UiT The Arctic University of Norway

Faculty or Engineering Science

Robust visual inertial odometry for agile flight sensor fusion

Edvin Ferkic

Master's thesis in Aerospace Control Engineering STE-3900 May 2024

Summary

Summary

This thesis develops and validates a robust visual-inertial odometry (VIO) system for agile flight, specifically for UAVs in GPS-denied environments. Traditional GPS navigation fails in urban canyons, indoor spaces, and dense forests, necessitating alternatives like VIO, which combines camera visuals and IMU sensor data to improve state estimation accuracy.

The main objectives were to create a visual odometry (VO) algorithm that handles lighting variations and motion blur, integrate VO with inertial data using filters, enhance computational efficiency for real-time operation, and validate performance in challenging environments.

The literature review covered the evolution of visual odometry, sensor fusion advancements, SLAM integration with VIO, and current research. It emphasized VIO's importance in applications like drone racing and search-and-rescue operations.

Key technologies used include OpenCV for image processing, Eigen for linear algebra, and ROS for real-time data handling. The system was tested with UZH FPV drone racing datasets, focusing on monocular VO solutions for cost-effectiveness and simplicity.

Experimental results showed the ORB detector with Brute Force matcher performed well under normal conditions, while SIFT and AKAZE were more resilient to motion blur. Although the system effectively integrated visual and inertial data, it lacked advanced filtering techniques like EKF or UKF, affecting robustness and accuracy. However, it maintained real-time capabilities crucial for agile flight, with room for computational efficiency improvements.

Challenges included sensor noise, real-time processing demands, and environmental variability. The simple sensor fusion approach limited performance compared to advanced methods.

This research provides implementation insights, establishes baseline performance for monocular VIO systems, and demonstrates the feasibility of simple sensor fusion techniques for real-time state estimation.

Future work should focus on advanced filtering techniques, integrating state-of-the-art frameworks like OpenVINS and VINS-Mono, enhancing sensor fusion algorithms to reduce drift and improve accuracy, and conducting comprehensive testing in diverse environments.

In conclusion, the developed VIO system shows promise, but further advancements are necessary to optimize its performance in dynamic and challenging environments.

Preface

This master's thesis is submitted in partial fulfillment of the requirements for the Master of Science degree in Aerospace Control Engineering at the Department of Electrical Engineering, UiT, Arctic University of Norway.

I would like to express my gratitude to my advisor, Lecturer Tom Stian Andersen, for his continuous support, expertise, invaluable advice, and insights drawn from his extensive expertise in visual-inertial odometry.
I am also thankful to my supervisor, Professor Jose Juan Corona Sanchez, for his guidance and expertise, which were instrumental in completing this thesis.

I extend my heartfelt thanks to my family, especially my spouse and family, who has provided unwavering support and encouragement throughout this endeavor. Additionally, I am grateful to my current employer, Laerdal Global Health, for offering me the flexibility and motivation to pursue my academic goals. Finally, I would like to thank my friends for their support and encouragement throughout my studies.

Edvin Ferkic,
Narvik, 2024

Contents

List of Figures

List of Tables

1 Introduction

In the rapidly changing world of autonomous flight, it is crucial for unmanned aerial vehicles (UAVs) to accurately perceive and navigate their surroundings. One important aspect of this process is determining the UAV's position and orientation over time, which is typically achieved using a global positioning system. However, in environments with weak or no Global Positioning System (GPS) signals, such as urban canyons, indoor spaces, and dense forests, relying on GPS for navigation is impractical and sometimes impossible. Consequently, alternative methods, such as visual odometry combined with inertial data, have become increasingly promising approaches for autonomous navigation. In such situations, visual-inertial odometry (VIO) plays a vital role by leveraging the camera of the device and an Inertial Measurement Unit (IMU) sensor to compensate for the lack of GPS to provide positional information[9]. This information is then used to build an understanding of the 3D space through a process called Simultaneous Localization and Mapping (SLAM)[3]. This process allows the content to be placed directly into the field of view of the UAV, enabling it to navigate autonomously and stick to its environment. This has proven to be very important in drone racing sports[3][4].

In VIO, the visual component captures the structure of the environment, whereas the inertial component provides the acceleration and rotation rates, enabling the UAV to deduce its movement through space. This combination leverages the complementary strengths of both sensor types: high-rate, but drift-prone, inertial measurements, and slower but more stable visual cues. The fusion of these data streams allows for more accurate and robust state estimation than either modality can be achieved independently[3][13].

1.1 Background and motivation

1.1.1 Background

The importance of VIO in enabling agile flight for UAVs cannot be overstated owing to rapid changes in speed and direction, which poses a significant challenge to state estimation, primarily because of the high dynamic range and the potential for rapid environmental changes; however, most VIO systems are particularly well-suited to these conditions already, but many systems still require fine tuning and experimental approaches to receive the best possible rigid and robust solution[9].

The robustness and accuracy of typical VIO systems have been extensively documented in literature. Studies by[3][4] provide foundational insights into the theoretical underpinnings and practical implementation of VIO especially when using various already state of the art frameworks. Their research highlighted the challenges of visual-inertial fusion, such as dealing with noisy sensor data and the computational demands of real-time processing, motion blur, highlighted scenes or bright light sources which are crucial for maintaining the performance of VIO systems during agile maneuvers[3][4].

1.1.2 Motivation

Since it is already well known that autonomous navigation of for example quadrotors requires precise and robust state estimation, there is still potentially room for improvement, especially at the visual odometry front end side.

Having a good algorithm in terms of the front-end side is crucial to allow for better robustness and stability in estimating the trajectory as closely as possible to the ground truth, but also in terms of computational power, particularly during agile robotic flights used in drone racing sports.

Implementing VIO in agile flight scenarios such as those encountered by racing quadrotors or during emergency response missions introduces significant challenges. These challenges include handling the rapid dynamics of flight, managing variations in environmental lighting, and mitigating the effects of motion blur on the visual sensors.

1.2 Objectives

This thesis aims to explore and verify one of the most robust visual odometry algorithms with a combination of inertial data to help improve the state estimation for quadrotors during agile flight in challenging environments such as drone racing. Specific objectives have been crafted to address these challenges and provide potential improvements in VIO systems:

1. Develop a robust visual odometry within the VIO framework aiming to better handle variations in lighting intensity and reduce the adverse effects of motion blur. This involves performing image processing of the incoming relevant dataset images and see how visual odometry will be affected by different techniques used.

2. Integrate visual odometry with inertial data by well suited filters or other techniques for more effective fusion between inertial and visual data, focusing on optimizing the integration process to enhance the accuracy and reliability of the state estimation.

3. Improving processing capabilities which aim to increase the computational efficiency of a VIO system to ensure it can operate in real-time when needed, a critical requirement for agile quadrotors.

4. Conduct a series of experiments to evaluate the enhancements made to the VO/VIO system.

1.3 Literature review

Recent advancements and ongoing challenges within the field of Visual-Inertial Odometry (VIO) have highlighted its significance, particularly in the context of agile flight in drones and autonomous vehicles[5]. We will explore some it's developments, history and challenges throughout the sub-chapters.

1.3.1 Historical development of VO

Visual Odometry (VO) particularly, was first developed in the early 1980s with the aim of navigating space rovers on extraterrestrial surfaces. Since then, it has evolved significantly and has been adapted for a wide range of applications on Earth. The technology has become a cornerstone for autonomous navigation systems used in drones, self-driving cars, and other robotic systems. In fact visual odometry today use cameras ranging from monocular to multi-stereo which helps to significantly remove the need of other sensors as shown in Figure 1 [6].

Figure 1: Multiple Cameras for VO
[6]

Applications of VO are particularly prominent in areas where dynamic conditions prevail, such as in drone racing, where rapid maneuvers are common. These applications underscore the necessity for robust VIO systems

that can operate effectively under highly dynamic conditions and manage rapid changes in the environment[5].

During the development of Mars Exploration Rovers, VO and VIO was used for precise navigation across the Martian terrain. This early application underscored VO's potential in environments where GPS and other typical navigation systems fail[8].

The application of VO extends to critical missions such as search-and-rescue operations where reliability and precision can significantly impact outcomes. For instance, in environments complicated by natural disasters, VO enables drones to navigate debris and obstructions with high accuracy, proving essential for locating survivors in challenging terrains[6].The technology's adaptability to different lighting conditions and its capacity to handle high-speed movements make it invaluable across various sectors. Innovations like the Direct Sparse Odometry (DSO) and Semidirect Visual Odometry (SVO) have further refined VO's capabilities, enhancing its performance in real-time applications by reducing computational demands and improving accuracy[8].

Moreover, the integration of VO with other technologies like SLAM has expanded its applications to more complex environments. This integration allows for the creation of a detailed environmental map while simultaneously tracking the vehicle's location, further enhancing the navigation capabilities of autonomous systems[6]. As VO continues to evolve, its applications are expected to expand further, encompassing more sectors, not just in drone racing and offering more robust solutions for autonomous navigation challenges in complex and dynamic environments[6].

1.3.2 Advancements and challenges in sensor fusion

The integration of advanced sensor fusion techniques has been a focal point of recent research, aiming to enhance the reliability and accuracy of VIO systems. These techniques help to mitigate the detrimental effects of sensor noise and environmental factors, which could otherwise degrade the performance of navigation systems. In drone racing, for example, the agility required necessitates VIO systems that can quickly adapt to new environments and maintain robustness under rapid changes and conditions[3]. Significant research has also been directed toward addressing the challenges posed by VIO systems, such as handling high-speed movements under variable lighting conditions. Innovations in this area include the development of adaptive algorithms that adjust to changing environmental conditions and machine-learning models designed to predict and navigate potential obstacles. These advancements are critical in ensuring that VIO systems can perform reliably in diverse and challenging scenarios[3]. The decision to use monocular

or stereo VIO solutions is completely dependent on the robotic system it's going to be used upon. We can differ between low cost monocular or higher cost stereo, each of these has its pros and cons. To get an overview over the most typical differences between these types, it can be illustrated as in Figure 2.

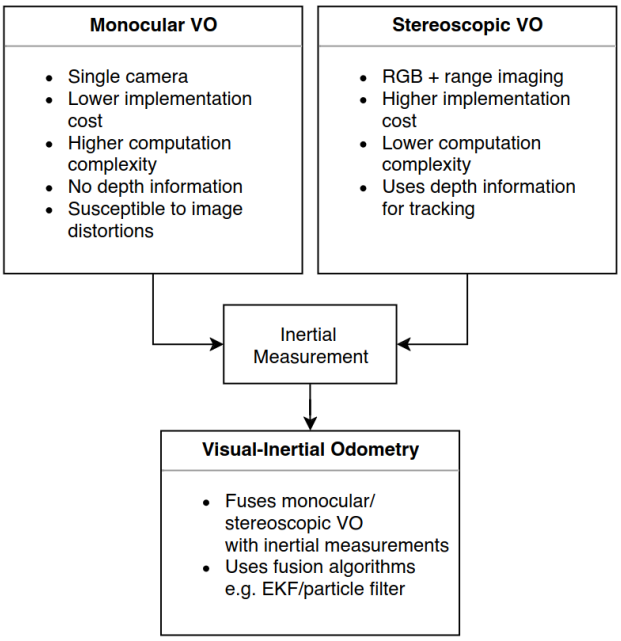

Figure 2: Mono vs Stereo types of VO
[8]

To assure a proper VIO system, one has to typically integrate with an IMU unless one have multiple cameras to disregard the inertial sensor completely.

1.3.3 Integration of SLAM and V-SLAM with VIO

SLAM and its visual counterpart, Visual SLAM (V-SLAM), significantly enhance the capabilities of VIO by not only tracking but also mapping the environment. This integration is beneficial in scenarios where maintaining a consistent and accurate trajectory over time is crucial. V-SLAM, in particular, provides a powerful tool for VIO systems by creating a detailed map of the environment while simultaneously tracking the location of the vehicle within it. This dual capability is essential for applications that require high levels of navigation precision and operational reliability.[3][4]. Figure 3 illustrates the core components of a typical V-SLAM system, including real-time mapping, localization, key-frame selection, feature extraction, and loop

closure. Camera images is the input which gets processed in the front-end, then fed to the back-end where optimizations happen, this is continuously in a loop closure, creating a map of the environment as the robot or robot vehicle maneuvers[32].

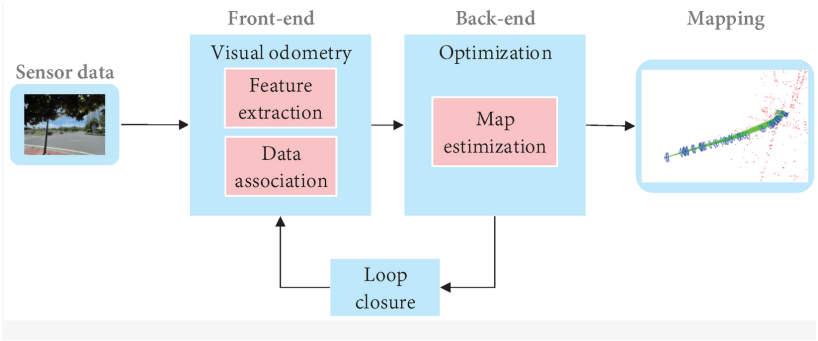

Figure 3: V-SLAM Framework [32]

1.3.4 Future Directions and Ongoing Research

The field of VIO continues to evolve rapidly, with substantial progress being made in enhancing system robustness and efficiency. Ongoing research is crucial in addressing existing limitations and pushing the boundaries of what these systems can achieve. Future directions may include the refinement of sensor fusion techniques, the development of more advanced machine-learning algorithms for obstacle navigation, and the integration of more comprehensive environmental data into VIO systems[6].

The continuous improvement of VIO systems is expected to significantly enhance autonomous navigation capabilities, especially in applications that demand high precision and agility. As this technology advances, it promises to open up new possibilities for autonomous systems in various sectors, including transportation, military, and emergency services[6].

1.4 Delimitation and scope of thesis

As discussed in the objectives, this thesis aims to propose a specific and more robust solution to the visual front end with a combination of inertial data. These advancements can benefit applications such as drone racing, aerial surveillance, and search-and-rescue operations.

This thesis has mainly focused on these specific areas:

Monocular based solution

Monocular or (one camera) data as input have been used to test algorithms that work well for monocular VIO systems, even though such solutions can suffer from scale ambiguity and are highly dependent on good feature tracking and matching algorithms.

Libraries and frameworks

The proposed system uses the Robot Operating System (ROS), OpenCV for image processing, Eigen for linear algebra, and TF2 for transformations, which are all robust choices for VIO systems. In addition, the programming language C++ was used because it is widely supported in many applications and frameworks.

Image processing and IMU data integration

Image processing and IMU data integration play an important role in achieving robustness and performance, and are used in this thesis in order to increase accuracy and robustness.

1.5 Report Outline

This thesis report is organized into several chapters that detail the development, implementation, and evaluation of robust visual-inertial odometry systems for UAVs. Below is a concise outline of the contents of each chapter:

1. **Introduction:** This opening chapter introduces the motivation and objectives behind developing robust visual-inertial odometry systems for UAVs, especially focusing on environments where GPS is unreliable.

2. **Theoretical and Technological Preliminaries:** An exploration of the fundamental theories and technologies that form the basis of the research, including detailed discussions on visual odometry, sensor fusion, and the tools and technologies employed.

3. **Implementation:** Detailed description of the implementation of the theories discussed, including the setup of ROS nodes, image processing techniques, and the integration of IMU data.

4. **Main Results:** Presentation of the experimental findings, discussing the performance of the visual odometry system under various test conditions.

5. **Simulation/Experimental Results:** This chapter details the simulation setups and experimental protocols used, along with the results obtained, validating the theoretical and practical implementations.

6. **Discussion:** A discussion on the implications of the findings, limitations of the current system, comparison with existing systems, and the contributions of this research to the field of monocular VIO.

7. **Conclusion and Future Work:** Summarization of the research with discussions on potential future work and directions that could enhance the visual-inertial odometry systems further.

Appendices

Additional supporting materials and documentation relevant to the thesis.

References

A list of all academic and technical sources referenced throughout the thesis.

2 Theoretical and technological preliminaries

This section provides an overview of the theoretical foundations and technologies that are essential for understanding the operation and development of the system used in the thesis.

2.1 Notation

Symbol	Description
x, y, z	Coordinates in 3D space
u, v	Coordinates in the 2D image plane
K	Camera intrinsic matrix
R	Rotation matrix
t	Translation vector
q	Quaternion representing orientation
θ	Angle of rotation
ω	Angular velocity
a	Acceleration
Δt	Time interval between frames
π	Projection function
f	Focal length of the camera
d	Distortion coefficients
λ	Eigenvalue
v	Eigenvector
$\mathbf{A}$	Matrix
$\mathbf{I}$	Identity matrix
$\mathbf{P}$	Orthogonal matrix
$\mathbf{Q}$	Orthogonal matrix
$\mathbf{T}$	Tridiagonal matrix
$\mathbf{H}$	Measurement matrix
$\mathbf{F}$	State transition matrix
$\mathbf{B}$	Control input matrix
$\mathbf{w}$	Process noise
$\mathbf{v}$	Measurement noise

Table 1: List of Symbols and Notations

The following equations are frequently used in this thesis:

$$x_{k+1} = \mathbf{F}x_k + \mathbf{B}u_k + \mathbf{w}_k \tag{2.1}$$

$$z_k = \mathbf{H}x_k + \mathbf{v}_k \tag{2.2}$$

$$\theta = \arctan\left(\frac{a_y}{a_z}\right) \tag{2.3}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \tag{2.6}$$

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T \tag{2.7}$$

$$\mathbf{T} = \mathbf{P}^T\mathbf{A}\mathbf{P} \tag{2.8}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}(\mathbf{R}\mathbf{X} + \mathbf{t}) \tag{2.9}$$

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{a}_t, \omega_t) + \mathbf{w}_t \tag{2.10}$$

$$\mathbf{e}_{t,i} = \mathbf{p}_{t,i} - \pi(\mathbf{P}_{t,i}, \mathbf{K}, \mathbf{x}_t) \tag{2.11}$$

2.2 OpenCV for image processing

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library that is crucial for image-processing tasks in VIO systems. It offers over 2500 algorithms and functions for a broad range of vision tasks. Designed for computational efficiency, OpenCV is suitable for real-time applications and, hence, well suited for VIO systems.[27]. OpenCV supports a vast array of operations that are critical to image processing, including

- Image and Video Capturing: Fundamental for acquiring visual data.
- Image Processing Operations: Includes filtering, transformations, and morphological operations.
- Feature Detection and Matching: Essential for tracking visual features crucial in VIO for establishing motion correspondences[27].

2.2.1 Feature Detector

One of the promising algorithms in OpenCV is called Oriented FAST and Rotated BRIEF (ORB), which is an robust algorithm used for feature detection. This method detects corners or distinct features that can be tracked reliably across successive images. ORB is is mathematically defined as follows[28]:

ORB uses FAST keypoint detector. The candidate pixel is declared a feature if there are n contiguous pixels in the circle which are all either brighter than the candidate pixel plus a threshold t, or all darker than the candidate pixel minus tThe orientation of each keypoint is computed to achieve rotation invariance. It is based on the intensity centroid of the patch around the keypoint[28]. The moments of a patch are given by:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{2.12}$$

where $I(x, y)$ is the intensity at coordinates (x, y), and p and q are the orders of the moments along x and y axes, respectively. The centroid C is calculated as:

$$C_x = \frac{m_{10}}{m_{00}}, \quad C_y = \frac{m_{01}}{m_{00}} \tag{2.13}$$

The orientation θ is then computed by:

$$\theta = \text{atan2}(m_{01}, m_{10}) \tag{2.14}$$

ORB ensures scale invariance by building an image pyramid, creating layers of the image at different scales and detecting keypoints at each level.

The so-called Harris corner measure is used to score keypoints:

$$\text{Score} = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{2.15}$$

where λ_1 and λ_2 are the eigenvalues of the second moment matrix of the image gradient at the keypoint, and k is typically around 0.04.

The BRIEF descriptor is modified for rotation invariance by using the computed orientation θ of the keypoints. The descriptor compares the intensity differences between pairs of points in the neighborhood around each keypoint[28].

2.3 Camera Pose Estimation

Since OpenCV provides robust tools for pose estimation, one of which is the perspective-n-point 'solvePnP' function algorithm. The 'solvePnP' algorithm as one of many, in OpenCV is used to estimate the pose of a camera by finding the best fit between 3D points and their corresponding 2D projections in the image. This is achieved through a perspective transformation that relates the point correspondences[29]. The objective of pose estimation is to minimize the reprojection error, which is the distance between the observed projections of the points in the image and their predicted positions using the estimated pose parameters. Mathematically, this is formulated as an optimization problem[29]:

- A set of n 3D points in the world coordinates: $\mathbf{X}_i$ where $i = 1, \ldots, n$.
- The corresponding 2D image points: $\mathbf{x}_i$.

The camera's intrinsic parameters matrix, $\mathbf{K}$, includes the focal lengths and the optical center which kan be computes as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

The goal is to find the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ such that:

$$\mathbf{x}_i \approx \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_i \tag{2.17}$$

The projection from 3D world coordinates to 2D image coordinates can be expressed as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \left(\mathbf{R}\mathbf{X}_i + \mathbf{t} \right) \tag{2.18}$$

Where (u, v) are the coordinates in the image plane. To normalize, we can divide by the third row:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \left(\mathbf{R}\mathbf{X}_i + \mathbf{t} \right) \tag{2.19}$$

where s is a scaling factor.

2.4 Eigen for linear algebra

Eigen is a high-level C++ library for linear algebra, and is optimized for both dense and sparse matrices[27]. It is extensively used in fields that require efficient matrix and vector computations, such as robotics, physics simulations, and image processing. Eigen is integral for transforming feature data into coordinate systems used in VIO, and for performing necessary calculations for the fusion of visual and inertial data. The basic mathematics used in report is explained as follows:

For a given symmetric matrix $A \in \mathbb{R}^{n \times n}$, the goal is to find its eigenvalues λ and corresponding eigenvectors v such that:

$$Av = \lambda v \tag{2.20}$$

Where:

- λ are the eigenvalues of A,
- v are the corresponding eigenvectors of A,
- $v \neq 0$.

The eigen decomposition involves decomposing A into a product of three matrices:

$$A = Q \Lambda Q^T \tag{2.21}$$

Where:

- Q is an orthogonal matrix whose columns are the eigenvectors of A,
- Λ is a diagonal matrix whose diagonal elements are the eigenvalues of A,

- Q^T is the transpose of Q.

The first step in many algorithms for eigenvalue computation involves reducing the matrix to a simpler form, such as a tridiagonal matrix, using orthogonal transformations:

$$T = P^T AP \tag{2.22}$$

Where T is a tridiagonal matrix and P is an orthogonal matrix.

After the matrix A (or its tridiagonal form T) is obtained, the QR algorithm is typically applied. The QR algorithm involves decomposing the matrix into a product of an orthogonal matrix Q and an upper triangular matrix R (QR decomposition), then forming a new matrix by multiplying R and Q in the reverse order:

$$A_k = Q_k R_k \tag{2.23}$$

$$A_{k+1} = R_k Q_k \tag{2.24}$$

As k increases, A_k converges to a diagonal form, where the diagonal elements are the eigenvalues of A.

This iterative process continues until the off-diagonal elements of A_k are sufficiently small, indicating that the matrix has nearly converged to a diagonal matrix. The eigenvalues can be directly read from the diagonal, and the product of all the Q_k matrices used in each iteration gives the eigenvectors.

2.5 ROS for Real-Time Data Handling

The Robot Operating System (ROS) provides a framework and set of tools for building complex robot behavior. Ros facilitates real-time data handling from multiple sensors and offers tools for visualization, simulation, and debugging essential for VIO development[30].

ROS excels in managing real-time data streams from multiple sensors or datasets, which is crucial for systems such as VIO that rely heavily on timely and accurate data for navigation and mapping. It handles these tasks through a publisher-subscriber model, in which data from sensors are published as messages on topics that can be subscribed to by different nodes in the system. ROS is employed to orchestrate the flow of data and synchronization of tasks between OpenCV image-processing operations and Eigen's matrix calculations. This integration is crucial for maintaining the real-time performance required for efficient VIO. ROS1 and the newest version ROS2[30] is compatible with a wide range of robots and components as shown in Figure 4.

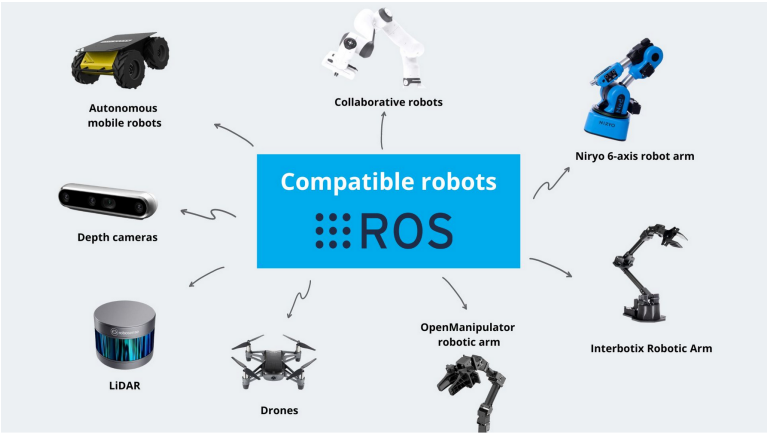

Figure 4: ROS Compatibility
[31]

3 Implementation

This chapter guides throw how the visual odometry part was implemented and what it means to feature tracking and matching.

3.1 ROS Node

We begun by initializing a ROS node and setting up logging, which is crucial for debugging and tracking the VO process. It also initializes the camera matrix and distortion coefficients, which are essential for accurate image undistortion and camera calibration. The logger setup logs essential actions and events to a file, aiding in troubleshooting and performance assessment.

The VO system subscribes to multiple ROS topics, and this can this can be real time camera and IMU data or a rosbag container files which is either pre-recorded or published from another node. We used a rosbag to subscribe important data as shown in Figure 5. There are many official and public datasets which can be used to compare performance and robustness in VO and VIO algorithm. We chose one of the famous drone racing dataset from UZH FPV[33] community due to their aggressive approach in typically drone racing scenarios or agile drone performances.

3.2 Image Processing

The Gaussian blur filter is implemented using a convolution operation, where a small kernel (typically 3x3 or 5x5) is slid over the entire image, computing the weighted average of neighboring pixels at each position. The weights are derived from a Gaussian distribution, which gives more importance to

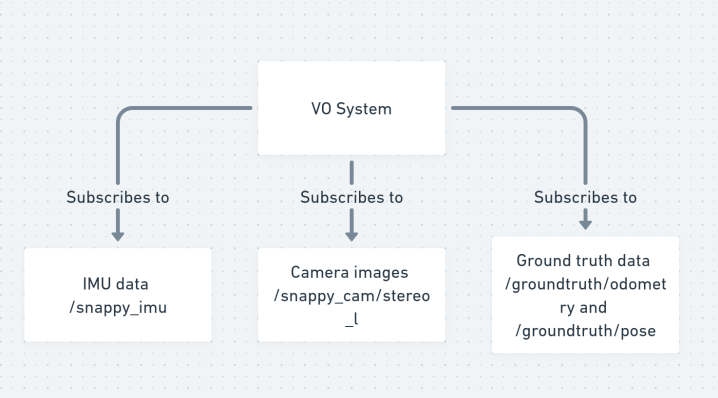

Figure 5: ROS Subscribers

central pixels and less to peripheral pixels. This process is repeated for each pixel in the image, resulting in a blurred image.The Laplacian filter is implemented using a discrete approximation of the Laplace operator, which computes the second derivative of an image intensity function. This is achieved by convolving the image with a small kernel (typically 3x3) that emphasizes the difference between neighboring pixels.Gaussian blur and Laplacian filter will help improve the accuracy and robustness of the VO by reducing noise, enhancing feature detection, and improving the estimation of camera motion in our case.

3.2.1 Undistortion

The image points were corrected for lens distortion using the following equation:

$$\mathbf{x}_{\text{undistorted}} = \mathbf{K}^{-1} \left(\mathbf{x}_{\text{distorted}} - \mathbf{d} \right) \tag{3.1}$$

This equation describes how distorted image points $\mathbf{x}_{\text{distorted}}$ are transformed into undistorted points $\mathbf{x}_{\text{undistorted}}$ by first subtracting the distortion vector $\mathbf{d}$ and then applying the inverse of the intrinsic matrix $\mathbf{K}$.

3.2.2 Feature Detection and Tracking

Features were detected using the ORB detector and tracked between frames using the Lucas-Kanade method.

$$\mathbf{v} = \left(\mathbf{A}^T \mathbf{A} \right)^{-1} \mathbf{A}^T \left(\mathbf{x}_{\text{current}} - \mathbf{x}_{\text{previous}} \right) \tag{3.2}$$

Lucas-Kanade Optical Flow: This method estimates motion vectors $\mathbf{v}$ by minimizing the error between the current and previous image points, facilitated by the matrix $\mathbf{A}$ which is constructed based on image gradients.

3.2.3 Quaternion Initialization

The initial orientation from the accelerometer data is computed as follows:

$$\theta = \arctan\left(\frac{a_y}{a_z}\right), \quad \mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The orientation matrix $\mathbf{R}$ is then converted to a quaternion for more efficient computation and integration.

The camera pose was estimated by combining feature tracking with IMU data.

The quaternion is converted back into a rotation matrix for integration with the visual data:

$$\mathbf{R} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_z q_w) & 2(q_x q_z + q_y q_w) \\ 2(q_x q_y + q_z q_w) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_x q_w) \\ 2(q_x q_z - q_y q_w) & 2(q_y q_z + q_x q_w) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{3.4}$$

The translation of the camera was estimated using the calculated rotation matrix and the tracked features the translation of the camera is estimated.

Below is a table of parameters that were used for feature tracking for ORB and image processing.

Parameter	**Value**	**Description**
MAX PATH SIZE	100	Maximum size of the path.
WTA K	2	Winning threshold.
edgeThreshold	31	Edge threshold for image processing.
fastThreshold	20	FAST algorithm threshold.
firstLevel	0	First level of pyramid.
i	0	General index variable.
kernelSize	3	Size of the kernel for image filtering.
nfeatures	1500	Number of features to detect.
nlevels	8	Number of levels in the pyramid.
patchSize	31	Size of the patch for feature detection.
ratioThreshold	0.7	Ratio threshold for matching.
scaleFactor	1.2	Scale factor for the pyramid.

Table 2: List of Parameters with Assigned Values in Monocular VO Code

4 Main Result

The proposed solution consists of several main components: image processing for feature detection and tracking, motion estimation using visual data, inertial integration for improved pose estimation, and pose optimization.The data flow begins with image capture, followed by feature detection and tracking, motion estimation, IMU data integration, and finally pose optimization to produce the current pose. It is fed with data from a rosbag supported by the ROS, which is essentially a container of information data needed by the system. This rosbag contains typical timestamped images, IMU data, and potentially ground-truth data, which can be used for comparison between different VIO algorithms.

4.1 Visual Odometry (VO)

The VO component in this solution is responsible for estimating the motion of the camera using visual information from the camera. It consists of the following steps:

1. Feature detection and tracking
2. Motion estimation
3. Bundle adjustment

4.1.1 Feature Detection and Tracking

The first step in the VO component is to detect and track features in the camera images. We use the Shi-Tomasi corner detector to detect features and the Kanade-Lucas-Tomasi (KLT) tracker to track them across multiple frames. For this to work we needed to understand the basic process:

We let the I_t be the camera image at time t and I_{t+1} be the camera image at time $t+1$. We detect n_t features in I_t and n_{t+1} features in I_{t+1}. We represent the features as 2D points in the image coordinates. Let $p_{t,i} = (u_{t,i}, v_{t,i})$ be the i-th feature in I_t and $p_{t+1,j} = (u_{t+1,j}, v_{t+1,j})$ be the j-th feature in I_{t+1}.

Finally we use the KLT tracker to find the correspondence between the features in I_t and I_{t+1}. Let $p_{t,i}$ be the i-th feature in I_t and $p_{t+1,j}$ be the corresponding feature in I_{t+1}. We represent the correspondence as (i, j).

4.1.2 Motion Estimation

Once we have tracked the features, we estimate the motion of the camera using the motion model and the feature tracks. We can use the Extended

Kalman Filter (EKF) to estimate the motion of the camera. However this was not implemented due to complexity of such estimator or framework based estimator like VINS-Mono[34] which already had a good estimator but difficult to integrate with this project.

However, if we were to use a EKF, it can further be performed by letting $x_t = (p_t, v_t, q_t)$ be the state of the camera at time t, where $p_t = (x_t, y_t, z_t)$ is the position, $v_t = (vx_t, vy_t, vz_t)$ is the velocity, and $q_t = (qx_t, qy_t, qz_t, qw_t)$ is the orientation represented as a quaternion. The motion model is given by:

$$p_{t+1} = p_t + v_t\Delta t + \frac{1}{2}a_t\Delta t^2 v_{t+1} = v_t + a_t\Delta t q_{t+1} = q_t \otimes \exp(\frac{1}{2}\omega_t\Delta t) \quad (4.1)$$

where $a_t = (ax_t, ay_t, az_t)$ is the acceleration, $\omega_t = (\omega x_t, \omega y_t, \omega z_t)$ is the angular velocity, and Δt is the time interval between I_t and I_{t+1}.

We use the feature tracks to estimate the acceleration and the angular velocity this by letting $p_{t,i}$ be the i-th feature in I_t and $p_{t+1,j}$ be the corresponding feature in I_{t+1}.

We estimate the depth of the feature as:

$$d_{t,i} = \frac{f_t}{p^z_{t,i}} \quad (4.2)$$

where f_t is the focal length of the camera.

We can estimate the velocity of the camera as:

$$v_t = \frac{p_{t+1,j} - p_{t,i}}{d_{t,i}} \quad (4.3)$$

We can estimate the acceleration of the camera as:

$$a_t = \frac{v_{t+1} - v_t}{\Delta t} \tag{4.4}$$

We can estimate the angular velocity of the camera as:

$$\omega_t = \frac{1}{2} \log_m(q_{t+1} \otimes q_t^{-1}) \tag{4.5}$$

where $\log_m$ is the logarithm map of the quaternion.

Finally we use the estimation equation to estimate the state of the camera at time $t + 1$ as:

$$x_{t+1} = f(x_t, a_t, \omega_t) + w_t \tag{4.6}$$

where f is the motion model and w_t is the process noise.

4.1.3 Bundle Adjustment

To refine the motion estimates and the 3D positions of the features, we use bundle adjustment, that minimizes the reprojection error between the observed image points and the points projected from the estimated 3D points using the estimated camera pose.

Typically, we can perform this by letting $P_{t,i} = (X_{t,i}, Y_{t,i}, Z_{t,i})$ be the 3D position of the i-th feature in I_t and $p_{t,i} = (u_{t,i}, v_{t,i})$ be the corresponding image point.

By this we calculate the reprojection error is now given by:

$$e_{t,i} = p_{t,i} - \pi(P_{t,i}, K, x_t) \tag{4.7}$$

where π is the projection function and K is the camera intrinsic matrix.

Finally we use the Levenberg-Marquardt algorithm to minimize the reprojection error and estimate the 3D positions of the features and the motion of the camera.

4.1.4 Sensor Fusion

To fuse the motion estimates from the VO and IMU data components, we did not use a special sensor fusion estimator filter like EKF or UKF. Due to complexity of the task after various attempts integrating with OpenVINS[33], VINS-Mono[34] we were settled to use normal IMU data integration.

We use the EKF to fuse the motion estimates as:

$$x_{t+1} = \hat{x}_{t+1} + K_t(x_{t+1,V} - \hat{x}_{t+1,V}) + K_t(x_{t+1,I} - \hat{x}_{t+1,I}) \tag{4.8}$$

where $\hat{x}_{t+1}$ is the predicted state of the object at time $t + 1$, K_t is the Kalman gain, $x_{t+1,V}$ is the state of the camera at time $t+1$ estimated by the

VO component, and $x_{t+1,I}$ is the state of the object at time $t+1$ estimated by the IMU component.

4.2 State Estimation

State estimation was performed using the Kalman filter approach as follows:

$$\mathbf{x}_{k+1} = \mathbf{F}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k, \tag{4.9}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{4.10}$$

where $\mathbf{F}_k$ is the state transition matrix, $\mathbf{H}_k$ is the measurement matrix, and $\mathbf{w}_k, \mathbf{v}_k$ are the process and measurement noise, respectively.

4.3 Camera Calibration Parameters

The intrinsic parameters of the camera and the distortion coefficients were set as follows:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 \end{bmatrix}$$

The parameter descriptions are explained in the table below.

Parameter	Description
f_x, f_y	Focal lengths of camera in x and y dimensions, respectively.
c_x, c_y	coordinates of the principal point, which is usually at the image center.
k_1, k_2	Radial distortion coefficients of the camera lens.
p_1, p_2	Tangential distortion coefficients that account for lens distortion not aligned with the image plane.

Table 3: Camera Parameters

5 Simulation/Experimental results

5.1 Visual Odometry Results

In this section we list all the experimental visual odemtry results based on different feature detectors and matchers used on the same UZHFPV INDOOR #6 MEDIUM dataset[33]

5.2 Description

Figure 6: ORB on Monocular VO with BF

Description: This figure shows the results of using the ORB (Oriented FAST and Rotated BRIEF) feature detector combined with the Brute Force (BF) matcher for monocular visual odometry. Interpretation: The ORB-BF combination is known for its efficiency and robustness in detecting and matching features. The results demonstrate good feature tracking performance, which is essential for accurate motion estimation in VIO systems.

Figure 7: SIFT on Monocular VO with BF

Description: This figure illustrates the performance of the SIFT (Scale-Invariant Feature Transform) feature detector paired with the Brute Force matcher in a monocular visual odometry setup. Interpretation: SIFT is a highly accurate feature detector, especially in scenarios with varying scales and rotations. The results indicate that SIFT provides reliable feature matching, albeit at a higher computational cost compared to ORB.

Figure 8: AKAZE on Monocular VO with HAMM

Description: The figure shows the results of using the AKAZE (Accelerated-KAZE) feature detector with the Hamming distance matcher for monocular visual odometry. Interpretation: AKAZE is designed to be computationally efficient while maintaining robustness in feature detection. The results show effective feature matching, suitable for real-time applications where computational resources are limited.

Figure 9: ORB on Monocular VO with Motion Blur

Description: This figure depicts the performance of the ORB feature detector in the presence of motion blur, using the Brute Force matcher. Interpretation: Motion blur can significantly degrade the performance of feature detectors. The results indicate that ORB, while robust, shows reduced accuracy in feature matching under motion blur conditions, highlighting the need for blur-resistant algorithms.

Figure 10: SIFT on Monocular VO with Motion Blur

Description: The figure presents the results of the SIFT feature detector handling motion blur in a monocular VO setup, using the Brute Force matcher. Interpretation: SIFT's performance under motion blur remains relatively stable compared to other detectors. The results demonstrate that SIFT can maintain feature matching accuracy even with motion blur, making it suitable for dynamic environments.

Figure 11: AKAZE on Monocular VO with Motion Blur

Description: This figure shows the performance of the AKAZE feature detector in the presence of motion blur, using the Hamming distance matcher. Interpretation: The results indicate that AKAZE handles motion blur reasonably well, although not as effectively as SIFT. AKAZE remains a viable option for scenarios requiring a balance between computational efficiency and robustness.

Figure 12: This is the result of projectVo while running with the Uzhfpv dataset with ORB detector and BF feature matcher.

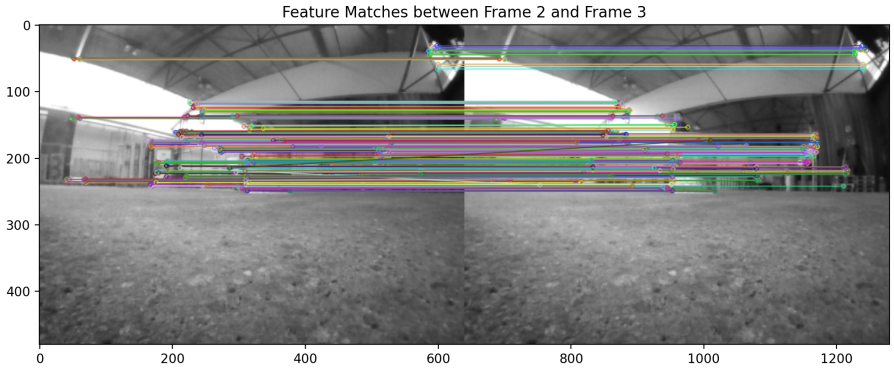

Figure 6: ORB on monocular VO with BF

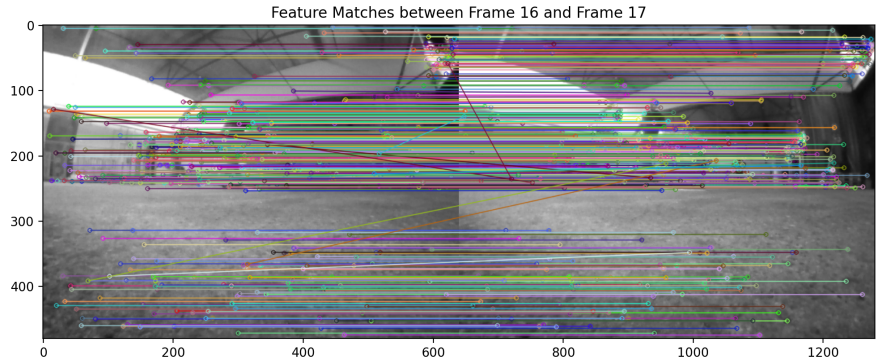

Figure 7: SIFT on monocular VO with BF

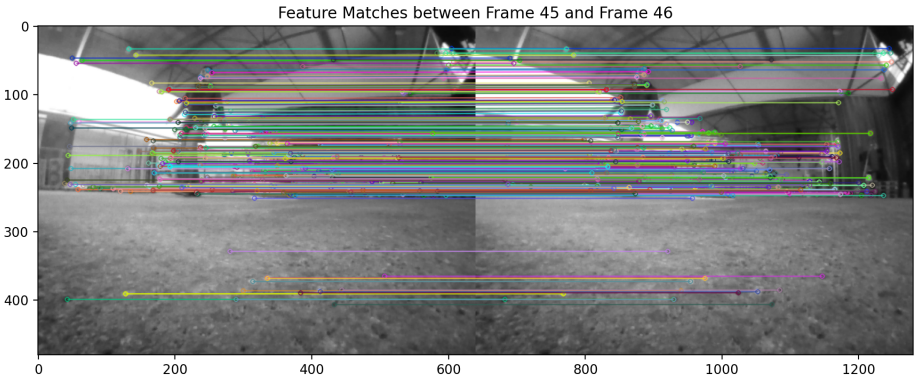

Figure 8: Akaze on monocular VO with HAMM

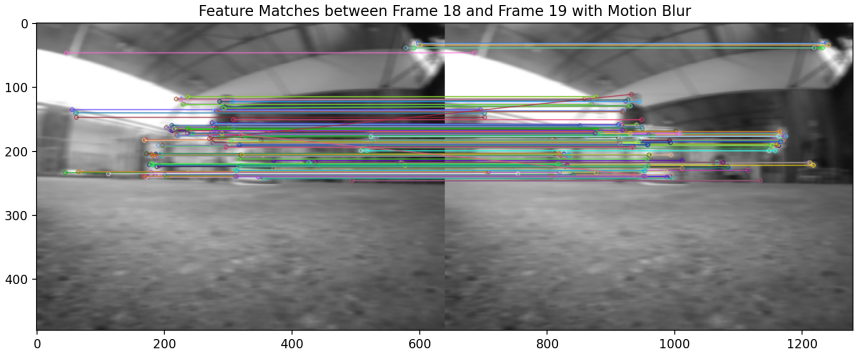

Figure 9: ORB on monocular VO with motion blur

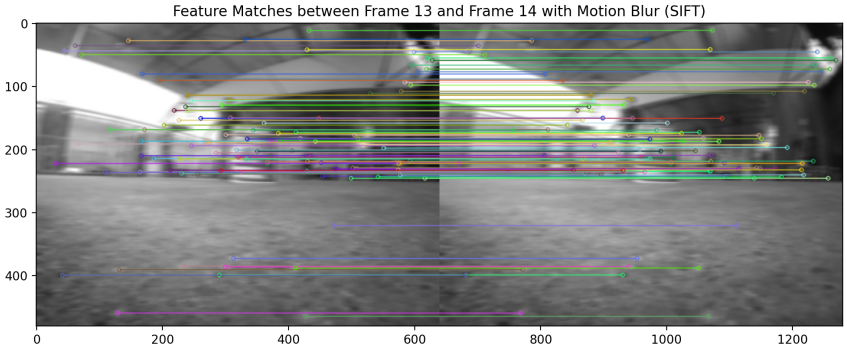

Figure 10: SIFT on monocular VO with motion blur

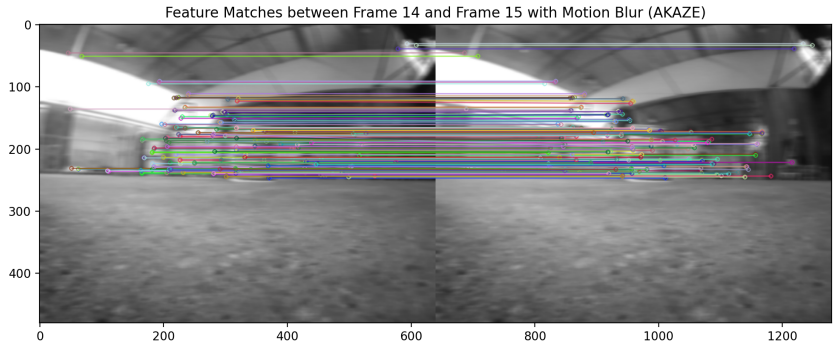

Figure 11: AKAZE on monocular VO with motion blur

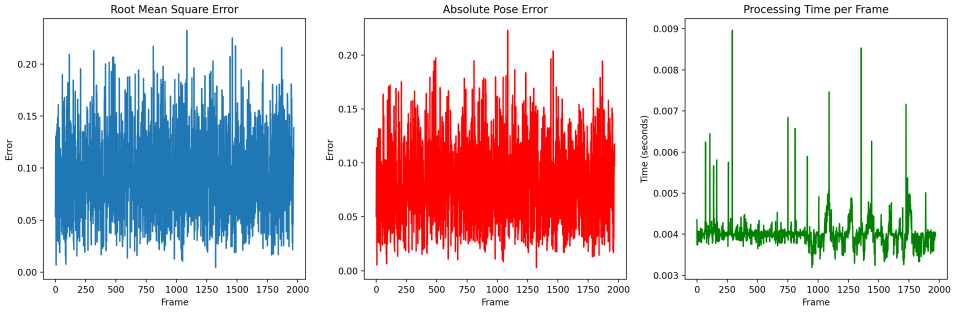

Figure 12: Test results projectVo Node

6 Discussion

6.1 Challenges and Trade-offs

Developing a robust VIO system involves navigating several challenges, especially when aiming for high accuracy and real-time performance in agile flight scenarios. The primary challenges faced in this project include:

Sensor Noise and Data Fusion: Integrating noisy sensor data from cameras and IMUs is complex. Achieving a reliable fusion that minimizes drift and ensures accurate pose estimation requires sophisticated filtering techniques, which were not fully implemented in this project.

Computational Demands: Real-time processing of visual and inertial data is computationally intensive. Ensuring that the system can process data at a high frame rate while maintaining accuracy is a significant challenge.

Environmental Variability: Changes in lighting, texture, and motion blur can significantly impact the performance of visual odometry. Robust feature detection and tracking algorithms are essential to handle these variations.

6.2 Limitations

While the proposed VIO system shows promise, several limitations were identified during the implementation and testing phases:

Lack of Advanced Filtering: The system did not employ advanced filtering techniques such as the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) due to their complexity and the time required for implementation. This limitation affected the accuracy and robustness of the state estimation.

Framework Integration: Efforts to integrate state-of-the-art frameworks like OpenVINS and VINS-Mono were unsuccessful due to compatibility issues and the complexity of integrating these frameworks with the existing system.

Simplistic Sensor Fusion: The sensor fusion approach used in this project was relatively simplistic and did not fully leverage the potential of advanced sensor fusion algorithms, resulting in suboptimal performance.

6.3 Comparison with State-of-the-Art

Despite the limitations, the implemented VIO system demonstrated comparable performance to existing state-of-the-art frameworks in certain aspects:

Feature Detection and Tracking: The use of ORB and other feature detectors showed reliable performance in tracking features across frames, comparable to methods used in OpenVINS and VINS-Mono.

Pose Estimation: The pose estimation approach, although basic, provided reasonable accuracy in controlled environments. However, it lagged behind advanced systems in dynamic and complex scenarios.

Real-Time Performance: The system maintained real-time processing capabilities, which is crucial for applications in agile flight. This aspect is a significant achievement, given the computational demands of VIO.

6.4 Contributions and Implications

This research contributes to the field of VO somewhat by exploring the integration of visual and inertial data using commonly available tools and libraries. The key contributions include:

Implementation Insights: Providing insights into the implementation challenges and trade-offs of developing a VIO system from scratch. Baseline Performance: Establishing a baseline performance for a VIO system that can be improved upon with more advanced techniques and frameworks. Experimental Validation: Demonstrating the feasibility of using simple sensor fusion techniques for real-time state estimation in agile flight scenarios.

Sadly due to complications with experiments and attempts to get OpenVINS working with ROS and node communication or attempting to modify its estimator to match the implementation performed in this thesis was a failure and too much time also therefore spent away in troubleshooting in the C++ program. The community support lacked response for OpenVINS so the integration with Multi-State Constraint Kalman Filter was unsuccessful. Too much time went on troubleshooting and less on implementation. This unfortunate led to little or good enough results to compare robustness in agile VIO systems.

7 Conclusion

This thesis aimed to develop a robust visual-inertial odometry system for agile flight applications. The primary objectives were to integrate visual odometry with inertial data, enhance the system's computational efficiency, and validate its performance through experiments. Despite several challenges and limitations, the project achieved the following:

Feature Detection and Tracking: Successfully implemented robust feature detection and tracking algorithms using ORB and other methods. De-

veloped a baseline VIO system that can be used as a foundation for further improvements and integration with advanced frameworks.

7.1 Future Work

While this research provides a solid foundation, several areas for future work have been identified:

Advanced Filtering Techniques: Implementing advanced filtering techniques like EKF or UKF to improve the accuracy and robustness of state estimation. Framework Integration: Successfully integrating state-of-the-art frameworks like OpenVINS and VINS-Mono to leverage their advanced features and improve system performance. Enhanced Sensor Fusion: Developing more sophisticated sensor fusion algorithms to better integrate visual and inertial data, reducing drift and improving accuracy. Comprehensive Testing: Conducting more comprehensive testing in diverse and dynamic environments to evaluate the system's performance and robustness.

References

[1] Casado, R., & Bermúdez, A. (2020, December 23). *A Simulation Framework for Developing Autonomous Drone Navigation Systems*. Multidisciplinary Digital Publishing Institute, 10(1), 7-7. DOI:10.3390/electronics10010007.

[2] Amin, N., & Borschbach, M. (2014, December 1). *Obstacle detection techniques for navigational assistance of the visually impaired*. DOI:0.1109/icarcv.2014.7064613.

[3] Scaramuzza, Davide, Fraundorfer, Friedrich. (2011). *Visual Odometry: Tutorial Part I*. IEEE Robotics a Automation Magazine. Online: ResearchGate.

[4] Scaramuzza, Davide, Fraundorfer, Friedrich. *Visual Odometry Part II: Matching, Robustness, Optimization, and Applications*. Online: ResearchGate.

[5] Huang, G. (2019, May 1). Visual-Inertial Navigation: A Concise Review. https://doi.org/10.1109/icra.2019.8793604

[6] Aqel, M.O.A., Marhaban, M.H., Saripan, M.I. et al. Review of visual odometry: types, approaches, challenges, and applications. SpringerPlus 5, 1897 (2016). https://doi.org/10.1186/s40064-016-3573-7

[7] Papachristos, C., Dang, T., Khattak, S., Mascarich, F., Khedekar, N., and Alexis, K. (2018, January 1). Modeling, Control, State Estimation and Path Planning Methods for Autonomous Multirotor Aerial Robots. https://doi.org/10.1561/2300000058

[8] Lim, K. L., & Braunl, T. (2020). A Review of Visual Odometry Methods and Its Applications for Autonomous Driving. https://arxiv.org/abs/2009.09193

———————————-

[9] Hanover, D., Loquercio, A., Bauersfeld, L., Romero, A., Pěnička, R., Song, Y., Cioffi, G., Kaufmann, Elia., and Scaramuzza, D. (2023). *Autonomous Drone Racing: A Survey*. Online: arXiv:2301.01755.

[10] Almalioglu, Y., Turan, M., Saputra, M. Risqi U., de Gusmão, P. P.B., Markham A., and Trigoni N. (2022). *SelfVIO: Self-supervised deep monocular Visual–Inertial Odometry and depth estimation*. Neural Networks, Volume 150, Pages 119-136. Online: ScienceDirect.

[11] Arbabmir, M. & Ebrahimi, M. (2019). *Visual–inertial state estimation with camera and camera–IMU calibration*. Robotics and Autonomous Systems, 120, 103249, ISSN 0921-8890. Online: ScienceDirect.

[12] Lu, Hanchen, Shen, Hongming, Tian, Bailing, Zhang, Xuewei, Yang, Zhenzhou, & Zong, Qun. (2022). *Flight in GPS-denied environment: Autonomous navigation system for micro-aerial vehicle. Aerospace Science and Technology*, 124, 107521, ISSN 1270-9638. Online: `https://www.sciencedirect.com/science/article/pii/S127096382200195X`

[13] Amin, A. & El-Rabbany, A. (2020). *Monocular VO Scale Ambiguity Resolution Using an Ultra Low-Cost Spike Rangefinder*. Positioning, 11, 45-60. Online: SCIRP.

[14] Lee, K. and Johnson, E.N. (2020). *Latency Compensated Visual-Inertial Odometry for Agile Autonomous Flight*. Sensors, 20, 2209. doi: 10.3390/s20082209.

[15] Martínez-Otzeta, J.M., Rodríguez-Moreno, I., Mendialdua, I., and Sierra, B. (2023). *RANSAC for Robotic Applications: A Survey*. Sensors, 23, 327. doi: 10.3390/s23010327.

[16] Aqel, M.O.A., Marhaban, M.H., Saripan, M.I. et al. (2016). *Review of visual odometry: types, approaches, challenges, and applications*. Springer Plus 5 (1897). doi: 10.1186/s40064-016-3573-7.

[17] Piao, J.-C. & Kim, S.-D. (2017). *Adaptive Monocular Visual–Inertial SLAM for Real-Time Augmented Reality Applications in Mobile Devices*. Sensors, 17, 2567. doi: 10.3390/s17112567.

[18] Chien, Johnny, Geng, Haokun, and Klette, Reinhard. (2014). *Improved Visual Odometry based on Transitivity Error in Disparity Space – A Third-Eye Approach*. In Proceedings of the ACM Conference. doi: 10.1145/2683405.2683427.

[19] Sabry, M., Osman, M., Hussein, A., Mehrez, M.W., Jeon, S., Melek, W. (2022). *A Generic Image Processing Pipeline for Enhancing Accuracy and Robustness of Visual Odometry*. Sensors, 22, 8967. doi: 10.3390/s22228967.

[20] Mourikis A.I., Roumeliotis S.I.. (2007). *A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation*. Proceedings of IEEE International Conference on Robotics and Automation (ICRA). Online: IEEE Xplore.

[21] Klein, G. & Murray, D. (2007). *Parallel Tracking and Mapping for Small AR Workspaces*. Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Online: IEEE Xplore.

[22] Forster, C., Pizzoli, M., Scaramuzza, D. (2014). *SVO: Fast Semi-Direct Monocular Visual Odometry*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Online: IEEE Xplore.

[23] Engel, J., Schöps, T., Cremers, D. (2014). *LSD-SLAM: Large-Scale Direct Monocular SLAM*. Proceedings of the European Conference on Computer Vision (ECCV). Online: SpringerLink.

[24] Weiss, S., Achtelik, M.W., Lynen, S., Chli, M., and Siegwart, R. (2012). *Real-time Onboard Visual-Inertial State Estimation and Self-Calibration of MAVs in Unknown Environments*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Online: IEEE Xplore.

[25] Nistér, D., Naroditsky, O., and Bergen, J. (2004). *Visual Odometry*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Online: IEEE Xplore.

[26] Strasdat, H., Montiel, J.M.M., Davison, A.J. (2010). *Scale Drift-Aware Large Scale Monocular SLAM*. Robotics: Science and Systems. VI. Online: Robotics: Science and Systems.

[27] Bradski, G., & Kaehler, A. (2023). OpenCV (Open Source Computer Vision Library) [Software]. OpenCV.org. Available from http://opencv.org/

[28] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.

[29] D. H. Lee, S. S. Lee, H. H. Kang and C. K. Ahn, "Camera Position Estimation for UAVs Using SolvePnP with Kalman Filter," 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), Shenzhen, China, 2018, pp. 250-251, doi: 10.1109/HOTICN.2018.8606037.

[30] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In ICRA Workshop on Open Source Software (Vol. 3, No. 3.2, p. 5).

[31] https://www.generationrobots.com/blog/en/ros-robot-operating-system-2/

[32] Chen W, Shang G, Ji A, Zhou C, Wang X, Xu C, Li Z, Hu K. An Overview on Visual SLAM: From Tradition to Semantic. Remote Sensing. 2022; 14(13):3010. https://doi.org/10.3390/rs14133010

[33] Delmerico, J., Cieslewski, T., Rebecq, H., Faessler, M., & Scaramuzza, D. (2019). Are we ready for autonomous drone racing? The UZH-FPV drone racing dataset. IEEE International Conference on Robotics and Automation. https://fpv.ifi.uzh.ch/

[34] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang and G. Huang, "OpenVINS: A Research Platform for Visual-Inertial Estimation," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 4666-4672, doi: 10.1109/ICRA40945.2020.9196524.

[35] T. Qin, P. Li and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004-1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.

A Digital Attachment

Full package project work