TÉCNICO LISBOA

Visual Inertial Odometry based on Image Lines

Duarte Nuno Soares Correia

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor

Professor José António da Cruz Pinto Gaspar

Examination Committee

Chairperson: Professor João Fernando Cardoso Silva Sequeira
Supervisor: Professor José António da Cruz Pinto Gaspar
Member: Professor Pedro Daniel dos Santos Miraldo

December 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Firstly, I want to thank my supervisor, Prof. José Gaspar, for guiding me through the journey of writing this thesis.

I want to praise a colleague and great friend, João Pinto, who has helped me throughout my MSc of Electrical and Computer Engineering journey. I also want to acknowledge José Gomes and André Nogueira for their help in writing this thesis.

I want to thank all the people from IST that helped me through these past years, not only by showing their continued support but also by providing a way of escaping the academic environment; which names are listed in alphabetical order: António Capela, Bruno Bento, Bruno Silva, Clara Pereira, David Neto, Diogo Pires, Diogo Silva, Diogo Valada, Francisco Faro, Francisco Gomes, Glória Pereira, Guilherme Soares, Isabel Raposo, João Cunha, Luís Franco, Pedro Ferreira, Rodrigo Almeida, Rodrigo Câmara, Simão Correia, Vasco Martins and Victor Negîrneac.

I want to thank some of my long-time friends who stayed with me and always provided a place where I could freely discuss and helped me to not deviate from my studies, in particular, Bernardo Brito, Daniel Santos, Eduardo Correia, Jorge Almeida and Tiago Baptista.

I want to express my appreciation to my grandmothers, Maria and Luísa, who were always eager to receive and listen to my problems. I'd like to also mention my aunt, Anabela and uncle, António, by always keeping in touch even from a long distance. Last, but most certainly not least, I want to express my profound gratitude towards my family, in particular, my sisters, Sofia and Beatriz which helped me to always keep smiling, and my parents, Laura and Duarte, who always believed in me with constant support and provided me with the means to make my life possible.

Resumo

Este trabalho propõe fornecer os blocos necessários para uma solução de Localização e Mapeamento Simultâneo usando técnicas de Odometria Inercial Visual com Linhas como características. Motivado por avanços na aprendizagem automática que facilitam a detecção de linhas em imagens fornecendo uma ferramenta de visualização do espaço mais rica do que os pontos clássicos. O principal obstáculo encontrado foi na correspondência de linhas através de múltiplas imagens para posterior inicialização como estruturas 3D. Foi introduzida uma variação do algoritmo Iterative Closest Point (ICP) para abordar este problema, onde o foco principal é a correspondência das características em vez de encontrar a transformação entre diferentes vistas. São introduzidos e analisados dois métodos de inicialização de linhas a partir de imagens 2D, um que utiliza coordenadas Plucker para representar linhas e outro que minimiza a distância 3D de um ponto geral aos raios retroprojectados. É proposto o uso de um filtro de Square Root Unscented Kalman Filter com base em Grupos de Lie para o processo de filtragem, que integra medições de uma Unidade de Medição Inercial e informação visual. A abordagem proposta proporcionou uma localização rigorosa do drone através de uma trajectória complexa e forneceu inicializações de linha que mantêm o filtro consistente. Foi demonstrado que as linhas fornecem um descritor rico de um ambiente em comparação com uma nuvem de pontos e que os algoritmos baseados em linhas propostos podem dar uma primeira aproximação do mapa sem a necessidade de processos de filtragem nos casos em que a localização é fornecida.

Palavras chave: Odometria Inercial Visual, Localização e Mapeamento Simultâneo, Features de linhas, Square Root Unscented Kalman Filter, Grupos de Lie

Abstract

This work proposes building blocks for Simultaneous Localization and Mapping (SLAM) solution based on Visual Inertial Odometry (VIO) techniques with Lines as features. It is motivated by advances in machine learning that facilitate the detection of lines in images that provide a richer mapping visualization tool than classical point features. The main building block, proposed in this work, is the matching of lines through multiple frames to be initialized as 3D structures. An Iterative Closest Point (ICP) variation is introduced as the principal approach, where the main focus of the algorithm is to match the features instead of finding the transformation between different views. Two methods are introduced and analyzed for line initialization from 2D images, one that utilizes Plucker coordinates to represent the lines and another that minimizes 3D the distance of a general point to back-projected rays. A Square Root Unscented Kalman Filter (SR-UKF) is proposed based on Lie Groups as a base for the filtering process, which can integrate measurements from an Inertial Measurement Unit (IMU) and visual information. As a proof of concept, the proposes approach provided a sound localization of a drone through a complex trajectory and line initializations that maintain the filter consistent. It is experimentally shown that lines provide a rich descriptor of an environment compared to a point cloud and that the line algorithms proposed can give a first approximation of the map without the need for filtering processes in cases where the localization is provided.

Keywords: Visual Inertial Odometry, Simultaneous Localization and Mapping, Line Features, Square Root Unscented Kalman Filter, Lie Groups

Contents

List of Figures

List of Tables

Nomenclature

$\mathbb{R}$	Set of real numbers
$\mathbb{R}^N$	Real vector space of dimension N
G	General Lie Group
$\mathrm{SO}(3)$	Special Orthogonal Group of dimension 3
$\mathfrak{so}(3)$	Lie Algebra associated with $SO(3)$
$\mathrm{SE}(3)$	Special Euclidean Group of dimension 3
$\mathfrak{se}(3)$	Lie Algebra associated with $SE(3)$
$SE_{2+p}(3)$	Special Euclidean Group used in this work
$\mathfrak{se}_{2+p}(3)$	Lie Algebra associated with $SE_{2+p}(3)$
$\mathsf{A}, \mathsf{B}, \mathsf{R}$	Matrices use sans-serif fonts
$\mathbf{x}, \mathbf{b}, \mathbf{p}$	Vectors are typeset using bold fonts
k, i, j	Scalars are in italics
$\sim$	Represents equality up to a scale
$[\mathbf{a}]_\times$	Represents the Skew-symmetric matrix of vector $\mathbf{a}$

Chapter 1

Introduction

Visual Odometry systems have been used successfully to provide robot's pose estimations for over 16 years, with inertial sensors being added in recent years. One approach to visual odometry is to solve a Simultaneous Localization and Mapping (SLAM) problem by employing Kalman filters to propagate 3D points as features. Since visual odometry should be run online, the system cannot create detailed maps of the environment traversed using only points. A solution to this problem comes by using lines as features since they provide a rich map of the environment, even though they can be described from only six parameters. The main cost of updating a state through a filter is usually bound to the dimension of the observation. Consequently, lines could reduce the execution time of filters, given that they can provide more visual information than a single point and provide a richer estimation of the robot's pose while using a smaller number of features.

One of the main challenges in using lines as features has been the difficulty in line detection using classical methods. With the appearance of neural networks trained explicitly for line detection in images, this process has become faster, more reliable and consistent. Consequently, methods that take advantage of these new detectors should be explored.

Another weakness of using lines is that usually, they do not provide good descriptors that make it challenging to match features across multiple images so that mapping the environment is possible. With this problem in mind, line matching methods should be researched.

1.1 Related Work

One important precedent of SLAM is the Structure from Motion (SfM) approach, which has been studied and used with impressive results for the past three decades [Hartley & Zisserman, 2000]. Monocular Simultaneous Localization and Mapping (MonoSLAM [Davison, 2003]) was the first approach in which a real-time filter achieved good results, which leverages an Extended Kalman filter (EKF) [Kalman, 1960a] to propagate the state. New parameterizations were proposed to worth features in the infinite in Inverse Depth Parameterization [Montiel et al., 2006]. The Unscented Kalman Filter [Wan & Van Der Merwe, 2000], UKF, was introduced as an alternative to the EKF, leveraging the Unscented transform [Julier & Uhlmann, 1997] to better propagate the state and Covariance without the need for the linearization of the system. Since the UKF increased the computational complexity to the filter, a Square Root-UKF (SR-UKF [Van der Merwe & Wan, 2001]) proposes propagating the

squared root of the Covariance, reducing the complexity closer to the one of the EKF. Novel approaches like [Brossard et al., 2017] take advantage of Lie Groups and algebra which present a natural state representation and permit Inertial Measurement Units (IMUs) for state estimation.

SLAM variations that use Lines as features were first presented in [Smith et al., 2006], where an EKF filter is used and is introduced an initialization method for lines that propagates plane intersections until their endpoints can represent the line; until then, lines are propagated using an exterior EKF filter. In [Solà et al., 2011], different approaches of landmark parameterizations that can be used in EKF-SLAM with points and lines are shown. In [Perdices et al., 2014], a UKF is used to propagate 6-dimension Plücker Coordinates directly and proposes a transformation of the 2D lines to a subset of $\mathbb{R}^3$ paired with a representative patch of the line for matching purposes. However, these line approaches do not use inertial measurements to update the state. As such, the mapping and localization have an inherent scale to the actual map associated with them.

1.2 Problem Formulation

This thesis aims to provide the necessary building blocks to implement a Simultaneous Localization and Mapping (SLAM) solution using Visual Inertial Odometry with lines as features. It is proposed to take advantage of a line detection neural network to provide feature detection. For matching, it is proposed a variation the Iterative Closest Point (ICP) where the objective is not to obtain the transformation between the points, but the line matches between two sets of lines.

A method that performs line initializations from the obtained matches and benefits from using Plucker line coordinates is analyzed. It is also introduced an initialization technique based on the 3D distance to back-projected rays.

At last, the filtering strategy takes advantage of the lines obtained and inertial information given by an Inertial Measurement Unit (IMU) to estimate a drone's pose and a map of the environment using a Square Root Unscented Kalman Filter based on Lie Groups.

1.3 Thesis Structure

Chapter 1 introduces the problem to approach in the thesis, in particular presents a short discussion on lines as features of Visual Inertial Odometry systems and a revision of the literature that used filtering processes and used lines as features. Chapter 2 presents the mathematical and engineering background needed to understand the concepts being introduced and/or used in the context of this thesis. Chapter 3 introduces the problem of Visual Inertial Odometry and provides a proposed approach to that problem. Chapter 4 provides an overview of the different methods needed to use lines as features on the proposed approach. Chapter 5 provides an overview of the different experiments executed as well as the results attained and a critical analysis of these results. Chapter 6 summarizes the work performed and highlights the main achievements in this work. Moreover, this chapter proposes further work to extend the activities described in this document.

Chapter 2

Background Theory

This chapter provides an overview of the necessary mathematical basis for the development of this thesis. More in detail, Section 2.1 gives an overview of the pinhole camera model. Section 2.2 gives an introduction to line detection and a classic approach employing the Hough Transformation. Section 2.3 describes the different features and cost functions used to match lines across images. Section 2.4 introduces the Plucker Coordinates representation for lines. Finally, Section 2.5 introduces the Unscented Kalman Filter (UKF) and the Square-Root UKF variation.

2.1 Pinhole Camera Model

The pinhole camera model describes the mathematical relation between the 3D coordinates of a point in space and its 2D projection on an image. It serves as a first-order approximation of the mapping from a 3D scene to a 2D image since it assumes no lenses or radial distortion.

The reference frame is set on the camera using its camera center of projection $\mathbf{C}$ as its origin in the pinhole model. The image frame is orthogonal to the Z-axis direction of the frame at a focal distance, f, the distance between the lens and the image sensor when the subject is in focus.

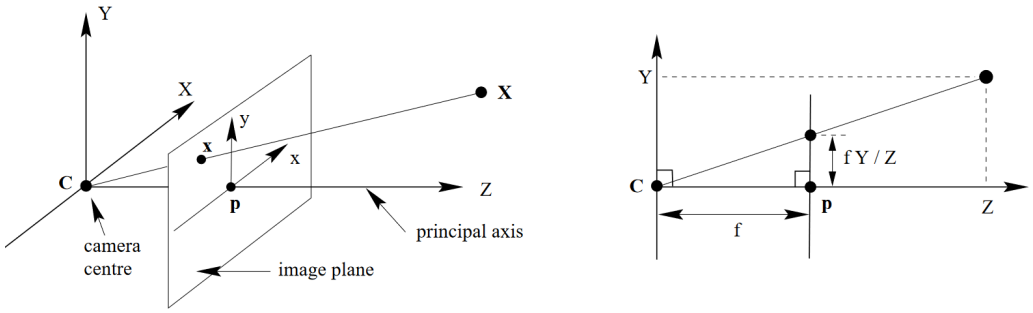


Figure 2.1: ***Pinhole Camera Geometry.*** $\mathbf{C}$ *is the camera center and* $\mathbf{p}$ *the principal point. The camera center is here placed at the coordinate origin. Note the image plane is placed in front of the camera center. From [Hartley & Zisserman, 2000]*

Using the geometric properties of the frames, the normalized pinhole image projection is given by:

$$\mathbf{x_n} = \begin{bmatrix} x \\ y \end{bmatrix} = f \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} \tag{2.1}$$

From [Heikkila & Silven, 1997] comes that after including lens distortion, the new normalized point coordinate $\mathbf{x_d}$ can then be calculated as follows:

$$\mathbf{x_d} = \begin{bmatrix} \mathbf{x_{d1}} \\ \mathbf{x_{d2}} \end{bmatrix} = (1 + k_{c_1} r^2 + k_{c_2} r^4 + k_{c_5} r^6)\mathbf{x_n} + \mathrm{d}\mathbf{x} \quad , \tag{2.2}$$

where $r^2 = x^2 + y^2$ and $\mathrm{d}\mathbf{x}$ is the tangential distortion vector:

$$\mathrm{d}\mathbf{x} = \begin{bmatrix} 2k_{c_3} xy + k_{c_4}(r^2 + 2x^2) \\ k_{c_4} xy + k_{c_3}(r^2 + 2y^2) \end{bmatrix}. \tag{2.3}$$

Finally, the points can be converted into homogenous pixel coordinates in the image plane using the matrix known as the intrinsic camera matrix K:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathsf{K} \begin{bmatrix} \mathbf{x_{d1}} \\ \mathbf{x_{d2}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{f_{c1}} & \alpha_c \mathbf{f_{c1}} & \mathbf{cc}_1 \\ 0 & \mathbf{f_{c2}} & \mathbf{cc}_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x_{d1}} \\ \mathbf{x_{d2}} \\ 1 \end{bmatrix} \quad , \tag{2.4}$$

where $\mathbf{f_c}$ are the focal distances (usually expressed in mm), $\mathbf{cc}$ are the coordinates of the principal point of the camera, and $s = \alpha_c \mathbf{f_{c1}}$ encodes the skew of the angle between the x and y sensor axes, in digital cameras it is usually 0.

Since the camera is not usually centered at the origin of the world frame, points need to be mapped to the camera frame. Supposing a general camera rotation, $\mathsf{R} \in \mathrm{SO}(3)$, and translation, $\mathbf{t} \in \mathbb{R}^{3\times 1}$, the full projection matrix becomes

$$\mathsf{P} = \mathsf{K} \begin{bmatrix} \mathsf{R} & \mathbf{t} \end{bmatrix} \in \mathbb{R}^{3\times 4}. \tag{2.5}$$

Given a point $\mathbf{x}$ in an image, the 3-dimensional points that map to that point constitute a ray in space that passes through the camera center. The ray can be represented as the connection of two points, one being the camera center and the other $\mathsf{P}^+\mathbf{x}$, where P^+ is the pseudo-inverse of P. The line can then be formally described as

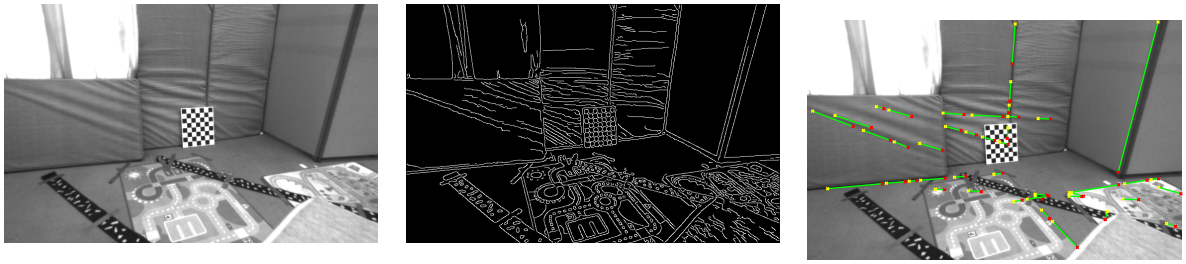
$$\mathbf{X}(\lambda) = \mathsf{P}^+\mathbf{x} + \lambda \mathbf{c} \quad . \tag{2.6}$$

2.2 Line Detection, Hough Transform

One of the most critical aspects of computer vision and image processing is to detect image features that should be distinct from others so that matching between images is possible. Since the type of features used in this thesis is lines, detecting them in images is crucial.

A classical way of detecting lines on an image is by using the Hough transformation [Hough, 1959]. This approach takes advantage of an edge detector like the Canny edge detector ([Canny, 1986]) to select points to be mapped to Hough Space. Then an accumulator bin is created that records the number of possible lines that may

pass through a single transformed point. The most elementary way to detect a line is to interpret all values above a threshold set on the accumulator as a line. The lines must then be intersected with edge points in order to create line segments.



(a) *Original Image* **(b)** *Edge detection using Canny edge detector* **(c)** *Lines detected using Hough Transformation*

Figure 2.2: *Example of lines detected using the Hough Transformation.*

Even though this method provides some line features, as shown in Figure 2.2, they are insufficient to maintain feature tracking across multiple images since the method has difficulty reliably identifying the same lines and, in the cases where it can recognize the same endpoints. As such, a neural network will be employed to detect lines; this approach will be more detailed explained in Section 4.1.

2.3 Line Matching

Since the final objective is to have a 3D reconstruction of the world, it is vital to have correct matches between lines over different camera frames. Lines will also be matched to reprojections of already seen lines to improve the estimations of those features.

SIFT (Scale Invariant Feature Transform) **Features and Descriptors** [Lowe, 1999] are regularly used to match points since they provide an attractive way to select relevant points and a 128 dimension vector descriptor per feature, which simplifies the matching process.

Since lines are not selected with this descriptor in mind, they often do not provide good features in their endpoints, and as such, they should not be used as a sole parameter to perform line matching. They can, however, be used to ensure match validity to some degree.

Sum of Squared Distances (SSD) is a naive approach where the cost function is simply the squared distance of the endpoints of the line between the frames. This method expects a high image frequency or a slow-moving drone, so that lines have limited movement between poses keeping them in similar pixel positions.

Using only SSD proved unsuccessful to match lines but gave an initialization to the Procrustes Orthogonal problem [Schönemann, 1966], which can then calculate a transformation between both sets of matched lines. This transformation can then be applied to all lines making them closer to other lines to be matched, providing that the first set of matches is correct. Iteratively repeating this method is known as the **Iterative Closest Point (ICP)** first proposed by [Chen & Medioni, 1992].

2.4 Plücker Coordinates

Introduced by Julius Plücker in the 19th century Plücker coordinates present a convenient representation for directed lines in 5-dimensional projection space. The work [Bartoli & Sturm, 2005] is used in this thesis as a guide for the Plücker conventions. Given two 3D points $\mathbf{M}^T \sim (\bar{\mathbf{M}}^T|m)$ and $\mathbf{N}^T \sim (\bar{\mathbf{N}}^T|n)$, where $\mathbf{A}^T$ are the homogeneous coordinates of the points $\mathbf{A}$, $\bar{\mathbf{A}}^T$ is the inhomogeneous part and a is the homogeneous part, one can represent the line joining them by a homogeneous "Plücker" 6-vector $\mathbf{L^T} \sim (\mathbf{a^T}|\mathbf{b^T})$ with 5 degrees of freedom (DoF), where:

$$\begin{cases} \mathbf{a} = \quad \bar{\mathbf{M}} \times \bar{\mathbf{N}} \\ \mathbf{b} = m\bar{\mathbf{N}} - n\bar{\mathbf{M}} \end{cases} \tag{2.7}$$

To reduce to 4 degrees of freedom, the same of 3D lines, a bilinear constraint must be satisfied by the 6-vector. For the definition chosen, the constraint comes as

$$\mathcal{C}(\mathbf{L}) = 0 \qquad \text{where} \qquad \mathcal{C}(\mathbf{L}) = \mathbf{a^T b} \quad . \tag{2.8}$$

Given a standard (3×4) perspective projection matrix $\mathsf{P} \sim (\bar{\mathsf{P}}|p)$, as defined in Equation 2.5, a (3×6) matrix projecting Plücker line coordinates presented in [Bartoli & Sturm, 2005] is given by

$$\tilde{\mathsf{P}} \sim (\det(\bar{\mathsf{P}})\bar{\mathsf{P}}^{-T}|[\mathbf{p}]_x\bar{\mathsf{P}}) \quad . \tag{2.9}$$

2.5 Unscented Kalman Filter (UKF)

The Extended Kalman Filter [Kalman, 1960b], EKF, has been widely used to do state-estimation on non-linear systems by approximating the state distribution to a Gaussian distribution using a first-order linearization of the non-linear system. As a recurrent first-order approximation the EKF can introduce significant errors in the true posterior mean and covariance of the system.

The Unscented Kalman Filter [Wan & Van Der Merwe, 2000] proposes that the state's Gaussian random variable should be approximated by a minimal set of carefully chosen sample points denoted by sigma points. After selection, the points are propagated through the non-linear system. This improvement is demonstrated in Figure 2.3.

2.5.1 The Unscented Transform

[Julier & Uhlmann, 1997] introduced the unscented transform to calculate the sigma points deterministically. Assuming a random variable $\mathbf{x}$ of dimension L that has mean $\bar{\mathbf{x}}$ and covariance $\Sigma_{\mathbf{x}}$, a matrix $\mathcal{X}$ of $2L+1$ sigma vectors $\mathcal{X}_i$ are created by:

$$\begin{aligned} \mathcal{X}_0 &= \bar{\mathbf{x}} \\ \mathcal{X}_i &= \bar{\mathbf{x}} + (\sqrt{(L+\lambda)\Sigma_{\mathbf{x}}})_i \qquad i = 1, \ldots, L \\ \mathcal{X}_i &= \bar{\mathbf{x}} - (\sqrt{(L+\lambda)\Sigma_{\mathbf{x}}})_{i-L} \quad i = L+1, \ldots, 2L \end{aligned} \quad , \tag{2.10}$$

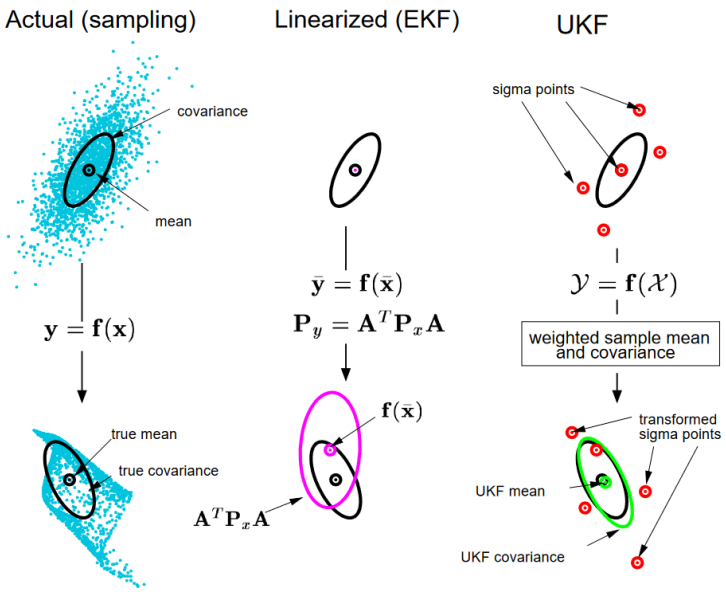


Figure 2.3: *Example of mean and covariance propagation. On the left is shown how the true mean and covariance are propagated through the model $f()$. At the center, the first order linearization is used by the EKF, and on the right is displayed the propagation using sigma points, which much closely relates to the actual sampling. From [Van der Merwe & Wan, 2001]*

where $\lambda = \alpha^2(L + \kappa) - L$ is a scaling parameter. α determines the spread of the sigma points around the mean and is usually a small positive value. κ is a secondary scaling parameter and usually set to 0. $(\sqrt{(L+\lambda)\Sigma_{\mathbf{x}}})_i$ is the *i*th row of the matrix square root that may be obtained using, *e.g.*, Cholesky decomposition.

The sigma points also have associated weights that are to be used when recovering the Gaussian distribution from the transformed points:

$$\begin{aligned} W_0^{(m)} &= \lambda/(L+\lambda) \\ W_0^{(c)} &= W_0^{(m)} + (1 - \alpha^2 + \beta) \\ W_i^{(m)} &= W_i^{(c)} = 1/\{2(L+\lambda)\} \quad i = 1, \ldots, 2L \end{aligned} \tag{2.11}$$

where β is used to incorporate prior knowledge of the distribution of $\mathbf{x}$.

The state can then be propagated through the non-linear model, $g()$, by:

$$\begin{aligned} \bar{y} &\approx \sum_{i=0}^{2L} W_i^{(m)} g(\mathcal{X}_i) \\ \Sigma_y &\approx \sum_{i=0}^{2L} W_i^{(c)} \{g(\mathcal{X}_i) - \bar{\mathbf{y}}\}\{g(\mathcal{X}_i) - \bar{\mathbf{y}}\}^T \end{aligned} \tag{2.12}$$

This method is accurate to the third order for Gaussian inputs for all non-linearity, with the accuracy of third and higher-order moments determined by choice of α and β.

2.5.2 Prediction Step of the UKF

The filter's prediction step consists of the choice of sigma points (Equation. 2.10) and consequent propagation through the non-linear action model using the inputs given to the model. Then equation 2.12 is used to calculate the prediction step median and covariance.

UKF Prediction Step

Calculate Sigma Points:

$$\mathcal{X}_{t-1} = \begin{bmatrix} \bar{\mathbf{x}}_{t-1} & \bar{\mathbf{x}}_{t-1} + (\sqrt{(L+\lambda)\Sigma_{t-1}}) & \bar{\mathbf{x}}_{t-1} - (\sqrt{(L+\lambda)\Sigma_{t-1}}) \end{bmatrix} \tag{2.13}$$

Propagate through the action model:

$$\mathcal{X}_t = g\left(\mathbf{u}_t, \mathcal{X}_{t-1}\right) \tag{2.14}$$

Calculate new average and covariance:

$$\bar{x}_t = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_t^{*[i]} \tag{2.15}$$

$$\Sigma_t = \sum_{i=0}^{2L} W_i^{(c)} \{\mathcal{X}_t^{*[i]} - \bar{\mathbf{x}}_t\}\{\mathcal{X}_t^{*[i]} - \bar{\mathbf{x}}_t\}^T + \mathsf{R}_t \tag{2.16}$$

where R_t is a tuning parameter of the filter and is associated with the covariance gained by doing a prediction step and $\mathbf{u}_t$ are the inputs given to the system at time step t.

2.5.3 Update Step of the UKF

The update step is similar to the prediction step by propagating the sigma points through the observation model, $h(\mathbf{x})$, to obtain the expected observation. The Kalman gain is then calculated using the observation's covariance and used to update the state.

UKF Update Step

Calculate Sigma Points:

$$\mathcal{X}_t = \begin{bmatrix} \mathbf{x}_t & \mathbf{x}_t + (\sqrt{(L+\lambda)\Sigma_t}) & \mathbf{x}_t - (\sqrt{(L+\lambda)\Sigma_t} \end{bmatrix} \tag{2.17}$$

Propagate through the measurement model and calculate prediction:

$$\mathcal{Z}_t = h(\mathcal{X}_t) \tag{2.18}$$

$$\hat{\mathbf{z}}_t = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Z}_t^{[i]} \tag{2.19}$$

UKF Update Step (cont.)

Calculate covariance and Kalman gains from the prediction:

$$\mathsf{S}_t = \sum_{i=0}^{2L} W_i^{(c)} \{\mathcal{Z}_t^{[i]} - \hat{\mathbf{z}}_t\}\{\mathcal{Z}_t^{*[i]} - \hat{\mathbf{z}}_t\}^T + \mathsf{Q}_t \tag{2.20}$$

$$\Sigma_t^{\mathbf{x},\mathbf{z}} = \sum_{i=0}^{2L} W_i^{(c)} \{\mathcal{X}_t^{[i]} - \bar{\mathbf{x}}_t\}\{\mathcal{Z}_t^{*[i]} - \hat{\mathbf{z}}_t\}^T \tag{2.21}$$

$$\mathsf{K}_t = \Sigma_t^{\mathbf{x},\mathbf{z}} \mathsf{S}_t^{-1} \tag{2.22}$$

Calculate new average and covariance:

$$\bar{\mathbf{x}} = \bar{\mathbf{x}} + K_t(\mathbf{z}_t - \hat{\mathbf{z}}_t) \tag{2.23}$$

$$\Sigma_t = \Sigma_t - \mathsf{K}_t \mathsf{S}_t \mathsf{K}_t^T \tag{2.24}$$

where Q_t is a tuning parameter of the filter and is associated with the covariance gained by making an observation, and it depends on the quality of the sensor used. $\mathbf{z}_t$ and $\hat{\mathbf{z}}_t$ are the observation and expected observation at time step t, respectively.

2.5.4 The Square Root UKF (SR-UKF)

The most computationally expensive operation of the UKF is on the square root of the covariance needed whenever sigma points are calculated (see equations 2.13 and 2.17). An efficient implementation using Cholesky factorization has $\mathcal{O}(L^3/6)$ time complexity. The Square-Root UKF proposed by [Van der Merwe & Wan, 2001] reduces this complexity to $\mathcal{O}(LM^2)$ by propagating the covariance directly, where M is the observation dimension.

In order to achieve this performance, it takes advantage of three powerful linear algebra techniques, QR decomposition, Cholesky factor updating and efficient least squares. From [Van der Merwe & Wan, 2001]:

- **QR decomposition:** The QR decomposition or factorization of a matrix $\mathsf{A} \in \mathbb{R}^{L \times N}$ is given by, $\mathsf{A}^T = \mathsf{QR}$, where $\mathsf{Q} \in \mathbb{R}^{N \times N}$ is orthogonal, $\mathsf{R} \in \mathbb{R}^{N \times L}$ is upper triangular and $N \geq L$. The upper triangular part of R, $\tilde{\mathsf{R}}$, is the transpose of the Cholesky factor of $\Sigma = \mathsf{A}\mathsf{A}^T$, *i.e.*, $\tilde{\mathsf{R}}$ is the square root of the covariance such that $\tilde{\mathsf{R}}^T\tilde{\mathsf{R}} = \mathsf{A}\mathsf{A}^T$.
- **Cholesky factor updating:** If S is the original Cholesky factor of $\Sigma = \mathsf{A}\mathsf{A}^T$, then the Cholesky factor of the rank-1 update $\Sigma \pm \sqrt{v}\mathbf{u}\mathbf{u}^T$ is denoted as $\mathsf{S} = \text{cholupdate}(\mathsf{S}, \mathbf{u}, \pm v)$. If $\mathbf{u}$ is a matrix and not a vector, then the result is M consecutive updates of the Cholesky factor using the M columns of $\mathbf{u}$.
- **Efficient Least Squares:** For a square matrix A, the solution to the equation $(\mathsf{A}\mathsf{A}^T)\mathbf{x} = \mathsf{A}^T\mathbf{b}$ also corresponds to the solution of the over determined least squares problem $\mathsf{A}\mathbf{x} = \mathbf{b}$. This can be solved efficiently using a QR decomposition with pivoting.

According to [Van der Merwe & Wan, 2001] even though the computational complexity of the filters are of the same order the SR-UKF is about 20% faster than the UKF and about 10% faster than the EKF.

Chapter 3

Visual Inertial Feature-Based Simultaneous Localization And Mapping

This chapter formulates the problem in Section 3.1 and then explains the proposed solution in Section 3.2. In Section 3.2.1, the state, action and measurement models are defined. Section 3.2.2 gives a brief overview of Matrices on Lie Groups, and Section 3.2.3 explains the uncertainties function on those same groups. Finally, the filter implementation is showcased in Section 3.2.4.

3.1 Problem Formulation

Odometry comes from the Greek words *odos* (meaning "route") and *metron* (meaning "measure") and, as such, odometry is the problem of measuring a robot route through an environment. Rotary encoders in robots that use wheels as a way of moving can estimate the movement, but this may not be possible for, e.g., robots that slide too much (giving wrong estimations) or aerial drones that do not use wheels. From this problem came the approach of using a camera to estimate the robot's pose by adjusting the position of specific features that it could see over time; this approach is called Visual Odometry, first proposed by [Nister et al., 2004].

An Inertial Measurement Unit (IMU) provides an inertial estimation of the movement that can help the camera measure its pose; when this happens, it is then called Visual Inertial Odometry. IMUs report the angular velocity (w) and linear accelerations (a) in the sensor frame. A rigid transformation can be used to map one frame to another:

$$\mathsf{T} = \begin{bmatrix} \mathsf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \quad , \tag{3.1}$$

where T is the transformation matrix, $\mathsf{R} \in \mathrm{SO}(3)$ the rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ the translation from one from to the other. This set of transformations constitute the Special Euclidean Group ($\mathrm{SE}(3)$).

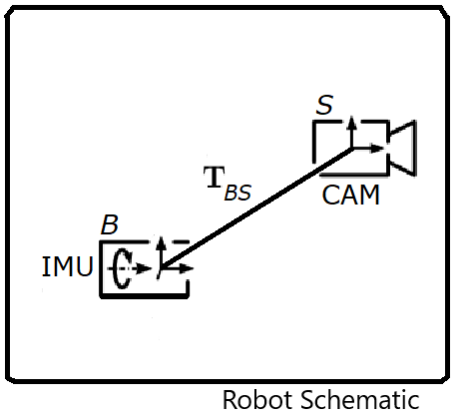


Figure 3.1: *Example transformation from SE(3) where T provides a transformation from the IMU frame to the camera frame. Adapted from [Burri et al., 2016].*

One of the main advantages of using an IMU as an odometry sensor is the speed at which it can update the system, given its high frequency (higher than $100\,\mathrm{Hz}$) and that the update on the pose is usually straightforward. However, an IMU cannot be used as the only sensor since the pose is obtained through continuous integration of the accelerations from the IMU with quantification noise. Since this noise is additive, it introduces a bias that drifts the system away from the absolute pose.

3.2 Proposed Approach

The proposed approach closely follows the work done by [Brossard et al., 2017] in which an SR-UKF is used to integrate IMU and visual information. One of the main innovations that it introduces is the use of Lie Groups as a structure for the state leading to a direct representation and propagation of the uncertainty in the Lie Group. The filter estimates the robot's pose (Rotation, velocity and location), the 3D location of the features detected, and the IMU biases, the latter being appended to the state estimation.

3.2.1 State, Action and Measurements Model

In [Brossard et al., 2017], the Special Euclidean of multiple direct spatial isometries $\mathrm{SE}_{2+p}(3)$ group is presented to naturally represent the problem in a Lie Group structure as leveraged in [Barrau & Bonnabel, 2015], where the $(2+p)$ subscript signifies that the group includes the position and velocity (2) and all of the landmarks positions (p). This group extends the Special Euclidean group and provides the state representation and a closed-form solution to the Lie exponential of the uncertainties. The state can be then represented as

$$\chi = \begin{bmatrix} \mathsf{R} & \mathbf{v} & \mathbf{x} & \mathbf{p}_1 \cdots \mathbf{p}_p \\ 0_{(p+2)\times 3} & & I_{(p+2)\times(p+2)} & \end{bmatrix} \in \mathrm{SE}_{2+p}(3) \quad , \tag{3.2}$$

where $\mathsf{R} \in \mathrm{SO}(3)$ is the rotation matrix, $\mathbf{v} \in \mathbb{R}^3$ is the velocity of the robot, $\mathbf{x} \in \mathbb{R}^3$ is the position of robot and finally $\mathbf{p}_k \in \mathbb{R}^3$ is the position of the kth landmark. The uncertainties, defined as $\xi = [\xi_{\mathsf{R}}^T\ \xi_{\mathbf{v}}^T\ \xi_{\mathbf{x}}^T\ \xi_{\mathbf{p_1}}^T\ \cdots\ \xi_{\mathbf{p_p}}^T] \in \mathbb{R}^{9+3p}$ are mapped to the Lie algebra through the transformation $\xi \to \xi^\wedge$ defined as

$$\xi^{\wedge} = \begin{bmatrix} [\xi_{\mathsf{R}}]_{\times} & \xi_{\mathbf{v}} & \xi_{\mathbf{x}} & \xi_{\mathbf{p_1}} \cdots \xi_{\mathbf{p_p}} \\ & & 0_{(2+p)\times(5+p)} & \end{bmatrix} \quad . \tag{3.3}$$

The biases of the IMU are appended to the state and are represented as

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_{\omega} \\ \mathbf{b}_{a} \end{bmatrix} \quad \in \mathbb{R}^6 \quad , \tag{3.4}$$

where $\mathbf{b}_{\omega}$ is the bias of the angular velocity and $\mathbf{b}_{a}$ the bias of the linear velocity reported by the IMU.

Action model

The action model represents how the system can change in time. It uses the information given by the IMU to change the robot's pose and assumes that the landmarks do not move and that the robot's frame is the same as the IMU's. The dynamics of the system read

$$\text{body state} \quad \begin{cases} \dot{\mathsf{R}} = \mathsf{R}\left[\omega - \mathbf{b}_{\omega} + \mathbf{n}_{\omega}\right]_{\times} \\ \dot{\mathbf{v}} = \mathsf{R}\left(\mathbf{a} - \mathbf{b}_{a} + \mathbf{n}_{a}\right) - g \\ \dot{\mathbf{x}} = \mathbf{v} \end{cases} \tag{3.5}$$

$$\text{IMU biases} \quad \begin{cases} \dot{\mathbf{b}_{\omega}} = \mathbf{n}_{\mathbf{b}_{\omega}} \\ \dot{\mathbf{b}_{a}} = \mathbf{n}_{\mathbf{b}_{a}} \end{cases} \tag{3.6}$$

$$\text{landmarks} \quad \dot{\mathbf{p_i}} = 0,\ i = 1, \ldots, p \quad , \tag{3.7}$$

where the various white Gaussian noises can be stacked as

$$\mathbf{n} = \begin{bmatrix} \mathbf{n}_{\omega}^T & \mathbf{n}_{a}^T & \mathbf{n}_{\mathbf{n}_{\omega}}^T & \mathbf{n}_{\mathbf{n}_{\mathbf{a}}}^T \end{bmatrix}^T \sim \mathcal{N}(0, \mathbf{Q}) \quad . \tag{3.8}$$

Measurement Model

The measurements are given by the calibrated monocular camera that observes the p landmarks through a standard perspective projection model, like the on described in Section 2.1. Each landmark $\mathbf{p}_i$ is observed through the camera as

$$\mathbf{y}_i = \begin{bmatrix} y_u^i \\ y_v^i \end{bmatrix} + \mathbf{n}_y^i \quad , \tag{3.9}$$

where $\mathbf{y}_i$ is the measured pixel location of the landmark in the camera, that is,

$$\lambda \begin{bmatrix} y_u^i \\ y_v^i \\ 1 \end{bmatrix} = \mathsf{P}\mathbf{p}_i \quad , \tag{3.10}$$

with λ being the scale parameter, P the camera projection matrix from Equation 2.5 and $\mathbf{p}_i$ the 3-dimensional position of the landmark in homogeneous coordinates. Finally, $\mathbf{n}_y^i \sim \mathcal{N}(0, N)$ represents the pixel image noise.

3.2.2 Matrix Lie Groups

From [Brossard et al., 2017] a matrix Lie group $G \subset \mathbb{R}^{N \times N}$ is a subset of square invertible matrices such that the following properties hold

$$\mathbf{I} \in G; \quad \forall \chi \in G, \chi^{-1} \in G; \quad \forall \chi_1, \chi_2 \in G, \chi_1 \chi_2 \in G \quad . \tag{3.11}$$

Locally about the identity matrix $\mathbf{I}$, the group G can be identified with a Euclidean space $\mathbb{R}^q$ using the matrix exponential map $\exp_m(.)$ where $q = \dim G$. Indeed, to any $\tau \in \mathbb{R}^q$ one can associate a matrix $\tau^\wedge$ of the tangent space of G at $\mathbf{I}$ called the Lie algebra $\mathfrak{g}$. The exponential map Exp: $\mathbb{R}^q \to G$ for Lie group is defined as

$$\text{Exp}(\tau) = \exp_m(\tau^\wedge) \tag{3.12}$$

Locally, it is a bijection, and one can define the Lie algorithm map log: $G \to \mathbb{R}^q$ as the exponential inverse, leading to

$$\text{Log}(\text{Exp}(\tau)) = \tau \quad . \tag{3.13}$$

An overview of the Lie Mappings is shown in Figure 3.2.

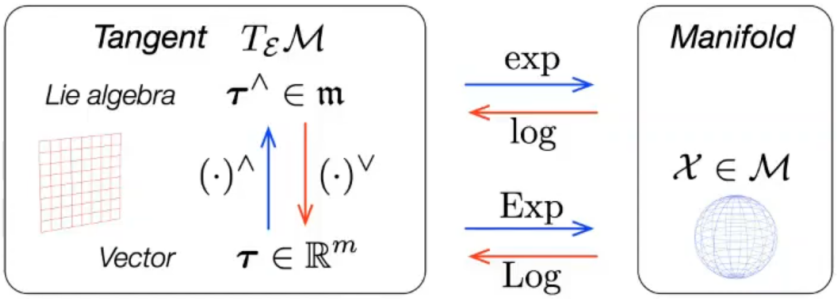


Figure 3.2: *Lie Group Mappings - From the Group (Manifold) to the Lie algebra to the vector space and the inverse mappings. From [Solà et al., 2018].*

3.2.3 Uncertainty on Lie Groups

A random variable on the Lie Group cannot be defined as an additive noise for $\chi_1, \chi_2 \in G$ since G is not a vector space and $\chi_1 + \chi_2 \notin G$ does not hold. Following [Brossard et al., 2017] the probability distribution $\chi \sim \mathcal{N}_R(\chi, \mathsf{P})$ for the random variable $\chi \in G$ is defined as

$$\chi = \text{Exp}(\xi)\bar{\chi}, \quad \xi \sim \mathcal{N}(0, \mathsf{P}) \quad . \tag{3.14}$$

In the case of the implementation to be used, the uncertainty ξ is defined by equation 3.3. It was chosen the right multiplication variation since it provided increased accuracy and numeral consistency compared to the left variant.

3.2.4 Filter Implementation

Since the filter needs a discrete model to propagate the state through time, To integrate the action model on the filter, it was discretized using the Euler method except for the Rotation. From [Brossard et al., 2017], the discrete action model becomes:

$$\text{body state} \quad \begin{cases} \mathsf{R}(t+\Delta t) = \mathsf{R}(t)\exp\left[(\omega(t)-\mathbf{b}_\omega(t))\Delta t + \text{Cov}(\mathbf{n}_\omega)^{1/2} - g\sqrt{\Delta t}\right]_\times \\ \mathbf{v}(t+\Delta t) = \mathbf{v}(t) + (\mathsf{R}(t)\,(\mathbf{a}(t)-\mathbf{b}_a(t)) - g)\Delta t \\ \mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t \end{cases} \tag{3.15}$$

$$\text{IMU biases} \quad \begin{cases} \mathbf{b}_\omega(t+\Delta t) = \mathbf{b}_\omega(t) \\ \mathbf{b}_a(t+\Delta t) = \mathbf{b}_a(t) \end{cases} \tag{3.16}$$

$$\text{landmarks} \quad \mathbf{p_i}(t+\Delta t) = \mathbf{p}_i(t),\ i = 1,\ldots,p \tag{3.17}$$

Having the measurement from the IMU at time step k given as $\mathbf{u}_k = \begin{bmatrix}\omega_k^T & a_k^T\end{bmatrix}^T$ the equations from 3.15 to 3.17 correspond to the f function that serves as the action model for the filter

$$\chi_{k+1} = f(\chi_k, \mathbf{u}_k, \omega_k) \quad , \tag{3.18}$$

where the state χ_k lives in G, and $\omega_k \sim \mathcal{N}(0, \mathsf{Q}_k)$ is a white Gaussian noise, associated with generic discrete measurements of the form

$$\mathbf{y}_k = h(\chi_k, \mathbf{v}_k) \quad , \tag{3.19}$$

where $\mathbf{v}_k \sim \mathcal{N}(0, \mathsf{R}_k)$ is a white Gaussian noise. The final full model associated to the Right-SR-UKF-LG (Right Square Root Unscented Kalman Filter on Lie Groups, from [Brossard et al., 2017]), denominated as Fusion2018 from now on, becomes

$$\text{state} \quad \begin{cases} \chi_k = \exp(\xi)\bar{\chi}_k \\ \mathbf{b}_k = \bar{\mathbf{b}}_k + \tilde{\mathbf{b}} \end{cases} \quad , \quad \begin{bmatrix}\xi \\ \tilde{\mathbf{b}}\end{bmatrix}, \sim \mathcal{N}(0, \mathsf{P}_k), \tag{3.20}$$

$$\text{dynamics} \quad \begin{cases} \chi_k, \mathbf{b}_k = f(\chi_{k-1}, \mathbf{u}_k - \mathbf{b}_{k-1}, \mathbf{n}_k) \end{cases} \quad , \tag{3.21}$$

$$\text{observations} \quad \begin{cases} \mathbf{Y}_k = \begin{bmatrix}\mathbf{y}_1^T & \cdots & \mathbf{y}_p^T\end{bmatrix}^T := \mathbf{Y}(\chi_k, \omega_k) \\ \quad \mathbf{y}_k \text{given in 3.9},\ i = 1,\ldots,p \quad , \end{cases} \tag{3.22}$$

such that $(\bar{\chi_k}, \bar{\mathbf{b}}_k) \in \mathbb{R}^{(15+3p)}$ represents the mean (estimated) state at time k, $\mathsf{P}_k \in \mathbb{R}^{(15+3p)\times(15+3p)}$ is the covariance matrix that defines the state uncertainties $(\xi, \tilde{\mathbf{b}})$, and the vector $\mathbf{Y}_k$ contains the observations of the p landmarks with associated Gaussian noise $\mathbf{w}_k \sim \mathcal{N}(0, \mathsf{W})$.

The complete filter operation is shown in Figure 3.3, where the Propagation and Update steps are the ones earlier defined, and steps to detect features use the method described in [Shi & Tomasi, 1994] and match features using the Matlab built-in functions. Feature Initialization is done by triangulation using the external map.

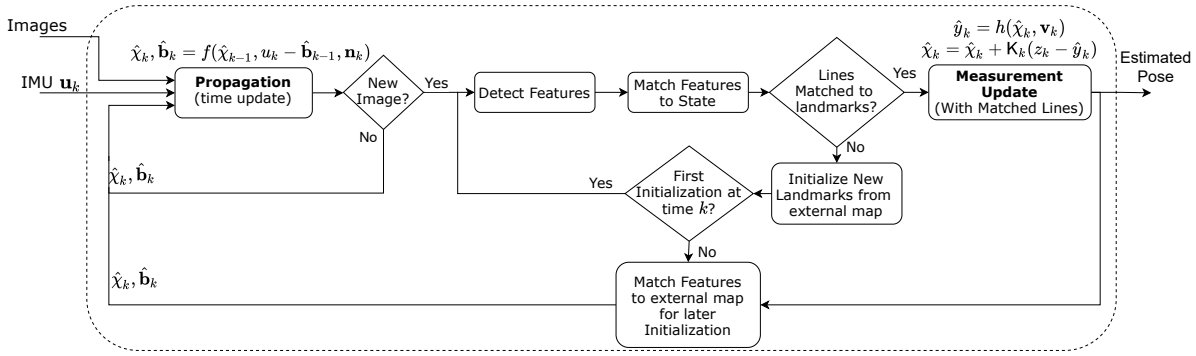


Figure 3.3: ***Fusion2018 Block Diagram*** *Filter Proposed by [Brossard et al., 2017] to perform Visual Inertial Odometry.*

It should be noted that the Fusion2018 filter expects a good initialization, where the state is initialized with the ground truth values from the dataset for the pose (initial conditions of the robot) and 30 3D landmark positions given by [Mur-Artal & Tardos, 2017]. The external map also uses 200 initial features given by [Mur-Artal & Tardos, 2017].

Chapter 4

Lines based Visual Inertial Odometry

This chapter details the main building blocks that make Lines based VIO. In Section 4.1 is shown the Line Detection method used. Section 4.2 introduces a matching algorithm created for Line Matching. Section 4.3 presents a method to initialize Lines using Plucker coordinates. In Section 4.4 is presented a Line Initialization method that uses a minimum square error formulation to find points that minimize the distance to the back-projected rays. Finally, in Section 4.5, is shown how these systems are integrated into the filter described in chapter 3.

4.1 Line Detection using Unified Line Segment Detection (ULSD)

Following the approach in Section 2.2, where lines are detected using the Canny Edge detector and the Hough Transform, a more robust way of extracting 2D lines from images was needed. A solution is presented in [Li et al., 2020] by taking advantage of a stacked hourglass network ([Newell et al., 2016]) to extract a feature map used to propose and validate line predictions.

The complete network can be separated into two modules; the first module, denominated by "Line Proposal Network", tries to predict the junctions of lines using another network outputting confidence and a junction offset map; in this first step, duplicated junctions are also removed from the map. Lines are predicted by trying to find line segments represented by a Bezier curve. Finally, line segments that can match the endpoints to two junction proposals based on the Euclidean distance are kept. If multiple line proposals are matched with the same junctions, only those with the shortest cost are kept.

The second module, "Line of Interest", takes the candidate lines from the first module and feature map from the first network to validate the line segment prediction. The candidate is first passed through a Bezier Align function that samples uniformly the line to form the final prediction, which is then fed into another classifier network to get a confidence level for each line. In Figure 4.1 is shown an overview of the network.

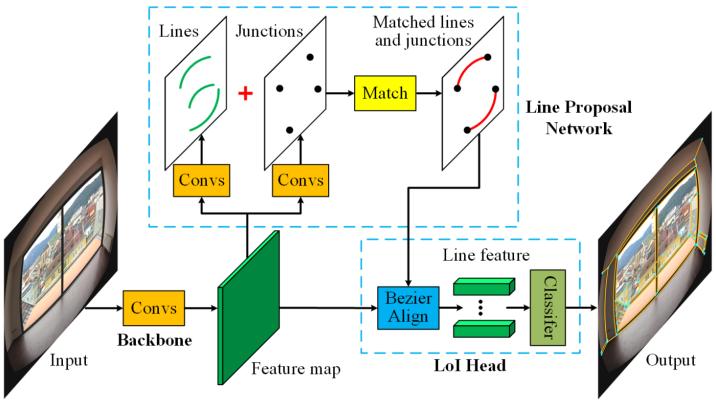


Figure 4.1: *Unified Line Segment Detection (ULSD) - Network Overview. From [Li et al., 2020]*

4.2 Line Matching Algorithm

Following Section 2.3 the chosen algorithm to pair lines between images or the projection of 3D lines is a variation of the Iterative Closest Point. The first step is to match lines using the Sum of Squared Distances (**SSD**) between the endpoints of two lines; minimizing the sum of this error is known as the Assignment Problem, which can be formally described as

$$\underset{a \in A}{\text{minimize}} \quad \sum \mathsf{C}_{a,f(a)} \tag{4.1}$$

where C is a square real-valued matrix where each matrix position $(a, f(a))$ is the cost of matching the line of ID a to the line of ID $f(a)$. The cost of the match between two lines is defined by

$$\mathsf{C}_{a,f(a)} = \|\mathcal{L}_1^a - \mathcal{L}_1^{f(a)}\|^2 + \|\mathcal{L}_2^a - \mathcal{L}_2^{f(a)}\|^2 \tag{4.2}$$

where $\mathcal{L}_i^k$ represent the ith endpoint of the line with ID k. In order to correctly match endpoints, the endpoints of the second line are swapped, and the distance is recalculated; It is then chosen the orientation which provides the smaller distance. The minimization problem is solved using the Hungarian method proposed by James Munkres in [Munkres, 1957]. Matches are also filtered by a maximum error cost between any two lines.

After the first match, Procrustes [Gower & Dijksterhuis, 2005] is applied to the endpoints of the matched lines. The obtained transformation is then applied to all the points in the image so that lines that should be matched to each other get closer together. As such, this method requires that at least a match so that a transformation can be calculated. An overview of this iteration is shown in Figure 4.2.

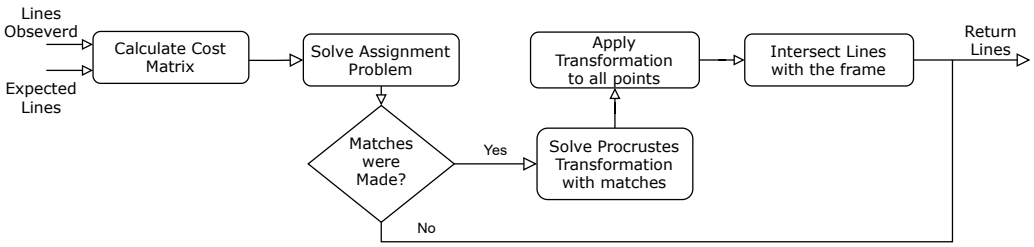


Figure 4.2: ***ICP Line Iteration Block Diagram*** *This algorithm matches lines using the* ***SSD*** *ant then adjusts the endpoints positions of all points using the transformation found by applying Procrustes to the matched endpoints.*

Since the objective is to minimize the error given by equation 4.1, the method described in Figure 4.2 is iteratively run until the error no longer decreases above a fixed threshold. Since the ICP is used with the objective of matching in mind, it simply returns the matches obtained. After the ICP algorithm is done, RANSAC is used to filter some outliers that may appear from overfitting to some matches. The entire algorithm is shown in Figure 4.3.

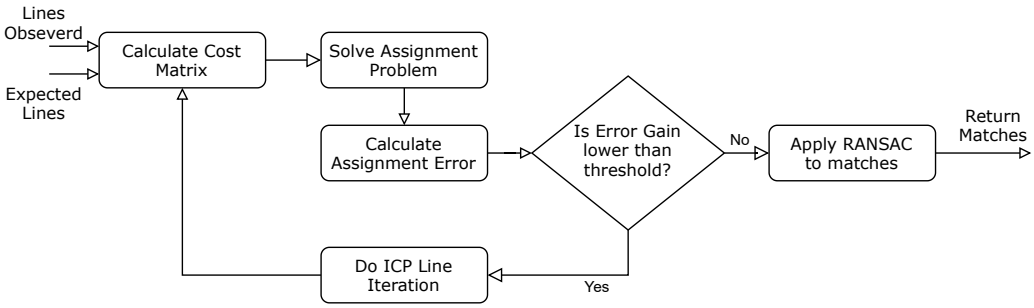


Figure 4.3: ***ICP Line Block Diagram*** *This algorithm performs iterative matches and lines transformations until the error gain is lower than a fixed threshold (10^{-6} in the case of this thesis). The first solve of the assignment problem tolerates a higher error in matches than subsequent iterations.*

4.3 Line Initialization in Plücker Coordinates

Line initialization is crucial to obtain 3D features that the filter can propagate to update the system's state. With this objective in mind, it was proposed by [Bartoli & Sturm, 2005] a maximum likelihood estimator using the squared 2D orthogonal (Euclidean) distance from the line $\mathbf{L}$ to the projected endpoints weighted by their distance to the camera, which is formally written as

$$\mathcal{E}(\mathbf{L}, [\mathbf{x}, \mathbf{y}]) = d_{\perp}^2(\mathbf{x}, \mathbf{l}) + d_{\perp}^2(\mathbf{y}, \mathbf{l}) \quad , \tag{4.3}$$

where $[\mathbf{x}, \mathbf{y}]$ are the two weighted points, $\mathbf{l}$ is the projection of the line $\mathbf{L}$ and $d_{\perp}^2(\mathbf{q}, \mathbf{l})$ is defined by

$$d_{\perp}^2(\mathbf{q}, \mathbf{l}) = (\mathbf{q}^{\mathbf{T}}\mathbf{l})^2/(l_1^2 + l_2^2) \quad . \tag{4.4}$$

Having 3D lines $\mathcal{S} = \{\boldsymbol{L}^1, \ldots, \boldsymbol{L}^m\}$ and cameras $\mathcal{M} = \{\mathsf{P}^1, \ldots, \mathsf{P}^n\}$ the negative log likelihood function $\mathcal{E}(\mathcal{S}, \mathcal{M})$ for the reconstruction, corresponding to the total reprojection error, can be written in terms of individual reprojection errors $\mathcal{E}(\mathbf{L}^j, \mathcal{M})$ for each line j:

$$\mathcal{E}(\mathcal{S}, \mathcal{M}) = \sum_{j=1}^{m} \mathcal{E}(\mathbf{L}^j, \mathcal{M}) \tag{4.5}$$

$$\mathcal{E}(\mathbf{L}^j, \mathcal{M}) = \sum_{i=1}^{n} \left(d_\perp^2(\mathbf{x}^{ij}, \mathbf{l}^{ij}) + d_\perp^2(\mathbf{y}^{ij}, \mathbf{l}^{ij}) \right) \quad . \tag{4.6}$$

Since all projections should correspond to the same 3D line, the final functional to be minimized becomes

$$\underset{\mathbf{L}}{\text{minimize}} \qquad \mathcal{E}(\mathbf{L}, \mathcal{M}) = \sum_{i=1}^{n} \left(d_\perp^2(\mathbf{x}^i, \mathbf{l}^i) + d_\perp^2(\mathbf{y}^i, \mathbf{l}^i) \right) \quad . \tag{4.7}$$

4.3.1 Linear Algorithm

Ignoring the denominator in equation (4.4) leads to an algebraic distance denoted d_a, biased compared to the orthogonal distance. It is linear in the predicted line and defined by $d_a^2(\mathbf{q}, \mathbf{l}) = d_\perp^2(\mathbf{q}, \mathbf{l}) w^2 = (\mathbf{q^T l})^2$, where the scalar factor w encapsulates the bias as $w^2 = l_1^2 + l_2^2$:

$$(w^i)^2 = \left((\tilde{\mathsf{P}}^i \mathbf{L})_1 \right)^2 + \left((\tilde{\mathsf{P}}^i \mathbf{L})_2 \right)^2 \quad . \tag{4.8}$$

The biased linear least squares error function is then defined by

$$\begin{aligned} \mathcal{B}(\mathbf{L}, \mathcal{M}) &= \sum_{i=1}^{n} \left(\left(x^{iT} \tilde{\mathsf{P}}^i \mathbf{L} \right)^2 + \left(y^{iT} \tilde{\mathsf{P}}^i \mathbf{L} \right)^2 \right) \\ &= \|\mathsf{A}_{(2n \times 6)} \mathbf{L}\|^2 \qquad \text{with} \qquad \mathsf{A} = \begin{pmatrix} \cdots \\ x^{iT} \tilde{\mathsf{P}}^i \\ y^{iT} \tilde{\mathsf{P}}^i \\ \cdots \end{pmatrix} \quad . \end{aligned} \tag{4.9}$$

Since $\mathbf{L}$ is a homogeneous vector, the constraint $\|\mathbf{L}\|^2 = 1$ must be added. The $\mathbf{L}$ that minimizes $\mathcal{B}(\mathbf{L}, \mathcal{M})$ is then given by the singular vector of A associated to its smallest singular value, that can be computed using SVD.

The linear algorithm provides a biased estimate of the plucker line $\mathbf{L}$ that can estimate the weight factors that contain the bias of the linear least-squares error function. This process is then used iteratively until convergence, using the difference between consecutive errors as a threshold. A correction should be made between each iteration to ensure that the plucker constraint is satisfied, as described in 4.3.2.

4.3.2 Plücker Correction

Given a 6-vector $\mathbf{L}^T \sim (\mathbf{a}^T | \mathbf{b}^T)$, this algorithm computes the closest Plücker coordinates $\widehat{\mathbf{L}}^T \sim (\mathbf{u}^T | \mathbf{v}^T)$, i.e., $\mathbf{u^T v} = 0$, in the sense of the $\mathcal{L}_2$-norm, i.e, $\|\widehat{\mathbf{L}} - \mathbf{L}\|^2$ is minimized.

Plücker Correction

Compute the Singular Value Decomposition $(\mathbf{a} \quad \mathbf{b}) = \bar{\mathsf{U}}\bar{\Sigma}\bar{\mathsf{V}}^T$.

Let $\bar{\mathsf{Z}} = \bar{\Sigma}\bar{\mathsf{V}}^T$, form matrix $\mathsf{T} = \begin{bmatrix} z_{21} & z_{22} \\ z_{12} & -z_{11} \end{bmatrix}$.

Compute singular vector $\hat{\mathbf{v}}$ associated to the smallest singular value of matrix T.

Define $\widehat{\mathsf{V}} = \begin{bmatrix} \hat{v}_1 & -\hat{v}_2 \\ \hat{v}_2 & \hat{v}_1 \end{bmatrix}$ and set $(\mathbf{u} \quad \mathbf{v}) \sim \bar{\mathsf{U}}\widehat{\mathsf{V}}\,\mathrm{diag}\left(\widehat{\mathsf{V}}^T\bar{\Sigma}\bar{\mathsf{V}}^T\right)$

Return $\widehat{\mathbf{L}}^T \sim (\mathbf{u}^T|\mathbf{v}^T)$

4.3.3 Quasi-linear Algorithm

According to [Bartoli & Sturm, 2005], the linear algorithm misbehaves since the plücker constraint is not considered while solving the linear least-squares problem.

A new approach is suggested by rewriting the constraint as $\mathcal{C}(\mathbf{L}) = \mathbf{L}^T\mathsf{G}\mathbf{L} = 0$ where $\mathsf{G} = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$. Computing the null space of $\mathbf{L}^T\mathsf{G}$, using SVD, gives a base of all vectors that satisfy the plücker constraint; this base can then be plugged into the linear estimator (Equation. 4.9) to give a line $\mathbf{L}$ that satisfies the plücker constraint. The estimate is then used iteratively, as before, to compute the bias that reweighs the least-squares problem (Equation. 4.8) until convergence.

4.3.4 Endpoints Extraction from the Plücker Coordinates

Since the endpoints will be propagated through the SR-UKF filter, they must be extracted from the Plucker Coordinates. First, the Plucker line is transformed back into Euclidean Coordinates, then the endpoints are estimated by finding the closest line points to the back-projected rays of the endpoints observed in each image. These values are then averaged to give a final prediction of the 3D endpoints. An example is shown in Figure 4.4 in which, since there is no noise, the closest points to the line are the intersection. The points chosen are always part of the line estimated using the quasi-linear algorithm.

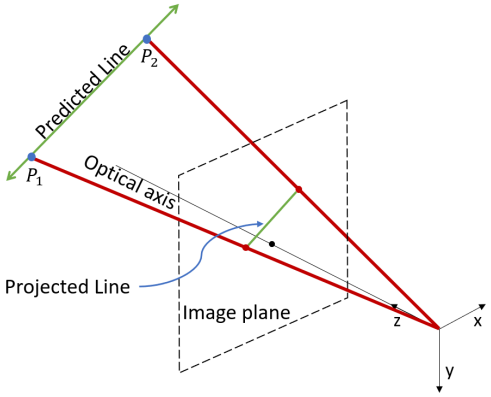
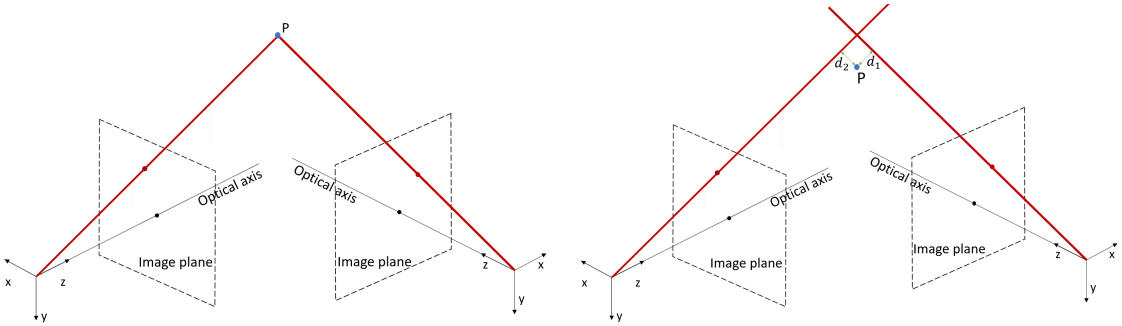


Figure 4.4: *Example of the usage of back-projection rays to determine the endpoints of an estimated Plucker Line.*

4.4 Line Initialization Using 3D Distance to Back Projected Rays.



(a) *Noiseless Projection where the Back Projected rays intersect in a single point P.*

(b) *Back Projected rays in the presence of noise measurements not intersecting in a single point*

Figure 4.5: *Example of back-projection of rays with and without noise. Both rays intersect the original point on the left since there is no noise, but on the right, those back-projected rays do not cross in a noisy reality.*

In a noiseless world, all back-projected rays from a single 3D point should intersect in that same point, but this is not always true due to noises in the image or the camera's pose. In reality, the rays will pass close to the real point but probably not over, as shown in Figure 4.5 ; as such, the real point should be close to every back-projected ray from every view that can see him. The 3D distance from a generic point $\mathbf{p}$ to a line defined by $\mathbf{L} = \mathbf{a} + \lambda\mathbf{u}$ can be calculated as the height of a parallelogram formed by a segment of the line and a vector from the line to the generic point:

$$d(\mathbf{L}, \mathbf{p}) = \frac{||\overrightarrow{\mathbf{ap}} \times \mathbf{u}||}{||\mathbf{u}||} \Rightarrow ||\overrightarrow{\mathbf{ap}} \times \mathbf{u}||, \qquad \text{if} \qquad ||\mathbf{u}|| = 1. \tag{4.10}$$

Having the back-projected rays $\{\mathbf{L}^1 = \mathbf{a}^1 + \lambda\mathbf{u}^1, \ldots, \mathbf{L}^n = \mathbf{a}^n + \lambda\mathbf{u}^n\}$, a mean squared error formulation can be written as

$$\underset{\mathbf{p}}{\text{minimize}} \quad F(\mathcal{L}, \mathbf{p}) = \frac{1}{2}\sum_{k=1}^{n} ||d(\mathbf{L}^k, \mathbf{p})||^2 \tag{4.11}$$

$$F(\mathcal{L}, \mathbf{p}) = \sum_{k=1}^{n} \left[\overrightarrow{\mathbf{a}^k\mathbf{p}}_y \mathbf{u}_3^k - \overrightarrow{\mathbf{a}^k\mathbf{p}}_z \mathbf{u}_2^k\right]^2 + \left[\overrightarrow{\mathbf{a}^k\mathbf{p}}_z \mathbf{u}_1^k - \overrightarrow{\mathbf{a}^k\mathbf{p}}_x \mathbf{u}_3^k\right]^2 + \left[\overrightarrow{\mathbf{a}^k\mathbf{p}}_x \mathbf{u}_2^k - \overrightarrow{\mathbf{a}^k\mathbf{p}}_y \mathbf{u}_1^k\right]^2 \quad , \tag{4.12}$$

where $\overrightarrow{\mathbf{a}^k\mathbf{p}}_i$ is ith coordinate of the vector defined by $\overrightarrow{\mathbf{a}^k\mathbf{p}} = \mathbf{p} - \mathbf{a}^k$, and $\mathbf{u}_i$ is the ith coordinate of the vector $\mathbf{u}$. The vectors $\mathbf{u}$ that define the direction of the line should be normalized so that the Equation 4.12 is valid. Since the minimum of the functional is found when the gradient is equal to zero, a closed form solution is given by:

$$\nabla F(\mathcal{L},\mathbf{p})=0 \Leftrightarrow \sum_{k=1}^{n}\begin{bmatrix} {\mathbf{u}_2^k}^2+{\mathbf{u}_3^k}^2 & -\mathbf{u}_1^k\mathbf{u}_2^k & -\mathbf{u}_1^k\mathbf{u}_3^k \\ -\mathbf{u}_1^k\mathbf{u}_2^k & {\mathbf{u}_1^k}^2+{\mathbf{u}_3^k}^2 & -\mathbf{u}_2^k\mathbf{u}_3^k \\ -\mathbf{u}_1^k\mathbf{u}_3^k & -\mathbf{u}_2^k\mathbf{u}_3^k & {\mathbf{u}_1^k}^2+{\mathbf{u}_2^k}^2 \end{bmatrix}\begin{bmatrix} \overrightarrow{\mathbf{a}^k\mathbf{p}_x} \\ \overrightarrow{\mathbf{a}^k\mathbf{p}_y} \\ \overrightarrow{\mathbf{a}^k\mathbf{p}_z} \end{bmatrix}=0$$
$$\Leftrightarrow \sum_{k=1}^{n}\mathsf{U}^k(\mathbf{p}-\mathbf{a}^k) \Leftrightarrow \left(\sum_{k=1}^{n}\mathsf{U}^k\right)\mathbf{p}=\sum_{k=1}^{n}\mathsf{U}^k\mathbf{a}^k \tag{4.13}$$

4.5 Integration of Line Features into Fusion 2018

The algorithms proposed in this chapter served as a replacement to the algorithms used to detect, match and initialize features to the Fusion2018. One main difference is that even though the filter propagates the endpoints of the lines as independent landmarks, they maintain the line structures that are then used to match the state landmarks and the images taken by the camera. The blocks that are replaced in the Fusion2018 filter are shown in green in Figure 4.6.

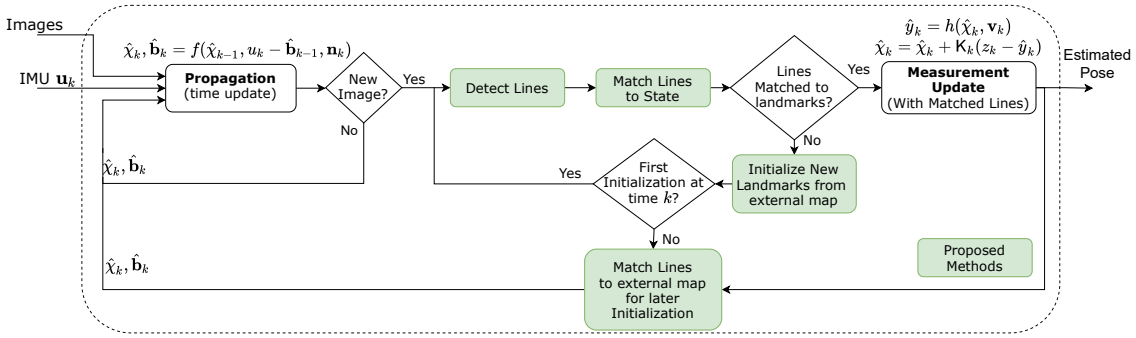


Figure 4.6: ***Fusion2018 Block Diagram*** *Filter Proposed by [Brossard et al., 2017] to perform Visual Inertial Odometry with Lines as Features. Green blocks are the new blocks introduced to use line features.*

The external map saves the image coordinates and projection matrix of each matched view for every line ID that was still not initialized. If there is a low number of matches to the landmarks (they do not appear in frame) the initialization step can be called to try and create landmarks which can be seen by the system. The state initialization will be discussed in Section 5.5.

Chapter 5

Experiments

This chapter describes the experiments performed to validate the methodologies presented and introduced in previous chapters. Section 5.1 introduces the dataset used to test the proposed algorithms with real data. Section 5.2 validates the line detection method introduced in Section 4.1. Section 5.3 evaluates the matching algorithm proposed in Section 4.2. Section 5.4 tests the initialization techniques from Sections 4.4 and 4.3 using a simulator created in Matlab. Finally, Section 5.5 tests the full proposed solution and compares it to the Fusion2018 filter.

5.1 The EuRoC MAV Dataset

The dataset used in this thesis was recorded by [Burri et al., 2016] in the context of the European Robotics Challenge (EuRoC) to assess the contestant's visual-inertial simultaneous localization and mapping (SLAM) and 3D reconstruction capabilities on micro aerial vehicles (MAVs). More specifically it will be used the medium difficulty data set recorded on the Vicon MoCap Room.

The dataset features an accurate 3D point cloud of the environment captured with a laser scanner and also the 6D pose ground truth (position and rotation) from the flight of the AscTec "Firefly" hex rotor helicopter, shown in Figure 5.2. The drone features an Inertial Measurement Unit (IMU) and two cameras (even though only one is used). The full flight of the drone and the room ground truth point cloud is shown in Figure 5.1.

The Fusion18 dataset consists of 16020 IMU measurements and 1602 images. The IMU values can be used directly since its frame is aligned with the body's referential. On the other hand, when used for image reconstruction, the camera image coordinates need an extra transformation to ensure that the values are mapped to the correct frame. As such, the camera projection matrix from equation 2.5 becomes

$$\mathsf{P} = \mathsf{K}(\mathsf{R}_w^b \mathsf{R}_b^c)^T \begin{bmatrix} I_3 & -(\mathbf{t}_w^b + \mathsf{R}_w^b \mathsf{R}_b^c \mathbf{t}_b^c) \end{bmatrix} \in \mathbb{R}^{3\times 4} \quad , \tag{5.1}$$

where $\mathsf{R}_a^b \in \mathrm{SO}(3)$ is the rotation matrix from referential a to b, and $\mathbf{t}_a^b \in \mathbb{R}^3$ is the translation vector from referential a to b; w, b and c are the referential of the world, the drone (IMU) and the camera, respectively. Using the 3D point cloud given by the dataset, 3D lines were initialized and used to verify that the projection matrix was working correctly by projecting them onto different images using the pose ground truth; an example is shown in Figure 5.3.

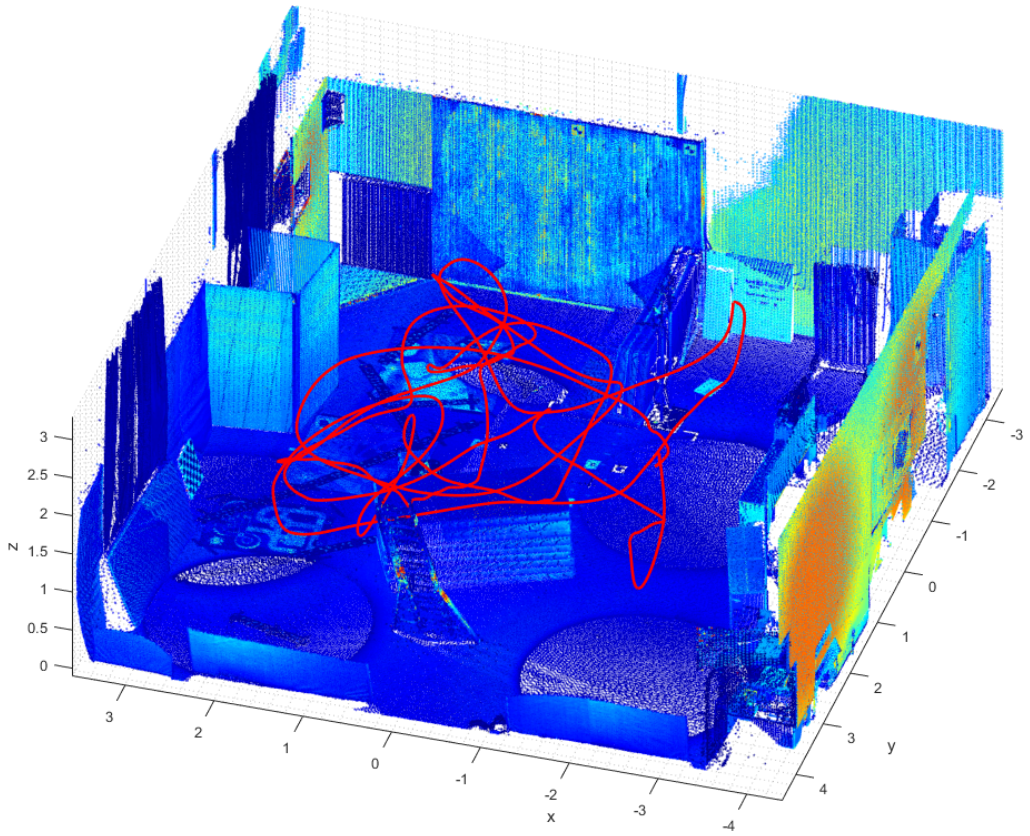
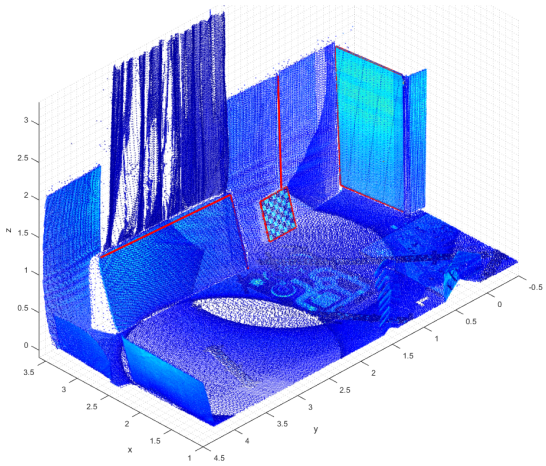
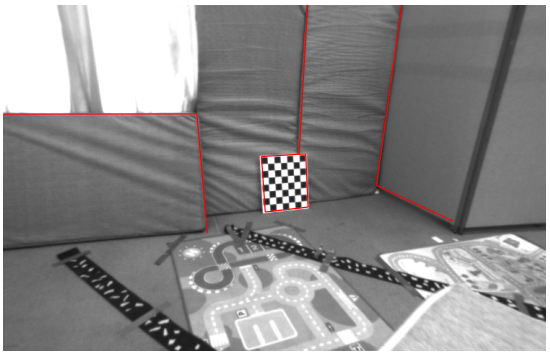


Figure 5.1: *Full drone flight.*

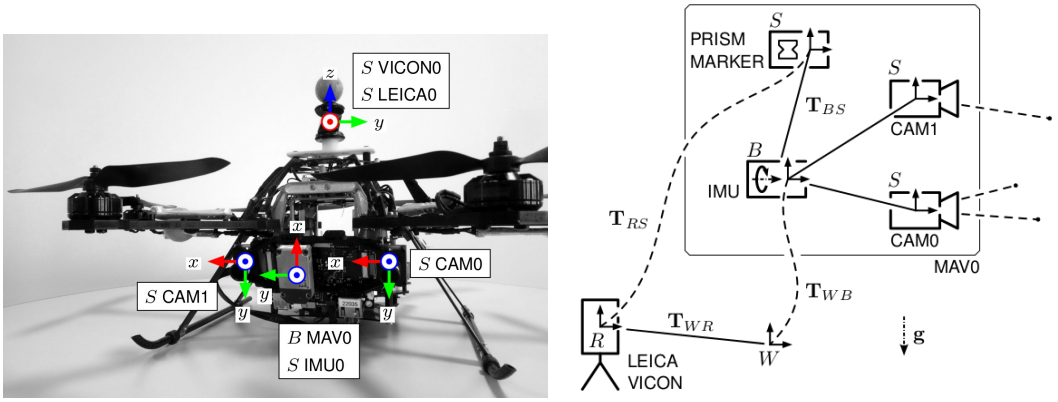


(a) *Sector of the Vicon MoCap Room 1 with the 3D lines used to verify the projection matrix.*



(b) *Example of the lines projected using the projection matrix and ground truth of the pose.*

Figure 5.3: *Lines used to test Projection Matrix and ground truths of the dataset. The Lines used are marked in red.*



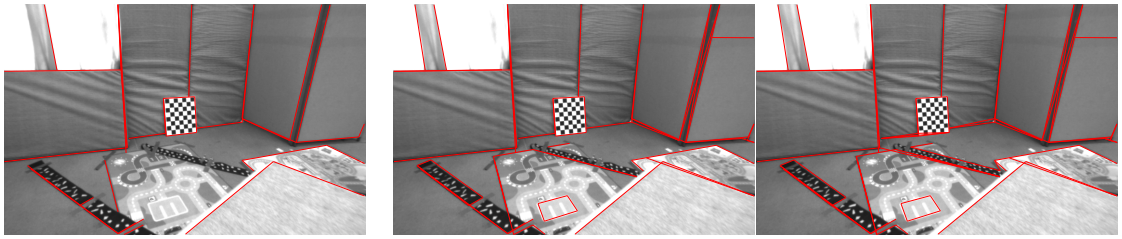
(a) *The AscTec "Firefly" helicopter.*

(b) *Frame transformations between the world and sensors.*

Figure 5.2: *The drone used in the recorded dataset and the transformations between frames of the sensors in the drone and the world. From [Burri et al., 2016].*

5.2 Line Detection with ULSD

The ULSD network can be trained separately, but since [Li et al., 2020] provide a pre-trained model for each type of camera, the pinhole model is used for line detection. The confidence interval (*CI*) which declines or accepts lines can be adjusted if there is a need to detect more or fewer lines, Figure 5.4 shows that effect. The *CI* set to 70% is used since there is an improvement in the number of detected lines without losing accuracy, being detected on average 67 lines per image. Lower thresholds identify repeated lines, increasing the complexity for the matching task to maintain lines alive through multiple frames.



(a) *Lines detected with* 90% *CI (default)*

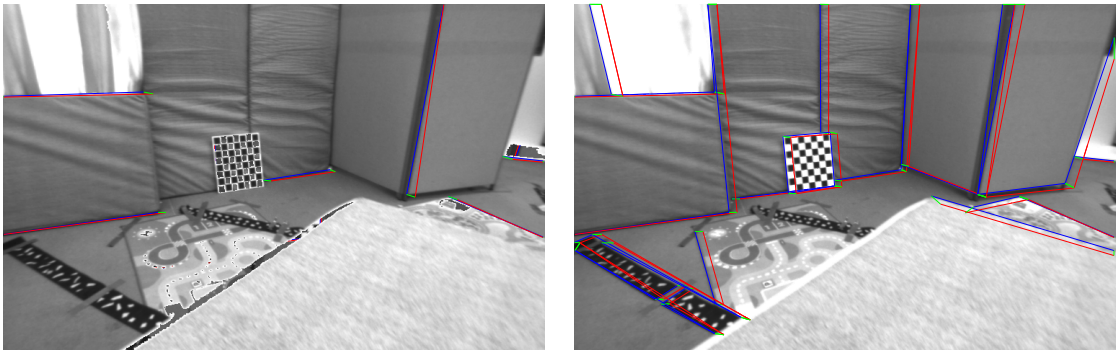
(b) *Lines detected with* 70% *CI*

(c) *Lines detected with* 50% *CI*

Figure 5.4: *ULSD Line Detection - Example of multiple confidence intervals.*

5.3 Line Matching Validation

The algorithm was validated using subsequent images from the Fusion18 dataset where lines were identified using the method in Section 4.1. It is possible to check how much the algorithm improved the number of matches obtained in Figure 5.5. This growth is vital so that lines are consistently matched across images making possible a posterior 3D initialization.

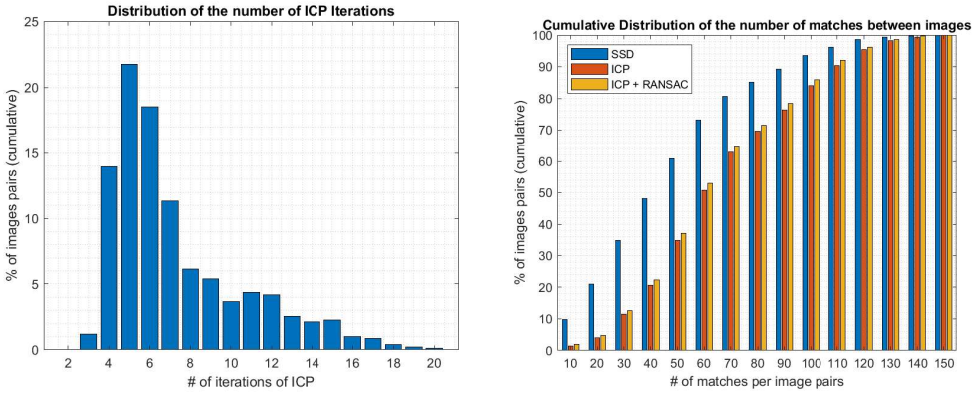


(a) *Matches obtained by SSD*

(b) *Matches obtained by ICP Line*

Figure 5.5: *Example of ICP Line improvement on matching. Blue lines represent the lines identified on the image, and red lines correspond to lines found in other images that were matched to this image. Finally, green lines join the matched endpoints.*

The Fusion18 dataset was then used to better understand how long the algorithm would take to converge and how many matches were gained by utilizing the ICP. It was found that it took 6.3 iterations to converge, and more than 17 matches were gained per a pair of images. A complete overview of these results can be seen in Figure 5.6. More importantly, the percentage of pairs of images with less than 20 matches went from 20% to less than 5%, demonstrating the power of this algorithm in increasing the matches found. The algorithm also showed promising results when matching lines from images with greater time steps between them.



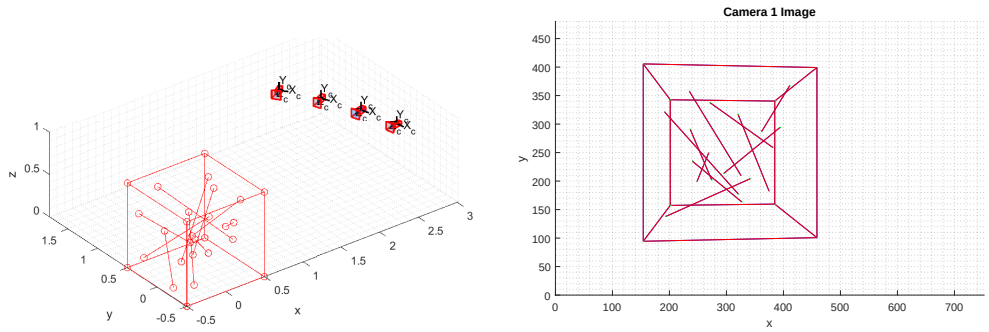
(a) *Distribution of the number of ICP iterations.*

(b) *Cumulative Distribution of the number of matches made between images.*

Figure 5.6: *Quantitive analysis of the ICP Line algorithm. On the left is presented the number of iterations that the algorithm took to reach the fixed error gain. On the right is the cumulative distribution of the number of matches between images. These results come from the counts achieved over 1600 images.*

5.4 Effect of Noise on 3D Lines Reconstruction

A Matlab simulator was created to test the line initialization techniques, which consisted of a cube centered on the origin with random lines inside and a variable number of cameras that would take pictures of the cube and lines. The line initialization techniques were then tested and then compared to the "real" lines. In the simulator, it is possible to modify the number of cameras, the positions of each camera and the option to use a camera for the projection of the reconstructed lines to assess the reprojection errors of the initialized lines. The simulator setup is shown in Figure 5.7.



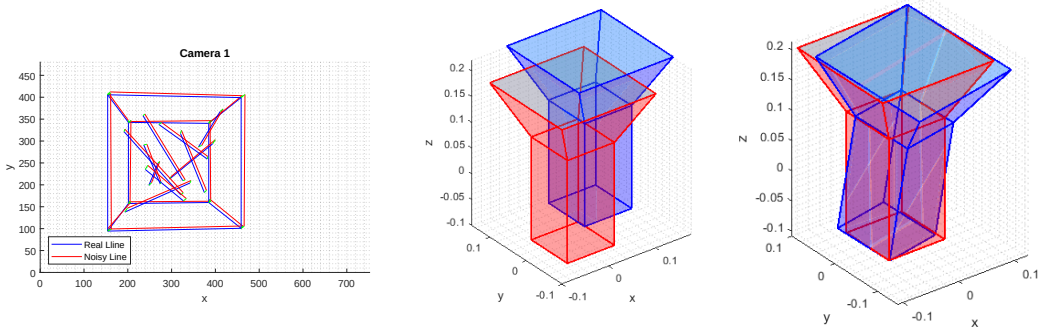
(a) *Example of possible camera positions in the simulator and the cube with the random lines inside.*

(b) *Example of a "picture" taken by a camera in the simulator.*

Figure 5.7: ***Simulator used to test initialization techniques.*** *On the left is shown the 3D setup and on the right a picture taken by one of the cameras.*

The first validation was done with no noise added to the simulator to ensure the validity of the algorithms in a case where perfect measurements are made. The techniques resulted in euclidean distance errors lower than 10^{-15}, appearing from Matlab functions' numerical precision. These results supported the validity of the initializations using the [Bartoli & Sturm, 2005] and Section 4.4 methods, at least in a noiseless environment.

The simulator provides three types of noise, shown in Figure 5.8, that can be tested separately or at the same time. The first type of noise appears from vibrations on the camera or for incorrect line detection, and it is manifested as pixel deviations on the line positions in the pictures taken by the camera. The other two errors come from the camera's pose and appear as location and rotation differences from the actual pose. The noise was modelled as a Gaussian distribution with zero average and covariance given by noise value selected.



(a) *Pixel Noise* **(b)** *Camera Location Noise* **(c)** *Camera Rotation Noise*

Figure 5.8: *Different types of Noise added in the simulation to test the initialization techniques.*

The cameras were placed near each other to understand the effect of noise on the lines' initialization. The reason was that the simulation could be closely related to the drone flight where there is little movement between each frame. As such, cameras were placed less than $20\,\mathrm{cm}$ apart and with less than 10 degrees of total rotation between each consecutive frame. The full cameras' pose is shown in Figure 5.9.

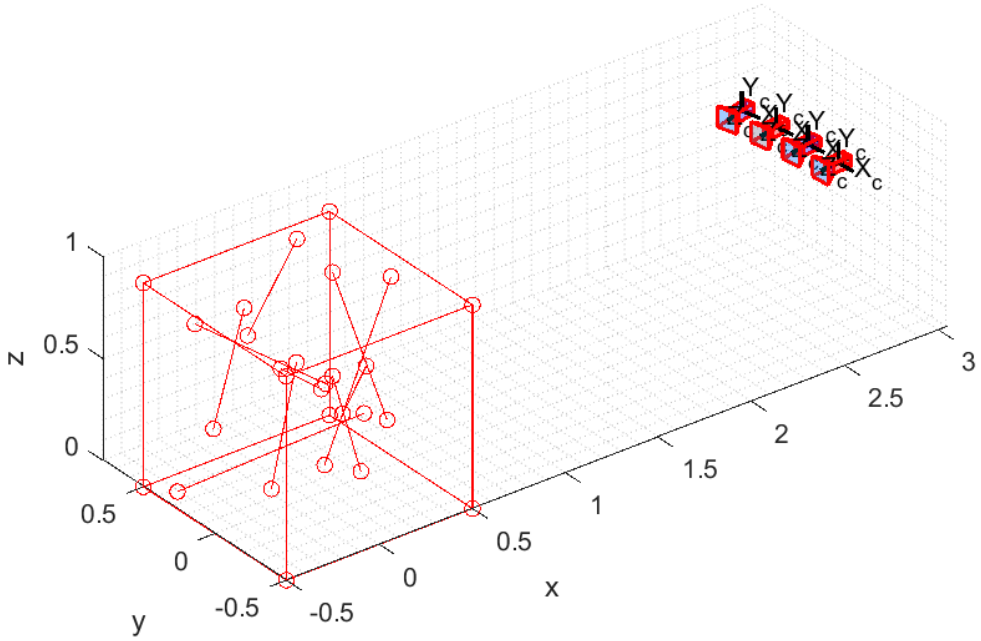
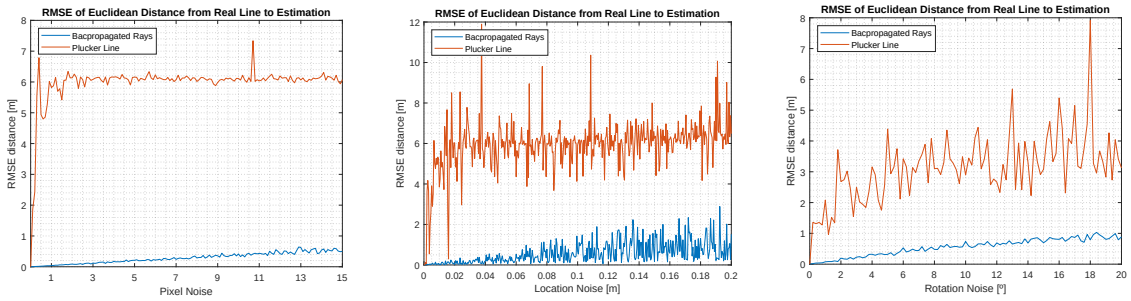


Figure 5.9: *Simulator setup used to measure the effect of noise to the line initialization techniques.*

As a first validation, all noise types were tested separately by setting all types to zero minus the one to be tested. A common occurrence is that the method described in section 4.3 using the Plücker coordinates misbehaves in the presence of noise and rapidly increases the reprojection error to values where the last frame would discard the

lines. One interesting aspect is that the error seems to reach a plateau, which may come from intersecting the infinite Plucker line with the back-projected rays to retrieve the endpoints.

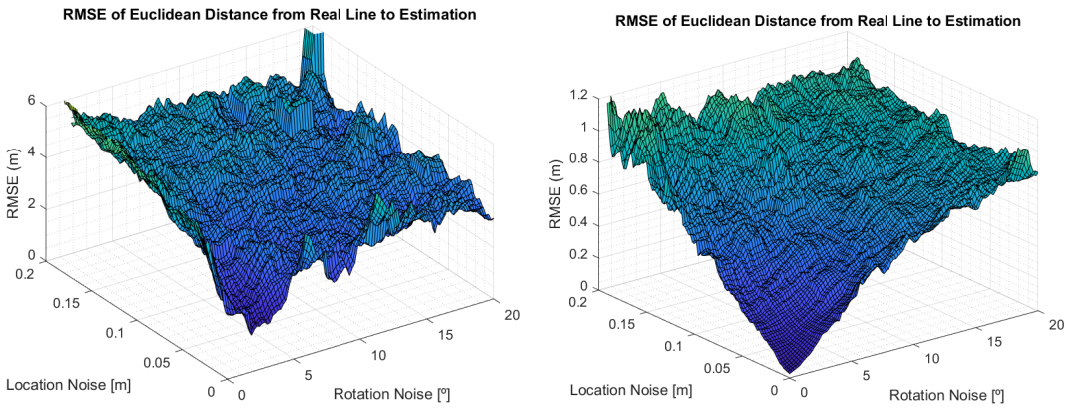
On the other hand, the technique using the minimum 3D distance to the back-projected rays seemed to behave predictably when exposed to all kinds of noise. It presented worse results in the cases where the location noise would come close to the difference in location of each frame (around 7 cm and higher noise). Even though it presents a lower error than the first method, this latter algorithm seems to be more affected by rotation noises.



(a) *RMSE in function of Pixel Noise* **(b)** *RMSE in function of Location Noise* **(c)** *RMSE in function of Rotation Noise*

Figure 5.10: *RMSE of the euclidean distance of the initialized lines to the real lines in function of the different types of noise.*

Since the pixel noise should suffer fewer variations while running the SR-UKF, it was set as fixed for the following two experiments, where location and rotation noise were analyzed in detail. First, the pixel noise was set to zero, the location noise varied from 0 to 20 cm, and the rotation noise from 0 to 20 degrees, which is higher than the difference in the camera's pose for consecutive frames, giving a complete picture on how the error of the initialization should vary for different combinations of noise. The error was measured by averaging the length from the actual endpoints and their initialization. The results are shown in Figure 5.11.



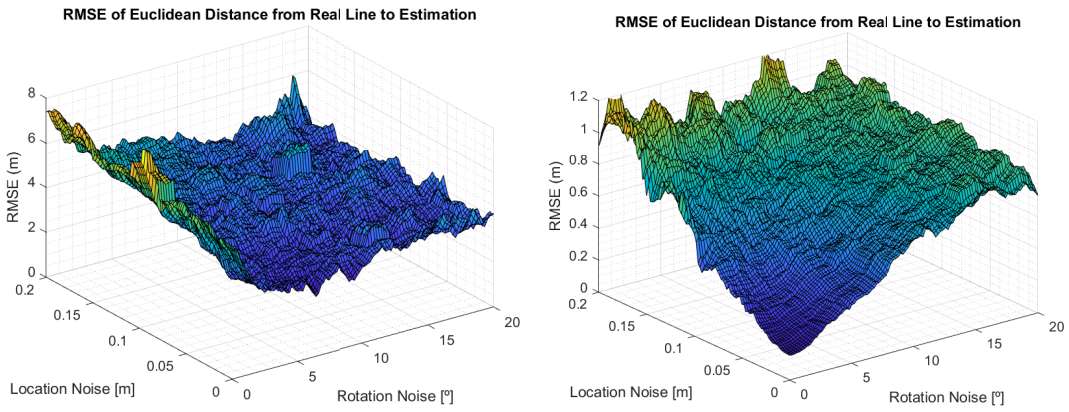
(a) *RMSE for the Plücker Line Method.* **(b)** *RMSE for the Back-Projected Rays Method.*

Figure 5.11: *RMSE of the euclidean distance of the initialized lines to the real lines in function of the rotation and location noise with pixel noise fixed as zero.*

As seen previously, the Plücker coordinates method showed little resistance to noise, increasing almost instantly to values above $2\,\mathrm{m}$ which would place the lines outside 1 by 1-meter cube. A strange phenomenon is that it seemed to achieve a little better results for small values of rotation noise but higher than zero. This phenomenon was found across multiple experiments, and it may be a result of the representation used where it benefits from a slightly higher rotation between frames to converge to a result, even though this rotation came from noise.

Once again, the Back-Projected Rays method proved to behave with a consistent increase in error for similar increases in noise, one crucial aspect that should be noted is that for the values of noise 5 degrees and $5\,\mathrm{cm}$, the method showed an error of $20\,\mathrm{cm}$ (on average $10\,\mathrm{cm}$ for each endpoint) in the initialization close to having a pixel noise of 4 as seen in Figure 5.12. For a reliable line initialization, the filter should maintain the location noise under $5\,\mathrm{cm}$ and rotation noise under 5 degrees, expecting that the filter can then correct the position of the lines.

It should be noted that the surfaces were passed through a median filter and so the values on the edges may not be entirely correct, this is evident in Figure 5.11 where the Plücker Line method shows a non-zero value for a noiseless initialization. For these values Figure 5.10 should be used as a reference.



(a) *RMSE for the Plücker Line Method.*

(b) *RMSE for the Back-Projected Rays Method.*

Figure 5.12: *RMSE of the euclidean distance of the initialized lines to the real lines in function of the rotation and location noise with pixel noise fixed as 4.*

5.5 Complete System Experiments, EuRoC MAV Dataset

5.5.1 Experiment 1, Lines Matching and 3D Initialization

The matching algorithm was run across all images of the EuRoC MAV Dataset, and then lines that survived for longer than 15 frames were initialized using the method of the back-projected rays described in Section 4.4. These results can be observed in Figure 5.13, and they show that many of the lines follow the environment correctly.

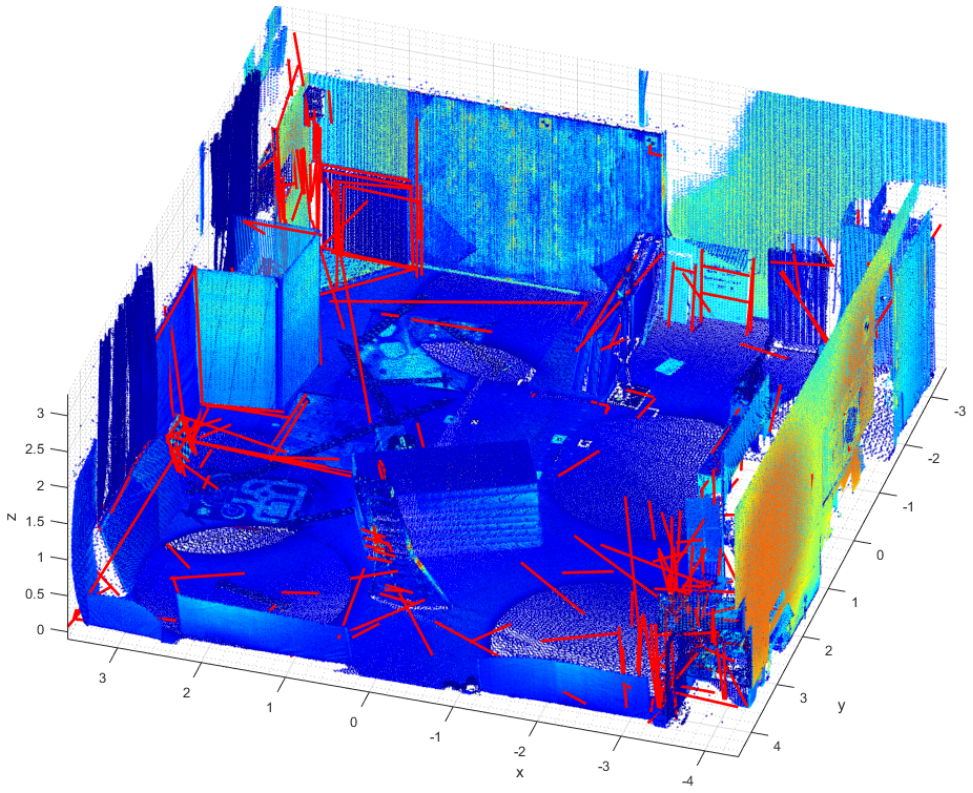
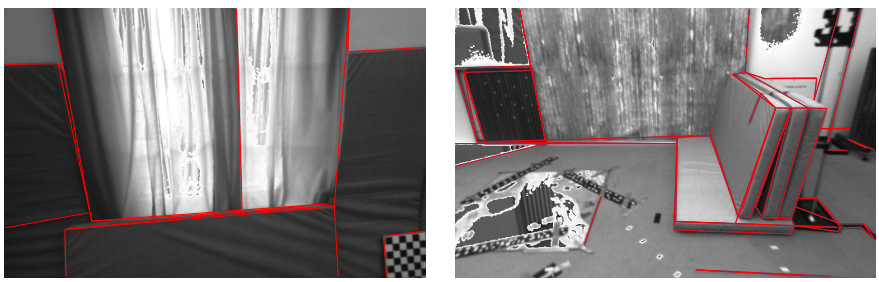


Figure 5.13: *Line Initialization using the matching and Initialization methods proposed and using the ground truth for the robot's pose. It shows that there are certain lines that were correctly initialized, but that there are zones of the map that don't have many lines or that have many that are mostly noise.*

Two main zones were identified where the technique could not initialize many lines, one where the drone observes a curtain and the other where it sees a wall. It was recognized that this problem appears from the line detector. This theory was validated upon inspection of the pictures taken in those areas, as shown in Figure 5.14. The main problem from those frames is that most observed lines disappear. Since lines are only matched on subsequent images, they are interpreted as new features when the system sees the same line again, with no relation to the past observations.



(a) *Example of frame with a curtain where few lines are detected.*

(b) *Example of frame with a wall where few lines are detected.*

Figure 5.14: *Example of frames where few lines are detected.*

Another important aspect is that lines that are never entirely in the camera's frame cannot be initialized, even though they were matched across many frames. Ignoring these lines is deliberate due to potential errors in the endpoint locations. These lines could be initialized using the method from Section 4.3 since it does not need the explicit endpoints' location.

5.5.2 Experiment 2, Filtered Approach

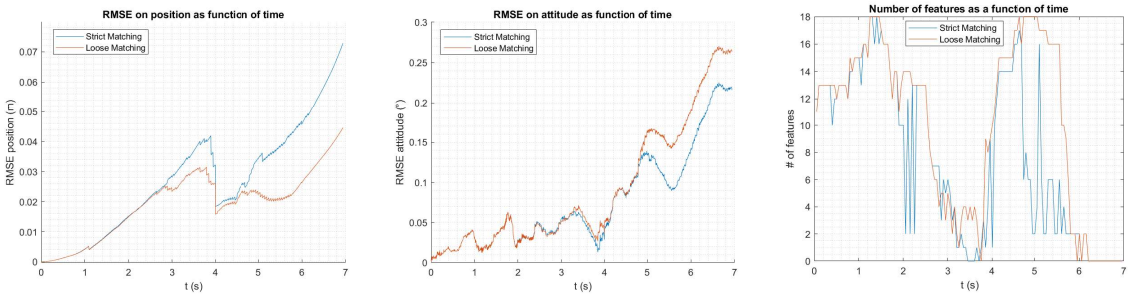
Following the Fusion2018 filter, the robot's pose was initialized with the ground truth. The landmarks were initialized using ten lines selected from Figure 5.13; the proposed filter needed a higher initial landmark covariance since the method's precision was lower than the ORB SLAM filter. The external map was initialized without any feature and is constructed as the filter progresses.

One of the problems encountered was that lines exhibited more pixel error from the reprojections of the landmarks than initially expected. This error is again due to the not entirely perfect line initializations selected for the landmarks. One solution to this problem was to allow a higher error for the first ICP line iteration. This problem is significantly intensified in the zone where the drone looks to curtains displayed in Figure 5.14, where few lines could be used to update the system state. This technique is termed "loose matching", and the results in Figure 5.15 show that it increases the number of features matched during the critical time steps between 3 and 4 seconds where the curtains occupy most of the frame.

The position error is obtained from the Euclidean distance between the robot's estimated pose and the ground truth. The rotation error is calculated using the metric proposed by [Park, 1995], where the distance between two rotation matrices can be calculated using the logarithm map of the product of the inverse of one of the matrices by the other. This metric can be formally described as

$$d(\mathsf{R}_1, \mathsf{R}_2) = \| \log(\mathsf{R}_1^{-1}\mathsf{R}_2) \| = \left\| \cos^{-1}\left(\frac{\mathrm{Tr}(\mathsf{R}_1^{-1}\mathsf{R}_2) - 1}{2} \right) \right\| \quad , \tag{5.2}$$

where $\|.\|$ denotes the Euclidean norm, $\mathsf{R}_1, \mathsf{R}_2 \in \mathrm{SO}(3)$ are rotation matrices and $\log(.)$ represents the logarithm map $\log : \mathrm{SO}(3) \to \mathfrak{so}(3)$.



(a) *RMSE of the position in function of time.*

(b) *RMSE of the rotation in function of time.*

(c) *Number of features detected at each timestep.*

Figure 5.15: *Error of the full proposed solution for two different types of ICP Line matching. The "loose matching" variation shows an improved matching that leads to lower errors in the position during the sector where the drone is affected by few lines detected.*

After the first six seconds of flight, the filter starts to deviate and is incapable of initializing lines soon enough due to loss of the combination of observing a new environment and passing through another zone with a low number of lines, shown in Figure 5.14 as the "wall". The filter tries to initialize the lines shown in Figure 5.16 to replace the state landmarks but diverges before fixing its pose beyond irreparable damage. A mix of points and lines could probably be used, in future works, to combat these zones where the environment is not very geometric.

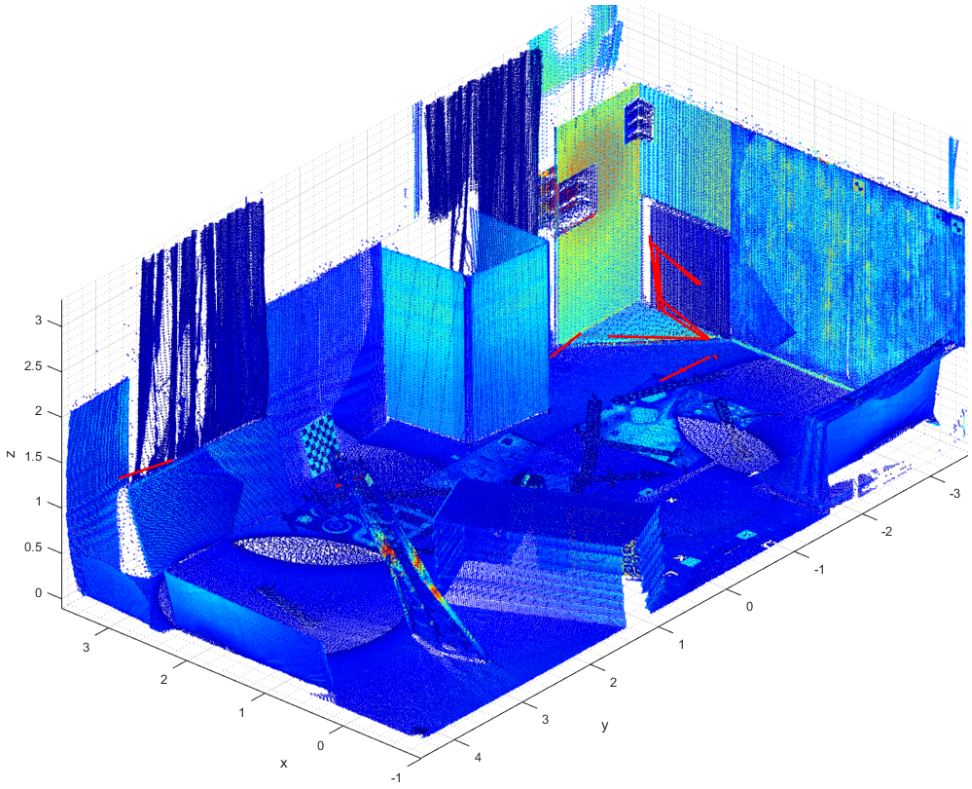
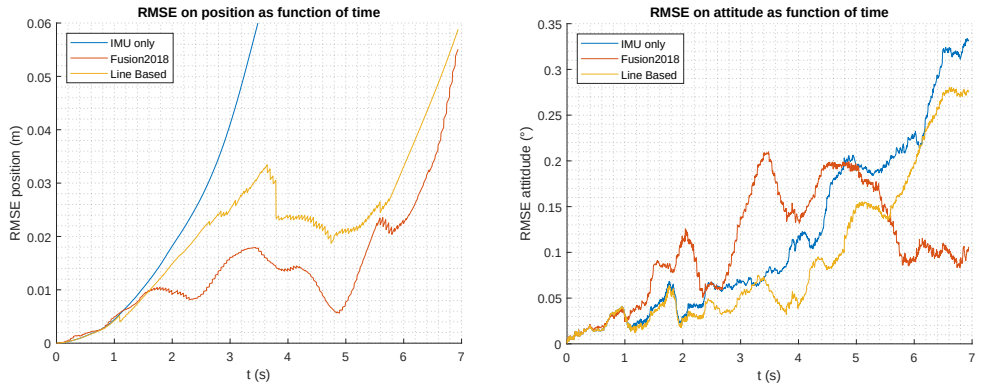


Figure 5.16: *Lines initialized by the filter before it diverges.*

In the following steps, the filter was compared to the Fusion2018, which works in the same circumstances since it does not initialize new features before seven seconds into the flight. The Fusion2018 filter is not lead to divergence since it can find features on the textured wall.

Since the used lines have lower precision than the landmarks used by the Fusion2018 filter, the proposed filter takes a longer time to start updating its position and trusts the IMU readings for the first three seconds. After the curtain section, the filter rapidly updates its position, almost obtaining the same error as the Fusion2018 filter. The filter excels in the rotation error while it has observable features, always maintaining a lower error than the IMU and the Fusion2018 filter; this may be due to lines giving better rotational information than points. These results are shown in Figure 5.17



(a) *RMSE of the position in function of time.* **(b)** *RMSE of the rotation in function of time.*

Figure 5.17: *Error of the full proposed solution compared to the Fusion2018 filter and only using IMU.*

The complete filter and ground truth trajectories are shown in Figure 5.18, and with more detail in Figure 5.19, where it is possible to check that the filter responds to the aggressive change in direction faster than the IMU and maintains the actual trajectory path where IMU rapidly diverges.

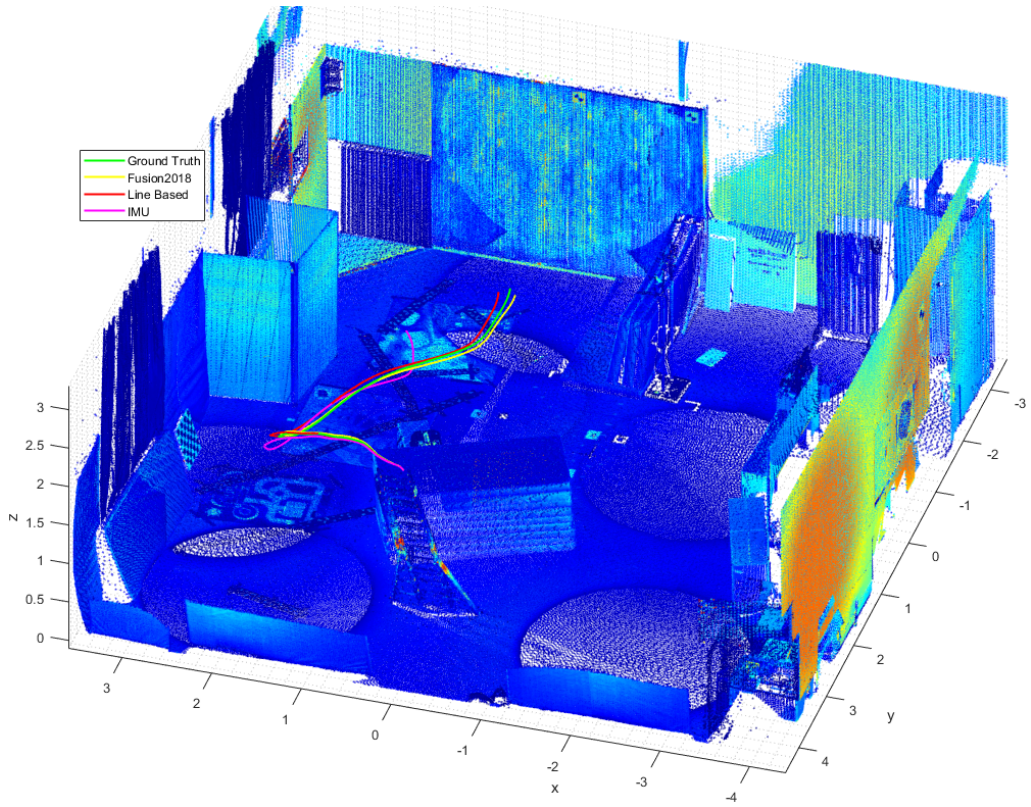
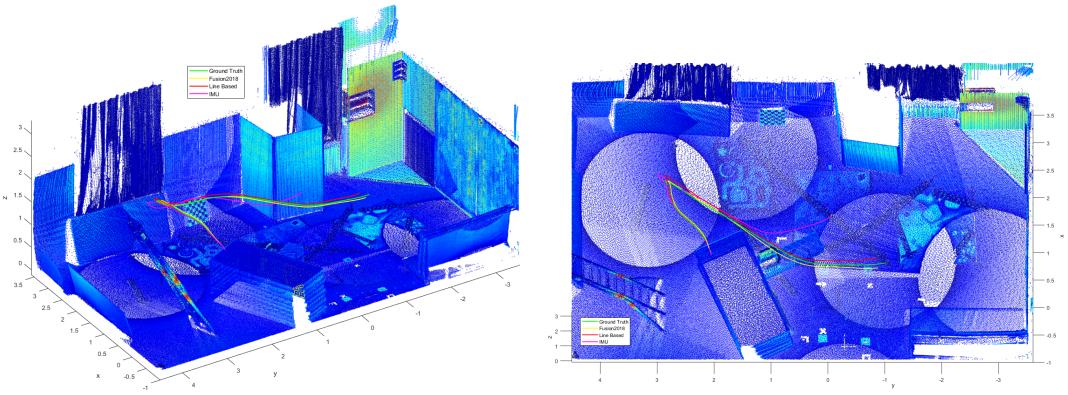


Figure 5.18: *Flight estimated by the filters and the ground truth trajectory.*



(a) *Side view of the flight estimated by the filters.* **(b)** *Top view of the flight estimated by the filters.*

Figure 5.19: *Detailed view of the flight estimated by the filters and the ground truth trajectory.*

A summary of the obtained errors is presented in Table 5.1, where the improvement of using image-based features to update the IMU predictions is evident. An aspect that should be mentioned is that the tuning parameters of the proposed filter had to change so that the filter would not diverge due to noise of not only the initializations but also to the difficulty of accurately identifying the same endpoints of lines in the dataset.

Table 5.1: *Average RMSE of the filters tested on the EuRoC MAV dataset.*

Filter	Mean Position error [cm]	Mean Rotation error [deg]
IMU Only	11.18	0.117
Fusion2018	1.42	0.108
Line Base	2.15	0.091

The proof of concept presented shows that lines can be used with success to perform Visual Inertial Odometry. An exciting observation is that the filter can maintain a low error value even with fewer landmarks since only 10 lines are used (20 points) compared to the 30 used by the Fusion2018 filter. This reduction in the state size may allow using a higher number of updates and a higher frequency camera since the update step scales poorly with the increase of landmarks, having a complexity of $\mathcal{O}(LM^2)$, where M is the observation dimension and L one of the dimensions of the state.

The line-based approach achieved worse results when compared to the Fusion2018 filter for the position error, explained by the difficulty of providing noiseless endpoints of the lines. In contrast, the line-based method reduces the rotation error since lines supply directional information not provided by points.

Another promising result is observed by viewing the initialized lines in Figure 5.13 without the ground truth point cloud, where it is possible to get a general idea of the environment while using a limited number of lines when compared to the point cloud, that for reference, uses over 3 million points. In contrast, in Figure 5.20, there are only 4 hundred lines, and most of the lines could be filtered by proximity since they are almost above each other. This problem comes from the matching process where 1 to 1 matches are made, making it impossible to match a line against multiple lines from a subsequent image. These results are aggravated in some zones due to the confidence interval used in line detection, which in some areas could be higher, giving fewer repeating lines.

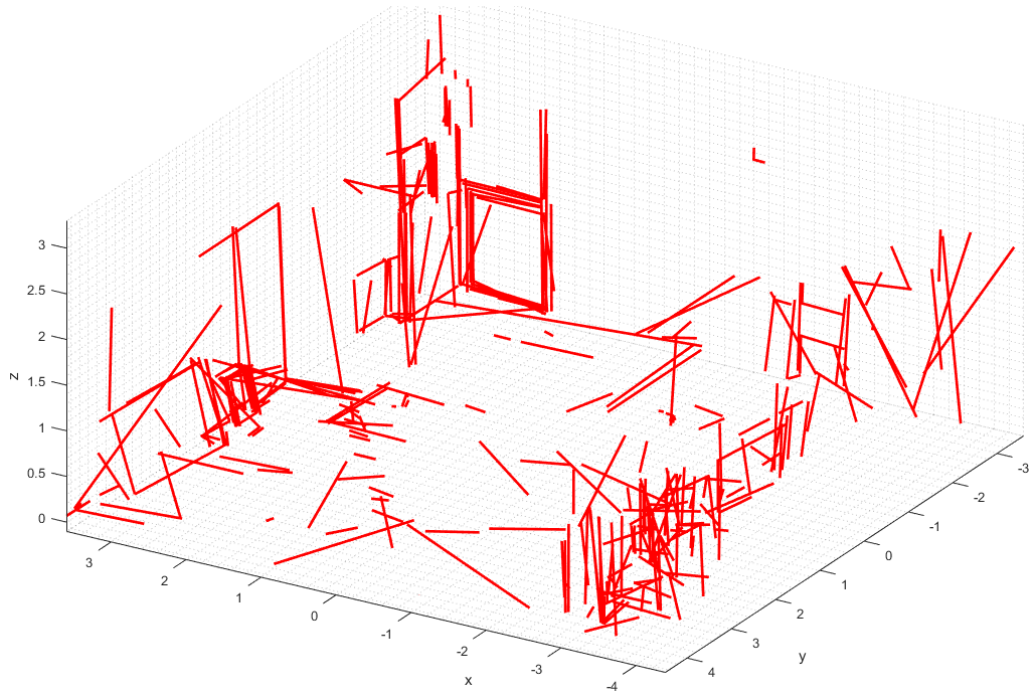


Figure 5.20: *Line Initialization using the matching and Initialization methods proposed and using the ground truth for the robot's pose without the ground truth point cloud. It gives a general map of the environment while using around 4 hundred lines compared to over 3 million points that the point cloud uses.*

Chapter 6

Conclusion and Future Work

This chapter concludes the thesis by presenting an overview of the work developed, the methods proposed, and details possible interesting paths for a future exploration. Section 6.1 provides the conclusion of the thesis and Section 6.2 explores future possibilites to expand the work done by this thesis.

6.1 Conclusion

This work proposed to give the needed building blocks to achieve a SLAM solution using VIO techniques with Lines as features. It was motivated by new advances in machine learning that facilitated the detection of lines in images that could provide a richer mapping visualization tool than classical point features.

The main obstacle encountered was the matching of lines through multiple frames to be initialized as 3D structures. An ICP variation was introduced to handle this problem, where the main focus of the algorithm was to match the features instead of finding the transformation between different views. This solution proved to be very successful in increasing the number of matches by 17 for each image pair. It also reduced the percentage of images with less than 20 matches from 20% to less than 5%.

It was also shown that the method proposed to initialize lines using Plucker coordinates did not provide a great response to noise and rapidly diverged. As such, a method was proposed that minimizes the distance of the 3D point to back-projected rays; this method shows a robust and predictable response to the tested types of noise.

Finally, it was found that some zones of the environment present few lines, which leads to a potential loss of lines to be initialized and leads the filter to diverge in the worst-case scenario; it is shown that the filter can survive through brief moments through those zones, but some adaptations may be needed to surpass these obstacles.

As a proof of concept, the proposed approach provided a sound localization of a drone through a somewhat complex trajectory and provided some line initializations which could potentially maintain alive the filter. It was shown that lines provide a rich descriptor of an environment compared to a point cloud and that the line algorithms proposed could give a first approximation of the map without the need for filtering processes in cases where the localization is provided.

6.2 Future Work

A mixed approach involving points and lines may solve the problem of zones where the filter cannot detect lines. A parallel map or even filter, which uses the robot's pose from the primary filter, can save or propagate the point features. In the zones, with too few visible lines, the point features would substitute the line features. Another approach that may be simpler is a truly mixed filter where, e.g., the first 20 points would correspond to line feature (10 lines), and the last 10 points use classical features and descriptors.

Another aspect that could be improved is giving each line the possibility to match multiple lines in a subsequent image, increasing the probability that features are kept alive during the filter propagation and providing a method not to initialize repeated lines.

Finally, more techniques that take advantage of the point independent Plucker line coordinates should be researched so that lines that do not show endpoints in the frame can be initialized without the fear of introducing a wrong landmark to the system. Following this vein, directly propagating line coordinates inside the filtering process would be expected to improve the precision of the lines detected. This approach would require more complete external maps, where lines may be composed of multiple segments, but would also improve the resilience of the navigation system upon harder (lesser) visual data.

Bibliography

[Barrau & Bonnabel, 2015] Barrau, A. & Bonnabel, S. (2015). An EKF-SLAM algorithm with consistency properties. *Computing Research Repository (CoRR)*, abs/1510.06263.

[Bartoli & Sturm, 2005] Bartoli, A. & Sturm, P. (2005). Structure-From-Motion Using Lines: Representation, Triangulation and Bundle Adjustment. *Computer Vision and Image Understanding*, 100, 416–441.

[Brossard et al., 2017] Brossard, M., Bonnabel, S., & Condomines, J. (2017). Unscented Kalman filtering on lie groups. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2485–2491).: IEEE.

[Burri et al., 2016] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., & Siegwart, R. (2016). The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10), 1157–1163.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 679–698.

[Chen & Medioni, 1992] Chen, Y. & Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3), 145–155. Range Image Understanding.

[Davison, 2003] Davison (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision* (pp. 1403–1410 vol.2).: IEEE.

[Gower & Dijksterhuis, 2005] Gower, J. & Dijksterhuis, G. (2005). Procrustes problems. *Procrustes Problems, Oxford Statistical Science Series*, Vol. 30.

[Hartley & Zisserman, 2000] Hartley, R. & Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press.

[Heikkila & Silven, 1997] Heikkila, J. & Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 1106–1112).: IEEE Comput. Soc.

[Hough, 1959] Hough, P. V. C. (1959). Machine Analysis of Bubble Chamber Pictures. *International Conference on High Energy Accelerators and Instrumentation*, 590914, 554–558.

[Julier & Uhlmann, 1997] Julier, S. J. & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In I. Kadar (Ed.), *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068 (pp. 182 – 193).: International Society for Optics and Photonics SPIE.

[Kalman, 1960a] Kalman, R. (1960a). Contributions to the theory of optimal control. *Boletín de la Sociedad Matemática Mexicana*, 5(2), 102–119.

[Kalman, 1960b] Kalman, R. E. (1960b). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1), 35–45.

[Li et al., 2020] Li, H., Yu, H., Yang, W., Yu, L., & Scherer, S. (2020). ULSD: Unified line segment detection across pinhole, fisheye, and spherical cameras.

[Lowe, 1999] Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2 (pp. 1150–1157 vol.2).: IEEE.

[Montiel et al., 2006] Montiel, J. M. M., Civera, J., & Davison, A. J. (2006). Unified inverse depth parametrization for monocular SLAM. In *Robotics: Science and Systems*: Robotics: Science and Systems Foundation.

[Munkres, 1957] Munkres, J. R. (1957). Algorithms for the assignment and transportation problems. *Journal of The Society for Industrial and Applied Mathematics*, 10, 196–210.

[Mur-Artal & Tardos, 2017] Mur-Artal, R. & Tardos, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5), 1255–1262.

[Newell et al., 2016] Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. *European conference on computer vision*, (pp. 483–499).

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

[Park, 1995] Park, F. C. (1995). Distance metrics on the rigid-body motions with applications to mechanism design. *Journal of Mechanical Design*, 117(1), 48–54.

[Perdices et al., 2014] Perdices, E., Lopez-Ramos, L., & Cañas Plaza, J. (2014). LineSLAM: Visual real time localization using lines and ukf. In *ROBOT2013: First Iberian Robotics Conference*, volume 252 (pp. 663–678). Springer International Publishing.

[Schönemann, 1966] Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1), 1–10.

[Shi & Tomasi, 1994] Shi, J. & Tomasi (1994). Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 593–600).

[Smith et al., 2006] Smith, P., Reid, I., & Davison, A. (2006). Real-time monocular SLAM with straight lines. In *Procedings of the British Machine Vision Conference 2006* (pp. 17–26).: British Machine Vision Association.

[Solà et al., 2018] Solà, J., Deray, J., & Atchuthan, D. (2018). A micro lie theory for state estimation in robotics. *Computing Research Repository (CoRR)*, abs/1812.01537.

[Solà et al., 2011] Solà, J., Vidal-Calleja, T., Civera, J., & Martinez-Monti, J. M. (2011). Impact of landmark parametrization on monocular EKF-SLAM with points and lines. *International Journal of Computer Vision*, 97(3), pp.339–368.

[Van der Merwe & Wan, 2001] Van der Merwe, R. & Wan, E. (2001). The square-root unscented Kalman filter for state and parameter-estimation. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 6 (pp. 3461–3464 vol.6).: IEEE.

[Wan & Van Der Merwe, 2000] Wan, E. A. & Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)* (pp. 153–158).: IEEE.