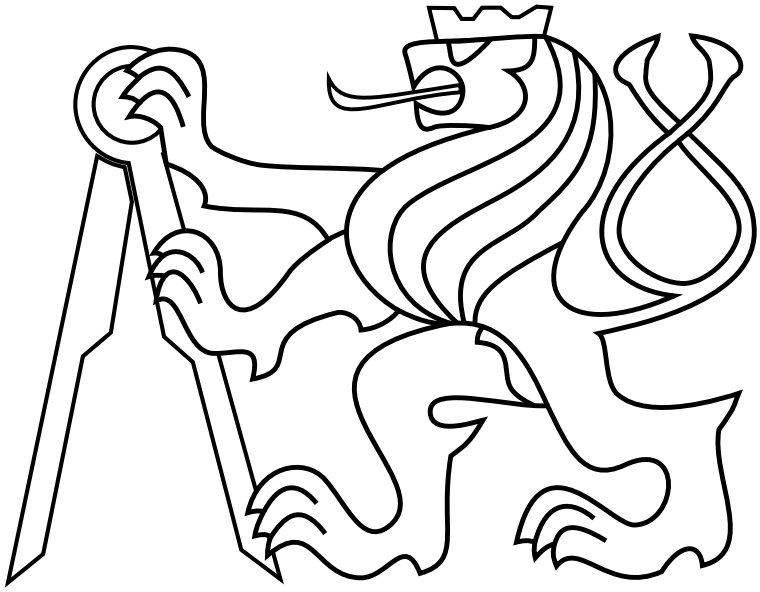
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS



Jan Macalík

Reconstruction of a 3D Scene from a Monocular Camera for Autonomous Unmanned Aerial Vehicles

Department of Cybernetics

Thesis supervisor: **Ing. Matěj Petrlík**

Prague, MAY 26, 2023

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 26, 2023 Jan Macalík

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Macalík Jan** Personal ID number: **492149**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Reconstruction of a 3D Scene from a Monocular Camera for Autonomous Unmanned Aerial Vehicles

Bachelor's thesis title in Czech:

Rekonstrukce 3D scény z monokulární kamery pro autonomní bezpilotní helikoptéry

Guidelines:

The focus of this thesis is to research, evaluate and compare algorithms for 3D scene reconstruction from a stream of monocular camera images. The following tasks will be solved:
• Familiarize yourself with the monocular scene reconstruction problem [1] and conduct research of state-of-the-art methods (e.g. [2,3]).
• Evaluate the performance of the selected methods on the open-source EuRoC dataset [4], which contains indoor flight sequences of a UAV equipped with cameras with provided ground-truth pose and structure.
• Integrate the methods into the Multi-robot Systems Group UAV system [5].
• Verify the functionality in the Gazebo simulator, discuss performance and parameter selection.
• Compare the methods on data from real-world flights, discuss results.

Bibliography / sources:

[1] Häming, Klaus, and Gabriele Peters, "The structure-from-motion reconstruction pipeline–a survey with focus on short image sequences." Kybernetika 46.5 (2010): 926-937.
[2] W. N. Greene and N. Roy, "FLaME: Fast Lightweight Mesh Estimation Using Variational Smoothing on Delaunay Graphs," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 4696-4704.
[3] A. Rosinol, et al., "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 1689-1696.
[4] Burri, Michael, et al., "The EuRoC micro aerial vehicle datasets.", The International Journal of Robotics Research 35.10 (2016): 1157-1163.
[5] Tomas Baca, et al. "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles". Journal of Intelligent & Robotic Systems 102(26):1–28, May 2021.

Name and workplace of bachelor's thesis supervisor:

Ing. Matěj Petrlík Multi-robot Systems FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **31.01.2022** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **30.09.2023**

Ing. Matěj Petrlík	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

First and foremost, I would like to thank my advisor, Ing. Matěj Petrlík, for his professional guidance and kind support followed by enormous patience. With his skills and wast knowledge of the researched topic, I was able to get through this challenging process. I also appreciate his availability and great effort he made to advise me on this project by all available means.

I would also like to thank all members of the MRS group for their assistance and friendly support. Furthermore, I extend my gratitude to the Faculty of Electrical Engineering for providing an excellent education, which proved to be crucial to my success.

I also wish to thank my parents, Zuzana and Tomáš, for their belief in my capabilities and for their love, moral support, and constant encouragement.

Without all the mentioned support, it wouldn't be possible for me to finish this thesis or remain sane during the process.

Abstract

One of the recently established fields of science is the development of unmanned aerial vehicles (UAVs), which are able to perform tasks in various environments without the need of human assistance. One key component to solve this task is the reconstruction of a 3D scene of an unknown environment, where the UAV needs to localize and safely navigate itself. One way of creating this 3D model is based on frames from a monocular camera attached to the UAV and the data from sensors of the inertial measuring unit (IMU). The advantage of this combination is availability and a low price compared to other setups for example with light detection and rangings (LIDARs). Thanks to the reconstructed 3D model, the UAV is then able to follow a safe trajectory through a real environment.

This work deals with two open-source algorithms, FLaME and Kimera, which process information from images captured by a monocular camera and use structure from motion (SfM) algorithm to reconstruct a 3D model of an unknown environment. The algorithms were evaluated and compared on publicly available EuRoC datasets and subsequently, their functionality was verified in a simulator that is used and developed by the multi-robot systems (MRS) group at the Department of Cybernetics, Czech Technical University (CTU). Finally, experiments in a real environment were conducted to evaluate algorithms for practical use.

Keywords: unmanned aerial vehicle, monocular camera, 3D scene reconstruction, structure from motion, monocular simultaneous localization and mapping

Abstrakt

Jedním z nedávno vzniklých odvětví vědy je vývoj autonomních bezpilotních prostředků (UAVs), které jsou schopny vykonávat úkoly v různých prostředích bez nutnosti lidského zásahu. Mezi části této problematiky patří rekonstrukce 3D scény neznámého prostředí, aby se v něm UAV dokázala lokalizovat a bezpečně pohybovat. Jeden ze způsobů, jak takový 3D model získat, je pomocí snímků z monokulární kamery připevněné na UAV a údajů o pohybu ze senzorů inerciální měřicí jednotky IMU. Výhodou této kombinace je dostupnost a nízká cena v porovnání s ostatními sestavami jako například s LIDARem. Díky výslednému 3D modelu je pak UAV schopna sama naplánovat bezpečnou trajektorii vyhýbající se všem překážkám a aplikovat ji v reálném prostředí.

Tato práce se zabývá dvěma open source algoritmy, FLaME a Kimera, které zpracovávají informace ze snímků pořízených monokulární kamerou a pomocí algoritmu structure from motion (SfM) vytváří 3D model neznámého prostředí. Algoritmy byly nejdříve otestovány na veřejně dostupných EuRoC datasetech a následně byla ověřena jejich funkčnost v simulátoru používaného a vyvíjeného skupinou MRS na katedře kybernetiky ČVUT. Nakonec byly provedeny i experimenty v reálném prostředí pro ověření reálného použití.

Klíčová slova: autonomní bezpilotní helikoptéra, monokulární kamera, rekonstrukce 3D scény, struktura z pohybu, monokulární lokalizace a mapování

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

The latest improvements in computational technology enabled to produce small, power efficient and computationally powerful devices such as micro-controllers, central processing units (CPUs) or graphics processing units (GPUs). This paved the way to a massive development in the area of unmanned aerial vehicles (UAVs), which are limited in the weight of the carried payload and have a constrained power supply. Equipped with these small and modern devices that can provide sufficient computing power, the UAVs are then able to process a large amount of data from onboard sensors and use them for localization, mapping or detection of obstacles and interesting objects.

There is a wide variety of UAV spanning from very large drones that can replace a piloted aircraft, to smaller and lighter types of UAVs such as multi-rotor UAVs that can perform a lot of interesting and diverse tasks, which are covered in the next section.

1.1 Motivation

The number of rotors and length of the arms of the UAVs depend on the situation and the task the drone is meant to do. Larger types of multi-rotor UAVs with more motors and longer arms are more stable and therefore are suited for carrying heavier and often expensive equipment for example professional cameras for filming or larger sensors, that can provide very precise data from the surroundings.

(a) Cave Býčí skála [5] (b) DARPA SubT challenge [6]

Figure 1.1: Examples of multirotor UAVs deployed undeground

Smaller UAVs have the advantage of lower weight and therefore smaller inertia when flying. This makes them able to perform agile maneuvers and effectively avoid any obstacles that could get in their way. Thanks to their small size, they can also fly through narrow passages or tight man-made environments which makes them well suited for reconnaissance missions in difficult terrains.

Such example are missions in underground environments such as caves or human-made structures. Successful testing experiments were conducted by the multi-robot systems (MRS) group in Southern Moravia Karst cave system Býčí skála [5] — figure 1.1a. The results proved, that the drones were able to map the cave system well and could be very helpful in further cave exploring. Another example of an underground task was at the DARPA SubT challenge [7, 6] — figure 1.1b, where the drones had to explore tunnel, cave and urban environments including a subway station-like structure.

The UAVs can be used also in emergency situations, where their efficiency was also proven in various tasks: the fire extinguishing in high-rise buildings on MBZIRC 2020 international challenge [8] — figure 1.2a and also during mapping and search in the earthquake-damaged buildings in Japan [9].

Swarms of UAVs are also useful if trying to cover larger areas, however the communication and coordination becomes more challenging with the increasing number of drones. Very interesting approach is the UVDAR project [10], where there is no centralized control system and the drones are using only blinking ultra violet (UV) light emitting diodes (LEDs) for both mutual localization and basic communication (by changing the frequency of the blinking). This system is very robust and can be used in real world conditions with changing illumination, presence of undesirable light sources or their reflections, because the blinking UV light can be easily separated from the background.

Another interesting field of usage is documentation of historical buildings and monuments covered by the project Dronument [11] — figure 1.2b. The UAV is used to map the interior or exterior of the object including remote areas that are difficult for humans to reach and would require the scaffolding. Usually, two drones in a formation are used: one

carrying a light source and the other the camera. This whole approach drastically reduces the required effort and time from number of months to only a few hours.

Concluding all the results from these projects, the UAVs are starting to play an important role in helping humans in solving a lot of often dangerous tasks and also making the work much more effective and less time consuming.

(a) MBZIRC 2020 Challenge [8] (b) Mapping remote areas in a church [11]

Figure 1.2: Examples of another multirotor UAV applications

1.2 Introduction to the problematic

In many tasks the UAV has to enter an unknown environment. In that case, it has no information about the layout of the surroundings and possible obstacles in the way. To be able to generate a safe trajectory and navigate through such environment, the UAV must be able to create a 3D structure of its surroundings. There are several ways of reconstructing it using a light detection and ranging (LIDAR), stereo pair of cameras, depth camera or just a single camera. The solution with the single camera has the advantage of being the most lightweight and the most affordable.

This work focuses on FLaME [3] and Kimera [12], two state-of-the-art methods that employ the single-camera solution. Both methods utilize the structure from motion (SfM) approach, where an image stream from a monocular camera onboard the UAV is used to reconstruct a 3D model. This is achieved by analyzing the image differences caused by motion of the camera between subsequent frames. The resulting model then provides vital information for planning safe trajectories.

1.3 Thesis outline

This chapter has introduced the broad range of UAV applications and touched on the problematic of reconstructing models of unknown environments. In the following chapters, we will delve deeper into this topic.

Chapter 2 presents state-of-the-art approaches that are relevant to this problem and are closely related to evaluated algorithms. Following this, the chapter 3 introduces the theoretical foundations for the evaluated methods FLaME and Kimera. These methods are further described in chapter 4. The implementation of the evaluation framework is outlined in chapter 5.

Chapter 6 discloses the results and performance assessments. The FLaME and Kimera methods were evaluated using publicly available EuRoC datasets, the UAV system with Gazebo simulator that is used by the MRS group [13] and data from a real-world UAV flight.

The results and outcomes from this thesis are discussed in chapter 7. Also, the possible future work is mentioned.

Chapter 2

State-of-the-art approaches

Contents

The object of this chapter is to outline the available cutting-edge methods and approaches that are utilized during the process of the 3D scene reconstruction from the monocular camera output. These are either integrated into one of both of the examined methods — FLaME and Kimera — or are closely related to them. The chapter begins with the description of SfM and simultaneous localization and mapping (SLAM). The rest of this chapter is devoted to the visual-inertial odometry (VIO) methods.

2.1 Structure from motion

SfM is utilized in both FLaME and Kimera. In general, SfM uses camera frames from varying positions, typically during motion, to reconstruct a 3D structure. The most common implementations of SfM, found in both tested methods, assume that the scene is static and the successive camera frames are ordered with only minor movements in between. Nonetheless, there are also alternatives to this approach. For instance, if the scene contains motion, there are methods that can estimate this movement from the optical flow in the images [14] and share this information with the SfM. If the set of images is unordered, the SfM can still be used, however it may result in larger error due to the increased difficulty in feature selection and matching. These types of SfM can be used for example on photographs of the same object from a photo collection [15, 16].

The outcome of this process has the form of 3D points, commonly referred to as a point cloud. Typically, the SfM doesn't need to be executed in real time, and is therefore often designed to produce a high-quality 3D structures, a process which requires longer computation time. However, both FLaME and Kimera are primarily designed to be used on UAVs, where real-time performance is more important than high precision, due to its application in navigation and obstacle avoidance. This means that it is possible to compute it in real-time on an UAV equipped with a CPU.

Another advantage of this method is that many SfM algorithms don't require any information about the camera poses and they can determine it on its own. As such, it can effectively work in areas without a global positioning system (GPS) signal or in environments with a high level of signal noise. However, relying solely on estimated poses may impact the quality of the result point cloud.

On the other hand, some methods require the camera pose for the reconstruction or just for initialization. To estimate the camera poses, techniques such as SLAM, VIO or visual-inertial navigation system (VINS) prove to be extremely beneficial. These methods will be discussed in the subsequent sections.

2.2 Simultaneous localization and mapping

SLAM is just like the SfM a computational problem of constructing a map of an unknown environment while simultaneously keeping track of an agent's location within it. However, there is a difference between these methods. While SfM can be used for producing a high-quality mesh, SLAM is more focused on real-time performance tailored to the available resources, which can be limited. It is not expected from SLAM to produce a perfect output, but rather one that enables the operational compliance. This can include a lower quality mesh often with outliers and reconstructed trajectory from previous positions of the UAV.

SLAM methods are based on concepts in computational geometry and computer vision, and are used in robot navigation, robotic mapping and odometry for virtual reality or augmented reality. The first monocular SLAM system to operate in real-time was created in 2003 and is commonly attributed to [17]. The method is also analyzed and described in more detail in paper [18].

Another examples of SLAMs that use monocular camera are Large-Scale Direct (LSD) Monocular SLAM [19] and the multi-level mapping (MLM) SLAM [20] from the authors of FLaME. LSD-SLAM allows to build consistent and large-scale maps of the environment and the MLM is able to distinguish between plane areas and areas, that have to be reconstructed in more detail.

The SLAM methods are also often using extended Kalman filter (EKF) [21] that is well designed for processing data from areas with high noise intensity. Considering a set

of measurements with noise and inaccuracies, the real values are estimated based on the assumption that the measurements are from normal (Gaussian) distribution.

In one of the two methods examined in this thesis, namely Kimera, the estimation of the UAV position is included. Generally, it can be determined using only a combination of on-board sensors: the camera and the inertial measuring unit (IMU), which includes acclelrometers and gyroscopes capable of measuring angular rate and specific force. The method using this combination of sensors is called VIO and is discussed in the following section 2.3.

2.3 Visual-inertial odometry

In general, the process of estimation of the UAV pose within an environment, based on information from its visual sensor is referred to as visual odometry (VO). However, the camera alone cannot retrieve the scale of the scene and therefore is only capable of estimating the motion, not the true scale of the trajectory and objects in the environment. By adding accelerometer measurements, the scale of the scene can also be estimated, derived from the absolute value of the acceleration. This method is then referred to as VIO, due to its use of inertial measurements from IMU.

The VIO is useful in correcting the bias from gyroscope measurements, because the sensors accumulate an error over time, which causes a drift in the measured values. By adding a camera with VIO to the IMU, there are two independent sources of information, which improves the precision.

Another practical example of use is with very distorted or none GPS signal, allowing the UAV to fly under bridges, between buildings or even indoors. VIO can also run parallel with the GPS and just improve the error of the path estimation same as with the IMU.

However, the VIO can be constrained by a bad visibility, too intensive motion blur or by the lack of texture in the environment. In these situations, the IMU measurements can replenish the data from visual track, that would be otherwise insufficient for the pose estimate.

When it comes to the pose estimation method, the VIO most frequently uses the EKF [22, 23]. More recently, the combination with the non-linear optimization was also developed [24].

2.4 Visual-inertial navigation system

The VIO can also be used for navigation purposes similar to SLAM. A navigation system with implemented VIO is called VINS, but unlike SLAM, it inherits all advantages of the VIO such as more precise estimation.

VINS can further be used with multiple cameras to improve accuracy [25] or even across multiple UAVs [26]. Nevertheless, a monocular VINS can achieve satisfactory results as well, requiring only a minimal setup consisting of a single camera and IMU.

The VINS can use the same methods for optimization as those mentioned in section 2.3. OpenVINS [27] is an example of the method that uses the EKF. It is well documented and extendable, making it a suitable foundation for further VINS development. Another example is VINS-mono [28], which is a monocular VINS system that is presently used by the MRS group, implemented with the non-linear optimization.

Chapter 3

Theoretical foundations

Contents

This chapter provides an overview of the theoretical principles and mathematical tools that are essential for the functionality of the reconstruction methods. To configure these methods for minimal error while creating the 3D mesh, it's crucial to have knowledge about the camera parameters, because these parameters determine the deformation of the scene projection on the 2D image plane. Equally important is the understanding of the used coordinate systems and how to perform the transformations between them. Such knowledge ensures the successful conversion of the resultant pointcloud into the desired coordinate frame to build a global pointcloud for evaluation.

3.1 Camera model

A camera can be characterized as a sensor that captures a static image of a desired scene. Mathematically, this can be described as mapping from 3D coordinates onto a 2D image plane. However, given the unique parameters of each camera, no two mappings are

identical. In order to determine how exactly the coordinates would be transformed, we need to define a camera model that approximates the actual camera as accurately as possible.

3.1.1 Camera parameters

The camera model is described by camera parameters, which define its technical properties. There are two types of parameters: the extrinsic and intrinsic parameters.

The extrinsic parameters determine the camera position relative to the reference frame, i. e. how to transform coordinates from the world coordinate system to the camera coordinate system. To perform this transformation, the extrinsic matrix **P** (3.1) has to be created from these parameters.

$$P = \begin{bmatrix} r_{11} & r_{12} & r_{13} & v_x \\ r_{21} & r_{22} & r_{23} & v_y \\ r_{31} & r_{32} & r_{33} & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The $\mathbf{r_{ij}}$ elements correspond to the rotation matrix that is part of **P**. The fourth column represents the translation vector **v** in homogenous coordinates that connect the two origins of the coordinate systems.

The intrinsic parameters provide the transformation from the metric 3D camera coordinate system to the 2D image plane with pixel coordinates. Similarly, we need to construct the intrinsic matrix **K** (3.2) to perform this transformation. This matrix is often called the calibration matrix, because it contains all technical parameters of the camera that can be obtained by a calibration (section 3.1.2).

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The $\mathbf{f_u}$ and $\mathbf{f_v}$ are the focal lengths of the camera and the $\mathbf{c_u}$ and $\mathbf{c_v}$ are the coordinates of the principal point.

With the correct camera parameters it is possible to perform a triangulation of feature points from the camera frames with high precision.

3.1.2 Obtaining camera parameters

Generally, there are two approaches to retrieve the parameters. The first, calibration, is to measure the parameters of the used camera with maximum precision before the scene reconstruction. A calibrated camera simplifies the reconstruction process substantially, because the images from the camera are transformed to the undistorted ones with minimal

error. On the other hand, it poses a disadvantage, because once measured, the parameters remain constant during the reconstruction, so the setup must not be altered.

The second approach is to calculate the camera parameters along with the 3D points, only relying on established correspondences between the observed images. One example of this approach is described in [29]. Compared to the calibration approach, it takes more computational effort and it is less accurate. In addition, it needs a subsequent auto-calibration step to remove the projective distortion. But besides these disadvantages it features greater flexibility and ease of use.

In this work, the first approach is used. The camera parameters are either obtained from already created configuration files or we perform the calibration by ouserlves. For that, the Kalibr [30] package was used.

3.1.3 Pinhole camera model

The pinhole camera model (figure 3.1) is the most simple, idealized approximation of the camera. The aperture is described as a point with no lenses used to focus the light. Therefore, no distortion is occurring. The model also does not include the effect of blurring of unfocused objects caused by lenses and finite sized apertures. The validity of the model depends on the quality of the camera. The model error typically increases from the center of the image to the edges as lens distortion effects increase.

Some of the effects that the pinhole camera model does not take into account can be compensated, for example by applying suitable coordinate transformations on the image coordinates. Other effects are sufficiently small to be neglected if a high quality camera is used. This means that the pinhole camera model often can be used as a reasonable description of how a camera depicts a 3D scene.

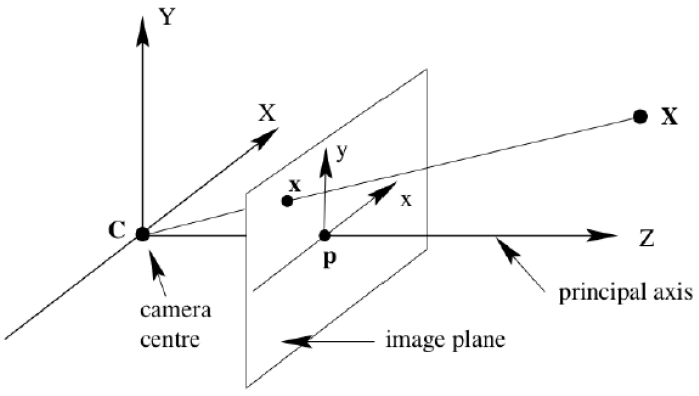


Figure 3.1: Pinhole camera model [1]

3.1.4 Distortion parameters

In many cases, the error of the pinhole camera model can be too high. As said before, the biggest source of the error comes from the distortion effects of the lens. Therefore, a compensation for this effect is needed. This can be done by undistorting the image using a distortion model. The most commonly used one is the radial distortion model. In this thesis, the equidistant distortion model is used which is suitable for the fish-eye camera which was used both in simulation and in real world setup.

The most common way to get distortion parameters of the camera is by pointing on a chessboard with the known sizes. Comparing it with the distorted image of this chessboard it is possible to get relatively precise parameters of the distortion [31]. In figure 3.2, an example of the distortion effect can be seen. The chessboard gets more curved on the sides as the effect intensity is increasing with respect to the distance from the image center.

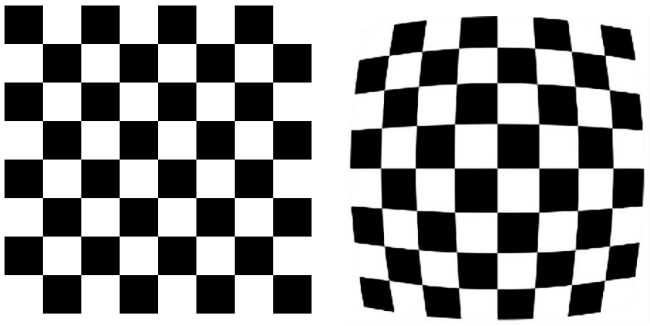


Figure 3.2: Comparison between the undistorted and distorted image from camera

3.2 Coordinate frames

During scene reconstruction, having a good coordinate representation of the present environment is vital. It can be represented by three primary coordinate frames: the world frame, the body frame and the camera frame. The world frame signifies a fixed point in the environment, such as the origin of a GPS system, a tracking ground station, or the location from which the UAV took off. The body frame usually signifies the position of the center of the UAV, which is commonly the IMU. The camera frame represents the camera's position on the UAV, with the origin is typically situated in the camera's focal point.

Both the world and body frames typically have the right-forward-up (RFU) orientation, or any other with the z-axis pointing upward. On the other hand, the camera frame uses a back-down-right (BDR) orientation, with the z axis aligned with the lens axis, that is, pointing in the direction where the camera is facing.

In this work, we also deal with a camera-world frame, which is used by FLaME. It shares the same position as the world frame, but uses the BDR orientation.

3.3 Transformations

In this work, we heavily rely on transformations between the coordinate frames such as described in 3.2. It is used to transform the pointclouds to the desired frame. The transformations are mainly performed with 3D vectors representing translations and quaternions (described in 3.3.1) representing rotations.

Nevertheless, there are also other methods such as 4 by 4 transformation matrix similar to **P**, which is explained in section 3.1. This matrix can transform any points or vectors in homogenous coordinates.

3.3.1 Quaternion

Quaternions, in a simpler form, can be represented as a 4D vector, as shown in 3.3:

$$q = [x, y, z, w] \tag{3.3}$$

The x, y and z components represent the axis of rotation in three-dimensional space. The w component, often called the scalar or real part, corresponds to the amount of rotation around this axis. By using quaternions for rotation calculations, we can avoid issues like gimbal lock that are commonly encountered with other methods, such as Euler angles. This makes quaternions a highly effective tool for transformations between coordinate frames.

3.4 Epipolar geometry

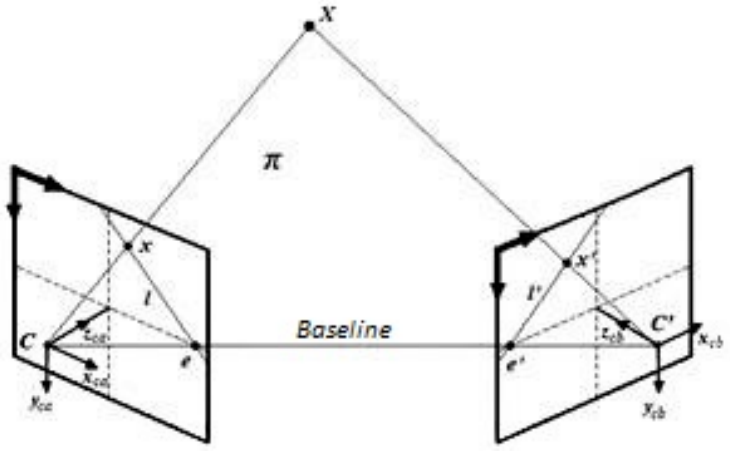


Figure 3.3: Epipolar geometry [1]

Epipolar geometry is a key component of the SfM monocular scene reconstruction method (section 2.1) that is used both in FLaME and Kimera. Leveraging this principle

allows us to effectively identify correspondences, which are the same points in space across two consecutive camera views. This concept uses the geometric relations between these views, effectively narrowing the the search of the correspondences from the entire image to just a line and its surroundings. The concept of this geometry is illustrated in figure 3.3 and is further explained in the following text.

If there is a point **X** in the space that is seen in two camera views, it can be represented as image points **x** and **x'** that are its projections onto each image plane. The point **X**, its projections and camera centers **c**, **c'** lie in the same plane π called the epipolar plane. The baseline is a line that connects the two camera centers. The intersection point of the baseline and the image plane is termed the epipole, while the the intersection of the epipolar plane with the image plane is referred to as the epipolar line.

With information about the camera's poses in 3D space (relative to the correctly set fixed world frame) and the coordinates of the image point **x**, it is possible to determine the epipolar plane π using only these points. This implies that the corresponding point **x'** must lie on the epipolar line **l'**. This characteristic can be used to restrict the search of the correspondences in the second image to the epipolar line or points within a specified proximity.

An example of epipolar lines in an image is depicted in figure 3.4 [2]. In this example, nine feature points were selected in the first image (right), followed by the computation of the corresponding epipolar lines and their transformation onto the second image (left). The result than enables to find the correspondences in the second image restricted onto the epipolar lines and finally compute the 3D coordinates of the point.

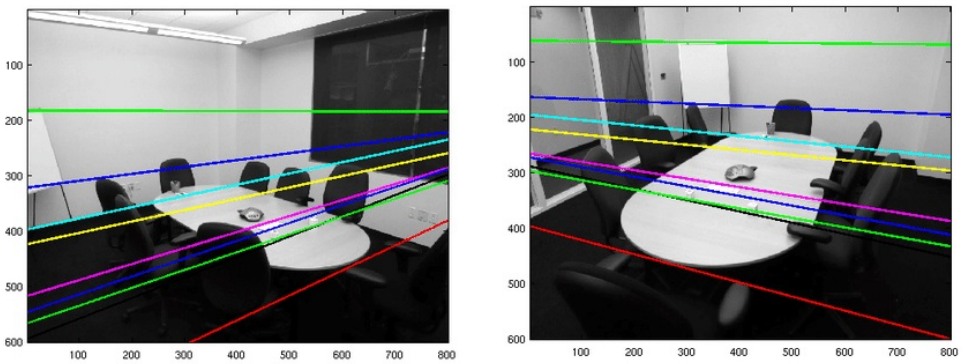


Figure 3.4: Epipolar lines from two images [2]

3.5 Feature detection

In order to construct epipolar lines and 3D points, it's essential to first identify the features in an image. Feature selection is a process in which a set of points is selected from

the image such that every point can be easily distinguished between subsequent camera frames. To identify the most suitable points, a simple metrics based on the gradient of the images can be used.

Over time, various descriptors and detectors have been developed. One of the oldest feature detectors is Good Features to Track (GFTT) [32], which is still widely used today. Among the most well-known are: Features from Accelerated and Segments Test (FAST), a feature detector that identifies features based on the pixel surroundings [33, 34]. Oriented FAST and Rotated BRIEF (ORB) [35] is a descriptor that combines the FAST detector and Binary robust independent elementary feature (BRIEF) [36] descriptor in order to benefit from the invariance to rotating and scaling. Adaptive and Generic Accelerated Segment Test (AGAST) [37], another notable detector, is build upon the FAST foundation.

The scale-invariant feature transform (SIFT) [38] descriptor is also widely used. It is robust to various changes like image scaling, translation, rotation and change of illumination. There is also more modern variant that builds on SIFT: SURF, which has much faster performance [39]. Sadly unlike ORB, these two descriptor types are patented and cannot be used in commercial applications.

3.6 Feature tracking

Once the features have been successfully obtained from the image, they are used for the triangulation to determine their corresponding points in 3D space. To accomplish this, we need to identify the same points - also known as correspondences - in different camera views. As previously mentioned in section 3.4, this can be achieved through epipolar geometry, which utilizes the geometric relationship between two consecutive camera images.

3.7 Delaunay triangulation

The Delaunay triangulation is a process for creating a structure, where all the triangulated feature points are connected into triangles to form a Delaunay graph. This graph has one important feature: no points should be inside any of circumscribed circle of the triangles in the graph (figure 3.5 left). As a result, the structure is clearly defined and optimal, because it maximizes the minimum angle of all triangles. Therefore, no narrow triangles, which would degrade its quality, are present in the structure. In the following figure 3.5, there is an example of the graph structure with circumscribed circles (on the left) and the practical usage (on the right).

The triangulated mesh thus creates a very efficient reconstruction of the environment and its computational complexity is lower compared to other algorithms that do not use this method.

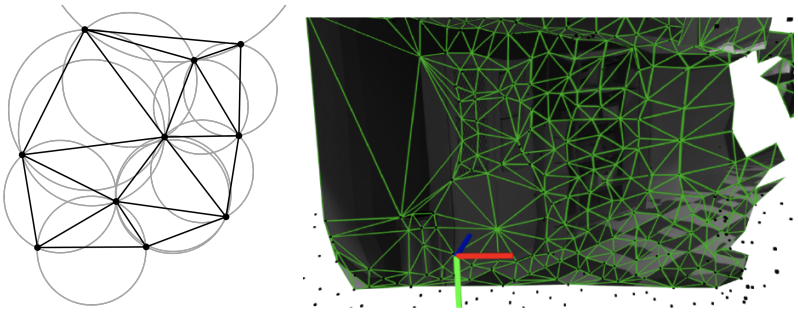


Figure 3.5: Delaunay graph structure with circumscribed circles and practical usage [12]

3.8 Variational smoothing

The methods that are using the Delaunay graph triangulation have effective and often precise output. However, there is room for further improvement as the mesh can be smoothed and cleaned from the outliers. The Variational smoothing solves exactly this problem by trying to minimize the general error of the mesh that can be represented by the function $\boldsymbol{f}$:

$$E(f) = E_{smooth}(f) + \lambda \cdot E_{data}(f) \tag{3.4}$$

The total mesh error is defined by the data error $\mathbf{E_{data}}$ which represents the amount of noise in the mesh and the $\mathbf{E_{smooth}}$ where the second-order Total Generalized Variation is used [3].

The result is then minimized by the Chambolle and Pock method [40].

The parameter λ controls the balance between those two error functions and divides the importance between the data-fitting and the smoothness.

This process is used in the FLaME algorithm. The impact of the variational smoothing can be seen in the figure 3.6. On the left, there is the standard approach to reconstruct the scene without any optimization compared with the optimized one on the right.

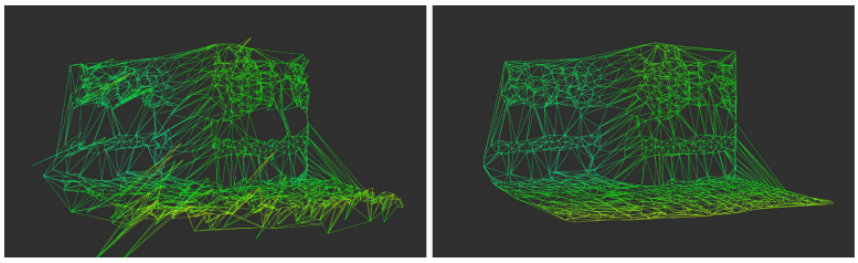


Figure 3.6: Comparison between the standard approach and the Variational smoothing [12]

Chapter 4

Evaluated methods

Contents

The aim of this chapter is to describe the selected open-source methods, FLaME and Kimera that implement the SfM and are based on previously described state-of-the-art methods (chapter 2) and the mathematical foundations (chapter 3). In the end, there is a comparison and analysis whether they are suitable for the usage in UAV control in unknown terrain with obstacles where the fast response and low computational complexity is required.

4.1 FLaME

The FLaME - Fast Lightweight Mesh Estimation [3] is a method based on the SfM which creates a 3D mesh of the surrounding environment from the camera frames taken in the motion (section 2.1). The mesh network is created from selected points in the image and their depth is estimated using an optimization algorithm (Chambolle and Pock [40]). The advantage of this 3D network is that it can be updated and expanded in real time using consecutive images captured by the monocular camera, while maintaining its optimality.

The figure 4.1 shows how the algorithm proceeds when creating a 3D mesh from the camera image and pose.

Firstly, the feature selection (section 3.5) is applied. Subsequently, all points are connected into triangles to form the optimal structure Delaunay graph as described in section 3.7. Furthermore, the Variational smoothing optimization (section 3.8) can be additionally applied on the structure.

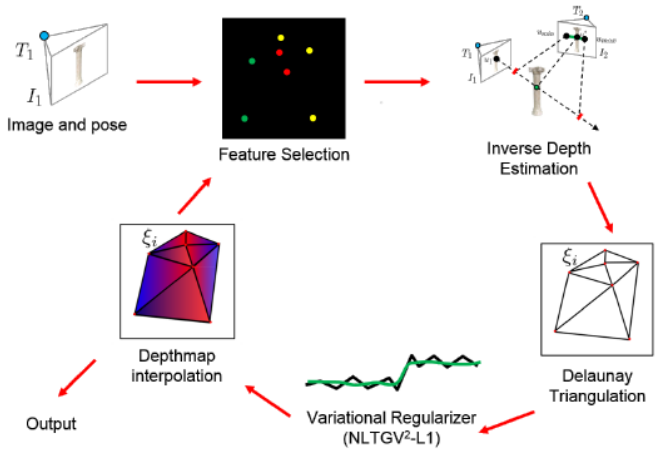


Figure 4.1: Overview of FLaME algorithm steps [3]

The network thus creates a very efficient reconstruction of the environment and its computational complexity is lower compared to other algorithms that do not use this method, such as Large-Scale Direct (LSD)SLAM [19] or multi-level mapping (MLM) [20]. The comparison can be seen in the following table 4.1:

Performance on Benchmark Datasets

		DM	RE [%]	AD [%]	Cores	Time [ms]	FPS [Hz]
TUM	LSD	181	18	19	2.5	16	62
	MLM	103	12	32	2.1	17	57
	L=3	**4950**	8.5	**54**	2.0	16	61
	L=4	**4950**	6.8	**54**	1.7	7.3	136
	L=5	**4950**	**6.6**	51	**1.4**	**4.2**	**236**
EuRoC	LSD	874	18	17	1.6	16	61
	MLM	734	17	25	1.0	14	69
	L=3	**12595**	12	36	1.3	13	78
	L=4	**12595**	11	**37**	1.2	7.0	143
	L=5	**12595**	**10**	33	**0.8**	**4.3**	**230**

Table 4.1: Efficiency comparison of FLaME with LSD SLAM and MLM [3]

The parameter L in the table corresponds to the density of points in the network (3 - densest, 5 - thinnest). FLaME can produce quite bigger number of depth maps (DM), while also maintaining lower relative inverse depth error (RE) and a higher density of accurate inverse depths of points, i. e. their distance from the camera (AD). "Time" represents the duration it took to process one frame and it is also substantially lower, mainly when L is 4 and 5.

Thanks to short single-frame processing time in order of miliseconds and Delaunay graph features, FLaME is able to provide accurate 3D reconstruction of the surroundings during each frame, which is very useful for the UAV navigation in an environment with unknown obstacles, where the lowest latency possible needs to be achieved.

4.2 Kimera

The Kimera algorithm is quite similar in its design to FLaME, differing only in a few improvements, such as the division of the program into separate modules, so that the whole algorithm can be set to provide for example only the visual odometry (VIO). In addition to FLaME, Kimera is also equipped with semantic segmentation, which is able to recognize and label individual types of objects in the resulting 3D mesh and color them with different colors. The colored mesh can be seen in the figure 4.2.

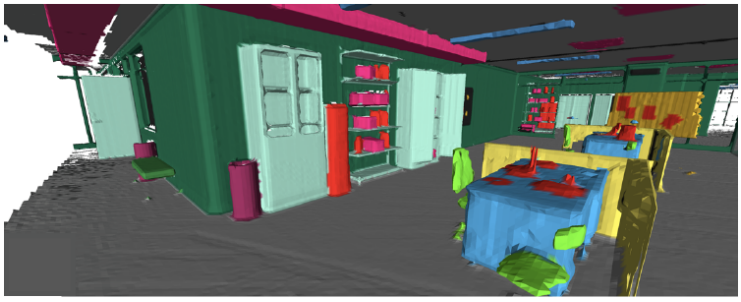


Figure 4.2: Example of semantic segmentation in 3D mesh [3]

Kimera contains a total of 4 modules that can work all together when using the algorithm at full scale or they can be combined in various ways, as can be seen on the figure 4.3. These 4 modules are:

- **Kimera-VIO: Visual-Inertial Odometry Module** - the visual-inertial odometry, that was dedcribed in section 2.3, is performed in this module. The depth of the obtained points in the image is estimated and also the position of the UAV relative to them is determined.
- **Kimera-RPGO: Robust Pose Graph Optimization Module** - this module is used for the optimisation of the reconstructed mesh. Outliers that do not contribute to the structure representation are removed and loop closures are detected, if the UAV visits the same place more than once.
- **Kimera-Mesher: 3D Mesh Reconstruction** - creates both a spatial mesh of all captured images and a mesh for each individual image. Although this module doesn't produce as accurate mesh as the following module and is more noisy as well, its computing time is very low (15 ms in average).
- **Kimera-Semantics: Metric-Semantic Segmentation** - a module that generates higher quality mesh and complements it with semantic labels that describe the type of surrounding objects. Labels are first determined for individual images, which are then used to mark labels in the overall 3D mesh. This whole process requires an order of magnitude more time to compute (0.1s on average), which is suitable for the use

in real-time.
To determine labels, the neural networks for semantic segmentation [41] are used. The probability of all labels for each point of the mesh is estimated and then the most probable one is assigned to it. Thanks to this, it is possible to precisely determine the shape of the recognized objects.

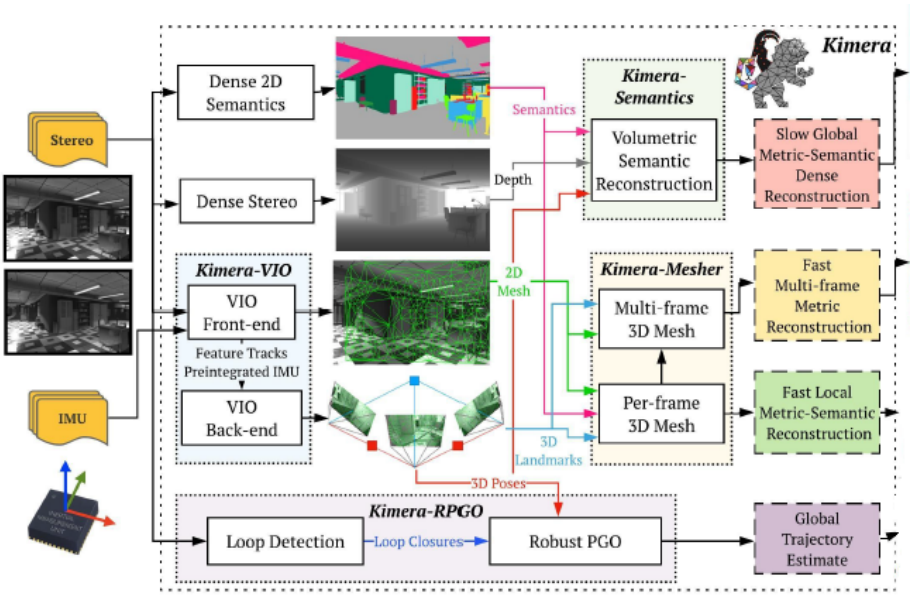


Figure 4.3: The architecture of the Kimera pipeline [3]

In order for the algorithm to work in an environment where low latency and fast response are essential, we decided to use the Kimera-Mesher module with low computational complexity. The Kimera-Semantics module is redundant for this application, because its computational speed is slower and when avoiding an object it is usually not necessary to have a semantic labeled mesh.
Therefore, the combination of the first three modules without the Kimera-Semantics module, which is not necessarily needed, seems to be the most suitable combination that saves additional computing power on CPU as a result.

Chapter 5

Implementation and used libraries

Contents

In this chapter, we discuss the implementation of the evaluation framework, along with other related implementations. The evaluation framework is specifically designed to handle corrections or transformations of input data to the evaluated methods, which can include image streams, camera information, IMU measurements and ground truth odometry data. It also facilitates processing the output of the evaluated methods, which predominantly consists of 3D polygon mesh. Rest of the implementations aim to enhance the performance of the methods, expand the scope of evaluation or facilitates the process of implementation.

5.1 Used libraries

Because the examined methods are designed to work on the UAVs, it is suitable to use the robot operating system (ROS) for managing the data streams and all the communication. We also use the OpenCV library, which is suitable for computer vision, here especially for image undistortion. We also use the PCL library for the construction of transformations of the method output meshes to form a global point cloud.

5.1.1 ROS

ROS is popular open-source library that is characteristic for the division of the program into so-called nodes, that solve specific part of the problem. The nodes form a network, where they can communicate by message interface provided by ROS. The communication between nodes can be recorded and stored in a file format called rosbag, which enables offline data analysis and simulation replay. ROS also provides an rosservice utility, which enables nodes to exchange requests and responses, allowing for the execution of specific functionalities or operations through service calls.

During the development process, the Rviz visualization package in ROS proved to be an essential and versatile utility. It offers a comprehensive set of tools for visualizing various types of data, including images, coordinate frame transformations, and point clouds. This capability proved invaluable for developing the framework and verifying the performance of the system, enabling effective debugging and validation of the implemented functionalities.

For the transformation between the coordinate systems (described in section 3.2), the tf2 ROS library is used. It allows to transform anything transformable between any two coordinate frames at any desired point in time.

5.1.2 OpenCV

OpenCV, which stands for Open Source Computer Vision Library, is a comprehensive library of programming functions mainly aimed at real-time computer vision. This library, is highly versatile and is used extensively in areas such as interactive art, stitching maps on the web, robotics, and more. In this work, we use the cv::fisheye::undistort function, which is particularly useful when dealing with images taken by a fisheye lens camera. This function is used to correct the distortions caused by the fisheye lens, converting the distorted image into a standard perspective image. By doing so, it enables us to use the methods with externally undistorted image stream as an input.

5.1.3 PCL

The point cloud library (PCL) is an open-source library dedicated to the processing of 3D point cloud data. The library contains numerous state-of-the-art algorithms for filtering, feature estimation, surface reconstruction, 3D segmentation, and more. PCL is designed to be highly efficient and scalable, being able to handle large volumes of data, and it is compatible with various sensor data types and formats. The modular architecture of PCL allows developers to easily adapt and use specific components for their needs. In this work, we leverage PCL's tools to process the output point clouds from the evaluated methods and to load reference point clouds. Both these point clouds are then visualized in Rviz.

5.1.4 Kalibr

The Kalibr [30] package is a robust and versatile toolbox developed for calibrating single or multiple cameras, as well as inertial sensors. It's an open-source tool, widely used in the robotics and computer vision communities. Kalibr allows for the calibration of both intrinsic camera parameters such as focal length and optical center, as well as extrinsic parameters like relative pose between sensors. This calibration process is crucial in systems involving multi-sensor fusion, such as UAVs and autonomous vehicles, where a high degree of accuracy and precision is required. Kalibr offers advanced algorithms and techniques to perform calibration in a consistent and efficient manner, making it a valuable tool in many projects.

5.2 Implemented framework

The implementation is written in the C++ programming language, leveraging its unique features. The C++ has a support in ROS and offers faster performance and better efficiency compared to languages such as Python. Hence, it stands as the optimal choice for real-world UAV applications and related use cases. Furthermore, a significant majority of the popular libraries are written in C++, enhancing its utility in this context.

The work was developed on Ubuntu 20.04 Linux Operating system (OS), which supports the newest version of ROS — Noetic, which is compatible with the MRS UAV system (section 6.3) used for the simulation. The FLaME and Kimera had been developed on Ubuntu 16.04, but after some configuration, it was possible to use them in the newest version.

The implemented evaluation process could be described like following: Input from a rosbag or simulation is processed before being passed to either FLaME or Kimera. Processing may include undistorting the image stream, correcting the values of camera parameters (as described in 3.1.1), distortion parameters (section 3.1.4) or discarding data with the identical time-stamps, which may disrupt the evaluated pipelines.

After the reconstruction process by the evaluated method, the output in the form of polygon mesh is further processed. Initially, 3D points are extracted and, if necessary, transformed to the coordinate system of ground truth. The transformation values are obtained either directly from simulation or extracted from the odometry data published by the rosbags. The transformed 3D points are then concatenated into a global point cloud, which constitutes the final result.

Throughout the entire process, the result cloud and the positions of coordinate frames are visualized and can be observed in Rviz. A reference point cloud is also published and visualized to verify the proper functioning of the methods.

Finally, once the rosbag record ends or the tracking of the planned path in simulation is completed, the resultant point cloud is saved. This cloud is subsequently evaluated using two metrics: mean distance to reference (MDR) with standard deviation and mean map entropy (MME).

The overall framework is illustrated in the diagram shown in figure 5.1.

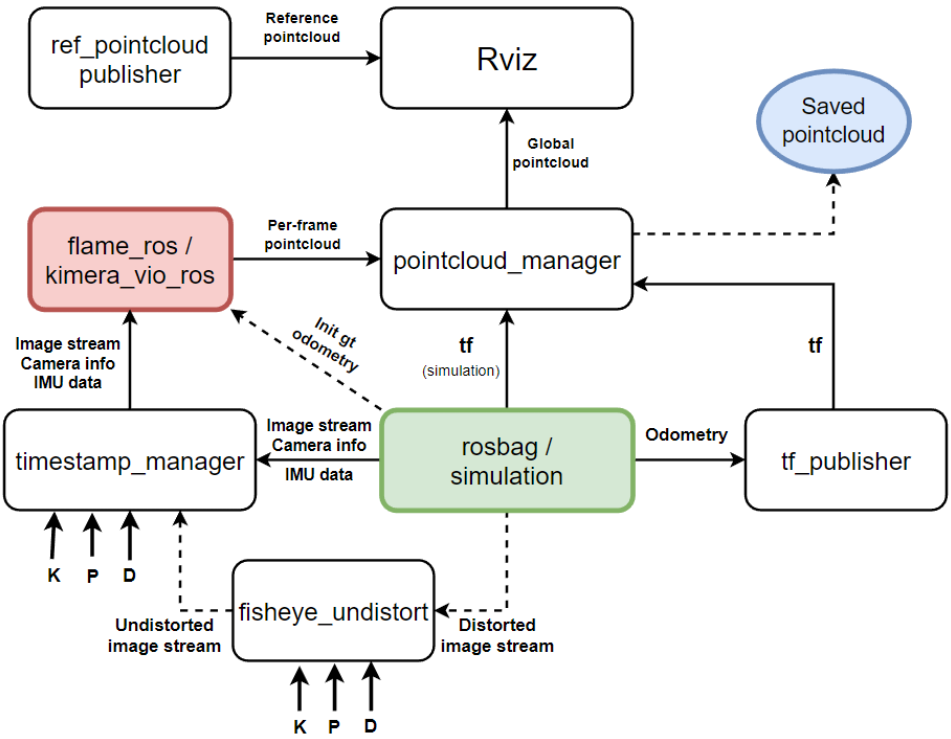


Figure 5.1: Diagram of the evaluation framework

5.3 Extending the evaluated methods

In Kimera, the primary emphasis is on stereo reconstruction, with the monocular variant considered more as an additional extension. This also reflected in the implementation, which didn't provide any module for 3D polygon mesh reconstruction. However, the authors were clearly intending to extend it to the monocular variant as well, so it was possible to implement it into this method.

Also, Kimera seemed to provide the possibility to receive external accurate poses. However, a closer examination of the code revealed that the callback functions for reinitialization were incomplete and would require significant changes in the pipeline.

To address this, we allowed the Kimera VIO to estimate the poses internally and the output point cloud extracted from the polygon mesh was transformed from the estimated

pose into the accurate pose. Sometimes, the VIO also tended to be unstable and drift out very quickly. To manage this, the distance between the accurate and estimated poses was being computed and if the difference exceeded a defined threshold, we reinitialized the Kimera pipeline by calling a reinitialization rosservice.

On the other hand, the FLaME method didn't require any modifications. This is because FLaME is specifically designed to generate a mesh from a monocular camera and relies on externally provided poses.

Chapter 6

Evaluation on experiments

Contents

In this chapter, we present the results obtained from our experiments. We start by analyzing the specific configuration options of FLaME and Kimera. We then select the optimal variant and proceed to compare the performance of the two methods. To conduct these evaluations, we utilize the publicly available EuRoC datasets [4]. In the next step, we integrate these methods into the MRS UAV system and assess their performance in the Gazebo simulator. Finally, we evaluate the methods using real-world flight data collected in outdoor environment and compare the outcomes with the ground truth model.

6.1 Metrics for pointcloud evaluation

To evaluate FLaME and Kimera, we employ two key metrics to evaluate the quality of the generated point clouds. Firstly, we utilize the MDR metric, which involves comparing the resulting point clouds against highly precise reference point clouds. Secondly, we utilize the MME metric to measure the degree of point scattering within the point cloud. Additionally for Kimera, we also consider the number of points in the point cloud as an important factor when selecting among the available feature detectors, because it produces smaller number of points, namely in simulation or on the recorded outdoor dataset.

6.1.1 Mean distance to reference

The MDR is a metric used to evaluate the accuracy of a constructed point cloud by measuring the distance between each point in the constructed cloud and its nearest neighbor in a reference point cloud. It relies on having a known reference point cloud, which is often obtained from high-resolution 3D scanners with negligible measurement error. The coordinate systems of the localization system on-board the UAV and the 3D scanner need to be aligned using a yaw-only rigid body transformation algorithm. It is assumed that the nearest neighbor in the reference cloud corresponds to the same point in the real world, which is a reasonable approximation for dense LIDAR scans.

6.1.2 Mean map entropy

When a high-quality reference point cloud is not available, alternative methods can be employed to assess the quality of the map. In these cases, it is assumed that a good map should exhibit sharpness, meaning that a large variation in points that are relatively close together can be attributed to noise. This assumption is particularly valid in man-made environments where planes are prevalent, and nearby points are expected to lie on the same plane and be evenly distributed. Conversely, natural environments like grass would yield more scattered point sets. To address this, we introduce a method known as MME for evaluating map quality.

The method how to compute the entropy of a point in the point cloud is defined in following equation 6.1:

$$H(x) = \frac{1}{2} \ln \left[det(2\pi e \sum(x; r)) \right] \tag{6.1}$$

The $\sum(x; r)$ represents the sample covariance of the points in the radius r around the point x. In our case, we use the value r = 0.6m.

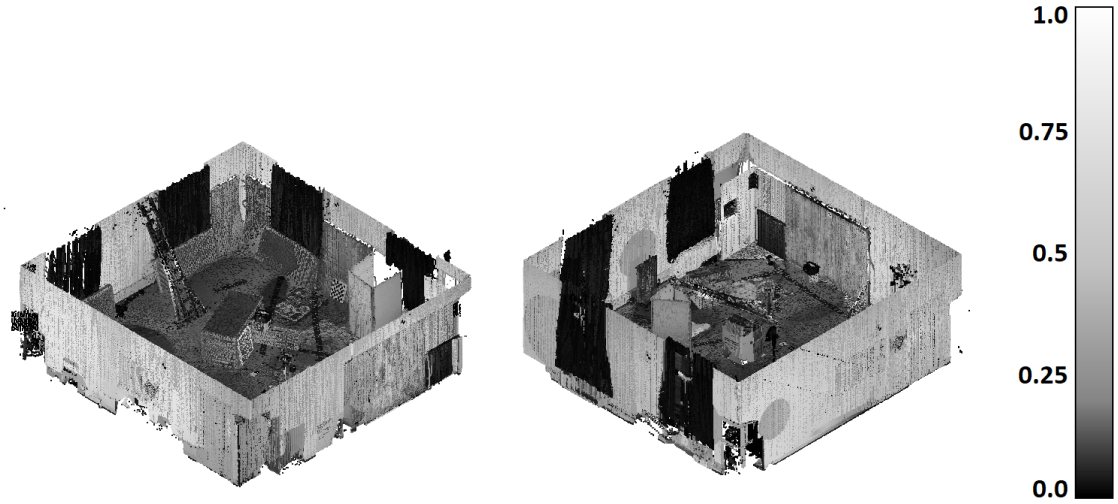
6.2 The EuRoC datasets

The EuRoC micro aerial vehicle (MAV) datasets [4] are one of the most popular datasets for 3D reconstruction, because they provide all the necessary data such as synchronized images, IMU measurements and ground-truth poses of the UAV. Also, the ground-truth laser scan of the environment is provided for the comparison with the point cloud reconstruction by the evaluated algorithms.

We use the part of the EuRoC with two Vicon Room datasets together with an accurate ground-truth 3D structure. The datasets are recorded in two different rooms and are captured over different flight paths through these rooms. Each of the flights has different

level of difficulty: easy, medium and difficult. The difficulty levels resemble the speed of the flight, agility of maneuvers, motion blur and the changes of the lighting.

For better visualization, we removed the upper part of the reference model containing the ceiling. Figure 6.1 illustrates the two modified reference point clouds after the cropping process. The colors in the figure represent the reflection intensity of the materials present in the room. In further figures with result point clouds, the reference model will have always a gray color.



(a) Vicon room 1 (b) Vicon room 2

Figure 6.1: Reference pointclouds from EuRoC [4]

To ensure the authenticity of the results, no point cloud alignment operations, like iterative closest point (ICP), were conducted. Only minimal adjustments were made to the point clouds obtained from FLaME, comprising just from a 90-degree rotation around the z-axis. In case of Kimera, we utilized its feature for initializing the VIO using ground truth odometry data. These ground truth odometry data were generated based on the data extracted from the rosbag.

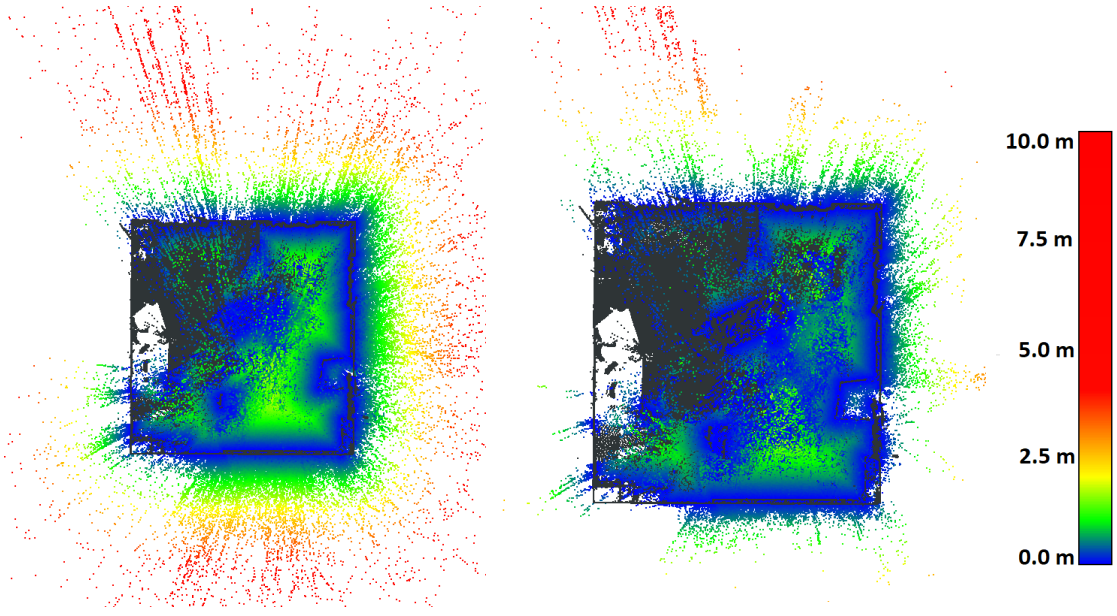
Following the successful construction of the result point clouds, the MDR and corresponding standard deviation (SD) were calculated using the CloudCompare software. MME and corresponding SD were computed using an approach from [42]. Firstly we verified the benefit of the Variational smoothing optimization method used in FLaME. After that we compared the different feature detectors in Kimera. Finally, we chose the variants of FLaME and Kimera that presented the best results and compared them together.

6.2.1 FLaME optimization

We conducted an evaluation to assess the impact of the optimization module in FLaME, specifically the Variational smoothing (as mentioned in section 3.8). When reconstructing the scene without applying this optimization, the variance in the depth map increased significantly. This had a noticeable effect on the resulting point cloud, which had twice higher MDR as shown in table 6.1. On the other hand, the MME was considerable lower. Furthermore, the impact of the optimization module is clearly visible in the fraying artifacts present in the resulting point clouds, as depicted in figure 6.2. These artifacts indicate a loss of detail and accuracy in the reconstructed 3D model when the Variational smoothing optimization is not applied.

Dataset	Without		With	
	MDR [m]	MME	MDR [m]	MME
easy	0.317 ± 1.548	**-0.234 ± 0.588**	**0.154 ± 0.403**	-0.416 ± 0.337
medium	0.390 ± 1.328	**-0.257 ± 0.8**	**0.203 ± 0.280**	-0.336 ± 0.443
difficult	0.803 ± 4.003	**-0.174 ± 1.088**	**0.355 ± 0.801**	-0.208 ± 0.602

Table 6.1: Results of FLaME without and with Variational smoothing on EuRoC Vicon room 1



(a) No optimization (b) Variational smoothing

Figure 6.2: Comparison of FLaME without and with Variational smoothing on EuRoC Vicon room 1

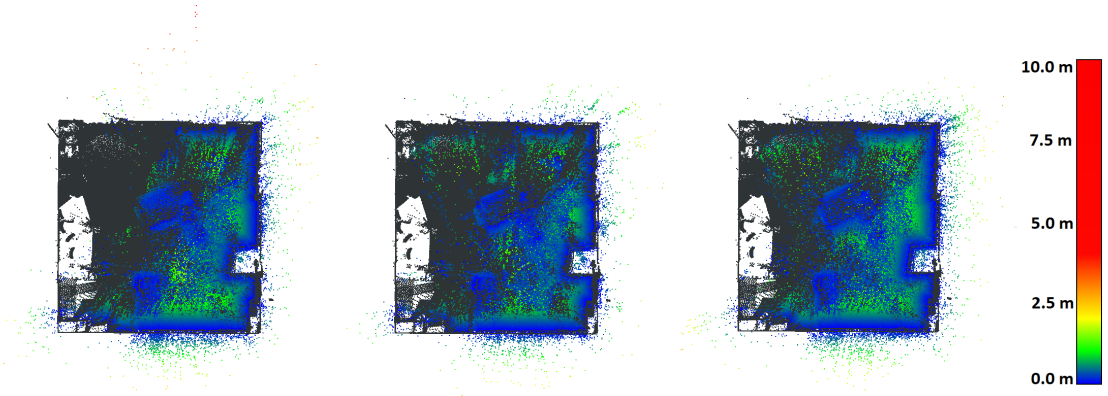
6.2.2 Kimera feature detectors

Authors of Kimera also provide an option to choose from multiple feature detectors: FAST, ORB or GFTT. To determine, which is the most suitable, we chose to run it on the easy variant of Vicon room 1. The results are presented in table 6.2:

Feature detector	MDR [m]	MME	N. of points
FAST	0.317 ± 0.403	-0.385 ± 0.448	289368
ORB	**0.276 ± 0.187**	-0.555 ± 0.472	254913
GFTT	0.298 ± 0.195	**-0.316 ± 0.345**	**405249**

Table 6.2: Results of Kimera with different feature detectors

The MDR values come around the same value, with ORB being slightly more accurate. However, if we look at the number of produced points, the GFTT stands out by a great margin. The values can be verified on the figure 6.3. Even though GFTT has slightly higher MDR, we consider it as the best option, because the pros of twice more 3D points overweights the slightly lower accuracy. Therefore, we use GFTT on Kimera in further experiments. The following visualization underlines the presented results (figure 6.3):



(a) FAST (b) ORB (c) GFTT

Figure 6.3: Comparison of Kimera feature detectors

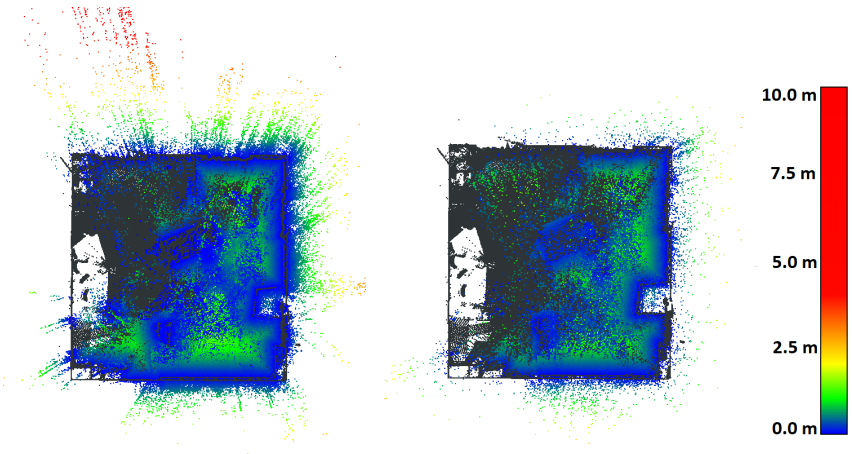
6.2.3 Comparison with best configurations

After selecting the best variants of both methods, we conducted a comparison between them. The results are summarized in table 6.3.

dataset		method	MDR [m]	MME
Vicon room 1	easy	FLaME	**0.154 ± 0.403**	-0.416 ± 0.337
		Kimera	0.303 ± 0.197	**-0.298 ± 0.346**
	medium	FLaME	**0.203 ± 0.280**	**-0.336 ± 0.443**
		Kimera	0.271 ± 0.252	-0.366 ± 0.577
	difficult	FLaME	0.355 ± 0.801	**-0.208 ± 0.602**
		Kimera	0.355 ± 0.318	-0.346 ± 0.752
Vicon room 2	easy	FLaME	0.444 ± 0.985	**-0.184 ± 0.585**
		Kimera	**0.260 ± 0.208**	-0.289 ± 0.395
	medium	FLaME	0.459 ± 0.757	**-0.062 ± 0.611**
		Kimera	**0.257 ± 0.248**	-0.262 ± 0.564
	difficult	FLaME	0.776 ± 1.956	**-0.185 ± 0.918**
		Kimera	**0.292 ± 0.277**	-0.305 ± 0.683

Table 6.3: Results of FLaME and Kimera on the EuRoC dataset

Based on the obtained results, it is evident that the FLaME method performed better in the first room of the evaluation. However it's accuracy decreased noticeably in the second room. Additionally, the standard deviation remained relatively high throughout all flights. This can be attributed to a significant number of points that were estimated in great distance from the rest of the estimated point cloud. The visualization of the comparison can be seen in the figure 6.4:



(a) FLaME (b) Kimera

Figure 6.4: Comparison of FLaME and Kimera point clouds on V1 room — easy

6.3 The MRS UAV system and Gazebo

The functionality of the algorithms was also tested in the Gazebo simulator using the MRS UAV system [13] developed by the MRS group. This framework can provide a simulation of a flight in various environments. The UAV setup can be also configured to be as similar to the real UAV as possible. It offers the selection between different UAV types and a lot of on-board sensors. In this work, a f330 UAV setup with fish-eye camera was used. In figure 6.5, a comparison between the real UAV and its counterpart in the simulation can be seen.

(a) DJI f330 in the simulation (b) DJI f330 in a real world

Figure 6.5: UAV model in Gazebo simulator compared to the real UAV [13]

The simulation was conducted using the model of the Stará Voda church (figure 6.6), which proved to be an ideal environment for the evaluation. This choice was based on several factors, including the church's moderate size and the presence of feature-rich walls. To ensure equal conditions for both methods, a path with multiple way-points was created, so that both methods operated on similar inputs. The reference model (in gray) along with the results from both methods can be seen in figure 6.7.

Figure 6.6: Stará Voda church reference model

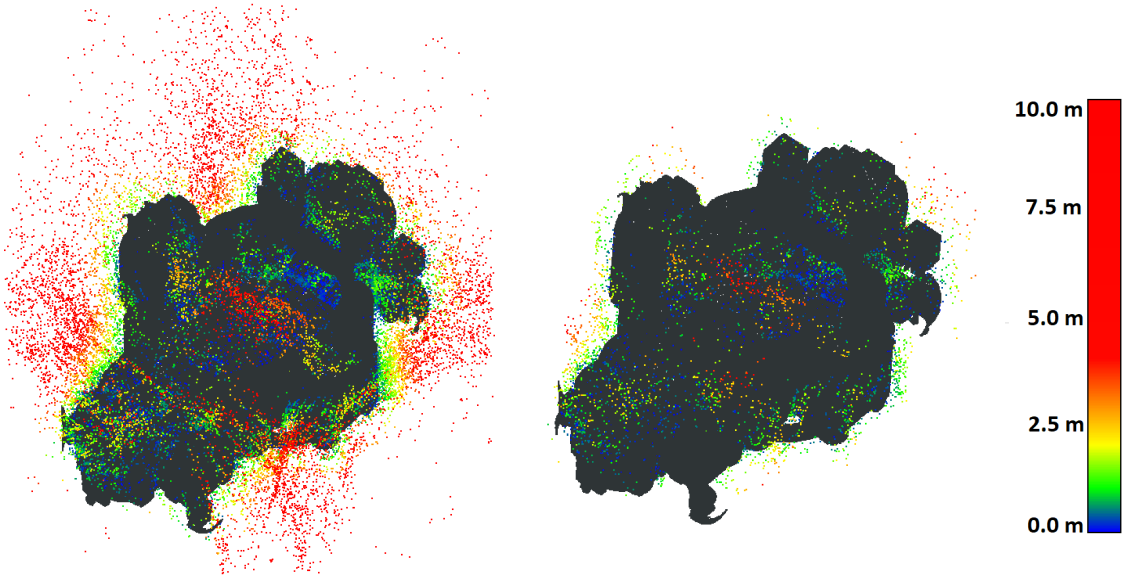
During the flight simulation, it was essential to specify the camera distortion model to ensure accurate estimation of the points' position. This involved identifying the intrinsic and extrinsic camera parameters, as well as selecting the appropriate distortion model (described in section 3.1.1). In this scenario, the equidistant distortion model for the pinhole camera was chosen. This model effectively represents the fish-eye camera and was one of the few supported by the evaluated algorithms.

Furthermore, the input had also been filtered from images and IMU data with identical subsequent time-stamps, which were produced by the simulator. This step was necessary, because otherwise it would cause errors in the evaluated methods, especially Kimera.

The results from the flight are in table 6.4 the resulting point cloud along with the reference (in gray) are illustrated in figure 6.7.

method	MDR [m]	MME
FLaME	2.109 ± 3.797	**-2.066 ± 3.339**
Kimera	**0.727 ± 0.825**	-2.701 ± 3.729

Table 6.4: Results of the comparison in the simulation in Stará Voda church



(a) FLaME (b) Kimera

Figure 6.7: Comparison of FLaME and Kimera in the simulation of Stará Voda church model

In this case, FLaME demonstrated more precise and convincing results compared to Kimera. However, the point cloud generated by FLaME contained a higher number of points that were significantly deviated from the reference model, in comparison to Kimera.

6.4 Testing in real outdoor environment

The experiments from the real-flight data took place on the MRS camp in Temešvár. The flights were conducted in the area with buildings using a UAV setup with real-time kinematic (RTK) positioning system and also a laser scanner for precise ground-truth model that can be used for the evaluation of the error of the tested methods.

The data were recorded using DJI f330 UAV setup, that can be seen in figure 6.8.

Figure 6.8: The UAV setup used for the real world flight experiments

The ground-truth 3D scan of the environment was done by the Leica BLK360 3D laser scanner. This scanner is able to produce a high precision point cloud of the environment with accuracy up to 5 cm. Scans from multiple areas were conducted in order to connect them into one 3D model of the whole area. For that, the ICP algorithm was used. To enable better visualization, the point cloud was cropped in order to cover only the area of the UAV flight path. In figure 6.9 the scanned 3D point cloud can be seen with the colors representing — same as in EuRoC — the intensity of reflection. The result metrics values are in table 6.5.

method	MDR [m]	MME
FLaME	**0.552738 ± 1.809013**	**-0.423 ± 1.406**
Kimera	0.795701 ± 0.788564	-1.779 ± 2.811

Table 6.5: Results of FLaME and Kimera on custom dataset from outdoor environment

To further underline the results from table 6.5, we present the pictures containing the resultant point clouds along with the reference point cloud in figure 6.10:

Based on the results, we can say that FLaME performed better than Kimera in the number of points produced and also in the accuracy. However, neither method presented such point cloud that could be represented as a faithful 3D model, because without the reference cloud it is hardly recognizable. This concludes that in open environment, these methods are not suitable for the precise scene reconstruction.

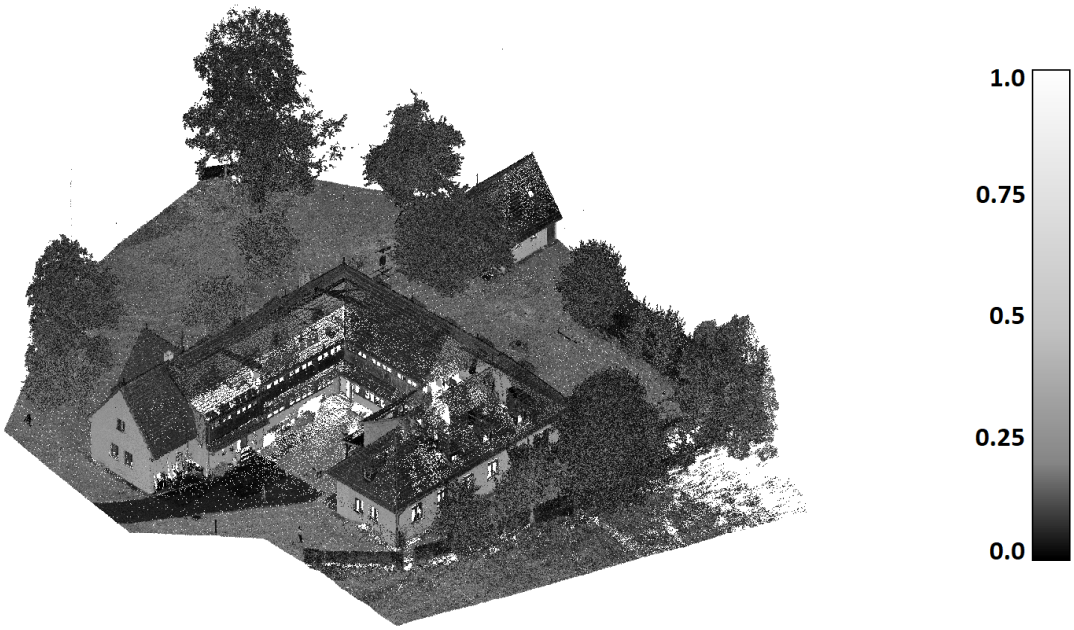
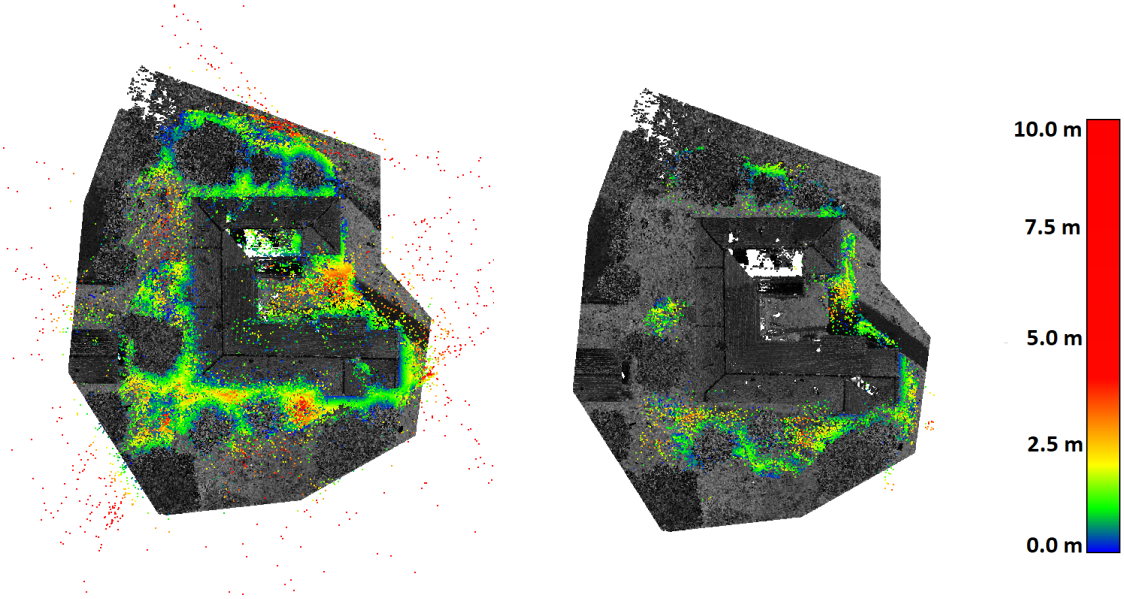


Figure 6.9: Reference 3D point cloud from Leica BLK360 laser scanner



(a) FLaME results (b) Kimera results

Figure 6.10: Comparison of FLaME and Kimera on custom real-world dataset

Chapter 7

Conclusion

Contents

In conclusion, the evaluation of the Kimera and FLaME methods for 3D scene reconstruction has yielded positive results. The evaluation was successfully conducted on the EuRoC dataset, demonstrating the effectiveness of both methods. Integration into the MRS UAV system of both of the methods was successful, performing with correctly modified inputs such as the camera extrinsic and intrinsic parameters and the input stream from camera that was cleared from the redundant images that the algorithms would not be able to process. Additionally, the custom dataset from the real world was successfully recorded and the reference scan of the environment was successfully created. The evaluation provided further insight into their capabilities and characteristics.

Furthermore, the Variational smoothing optimization module utilized by FLaME proved to have significant impact on the method, improving its reconstruction accuracy with point clouds having . The choice of the GFTT feature tracker for Kimera showed to be a good decision because the produced point clouds with this option had twice as many 3D points with comparable precision.

When comparing the two methods, FLaME showcased higher accuracy in the reconstruction, although with a larger standard deviation. This was mainly due to a significant number of points deviating far from the reference model. Kimera, on the other hand, had demonstrated better performance in EuRoC's Vicon room 2 and presented overall less accurate results, but with lesser deviation. The points generated by Kimera were generally more aligned with the reference point cloud.

It can be concluded that both methods are suitable for 3D scene reconstruction in indoor environments. However, they may not perform as effectively in outdoor settings where distances are larger and obtaining features may become more challenging.

7.1 Future work

With the implemented evaluation framework, we were able to evaluate and make comparisons between the FLaME and Kimera methods. The strength of this framework lies in its universality, as it can be applied to analyze and assess any other method that performs the monocular 3D scene reconstruction. This flexibility offers an exciting opportunity to further enhance the evaluation process by incorporating additional similar methods into the analysis. By including a wider range of these methodologies, we can gain a more comprehensive understanding of the strengths and weaknesses of various approaches, thereby contributing to the advancement of the field as a whole. Expanding the evaluation to these additional methods would not only enrich our findings but also provide a more robust basis for future research and development efforts.

Bibliography

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003.

[2] A. Coste, "3d computer vision : Stereo and 3d reconstruction from disparity," February 2013.

[3] W. N. Greene and N. Roy, "Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 4696–4704.

[4] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016.

[5] P. Petracek, V. Kratky, M. Petrlik, T. Baca, R. Kratochvil, and M. Saska, "Large-scale exploration of cave environments by unmanned aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7596–7603, October 2021.

[6] M. Petrlik, P. Petracek, V. Kratky, T. Musil, Y. Stasinchuk, M. Vrba, T. Baca, D. Hert, M. Pecka, T. Svoboda, and M. Saska, "UAVs Beneath the Surface: Cooperative Autonomy for Subterranean Search and Rescue in DARPA SubT," *Field Robotics*, vol. 3, pp. 1–68, January 2023.

[7] M. Saska, K. Zimmerman, J. Faigl, and col., "System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA subterranean challenge," *CoRR*, 2021.

[8] V. Walter, V. Spurny, M. Petrlik, T. Báca, D. Zaitlík, L. Demkiv, , and M. Saska, "Extinguishing real fires by fully autonomous multirotor uavs in the mbzirc 2020 competition," *Field Robotics*, vol. 2, pp. 406–436, April 2022.

[9] N. Michael, S. Shen, K. Mohta, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative mapping of an earthquake damaged building via ground and aerial robots," vol. 92, 07 2012.

[10] J. Horyna, V. Walter, and M. Saska, "UVDAR-COM: UV-Based Relative Localization of UAVs with Integrated Optical Communication," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2022.

[11] P. Petráček, V. Krátký, and M. Saska, "Dronument: System for reliable deployment of micro aerial vehicles in dark areas of large historical monuments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2078–2085, April 2020.

[12] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[13] T. Baca, M. Petrlik, and col., "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *Journal of Intelligent and Robotic Systems*, vol. 102, pp. 1–28, May 2021.

[14] M. Black and P. Anandan, "A framework for the robust estimation of optical flow," in *1993 (4th) International Conference on Computer Vision*, 1993, pp. 231–236.

[15] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH Conference Proceedings*. New York, NY, USA: ACM Press, 2006, pp. 835–846.

[16] ——, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, November 2008.

[17] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, IEEE International Conference on*, vol. 3. IEEE Computer Society, 2003, pp. 1403–1403.

[18] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, p. 1309–1332, 2016.

[19] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[20] W. N. Greene, K. Ok, P. Lommel, and N. Roy, "Multi-level mapping: Real-time dense monocular slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 833–840.

[21] L. Ljung, "Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, no. 1, pp. 36–50, 1979.

[22] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 298–304.

[23] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[24] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[25] K. Eckenhoff, P. Geneva, J. Bloecker, and G. Huang, "Multi-camera visual-inertial navigation with online intrinsic and extrinsic calibration," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3158–3164.

[26] P. Zhu, P. Geneva, W. Ren, and G. Huang, "Distributed visual-inertial cooperative localization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8714–8721.

[27] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020.

[28] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[29] M. Vergauwen and L. Van Gool, "Web-based 3d reconstruction service," *Mach. Vis. Appl.*, vol. 17, pp. 411–426, 12 2006.

[30] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[31] K. Douterloigne, S. Gautama, and W. Philips, "Fully automatic and robust uav camera calibration using chessboard patterns," in *2009 IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, 2009, pp. II–551–II–554.

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

[33] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, 2005, pp. 1508–1515 Vol. 2.

[34] ——, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

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

[36] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.

[37] H. Zhang, J. Wohlfeil, and D. Grießbach, "Extension and evaluation of the agast feature detector," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. III-4, pp. 133–137, 06 2016.

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

[39] E. Oyallon and J. Rabin, "An Analysis of the SURF Method," *Image Processing On Line*, vol. 5, pp. 176–218, 2015.

[40] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of mathematical imaging and vision*, vol. 40, no. 1, pp. 120–145, 2011.

[41] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[42] M. Melecký, "Monocular 3d scene reconstruction for an autonomous unmanned aerial vehicle," Department of Cybernetics, 2021.

Appendices

Archive Content

All root directories in the archive, which contain all necessary accessories to this thesis, are listed in Table 1.

Directory name	**Description**
f330_simulation	the configuration for simulation
eval_package	evaluation framework
flame_scripts	scripts to launch the framework with FLaME
kimera_scripts	scripts to launch the framework with Kimera
thesis	the thesis in pdf format

Table 1: Archive Content

List of acronyms

AGAST Adaptive and Generic Accelerated Segment Test

BRIEF Binary robust independent elementary feature

BDR back-down-right

CPU central processing unit

CTU Czech Technical University

EKF extended Kalman filter

FAST Features from Accelerated and Segments Test

GFTT Good Features to Track

GPS global positioning system

GPU graphics processing unit

ICP iterative closest point

IMU inertial measuring unit

LED light emitting diode

LIDAR light detection and ranging

LSD Large-Scale Direct

MAV micro aerial vehicle

MDR mean distance to reference

MLM multi-level mapping

MME mean map entropy

MRS multi-robot systems

ORB Oriented FAST and Rotated BRIEF

OS Operating system

PCL point cloud library

RFU right-forward-up

ROS robot operating system

RTK real-time kinematic

SD standard deviation

SfM structure from motion

SIFT scale-invariant feature transform

SLAM simultaneous localization and mapping

UAV unmanned aerial vehicle

UV ultra violet

VINS visual-inertial navigation system

VIO visual-inertial odometry

VO visual odometry