation part of the odometry is easier to estimate. That is not the same for translation, as straight accelerated trajåectories are harder to estimate translation with correct relative scale than constant velocity movement.



(a) 00

(b) 01

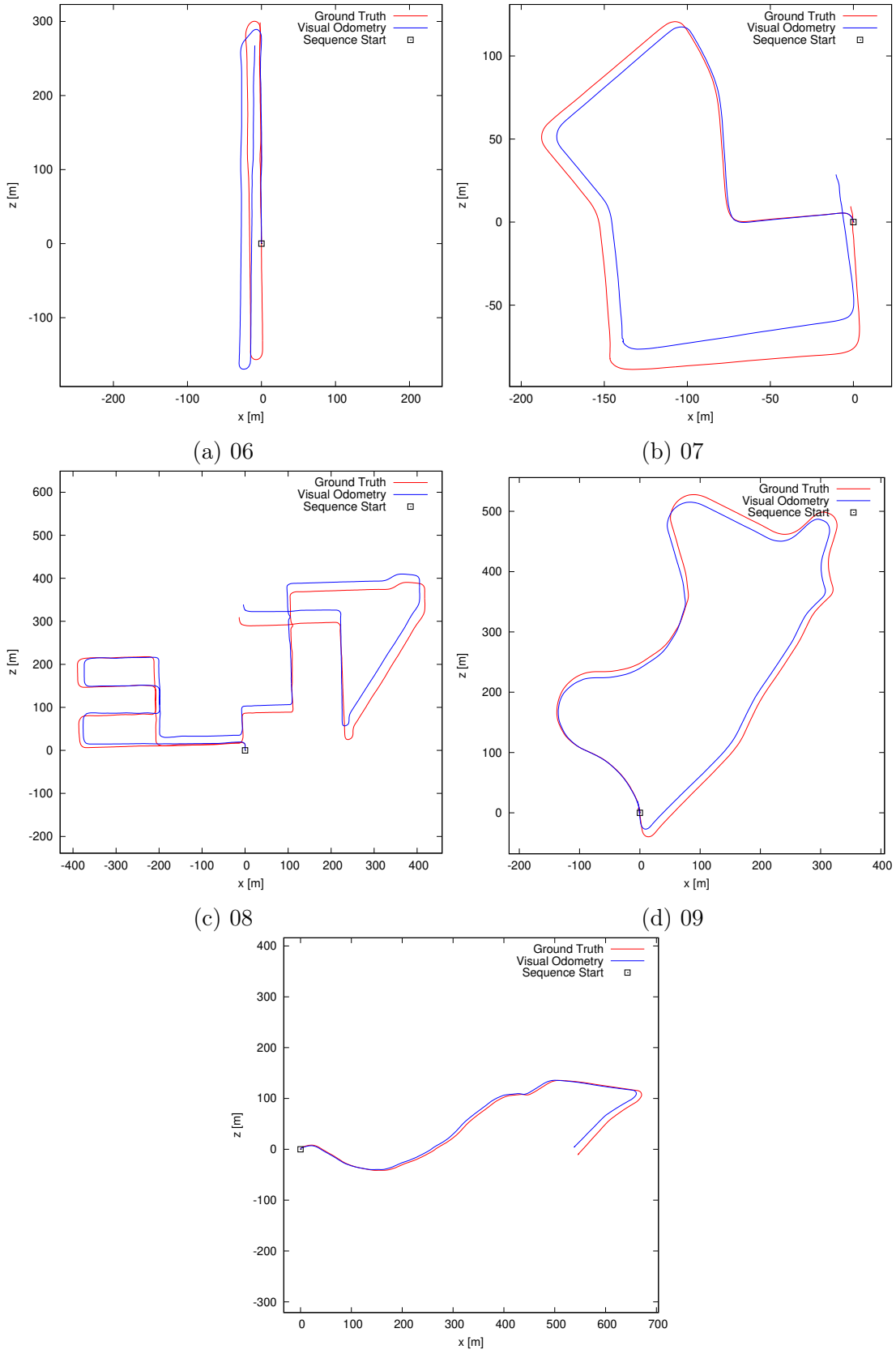
(c) 02

(d) 03

(e) 04

(f) 05

Figure 6.3: KITTI sequences from 00 to 05.



(a) 06

(b) 07

(c) 08

(d) 09

(e) 10

Figure 6.4: KITTI sequences from 06 to 10.

Table 6.2: Detailed error comparison (sequences 00-10). Translation and rotation errors are in % and deg/100m, respectively. The rows marked with **I** to **V** denote the configurations in Table 6.1. The last rows refer to Pereira et al. [46] and Song et al. [54].

	Seq.	00	01	02	03	04	05	06	07	08	09	10
I	**Tr.**	4.17	35.23	3.40	1.25	5.76	3.37	5.05	5.60	3.03	2.40	1.73
	Rot.	1.21	2.17	1.05	0.15	0.21	0.55	1.00	1.85	0.77	0.25	1.06
II	**Tr.**	2.83	35.89	2.29	1.29	5.75	4.67	3.00	2.29	2.50	2.47	1.24
	Rot.	0.32	2.30	0.29	0.13	0.21	1.27	0.36	1.05	0.30	**0.20**	0.28
III	**Tr.**	3.98	34.88	4.10	1.23	5.75	3.45	4.04	4.74	2.62	2.57	1.39
	Rot.	0.97	1.82	1.06	0.14	**0.14**	0.52	0.62	1.52	0.39	0.23	0.60
IV	**Tr.**	4.07	34.72	2.59	1.23	5.75	2.86	3.99	5.40	2.67	2.53	1.51
	Rot.	1.05	2.01	0.43	0.13	0.18	0.34	0.63	1.85	0.33	0.24	0.41
V	**Tr.**	2.82	34.50	2.18	1.24	5.75	2.73	3.12	2.49	2.47	2.53	1.25
	Rot.	0.32	0.62	**0.26**	**0.11**	0.20	**0.22**	0.37	0.65	**0.28**	0.23	0.26
[46]	**Tr.**	**1.03**	**1.37**	**1.33**	**0.87**	**0.86**	**0.99**	**0.73**	**1.12**	**1.23**	**1.54**	**1.02**
	Rot.	**0.30**	**0.23**	0.38	0.22	0.24	0.37	**0.26**	**0.57**	**0.28**	0.28	**0.24**
[54]	**Tr.**	2.04	-	1.50	3.37	1.43	2.19	2.09	-	2.37	1.76	2.12
	Rot.	0.48	-	0.35	0.21	0.23	0.38	0.81	-	0.44	0.47	0.85

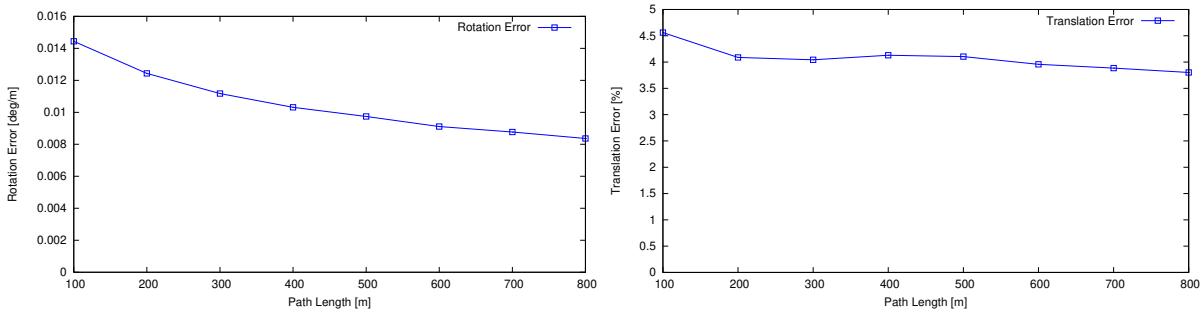


Figure 6.5: Errors by length for sequence 00.

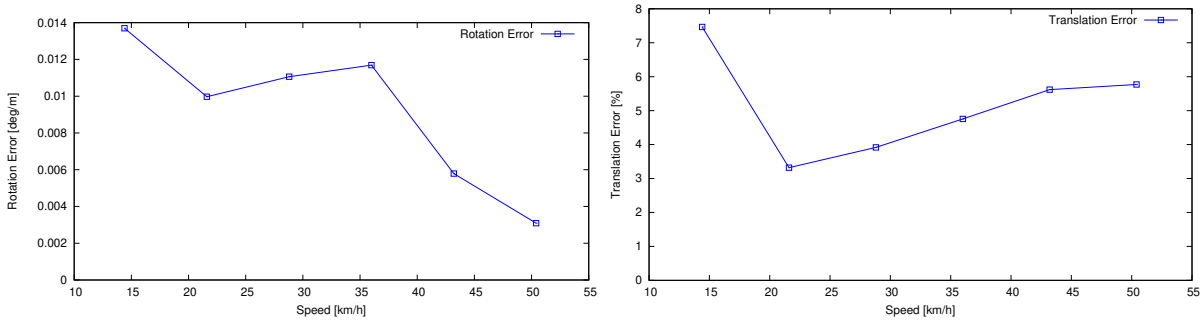


Figure 6.6: Errors by speed for sequence 00.

6.2.2 Sequence 01

The resulting trajectory for this sequence (Figure 6.3b) is the most affected in rotation and translation, the latter having a notable high error. This happens because of bad

initialization of translation, magnified because the global scale (Section 4.5) is computed based on it. This sequence is very poor in terms of texture and contains many moving objects that can divert the pose estimation if the key-points are over those objects.

Like in sequence 00, the rotation error statistics (Figure 6.7) is lower for longer subsequences for the same reason: longer subsequences contain more straight-line trajectories that decrease the mean rotation error raised in the parts with turns. For the translation part, the big error is accumulated after the main turn at the beginning, followed by a long and slightly curved trajectory that is almost straight locally. This region is heavily affected by the presence of other cars moving at different speeds.

The distribution of error given speed (Figure 6.8) indicates another possible reason for the difficulty of sequence 01: the range of speeds in the sequence is the biggest among all, reaching 85Km/h. The high velocities can make the key-point correspondence step very difficult because a point in one frame is too far apart from its counterpart in the other. Figure 6.8 also shows that the translation errors around 35.0% happen exactly at the higher velocities.

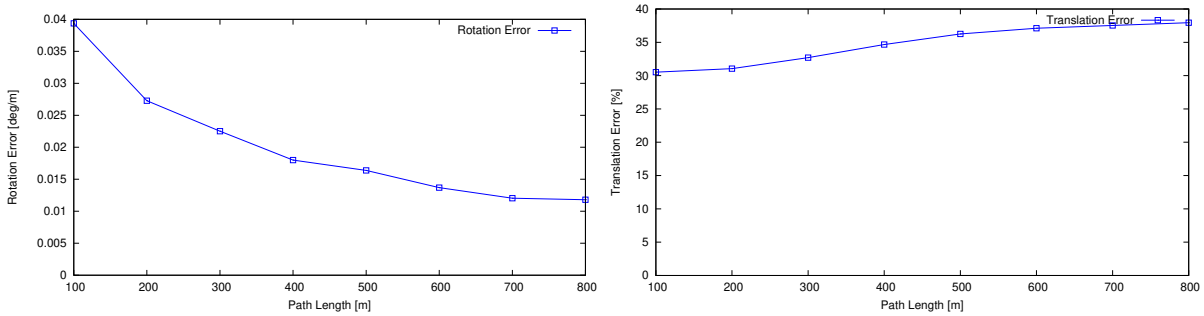


Figure 6.7: Errors by length for sequence 01.

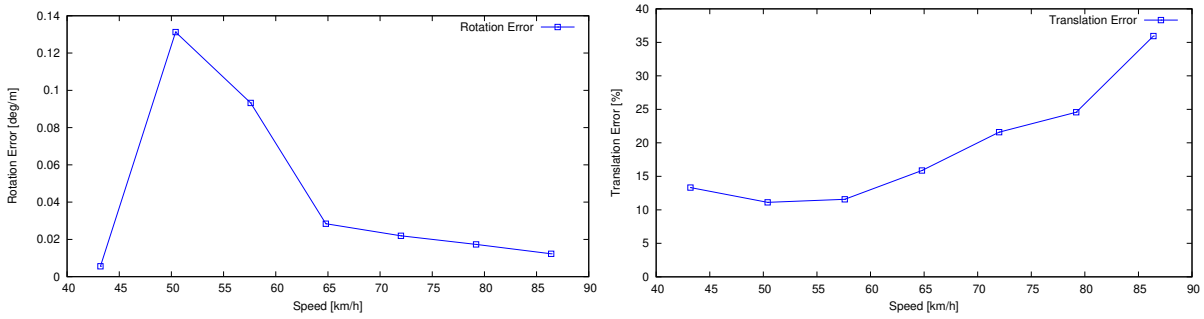


Figure 6.8: Errors by speed for sequence 01.

6.2.3 Sequence 02

Despite being the longest of all sequences and having many long soft turns and smaller but more sharp turns, the trajectory for sequence 02 (Figure 6.3c) has a global scale very close to the ground-truth and has seemingly just one overshoot problem at the third turn that deviates it from being aligned with the true trajectory. The scene contains almost just trees, bushes, and the road, which is a texture problem and can negatively influence the key-point matching step. In general, that is not a problem of this sequence, as it is rich in texture and compensates the presence of some moving vehicles along the way.

The distribution of translation error by subsequence length decreases as well as the rotation error (Figure 6.9). Unlike what happens for sequences 00 and 01, in sequence 02, the initializations are almost always good. Therefore, the global scale is also good, resulting in a low translation error.

Another factor contributing to small translation and rotation errors is the velocities lower than 50 Km/h (Figure 6.10), as it makes the key-point matching processes with fewer outliers and then the initial pose very close to the optimum.

As expected, the translation error increases with the increase in velocity because the velocities at the upper end happen in regions of the trajectory where the car accelerated. Then the error in the bundle adjustment window becomes harder to optimize.

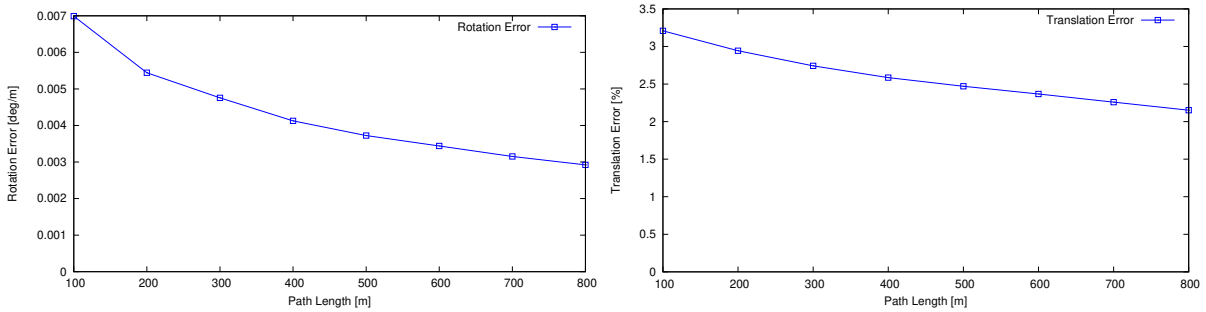


Figure 6.9: Errors by length for sequence 02.

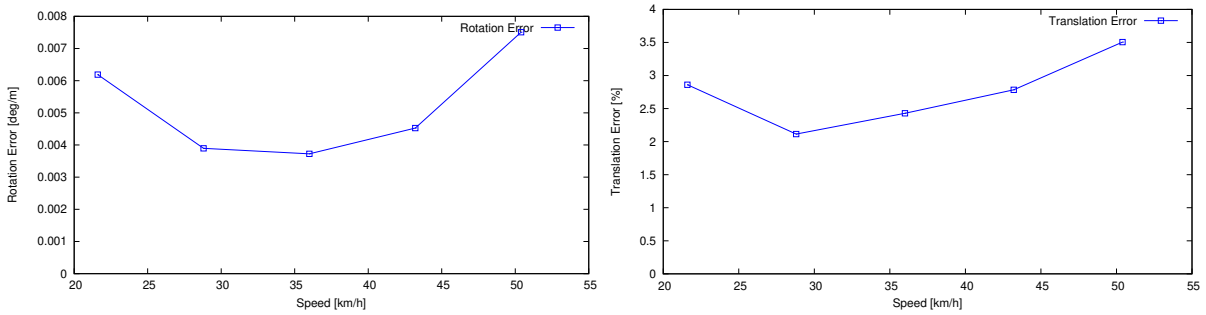


Figure 6.10: Errors by speed for sequence 02.

6.2.4 Sequence 03

The odometry for this sequence (Figure 6.3d) has the lowest rotation and translation errors among all. It is almost perfectly aligned with the ground-truth, except for a couple of locations where it seems to miscalculate the scale a little. Thus, the general aspect of the estimated trajectory is good. This is because there is only one major turn at the beginning and then three soft turns along the sequence. Also, there is no longer a straight trajectory so that the vehicle does not get to high velocities or accelerations. Indeed, the maximum velocities in this sequence are around 35 Km/h (Figure 6.12).

Concerning texture, even though there are many trees and bushes, there are also buildings and shadows in the road. Those last objects give better points for the matching step than the first ones. There is only one moving car in the scene about moving objects, but it does not interfere with the pose initializations.

The distribution of errors given path length (Figure 6.11) behaves as expected, with rotation errors being smoothed by straight trajectories and the translation errors staying

stable as most of the straight trajectories have less than 350 meters.

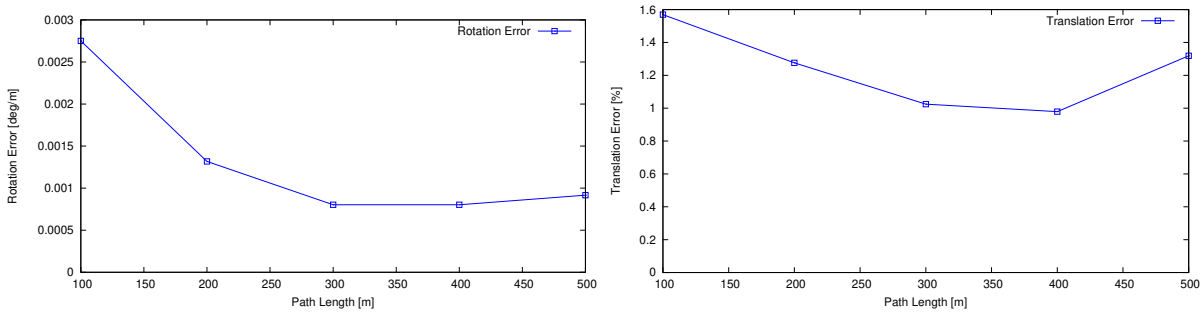


Figure 6.11: Errors by length for sequence 03.

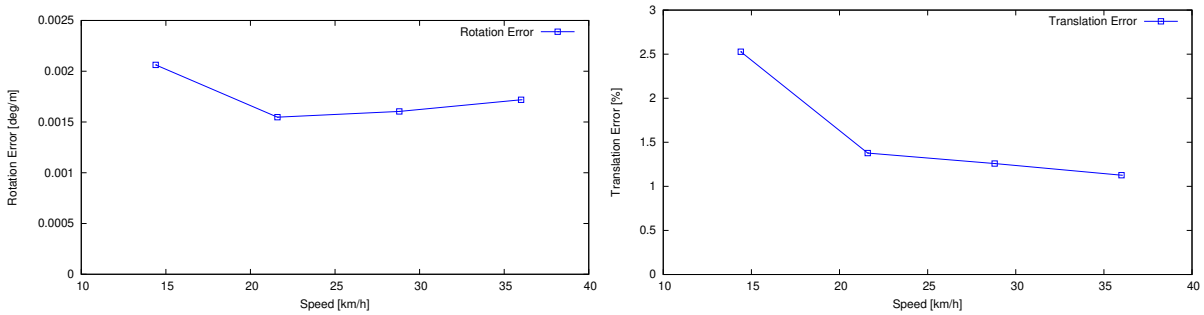


Figure 6.12: Errors by speed for sequence 03.

6.2.5 Sequence 04

Sequence 04 is the simplest of all in terms of curves, as it consists of one straight line (Figure 6.3e). It is also the smallest of all sequences. However, even though it has no considerable curves, it has, in all of the configurations **I** to **V** and in the works of Pereira et al. [46] and Song et al. [54], more significant rotation error than sequence 03, which has curves. This may be the case because the amount of static texture available at some points might not compensate all the moving vehicles in the opposite lane of the road, which results in too many outlier matches and then gives sub-optimal initializations.

The rotation and translation error distributions (Figure 6.13 and Figure 6.14) are all steady, which reflects at the beginning of the sequence. At this point, the car already has some velocity, so that the range of velocities is small even if the velocities are around 50 Km (Figure 6.14). In other words, the effect of error variation seems to be present when the higher and lower velocities are present, as we saw sequences 00 to 03.

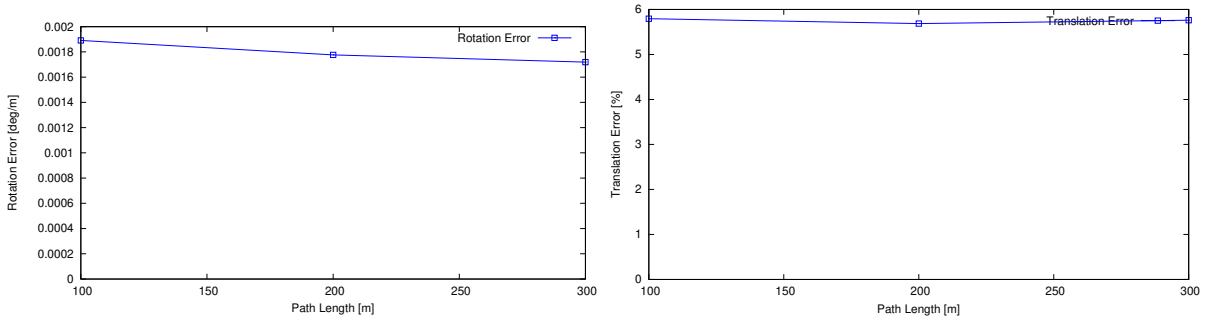


Figure 6.13: Errors by length for sequence 04.

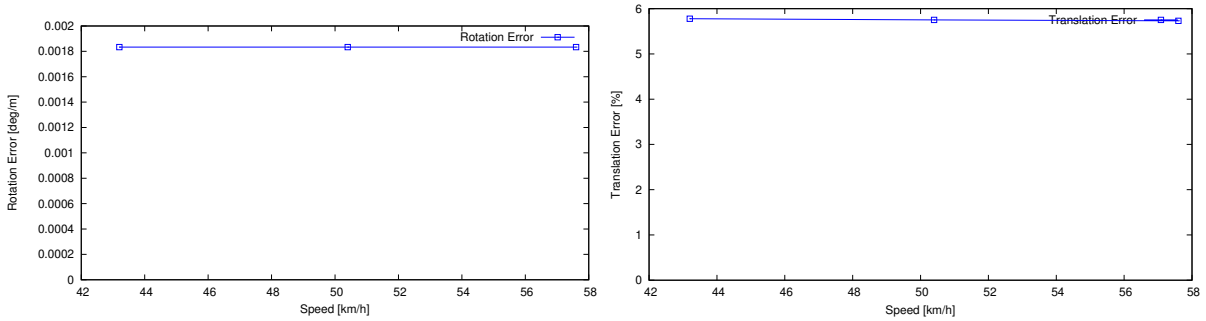


Figure 6.14: Errors by speed for sequence 04.

6.2.6 Sequence 05

The major problem that we can identify by visually inspecting the odometry for this sequence (Figure 6.3f) is the global scale at the second line from right to left, especially in the first half. This causes part of the trajectory to reflect a right-shift translation, which remains until the end of the sequence. Again, it seems a problem happens in long lines (e.g., greater than 200 m) with very soft or no rotation. However, in the case of this sequence, the translation error only happens at one location. In fact, despite having a moderate translation error, this sequence has one of the lowest rotation errors (see Table 6.2). It is not clear what causes this problem, as in that section, the scene has good texture, no moving vehicles, and the car's speed is not high.

As in most other sequences, the rotation errors are higher at low speeds (Figure 6.16) and decrease with the length of the subsequence (Figure 6.15).

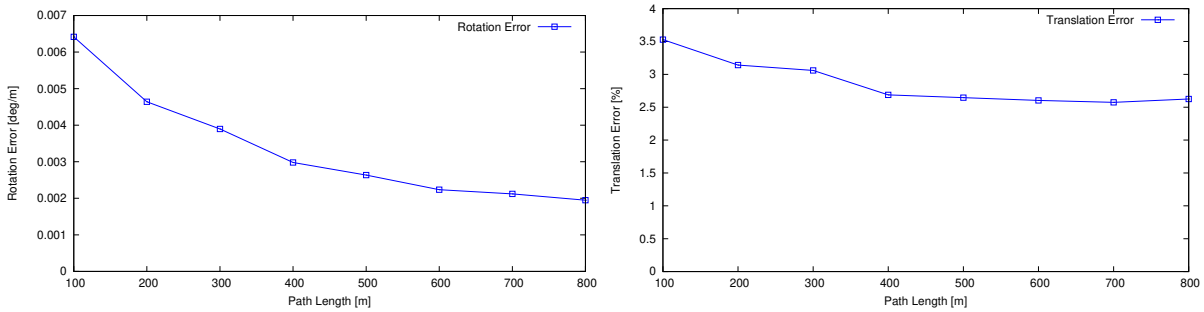


Figure 6.15: Errors by length for sequence 05.

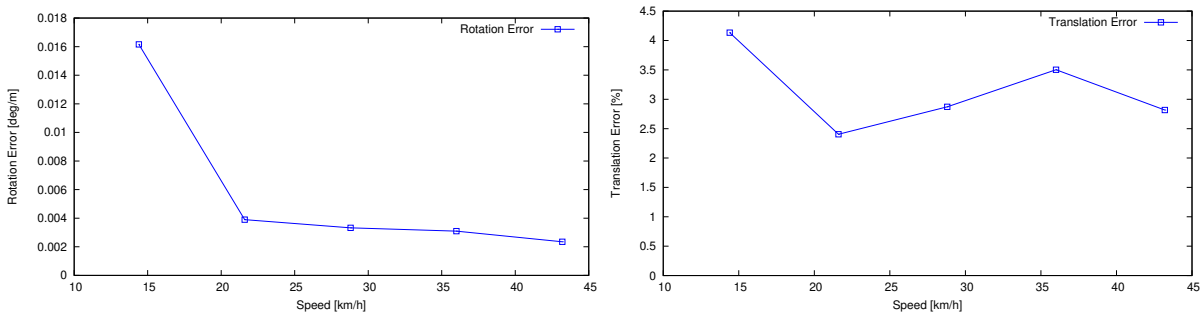


Figure 6.16: Errors by speed for sequence 05.

6.2.7 Sequence 06

Sequence 06 is an interesting case for investigating the quality of the rotation from odometry. This sequence's accumulated rotation returns to the initial orientation, as it is a closed-loop (Figure 6.4a). It has two curves, each one with 180°. The odometry is capable of making the second curve well, but the first one not so much. A visual inspection indicates that it might be due to the lack of good key-points at the first curve, as it faces many house walls with no texture but uniform color.

The appearance of scale and translation looks good, although the rotation error implies high translation error in the sequence's long term. In the case of sequence 06, the translation error would be much smaller if there was no such significant rotation error at a single location.

The high rotation and translation errors (Figure 6.17 and Figure 6.18) happen at the lower speeds because those speeds coincide with curves, including the problematic one.

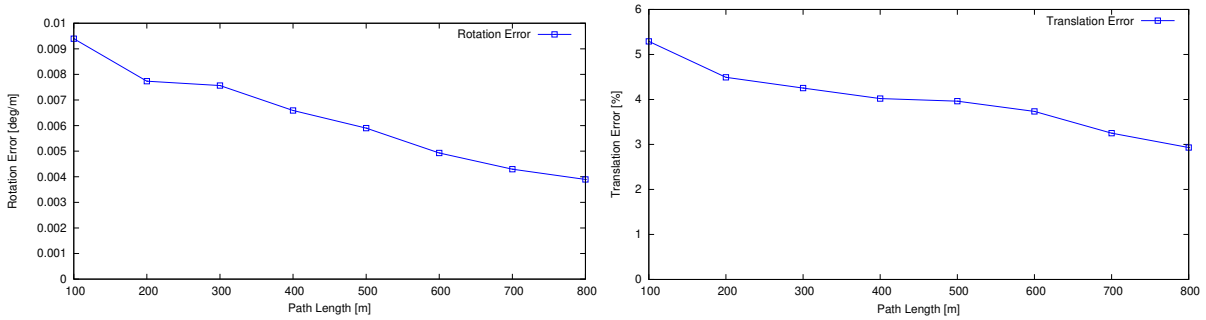


Figure 6.17: Errors by length for sequence 06.

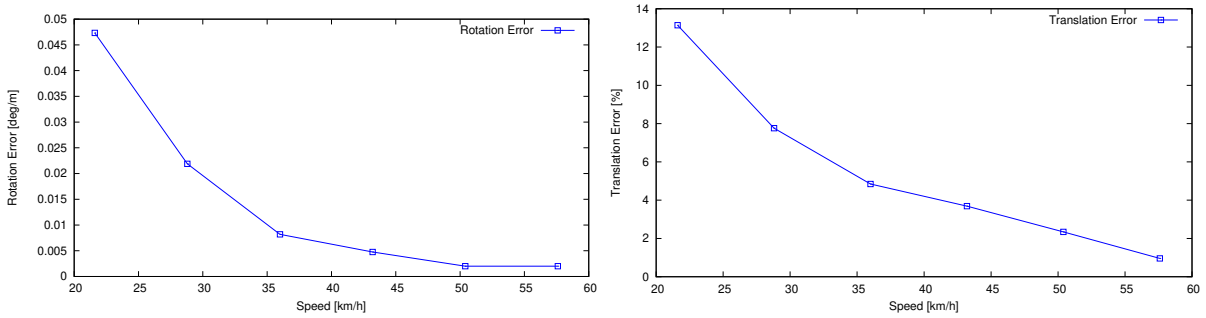


Figure 6.18: Errors by speed for sequence 06.

6.2.8 Sequence 07

The odometry for this sequence (Figure 6.4b) starts very close to the ground truth, including the entirety of the first two main road segments. Then, it suffers from scale problems in the third, fourth, and seventh segments. It also contains a major rotation error in the seventh curve, which increases the translation error from then on.

On what concerns the scale problems, at the end of the third segment, we verify some regions with very uniform color and others with a poor texture of big tree bushes. Then, there are many repeating patterns of windows of very similar houses in the fourth segment, only white walls and many bushes on the other side of the road. All those factors can result in bad key-point correspondences. The scale problem in the seventh segment is very similar to what happens in the first segment of sequence 00, as the respective roads are very similar if not the same. Again, those contain repeated patterns of windows in uniformly colored walls.

The big rotation error at the seventh curve is very probably due to many moving cars at the road intersection, where the camera car is stopped waiting for its time to proceed. It looks like the initialization of poses is difficult at regions where the car is stopped because the translation part has magnitude zero and cannot be "normalized" to a vector of norm 1.

The distribution of errors by speed (Figure 6.20) show the main tendency in other sequences: those errors are higher at the speeds in the extremes, where the lower extreme is around 20 Km/h or lower and the higher extreme is around 80 Km/h or higher. We believe this is due to deceleration in curves in the lower speeds, which make windowed optimization harder in our setup, while in the higher speeds, this is due to bad key-point matching.

The distribution of errors by length (Fig.6.19) behaves like most of the other sequences: the errors decrease with bigger subsequences as the hard parts are rarer and get smoothed.

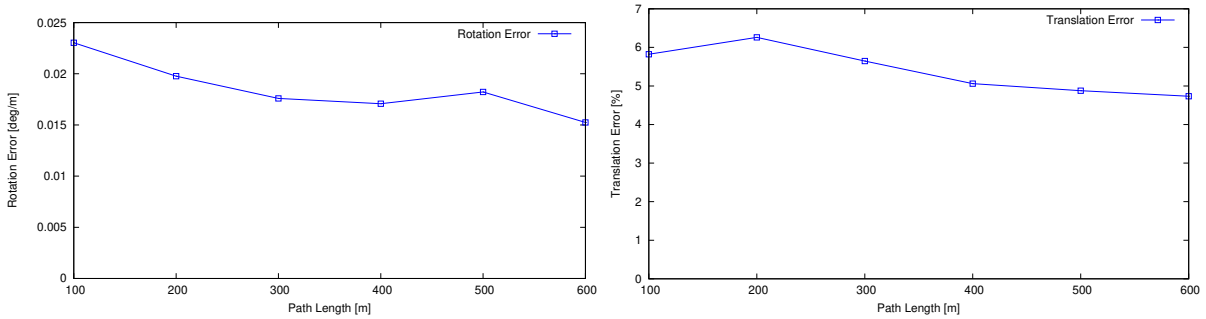


Figure 6.19: Errors by length for sequence 07.

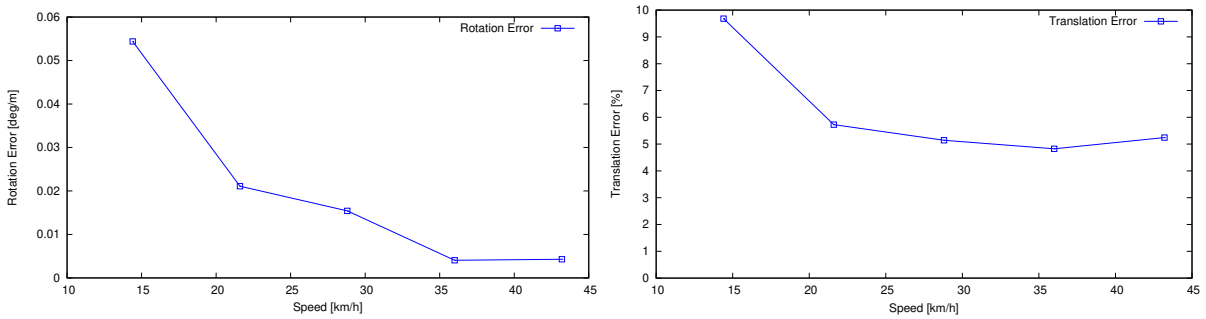


Figure 6.20: Errors by speed for sequence 07.

6.2.9 Sequence 08

In the odometry of this sequence (Figure 6.4c), the orientation seems to keep aligned with the ground truth until the end, although some small rotation errors seem to compensate each other. The visible difference between the odometry and ground-truth is a translational shift caused by scale errors, as some segments shrink when approaching some curves. Indeed, at the lower speeds close to the curves, we see that the translation errors are highest (Figure 6.22), while the rotation errors keep relatively low. The distribution of errors by subsequence length (Figure 6.21) follows the dominant trend detailed in the previous sections.

This sequence's scene problems are the same as sequence 07: a high amount of bushes and trees, some moving cars (but less than 07), repeating patterns, and uniform coloring of walls.

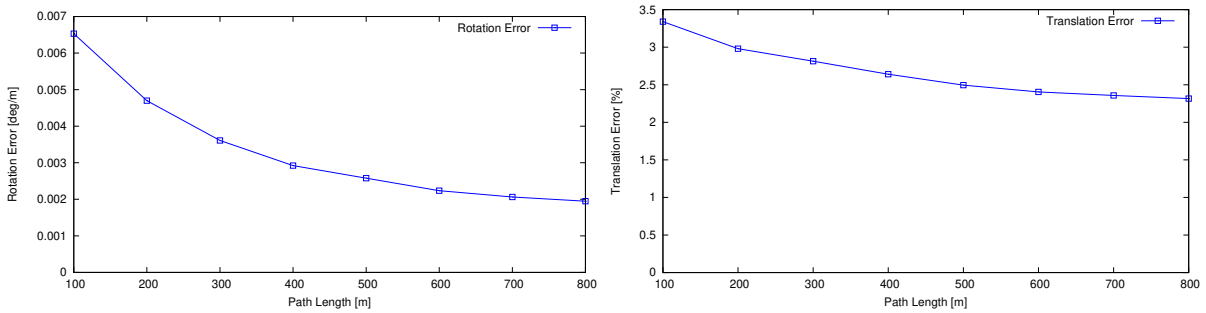


Figure 6.21: Errors by length for sequence 08.

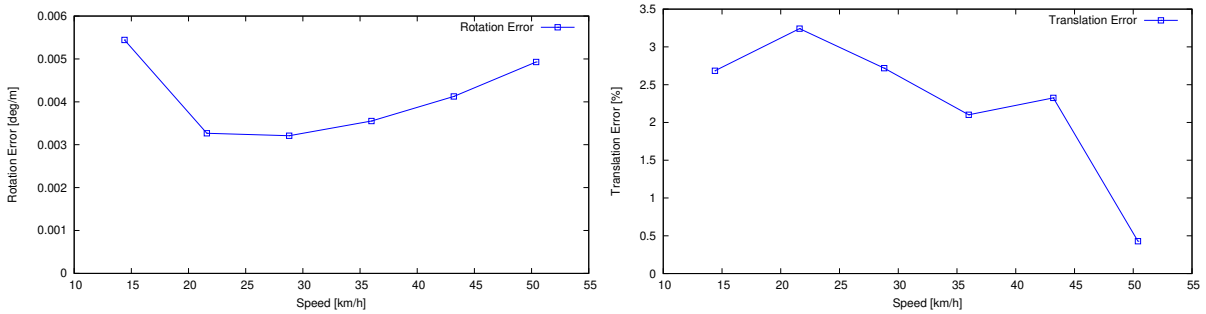


Figure 6.22: Errors by speed for sequence 08.

6.2.10 Sequence 09

This sequence is a long loop formed by soft and acute curves. Like in other sequences that have short curves close to 90° degrees or higher, the odometry (Figure 6.4d) shrinks the scale as it approaches those curves, but much less in the long soft curves. However, this time, we cannot attribute this to deceleration, as configuration **V**, which has no windowed adjustment, has precisely the same translation error than **IV** (see Table 6.2). At this point, we cannot figure out what causes the scale problems, as the sequence has a good texture for the most part, and there are just a few moving vehicles.

The orientation returns to the initial state, as in the ground-truth. The total rotation error for this sequence is one of the smallest among all sequences. The distribution of errors by subsequence length (Figure 6.23) behaves like most other sequences, but the distribution of errors by speed (Figure 6.24) shows a tendency of increasing rotation error already at speeds close to 60 Km/h, although those errors are still relatively low in all velocities of the range if we compare to other sequences.

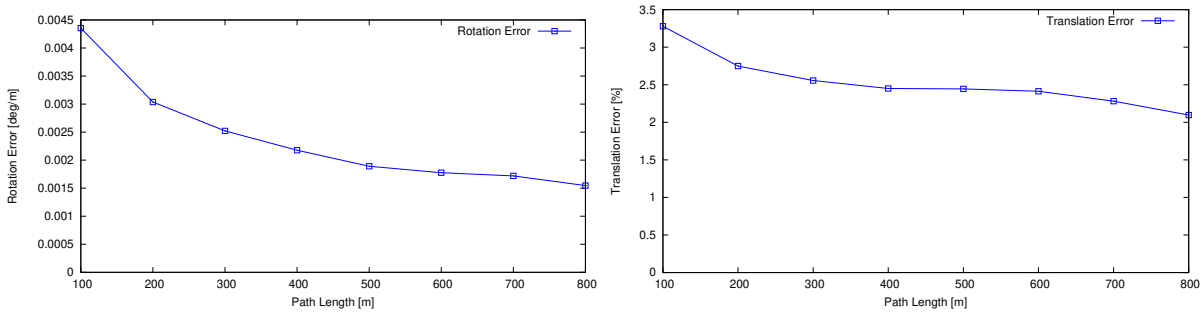


Figure 6.23: Errors by length for sequence 09.

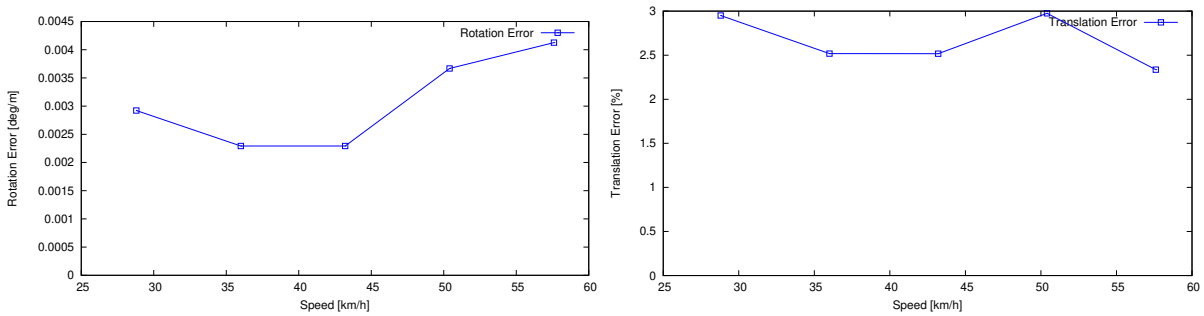


Figure 6.24: Errors by speed for sequence 09.

6.2.11 Sequence 10

This odometry for this sequence (Figure 6.4e) has no big rotation problems, but just one major scale problem at the last acute curve. Like in sequence 09, this is not clear why there is a scale problem, as the scene seems to have good texture and no moving objects to introduce initialization errors. However, there is an apparent increase in error compared with configuration **V** that may indicate the problem of the windowed adjustment in regions with high deceleration.

The distribution of errors by path length (Figure 6.25) follows the sequences' common trend. The rotation errors by speed (Figure 6.26) follow the trend of an increase next to higher speeds, in this case, when it approaches 60 Km/h. The translation errors by speed (see Figure 6.26) agree with the deceleration hypothesis, as those are relatively high when approaching 15 Km/h.

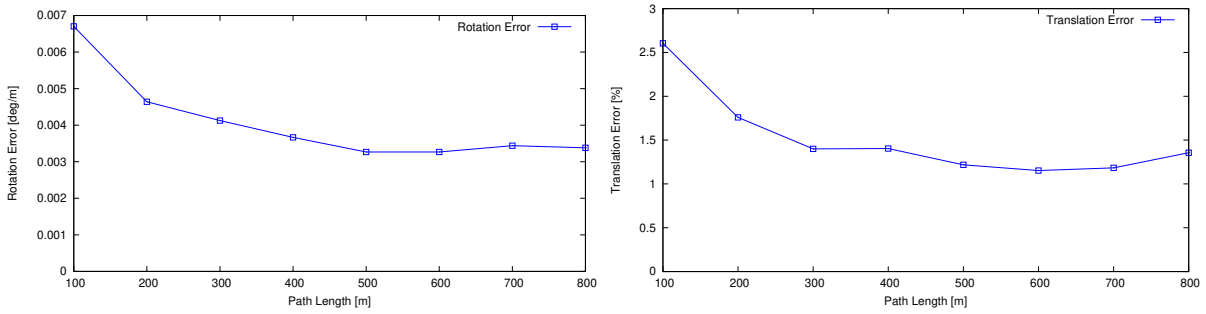


Figure 6.25: Errors by length for sequence 10.

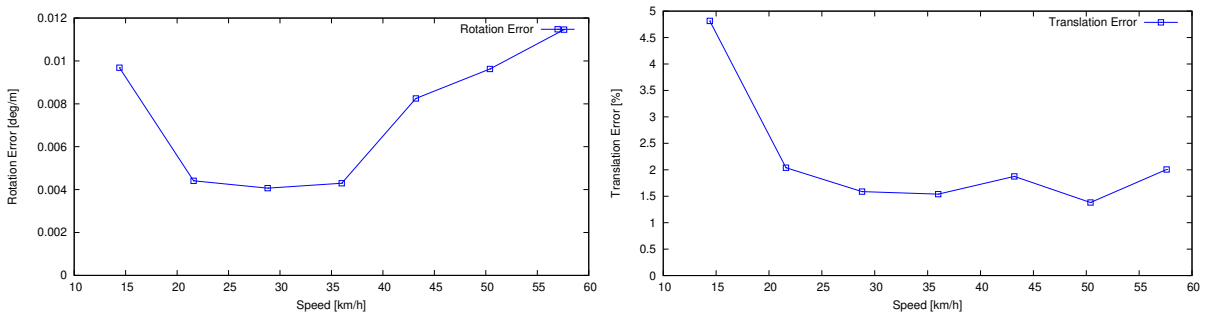


Figure 6.26: Errors by speed for sequence 10.

6.2.12 General Tendencies

We could consider sequence 01 (Figure 6.3b) to have smooth rotation, but we keep it apart because it contains much less texture than the others, the scene contains a uniformly textured road and sky. Besides that, sequence 01 also contains many more moving cars in the scene. It is the most challenging sequence for our algorithm.

As we expect from sequences 03 and 04, those with smaller rotation errors (see Table 6.2). Those sequences have just a few and soft turns - virtually none in 04. Also, there is plenty of landmarks with parallax close to the camera and on the horizon. The number of moving objects in the scene is small, too, so that the key-point selection is easy and with few outliers.

We see that indirect bundle adjustment (configurations I to IV) has similar performance to our initialization (configuration V) in sequences with soft rotation, namely 03, 04, and 09. Sequence 09 (Figure 6.4d) in particular has more acute rotations than 03 but still have its results in (IV) comparable with the initialization. We think this is the case because the initial pose and the resulting scale for this sequence are good and facilitate the optimization.

For the configurations with indirect bundle adjustment, sequences 01 and 07 are the most difficult in the rotation. As we saw in the last paragraph, this is expected from 01. In the case of sequence 07, even though it is well-textured and with few moving objects, for the most part, it contains many moving vehicles at one point. It tricks the pose initialization because at that point, the car is stopped at an intersection, and many other cars pass closely in the orthogonal street, thus having big apparent movement in the scene.

In the remaining sequences, the configuration with the biggest window (IV) is, on

average, better than the sequences with window size 3 (I and III). This indicates that there may be a configuration with a bigger window and some reprojection pattern that gives better results than the initialization, although we were not able to find it yet.

The translational error also reflects the error in scale, as the magnitude of translation from the essential matrix initialization is multiplied by a scale factor, which is computed based on the initial estimate of the pose and the relative pose between the stereo pair. The scale is not optimized afterward, so translation error is heavily related to it, even after pose optimization. The recovery of the global scale of Pereira et al. [46] is better than ours, and it does not use the stereo pair. Nevertheless, for the rotation part, our initialization looks competitive to theirs.

It is possible that, inside a reprojection window, the drift in the relative scale between frames (due to acceleration) pushes the optimization to the wrong minima, as the initial values of the poses all have translation with norm 1. Therefore, there are two options: to include the optimization scale or have a better estimate for scale and use it to re-scale initial poses. As our initial estimates of poses determine the scale, bad initialization of poses has a double impact on the result after optimization. Also, it may be the case that the compound reprojection error of a window is not robust to pairs of frames with a bad selection of key-points and bad initial poses and epipoles, even with the filter of bad reprojections we described in Algorithm 2. Therefore, to get results superior to the initialization, fixing the scale and robustness of window error are prerequisites.

6.3 Scene reconstruction

After optimizing relative pose and second epipole, we can reconstruct the scene with the same density of any given set of point matches between the corresponding pair of images. Thus, this is not limited to the set of key-points used previously, but the quality of reconstruction depends on the point matches' quality. In Figure 6.27, we compare the depth maps of ground-truth Velodyne (top) with our estimate (bottom) given the dense optical flow from Farneback (Farnebäck [14]). We restrict the optical flow to the Velodyne points that are visible for the sake of comparison. We also remove the global scale from both ground-truth and our estimate. So, in both, the furthest points have a distance equal to 1.

Because the optical flow we used is not robust to big displacements, the bigger parallax points at the lower left part of the image have worse depth prediction. For this pair of images, in particular, the pose error is low, and the epipole is not too far from the image center because of the car's smooth movement. Therefore, provided that our pose-depth-epipole relationship is correct, we can attribute most depth errors to the optical flow quality.

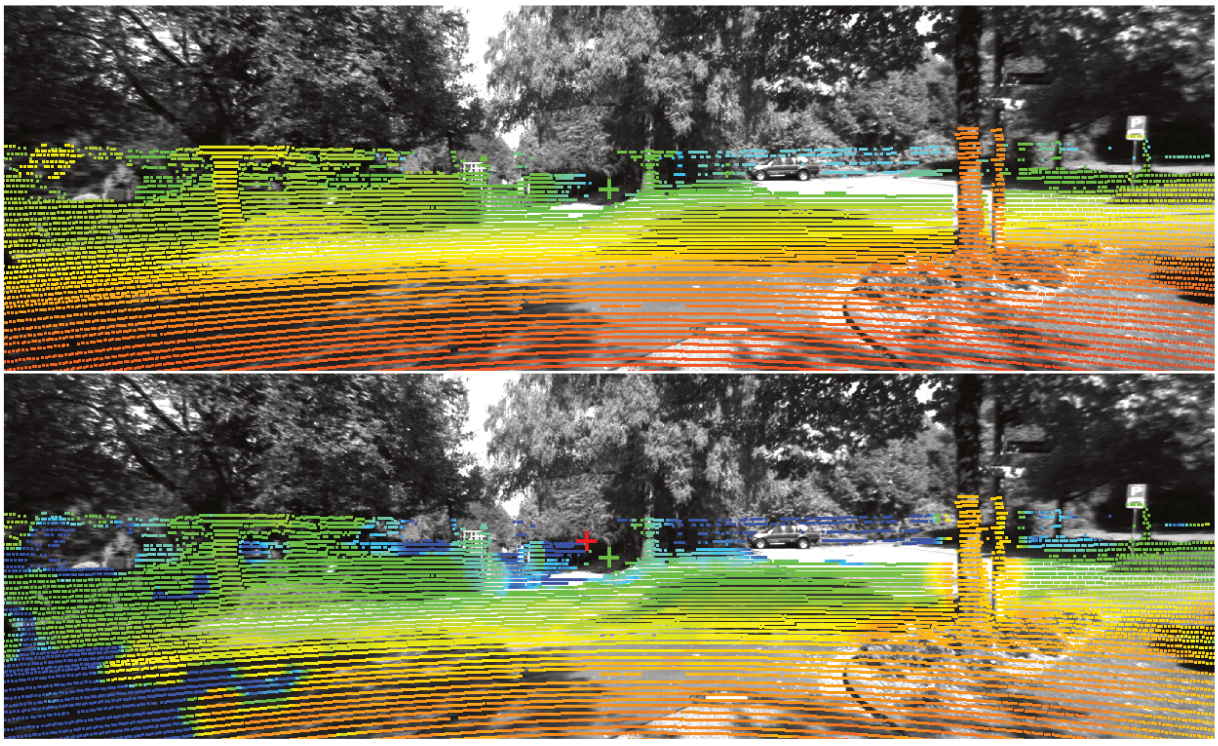


Figure 6.27: Depth map from Velodyne (top) and ours (bottom). The image center is marked with a green cross, and the epipole is marked with a red one.

Chapter 7

Conclusion and Future Work

In this work, we proposed a VO algorithm with a parametrization based on the epipolar geometry. We aimed at finding a characterization of VO that relies on a small number of parameters and that is able to adjust pose and geometry. From the work, we answered the following research questions:

RQ1 Is there a formulation of the reprojection error in terms of pose parameters plus a constant number (per frame) of parameters that also model the depth of key-points in the 3D scene?

We showed that we can write the reprojection error of image points without depending on feature parameter by using the relationship between key-point depth, camera pose, and second epipole from two-view geometry. Indeed, we were able to write the reprojection error of image points without depending on feature parameters like inverse depth.

RQ2 If so, is it compatible with windowed optimization like Bundle Adjustment?

We showed that our characterization of odometry and scene reconstruction is not limited to just two camera frames, but it extends to an arbitrary sequence of frames. Therefore, we also answered whether the reprojection error that does not depend on point parameters explicitly could be extended and optimized in a windowed way.

RQ3 Can it be implemented in a VO pipeline?

From the experiments conducted we the proposed Epipolar VO algorithm, we demonstrate that an optimization based on the proposed parametrization works in a VO pipeline. However, it needs more evaluation and more practical modifications for real-world applications.

We evaluated the algorithm's performance in a specific setup, consisting of a moderate set of hyper-parameter combinations. To make a complete investigation of the proposed parametrization (e.g., bigger windows and reprojection strides), we started a new implementation of the optimization back-end in C++ that does not make expensive type conversions between PyTorch tensors and Numpy arrays at every single iteration. It also

promises to have much less overhead in the Jacobians' computation, as it will not be made of general-purpose automatic gradients. We can also replace FAST with a point descriptor like ORB and replace Lucas-Kanade optical flow with a robust point-matching algorithm. There are also strategies for selecting key-frames and loop-closure detection that could make the algorithm more robust.

Another aspect that can be improved regards better computation of global scale, with two-fold benefits. First, the final translation error would decrease. Second, we could initialize the poses inside the reprojection window so that the magnitude of translation is close to the real value (not all equal to 1), which would make the optimization easier when there is acceleration inside the window. A stereo version of the algorithm could handle the scale problems easily.

On what concerns the type of motion, we could include indoor and drone datasets as well. Also, using IMU data could be useful in this context for more robust initialization of pose (e.g., identifying bad rotation, checking the local consistency of scale). Another possibility is investigating the recovery pose from the 3D laser points in the context of two-view geometry and bundle adjustment, without the feature extraction step. Finally, we could also test our proposed algorithm with a direct approach.

Bibliography

[1] M. Aladem and S. Rawashdeh. Lightweight visual odometry for autonomous mobile robots. *Sensors*, 18:2837, 08 2018. doi: 10.3390/s18092837.

[2] A. Bain and D. Crisan. *Fundamentals of Stochastic Filtering*. Stochastic Modelling and Applied Probability. ISBN 978-0-387-76895-3. doi: 10.1007/978-0-387-76896-0.

[3] J. L. Blanco. A tutorial on se(3) transformation parameterizations and on-manifold optimization. 09 2010.

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[5] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016. doi: 10.1177/0278364915620033. URL `https://doi.org/10.1177/0278364915620033`.

[6] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16, 02 2003. doi: 10.1137/0916069.

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. volume 6314, pages 778–792, 09 2010. ISBN 978-3-642-15560-4. doi: 10.1007/978-3-642-15561-1_56.

[8] K. Celik, S. Chung, M. Clausman, and A. K. Somani. Monocular vision slam for indoor aerial vehicles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1566–1573, 10 2009. doi: 10.1109/IROS.2009.5354050.

[9] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29:2007, 2007.

[10] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10605-2.

[11] J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942, 9 2015. doi: 10.1109/IROS.2015.7353631.

[12] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry, 2016.

[13] J. Engel, V. C. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. volume abs/1607.02555, 2016. URL `http://arxiv.org/abs/1607.02555`.

[14] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. volume 2749, pages 363–370, 06 2003. ISBN 978-3-540-40601-3. doi: 10.1007/3-540-45103-X_50.

[15] M. Ferrera, J. Moras, P. Trouvé-Peloux, and V. Creuze. Real-time monocular visual odometry for turbid and dynamic underwater environments. 06 2018.

[16] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `https://doi.org/10.1145/358669.358692`.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 5 2014. doi: 10.1109/ICRA.2014.6906584.

[18] X. Gao, R. Wang, N. Demmel, and D. Cremers. Ldso: Direct sparse odometry with loop closure. *CoRR*, abs/1808.01111, 2018.

[19] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 963 – 968, 07 2011.

[20] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: the kitti dataset. *The International Journal of Robotics Research*, 32:1231–1237, 09 2013. doi: 10.1177/0278364913491297.

[21] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531, 2014.

[22] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[23] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[24] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[26] S. Jha and V. Raman. Automated synthesis of safe autonomous vehicle control under perception uncertainty. In *Proceedings of the 8th International Symposium on NASA Formal Methods - Volume 9690*, NFM 2016, pages 117–132, New York, NY, USA, 2016. Springer-Verlag New York, Inc. ISBN 978-3-319-40647-3. doi: 10.1007/978-3-319-40648-0_10. URL http://dx.doi.org/10.1007/978-3-319-40648-0_10.

[27] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952. doi: 10.1214/aoms/1177729392. URL https://doi.org/10.1214/aoms/1177729392.

[28] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[29] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. pages 3607 – 3613, 06 2011. doi: 10.1109/ICRA.2011.5979949.

[30] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248, 10 2016. doi: 10.1109/3DV.2016.32.

[31] Y. Latif, C. Cadena, and J. Neira. Robust loop closing over time for pose graph slam. *The International Journal of Robotics Research*, 32(14):1611–1626, 2013. doi: 10.1177/0278364913498910. URL https://doi.org/10.1177/0278364913498910.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[33] H. C. Longuet-Higgins. *A Computer Algorithm for Reconstructing a Scene from Two Projections*, page 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. ISBN 0934613338.

[34] S. Lovegrove, A. Patron-Perez, and G. Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *BMVC*. BMVA Press, 2013.

[35] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004. doi: 10.1023/B:VISI.0000029664.99615.94.

[36] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). volume 81, 04 1981.

[37] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2002.

[38] H. Moravec. Towards automatic visual obstacle avoidance. In *IJCAI*, 1977.

[39] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Carnegie Mellon University, Pittsburgh, PA, September 1980.

[40] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[41] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:756–770, 2004.

[42] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004.

[43] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2262–2269, 2006.

[44] E. Parisotto, D. S. Chaplot, J. Zhang, and R. Salakhutdinov. Global pose estimation with an attention-based recurrent network. *CoRR*, abs/1802.06857, 2018. URL `http://arxiv.org/abs/1802.06857`.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[46] F. I. Pereira, G. Ilha, J. Luft, M. Negreiros, and A. Susin. Monocular visual odometry with cyclic estimation. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 1–6, 2017.

[47] V. Rengarajan, Y. Balaji, and A. N. Rajagopalan. Unrolling the shutter: CNN to correct motion distortions. In *CVPR*, pages 2345–2353. IEEE Computer Society, 2017.

[48] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 11 2011. doi: 10.1109/ICCV.2011.6126544.

[49] D. Scaramuzza. Performance evaluation of 1-point-ransac visual odometry. *J. Field Robotics*, 28:792–811, 09 2011. doi: 10.1002/rob.20411.

[50] D. Schubert, V. Usenko, N. Demmel, J. Stueckler, and D. Cremers. Direct sparse odometry with rolling shutter. In *European Conference on Computer Vision (ECCV)*, Sept. 2018. accepted as oral presentation, to appear.

[51] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision, ECCV 2012 - 12th European Conference on Computer Vision, Proceedings*, volume 7576 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 746–760, 2012. ISBN 9783642337147. doi: 10.1007/978-3-642-33715-4_54.

[52] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

[53] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. volume 28, pages 595–599, Thousand Oaks, CA, USA, 5 2009. Sage Publications, Inc. doi: 10.1177/0278364909103911. URL `http://dx.doi.org/10.1177/0278364909103911`.

[54] S. Song, M. Chandraker, and C. C. Guest. High accuracy monocular sfm and scale correction for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):730–743, 2016.

[55] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 10 2012. doi: 10.1109/IROS.2012.6385773.

[56] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph slam. pages 1879–1884, 10 2012. ISBN 978-1-4673-1737-5. doi: 10.1109/IROS.2012.6385590.

[57] M. Trajkovic and M. Hedley. Fast corner detection. *Image and Vision Computing*, 16:75–87, 02 1998. doi: 10.1016/S0262-8856(97)00056-5.

[58] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, 5 2017. doi: 10.1109/ICRA.2017.7989236.