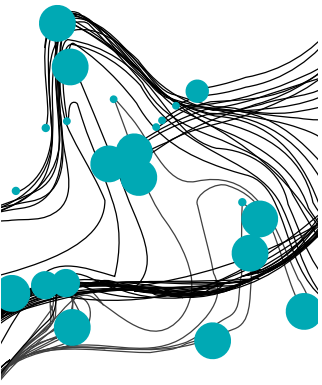
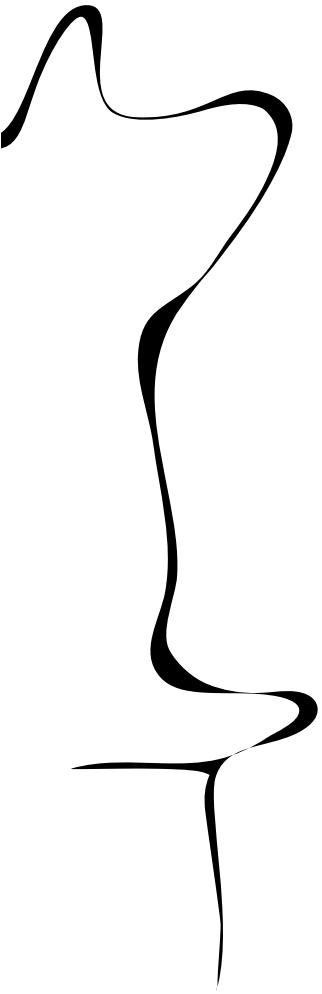


RAM ROBOTICS AND MECHATRONICS



ACHIEVING FULL AUTONOMY IN AERIAL AND PHYSICAL HUMAN-ROBOT INTERACTION CONTROL VIA ONBOARD PERCEPTION ALGORITHMS RELYING ON COMPUTER VISION

M.A. (Miguel) Alonso Calderon

MSC ASSIGNMENT

Committee:
dr. ir. M. Abayazid
ir. H. Das
prof. dr. ir. A. Franchi
dr. N. Strisciuglio
dr. F.C. Nex

June, 2022

015RaM2022
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE
UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

In the context of aerial physical interaction, a branch of aerial robotics focuses on the study and the accomplishment of contact-based interaction and manipulation tasks with aerial robots. The goal of the assignment is to enhance the level of autonomy of the aerial robots developed at RAM-UT, providing them with an accurate localization system to perform a handover to a human. The handover will be performed in both indoor and outdoor scenarios in GPS-denied environments. Currently, the system relies on Motion-Capture (MoCap) state feedback for indoor localization.

In order to find an optimal odometry technique, a comparison of the alternatives was conducted. Consecutively, deep research and evaluation on the conventional open-source VO/VIO alternatives were performed, discarding the machine learning implementations as these techniques cannot outperform the conventional ones for now. This study ranks appearance-based and feature-based methods depending on their convenience for our application.

The insight gained from the analysis of open-source alternatives allows us to select ORB-SLAM2, ORB-SLAM3 and SVO Pro as the most promising solutions to obtain localization in a resource-constraint platform like the MAV. These solutions were tested again the EuRoC datasets which were recorded from a MAV and are broadly used by the research community and against a custom dataset recorded at RAM-UT Arena using Intel Realsense camera D435i. The algorithm evaluation considered the overall performance of the algorithm to estimate the real trajectory, the maximum error achieved to perform human safety and the robustness of the implementations along with multiple runs.

It was concluded that the algorithms SVO Pro stereo, ORB-SLAM3 stereo and ORB-SLAM3 VIO stereo are valid alternatives that would provide optimal performance under the tested conditions, providing both position and orientation of the MAV at each time step.

Contents

1 Introduction

Autonomous robotics has been continuously growing in the past years supported by a strong research community which has led to a big commercial success. Big improvements have been made in multiple fields such as the development of accurate and lightweight sensors, the increase of power storage on batteries or the higher power capabilities of onboard devices. Altogether has allowed achieving affordable and high efficient robotic platforms. This progress has been especially effective in aerial robotics, the small size platforms used in this area are commonly referred to as miniature unmanned aerial vehicles (mini-UAVs) or micro aerial vehicles (MAVs).

Promising applications have appeared in the context of these small-sized airborne platforms, for instance, power line inspections, surveillance tasks, dangerous terrain explorations, transportation, etc., unmanned robots are expected to replace humans when physically performing these tasks in the not too distant future. MAVs must be provided with the capabilities of seeing, thinking and acting to perform these tasks safely and effectively with the least human intervention. Electric power line rehabilitation has always been a tedious and challenging task accomplished by a human. Within this context appears aerial physical interaction, a branch of aerial robotics focused on the study and accomplishment of contact-based interaction and manipulation with aerial robotics. This thesis address how MAVs could help to ease this task by supporting humans with the handover of tools. Before being able to think or act, the MAV needs to find its position concerning a fixed reference frame. Afterwards, it will be able to plan its path (think) and apply controls (act) to reach its target.

Self-localization, which is the first step toward providing full autonomy to a MAV, is the main focus of this work. We will explore the available odometry technologies in robot navigation systems for GPS and GNSS denied environments. The most promising implementations are compared, selecting the one that applies better to resource-constraint platforms. Performing an accurate localization of the platform is key when performing human interaction as the first goal is preserving human safety. Validations of the algorithms are based on real data using available datasets and using custom datasets more specific to our use case.

1.1 Aerial Core Project

This master thesis assignment is within the scope of the European project (1), it is the largest European project in drones, where several European universities and companies cooperate towards a common goal. The main project's ambition is the integration of aerial cognitive robotic systems, providing them with unprecedented abilities to interact with humans safely. This project aims to make feasible co-working between drones and human, with the task of handling the maintenance and inspection of linear infrastructures. The challenges that this project has to face are:

- Accurately inspecting local and long-range infrastructure.
- Developing safe and efficient aerial co-workers for inspection and maintenance operations.
- Providing drones with the capability of accomplishing maintenance jobs that involve force interaction.

The Aerial-coworkers will be proved on the electrical power systems of ENDESA, the project will have a big impact on this field as nowadays the maintenance and inspection of these power systems have not only a big economic impact but also imply improving workers safety and wildlife conservation.

This thesis assignment is part of the fourth stage of the AERIAL-CORE European project. This fourth stage aims to accomplish contact-based interaction and manipulation tasks with aerial robots. The drone must be able to localize itself with reference to the human's position in real-time. The worker must be able to signalize through gestures different instructions that need to be performed by the aerial co-worker, hence, the drone must be ready to detect, recognize and execute any of the orders for smooth human-drone interaction. A robotic arm is mounted at the bottom of the drone's body to allow the required physical interactions such as handover or reception of the tools from the human.

1.2 Problem Statement

This MSc assignment goal is to enhance the level of autonomy of the aerial robots developed at RAM-UT. The available robot at UT should be able to effectively perform handovers to humans in both indoor and outdoor environments in an autonomous manner.

To accomplish this task the aerial vehicle must be provided with an accurate self-localization system. This would grant the system with capabilities to determine its position and pose, required information for obstacle avoidance or object tracking. The Global Positioning System (GPS) is the traditional approach for autonomous navigation. GPS is used worldwide and computes trajectory and speed making use of radio signals. It became available for civilians in the 1980s, providing localization with an accuracy of just a few meters (2).

Nonetheless, GPS has several drawbacks that make it inoperative for our application. GPS produces limited data about the vehicle only giving information about linear velocity. Vehicles that move in the 3D space such as underwater or aerial vehicles also require angular velocity for an accurate self-localization. Moreover, this traditional solution is not accurate enough for autonomous navigation systems, where its highest accuracy causes an error of few centimetres. The system is quite sensitive as the signal strength highly depends on the surrounding environment, for instance, the forest has weaker signals than urban areas and the signal is corrupted by walls or other objects which also make it unsuitable for indoor navigation. The final drawback is the multipath effect, the Global Positioning System is not always the most robust as it requires the signal from multiple satellites simultaneously to provide an accurate estimation. Atmospheric conditions may affect the signal travel from GPS to the receiver causing delays in

the signal reception. In the past years, Visual odometry (VO) and Visual inertial odometry (VIO) have called researchers attention as an efficient and cost-effective alternative for autonomous navigation. The advantage of these vision-based localization systems over GPS is their independence from external resources, namely, it relies on local sensors to retrieve its position and pose relative to its starting point.

Currently, drone localization totally relies on indoor Motion-Capture (MoCap) state feedback. Therefore, an extensive analysis of the Visual-Inertial Navigation System (VINS) techniques available in literature was conducted and then the most promising implementations were selected according to the project requirements. Hardware components have to be effectively chosen and used to implement the selected VINS method. Both hardware and software components were integrated into the software architecture used in RAM-UT during the implementation phase, and finally, they were extensively tested using custom datasets. The final software implementation must be able to:

- Localize itself effectively with respect to the human.
- Generate a map of the environment.
- Recognize and understand human's instructions to navigate to the correct location and perform the given task.

Most of the available VINS solutions are focused on either applying the algorithm to run against open datasets, performing real-time indoor or outdoor robot navigation using a powerful CPU. On the other hand, this assignment approaches the problem from a different perspective, RAM solution has been implemented on a MAV to navigate at both indoor and outdoor environments, depending on limited software performance. The reason of this constraint performance on MAVs is due to the correlation between weight and computational capabilities of hardware. As MAVs need to lift all its weight to hover then the lighter the longer time of flight (ToF) that will be able to achieve.

This Master thesis project is application-based, where the main focus is evaluating and integrating the most promising onboard localization solutions for MAV with the final goal of performing a handover to a human. The handover should be performed in both indoor and outdoor scenarios at GPS-denied environments.

The following research questions are under the frame of the main research question. The questions will guide us to address the main research questions in the best way and fulfilling all the set goals :

- Which are the hardware options to provide vehicles with localization in GPS-denied environments? And which is the most convenient option for MAV?
- Which are the most promising open-source algorithms to provide the MAV with localization capabilities (position and orientation)?
- How different are the conditions in outdoor scenarios? Can the same framework be seamlessly used? How do changes in light affect the estimation?
- Should the solutions integrating Machine Learning be considered as optimal localization method for our use-case?
- In a simplified scenario without too many obstacles, is it necessary to have loop closures as a feature in the VIO/SLAM pipeline?
- Which is the most promising open-source algorithm to provide the MAV with localization capabilities (position and orientation)?

- Is one camera enough to perform vio/slam while also recognizing the man and his/her gestures?
- How should the camera be placed on the MAV? What is the optimal placement? Is the presence of 3DOF constraining the placement?

2 State of the Art

In this section it is provided an overview of the theoretical side of the project. The knowledge gathered in this section explains the concepts behind odometry, visual-inertial navigation systems and how they apply to MAVs to gain an understanding of the topics. It also includes a comparison between the available VINS algorithms in open-source.

2.1 Overview of Odometry techniques

As described in 1.2 a GPS solution is not convenient to perform autonomous navigation due to its limitations. The main drawbacks are its constraints to achieving self-localization with high precision and the influence that environmental conditions have on the satellites' signals. There are many alternatives for GPS-denied self-localization techniques that we will briefly cover in this section, the basic odometry approaches (3) are:

- **Wheel Odometry (WO):** it applies only to ground vehicles with 2 or 4 wheels. The position and orientation computation is found on wheel encoders that count the number of revolutions each wheel makes (4). Each side of the vehicle needs to work independently in different directions and speeds. The main drawbacks are that his method accumulates position drift error over time and, position and orientation estimations decrease with irregular terrains and slippery surfaces.

- **Inertial Odometry (IO):** An Inertial Measurement Unit (IMU) is used to estimate the altitude, linear velocity, orientation and position of the robot relative to the starting point. A gyroscope measures orientation through magnetism and gravity detection while non-gravitational accelerations are estimated with an accelerometer (5). They are highly used on resource-constrained platforms such as MAVs due to their low power consumption and the small size of this MEMS sensor. The main drawbacks of IO are its drifting errors and bias on the accelerometer and gyroscope measurements, thus, this method is unsuitable for lifelong localization applications.

- **Radar Odometry (RO):** an onboard radar sensor is in charge of generating and interpreting scans to retrieve the relative motion of the vehicle. The surrounding objects are detected through the radio waves. The main advantages of this technique are its low sampling rate, the small range to target for detection and the low power consumption. Moreover, radars are immune to unfavourable environmental conditions or low textures. Nonetheless, the main disadvantages are its deficient performance on irregular terrains and the high storage required when performing large-scale mapping, then RO is not advisable for resource-constraint platforms.

- **Laser Odometry (LO):** it produces light emissions and tracks laser speckle patterns coming from the surrounding objects on the 2D plane and uses consecutive 2D images to build the 3D reconstruction using the iterative closest point method (ICP). Subsequently, both position and orientation are retrieved from these measurements. LO performance is not distressed either by low textures or environmental light. However, this solution is difficult to implement on resource-constrained platforms given the need for matching points of two sets, therefore, it is computationally expensive. Also, in presence of dynamic objects, distortion appears on the scan, this effect needs to be corrected inducing poor performance.

- **Visual Odometry (VO):** estimates the ego-motion of a vehicle through the analysis of the sequence of images of the surrounding environment which are captured by one or mul-

Localization Technique	Real Time	Power	Accuracy	Energy	Rosbustness	Dimensions
GPS	Soft	Low-power	Semi-accurate	Non-efficient	High	2D
WO	Hard	Low-power	Non-accurate	Efficient	Low	2D
IO	Hard	Low-power	Non-accurate	Efficient	High	3D
RO	Hard	Low-power	Accurate	Efficient	High	3D
VO	Firm	High-power	Accurate	Non-efficient	Low	3D
LO	Hard	High-power	Accurate	Non-efficient	Medium	3D
Filter-based VIO	Firm	High-power	Accurate	Non-efficient	Medium	3D
Optimization-based VIO	Soft	High-power	Accurate	Non-efficient	Medium	3D
Loosely-coupled VIO	Firm	High-power	Non-accurate	Non-efficient	Low	3D
Tightly-coupled VIO	Soft	High-power	Accurate	Non-efficient	Medium	3D

Table 2.1: Comparison of the most common localization techniques.(3)

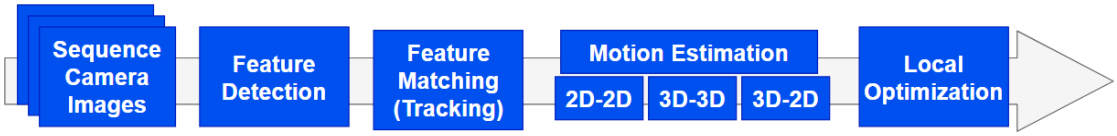


Figure 2.1: VO pipeline

tiple cameras. This localization technique is highly affected by environmental conditions such as blur, frame drops or environmental light.

A much more extensive comparison of these algorithms was conducted by (3), where they are also evaluated in terms of accuracy, robustness, energy efficiency, performance and response time.

2.2 Visual Odometry

The simplicity of visual odometry to perform state estimations, along with its appropriateness in resource-constrained platforms has motivated robotic researchers to explore and make progress on this odometry approach, maximizing its reliability and robustness. It is an odometry algorithm able to compute the vehicle orientation and position by incrementally analysing the deviation on a sequence of images as a result of motion. This deviation is computed by detecting features within the images and tracking their displacement through the input frame sequence. As a final step, some local optimizations can be made over the last number of poses (windowed bundle adjustment). Applying VO implies taking into account several assumptions (6): sufficient illuminations are needed, it should check that overlapping is sufficient between consecutive frames, the static scene should have high texture and predominate over dynamic objects to be able to remove apparent motion.

VO is a subset inside a technique well-known as Structure from Motion (SFM) (7). This technique reconstructs the structure of 3D scenes by making use of sets of unordered images, the algorithm optimizes the camera and computes the position of the feature points. The final step at SFM consists of an optimization which is done offline as the refinement of the scene reconstruction and camera pose is a computationally expensive operation that grows with the number of images.

In Figure 2.3 it is represented the general pipeline for visual odometry. Now it is reviewed the processes required in VO to complete a successful localization in real-time.

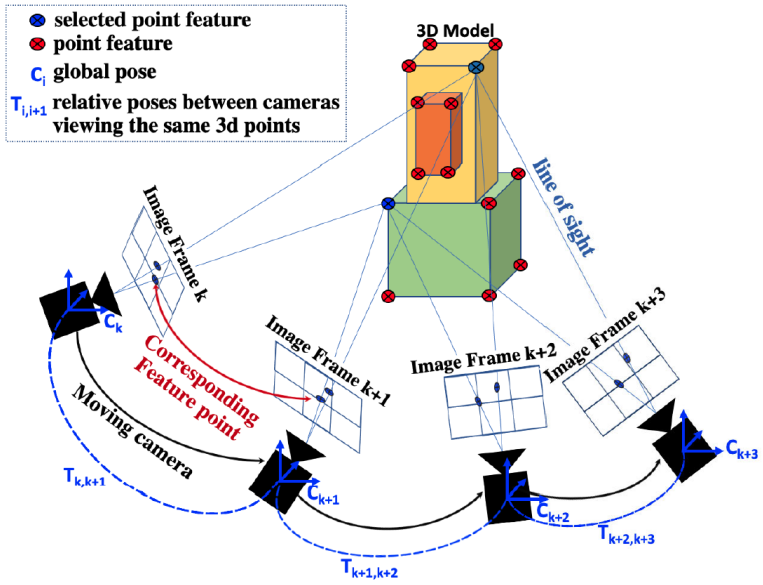


Figure 2.2: Ilustration of VO scheme.(8)

Vision Systems

Numerous types of camera sensors and configurations can be applied to capture the sequence of images from the environment, each sensor has its benefits and limitations (3).

- Stereo: the 3D information is reconstructed from a pair of images. Two cameras are aligned and calibrated with respect to the projective centre of each camera, at least two points have to be projected on the camera projection planes to build the absolute scale by applying a technique called triangulation. This approach consists of matching features in a stereo pair of images and aligning them along an epipolar line, this method is also known as epipolar geometry. The disadvantage of stereo cameras is the need of performing a very precise extrinsic calibration, this calibration gets degraded as an effect of vibrations or sharp movements so over time a re-calibration is required. Stereo-cameras count with a fixed baseline distance which affects the depth estimation, the bigger this distance the better accuracy to detect larger depths, oppositely to detect very close objects an ultra-short baseline is convenient.

- Monocular: the baseline issue is not present in this type of camera. At least three successive frames are required to do the 3D reconstruction. The main disadvantage is that the translation vector is computed concerning a relative scale since the transformation matrix between the two initial frames is not fully known, so a predefined value is set. However, several ways have been developed to resolve the drift and ambiguity on the scale such as using the information on the planarity of surfaces or making use of convolutional neural networks (CNN).

- RGB-D: single camera that counts with an IR transmitter that projects speckle patterns on the surfaces and saves the IR image as a reference image. Successively, the reference image and the captured image are triangulated to compute a real depth estimation, this method helps to overcome scale ambiguity when using a single camera. RGB-D cameras are advantageous to obtain poses in low texture environments. However, they have a significant drawback, their depth sensor is only accurate when measuring depths in the range of 0 and 3 meters (9), which can lead to scale accumulative error.

- Omnidirectional: this camera provides a 360 field of view (FoV) boosting accuracy pose estimations in comparison to cameras with lower FoV, as a higher area of the surroundings is captured.
- Fisheye: allows a panoramic view of almost 180 degrees from side to side, as it happens with omnidirectional cameras it also allows to capture more information from the environment compared to traditional cameras. The corresponding pixels on fisheye-stereo cameras are projected on epipolar curves, preventing the usage of traditional disparity algorithms. Fisheye cameras imply higher computational costs due to the epipolar curves.
- Thermal Camera: these cameras are promising for low visibility environments, they capture the surrounding temperature environment without the need for an additional source of light. On the other hand, these cameras accumulate noise over time and low-resolution images (8).
- Event Based: these are dynamic vision sensors able to capture intensity changes across the pixels on the camera, these changes are also known as events. The camera output works asynchronously as it is triggered by any variation in the scene illumination. These cameras outstand for their high dynamic range and low latency compared to regular cameras. Additionally, blur does not influence the output because the value of each pixel is calculated independently.

Camera sensors can either be placed downward- or front-facing. The front-facing solution captures more information about the surrounding environment, while shows deficiencies to capture small movements. Furthermore, this position is more prone to be affected by shadows or sunlight changes. On the other hand, the downward-facing position is useful for pre-explored maps but, has difficulties finding appropriate matching points when the are fast dynamics. Normally, it is seen that cameras pointing downwards generate 2.5D maps due to the flatness of the ground plane. Meanwhile, front-facing cameras are normally applied to build 3D reconstructions, as they are better for indoor navigation and can detect better the obstacles such as buildings or trees.

Problem Formulation

The chosen camera is responsible of taking images recurrently from the environment. The relation between two successive camera poses *k-1* and *k* can be established through rigid body transformation (6). The set $T_{1:n} = T_{1,0}, ..., T_{n,n-1}$ contains all the transformation matrices of the previous camera poses, where:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

where $R_{k,k-1}$ represents the rotational matrix from pose $k-1$ to pose k, while $t_{k,k-1}$ represents the translation on the 3D space from pose $k-1$ to the pose k. The camera successive camera transformations with respect to the initial frame $k = 0$ are contained by the set $C_{0:n} = C_0, ..., C_n$, hence, the ongoing position C_n can be deduced from concatenating the previous transformations form $k = 1$ to $k = n$, this means $C_n = C_{n-1}T_n$.

Feature Extraction and Tracking

VO's main task is to find the relative transformations between camera poses and then concatenate them to recover the full trajectory. Several approaches can be used to extract the data contained in the captured images, more conventional ones such as appearance-based (direct method) or feature-based (indirect method) and the non-conventional approach is applying machine learning.

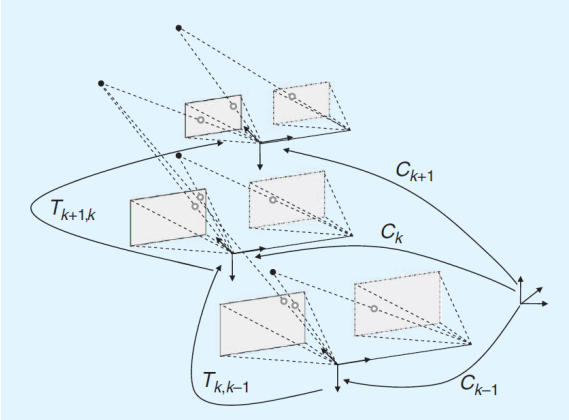


Figure 2.3: VO problem, where the transformation between adjacent camera positions are computed from the matching features. As a result, the absolute poses C_k can be deduced regards the initial frame.(6)

Appearance-based/Direct methods: Camera poses are estimated through the minimization of the photometric error between the pixels of two consecutive frames. All the frame information is used and the feature extraction step is skipped, allowing to quickly gather the mapping data. Direct methods make use of geometric constraints from the image frame, reducing aliasing problems with other similar patterns and improving accuracy and robustness in low textured or low visibility situations. The drawback of this method is its lack of robustness to moving objects because very close pixels may cause mismatches while pixels with high depth are sensitive to calibration errors, desynchronisation between depth and colour or the rolling shutter effect.

This method can be classified into *optical-flow based* and *region/template matching based*: the former uses an optical flow algorithm which uses as input the raw pixel data when the intensity of a pixel changes between two consecutive images then this pixel is analyzed to estimate camera motion. With the intensity pixel variations, it is possible to compute the 2D displacement vector of the projected points of the two frames. One of the most relevant implementations of this OF algorithm is the one that (10) proposed, where the 3D points of the global map between the two frames get aligned by an iterative closest point (ICP) algorithm. These algorithms have weak performance in low texture environments and are computationally expensive. On the other side, *regional-based methods* aligns consecutive images to estimate the motion between two ensuing camera poses. For instance, an algorithm was proposed by (11) where a correlation mask is applied to the size and location of the objects, this data is used to predict the position of the object in the next frame. Later on, an extension to apply rotation and translation to the mask was suggested by (12).

Feature-based/Indirect methods: Use outstanding and repeatable features from each frame to approximate the motion between camera poses. This approximation is based on geometric error minimization over the matched features between frames. Indirect methods start by identifying areas where key features can be found such as corners, edges, lines or curves, often intensity gradients are employed to detect these key zones. Some of the most famous feature descriptors are Harris, SURF, SIFT, FAST and ORB, the choice between them is driven by the trade-off between computational efficiency and feature detection robustness. Descriptors have been optimized to be robust against noise and geometric distortions. After the descriptors have detected the key features an optimization method is used to minimize the geometric error between them along the sequence of images. Lastly, the camera poses are estimated with the transformation matrices. Feature-based methods achieve high robustness in environments

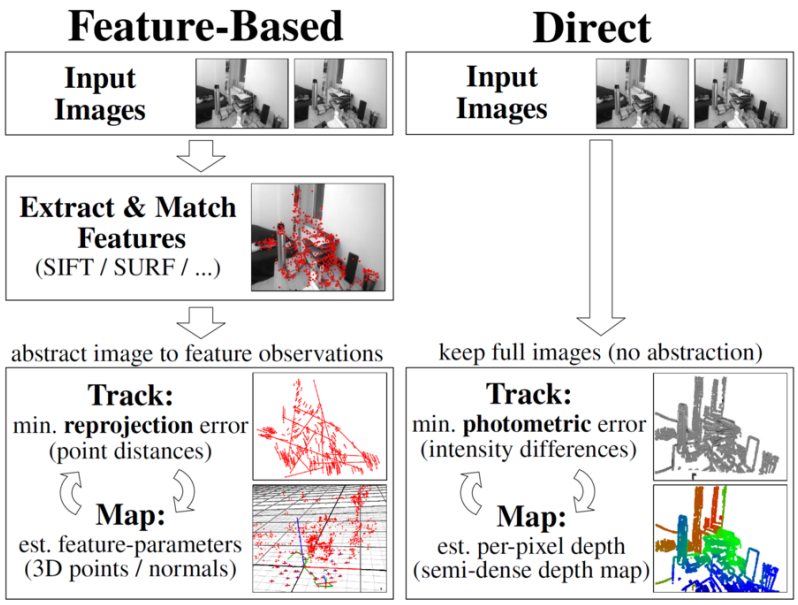


Figure 2.4: VIO: direct methods vs feature-based methods

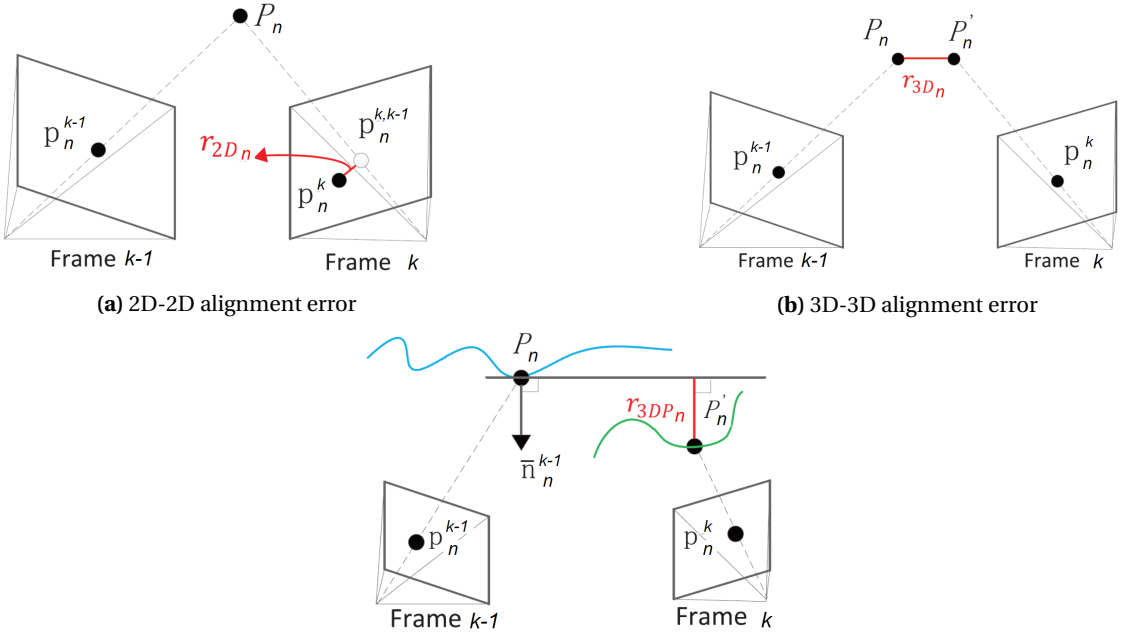
with high geometric distortions. This method is independent of illumination variations. The detection, matching and tracking of features is a computationally expensive operation that increases proportionally with the extracted number of features, nonetheless, the pose estimation accuracy increases proportionally as well.

The main focus of this section is set on conventional models, discussing machine-learning-based implementations in Section 2.5, more information about any of these approaches can be found in (3). The input space is linked to the election between direct and indirect methods, on the other side, the output space depends on the election between sparse and dense maps (13). Sparse maps are typically used for correcting the trajectory in vSLAM algorithms, these maps are composed of just a small subset of the pixel contained in the image frame. Contrary, dense maps generate maps that contain all the possible pixel information from the image frames, creating these maps implies a heavy parallel computation and it is very probable that a GPU is needed to produce these dense maps. The most common combinations for input/output solutions are indirect/sparse methods and direct/-dense methods.

Motion estimation

Once the matching features are detected in consecutive images (f_k and f_{k-1}) then the corresponding transformation matrices need to be computed. Finding this matrix between two camera poses depends on the dimensional space in which the features are defined, either in 2D or 3D space. As a result, there are 3 different cases, (13; 8):

- 2D to 2D: the points at both f_k and f_{k-1} are represented on the 2D plane, at least 5 matching points are needed. The solution involves finding the transformation matrix that can minimize the distance between the point reprojection and its matching point in the reference frame (see Figure 2.5a).
- 3D to 3D: the points at both f_k and f_{k-1} are represented on the 3D space, therefore, it is needed to triangulate points at each time step. Given 3D points in the real world which are mapped to their 2D projection, it is possible to estimate the camera pose. The optimal solution is computed by the minimization of the error from the euclidean distance



(a) 2D-2D alignment error

(b) 3D-3D alignment error

(c) 3D-2D alignment error

Figure 2.5: Different types of geometric error (14)

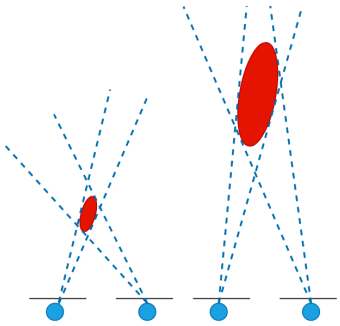
.

between the triangulated 3D matching points, Figure 2.5b. This method provides higher uncertainties than 3D-2D, so it is rarely used.

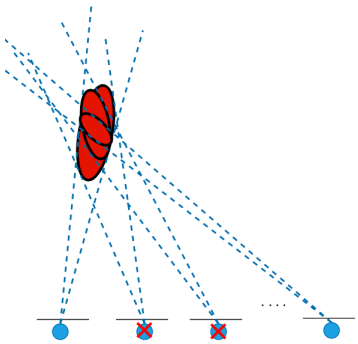
- 3D to 2D: also called 2.5D alignment or PnP (perspective from n points), where the points of the frame f_{k-1} are specified in 3D space and f_k points on the 2D plane. At least 3 points are needed for the minimal case to be solved. The points in the 3D space are reprojected on the 2D space of the new frame, the solution is the transformation where the euclidean distance between the tangent plane and the reprojected reference point is minimal, Figure 2.5c.

Ultimately, 3D-2D alignment estimation is more computationally efficient than 2D-2D alignment (15). Moreover, with 3D-2D alignment camera poses are estimated with a higher level of accuracy than 3D-3D alignment, as it minimizes the reprojection error instead of computing the euclidean distance between the 3D matching points. For more mathematical details please refer to (6; 14). Stereo cameras can capture motion on an absolute scale and generate less drift compared to monocular cameras, still when the stereo baseline is too small compared to the distance to the objects then VO gets degraded into monocular VO.

Some of these motion estimation approaches require the triangulation of their 3D points. The triangulated 3D points are found by intersecting the back-projected rays from 2D image correspondences of at least two image frames. However, this intersection never occurs, leading to uncertainty areas around the feature point position. This effect is a consequence of image noise, deficiencies in the camera model and calibration estimations or the uncertainty around feature matching. There are different ways of reducing this uncertainty, the first one is by having more observations or more distant views that can decrease the position uncertainty. Another alternative is by applying Keyframe selection, this procedure consists of skipping frames until the average uncertainty is under a chosen threshold, the selected frames are the named keyframes. Many times wrong data associations are made, these outliers must be removed to increase robustness. Traditionally this is done by the well known RANSAC algorithm (16), which is a non-deterministic method as the solution differs from each run.



(a) Representation of how uncertainty changes with distance when applying triangulation.



(b) Keyframe selection. Approach needed before each motion update to mitigate drift.

Windowed Camera-Pose Optimization

It is possible to graphically represent camera poses as nodes and their rigid-body transformations by using edges connecting these poses. The pose graph can provide information about the constraints from the camera poses by observing feature points from different camera positions (17; 18; 4). The final goal is the minimization of the cost function (Equation 2.2) by rearranging the camera poses to satisfy the given constraints.

$$\sum_{e_{ij}} \| C_i - T_{e_{ij} C_j} \|^2 \tag{2.2}$$

where C represents a camera pose and T_{ij} the transformation matrix between i and j. Rotations introduce nonlinearities in the optimization and therefore nonlinear optimizers need to be used such as Lovernberg-Marquardt. This optimization can be done not only between consecutive frames but also between non-adjacent frames which are inside a window of the m last keyframes, the optimization process is also known as *Windowed Bundle Adjustment*. The process of removing previous frames so they are not considered during the optimization is called Marginalization (19), refer to the paper (20) for information regarding marginalization with VIO techniques.

2.3 V-SLAM

V-SLAM stands for *Visual Simultaneous Localization and Mapping*, as its definition suggests this process describes the navigation of a vehicle in an unknown environment inferring its own location and creating a map of the surrounding environment where it positions itself. VO can be considered as a reduced SLAM system, where place recognition (loop closure) is disabled.

Previously, it was described how the VO objective is recovering the path pose after pose, and before every motion update, it optimizes over the last m camera positions (window bundle adjustment), achieving local consistency trajectories. The key to creating a robust map is loop closure detection. It allows the reduction of accumulative VO drift and the position of camera poses through global bundle adjustment optimization, accomplishing global consistency for both the trajectory and the map (18). The trade-off between VO and V-SLAM is simplicity, performance and consistency.

There are mainly two types of VSLAM algorithms (21):

- Filtering-based: the problem is modelled as an online state estimation, where states are estimated as soon as a new measurement comes available using Maximum a Posteriori (MAP) estimator. These models are similar to those initially used on the SLAM problem. They can be based on the Extended Kalman Filter (EKF), for instance MonoSLAM (22), on particle filters such as FastSLAM (23; 24) or Multistate Constraint Kalman Filter (MSCKF)

(25) or its stereo version, the S-MSCKF (26). Traditionally, filtering-based methods concatenate on the same state vector the camera poses and landmarks positions, this turns out to be inefficient in terms of scalability. As in the classic SLAM, when a landmark is revisited then the state vector updates its parameters. When applying this V-SLAM algorithm the 3D features are extracted from the state vector and stored in a static map. The mapping process takes place after the localization is finished rather than simultaneously. Filtering methods evolved and nowadays permit windowed optimizations improving computational efficiency.

- Smoothing: in which most of the recent SLAM algorithms are based, it estimates the full trajectory from the complete set of measurements. These methods are based on parallel methods which derive from PTAM (27). These algorithms are also known as "optimization-based" because the features are parametrized with respect to the keyframes enabling parallelization of tasks (multi-threading). Smooth algorithms perform a global optimization process known as bundle adjustment (BA) which corrects drift effects. This non-linear least-square optimization problem can be addressed using multiple solvers (i.e.,iSLAM2 (28), g2o (29) ,Ceres(30)). These algorithms were mainly conducted offline due to the computational costs of BA optimization until PTAM (27) appeared proposing the parallelization of tasks. BA is capable of optimizing both camera poses and the built map, however, it is also possible to limit its optimization process to motion-only BA and structure-only BA. Besides, BA optimization can be also combined with other optimizations such as the Windowed camera-pose optimization. The keys to achieving high accuracy when applying BA to sparse feature correspondence are: 1) a Large number of uniformly dispersed features in the image plane. 2) Tracking feature in multiple frames with small drift. 3) Association of old landmarks with newly detected features (i.e.,loop-closures).

To sum up, filtering-based algorithms are solved incrementally in a single thread, while Smoothing-based algorithms use multi-thread and are based on keyframe-based solutions using both windowed and global optimization techniques. While the latter has higher accuracy, the former holds a higher computational speed.

2.3.1 Loop Detection for Global Bundle Adjustment

As depicted in Figure 2.7, a simple way of visualizing loop closure is considering that when the vehicle has loop closure deactivated it sees the world as an "infinite corridor", so the vehicle is continuously exploring new areas, even when revisiting previous landmarks. Contrarily, when loop closure is activated the vehicle can visualize the world as a "corridor" that keeps intersecting itself. An intersection occurs whenever an old landmark is revisited, this gives him an understanding of the topology and helps to find shortcuts.

Loop closures are a valuable tool for pose-graph optimization, when a landmark is revisited a new constraint is created between graph nodes. Usually, these nodes are far from each other and drift has accumulated between them. This drift in position is highly reduced and all the past poses are updated gaining a higher understanding of past vehicle positions, and it is at this point when *Global Bundle Adjustment* happens. However, special care should be taken when adding a new loop closure due to the high impact of the new constraints on the camera-pose optimization, it is preferable to omit a loop closure than add a wrong one.

2.3.2 V-SLAM Architecture

Modern V-SLAM systems are divided into two main components:

- Front-end: this part is focused on receiving the input from the visual sensors to extract features and make data associations, this data association can either be short-term

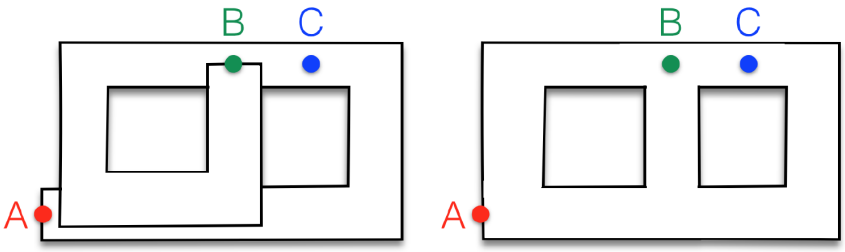


Figure 2.7: The left map does not consider loop closure, it starts in A and ends in B. In the built map points that might be close in reality such as B and C are seen as far from each other in the odometric map. The right map leverages loop closure and understand the real topology of the environment, finding shortcuts within the map (21)

or long-term associations. Short-term associations are the ones that match 3D feature points from consecutive frames, this is done by means of descriptors or optical flow (6) methods as commented in Section 2.2. Oppositely, Long-term data association is a more challenging task as it detects and validates loop closures. It would be impractical trying to match by brute force a detected feature against all the known features, therefore a more practical approach named bag of words is applied.

- Back-end: provides feedback to the front-end for loop closure detection and validation and is encharged of motion and structure optimizations. The final goal is to detect loop closures robustly avoiding perceptual aliasing, this phenomenon consists of having different input from sensors that cause the same sensor signature. Nevertheless, visual aliasing is unavoidable, wrong loop closures degrade the pose graph estimation (31), so there are some proposed techniques (31; 32; 33; 34) to protect the back-end against false positives. For instance, one of the methods proposed was checking the loop closure against the odometry measurements (35).

Bag of Words Approach

As mentioned, trying to match every feature against all the others by brute force seems an impractical procedure, alternatives have been found to do this process more efficiently. One of the most well-known methods is the so-called visual words (18; 36; 37; 38; 39). In this method every visual feature can be represented by a bag of visual words. This method simplifies the representation of a high-dimensional feature descriptor with just an integer number. The initial high dimensional descriptor space is divided in k-means clusters (40), named as visual vocabulary. Each feature descriptor that falls into the same cell is assigned with the same cell number, representing a visual word. The visual similarity between two images is measured by using the distance of the visual word histograms between them. Initially, the first n-similar images are found, then a geometric verification. In V-SLAM the method applied to reject outliers is RANSAC (6), once this is checked the maps and poses are corrected and the transformation pose between the two frames is computed. In Section 2.3.2 we commented how this aliasing is unavoidable and some ways to tackle this have been proposed.

Several methods can be used to search within the bag of words. TF-IDF (41) were the words are classified in terms of the frequency in which they appear on the images, the more they appear the easier to recognise. Other methods arrange visual words on hierarchical trees (42) which work as a lookup table. These approaches are not able to handle illumination variations, therefore there are techniques with account for these variations by matching sequences (43) or consider both spatial and appearance information (44).

2.4 Visual Inertial Odometry (VIO)

These algorithms are state estimations that fuse the camera images with IMU measurements to provide an accurate motion tracking of the mobile platform.

The inertial measurement system is a lightweight and low-cost module that provides local linear acceleration and angular velocity. The IMU data is corrupted by biases and noise, suffering a poor signal-noise ratio at low angular velocities and linear accelerations (45). Inertial-based odometry methods deteriorate with time, making the position estimation unreliable for lifelong experiments. On the other hand, camera-based approaches can provide precise and rich information during slow-motion displacements, but suffer from a very limited output rate (100Hz) and have scaling problems in their monocular setup. The quality of the pose estimations highly decreases in low textured environments, when features are occluded, under High Dynamic Ranges (causes underexposure of images) or with high-speed motions (causes motion blur).

The integration of the VO pipeline with these inertial-based approaches helps to overcome these limitations on the VIO systems. In addition, it helps to make the scale observable for monocular systems. The IMU provides high-frequency data (1000Hz) for short term estimation, cushioning the impact of dynamic scenes on the visual approach. Both methods complement each other producing a more robust and accurate state estimation.

2.4.1 VIO Classification

VIO solutions can be classified based on the stage at which the sensor fusion takes place loosely-coupled and tightly-coupled. Also, VIO methods can be categorized depending on the type of data fusion between IMU and visual data into filtering-based and optimization-based (8).

Loosely-coupled VIO: considers Inertial Odometry and Visual Odometry as two separate entities, each one delivering an independent pose estimation, this results in sub-optimal pose estimations as the correlations amongst the multiple sensor measurements are discarded. These two estimations are post-processed to estimate a unique position and orientation for the mobile platform. The main benefit of this method is its computational efficiency, and the ease to integrate new sensors. The most well-known technique for data sensor fusion is Kalman Filters (KF). Additionally, nonlinear optimizations can be implemented to improve the accuracy and robustness of pose estimation at the expense of computational load, driving them to be infeasible for resource-constrained platforms (i.e. UAV (46)). The main drawback of fusing the two decoupled pose estimations is the information loss, which has an impact on accuracy.

The loosely coupled methods can be classified into two groups depending on how data is processed for prediction and observation when data is fused ().

The first group uses IMU measurements to estimate the states of the kinematic model, while visual data is used as observation data that updates the Kalman filter, such as (46) and (47). This implementation is suitable for accurate high-rate linear velocity estimation, suitable for robotic motions where manoeuvres with variable speeds are performed. The main drawback of this solution is that as it mainly relies on the IMU it is then sensitive to its biases and drift, hence accumulating errors in the pose estimation over time.

The second group uses visual odometry for state estimation, in this case, the IMU data is used as the observation data for the Kalman filter updates. The method is capable of estimating the poses accurately and being drift-free on long runs. On the other hand, as it is not founded on

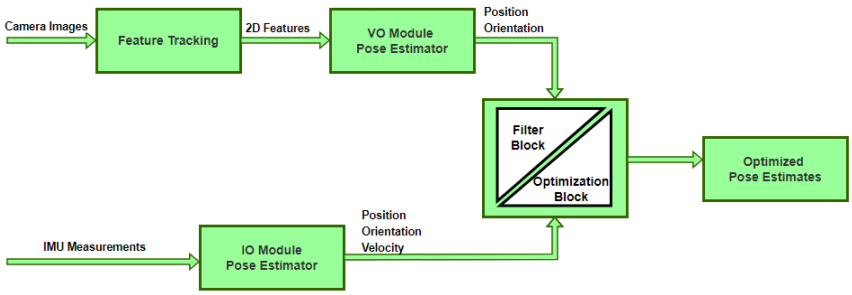


Figure 2.8: Loosly-coupled VIO

the IO, the linear velocity is estimated with lower accuracy. Another drawback of this approach is that the occlusions or high dynamics can cause the VO to fail and it would stop estimating the poses of the mobile platform.

Tightly-coupled VIO : All the inertial and visual sensors are together optimized to produce an accurate state estimation. Applies IMU integration to predict the location of the features on the 2D space to ease feature tracking on consecutive frames. Classically the tightly coupled Extended Kalman Filter(EKF) is the multi-state constraint Kalman filter (MSCKF) presented in (48) or ROVIO (49). The results of this approach showed high accuracy of pose estimation when performed in real-time(RT) and large-scale environments.

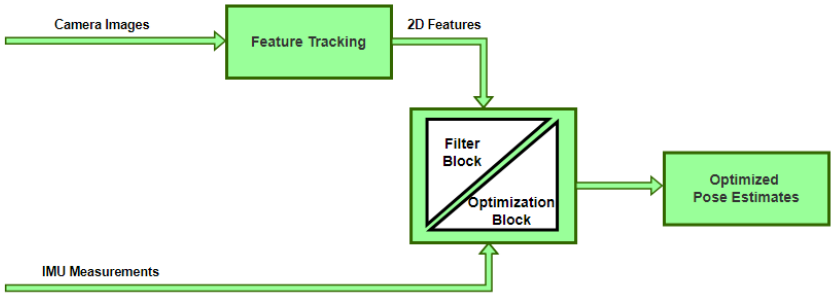


Figure 2.9: Tightly-coupled VIO

Depending on the type of data fusion, the VIO can be classified into:

- Filtering-based: IMU data is used to process the vehicle pose from its dynamic model (process model). This model is later employed to update the state of the vehicle using the information obtained from visual data (measurement model). The filtering approaches can be split into four frameworks: Extended Kalman Filter (EFF), Unscented Kalman Filter (UKF), Multi-State Constraint Kalman Filter (MSCKF) and Particle Filter (PF). The main disadvantages of this approach are that older states are dropped when estimating a new state, causing linearization errors and outliers to be enclosed in the filter state (50). Secondly, linearizations at wrong estimations cause inconsistencies in the unobservable directions when using filters (51) (52). There are four unobservable directions when using VIO: the global position and the orientation around the gravity axis.
- Optimization-based: Computes state estimation from visual data and IMU measurements by minimizing a least-square nonlinear problem (8). This method allows the linearization of multiple points achieving a state estimation with a higher degree of preciseness. While filters add a new state at every IMU measurement, this is infeasible on

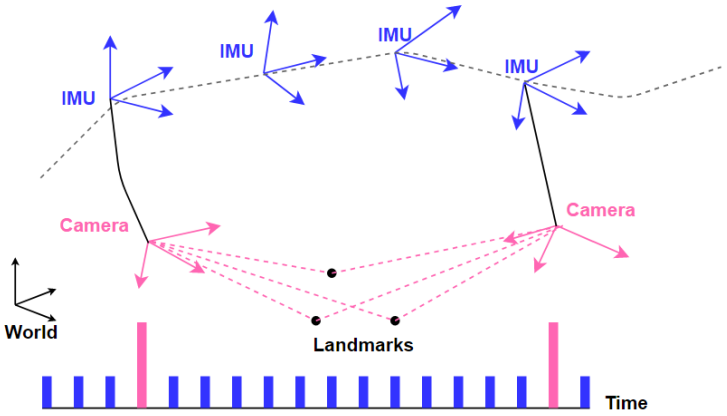


Figure 2.10: Visual description of the frequency in which the IMU and the camera publish the estimated data. The IMU provides information about the MAV position with a higher frequency but lower accuracy. Cameras use visual landmarks to retrieve the position and correct the drift from IMU estimations.

optimization approaches as complexity grows with the dimension of the states. Therefore, the integration of inertial data between frames aids to establish relative motion constraints. This technique requires reintegrating every time the state updates (i.e., after each optimization). A reparametrization of the relative motion constraints avoids the repetition of these integrations, named IMU preintegration. It was proposed by (53) built upon (54) and is widely accepted in VIO to avoid duplicated integrations.

The optimization-based approach has higher precision than filtering-based methods. However, a higher computational load is needed as a larger number of landmarks are considered on the state estimation. If the complete history of landmarks is used for the estimation we will call them Full smoothers, while Fixed-lag smoothers (or sliding window estimators) just consider the last states.

Full smoothers allow re-linearization with each update, while fixed-lag smoothers lock linearization permanently on the marginalized states, this provides less accuracy in exchange for a lower computational load. Nowadays, research focus on Fixed-lag and Full-smoothers as computational power has improved and their higher accuracy.

2.4.2 VIO vs VI-SLAM

VINS algorithms are included as an instance of VI-SLAM and visual-inertial odometry (VIO). While the former uses the features positions and IMU and camera poses together on the state vector, the latter does not include the features in the state vector but uses the features to set motion constraints between the IMU and the camera poses. When performing loop closures VI-SLAM boosts the accuracy of features position at the expense of higher computational complexity. VIO state estimators' error grows unbounded as no information from previous locations is known to constrain them.

2.5 Machine Learning in VINS algorithms

Machine Learning (ML) algorithms have attracted significant attention from the research community. ML approaches benefit from the increasing volumes of data and computational power, applying their approaches to many different domains. Pose estimation is not an exception, where ML allows more precise state estimations and faster data processing. Humans perceive their self-motion and surroundings with multiple perception sensors that allow them to nav-

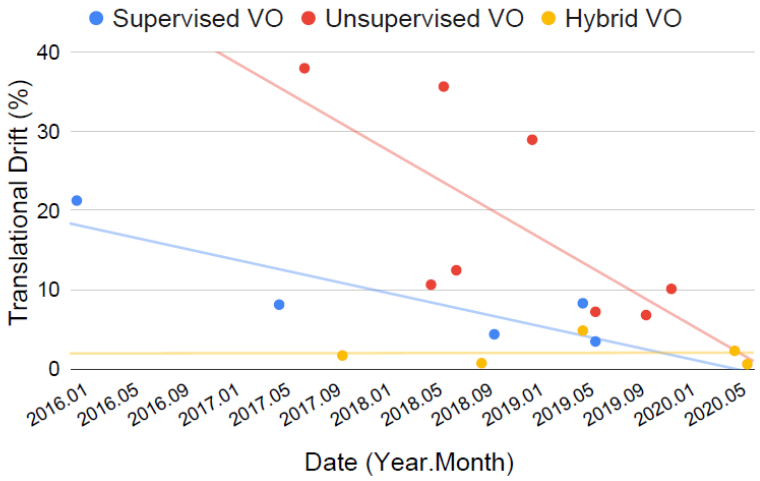


Figure 2.11: Comparison of the Deep learning approaches based on VO on Trajectory 10 of KITTI dataset. (55)

igate in the 3D space. Similarly, neural networks perceive the environment and estimate the system position using multiple sensors (55).

How can Machine Learning improve VINS algorithms?

Data-driven algorithms avoid the human effort of defining all the mathematical and physical rules for the algorithm. The NN is provided with a set of successive images where depth, motion parameters and pose are predicted and trained against a ground truth. ML algorithms not only learn from past experience but also adapt to the changing world by employing new training and including it in their evolving localization algorithm.

One of the main advantages of conventional algorithms is the ability to predict motion without any prior knowledge of the camera parameters. Machine learning algorithms can make use of the increasing amount of sensor data and the growing computational power. They easily scale to large problems, where the parameters inside the model are minimized through optimization. Nevertheless, these algorithms are more computationally costly than conventional models.

2.5.1 Working principals of Machine Learning and VO/VIO

Deep learning algorithms extract high-level features from images providing alternatives to solve VO/VIO predictions. Deep learning algorithms may combine classical VO/VIO algorithms with neural networks (*hybrid VO/VIO*) or be purely based on neural networks (*end-to-end VO/-VIO*). This last group can be *supervised VO/VIO* if there is ground-truth data available during the training phase or *unsupervised VO/VIO* if it is not available.

A typical choice for supervised learning is DeepVO (56) which combines a convolutional neural network with a recurrent neural network. DeepVO estimates the position of a vehicle when driving it in unseen scenarios, outperforming monocular conventional methods such as VISO2(57) or ORB-SLAM (58) when there are no loop closures (57). Furthermore, it can extract the scale even with monocular cameras as the NN deduces the scale metric from the experience of large sets of images.

The research community's interest in exploring efficient unsupervised learning VO algorithms is growing. They make use of unlabelled data in the training phase, saving human effort. This also provides better generalization and adaptation to new scenarios. A typical unsupervised algorithm predicts depth maps and poses to estimate the transformation between the captured

images. The two main problems that prevail unsolved from the original work (59) are the impossibility to retrieve a good scale and the assumption of a static environment without camera occlusions. Today the performance of unsupervised learning is lower than supervised methods as depicted in Figure 2.11, however, the constant evolution of performance and the superior advantages of unsupervised learning make it a promising option for the not too distant future.

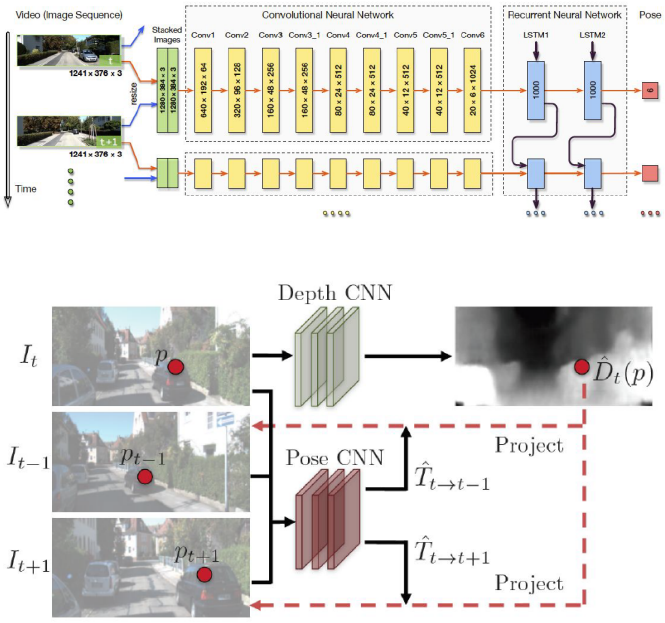


Figure 2.12: Top: Supervised learning framework for VO. Bottom: Unsupervised learning framework for VO.(60)

Hybrid approaches use neural networks but rely on conventional VO frameworks, where deep learning is used to replace part of the mathematical model, normally by estimating the scale. D3VO (61) is one of the best hybrid algorithms which integrated depth, uncertainty and pose directly on visual odometry. The combination of both frameworks makes these algorithms more accurate and robust than end-to-end deep learning approaches. For instance, D3VO defeats several VO/VIO alternatives such as DSO (62) or ORB-SLAM (58).

VIO algorithms encompass visual and inertial data, opening the possibility to train NN with IMU measurements, such as DeepVIO (63), to reconstruct trajectories on a global scale. As mentioned, they are also categorized into hybrid VIO and end-to-end VIO(supervised or unsupervised).

3 VINS algorithms comparison for MAVs

One of the end goals of the project is providing the team with more flexibility to fly our MAVs autonomously in any indoor or outdoor environment. Currently, aerial robots developed at RAM-UT rely mainly on Motion-Capture (MoCap) state feedback for indoor localization. VINS algorithms play a fundamental role in enhancing the level of autonomy of our MAV's, granting pose information employed in localization.

3.1 What is a MAV?

An unmanned aerial vehicle (UAV) is the term used for an aerial vehicle with no human pilot. Over the years, systems and sensors have been continuously improving allowing us to create UAVs that fit in the palm of our hands.

MAV is a term used to describe a fairly small UAV. Commonly, UAVs come with flapping wings, fixed wings or rotatory wings. The research community have been attracted by the wide variety of applications, and continuously expanding list of new algorithms developed for them. MAVs open a huge number of possibilities in research, where research is conducted in multiple types of environments (indoor and outdoor, know and unknown, obstacle-dense and obstacle-free), and involving challenging tasks such as physical interactions.

3.2 System requirements for the selecting the VINS algorithm

The final ambition of the project is the integration of aerial cognitive robotic systems, providing them with unprecedented abilities to interact with humans safely. In this section, it is explored the most used opensource VINS solutions to accurately estimate MAV poses, with the final goal of making feasible co-work between humans and drones. Requirements for choosing the most convenient VINS algorithm:

1. Accuracy will be the highest priority to preserve human safety and accomplish the tasks successfully.
2. The VINS algorithm will run in a MAV which has hardware limitations due to the reduced size of these devices.
3. MAVs will be working with humans for an indeterminate time, we will look for optimal algorithms for lifelong experiments
4. Providing a final map of the surrounding environment is not critical but it is a plus.
5. Ultimately, the MAV should not only run indoors, where most of the algorithms are normally tested but also outdoors where the drone is exposed to more challenging scenarios such as more critical illumination changes or longer distance to objects.
6. Strong community using the algorithm. We believe there is a correlation between the high frequency of appearance on papers of the algorithms and a strength of the community supporting and using this algorithm.

3.3 Open Source VO/VIO Comparison

In the past few decades, VIO has received much attention for efficiently estimating the ego-motion of unmanned aerial vehicle systems. Numerous researches on VIO and the multiple possibilities that camera sensors offer to interact with humans (e.g., gesture recognition) and the surrounding environment (e.g., object detection, semantic recognition) motivate us to exploit the alternatives that include camera sensors.

In this section, it is pictured the main open-source VINS algorithms used in research. The analysis just includes the most recent algorithms, the ones that guarantee the highest accuracy and most used. In this analysis, the papers selected were those which compare multiple algorithms at the same to be sure the tested algorithms were graded under the same conditions.

The evaluated algorithms only include conventional approaches as machine learning implementations were either not available for the general public or not as robust as conventional approaches, therefore, these methods have been discarded. A descriptive table has been included in Figure A.2 for reference. As mentioned on (55), VIO state-of-the-art classical models still outperform learning-based VIO and require less computational resources than ML approaches, however, the latter is more robust to real time issues such as noise. ML approaches should not be forgotten, as their potential is very high and have a rapid rate of progress, see Figure 2.11. The fast development on power capabilities of onboard devices points make them a promising alternative in the not too distant future. Some of them already outperform several popular VO/VIO systems such as DSO (62) or VINS-Mono (64).

In Table 3.1 it is summarized the most relevant algorithms extracted from the analysis. It concludes that the typical cameras used in localization algorithms are either monocular, stereo and RGB-d. Nonetheless, most of the novel algorithms are exploring other types of cameras mentioned in 2.2, such as (65) which uses event cameras or (66) dependent on thermal cameras.

Event cameras would be relevant in high dynamic scenarios and thermal cameras are oriented to perform in low visibility capturing temperature information from the environment. Neither of the described scenarios applies to our case, therefore we to sticked to monocular, stereo and RGB-d algorithms which have been tested more extensively. Lastly, fisheye cameras could have been considered as an option, however, almost none of the recent algorithms included the possibilities to use these cameras, this is due to the higher distortions that they caused on images and the higher computational cost as a result of their curved epipolar line.

Algorithms	Hardware requirements	Approach	Input Treatment	Localis./Mapping	Loop Closures	Map	Comparison Papers	Comment
ORB-SLAM (58)	Monocular	Optim.	Indirect	2D-2D/3D-2D	Yes	Sparse	(67) (68) (13) (69) (70) (71)	Robust for large path tracking
ORB-SLAM2 (72)	Stereo/RGB-D	Optim.	Indirect	2D-2D/3D-2D	Yes	Sparse	(67) (68) (73) (69) (74) (70) (71)	Robust for large path tracking
RTAB-MAP (75)	Stereo/RGB-D	Filter	Direct	–	Yes	Dense	(67) (70) (71)	Scene reconstruction
Mono-SLAM (76)	Monocular	Filter	Indirect	3D-2D	No	Sparse	(68) (13)	Pose estimation in robotics
PTAM (77)	Monocular	Optim.	Indirect	3D-2D	No	Sparse	(68) (13) (71)	A.R. in small workspace
SVO (78)	Monocular/Stereo + IMU	Optim.	Direct/Indirect	2D-2D/3D-2D	No	Sparse	(68) (13) (69) (79)	Fast, consistent semidirect method
LSD-SLAM (80)	Monocular/Stereo	Optim.	Direct	2D-2D	Yes	Semi-Dense	(68) (13) (73) (69) (74)	Semidense trajectory estimation
VINS-Mono (64)	Monocular + IMU	Filter	Indirect	3D-2D	Yes	Sparse	(13) (69) (81) (79)	Full VI-SLAM method
DSO (62)	Monocular/Stereo	Optim.	Direct	2D-2D	No	Sparse	(13) (13) (73) (73) (69) (71)	Direct and sparse VO method
ROVIO (49)	Monocular/Stereo + IMU	Filter	Direct/Indirect	2D-2D	No	None	(13) (69) (81) (79)	Robust VIO for UAV
PRO-SLAM (82)	Stereo	Optim.	Indirect	-	No	Sparse	(73)	-
OpenVSLAM (83)	Monocular/Stereo/RGB-D	Optim.	Indirect	3D-2D	Yes	Sparse	(73)	-
MSCKF (48)	Monocular/Stereo	Filter	Indirect	3D-2D	No	None	(13) (69) (81) (79)	-
OKVIS (84)	Stereo	Optim.	Indirect	2D-2D/3D-2D	No	Sparse	(13) (81) (79)	Robust stereo VIO for UAV
ElasticFusion (85)	Monocular/RGB-D	Optim.	Direct	2D-2D	No	Dense	(13) (74)	Map-centrinc vSLAM
KineticFusion (86)	RGB-D	Optim.	Direct	3D-2D	No	Dense	(13) (74)	3D modelling with the Kinetic

Table 3.1: Comparative classification of main VO/VIO and V-SLAM/VI-SLAM methods.

Weights	100	5	2	1	2	1	1	2	3	1	1	1	
Algorithms	**Available**	**Accuracy**	**Low Cost Hardware**	**Memory Usage**	**Computational Cost**	**Type of Map**	**Frequency of Usage**	**Robustness**	**Outdoor**	**Low Texture**	**Ilumination**	**Loop Closure**	**Total Score**
ORB-SLAM	2	1	1	-1	1	1	2	-1	1	-1	1	1	213
ORB-SLAM2	2	2	1	1	0	1	2	2	2	0	1	1	228
RTAB-MAP	2	1	-0.5	-0.5	-0.5	2	0	0	-0.5	-0.5	-0.5	1	203
Mono-SLAM	2	-2	-0.5	-0.5	-0.5	1	0	-0.5	-0.5	-1	-1	1	185
PTAM	2	-1	-0.5	-1	-0.5	1	-0.5	-0.5	-1	-1	-1	-1	185.5
SVO	2	1	2	-0.5	2	1	2	1	-0.5	1	-0.5	-1	215.5
LSD-SLAM	2	-2	1	-1	1	2	2	0	1	-1	-0.5	1	199.5
VINS-Mono	2	1	1	-0.5	-1	1	2	1	1	-0.5	-1	1	212
DSO	2	1	-1	-1	-1	1	2	1	1	0	-0.5	1	208.5
ROVIO	2	1	1	1	1	-1	2	1	-1	-2	-0.5	-1	206.5
PRO-SLAM	2	-1	1	1	2	1	0	-0.5	-0.5	-0.5	-0.5	1	200.5
OpenVSLAM	2	-1	1	1	2	-0.5	0	-0.5	-0.5	-0.5	-0.5	-1	197
MSCKF	2	-1	2	2	2	-1	1	1	-1	-1	-1	-1	201
OKVIS	2	1	1	-0.5	-0.5	1	1	1	1	-1	1	-1	211.5
ElasticFusion	2	1	1	-1	1	2	1	-1	-1	1	-0.5	-1	205.5
KineticFusion	2	0	1	1	1	2	1	-0.5	-0.5	-0.5	0	-1	204

Table 3.2: VIO algorithm's evaluation against the properties which were considered relevant for the project.
Legend: Very Bad = -2, Bad = -1, No Data = -0.5, Decent = 0, Good = 1, Very Good = 2.
Type of Map: Dense/Semi-Dense = 2, Sparse = 1, No Map = -1.

3.3.1 Top 5 algorithms analysis and selection.

A numerical evaluation was given to each of the algorithms, taking into account the performance and conclusions reached on the numerous comparison papers presented on Table 3.1. Each of the algorithms presented here was tested against different environments (underwater, aerial vehicles, handheld, UGV ...) and, with different hardware configurations and power. The numerical grading is presented on Table 3.2. Different properties have been deduced and weighted in terms of the requirements and their impact on the final goal of the project. Based on the scoring obtained the extracted conclusions regarding top 5 algorithms are presented bellow:

- **ORB-SLAM**: monocular VO algorithm, it tracks features applying an optimization-based approach. Results highly improve when there are several loop closures, otherwise it loses track much more easily. Detecting a low number of features causes inaccuracies in the robot trajectory estimation and the projection of mapped features, leading to a loss in localization. Abrupt rotations have a negative impact on the estimations. Needs high memory compared to implementations without loop closure detection as it saves features on a bag of words to identify previously seen features, however, it does not have the best performance for long-life experiments where it was not able to track the complete trajectory. Experiences slow initialization in some cases. ORB-SLAM is a feature-based method that builds sparse point cloud maps. Therefore, there are difficulties to recover 3D scenes but performs good enough for navigation. Monocular V-SLAM systems suffer from scale drift, which could be solved with an additional recovery module.

- **ORB-SLAM2**: stereo/RGB-d version of ORB-SLAM algorithm. Very satisfying results, performing nicely in both outdoor and indoor environments and outperforming in several experiments other frameworks in translation estimations. As it is an indirect method it reinforces translation estimation by tracking valuable features, on the other hand, direct methods showed higher accuracy against rotation. Increasing movement speed has an inverse correlation with the number of stable features detected, hence, giving poorer results. It takes even 3 times longer to process each frame compare to non-optimization based algorithms. There is always a tradeoff between accuracy and efficiency. ORB-SLAM2 performs a pruning process of the key-frames to reduce memory consumption. For instance, in (74) the experiments start at 200MB and it only grows up to 240MB. It shows to be a computational effective algorithm, in spite of not using a GPU it only performs twice slower than KineticFusion. As ORB-SLAM2 is the stereo/RGB-d version of ORB-SLAM it does not suffer from the same scaling problems. The performance between stereo and RGB-d versions is very similar in indoor scenarios but stereo presents more robust and accurate results outdoors. Last but not least, ORB-SLAM2 handles challenges such as scale changes, pedestrian motion or glass reflection. ORB-SLAM2 is considered as one of the first options for its user-friendliness (i.e. easy initialization and set up), computational power and hardware requirements, and global accuracy and robustness.

- **SVO:** VO odometry SVO implementation has not the best accuracy. When only depending on stereo cameras it was subject to rotation errors as it is mainly an indirect method that detects corners and tracks their intensities, therefore having slightly better results in dealing with rotation than other indirect methods. SVO tracks feature effectively, but when a few of them are available it might lose track. It is one of the algorithms which provides the highest computational efficiency preserving a good level of accuracy for flying robots applications. Loop closures are not possible with this method, therefore, it accumulates drift over time. When adding an IMU it seems to work very well in both monocular and stereo versions. It worked well under large rotations, improving thanks to the IMU, and under high-intensity gradients becoming one of the best algorithms in

performance. This method provides good results both indoor and outdoor, being able to deal with low texture environments and providing robust tracking of movement. SVO has been evaluated taking into account the VIO version which provides competitive results.

- **VINS-Mono** : full VI-SLAM method, it tracks features applying a filtering-based approach. Uses loop closure more intensively as the error grows, the IMU preintegration helps to keep the error bound. Offers a very realistic scale estimation when the algorithm is properly initialized, otherwise it may run into inaccuracies as it only uses one camera. ORB-SLAM2 comes first when comparing easiness of being implemented and as it is a more user-friendly algorithm. Under good illumination conditions, it provides a satisfactory performance and is one of the few packages that does not look track underwater, however, when not enough features are detected it diverges easily. In terms of accuracy is one of the best among the other analysed algorithms, but also has slightly higher memory usage in comparison with other methods. VINS-mono was revealed as one of the tops in robustness across multiple platforms and sequences, nicely peformin in both outdoor and indoor environment and large environments. Its superior performance comes at the cost of a high level of resource usage which can turn out to be prohibitive in resource constraint hardware. Especially effective when loop closure was enabled, although this was not always a possibility in platforms with limited resources.

- **OKVIS** : robust VIO implementation with no loop closure, which uses an optimization-based approach to track features. OKVIS is an optimization method, having more potential than filtering ones, such as VINS-Mono, in terms of memory management and accuracy in exchange of more using more computational resources. It performs accurately across platforms with different power capabilities, despite its low update rates as a consequence of the long processing time per frame. Good feature detection and tracking despite low contrast images or low illumination. The stereo implementation has advantages in terms of memory management.

Upon this overview of the best conventional localization algorithms, it is time to select the ones that we considered optimal for our use-case. By using the final scores it was clear for us that ORB-SLAM2 had to be tested as it seemed to be the clear winner after our research. Furthermore, by the time the analysis had concluded we discovered the VIO version of ORB-SLAM2 (called ORB-SLAM3) had been released in the past months and seemed worthy to test its capabilities.
Furthermore, by the time the analysis concluded the improved version of SVO was made open-source. This new version was named SVO Pro and included several improvements mounted on top of SVO. It allowed loop closures performing local and global bundle adjustment that optimizes global poses and the map in real-time. SVO and SVO Pro were both developed by the Robotic and Perception group from Zurich. This team fosues on drone development and the results on SVO were already promising, therefore, we considered SVO Pro to have a big potential to be an effective localization method, hence we decide to test it as well.

3.3.2 State of the art of the selected algorithms

In this section, we briefly introduce the three selected algorithms, which allows our system to retrieve the position and orientation data from the IMU measurements and visual landmarks. As suggested by original papers of the presented algorithms, we will use the default configuration as it was tuned to generalize for most of the environment. Also with this configuration we would be able to achieve a fair comparison against the original papers.

ORB-SLAM2

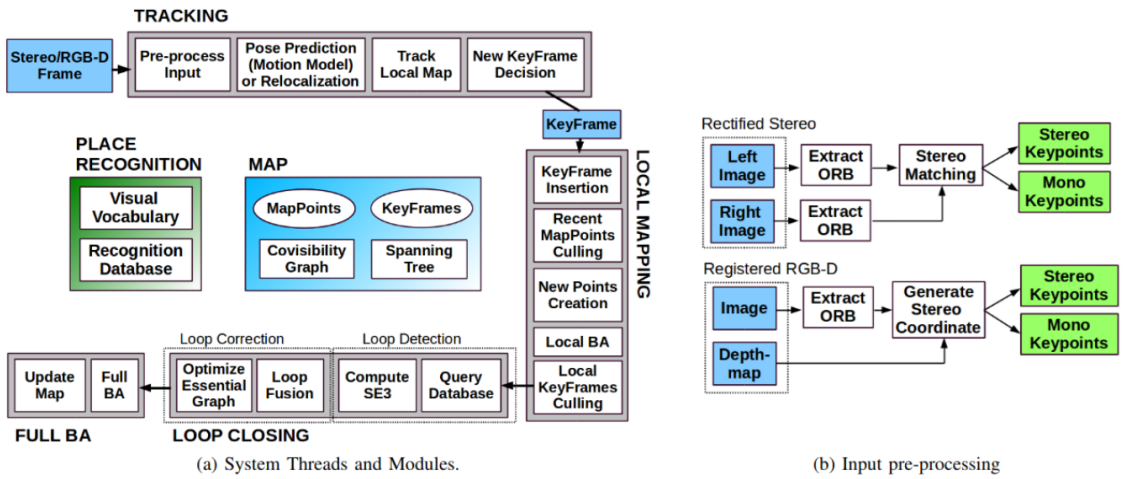
As its name suggests, it uses ORB (Oriented FAST and Rotated BRIEF) features (87), it is built on FAST keypoint detector and the BRIEF descriptor which are attractive for their good precision and low computational cost allowing RT. These features are invariant to rotation and scale and are applied for tracking, mapping and place recognition. ORB-SLAM2 (88) runs 3 threads in parallel:

- **Tacking thread:** finds the matching features between the image frame and the local map. The detected keypoints are detected as close (retrieve scale, translation and rotation) or far (retrieve accurate rotation) thanks to depth information, an improvemnt in comparison to (58). Then the reprojection error is minimized to estimate the camera motion in each frame.
  The map used for pose estimation has four components: 1) The keyframe stores the features extracted from the frame, the camera pose and its intrinsic parameters. 2) Map points store all the infromation about to the points, 3D positions, viewing direction, the descriptor of the points and the distance at which they are detected. 3) Covilibility graph link keyframes that share at least 15 matched points. This covisibility graph provies the key frame and map points that are used for computing the local trajectory and optimizing the map. 4) The spinning tree is the root of the covisibility graph as it contains a minimal number of connections and is essential for fast identification of map points and keyframes.

- **Local mapping thread:** performs bundle adjustment to a local window of keyframes and the points in the local map.

- **Loop closure thread:** every time a new loop is detected it performs a pose-graph optimization to reduce the accumulated drift. This last thread activates a fourth thread to execute a full BA optimizing all keyframes and map points.

There is a place-recognition module that employs a bag of words call DBoW2 (89) which has the purpose of detecting loops, relocalizing the system when it lost track of the position or reinitializing in the mapped environment. A major improvement compared to ORBSLAM is the quickest initialization as the scale is obsevable.



(a) System Threads and Modules. (b) Input pre-processing

Figure 3.1: ORBSLAM2 pipeline overview with its 3 mian threads: tracking, local mapping and loop closing.(88)

ORB-SLAM3

ORB-SLAM3 (90) is an improvement of ORB-SLAM2 where IMU measurements are tightly-coupled with visual data resulting in a VIO method that fully relies on MAP estimation, which corresponds to BA methods that minimizes feature reprojection error. Other major improvements are:

- Novel place recognition method that allows short, mid and long term associations improving map accuracy at the expense of negligible computational cost.
- Atlas (91): novel multiple map system that initializes a new map when it gets lost until it revisits mapped areas, at this time both maps get merged.
- Abstract camera representation that makes the SLAM algorithm independent from the camera model, this allowed the implementation of pinhole and fisheye camera. With this implementation, we do not encounter the typical problem in VIO methods which consider uniform reprojection error. This forces to crop the peripherical area of the fisheye camera, losing the advantages of large FOV.
- Fast initialization method that uses both visual and inertial uncertainties to estimate the scale with 5% error in 2 seconds and drops down to 1% after the first 15 seconds.

SVO Pro

SVO Pro is a VIO algorithm that results from a set of enhancements on top of the SVO (78) localization approach. Consequently, we will start by understanding SVO implementation. SVO is a monocular VO localization algorithm blend of direct and indirect methods. It emerged as a possibility to used dense structures and motion in RT. SVO is known for being a fast and versatile visual front-end,it has two parallel threads:

- **Motion-estimation thread:** an approach for motion estimation that consists of 3 steps: 1) Sparse alignment, estimates camera motion by the minimization of the difference between pixel intensities of matching 3D points, known as photometric error. It is considered a semi-dense approach as it only exploits pixels with high gradients to perform the alignment, these are features in corners and edges. It creates patches around the mentioned features and approximates them to the same depth. Eventually, the photometric error is minimized using those features for which depth is known. 2) Relaxation, minimizes drift in motion estimation by aligning the patches of the new frame to the frame in which they were first detected. 3) Refinement, to achieve optimal camera poses and 3D map points it performs a BA to minimize the projection error
- **Mapping thread:** depth filter which initializes for each feature whenever a new keyframe is selected. Through a Bayesian update using template matching, the initial depth uncertainty is reduced. When uncertainty is under a certain threshold a new point is added to the 3D map and used for motion estimation.

SVO Pro is the newest version of SVO and includes the following extensions:

- Supports not only perspective cameras but also fisheye and catadioptric (92) cameras in monocular and stereo set up, including active exposure control (93) for all cameras.
- Improvement to build an optimization-based VIO algorithm. SVO is used in the front-end and a tightly-coupled approach (94) is used for optimizations. In the back-end, it uses local BA (modification from (84)) and global bundle adjustment can be activated to detect loop closures using the bag of words DBoW2 (89).

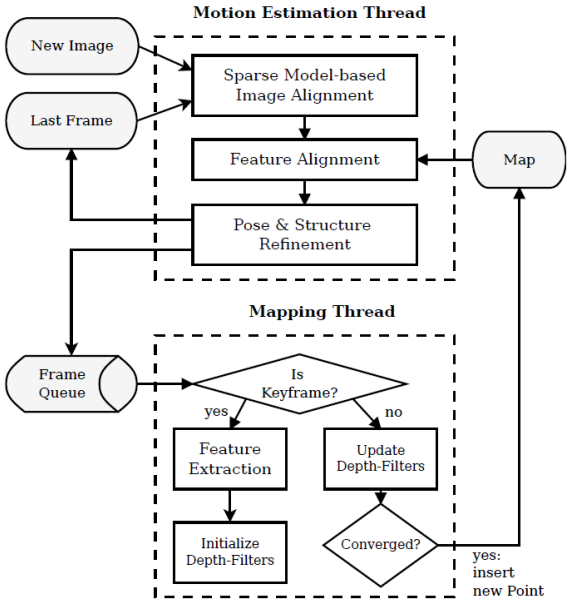


Figure 3.2: Motion and Mapping threads from SVO framework. (78)

-

4 System Design and Evaluation Tools

This chapter describes the selected hardware to fly the MAV to build our custom datasets and the hardware used in the EuRoC dataset, which was specifically recorded with an IMU and a pair of stereo cameras assembled on top of a MAV. The hardware calibration is described in detail and the tools required to evaluate the VINS algorithms. It also contains a description of the software architecture, showing how the hardware components communicate with the ROS nodes to produce the estimated poses and the map point-cloud.

4.1 Hardware

The experiments were performed using the FiberTHex drone developed by the RAM group at UT. FiberTHex is composed of 6 propellers with an angle of 60 degrees between them, Figure 4.1. The hardware components of the drone are shown in Table 4.1. The onboard computer of the FiberTHex runs on Ubuntu 18.04.6, it will be in charge of the control and the navigation of the drone while recording our custom datasets. FiberTHex reads the pose measurements from the Optitrack and uses this information to provide the correct input to the propellers.

Figure 4.1: FiberTHex drone with 6 propellers and the camera D435i mounted between propeller 6 and 1.

The datasets will be recorded using an HP Pavilion, its hardware components are described in Table 4.2. The computer will activate the camera and record the camera images and IMU measurements that were used to estimate the positions of the drone using VI-SLAM algorithms.

As mentioned earlier, the chosen technology to perform localization is complementing camera images with IMU measurements, allowing us to use VI-SLAM. The chosen hardware was the Intel RealSense camera D435i. This camera is provided with two infrared global shutter cameras, an RGB camera, a laser to improve depth estimation in scenes with low texture and an Inertial Measurement Unit (IMU) for 6 degrees of freedom data. The camera properties can

FiberTHex Properties	
MASS	2.2kg (minimual setup)
GPS	Sirius
Flight Controller	Mikrokopter f.c. V2, Paparazzi Chimera
Onboard computer	Intel NUC, Odrioid XU4
Motor	MK 3638
Propellers	Up to 13”
Battery	3500mAh 4S1P 14.8V 45C/90C

Table 4.1: FiberTHex Properties description

HP Pavilion	
Memory	15.5 GiB
Processor	Intel Core i7-7700HQ CPU @ 2.80GHz × 8
Graphics	Intel® HD Graphics 630 (KBL GT2)
SSD	141.5GB

Table 4.2: HP Pavilion properties

be seen in the Table 4.3. This camera comes with a ROS Wrapper that publishes the camera data on multiple ROS topics such as the camera images from the stereo and RGB cameras, the inertia measurements or the depth estimations provided from the image disparity of the stereo cameras. Additionally, the onboard processor is also capable of providing the rectified images and publishing them as a ROS topic so no rectification is required from our side.

Figure 4.2: Intel RealSense camera D435i, starting from the left we have the right infrared camera, the Infrared projector, the left infrared camera and the RGB module.

4.1.1 Intel RealSense Calibration

IMU Intrinsic parameters: The angular velocity $\widetilde{w}$ and linear acceleration $\widetilde{a}$ measured by the IMU are affected by noise (η^g, η^a) and bias (b^g, b^a):

$$\widetilde{w} = w + \eta^g + b^g \tag{4.1}$$

$$\widetilde{a} = a + \eta^a + b^a \tag{4.2}$$

being **w** and **a** the true angular velocity and acceleration at the body reference frame. The noise is assumed to have Gaussian distribution:

$$\eta^g \sim N(0, \sigma_g^2 I_3) \tag{4.3}$$

$$\eta^a \sim N(0, \sigma_a^2 I_3) \tag{4.4}$$

Intel Realsense Camera: D435i	
Image Senosory Technology	Global Shutter
Ideal Range	0.3m to 3m
Depth Techonology	Stereocopic
Minimum Depth distance at Max Resolution	~28cm
Depth Accuracy	<2% at 2m
Depth Field of View	87º x 58º
Depth Stream Output Resolution	Up to 1280x720
Depth Stream Output Frame Rate	Up to 90 fps
RGB Frame Resolution	1920x1080
RGB Frame rate	30 fps
RGB Sensor Techinolgy	Rolling Shutter
RGB Sensor FOV	69ºx42º
RGB Sensor Technology	Rolling Shutter
RGB Sensor FOV	69ºx42º
Camera Module	Intel RealSense Module D430 + RGB Camera
Vision Processor Board	Intel RealSense Vision Processor D4
Form factor	Camera Peripheral

Table 4.3: Intel RealSense D435i camera properties.

where σ_g^2 and σ_a are the noise densities provided by the IMU data-sheet. When working on discrete-time the IMU measurements are integrated. The resulting noise densities ($\sigma_{a,f}$) will be dependent on the IMU frequency f, also provided in the specification sheet.

$$\sigma_{g,f} = \frac{\sigma_g}{\sqrt{f}} \tag{4.5}$$

$$\sigma_{a,f} = \frac{\sigma_a}{\sqrt{f}} \tag{4.6}$$

Brownian motion is used to model the slow variations from sensor biases. Considering two consecutive instants of time:

$$b_{i+1}^g = b_i^g + \eta_{rw}^g \quad \text{with } \eta_{rw}^g \sim N(0, \sigma_{g,rw}^2 I_3) \tag{4.7}$$

$$b_{i+1}^a = b_i^a + \eta_{rw}^a \quad \text{with } \eta_{rw}^a \sim N(0, \sigma_{a,rw}^2 I_3) \tag{4.8}$$

where $\sigma_{g,rw}$ and $\sigma_{a,rw}$ are the random walk standard deviation which should also be given by the manufacturer.

IMU Calibration

The IMU is calibrated to keep the drone stable when flying. This calibration is straightforward by running the python code provided in the RealSense SDK. Following the procedure described on the SDK, where the camera is left steady on a flat surface for a long period and applying multiple orientations. With this approach, the algorithm estimates the biases random walk, noise densities and sensitivity of the calibrated IMU. The calibration is considered to be successful when the accelerometer norm is close to the expected value of 9.8, fitting the earth's gravity acceleration. On our calibration the gravity vector components were $[0.11278, -9.7918, -0.52477][m/s^2]$, and the module of the gravity vector was 9,8057 so the calibration was considered successful.

Finally, the homogeneous transformation from the left camera frame to the IMU frame (considered to be the camera body frame) is estimated following the procedure in (95). Although this metric is previously given by the manufacturer, the transformation proposed was inaccurate thus a further calibration was performed. This method consists of exciting the accelerometer and the gyroscope along their axes, making sure the camera always points to the pattern used for the camera calibration. For this purpose, the tool *Kalibr* was developed by the Robotics and Perception Group at ETH and the University of Zurich.

Camera Calibration - IntelRealsense
D435i camera sensors are built to be sturdy to maintain factory calibration and performance over their lifetime. Nevertheless, extreme vibrations or temperatures can cause degradation over time. IntelRealsense SDK provides an on-chip calibration tool in case of degradation, however, it will not be used in our case as the camera was new so little degradation was expected.

Camera Calibration - Kalibr
SVO Pro algorithm always rectifies the images given as an input, therefore, the calibration values for the stereo cameras were needed for the algorithm configuration. Kalibr tool was chosen for this aim as it is extensively used for camera calibration in the research community.

The camera calibration solves the intrinsic parameters and the relative camera transformation for the stereo cameras. The intrinsic parameters of a pinhole camera include the focal length (f_u and f_v), the optical center(p_u and p_v), also known as principal point.

Ideal cameras do not have any lens, therefore the camera matrix does not take into account the distortion that the lens cause. The accurate representation of a camera model includes radial and tangential distortion caused by lenses.

- Radial distortion: the rays of light bend more when they are closer to the edged than at the optical center. The smaller the lens the bigger the distortion.

$$x_{distorted} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6)$$
$$y_{distorted} = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k3 - \cdot r^6)$$

  where x,y are the undistorted pixel locations. Computed from the pixel coordinates by applying optical centre translation and dividing by the focal length in pixels. The variable r represents $r^2 = x^2 + y^2$, and k_1, k_2 and k_3 are the radial distortion coefficient of the lens.

- Tangential distortion: it is cause because the image plane and the lens are not completely parallel.

$$x_{distorted} = x + \left[2 \cdot_1 \cdot x \cdot y + p_2(r^2 + 2 \cdot x^2)\right]$$
$$y_{distorted} = y + \left[2 \cdot_2 \cdot x \cdot y + p_1(r^2 + 2 \cdot x^2)\right]$$

  where p_1 and p_2 are the tangential distortion coefficient of the lens.

To estimate the camera parameters using *Kalibr* the ArilTag pattern is used. Images of this pattern are taken from multiple points of view. With the captured images the tool estimated the parameters for our D435i camera which are depicted in Table 4.4 and Table 4.5.

4.1.2 Motion Capture System

Currently, the poses of the MAV are captured within a room thanks to a Motion Capture (MoCap) system. This system captures the 3D position of the MAV using an optical system, it uses

Camera		Left	Right
Focal Length	f_x	401.363	393.395
	f_y	400.85	394.86
Principal point	p_u	413.16	418.83
	p_v	234.13	223.98
Distortion Coefficients	k_1	−0.1041	−0.092
	k_2	0.0152	0.0073
	r_1	0.0002	−0.004
	r_2	−0.015	−0.014
Translation (mm)	t_x	−31.0	26.5
	t_y	12.7	−11.9
	t_z	97.2	98.6
Rotational Matrix		$\begin{pmatrix} 0.9988 & 0.00762 & 0.0463 \\ -0.00736 & 0.9999 & -0.0058 \\ -0.0463 & -0.00586 & 0.9989 \end{pmatrix}$	$\begin{pmatrix} 0.998 & -0.0073 & -0.0463 \\ 0.0076 & 0.9999 & 0.005518 \\ 0.0463 & -0.00586 & 0.9989 \end{pmatrix}$
TimeShift (ms)		0.03112	0.01885

Note that the relative transformations refer the camera frames to the IMU frame. For camera D435i it is use pinhole as camera model and radial-tangential as distortion model.

Table 4.4: Camera Calibration-IMU calibration Parameters

IMU	Accelerometer	Gyroscope
Noise density	0.00186	0.000187
Noise density (discrete)	0.0263043	0.00264457
Random Walk	0.000433	2.66e-5
Update Rate	200Hz	200Hz

Table 4.5: IMU Configuration Parameters

image sensors to triangulate the position of reflective markers. When three or more markers are used to define an object it is possible to determine not only its position but also its relative orientation. Motion Capture systems can run with very high frequency and precision. For instance, our room uses Flex 3 cameras from OptiTrack which provide an accuracy of +/- 0.5mm with a frame rate of 100Hz. While these systems provide superior positioning information, they restrict the MAV to fly within a limited space. Therefore, one of the goals of the project is substituting this system for a VINS algorithm, able to provide the localization data without depending on the MoCap system. This solution implies a sacrifice in precision for the benefit of running the MAV in any environment.

The MoCap system has very high precision, hence it will be used as ground truth to compare the trajectory traced by the MoCap and the one estimated by the VINS algorithms, which provide a poorer performance.

MoCap also requires calibration to guarantee high performance, in our case we used the calibration procedure proposed by OptiTrack on its documentation page.

[1]

[1]Procedure followed to calibrate the Optitrack: https://v22.wiki.optitrack.com/index.php?title=Calibration#Wanding

4.2 Overall Framework

This work tests different V-SLAM and VI-SLAM alternatives using various datasets. The localization algorithm must provide an optimal estimation of position in real-time. ROS environment is a powerful tool that will aid us to connect the different modules needed for localization. It easily establishes communication between hardware components, image processing tools, SLAM libraries and visualization systems to picture the 3D point-clouds and camera poses. Each component can be represented by a node, each node publishes data to topics (channels for messages of the same format) and subscribes to topics to receive inputs. The communication within ROS is done through ROS messages, a master node is in charge of coordinating the exchange of messages between nodes and topics.

The framework used in this thesis is described in Figure 4.3. Our framework subscribes to the camera topics, depth topic and IMU measurements (depending on the type of SLAM algorithm that we want to run). The SLAM node processes the input data to predict the localization of the system at each moment and publishes the pose and point cloud data. Lastly, the visualization node outputs graphically this data.

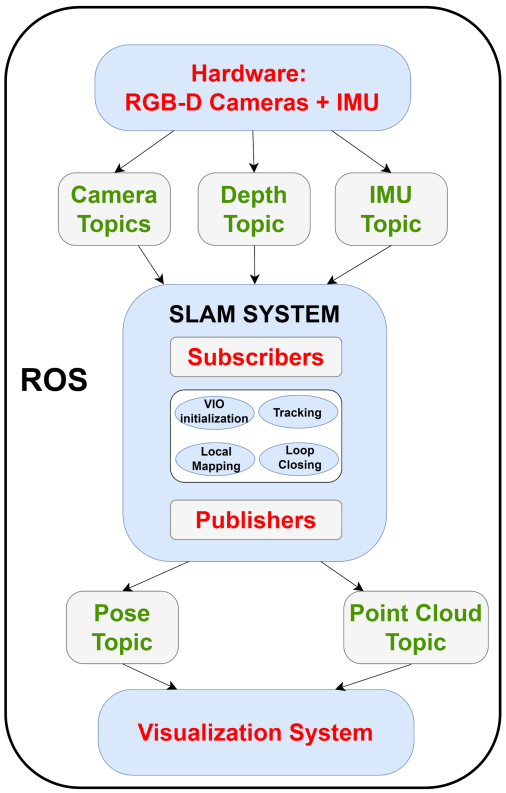


Figure 4.3: Communication between hardware, SLAM and the visualization system within ROS environment.

4.2.1 ROS Positioning Frames

In VIO systems, poses are expressed in computer vision as transformations from the camera frame to the world coordinate system, represented as T_{wc}. In this context, P_w is assumed to be the position of the camera in the world frame and P_c is the camera position in its world frame (equivalent to the identity matrix).

In the ROS framework, the camera frame is known as *base_link* as mobile robots can be composed of multiple sensors. Therefore, this frame represents the position of the moving platform

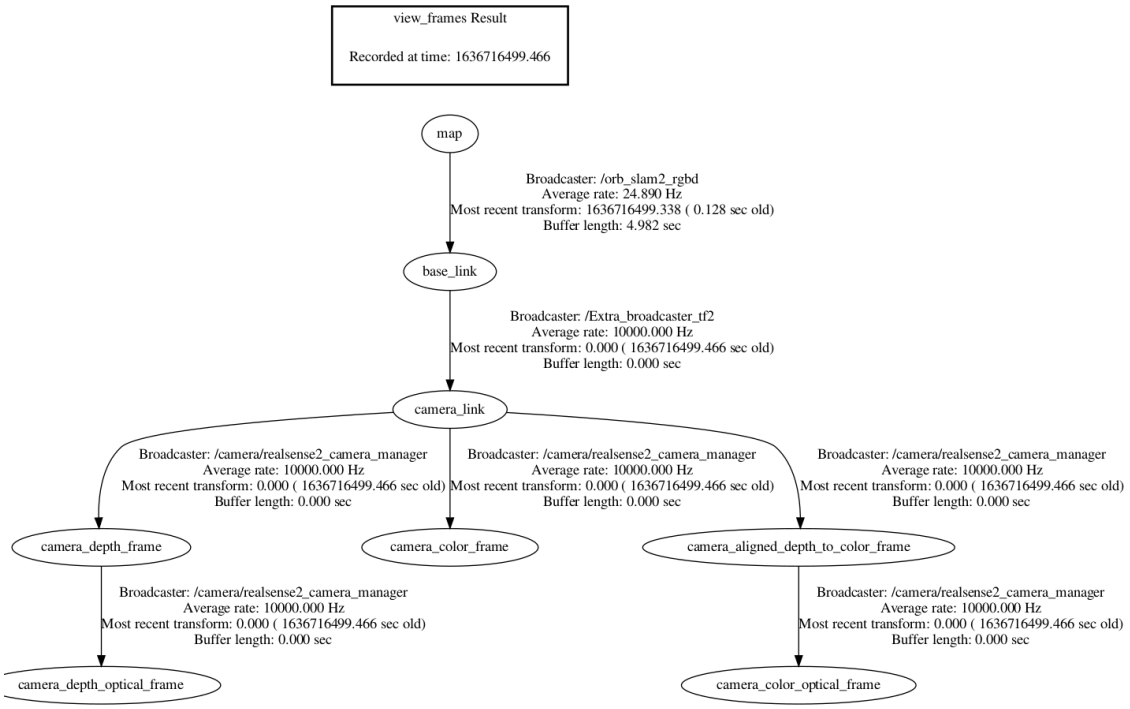


Figure 4.4: Schematic of ROS frames of ORB-SLAM2 RGB-d.

P_{base}. The *base_link* may have a different position and orientation depending on the hardware platform used.

The ROS frame *odom* is world-fixed, used as a reference by the *base_link* to represent the position of the robot in the world frame, P_{odom}. This frame is normally initialized to match with the initial *base_link* axis estimated. The robot poses may change and drift over time without any restrictions. As an effect of this drift, the *odom* frame is unreliable as a global reference frame for life-long localization. The benefit of the *odom* frame is that robot poses are continuous without making discrete jumps. To represent the robot position in the world frame it is needed the transformation matrix T_{base}^{odom}. Thus,

$$P_w = T_c^w \cdot P_c \quad \text{(Comp. Vis.)} \quad \Rightarrow \quad P_{odom} = T_{base}^{odom} \cdot P_{base} \quad \text{(ROS)}$$

Odom frame is normally initialized with the first

The *Map* frame is also fixed to the world with the z-axis pointing upwards. Contrary to *Odom*, its position should not drift over time. *Map* frame constantly recomputes robot poses eliminating drift using sensor data, though, this causes discrete jumps on the estimations. While *Odom* frame was useful as a shot-term the *Map* frame is used as a global long-term reference, making it useless for local navigation. There is no specific reference to initialize the *Map* frame. The default configuration is located at the origin of the reference frame with the x-axis east, y-north and z-axis up, complying with the Geodetic and ENU Coordinate systems.

ROS uses broadcasters to connect the multiple reference frames. A broadcaster represents an homogenous matrix which contains both translation and rotation from one frame to another. For instance, Figure 4.4 depicts the transformation tree from one component frame to another in ORB-SLAM2 RGB-d. The broadcaster */Extra_broadcaster_tf2* contains the geometrical transformation from *camera_link* and *base_link* frame. [2]

[2]For more information about ROS frames refer to: Coordinate Frames for Mobile Platforms

4.2.2 Frame Alignment

VI-SLAM algorithms are not only able to map the surrounding scene but also provide an estimation of the localization of the camera along the trajectory. The evaluation of the chosen VI-SLAM requires a ground truth of the trajectory to be compared, this data is provided by the OptiTrack measurement system. A key point to accurately compare the trajectories is an appropriate matching between them. Commonly, the Horn Closed-form (96) method is used for this purpose, however, in this work also frame alignment is used in some cases as it provides more relevant results for evaluating the algorithms drift, understanding when loop closures take place or extracting the maximum error. Frame alignment adds an extra degree of complexity as the estimations of the positions of the camera in the VINS coordinate frame (Estimated Trajectory) have to be matched to the drone position in the Optitrack coordinate frame (Ground Truth Trajectory).

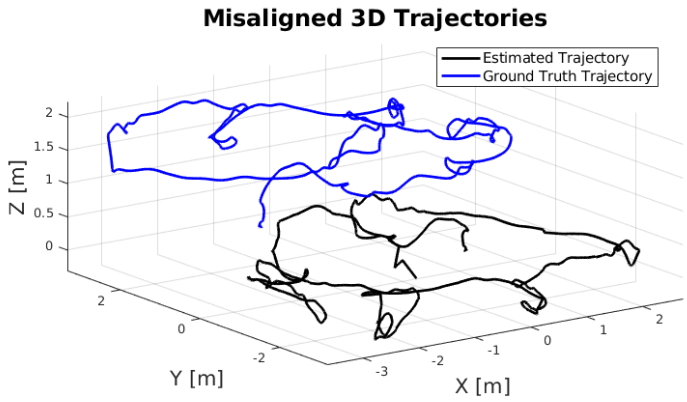


Figure 4.5: V1_01_easy EuRoC dataset: Trajectories without alignment

As depicted in Figure 4.5, the ground truth and the estimated trajectory are represented in two different coordinate frames, these are the OptiTrack system and the VINS system respectively. Therefore, the two methods used to align will be: aligning the trajectories to the initial frame or aligning the full trajectory minimizing the error between trajectories (Horn Closed-Form (96)).

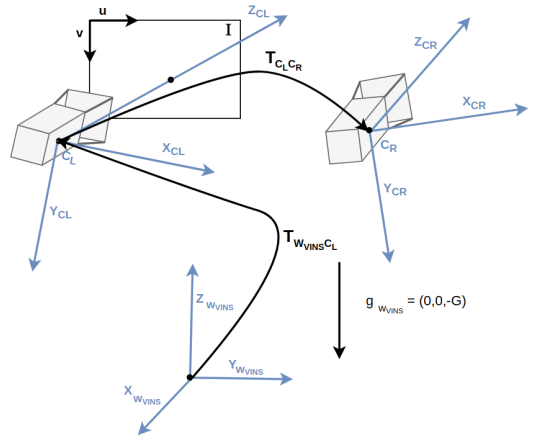
OptiTrack Frames

The OptiTrack system will have 2 coordinate frames: the OptiTrack world frame and the Drone reference frame captured by the OptiTrack. The OptiTrack world frame is initialized during the calibration, all the measurements taken by the OptiTrack will be concerning the previously defined world frame, in our case X and Y axes were in the ground plane and the Z-axis pointing upwards. The Drone reference frame is manually defined when creating the drone object on the OptiTrack using the reflective markers. The frame was set to be located on the centre of rotation around the Z-axis of the MAV and at the same height as the top base frame of the drone. The XY plane is contained on the top base plane of the drone and with the Z-axis pointing upwards.

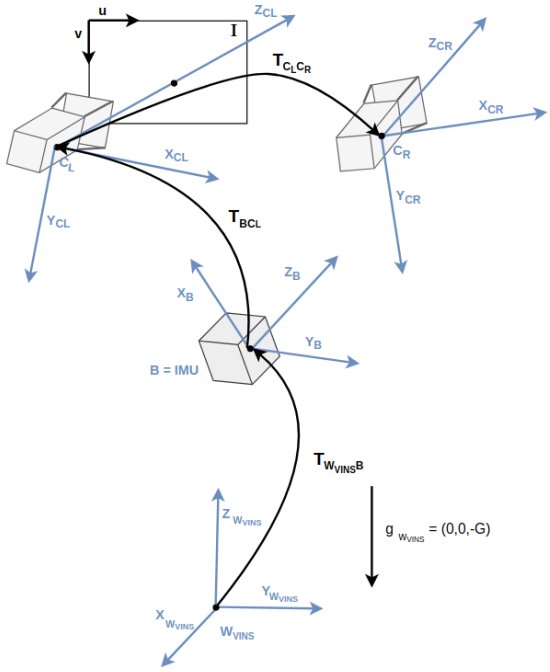
VINS Frames

Each VINS algorithm has 2 main coordinate frames: the VINS coordinate frame and the Camera coordinate frame. The VINS coordinate frame is the fixed reference frame, whose Z-axis points in the opposite direction to the gravity vector. In the pure case where no translation or rotations are applied, the world frame is set to align with the first camera pose.

In stereo algorithms, the left camera is used as the reference frame, where the points of the right image are matched to the left image points to compute disparity and estimate the depth of each matching pixel point as described in Figure 4.6a. In VI-SLAM the reference frame of the camera is the IMU frame, therefore the depth is estimated with respect to the left camera frame and then it is transformed to the IMU camera frame, being this the body camera frame as illustrated in Figure 4.6b. The orientation of each camera frame and ground frame varies depending on the used algorithm.



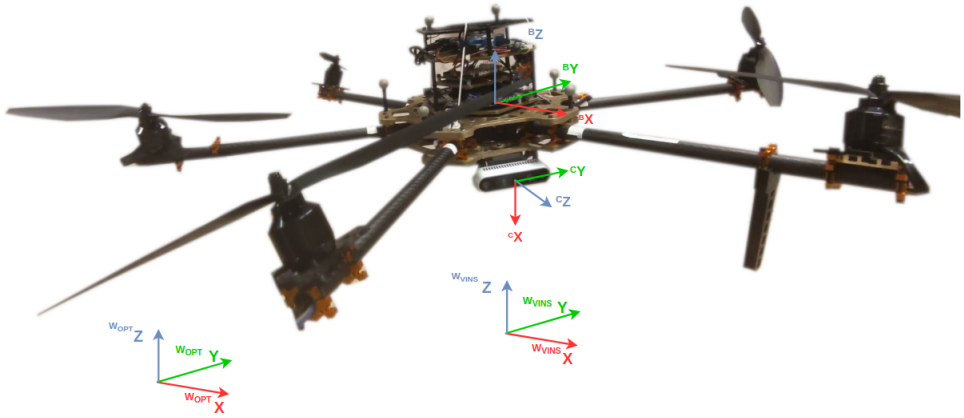
(a) Stereo mode.



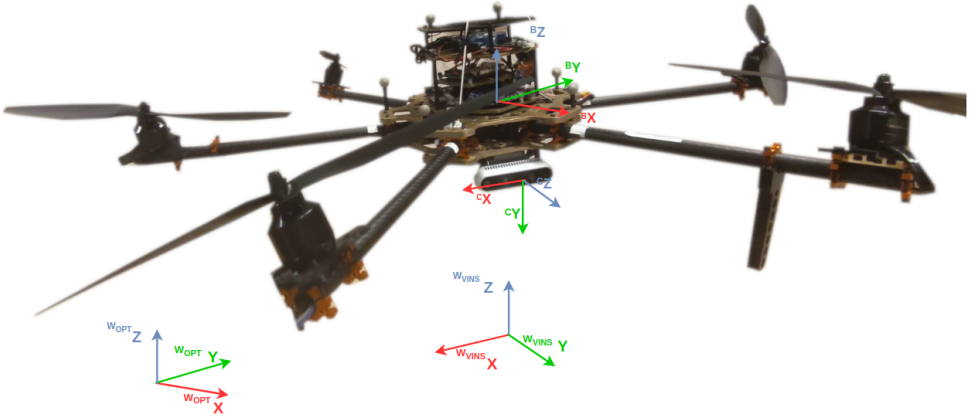
(b) Stereo mode with IMU.

Figure 4.6: Camera Frames

Figure 4.7 shows the frames in which the estimation and ground truth will be represented. The images show how the World frame of the OptiTrack and the Body frame remain constant for both algorithms (as described previously). On the other hand, the VINS World frame and Camera frame are initialized by the VINS algorithms, therefore Figure 4.7a and Figure 4.7b have different orientations but are initialized at the same positions.



(a) Coordinate frames from VINS and Optitrack in ORBLAM3 VIO stereo algorithm.



(b) Coordinate frames from VINS and Optitrack in ORBLAM2 stereo and ORBSLAM3 stereo algorithms.

Figure 4.7: Representation of the frames from the ground truth and the estimation systems from ORB-SLAM2 and ORB-SLAM3.

Changes in Reference frames

In the experiments, the effectiveness of the VINS estimated poses are estimated by means of comparing it against the ground truth estimations provided by the OptiTrack. Each of the trajectories denotes the poses of a different element. While the ground truth represents the position of the drone on the OptiTrack reference frame, VINS estimates represent the pose of the camera on the algorithm reference frame. This camera poses will be transformed to represent the estimations of the drone body on the OptiTrack reference frame.

$$^{W_{OPT}}P_C = R_B^{W_{VINS}} \cdot ^{W_{VINS}}P_C \tag{4.9}$$

$$^{W_{OPT}}P_C = R_B^{W_{OPT}} \cdot R_C^B \cdot R_{W_{VINS}}^C \cdot ^{W_{VINS}}P_C \tag{4.10}$$

Initially, with Equation 4.10 it is applied a transformation to change the camera poses from the VINS world frame (${}^{W_{VINS}}P_C$) to the OptiTrack world frame by applying a rotation matrix $R^{W_{VINSB}}$. $R^C_{W_{VINS}}$ is the inverse of $R_C^{W_{VINS}}$ which is equivalent to the transpose of this last matrix as for all rotational matrix $R^{-1} = R^T$. $R_C^{W_{VINS}}$ are the estimated orientations of the camera. R_C^B is the transformation from the camera frame to the body frame of the drone. Both frames are part of the drone's rigid body as depicted in Figure 4.7, thus this rotation matrix remains constant and will be manually computed for each of the algorithms.

$$R_C^B = Ry(90°) \cdot Rz(-30°) \quad \text{ORBSLAM2/ORBSLAM3} \tag{4.11}$$

$$R_C^B = Rz(120°) \cdot Rx(-90°) \quad \text{SVO Pro} \tag{4.12}$$

$R_B^{W_{OPT}}$ represents the change of orientation from the body to the OptiTrack world frame, which is directly extracted from the pose output from the OptiTrack measurements.

The Camera poses on the OptiTrack reference frame are now known, however, the Body poses of the drone are the ones required.

$$\boxed{{}^{W_{OPT}}P_B^{W_{OPT}} = {}^{W_{OPT}}P_C^{W_{OPT}} - R_B^{W_{OPT}\,B}P_C^B + t_{est}^{gnd} \quad \text{(Estimation Data)}} \tag{4.13}$$

Equation 4.13 represents the estimated drone poses on the world OptiTrack frame, being t_{est}^{gnd} the difference in position between VINS and OptiTrack frame. This difference is done using the data at the time i in which we want to align our frames.

$$t_{est}^{gnd} = {}^{GND}P_B^{W_{OPT}}(i) - {}^{EST}P_C^{W_{OPT}}(i) - {}^{EST}R_B^{W_{OPT}}(i) \cdot {}^{B}P_C^B(i)$$

A similar procedure applies to the orientation of the frames.

$${}^{EST}R_B^{W_{OPT}} = {}^{EST}R_C^{W_{OPT}} \cdot R_C^B$$

where R_C^B is the constant transformation between Camera and Body frame transformations. Then ${}^{EST}R_C^{W_{OPT}}$ is deduced from Equation 4.14.

$${}^{EST}R_C^{W_{OPT}} = R_{W_{VINS}}^{W_{OPT}} \cdot R^{W_{VINSC}} \tag{4.14}$$

where $R^{W_{VINSC}}$ is the orientation estimation of the camera by the VINS algorithms, and $R_{W_{VINS}}^{W_{OPT}}$ is a constant rotation matrix that represents the orientation difference between the VINS world frame and the OptiTrack world frame at the timestamp i at which it is preferred to align both frames, commonly at time 0s.

$$R_{W_{VINS}}^{W_{OPT}} = R_B^{W_{OPT}}(i) \cdot R_C^B \cdot R^C_{W_{VINS}}(i) \tag{4.15}$$

considering $R_B^{W_{OPT}}(i)$ to be the measurement of the OptiTrack at time i of the drone orientation, and $R^C_{W_{VINS}}(i)$ the inverse of the rotational matrix of the orientation at time i, $\left(R_C^{W_{VINS}}(i)\right)^{-1}$.

Once all the trajectories are representing the poses of the drone body on the OptiTrack reference frame, it is possible to evaluate the accuracy of each of the algorithms. The OptiTrack measurements are used as a reference to compute the difference between this trajectory and the estimated by VINS using different metrics.

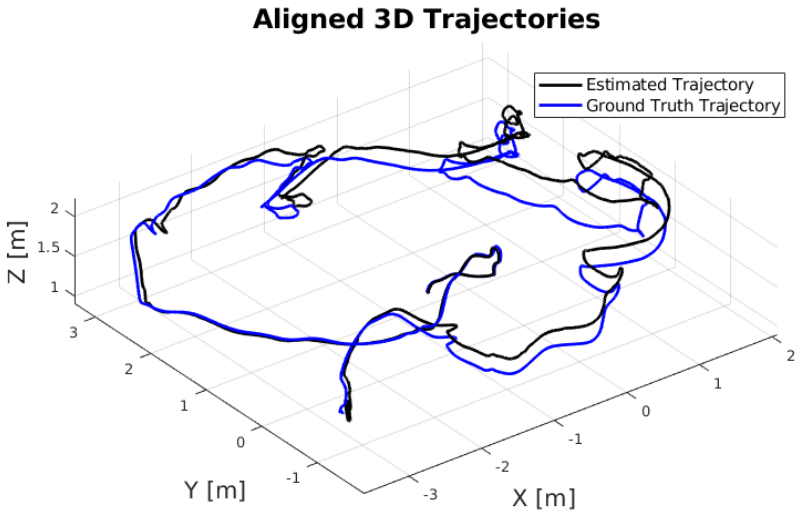


Figure 4.8: V1_01_easy EuRoC dataset: Trajectories with alignment to initial frame.

Figure 4.8 shows how we aligned the initial frames from both trajectories in position and orientation. The trajectories start at point (X, Y, Z) = (-0.979,0.436,1.444) and then the trajectories start to drift at certain points and optimization techniques will manage to decrease the error at certain points of the trajectory keeping the error bound.

4.2.3 Full Trajectory Alignment

The other alternative for aligning the trajectories is using the complete trajectories and minimizing the RMSE of the position. This option is broadly adopted by the community for both similarity and rigid body transformation, which is based on Horn closed-form (96). For severely corrupted data singular value decomposition implemented by Horn might reflect the rotation matrix ($det(R) = -1$). Umeyama method (97) overcomes this limitation, however, this is not our case and we mention this method to raise awareness in future projects.

Horn method is founded on a least-square minimization problem of the following function with at least three points. The matching between two different coordinate systems (p_i and $\hat{p}_i$) can be seen as a rigid body transformation. This transformation counts with 7 degrees of freedom: translations along 3 axes, rotations around 3 axes and the scale. 3 points provide 9 constraints for the 7 unknowns, therefore, discarding 2 constraints a system of equations with 7 equations and 7 unknowns can be set. The minimization problem is then defined as:

$$e^2(R,t,s) = \frac{1}{N}\sum_{i=1}^{N} \| p_i - sR\hat{p}_i - t \|^2 \tag{4.16}$$

where R,t and s correspond to applied rotation, translation and scale respectively. The translation is computed as the difference between the centroid coordinates of one system and the

scale rotation of the centroid of the other. Whereas the scale is equivalent to the ratio between the RMS (root mean square) of the coordinates of the two systems to their centroids. The rotational matrix is extracted through singular value decomposition.

There is no standard rule of how many points to use for the transformation computation. There are two possible ways: 1) Using all the positions. 2) Using only the initial states. The former gives a lower error when evaluating the complete trajectory, the latter gives a more intuitive error as it increases over time as seen from the experiments in (98).

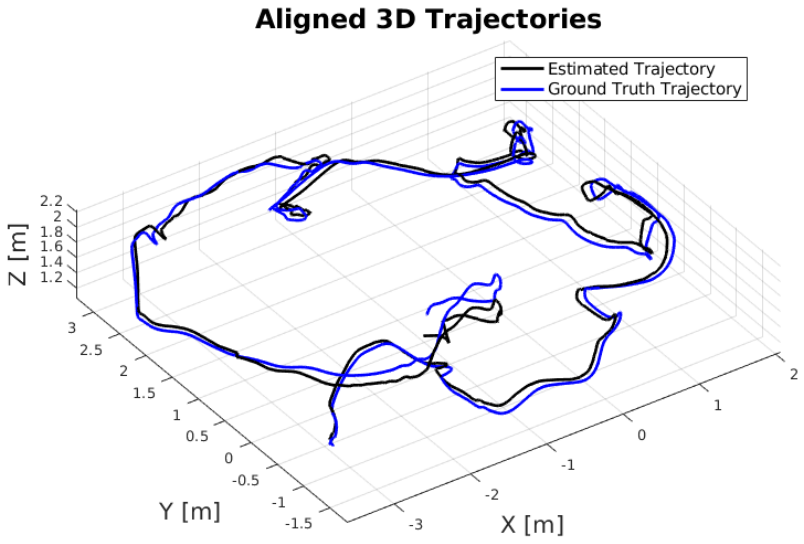


Figure 4.9: V1_01_easy EuRoC dataset: Trajectories with Horn Closed-Form alignment

As mentioned Horn Closed-Form alignment used as in Figure 4.9, minimizes the RMSE (root mean square error) of the matching positions between trajectories. The trajectories do not start aligned at the same position, it is seen how the estimated trajectory does not match with the initial ground truth point (X, Y, Z) = (-0.979,0.436,1.444). Nevertheless, it seems that the estimated trajectory follows more accurately the ground truth as by visual inspection it can be deduced that the errors between matching points are kept smaller.

4.3 Tools for Evaluation

The chosen algorithms will be tested against multiple metrics to evaluate their performance. The goal of using these metrics is to select the most convenient algorithm for our use case, performing the handover of a tool to a human. Human safety will be prioritised to guarantee against any other metric. The other properties that will also be evaluated are overall performance error along the complete trajectory, the robustness and consistency in the performance of these algorithms and their capability to run in long-life experiments without drifting.

4.3.1 Safety First!

At all times we want to guarantee that the drone will remain under a certain threshold error during the complete trajectory to avoid damaging the human during the cooperation. To measure the error an alignment of the initial frames between the estimated and the ground truth trajectories is accomplished as described in Section 4.2.2. Once the alignment is performed, the position from equivalent timestamps is compared to compute the error at each time step. This error should never exceed the threshold of 132.15 cm as deduced below.

Security distance

This section defines the security distance between the drone and the human. The error of the drone will never exceed this security distance at any point of the trajectory, otherwise, we would consider the algorithm to be unsuccessful in providing localization data. The security distance is defined by the capability of the human to stretch by just taking one step.

Considering the one step distance to stretch and the average lengths of the body represented in Figure 4.10 to be:

- Arm: 70 cm [3]
- Trunk: 54.3 cm [4]
- Step: 70 cm [5]

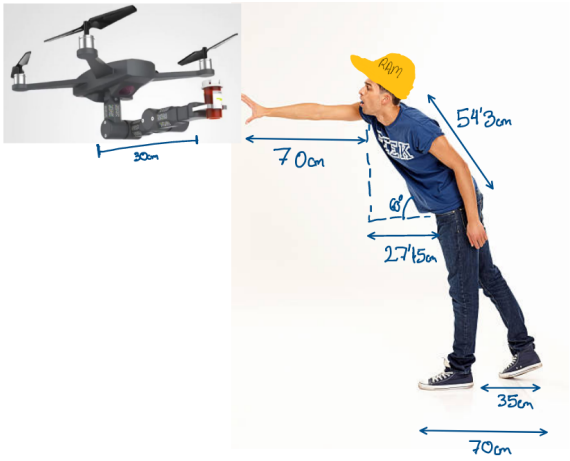


Figure 4.10: Estimated security distance used as threshold for VINS algorithms.

Finally, it is supposed that the human can bend 30 degrees to grab the object, as a result:

$$\begin{aligned}\text{Security Distance} &= \text{Arm} + \text{Trunk Projection} + \text{Half Step} \\ &= 70\text{cm} + 27.15\text{cm} + 35\text{cm} = 132.15\text{cm}\end{aligned}$$

It can be considered that an extra security distance of 30 cm is available in case of an emergency that corresponds to the drone arm that stretches to hand the tool to the human.

$$\begin{aligned}\text{Emergency Security} &= \text{Security distance} + \text{MAV arm} \\ &= 132.15\text{cm} + 30\text{cm} = 162.15\text{cm}\end{aligned}$$

[3] Average Arm length: https://hextobinary.com/unit/length/from/armlength/to/cm
[4] Average Trunk length: https://www.nature.com/articles/s41598-020-60813-w/tables/3
[5] Average Step Stride: https://www.healthline.com/health/stride-length#takeaway

4.3.2 ATE and RPE

Two frequently applied methods for comparing the estimated trajectories against their ground truth are the absolute trajectory error (ATE) and the relative trajectory error (RPE). These errors have been computed in multiple ways within the research community, however, we will stick to the implementations used on ORB-SLAM2 and SVO Pro to make the comparison between our results and our papers as fair as possible (99).

ATE measures the absolute pose difference estimated trajectory to the ground truth trajectory supposing both trajectories to be aligned. If we represent the true states as X and the estimated states as $\hat{X}$, then the difference between their orientation and position can be expressed as:

$$\Delta X_i = \{\Delta R_i, \Delta p_i, \Delta v_i\} \tag{4.17}$$

$$R_i = \Delta R_i \hat{R}_i, \quad p_i = \Delta R_i \hat{p}_i + \Delta p_i \tag{4.18}$$

We leave the error on the left side, hence:

$$\Delta R_i = R_i (\hat{R}_i)^T \tag{4.19}$$

$$\Delta p_i = p_i - \Delta R_i \hat{p}_i \tag{4.20}$$

Commonly, the root mean square is used to evaluate the quality of the trajectory:

$$ATE_{rot} = \left(\frac{1}{N} \sum_{i=0}^{N-1} \| \measuredangle \Delta R_i \|^2 \right)^{\frac{1}{2}} \tag{4.21}$$

$$ATE_{pos} = \left(\frac{1}{N} \sum_{i=0}^{N-1} \| \measuredangle \Delta p_i \|^2 \right)^{\frac{1}{2}} \tag{4.22}$$

This metric simplifies the evaluation of the algorithm's performance as it represents its accuracy with just a single number. Despite making comparison easy, ATE metric has some disadvantages as it is dependent on when the error occurs. When the error happens at the beginning the error will be larger than when it happens at the end.

RPE measures the difference between the estimated poses and the true poses. It is a useful tool to evaluate drift or the accuracy at loop closures. It provides extra information on ATE metric. This metric also requires an alignment between both datasets. RPE requires choosing k states, this k represents the number of poses along the trajectory that will be evaluated in each set. This provides us with details about the sub-trajectories in which we divided our poses such as the median, average or the percentiles. When k is small the RPE represents local consistency, alternatively, when k is large it denotes the performance of long-life experiments.

Following the RPE expression proposed by Scaramuzza in (99), where K are the pair of states selected by some kind of criteria such as a certain amount of displacement or a specific amount of seconds:

$$F = \{d_k\}_{k=0}^{K-1}, \quad d_k = \{\hat{x}_s, \hat{x}_e\}$$

where F contains the sets in which the trajectory is divided and d_k the initial and last set of a set. Similarly to ATE, for each set an alignment between the initial positions x_s and $\hat{x}_s$ is performed. To each subtrajectory the following errors are computed:

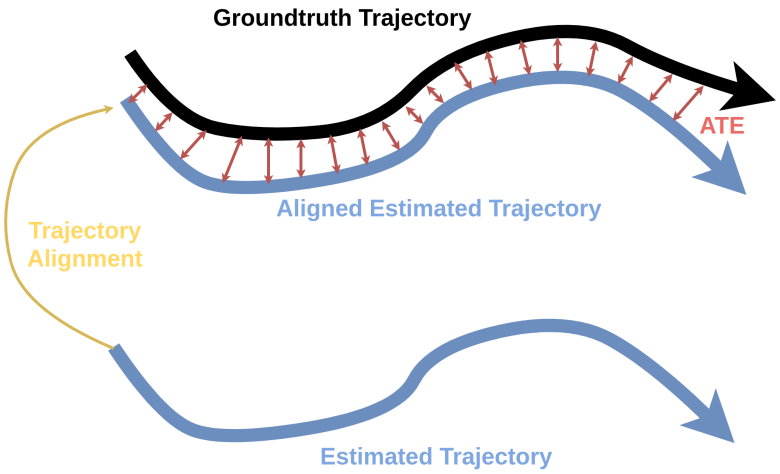


Figure 4.11: ATE error computation after trajectory alignment.

$$\delta\phi = \angle\delta R_k = \angle R_e (R'_e)^T$$
$$\delta p_k = \| p_e - \delta R_k \hat{p}'_e \|_2$$

Gathering all the errors from each of the sub-trajectories of F:

$$RE_{rot} = \{\delta\phi_k\}_{k=0}^{K-1}$$
$$RE_{pos} = \{\delta p_k\}_{k=0}^{K-1}$$

The RPE is harder to interpret compared to ATE which is just only a number. Nonetheless, it provides more information about the trajectory than ATE and different parameters can be evaluated by using different criteria to select the sub-trajectories. A disadvantage of this method is that the estimation quality is harder to be evaluated using an interval than a single number. A visual example of how RPE is computed can be seen in Figure 4.12. In our experiments we will apply frame alignment to evaluate the drift and loop closures using visual inspection, nevertheless, this is a useful tool to express this numerically.

4.4 Datasets

For the benchmarking evaluation, it is convenient to have datasets with the IMU measurements and visual feed. Datasets allow the running of multiple algorithms against the same inputs, they are especially helpful when testing the controls is not needed, as playing them is much quicker and repeatable than running them on real hardware and simulation platforms.

In our project, two datasets have been used to evaluate the most convenient algorithm to safely perform a handover to a human. The chosen datasets record a sequence of images and IMU measurements by flying a real MAV, contrary to most of the available Datasets which are recorded handheld or with UGV. The chosen datasets and the one recorded in RAM Arena at the University of Twente try to evaluate the performance of the algorithms under similar scenery conditions in terms of drone vibrations, visual inputs, IMU measurement and trajectory to a real handover to a human.

- The EuRoC MAV Dataset (100).
- Custom dataset collected at RAM Arena.

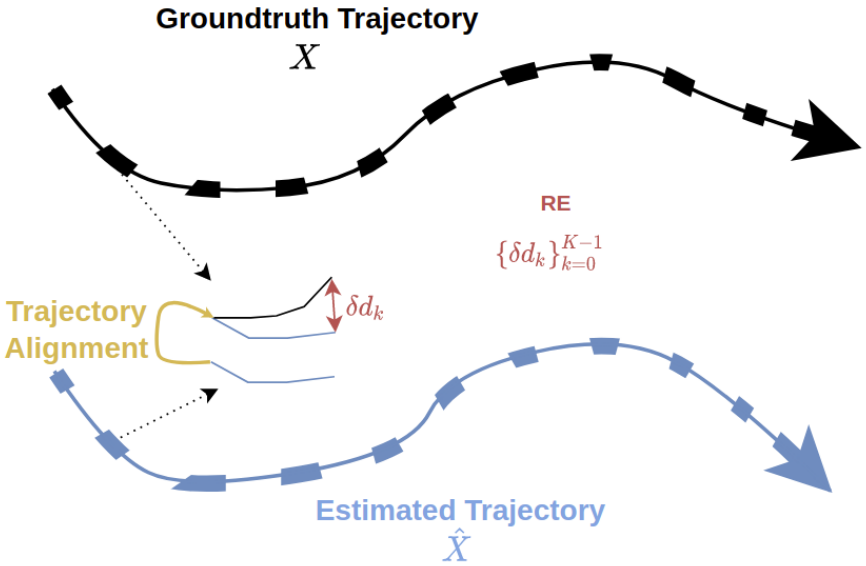


Figure 4.12: RPE error computation with initial frame alignment of each subtrajectory.

DATASET	YEAR	ENVIRONMENT	CARRIER	CAMERAS	IMU	TIME SYNC	GND TRUTH
EuRoC MAV	2016	Indoors	MAV	1 stereo gray 2x752x480 @20Hz	3-axis acc/gyro @200Hz	hw	Laser tracker pose @20Hz Motion Capture pose @100Hz, accuracy ~1mm
Custom RAM MAV	2022	Indoors	MAV	1 stereo gray 2x848x480 @30Hz	3-axis acc/gyro @200Hz	hw	Motion Capture pose @ 100Hz, accuracy~0.5mm

Table 4.6: Summary of the Datasets employed to evaluate the algorithms performance.

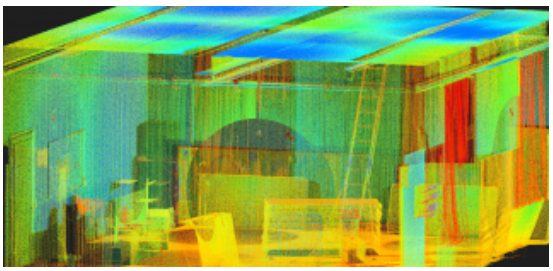
4.4.1 EuRoC Dataset

EuRoC Dataset contains sequences of images which were filmed by a real MAV. It contains 11 sequences in 3 different scenes, capturing stereo monochrome images at 20 FPS and taking IMU readings with acceleration and angular rate of 200Hz. These sequences contain ground-truth poses and scene information in case it is required to evaluate the generated map. The difficulty of each of the sequences is assigned by the authors in terms of speed in rotation and translation, lack of texture and darkness of the scenes. The algorithms will be tested against three of the datasets from EuRoC, these are "MH02_easy", "V1_01_easy" and "V1_02_medium". In "MH02_easy" the dataset is recorded inside a spacious hall, part of the room can be visualized on Figure 4.13a to give a sense of the space and the scale. This dataset has illumination changes and the detected visual features are far from the object. The datasets "V1_01_easy" and "V1_02_medium" test the MAV localization performance within a small office room, as seen in Figure 4.13b, where visual features are desks, stairs, boxes... With these datasets it is tested localization when features are closer, there are more rotations and, some dynamic elements such as curtains moving and a human moving around. For a simple description of the datasets see Table 4.7. Overall, these experiments help us to analyse the behaviour of lifelong experiments as the MAV is in constant movement throughout the dataset.

4.4.2 Custom RAM Datasets

Creating our custom dataset allows us to evaluate the performance of the algorithms using our hardware and deduce if we can achieve similar performance when running the MAV we have at RAM UT lab. The RAM Datasets have been recorded with the goal of testing specific dynamics that would be expected during the handover of a tool to a human. The MAV was

(a) Part of the room explored by the EuRoC Machine Hall sequences.



(b) Exploreed room on Vicon Dataset.

Figure 4.13: Partial overview of the two areas explored on EuRoC dataset.

NAME	LENGH/DURATION	AVG. VEL./ANGULAR VEL	NOTE
MH_02_easy	73.5m 150s	0.49m/s 0.21rad/s	Good texture, bright scene
V1_01_easy	58.6m 144s	0.41m/s 0.28rad/s	Slow motion, bright scene
V1_02_medium	75.9m 83.5s	0.91m/s 0.56rad/s	Fast motion, bright scene

Table 4.7: Description of the Datasets used from EuRoC copied from original paper (100).

controlled by inputting manually the position to the drone, the controller developed at RAM will be responsible for reaching that position by using the OptiTrack position measurements. As we know, one of the goals of this project is to remove the dependency on this system and be able to navigate without it.

The datasets have been recorded within the RAM Arena at UT, where the OptiTrack system was calibrated and tracks the MAV position accurately. The camera D435i was used to record stereo images and IMU readings while flying the MAV. The camera was configured with similar settings to EuRoC dataset as described in Table 4.7, the frame rate was set to 30 fps close to the 20 fps from EuRoC, the output resolution was 2x848x480, and the IMU worked with a frequency of 200Hz as in EuRoC. This camera uses laser measurements that are captured by the infrared stereo camera to improve the internal depth estimation, however, these laser projections are captured as points on the images by the infrared cameras. During experimentation, it was deduced that these points were wrongly interpreted as feature points generating misbehaviour of the VIO localization algorithms. Hence, the laser projections have to be deactivated to use VO/VIO localization.

The VIO algorithms require a starting period where the scales and the IMU readings are being initialized. Therefore all the custom datasets count with the following structure:

- Initialization Stage: the drone is manually moved along and around the 3 axes. This initialization takes around 15-25 seconds.
- Land Stage: the drone stays on the floor after the dynamic initialization while the RAM controller is initialized.
- Action Stage: during this period the MAV is flown by directly inputting the desired position. The dynamics from Table 4.8 are tested and experiments finished with a landing.

The datasets recorded are:

- Static2 and Static3: the drone takes off and remains to hover at the same position and at two different heights to test the drone does not drift under this situation.
- Translation_0_1ms: the drone performs a simple translation moving along the Z-axis.
- Waving and Dynamic: the drone takes off and hovers during the complete dataset. Meanwhile, apart from the static objects, we will move a stick with a glove as if a man was waving the drone close to the drone. On the Dynamic dataset, we also used two persons to constantly move in front of the drone at 3 meters distance to check if the drone would be able to handle localization under this situation.
- Square2, Square3, Square4_max and Square5_max: the drone draws a square trajectory at different speeds. When it reaches back the initial position the drone completes a pure yaw rotation. On "Square4_max" and "Square5_max" the drone moves as fast as possible between waypoints, and the lateral bound force is increased from 2N to 3N in "Square5_max". This is the maximum lateral force that the drone can achieve without tilting, therefore the MAV is able to move 0.2m/s faster between the waypoints.
- Translation_Frontback and Translation_Frontback_max: the drone performs pure translations moving backwards and forwards.

NAME	DURATION	NOTE
Static2	187s	Hovering at 60cm heigh
Static3	123s	Hovering at 70cm heigh
Translation_0_1ms	112s	Up and down 3 times (0.3m height) Speed 0.1m/s
Waving	124s	Hovering Dynamics with a glove in front of the camera
Dynamic	150s	Hovering People moving Dynamics with a glove in front of the camera Reflectoins
Square2	147s	Square motion (average speed 0.1m/s, distance between waypoints 0.5m) + Pure Yaw rotation (average speed 20º/sec, rotated distance 60º both ways)
Square3	182s	Square motion (average speed 0.2m/s, distance between waypoints 0.5m) + Pure Yaw rotation (average speed 20º/sec, rotated distance 60º both ways)
Square4_max	118s	Square motion (average speed 0.4m/s, distance between waypoints 0.5m) + Pure Yaw rotation (average speed 20º/sec, rotated distance 60º both ways)
Square5_max	110s	Square motion (average speed 0.6m/s, distance between waypoints 0.5m) + Pure Yaw rotation (average speed 20º/sec, rotated distance 60º both ways)
Translation_Frontback	203s	Moving forward and backwards for 90s Translation displacement ~0.25m/s Distance between waypoints 0.5m
Translation_Frontback_max	195s	Moving forward and backwards for 90s Translation displacement ~0.4m/s Distance between waypoints 0.5m

Table 4.8: RAM Dataset where we test dynamics similar the ones a MAV might perform during a handover to a human.

5 Results

In this section it is presented the outcomes of testing the algorithms that were chosen as the most promising conventional VINS solutions. The algorithms are tested against the EuRoC dataset recorded by a MAV and our custom dataset. The extracted results will provide us with the most convenient solutions for our research questions, helping to clarify which are the most appropriate algorithms to perform a handover of a tool to a human to permit secure cooperation between humans and MAVs.

5.1 Absolute Trajectory Error in Position

As a general rule, the ATE metric assists us to deduce which algorithms have the best performance against the tested dataset. To compare the performance it is mainly taken into account the median value of the absolute trajectory error of the runs of each algorithm in each dataset. Previously exposed papers in Table 3.1 considered that 5 runs per algorithm in each dataset were a decent number to evaluate the algorithm's performance as exposed. Then, the median ATE value of the 5 runs of each algorithm is used as a comparison metric between the algorithms, extracting which VINS provide the best localization performance.

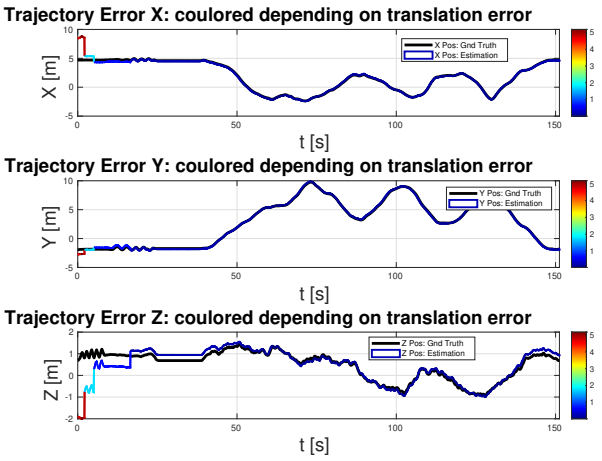
The algorithms do not provide consistent performance during all the runs because RANSAC iterative estimation is in charge of filtering outliers during feature detection, as it is a non-deterministic method the results vary from run to run.

All papers used in Table 3.1 and reference papers of the algorithms ((90),(94) and (88)) use full alignment between ground truth and estimations to extract ATE values. From our point of view, this metric is not fair as it does not represent the real drift that happens along the trajectory, however, to allow us to compare our results against the mentioned papers required to use to the same alignment. Nevertheless, the results will also be exposed using frame alignment, as it is a useful method to extract relevant deductions from the algorithms.

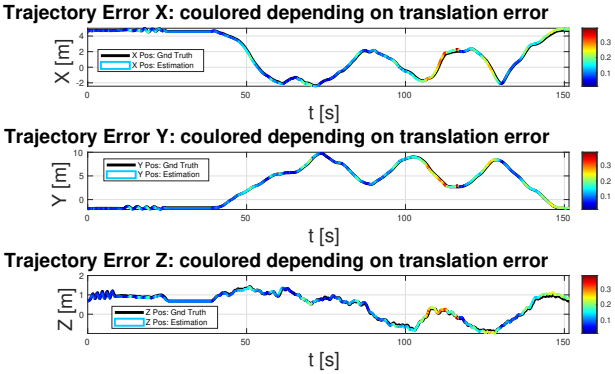
5.1.1 EuRoC

Table 5.1 presents the ATE run values in meters of the tested algorithms against each of the datasets. It is deduced that the algorithms that made use of an IMU took the initial dynamics of the run to initialize the IMU parameters, this behaviour caused step jumps on the trajectory until the scale was stable. (90) clearly states that the first 15 seconds of navigation where the drone is moving are used to make the map of ORB-SLAM3 to be stable (we will then consider the map to be mature). Once the initialization is successful and the map is mature it will not suffer from these discrete jumps again. The plots presented in Figure 5.1 show how SVO Pro does not make any discrete jumps while initializing the IMU parameters. However, if it is not properly initialized it will achieve poor performance as happened in some of our custom RAM datasets.

Results from Table 5.1 show our results and in parenthesis the results from the official algorithms papers, occasionally the values were provided scaled. From our side, it was decided not to scale the results for three main reasons, the former is that in real navigation there is no chance to scale the trajectory so our values are more realistic than what we would expect in real experiments. Additionally, if there are deviations on the trajectory increasing or decreasing the scale may allow reducing the error, but at the expense of decreasing the performance at those points where the trajectory was correctly estimated. Finally, when testing the performance of the scaled trajectories for these datasets the scale error was always between 0-5%. The ATE



(a) ORB-SLAM3 VIO



(b) SVO Pro VIO

Figure 5.1: VIO stereo algorithms

metric changed in the order of mm, considering then to have a decent performance without the need for scaling.

ATE values were larger when running the experiments from our side than the runs shown in the original research papers. Multiple factors might have an impact on the achieved results such as the hardware used, the RANSAC runs removing better outliers or possible fine tunning of each of the individual algorithms.

The Table 5.1 show that algorithms that only have a font-end and do not count with back-end optimizations provide worst results compared to the rest of the VINS algorithms. To visualize the loop closure effect it is required to align the initial frames of both trajectories as in Figure 5.2a. The plot represents the estimated trajectory, which changes its colour depending on the error between its corresponding point of the ground truth. By visual inspection, it is detected how the error starts very low after the alignment and consecutively the estimation starts to drift accumulating error. This algorithm counts with a window optimization that optimizes the last K poses. This behaviour is depicted in Figure 5.2b, where the error value increases from yellow to red and then optimization takes place reducing drift and going back to yellow colours. Yet, when reexamining previously seen features after a period out of the optimization window then optimizations cannot take place. On the other hand, algorithms that count with back-end

ALGORITHM\DATASET	MH02_EASY	V1_01_EASY	V1_02_MEDIUM
ORB-SLAM3 VIO MONO	0.309 (0.037-scale error 0.3%)	0.271 (0.049-scale error 2%)	0.311 (0.015-scale error 0.6%)
ORB-SLAM3 VIO MONO MATURE	0.149 (0.037-scale error 0.3%)	0.091 (0.049-scale error 2%)	0.105 (0.015-scale error 0.6%)
ORB-SLAM3 STEREO	0.128 (0.019)	0.369 (0.035)	0.485 (0.025)
ORB-SLAM3 STEREO MATURE	0.129 (0.019)	0.403 (0.035)	0.105 (0.025)
ORB-SLAM3 VIO STEREO	0.439 (0.033-scale error 0.2%)	0.171 (0.038-scale error 0.8%)	0.714 (0.014-scale error 0.6%)
ORB-SLAM3 VIO STEREO MATURE	0.143 (0.033-scale error 0.2%)	0.085 (0.038-scale error 0.8%)	0.111 (0.014-scale error 0.6%)
SVO PRO VIO MONO	0.151 (-)	0.193 (-)	0.162 (-)
SVO PRO VIO MONO FRONT-END	4.092 (-)	0.767 (-)	1.44 (-)
SVO PRO STEREO	0.1667 (0.05)	0.137 (0.05)	0.148 (0.05)
SVO PRO VIO STEREO	0.155 (0.036)	0.101 (0.041)	0.129 (0.048)

Table 5.1: ATE metric from EuRoC datasets without scaling. Units: meters

optimizations are able to correct the accumulated drift when a previously seen area is revisited. For instance, when looking at the Z-axis displacement in Figure 5.3b it is seen that the trajectory starts to drift at 60 seconds, then around 105 seconds when the initial positions are revisited the algorithm corrects the drift reducing the accumulated error.

The mature implementations demonstrate how removing the initial 15 seconds (where the IMU parameters are being initialized) cause a major improvement of the algorithm performance as depicted in Table 5.1. Due to an implementation bug from ORB-SLAM2 stereo ROS wrapper provided by appliedAI, rectification of the images was not done correctly. Hence, the feature points positions in these datasets were wrongly estimated and localization was not correct. As an alternative, ORB-SLAM2 RGB-D performance was evaluated in similar scenarios to the EuRoC ones. The datasets used were part of TUM RGB-D dataset where "Frieburg1_xyz" is a navigation around a desk within an office room and "Frieburg1_room" is a navigation within a spacious room, where the objects are far from the camera. These experiments provide us an intuition of the performance of ORB-SLAM2 algorithm and, as mentioned in (101), the expected results in indoor navigation are very similar in RGB-D and Stereo methods. Table 5.2 show that ORB-SLAM2 has similar ATE performance to ORB-SLAM3 and SVO Pro so ORB-SLAM2 is considered as a promising option and will be tested against our custom datasets. [1]

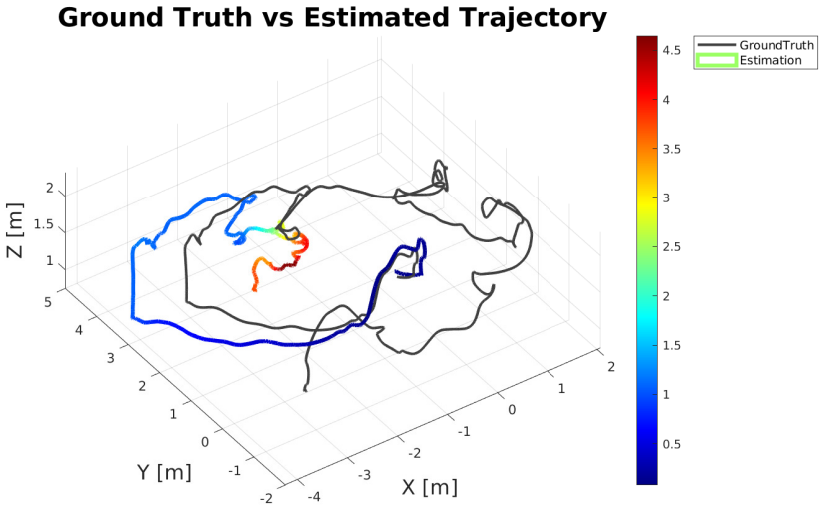
ALGORITHM\DATASET	FRIEBURG1_ROOM	FRIEBURG1_XYZ
ORB-SLAM2 RGB-D	0.143	0.197

Table 5.2: ATE metric from TUM RGB-D datasets without scaling. Units: meters.

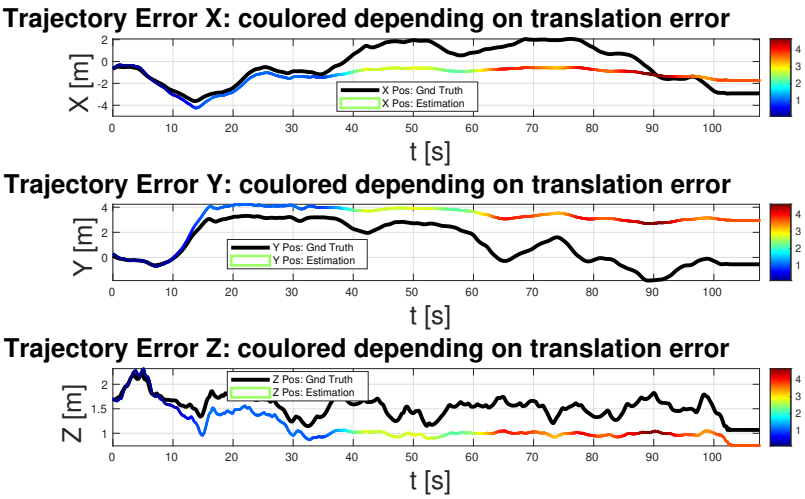
Robustness

Measuring how much the ATE deviates within an algorithm with the same dataset provides a good insight into the robustness of each algorithm. It is preferred a robust implementation that consistently provides low absolute trajectory error. Considering the 5 runs from each algorithm we build the graph from Figure 5.4a. This graph provides the information on the median value of the ATE metric among all of the runs and how the standard deviation of the rest of the runs. The interpretation of this graph provides a much higher understanding of the performance.

[1] ORB-SLAM2 library: https://github.com/appliedAI-Initiative/orb_slam_2_ros



(a) 3D graph.

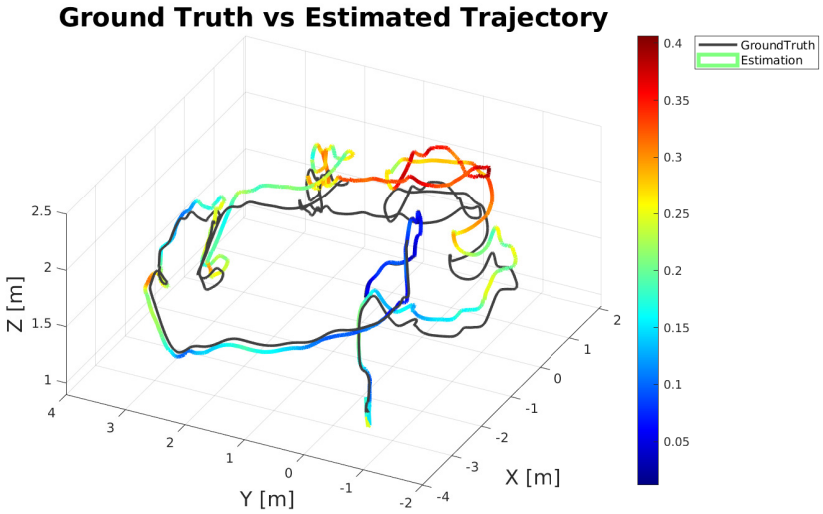


(b) Trajectories in X,Y and Z axis.

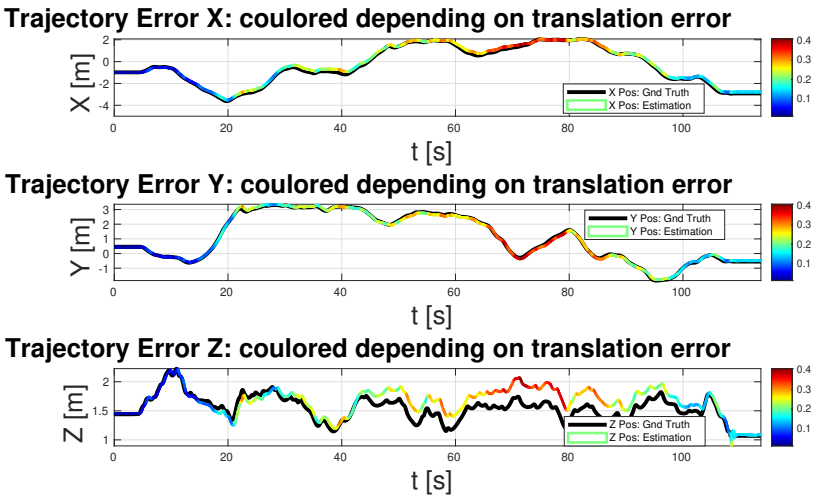
Figure 5.2: SVO Pro monocular front-end trajectory estimation of dataset V1_01_easy from EuRoC dataset.

It is deduced that ORB-SLAM3 VIO runs that include the initial dynamics are less robust and with a higher median than their corresponding mature implementations. The ORB-SLAM3 Stereo implementation does not require removing the initial dynamics as the results do not differ much between both implementations. The front-end implementation of SVO Pro has clearly the worst performance in comparison to the other alternatives, where the results for "MH02_easy" datasets are not visible on the graph because all the values provided a higher value than 2m.

Finally, the goal of this dataset is to find the most promising algorithms and filter them for more intensive testing using the custom dataset recorded at RAM. Inspecting the results from Figure 5.4b and Figure 5.4c both Monocular with IMU implementations are discarded. Although the results are very close to their corresponding stereo implementations they seem to have slightly worst robustness. The standard deviation in "MH02_easy" in ORB-SLAM3 VIO Monocular implementation is higher and in SVO Pro VIO Monocular has smaller standard



(a) 3D graph.



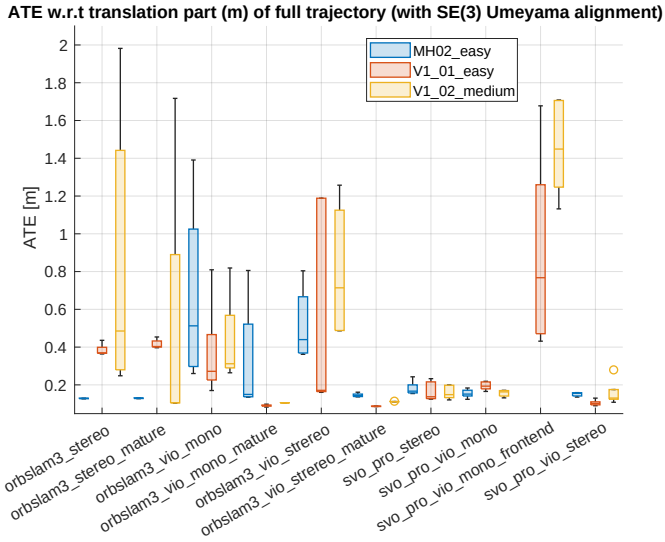
(b) Trajectories in X,Y and Z axis.

Figure 5.3: SVO Pro VIO monocular trajectory estimation of dataset V1_01_easy from EuRoC dataset.

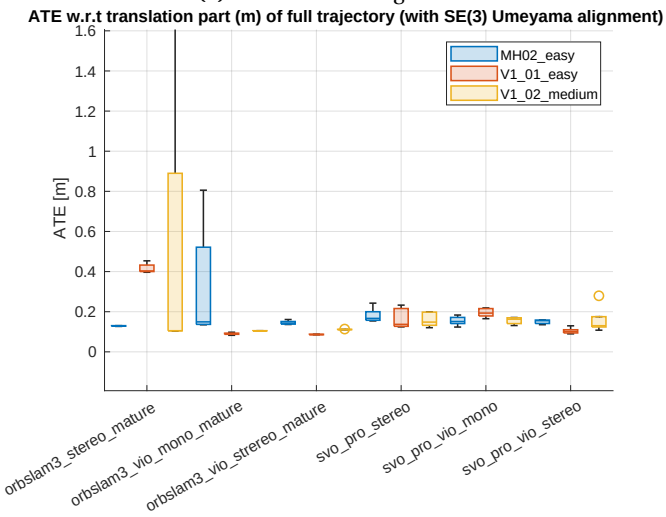
deviations in 2 out of 3 cases.

The algorithm's results with the best performance are gathered in Figure 5.4b and Figure 5.4c, these algorithms are robust due to their small standard deviation. The different runs vary in the order of centimetres. In general, the algorithms seem to perform with higher accuracy when the visual features are closer to the MAV during navigating, as the ATE results for "MH02_easy" dataset are slightly higher than for the Vicon datasets. This is not the case for ORB-SLAM3 stereo mature algorithm, which provides the best performance in terms of robustness in this dataset.

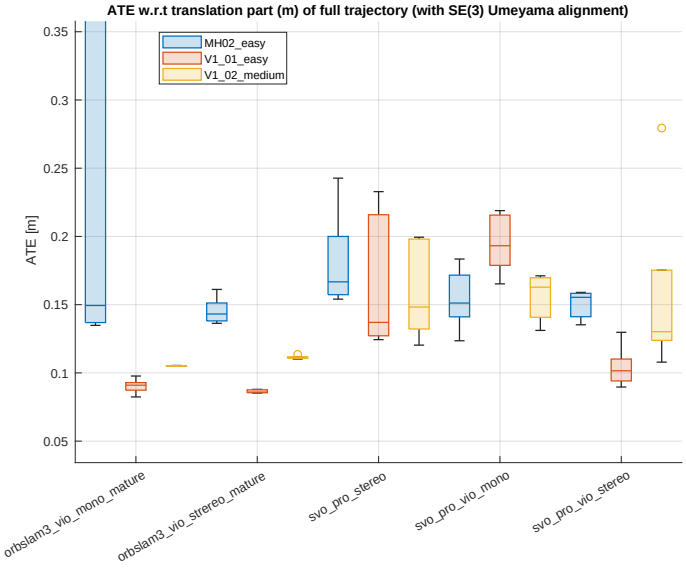
"V1_02_medium" dataset is the most challenging scenario in terms of dynamics. ORB-SLAM3 stereo mature has a low median ATE value, yet there are runs with very high ATE error. On the other hand, ORB-SLAM VIO algorithms keep the error low, therefore, in challenging scenarios with high dynamics the IMU readings allow to achieve more accurate estimations. ORB-SLAM3 algorithms tend to be much more robust than SVO Pro algorithms in these experiments as the



(a) All the chosen algorithms.



(b) Best algorithms.



(c) Best algorithms zoomed.

Figure 5.4: Absolute Trajectory Error (ATE) of chosen algorithms in EuRoC dataset

boxplot percentiles have a lower ATE value. When comparing the SVO Pro tests it seems that SVO VIO stereo provides not only the best median but also the lowest deviation in its results, it is considered then to be the most robust option.

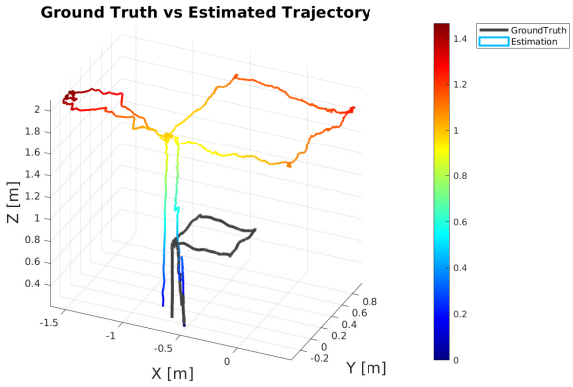
5.1.2 Custom RAM datasets

The EuRoC datasets were useful to provide us with higher insight into the performance of the algorithms. As the dataset was recorded using a MAV, it guaranteed similar vibrations to the ones expected when running purely on the VINS for localization. The next step is testing the algorithms under situations that the MAV will face when performing a real handover of a tool to a human. When recording these experiments the initialization was done handheld. After this initialization, the drone was left on the floor and then the desired positions were input programmatically to the MAV.

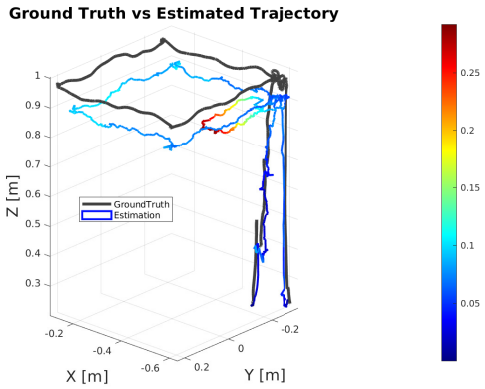
Full alignment is again applied to evaluate the performance of the algorithms against the built datasets to be consistent with the EuRoC dataset. From the initial tests, several discrepancies were found between algorithms when performing in the EuRoC dataset and our custom datasets. The extracted results showed scaling issues for all the implementations, where VIO algorithms seemed to be less affected. In Table A.1 all the scaled and non-scaled values are presented, for the stereo implementations the scaling error seemed to be consistent along with the multiple runs of each algorithm. VIO algorithms make use of the IMU readings to retrieve the scale together with the camera disparity, so with the support of the IMU measurements, they achieve a much more accurate scale. The problem seems to be with the camera calibration in the case of ORB-SLAM2/3 due to the improvement achieved with the IMU estimations.

Meanwhile, the VIO algorithms seemed to have much smaller scaling errors as depicted in Figure 5.5b. ORB-SLAM3 is the one with the smallest error, this effect is due to the IMU being used to retrieve the scale together with the camera disparity. As the IMU is providing good estimations the scale is closer to the correct value despite the inaccurate calibration of the camera images. A representation of the scaling errors can be found in Figure 5.5 where all the stereo implementations have similar scaling problems as Figure 5.5a, either overestimating or underestimating the trajectory.

The scaling error in ORB-SLAM2 and ORB-SLAM3 is caused by inaccurate camera calibration. As the Intel camera D435i claims to output rectified images, then only the on-chip calibration was done and we skipped the Kalibr camera calibration with this algorithm. However, rectification coming from the camera D435i is not accomplished correctly, as the used parameters for ORB-SLAM3 are identical (default parameter settings) in all runs from EuRoC and our custom datasets. Hence, the results have been scaled manually to provide a similar result to the ones we would have achieved by applying a proper image rectification. On the other hand, SVO Pro had a built-in functionality to rectify the images provided by the cameras, having this functionality as optional with ORB-SLAM3, nonetheless, the scale is incorrect as well. The reason behind the scaling errors in SVO Pro is that this algorithm is semi-direct, capturing all the pixel intensities and the infrared camera captures the projected rays by the OpriTrack. The rays caused a flickering effect on the sequence of images captured impacting the corners and edges detection in SVO Pro as it changed the contrast between consecutive frames. This behaviour has no impact on pure feature-based methods like ORB-SLAM3 as features are being detected locally in each frame from maximums and minimums pixel points. Nevertheless, the extracted results can still be used because SVO Pro is a feature-based algorithm supported by the direct measurements, using non-infrared cameras the results can only improve under these conditions.



(a) ORB-SLAM3 stereo



(b) ORB-SLAM3 VIO

Figure 5.5: Graph representing the scaling errors in Square2 dataset.

This last theory was proven by moving the camera along the edge of a table square table of 60cm x 60 cm located out of the arena. The experiment from Figure 5.6 proves that SVO Pro was able to estimate correctly the table dimensions without incurring the scale errors experienced in the arena. The side lengths of the left and top sides of the estimated table are approximately 0.6323 and 0.667 which are 5% and 10% of error, far from the 50% estimation error. The conducted experiments on EuRoC datasets show that the scaling error in the algorithms is minimal. Therefore, for future experiments, we would highly improve the results of navigations by performing a correct rectification for ORB-SLAM3 such as the one conducted in SVO Pro and by using non-infrared cameras to avoid capturing the laser from the OptiTrack.

The last point that should be considered is that the extracted results for our custom datasets shown in Table 5.3, Table 5.5 and Table 5.4 do not consider the full trajectory. The manual initialization phase is discarded because VIO algorithms tend to take some seconds to provide a stable scale and would not be a fair comparison between the VIO and VO algorithms. The landing section was also removed as some of the datasets were recorded with emergency landings due to the low battery available on the drone. The low battery caused the drone to enter an emergency landing more impacting negatively the measurements. Eliminating this section does not impact negatively our evaluation as the desired dynamics that we want to test are preserved in all of the datasets.

By inspecting absolute trajectory errors in Table 5.3, it is deduced that all the algorithms have a good localization estimation when performing hovering with and without dynamic features on the experiments performed. The difference between them is not significant and there is no clear winner. For the dataset "Static2" the VIO algorithms did not initialize correctly because

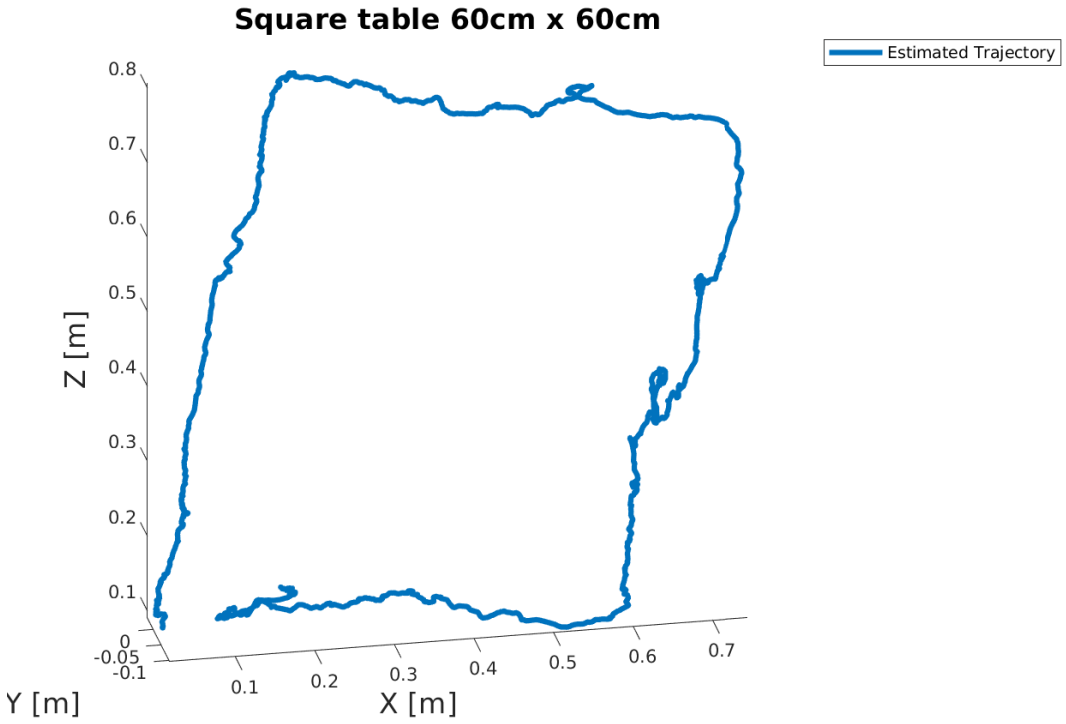


Figure 5.6: SVO Pro navigation around square table with side length of 60cm.

DATASET\ALGORITHM	STATIC2	STATIC3	WAVING	DYNAMIC
ORB-SLAM2 STEREO	0.004	0.022	0.019	0.008
ORB-SLAM3 STEREO	0.004	0.008	0.014	0.011
ORB-SLAM3 VIO STEREO	—	0.013	0.009	0.016
SVO PRO STEREO	0.007	0.005	0.033	0.008
SVO PRO VIO STEREO	—	0.014	0.035	0.021

Table 5.3: ATE metric from the custom RAM datasets Static2, Static3, Waving and Dynamic with scaled values. Units: meters.

the phase at which high dynamics were performed for the initialization was not long enough. This caused that after leaving the drone on the floor it required a completely new initialization, therefore, this dataset was useless for VIO algorithms.

DATASET\ALGORITHM	TRANSLATION_0_1MS	TRANSLATION_FRONTBACK	TRANSLATION_FRONTBACK_MAX
ORB-SLAM2 STEREO	0.024	0.041	0.024
ORB-SLAM3 STEREO	0.007	0.010	0.008
ORB-SLAM3 VIO STEREO	0.009	0.016	0.022
SVO PRO STEREO	0.010	0.033	0.028
SVO PRO VIO STEREO	0.045	0.098	0.435

Table 5.4: ATE metric from the custom RAM datasets Translation_0_1ms, Translation_frontback, translation_frontback_max with scaled values. Units: meters.

In Table 5.4 the datasets represent pure translations in the XY plane in "Translation_FrontBack" and "Translation_FrontBack_max", and translation along the Z-axis for "Translation_0_1ms". SVO Pro VIO stereo implementation provides an estimated localization which is more than 9 times more inaccurate than ORB-SLAM3 stereo, which is the algorithm with the best performance for all these three datasets, followed closely by ORB-SLAM3 VIO stereo.

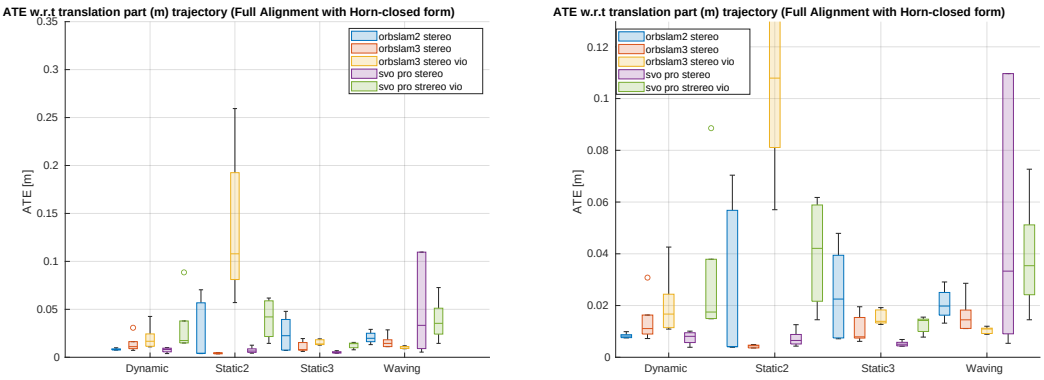
DATASET\ALGORITHM	SQUARE2	SQUARE3	SQUARE4_MAX	SQUARE5_MAX
ORB-SLAM2 STEREO	0.089	0.038	0.087	0.075
ORB-SLAM3 STEREO	0.071	0.049	0.057	0.065
ORB-SLAM3 VIO STEREO	0.066	0.018	0.087	0.056
SVO PRO STEREO	0.051	0.035	0.035	0.059
SVO PRO VIO STEREO	0.119	0.014	0.045	0.035

Table 5.5: ATE metric from the custom RAM datasets Square2, Square3, Square4_max and Square5_max with scaled values. Units: meters.

Lastly, it is evaluated the presented results on Table 5.5, the datasets record the translation along an imaginary square and finish with a pure yaw rotation. In these datasets it is highlighted the performance of SVO Pro implementations. Direct localization algorithms outperform the indirect ones when applying pure rotations, as deduced from the papers mentioned in Table 3.1. As SVO Pro is a semi-direct method it seems to have a positive impact when performing pure rotations. As mentioned earlier, a higher improvement of SVO Pro is expected when using non-infrared cameras as the direct feature tracking will not be affected by OptiTrack laser. ORB-SLAM3 VIO stereo seems to perform slightly better than the stereo implementation. The cause seems to be that VO only relies on visual features which are easily lost during pure rotations while VIO algorithms rely also on IMU measurements. As an outcome of testing these 11 datasets, it is deduced that novel algorithms outperform the ORB-SLAM2 stereo algorithm, which it was never chosen as the preferred solution. An example of the achieved performance can be found in Figure A.4, Figure A.6 and Figure A.5, where some ORB-SLAM3 stereo graph runs are plotted.

Robustness

The consistency of the algorithms was also tested on our custom datasets as it provides a higher insight into the performance than just referring to the ATE to choose an appropriate solution. To evaluate this metric correctly the runs in which VIO algorithms were not able to initialize correctly were removed. The reasons for bad initialization were always due to having a short initially dynamic phase, this caused that when leaving the drone on the floor the algorithm needed a new initialization from scratch. This problem can be easily avoided in future experiments by extending the time employed in the initialization phase.

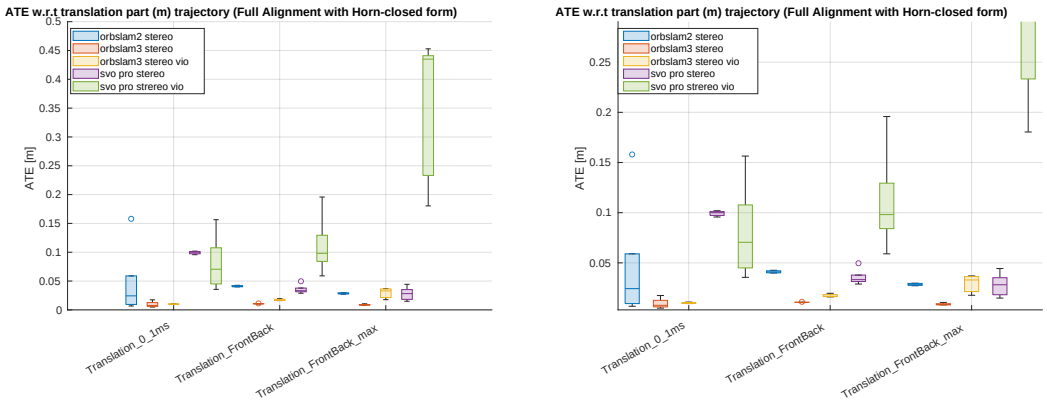


(a) Full view. **(b)** Zoomed view.

Figure 5.7: ATE runs in Custom RAM datasets Dynamic, Static2, Static3 and Waving

Figure 5.7 represents the datasets in which the MAV takes off and hovers in front of both static and dynamic features. In the represented results the values for ORB-SLAM3 VIO stereo should not be considered, as the algorithm was not able to initialize correctly in any of the cases. The

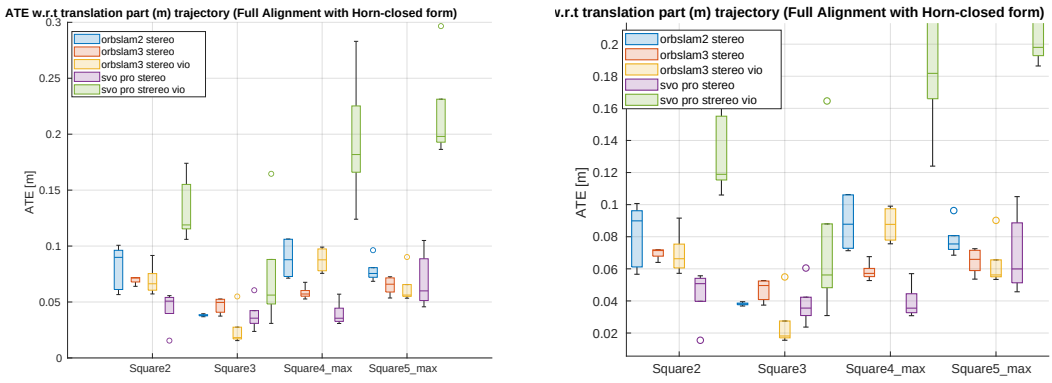
experiments show that adding dynamic features (Waving and Dynamic datasets) does not have a meaningful impact on the algorithm's performance. Overall, ORB-SLAM3 implementations tend to slightly outperform ORB-SLAM2 and SVO Pro, as the ATE values are lower. From this experiment, it is noticed that ORB-SLAM3 did improve its predecessor ORB-SLAM2. Additionally, VIO implementations seem to perform marginally worst than stereo algorithms. A possible explanation would be that when MAV remains hovering the dynamics are low and the readings provided by the IMU readings have lower accuracy than with high dynamics.



(a) Full view. **(b)** Zoomed view.

Figure 5.8: ATE runs in Custom RAM datasetsTranslation_0_1ms, Translation_FrontBack and Translaion_FrontBack_max.

To gain an intuition regarding pure translations in XY plane and Z axis the results from Translation_0_1ms, Translation_FrontBack and Translation_FrontBack_max are inspected in Figure 5.8. SVO Pro stereo achieved trajectories with higher robustness than the implementations which included IMU measurements. A possible cause could be that the position estimations provided by the IMU measurements highly differed from the estimations provided by the visual inputs as an effect of the OptiTrack lasers. In all three scenarios, ORB-SLAM3 seems to be the most robust option, achieving high performance followed by ORB-SLAM2 solution.



(a) Full view. **(b)** Zoomed view.

Figure 5.9: ATE runs in Custom RAM datasets Square2, Square3, Square4_max and Square5_max.

The last datasets are the most challenging ones, where a pure translation is preceded by a combination of translations that draw a square as depicted in Figure 5.9. Again, a similar conclusion is deduced, where ORB-SLAM3 accomplished better results than ORB-SLAM2. SVO Pro VIO stereo cannot be considered as a solution under these conditions because it performs very poorly, however, SVO Pro can provide accurate values being one of the top algorithms in per-

formance for these datasets, this is due to its capability to estimate better rotations than pure feature-based methods.

Frame Alignment vs Trajectory Alignment

The most common method used in literature to compare trajectories is the full trajectory alignment, providing the minimal ATE error achieved by the algorithm in the analysed trajectory. To compare VO and VIO methods both alignments provide significant results and allow to extract conclusions regarding which algorithms perform better in each dataset. Nonetheless, it is very difficult to extract relevant conclusions when observing the trajectory with full alignment. Initial frame alignment provides more significant results about when the trajectory is drifting or when a loop closure happens. Additionally, we consider this alignment fairer as it represents the real drift of the MAV at each time step during navigation. All this data provides important information about the algorithm performance, such as if it drifts with rotations or high speeds, or how good is to correct the position drift when features are revisited.

Therefore, for our custom datasets, we also explored robustness by applying frame alignment to understand better the algorithm's performance and see how the results differ from both implementations.

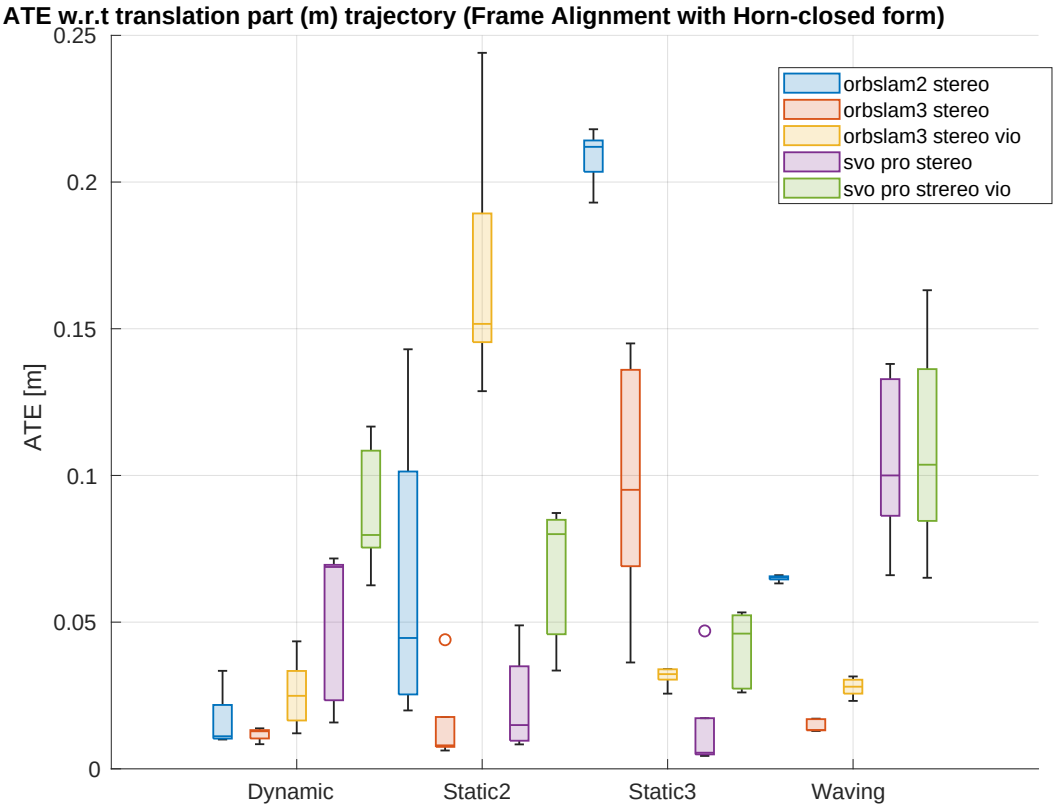
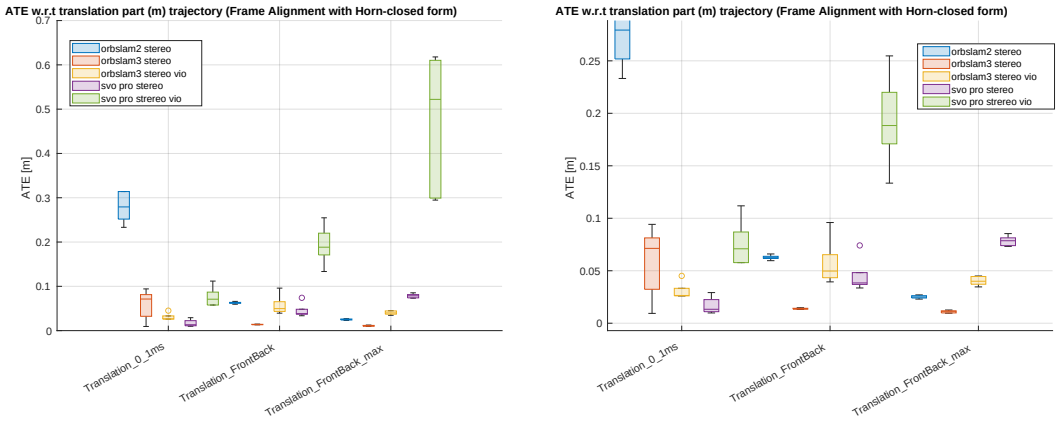


Figure 5.10: ATE runs in Custom RAM datasets Dynamic, Static2, Static3 and Waving

The boxplots Figure 5.10, Figure 5.11 and Figure 5.12 represent the runs with frame alignment. As expected, the ATE values increased compared to the ones in a full alignment, but represent trajectories more realistically. Both trajectories start with the same orientation and position and start to drift from this point. In hovering experiments, stereo implementations seem to be more robust than VIO stereo and in this case, ORB-SLAM3 stereo provides the lowest ATE values as deduced earlier. The main difference is that the standard deviation in ATE performance increases concerning full alignment. Again, it is concluded that ORB-SLAM3 commonly outperforms ORB-SLAM2 results, and SVO stereo can achieve more accurate estimations of the trajectory and SVO VIO stereo. ORB-SLAM implementations have very similar performance for

the tested datasets. However, in the square datasets, which are most challenging due to the translational speed and pure rotations, it seems the VIO implementation has better performance when increasing the speed of displacement. In these datasets, Square5_max has better performance due to the stabilization applied to the drone's robustness and due to the highest speeds.



(a) Full view. **(b)** Zoomed view.

Figure 5.11: ATE runs in Custom RAM datasets Translation_0_1ms, Translation_FrontBack,Translation_FrontBack_max

As previously commented, it seems that the big mismatch between IMU readings and stereo estimations in SVO Pro VIO stereo makes it the worst option. Figure 5.13b shows how the data is peakier and has a higher error than the stereo implementation from Figure 5.13a, where the data is smoother and the overall error is smaller.

When analysing the performance on Square4_max dataset it is concluded that peaks and higher errors are still present on Figure 5.14b. Meanwhile, the SVO Pro stereo trajectory from Figure 5.14a keeps the error low after performing the square trajectory. Feature-based algorithms have difficulties achieving rotations accurately, while SVO Pro stereo estimated better rotations because it is a semi-direct method and it tracks edges and corners. On the other hand, ORB-SLAM3 estimates translation with higher accuracy along the square as depicted in Figure 5.14c, but the error almost doubles SVO Pro stereo when performing a translation.

5.2 Maximum Error

One of the most important metrics for evaluating the maximum error achieved during navigation. This metric is crucial to guarantee that algorithms do not exceed a previously defined threshold in Section 4.3.1 named Security distance. Guaranteeing human safety during human and drone during their interaction is key to the success of this project.

The maximum error metric measures which is the maximum drift accumulated during the trajectory out of each of the 5 runs, in this case, the Full Alignment estimation does not provide realistic values. Frame alignment between the initial frames of the estimated and ground-truth trajectories from Section 4.2.2 allows us to evaluate the error in the real experiment at every time step. Therefore, we will be looking for this maximum difference between trajectories.

The results in EuRoC datasets from Table 5.6 discard the usage of the ORB-SLAM3 non-mature implementations where the security distance of 1.32 meters is exceeded. It seems ORB-SLAM3 stereo also requires some initialization seconds to achieve an accurate scale, in "V1_01_easy" dataset it exceeds the security distance but after the initialization period the maximum error dops down to 0.294 mm. Considering the front-end implementations it is seen how the algorithms exceed the maximum error allowed, so they are not valid for our application.

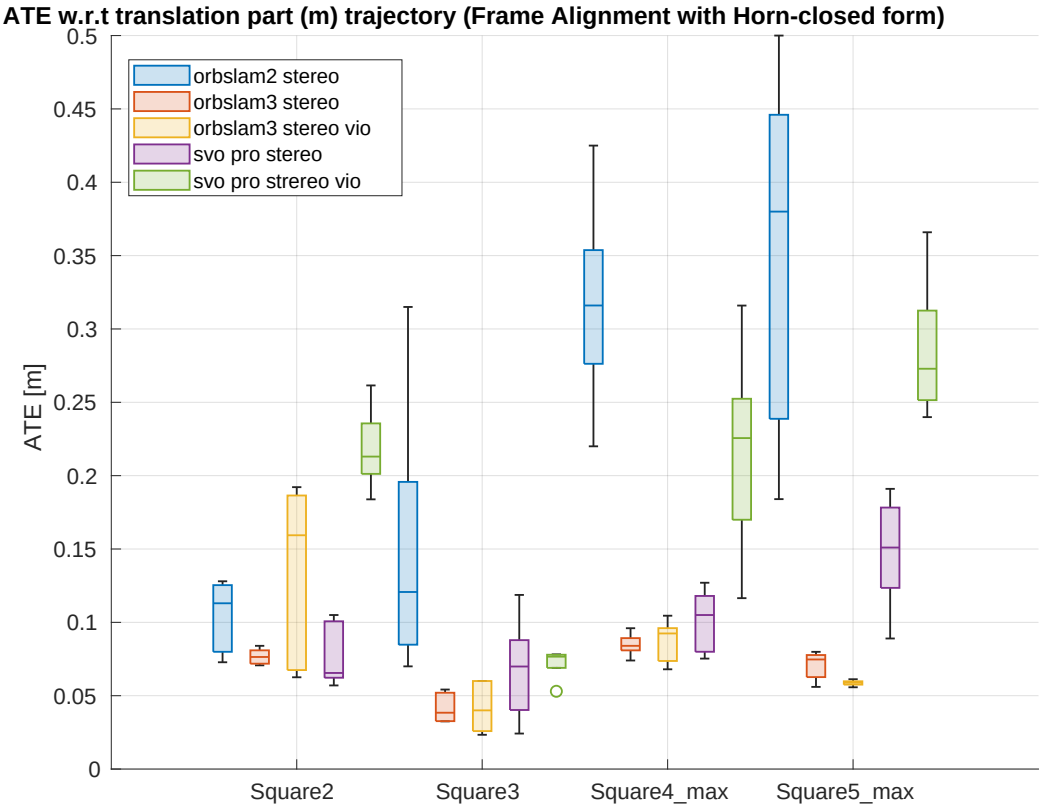
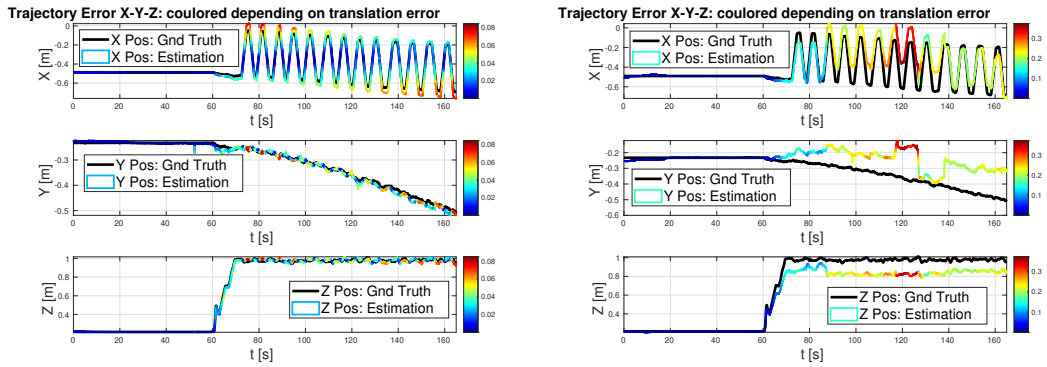


Figure 5.12: ATE runs in Custom RAM datasets Square2, Square3, Square4_max and Square5_max



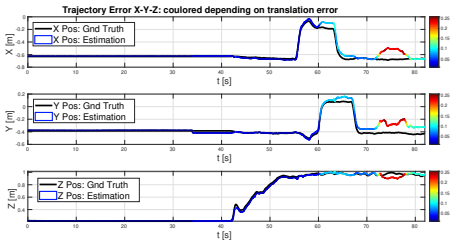
(a) SVO Pro Stereo.

(b) SVO Pro VIO stereo.

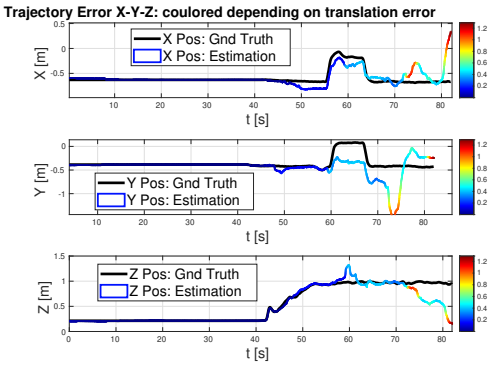
Figure 5.13: X,Y and Z displacement values for Translation_FrontBack dataset.

Table 5.7 represents the maximum error achieved by the algorithms on the datasets where the algorithm hovers in front of static and dynamic objects. In RAM datasets there were initialization problems in the scale, mainly caused by the insufficient initialization dynamics. This effect happened on VIO algorithms, therefore, to provide more realistic results the badly initialized runs were discarded in Figure 5.16b. The red line sets the maximum error it can be achieved, as none of the algorithms exceeds the line then they accomplish the requirement for these datasets.

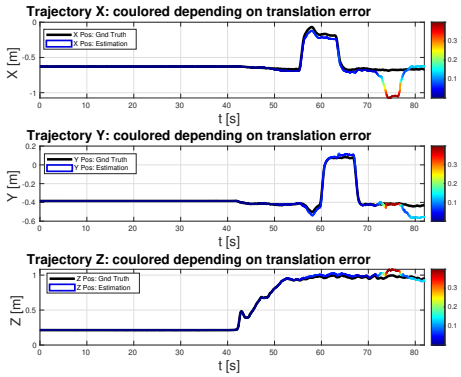
On the RAM datasets where we test translations in Z and the XY plane, SVO PRO Stereo does not comply with the security distance in "Translation_FrontBack_max" as depicted in Table 5.8. However, when looking at Figure 5.17 where Figure 5.17b shows the algorithms without outliers we can deduce that the security distance was exceeded in a run with a deficient initialization.



(a) SVO Pro Stereo.



(b) SVO Pro VIO stereo



(c) ORB-SLAM3 VIO stereo

Figure 5.14: X,Y and Z displacement values for Square4_max dataset.

ALGORITHM\DATASET	MH02_EASY	V1_01_EASY	V1_02_MEDIUM
ORB-SLAM3 VIO MONO	18.692	*3.313*	*3.32*
ORB-SLAM3 VIO MONO MATURE	0.540	*0.1821*	0.295
ORB-SLAM3 STEREO	*0.8401*	*0.1506*	1.809
ORB-SLAM3 STEREO MATURE	*0.541*	*0.594*	0.294
ORB-SLAM3 VIO STEREO	7.352	*11.95*	*7.619*
ORB-SLAM3 VIO STEREO MATURE	0.726	*0.552*	0.415
SVO PRO VIO MONO	1.915	*1.682*	*0.460*
SVO PRO VIO MONO FRONT-END	*12.169*	*4.646*	*4.645*
SVO PRO STEREO	0.745	*0.377*	*0.840*
SVO PRO VIO STEREO	*0.782*	*0.934*	*0.819*

Table 5.6: Maximum Error per algorithm in EuRoC datasets. Units: meters.

As it does not exceed the limit in Figure 5.17b then the algorithm is good to be used in similar experiments.

Finally, it is analysed the maximum error results from Table 5.9 for the datasets where a square is performed. It is deduced that ORB-SLAM2 surpasses the security distance. Also, SVO Pro VIO stereo is very close to exceeding these values and removing the outliers does not a big difference in the performance of this metric as compared in Figure 5.18a and Figure 5.18b.

To sum up, the algorithms ORB-SLAM2 stereo, SVO Pro mono and SVO Pro mono front-end are not the most convenient solution in terms of security. SVO Pro VIO stereo should be closely tracked as in some of our Custom datasets it was close to surpassing the security distance.

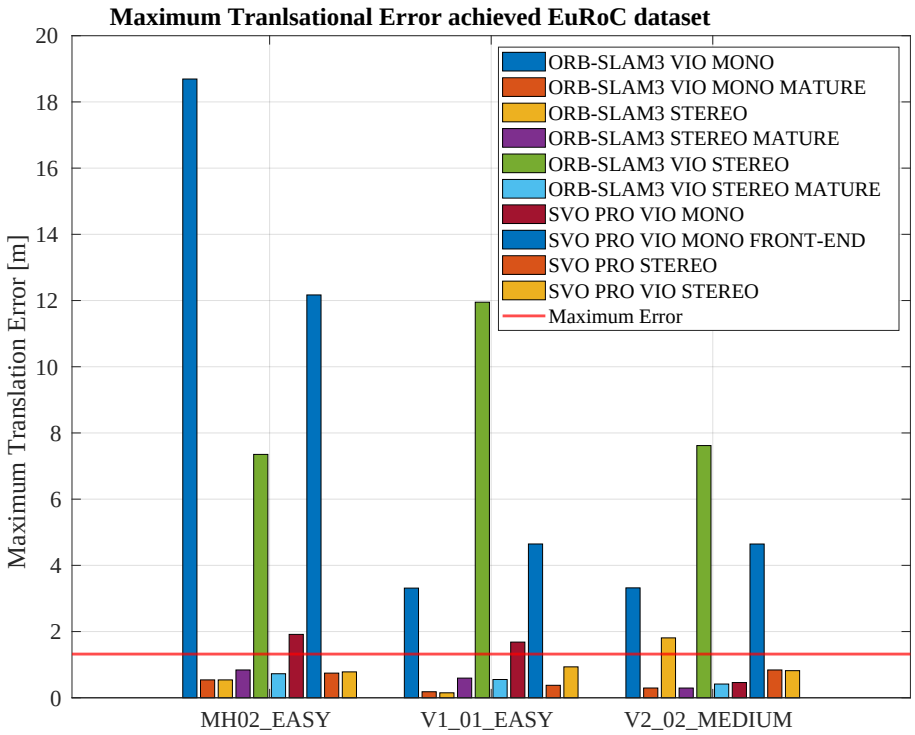


Figure 5.15: Maximum error achieved per algorithms in EuRoC datasets

DATASET\ALGORITHM	STATIC2	STATIC3	WAVING	DYNAMIC
ORB-SLAM2 STEREO	0.323	0.984	0.460	0.072
ORB-SLAM3 STEREO	0.101	0.259	0.046	0.034
ORB-SLAM3 VIO STEREO	0.729	0.519	0.792	0.578
SVO PRO STEREO	0.098	0.087	0.340	0.117
SVO PRO VIO STEREO	0.166	0.132	0.292	0.558

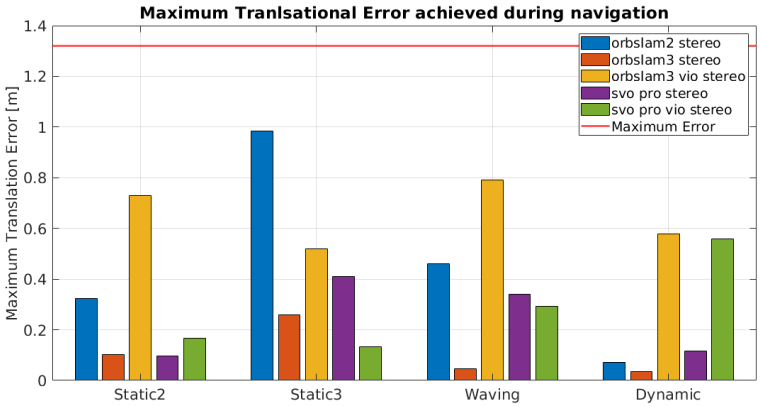
Table 5.7: Maximum error per algorithm in RAM datasets Static2, Static3, Waving and Dynamic with outliers.Units: meters.

DATASET\ALGORITHM	TRANSLATION_0_1MS	TRANSLATION_FRONTBACK	TRANSLATION_FRONTBACK_MAX
ORB-SLAM2 STEREO	0.728	0.151	0.0669
ORB-SLAM3 STEREO	0.219	0.044	0.042
ORB-SLAM3 VIO STEREO	0.127	0.454	0.072
SVO PRO STEREO	0.076	0.157	0.807
SVO PRO VIO STEREO	0.444	0.504	1.46

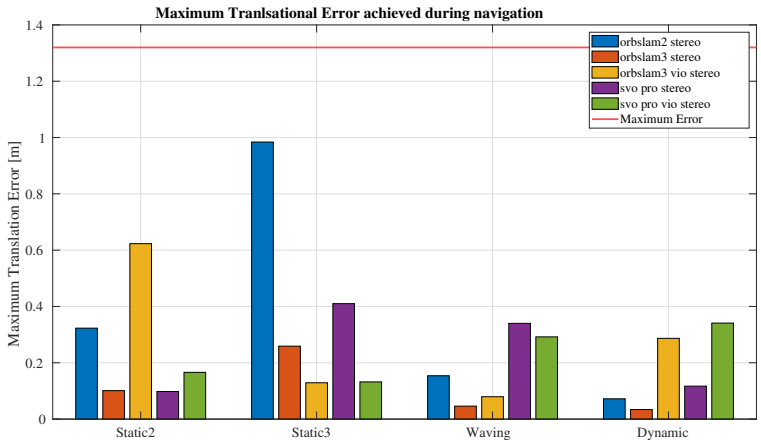
Table 5.8: Maximum error per algorithm in RAM datasets Translation_0_1ms, translation_frontback and translation_frontback_max with outliers.Units: meters.

DATASET\ALGORITHM	SQUARE2	SQUARE3	SQUARE4_MAX	SQUARE5_MAX
ORB-SLAM2 STEREO	0.482	1.194	1.136	1.34
ORB-SLAM3 STEREO	0.407	0.184	0.632	0.403
ORB-SLAM3 VIO STEREO	0.605	0.801	0.474	0.251
SVO PRO STEREO	0.265	0.410	0.300	0.541
SVO PRO VIO STEREO	0.674	1.107	1.292	1.192

Table 5.9: Maximum error per algorithm in RAM datasets Square2, Square3, Square4_max and Square5_max with outliers. Units: meters.



(a) With outliers.



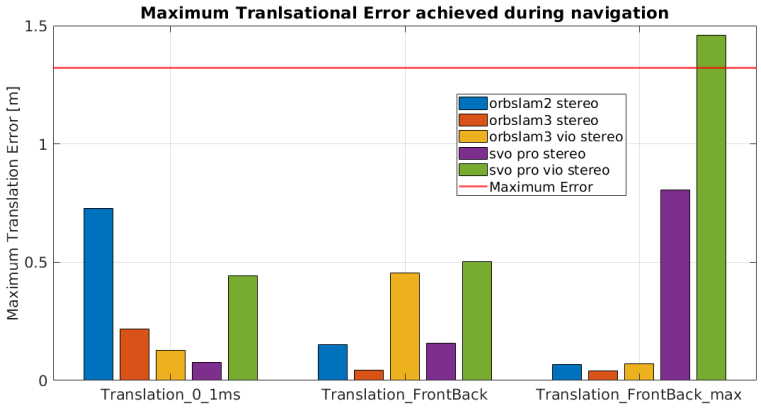
(b) Without outliers.

Figure 5.16: Maximum error per algorithm in RAM datasets Static2, Static3, Waving and Dynamic . The red line describes the security distance limit.Units: meters.

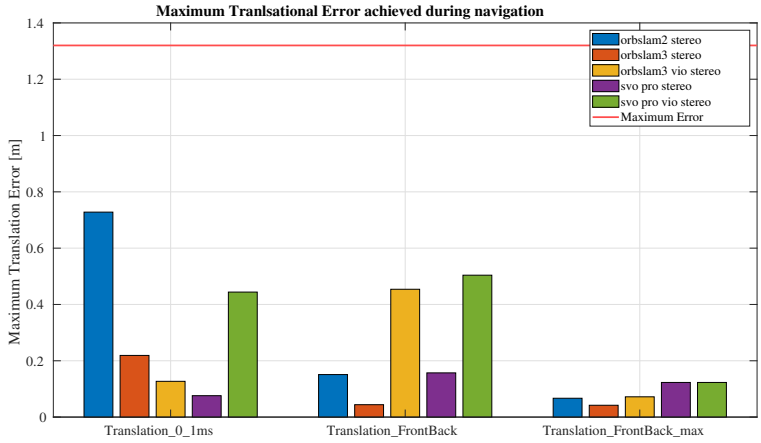
5.3 Orientation accuracy

Commonly, orientation is not used as a metric to measure the performance of the algorithms. However, it will have an impact when relying solely on VO/VIO estimations as control inputs. Therefore, ATE performance on rotation is analysed to guarantee good performance on the orientation estimations of our MAV body. ATE of the orientation is measured using angle-axis notation where the angle is expressed in radians. This representation comes from Euler's rotation theorem, which states that any rotation from a rigid body in a 3D space is equivalent to performing a pure rotation around a single fixed axis. For this metric it is chosen the frame alignment between the initial frames to guarantee the real difference between ground truth and estimation is computed at each instant of time.

The observed results on Table 5.10 show that in the hovering experiments where the rotation is just a cause of vibrations the median ATE stayed lower than 5 degrees for all the experiments. Similar results are extracted from Table 5.11 even though there are pure rotations involved in the experiments it maintains the median ATE rotation value under 6 degrees. Same conclusions are extracted from Figure 5.21, the median ate error remains under 6 degrees but for the dataset "Translation_0_1ms" for ORB-SLAM3 VIO stereo where the median ate error is around 20 degrees, probably the lack of visual features when going up and down impacted negatively these estimations.



(a) With outliers.



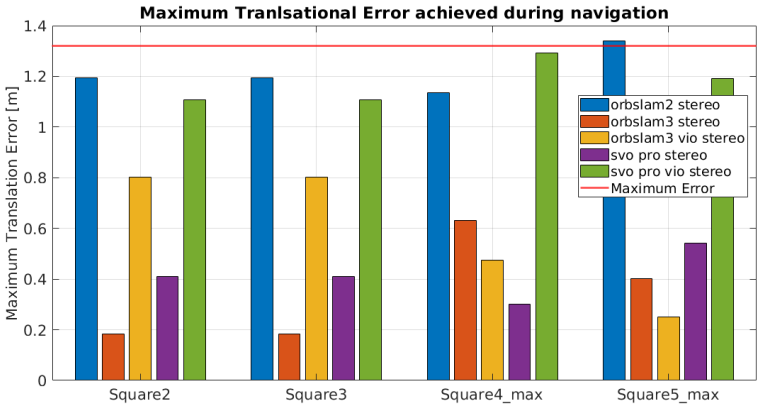
(b) Without outliers.

Figure 5.17: Maximum translation error in RAM datasets: Translation_0_1ms, Translation_FrontBack and Translation_FrontBack_max. The red line describes the security distance limit. Units: meters.

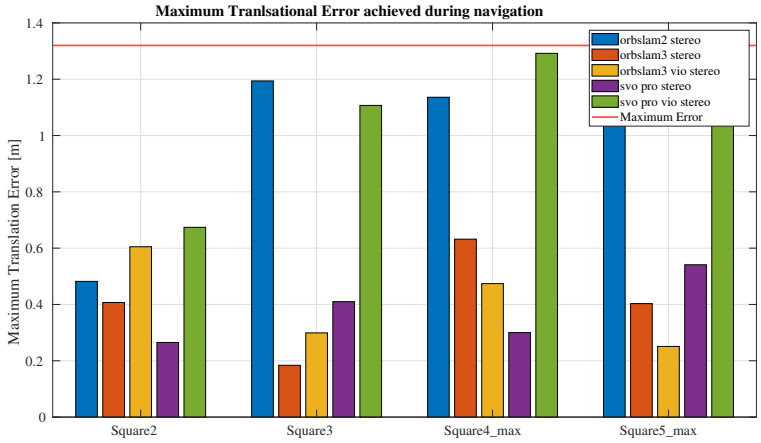
Method	Static2	Static3	Waving	Dynamic
ORB-SLAM2 STEREO	0.0104	0.012	0.026	0.0162
ORB-SLAM3 STEREO	0.0069	0.019	0.0106	0.0142
ORB-SLAM3 VIO STEREO	0.0234	0.008	0.0215	0.066244
SVO PRO STEREO	0.027	0.043	0.0846	0.061802
SVO PRO VIO STEREO	0.0094	0.0085	0.0126	0.0145

Table 5.10: Absolute Trajectory Error of the orientation in RAM datasets Static2, Static3, Waving and Dynamic. Units: radians

The maximum orientation error was also extracted from the aligned trajectories. In this case a general pattern is observed from the datasets represented on Figure 5.19, Figure 5.20 and Figure 5.21. The pattern is that ORB-SLAM2 stereo and SVO Pro stereo provide the highest orientation errors. Another relevant deduction extracted from Figure 5.20 was that when increasing the dynamics the algorithms that make use of IMU readings tend to provide more accurate estimations, this is clear with ORB-SLAM3 VIO stereo. To conclude, the values of the maximum error in the "Translation_0_1ms" dataset from Figure 5.21 provide a big rotational error. It should not be a major concern as it seems to be a punctual deviation considering that the ATE of the orientation decreases to 0.357 radians. Having an occasional big error estimation on orientation does not imply a failure in navigation if the algorithm manages to correct



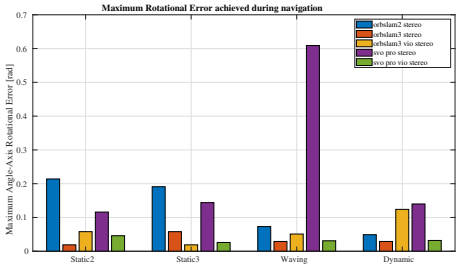
(a) With outliers.



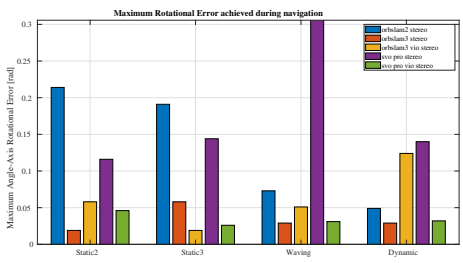
(b) Without outliers.

Figure 5.18: Maximum translation error in RAM datasets: Square2, Square3, Square4_max and Square5_max.

the orientation optimizing the poses. Also, ORB-SLAM3 VIO stereo tends to keep the orientation error low through all the datasets, this high error could be a result of the lack of tracked features from the homogeneous floor.



(a) Full View.



(b) Zoomed view.

Figure 5.19: Maximum rotation error in RAM datasets: Static2, Static3, Waving y Dynamic.

Method	Square2	Square3	Square4_max	Square5_max
ORB-SLAM2 STEREO	0.063	0.049	0.090	0.063
ORB-SLAM3 STEREO	0.0198	0.015	0.025	0.022
ORB-SLAM3 VIO STEREO	0.0143	0.015	0.015	0.013
SVO PRO STEREO	0.0615	0.044	0.044	0.063
SVO PRO VIO STEREO	0.017	0.009	0.017	0.0148

Table 5.11: Absolute Trajectory Error of the orientation in RAM datasets Square2, Square3, Square4_max and Square5_max. Units: radians

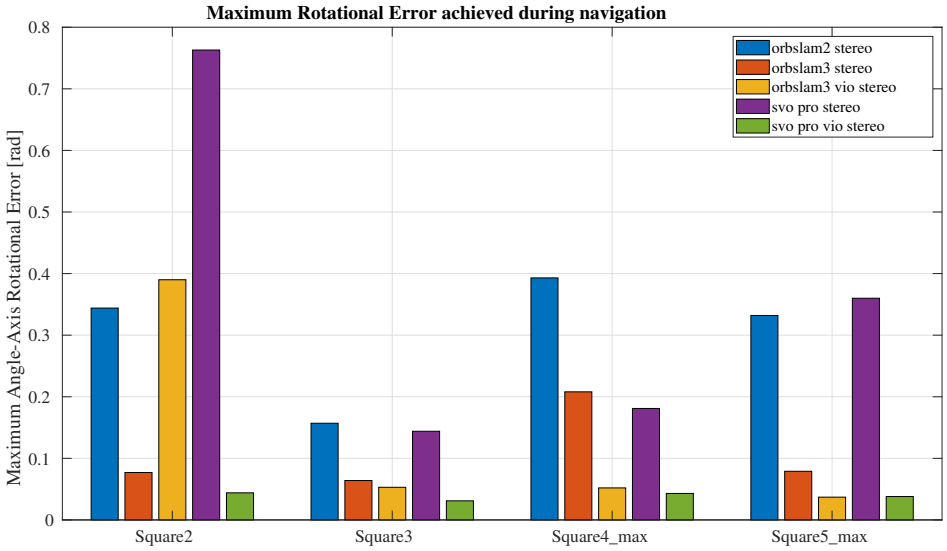
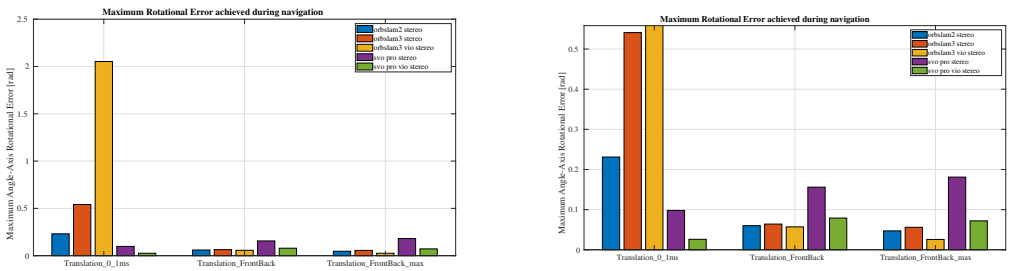


Figure 5.20: Maximum rotation error in RAM datasets: Square2, Square3, Square4_max and Square5_max.

Method	Translation_0_1ms	Translation_Front_Back	Translation_FrontBack_max
ORB-SLAM2 STEREO	0.011	0.011	0.020
ORB-SLAM3 STEREO	0.094	0.016	0.015
ORB-SLAM3 VIO STEREO	0.357	0.0117	0.09
SVO PRO STEREO	0.042	0.0838	0.0527
SVO PRO VIO STEREO	0.0102	0.0292	0.0307

Table 5.12: Absolute Trajectory Error of the orientation in each of the RAM datasets translation



(a) Full view.

(b) Zoomed view.

Figure 5.21: Maximum rotation error in RAM datasets: Translation_0_1ms, Translation_FrontBack, and Translation_FrontBack_max.

6 Discussion

The main goal of the work presented in this thesis is to answer the question:

How to enhance the level of autonomy of aerial robots developed at RAM-UT providing them with an accurate localization system to perform a handover to a human?

Our prime goal is to find localization algorithms which provide the MAV position and orientation at each time step. Open-source solutions that provided autonomous vehicles with localization are analysed and evaluated. Currently, the MAV is dependent on a MoCap system which limits MAV control to the room where this system is available. To achieve localization independently from this system we explored the odometry options available in the literature and evaluated the most convenient solutions for MAVs. The algorithms were tested under environments and trajectories similar to the ones expected on a real handover of a tool to a human.

This section answers the main research question by addressing the subquestions exposed in Section 1.2. After finding the optimal solutions for our use case we will expose our contributions to the robotics research community. Finally, in Section 6.3 the main ideas for future work that arise during the development of the project are treated.

6.1 Findings

1. ***Which are the hardware options to provide vehicles with localization in GPS-denied environments? And which is the most convenient option for MAVs?***

   Multiple odometry options have been exploited to provide vehicles with localization capabilities in GPS-denies environments: Wheel Odometry (WO), Inertial Odometry (IO), Radar Odometry (RO), Laser Odometry (LO), Visual Odometry (VO) and even the combination of multiple of this techniques.

   In our use case, we will apply these algorithms on a MAV to perform a handover of a tool to a human. The handover will be performed in a simple environment with few obstacles considering both indoor and outdoor scenarios.

   Micro aerial vehicles are resource-constrain platforms with restricted size and weight, which makes the design of sensory systems challenging.

   - **Wheel Odometry (WO)** cannot be considered in MAV as they are not UGV.
   - **Inertial Odometry (IO)** provides linear velocity, accelerations, position and orientation having little power consumption. The main drawback is its drifting errors and bias on the accelerometer and gyroscope measurements, therefore, it is not suitable for lifelong experiments.
   - **Radar Odometry (RO)** can retrieve the relative motion of the MAV and the position of the surrounding objects using radar sensors. It uses low power consumption, it is suitable for low textures and unfavourable environmental conditions. Yet, it is affected by inaccuracies of radar's Doppler measurements, especially while flying over non-flat terrain. Additionally, it requires high storage when performing large-scale mapping making it inconvenient for resource-constraint platforms.

- **Laser Odometry (LO)** uses light to accurately capture the position of the surrounding objects and uses these measurements to estimate the position and orientation of the MAV. It is not distressed either by the low textures or the environmental light but, struggles in presence of dust and bad weather. It is a computationally expensive algorithm and is affected by dynamic objects, which distort the scans. As well as RO, it grants accurate structure information in the lack of texture environments.
- **Visual Odometry (VO)** uses sequences of images to retrieve pose and orientation. It is a lightweight implementation, which is frequently used in MAV navigation systems as a standalone algorithm. It is considered moderately accurate and inexpensive. Nonetheless, localization estimations can be affected by illumination changes, weather conditions or lack of features of the surrounding scene.

The handover will be performed in a context where the number of obstacles in the scene is low. The MAV will navigate in a textured environment where there is a sufficient number of static features to perform visual navigation indoors and outdoors. Moreover, the goal is to achieve accurate localization to safely deliver a tool to a human but, simultaneously use the least amount of computational power. This way, the free computational resources could be used for other tasks, for instance, gesture recognition using ML which uses high computational power. Under these circumstances, VO seems to be the preferred solution that could be supported with inertial measurements if required, as adding these measurements will not suppose a high increase of the computational power.

2. ***Which are the most promising open-source algorithms to provide the MAV with localization capabilities (position and orientation)?***

   Initially an evaluation was performed of the most promising VO/VIO algorithms. The most recent papers were evaluated and the results were exposed in Table 3.2. From here it was deduced that the most promising open-source solutions were:

   (a) 1^{st} - ORB-SLAM2
   (b) 2^{nd} - SVO
   (c) 3^{rd} - ORB-SLAM
   (d) 4^{th} - VINS-Mono
   (e) 5^{th} - OKVIS

   Therefore, ORB-SLAM2 was selected as the algorithm to be tested against the EuRoC dataset and our custom dataset. In the initial stage of the evaluation, we found two encouraging algorithms that were public, ORB-SLAM3 and SVO Pro. ORB-SLAM3 was the improvement of our ORB-SLAM2, our top 1 algorithm, and SVO Pro was the improved version of SVO as it included back-end optimizations and versions of the algorithms which included IMU readings. Consequently, these three algorithms were tested to evaluate the performance from our side.

ALGORITHM	HARDWARE REQUIREMENTS	APPROACH	INPUT TREATMENT	LOOP CLOSURES	MAP	RELEASE
ORB-SLAM2	Stereo/RGB-D	Optimization	Indirect	Yes	Sparse	Oct 11,2017
ORB-SLAM3	Monocular/Stereo + IMU	Optimization	Indirect	Yes	Sparse	v0.4 beta Apr. 21,2021
SVO Pro	Monocular/Stereo + IMU	Optimization	Semi-Direct	Yes	Sparse	Oct. 21, 2021

3. ***How different are the conditions in outdoor scenarios? Can the same framework be seamlessly used? How do changes in light affect the estimation?***

The main differences between performing localization indoors and outdoors are related to the weather and illumination conditions. Visual odometry approaches fail under low illumination conditions, as they are impeded to perform feature extraction or photometric matching due to the lack of texture in images. Moreover, under rainy conditions, the images suffer from the rain vailing effect that can reduce the visibility range of the camera, as well as raindrops that can adhere to the lenses and degrade the quality of the images.

Another major factor when performing outdoor navigation is that normally the distance from the camera to the objects increases. RGB-d cameras have a limited range to estimate depth. For instance, our camera INTEL D435i has a range between 0.2 and m-10 m so these types of cameras have limitations when working outdoors.

4. ***Should the solutions integrating Machine Learning be considered as optimal localization method for our use-case?***

   ML implementations cannot outperform the top conventional algorithms yet. There are hybrid ML algorithms such as D3VO which defeat several VO/VIO alternatives such as DSO and ORB-SLAM. ML approaches seem a very convenient solution to overcome certain limitations of conventional algorithms, such as retrieving scale from monocular cameras or being able to handle information from multiple sensors to estimate the pose. The big efforts that the research community is doing and the growing computational power make us consider them a promising alternative in the not too distant future.

5. ***In a simplified scenario without too many obstacles, is it necessary to have loop closures as a feature in the VIO/SLAM pipeline?***

   Yes, loop closures have been proven to be a fundamental key to achieving accurate localization and reducing the accumulated drift during navigation. It provides the algorithm with the capability of doing relocalization and recognizing features from the past. This allows performing global bundle optimization which highly increases accuracy and allows the algorithms to comply with safety constraints. This was proven when applying the front-end option of the SVO Pro VIO Mono algorithm. Without loop closures, the maximum error achieved during the runs never complied with the 1.35m security distance. Furthermore, on the dataset V1_02_medium the algorithm failed during localization on half of the runs and as loop closure was disabled it was never possible to relocalize itself.

6. ***Is the sensorial suite composed of onboard camera + IMU (performing online VIO-SLAM) enough to perform an indoor handover of an object from a drone to a human?***

   A human can stretch a mean of 1.32m, considering his mobility is limited to half a step. Supposing the MAV counts with a robotic arm that stretches up to 30 cm being this an extra security distance where the value 1.32 should not be exceeded 1.32 m will be established as the security distance between humans and drones. An effective and secure localization occurs when the maximum error during the complete trajectory does not exceed this 1.32m.

   In Section 5.2 the results of the maximum error achieved during navigation are exposed. The trajectory and ground truth trajectories are compared point to point and the maximum error achieved out of the 5 runs per algorithm and dataset is extracted. Results prove that SVO Pro VIO versions and ORB-SLAM3 VIO versions that include the initial

dynamics, exceed the security distance in EuRoC dataset as visualized in Figure 5.15. When checking the same metric on our custom datasets from RAM we see that ORB-SLAM2 Stereo and SVO Pro VIO Stereo do not comply with the security distance either. This leaves us with ORB-SLAM3 stereo, ORB-SLAM3 VIO stereo and SVO Pro stereo as valid options to perform localization under the tested conditions, where the final goal is performing the handover of an object to a human.

7. ***Is one camera enough to perform VIO/SLAM while also recognizing the man and his/her gestures?***

   To detect human gestures at least one camera should be pointing toward the human. To perform an effective localization using VO/VIO a pair of stereo cameras are required, these cameras can be located at any position where the camera can track static features. If we put a pair of stereo cameras at the front, it is possible to reuse the input images for the cameras both for gesture recognition using ML and to track features for effective localization. Considering the real conditions are similar to the recorded dataset, where high dynamic objects appear on the images and there are enough static features to be tracked, we can assure that only with this pair of stereo cameras we would be able to accomplish both tasks.

8. ***How should the camera be placed on the MAV? What is the optimal placement? Is the presence of 3DOF constraining the placement?***

   The stereo camera pairs used for localization should be used in the orientation where we would expect more static features to be tracked and where the images are more likely to have a higher degree of contrast as features will be able to be detected more easily. For instance, a surveillance drone that files at 160 km high should have the cameras on the bottom to be able to track features from the floor. The drone will not be able to find many relevant features if the cameras are located at the front as clouds move at different speeds and tracking cloud features will provide poor localization. In the datasets that were recorded in RAM Arena, it would have been more optimal to record the data with the camera tilted a bit more upward. This orientation would have been more convenient because the mattress from where the drone takes off is textureless and the descriptors are not able to extract features from it. Moreover, in experiments where textureless environments were provided the algorithms were not even able to get initialized due to not being able to track enough features.

   The 3 DoF arm impacts negatively the localization algorithm. If the arm is in the field of view, it will detect features from the arm. These features will be always tracked in the same static position and will remain static in the same position during all the navigation as the camera and the arm are part of the same rigid body. These features will increase the localization error as the algorithm will be tracking on the one hand static features of the environment that change position from frame to frame (used for camera localization) and on the other hand the static features of the drone that wrongly indicate the algorithm that the camera is not moving. Two types of features should be avoided within the captured image to compute localization:

   - Features that are part of a dynamic object, for example, a hand that is moving in front of a camera.
   - Features that are part of the rigid body.

From the extracted results and the analysis of the data, it was mentioned that ORB-SLAM3 stereo, ORB-SLAM3 VIO stereo, SVO Pro stereo and SVO Pro VIO stereo were the algorithms that complied with the maximum error allowed in position. Overall, each algorithm has its pros and cons. As a general comparison of these three VO/VIO implementations it can be said:

- In EuRoC datasets ORB-SLAM3 VIO stereo proved the algorithm with the highest robustness and lowest ATE in position, keeping this error under 0.15 meters for all datasets and outperforming the stereo implementation in challenging scenarios (V1_02_medium). Its implementation without IMU showed also a good ATE median value but the lowest robustness when facing challenging trajectories. SVO Pro stereo had a bit worst results in ATE and robustness. EuRoC datasets are taken as reference for the expected results when performing lifelong experiments with our MAV, as these experiments are longer than the ones recorded in RAM datasets.

- Considering the custom RAM datasets we deduced that the VIO implementations are less sensitive to faulty camera calibrations. ORB-SLAM3 presented the best results when facing challenging datasets, where the error decreased when increasing the displacement speed, then we deduce that the IMU readings provided better estimations under his situations. Contrary, this algorithm has the worst ATE position performance when just hovering compared to its stereo implementations. SVO Pro showed the worst performance in all the datasets,however, they proved better estimations than ORB-SLAM3 when conducting pure rotations.

- ORB-SLAM3 algorithms provided lower ATE in orientation than SVO Pro. ORB-SLAM3 stereo provides better estimations with low dynamics and when including the IMU measurements the orientation estimations improved.

As explained in previous sections, SVO Pro is affected by the infrared lasers projected by the OptiTrack and we expect better performance when using conventional cameras to capture images. Overall, the preferred solution in terms of performance for both position and orientation is ORB-SLAM3 VIO stereo, however, an alternative for navigation should be considered until the map is mature.

6.2 Contributions

This section presents how this project contributes to the scope of autonomous navigation in robotics.

- Exposition and comparison of the open-source VO/VIO conventional algorithms.

- Evaluating the maximum error achieved by the most promising VO/VIO algorithms to guarantee safe co-working human and drone. The algorithms were tested under similar visual inputs and trajectories to the ones performed in a real handover of a tool to a human.

- Comparing SVO Pro, ORB-SLAM2 and ORB-SLAM3 against broadly extended EuRoC MAV datasets and our custom RAM datasets which perform trajectories similar to the ones accomplished in a real handover of a tool. The comparison uses both position and rotation ATE and evaluates the robustness of the VINS algorithms.

- Understanding ATE limitations as a result of relocalization. The main drawback of using this metric is that the relocalization periods are not considered within the ATE equation. Nevertheless, this limitation does not have a big impact on our results because only overall only the SVO Pro VIO mono algorithms are required to relocalize themselves on these datasets. This was proven through visual inspection when running the algorithms.

- Extracting results both from frame alignment and full alignment of the trajectories and understanding their utility.

6.3 Future Research

The experiments conducted were done with the default configurations from the algorithms, as previously mentioned, the way the algorithms come adjusted by default can provide decently to other environments. The datasets were all recorded indoors, the way we simulated outdoor environments was by choosing datasets where features were far from the camera. However, multiple improvements can be tested with the chosen algorithms:

- Test the localization algorithm against the RAM controller. Instead of using the OptiTrack measurements, it is now possible to use VO/VIO localization methods.
- The computational load can be tested together with the machine learning algorithm for feature recognition to understand the computational power required.
- The camera resolution used with the RAM dataset was similar to the one used in the EuRoC dataset. The camera resolution can be increased allowing to achieve better performance for feature extraction. This should have a positive impact on localization accuracy.
- Ground truth data can be recorded using a GPS device to build outdoor datasets and check the performance.
- Change infrared cameras for conventional ones to test the real performance of SVO Pro algorithms, they seem to be an interesting option if pure rotations are going to be frequent during navigation.
- Fisheye cameras can be an alternative to smaller FoV cameras, it is worth having them in mind although our FoV camera choice seems to be more appropriate. As pointed out in (102), fisheye cameras are preferred in small and confined environments, while a smaller field of view cameras such are the ones used here are suitable for large scale scenarios.
- Exploring solutions to remove dynamic features from the scene and the features captured which are part of the drone body. These features decrease the accuracy of the estimations.
- The biggest pain when using VO/VIO in our workflow are the faulty estimations when initializing the scale. Consider other alternatives when starting navigation until the map is mature and then switching back to VO/VIO.

A Appendix

A.1 IMU initialization in ORB-SLAM3

The goal of the novel IMU initialization method from ORB-SLAM3 (90) is to provide good initial values for IMU variables: body velocities, gravity direction and IMU biases. The problem is solved as a MAP estimation problem split into three steps:

1. **Vision-only MAP Estimation**: the initialization is done as in a pure Monocular SLAM implementation. The initial 2 seconds 10 frames and hundreds of points are stored, from this data, we retrieve the scale through BA optimization. The first frame coordinates and axis are used as a reference for the rest of the poses. From here we obtain an up-to scale trajectory $\overline{T}_{0:k} = \left[R, \overline{p}\right]_{0:k}$.

2. **Inertial-only MAP estimation**: the next set is to obtain optimal estimations for the inertial variables represented through the state inertial vector. The vector is computed using the poses from $\overline{T}_{0:k}$ and inertia measurements between keyframes.

$$Y_k = \left\{s, \mathbf{R}_{wg}, b, \overline{v}_{0:k}\right\}$$

The variables of the inertial state vector are: $s \in R^+$ is the scale factor from step 1, $\mathbf{R}_{wg} \in SO(3)$ is the rotation matrix to compute gravity vector $\mathbf{g}$ in the world frame as $\mathbf{g} = \mathbf{R}_{wg}\mathbf{g_I}$, being $\mathbf{g_I} = (0,0,G)^T$ and G the gravity magnitude. The biases of the accelerometer and the gyroscope are represented by $b = (b^a, b^g) \in R^6$ assumed as constants during the initialization and finally $\overline{v}_{0:k} \in R^3$ are the body velocities expressed on the body scale and initally estimated from $\overline{T}_{0:k}$. At this point we will only consider the inertial measurements $I_{0:k} = \left\{I_{0,1}...I_{k-1,k}\right\}$. From this parameters we can deduce the following minimization problem:

$$p(y_k|I_{0:k}) \propto p(I_{0:k}|y_k)p(y_k) \tag{A.1}$$

$$y_k^* = \arg\max_{y_k}\left(p(y_k)\prod_{i=1}^{k} p(I_{i-1}, i|s, \mathbf{R_{wg}}, b, \overline{v}_{i-1}, \overline{v}_i)\right) \tag{A.2}$$

$$y_k^* = \arg\min_{y_k}\left(||b||^2_{\sum_b^{-1}} \sum_{i=1}^{k} ||r_{I_{i-1,i}}||^2_{\sum_{I_{i-1,i}}^{-1}}\right) \tag{A.3}$$

The covariance matrix $\sum_b^{-1}$ represents the prior knowledge about the scope of the IMU biases. To optimize the direction of gravity we will parametrize the the optimization equation with two angles as the gravity is aligned with the z axis:

$$\mathbf{R_{wg}^{new}} = \mathbf{R_{wg}^{old}}\mathbf{Exp}(\delta\alpha_\mathbf{g}, \delta\beta_\mathbf{g}, \mathbf{0}) \tag{A.4}$$

In the equation above $Exp(.)$ stands for exponential map. TO preserve the positive value of the scale we will define its optimization function as:

$$s^{new} = s^{old}Exp(\delta_s) \tag{A.5}$$

When the inertial-only optimization finishes, the poses and velocities of the frame and the 3D map points will be in the correct scale and rotated to align the z-axis with the gravity. The biases will be updated and it will keep computing IMU preintegration to reduce linearization errors.

3. **Visual-Inertial MAP Estimation:** once all the initial parameters are computed we can combine both visual and inertial data to refine the values, but the biases are kept constant for all keyframes.

In the case the movements are too slow during the initialization phase may not provide optimal observability and fail to solve the optimization problem in the first 15 seconds. Therefore, a robust technique has been developed for this situation, in this case only scale and gravity are estimated during the inertial-only optimization. In this case, there will be no fixed biases, these IMU parameters will be estimated from the mapping thread. This optimization will take place until 100 keyframes are stored or 75 seconds have passed since the initialization, providing the MAP estimation with more data to achieve accurate results. This IMu initialization process can easily be extended to stereo-inertial by fixing the scale to 1 on the inertial-only optimization.

A.2 IMU Preintegration and Initialization

An IMU allows us to measure acceleration and rotation rate at a fixed frequency. These measurements are adopted into the factor graph-based V-SLAM framework through the IMU *preintegration theory* (53). The goal is to estimate the motion of the system from the IMU measurements. Broadly speaking, IMU preintegration can be employed as a measurement between two successive frames allowing us to define an estimation model and an error function. The IMU discrete positions P_b, rotations R_b and velocities V_b can be modelled through the following equations (53):

$$P^{t+\triangle t} = P_b^t + V_b^t \triangle t + \frac{1}{2} g_w \triangle t^2 + \frac{1}{2} R_b^t (a_b^t - b_a^t - \eta^{ad}) \triangle t \tag{A.6}$$

$$R_b^{t+\triangle t} = R_b^t Exp\left(\left(w_b^t - b_g^t - \eta^{gd}(t)\right) \triangle t\right) \tag{A.7}$$

$$V_b^{t+\triangle t} = V_b^t + g_w \triangle t + R_b^t (a_b^t - b_a^t - \eta^{ad}(t)) \triangle t \tag{A.8}$$

where b is the body frame and W is the world frame. The term g_w represents gravity, a_b^t is the acceleration of the IMU and w_b^t is its angular velocity at time t. The biases of the acceleration and the gyroscope are b_a and b_w respectively. The discrete time Gaussian noise of the gyroscope corresponds to η^{gd} and for the accelerometer is η^{ad}.

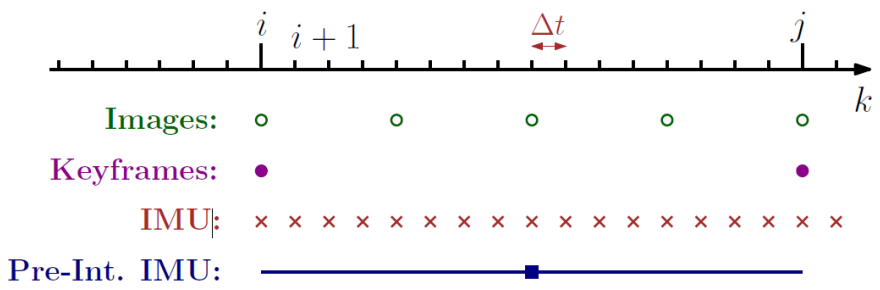


Figure A.1: Different rates for IMU and camera. The blue line represent the complete interval where IMU preintegrations take place. (53)

All the IMU measurements between two camera keyframes at times i and j can be simplified into a single synthetic formula, named the *preintegrated IMU measurement*. This formula constraints motion between consecutive keyframes relying on the assumption that the IMU is synchronized with the camera.

$$p_j = p_i + \sum_{k=i}^{j-1} \left[v_k \triangle t + \frac{1}{2} g \triangle t^2 + \frac{1}{2} R_k (a_k - b_k^a - \eta_k^{ad}) \triangle t^2 \right] \tag{A.9}$$

$$R_j = R_i \prod_{k=i}^{j-1} Exp\left(\left(w_k - b_k^g - \eta_k^{gd} \right) \right) \triangle t \tag{A.10}$$

$$v_j = v_i + g \triangle t_{ij} + \sum_{k=i}^{j-1} R_k (a_k - b\hat{a}_k - \eta_k^{ad}) \triangle t \tag{A.11}$$

Based on the IMU Gaussian noise model, the previous expresions of position, velocity and orientation evolve to elimiate the dependency on biases and the time dependency of velocity and position (otherwise the integrations need to be rerun when the linearized point t_i changes). Combining this enhancements we reach the *preintegrated measurement model* defining the relative changes between successive frames:

$$\triangle \tilde{R}_{ij} = R_i^T R_j Exp\left(\phi \sigma_{ij} \right) \tag{A.12}$$

$$\triangle \tilde{v}_{ij} = R_i^T \left(v_j - v_i - g \triangle t_{ij} \right) + \delta v_{ij} \tag{A.13}$$

$$\triangle \tilde{p}_{ij} = R_i^T \left(p_j - p_i - v_i \triangle t_{ij} - \frac{1}{2} g \triangle t_{ij}^2 \right) + \delta p_{ij} \tag{A.14}$$

Where random noise is described by the random vector $\left[\delta \phi_{ij}^T, \delta v_{ij}^T, \delta p_{ij}^T \right]^T$.

A.3 Machine Learning algorithms vs Conventional algorithms.

Figure A.3 represents the results of the ORB-SLAM2 stereo (72) algorithm on the KITTI datasets, where stereo images are recorded from a car navigation. The values from column t_{rel} correspond to the average relative translation error in position and are used as a benchmark against the machine learning algorithms from Figure A.2. The ML algorithms are tested against the KITTI datasets 9 and 10 and the values represented on the table correspond to the t_{rel}. It is concluded that D3VO is the only algorithm capable of outperforming ORB-SLAM2 in the current state of the art, providing a t_{rel} 0.09% lower for sequence 9, however, this algorithm could not be tested from our side because the algorithm was not opensource.

A.4 Complete results of ATE in position for RAM datasets.

Results visualized in Table A.1.

	Model	Sensor	Supervision	Scale	Performance Seq09	Performance Seq10	Contributions
VO	Konda et al. [44]	MC	Supervised	Yes	-	-	formulate VO as a classification problem
	Costante et al. [45]	MC	Supervised	Yes	6.75	21.23	extract features from optical flow for VO estimates
	Backprop KF [51]	MC	Hybrid	Yes	-	-	a differentiable Kalman filter based VO
	DeepVO [24]	MC	Supervised	Yes	-	8.11	combine RNN and ConvNet for end-to-end learning
	SfmLearner [29]	MC	Unsupervised	No	17.84	37.91	novel view synthesis for self-supervised learning
	Yin et al. [52]	MC	Hybrid	Yes	4.14	1.70	introduce learned depth to recover scale metric
	UnDeepVO [53]	SC	Unsupervised	Yes	7.01	10.63	use fixed stereo line to recover scale metric
	Barnes et al. [54]	MC	Hybrid	Yes	-	-	integrate learned depth and ephemeral masks
	GeoNet [55]	MC	Unsupervised	No	43.76	35.6	geometric consistency loss and 2D flow generator
	Zhan et al. [56]	SC	Unsupervised	No	11.92	12.45	use fixed stereo line for scale recovery
	DPF [57]	MC	Hybrid	Yes	-	-	a differentiable particle filter based VO
	Yang et al. [58]	MC	Hybrid	Yes	0.83	0.74	use learned depth into classical VO
	Zhao et al. [59]	MC	Supervised	Yes	-	4.38	generate dense 3D flow for VO and mapping
	Struct2Depth [60]	MC	Unsupervised	No	10.2	28.9	introduce 3D geometry structure during learning
	Saputra et al. [48]	MC	Supervised	Yes	-	8.29	curriculum learning and geometric loss constraints
	GANVO [61]	MC	Unsupervised	No	-	-	adversarial learning to generate depth
	CNN-SVO [62]	MC	Hybrid	Yes	10.69	4.84	use learned depth to initialize SVO
	Xue et al. [50]	MC	Supervised	Yes	-	3.47	memory and refinement module
	Wang et al. [63]	MC	Unsupervised	Yes	9.30	7.21	integrate RNN and flow consistency constraint
	Li et al. [64]	MC	Unsupervised	No	-	-	global optimization for pose graph
	Saputra et al. [49]	MC	Supervised	Yes	-	-	knowledge distilling to compress deep VO model
	Gordon [65]	MC	Unsupervised	No	2.7	6.8	camera matrix learning
	Koumis et al. [66]	MC	Supervised	Yes	-	-	3D convolutional networks
	Bian et al. [30]	MC	Unsupervised	Yes	11.2	10.1	scale recovery from only monocular images
	Zhan et al. [67]	MC	Hybrid	Yes	2.61	2.29	integrate learned optical flow and depth
	D3VO [25]	MC	Hybrid	Yes	0.78	0.62	integrate learned depth, uncertainty and pose
VIO	VINet [68]	MC+I	Supervised	Yes	-	-	formulate VIO as a sequential learning problem
	VIOLearner [69]	MC+I	Unsupervised	Yes	1.51	2.04	online correction module
	Chen et al. [70]	MC+I	Supervised	Yes	-	-	feature selection for deep sensor fusion
	DeepVIO [71]	SC+I	Unsupervised	Yes	0.85	1.03	learn VIO from stereo images and IMU
LO	Velas et al. [72]	L	Supervised	Yes	4.94	3.27	ConvNet to estimate odometry from point clouds
	LO-Net [73]	L	Supervised	Yes	1.37	1.80	geometric constraint loss
	DeepPCO [74]	L	Supervised	Yes	-	-	parallel neural network
	Valente et al. [75]	MC+L	Supervised	Yes	-	7.60	sensor fusion for LIDAR and camara

- *Model:* VO, VIO and LO represent visual odometry, visual-inertial odometry and LIDAR odometry respectively.
- *Sensor:* MC, SC, I and L represent monocular camera, stereo camera, inertial measurement unit, and LIDAR respectively.
- *Supervision* represents whether this work is a purely neural network based model trained with groundtruth labels (Supervised) or without labels (Unsupervised), or it is a combination of classical and deep neural network (Hybrid)
- *Scale* indicates whether a trajectory with a global scale can be produced.
- *Performance* reports the localization error (a small number is better), i.e. the averaged translational RMSE drift (%) on lengths of 100m-800m on the KITTI odometry dataset [46]. Most works were evaluated on the Sequence 09 and 10, and thus we took the results on these two sequences from their original papers for a performance comparison. Note that the training sets may be different in each work.
- *Contributions* summarize the main contributions of each work compared with previous research.

Figure A.2: General overview of the Machine Learning algorithms available and the average relative translation error in sequences 9 and 10 of KITTI datasets.(60)

	ORB-SLAM2 (stereo)			Stereo LSD-SLAM		
Sequence	t_{rel} (%)	r_{rel} (deg/100m)	t_{abs} (m)	t_{rel} (%)	r_{rel} (deg/100m)	t_{abs} (m)
00	0.70	**0.25**	1.3	**0.63**	0.26	**1.0**
01	**1.39**	**0.21**	10.4	2.36	0.36	**9.0**
02	**0.76**	0.23	5.7	0.79	0.23	**2.6**
03	**0.71**	**0.18**	**0.6**	1.01	0.28	1.2
04	0.48	**0.13**	0.2	**0.38**	0.31	0.2
05	**0.40**	**0.16**	**0.8**	0.64	0.18	1.5
06	**0.51**	**0.15**	**0.8**	0.71	0.18	1.3
07	**0.50**	**0.28**	0.5	0.56	0.29	0.5
08	**1.05**	0.32	**3.6**	1.11	**0.31**	3.9
09	**0.87**	0.27	**3.2**	1.14	**0.25**	5.6
10	**0.60**	**0.27**	**1.0**	0.72	0.33	1.5

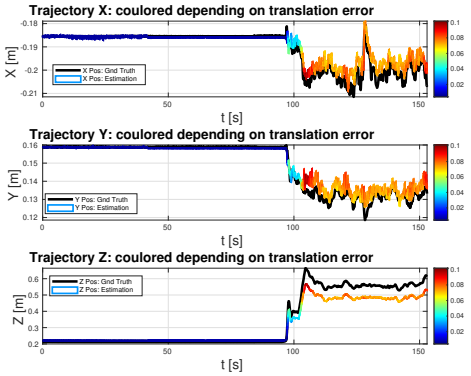
Figure A.3: ORB-SLAM2 stereo performance on KITTI datasets.(72)

DATASET\ALGORITHM	STATIC2	STATIC3	TRANSLATION_0_1MS	WAVING	DYNAMIC	SQUARE2	SQUARE3	SQUARE4_MAX	SQUARE5_MAX	TRANSLATION_FRONTBACK	TRANSLATION_FRONTBACK_MAX
ORB-SLAM2 STEREO	0.117	0.167	0.199	0.234	0.262	0.272	0.220	0.284	0.271	0.283	0.271
SCALED	0.004 (250%)	0.022 (250%)	0.024 (240%)	0.041 (220%)	0.008 (250%)	0.089 (190%)	0.038 (130%)	0.087 (120%)	0.075 (130%)	0.041 (220%)	0.024 (220%)
ORB-SLAM3 STEREO	0.195	0.274	0.355	0.405	0.435	0.548	0.466	0.551	0.505	0.506	0.491
SCALED	0.0045 (-55%)	0.0079 (-55%)	0.0065 (-55%)	0.014 (-55%)	0.011 (-55%)	0.0714 (-60%)	0.0496 (-55%)	0.0572 (-55%)	0.065 (-55%)	0.0105 (-55%)	0.0084 (-55%)
ORB-SLAM3 VIO STEREO	0.10798	0.016703	0.009871	0.02468	0.014988	0.10054	0.0277	0.087831	0.059075	0.031113	0.32952
SCALED	0.10798 (0%)	0.0139 (4%)	0.009871 (0%)	0.009 (+3%)	0.0167 (+5%)	0.0663 (+25%)	0.018162 (+5%)	0.08785 (+1)	0.0562 (+5%)	0.009871 (0%)	0.0167 (7%)
SVO PRO STEREO	0.067195	0.098724	0.099954	0.1409	0.12084	0.13861	0.12588	0.11989	0.10247	0.13583	0.12853
SCALED	0.0065 (70%)	0.005 (+60%)	0.0097 (+50%)	0.033 (90%)	0.0081 (+50%)	0.0508 (+45%)	0.0355 (+60%)	0.0355 (+60%)	0.059 (20%)	0.033 (+50%)	0.0282 (40%)
SVO PRO VIO STEREO	0.042173	0.031948	0.070498	0.068895	0.068102	0.14689	0.068017	0.18184	0.19803	0.11266	0.43548
SCALED	————	0.0142 (15%)	0.045 (-15%)	0.0354 (+20%)	0.021 (+10%)	0.1189 (+25%)	0.056 (+15%)	0.18184 (0%)	0.1903 (0%)	0.098 (10%)	0.43548 (0%)

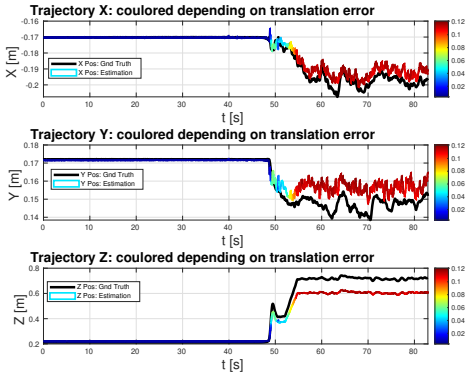
Table A.1: ATE values of RAM datasets with scaled and out of scaled data.

A.5 Visualization of trajectories in RAM datasets.

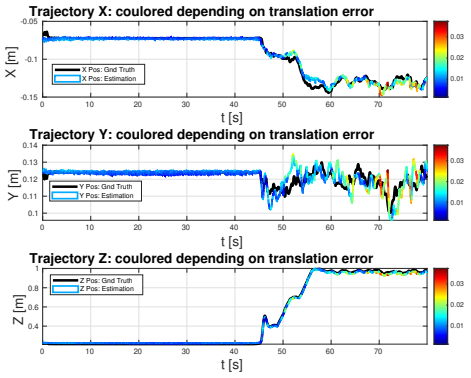
Figure A.4, Figure A.5 and Figure A.6 represent the ground truth trajectories against the estimated trajectories of ORB-SLAM3 stereo.



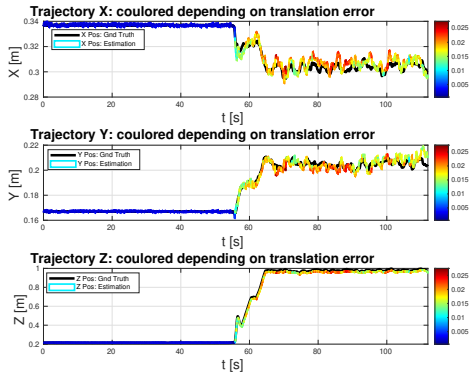
(a) Static2.



(b) Static3.

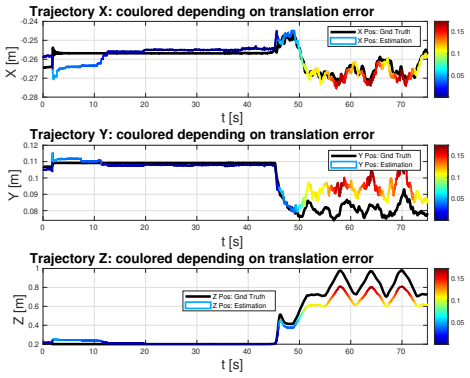


(c) Waving.

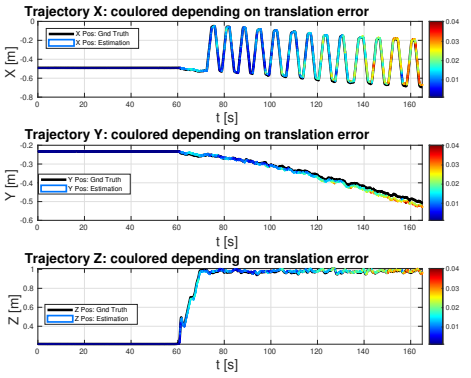


(d) Dynamic.

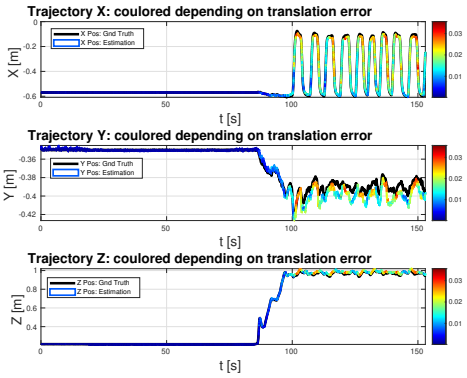
Figure A.4: ORBSLAM3 stereo with Frame Alignment for datasets Static2, Static3, Waving and Dynamic.



(a) Translation_0_1ms.

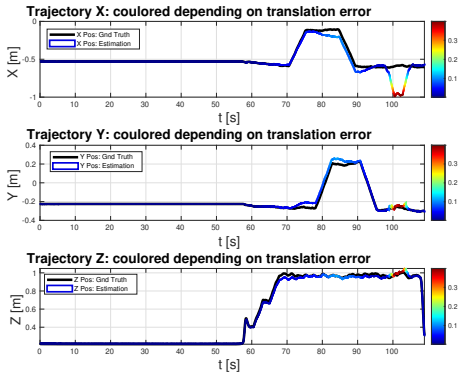


(b) Translation_FrontBack.

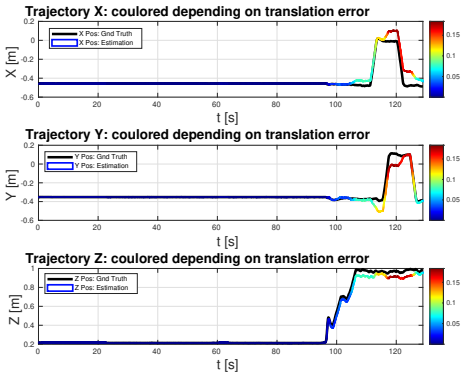


(c) Translation_FrontBack dataset_max.

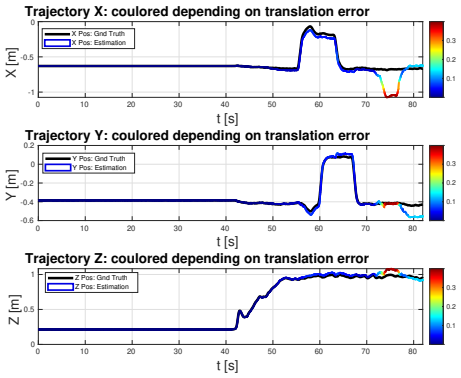
Figure A.5: ORBSLAM3 stereo with Frame Alignment for datasets Translation_0_1ms, Translation_FrontBack dataset and Translation_FrontBack dataset_max.



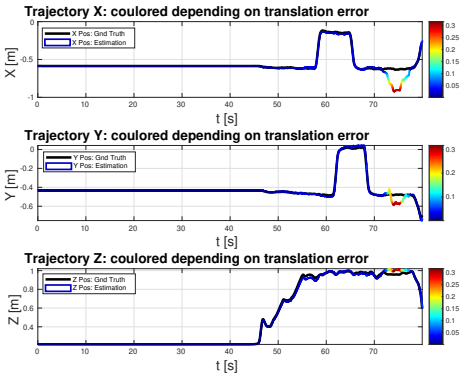
(a) Square2.



(b) Square3.



(c) Square4_max.



(d) Square5_max.

Figure A.6: ORBSLAM3 stereo with Frame Alignment for datasets Square2, Square3, Square4_max and Square5_max.

Bibliography

[1] Aerial-Core, "AERIAL-CORE Project Overview," 2021.

[2] S. Vatansever and I. Butun, "A broad overview of GPS fundamentals: Now and future," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC 2017*, 2017.

[3] S. A. Mohamed, M. H. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, "A Survey on Odometry for Autonomous Navigation Systems," *IEEE Access*, vol. 7, pp. 97466–97486, 2019.

[4] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, 2015.

[5] S. Du, W. Sun, and Y. Gao, "An investigation on MEMS IMU error mitigation using rotation modulation technique," in *27th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2014*, vol. 3, 2014.

[6] D. Scaramuzza and F. Fraundorfer, *Visual Odometry: Part I: The First 30 Years and Fundamentals*. IEEE ROBOTICS & AUTOMATION MAGAZINE, 2011.

[7] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer, "A survey of structure from motion," 2017.

[8] Y. Alkendi, L. Seneviratne, and Y. Zweiri, "State of the Art in Vision-Based Localization Techniques for Autonomous Navigation Systems," *IEEE Access*, vol. 9, pp. 76847–76874, 2021.

[9] S. Tang, Q. Zhu, W. Chen, W. Darwish, B. Wu, H. Hu, and M. Chen, "Enhanced RGB-D mapping method for detailed 3D indoor and outdoor modeling," *Sensors (Switzerland)*, vol. 16, no. 10, 2016.

[10] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from RGB-D data," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013.

[11] N. Nourani-Vatani, J. Roberts, and M. V. Srinivasan, "Practical visual odometry for car-like vehicles," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009.

[12] Y. Yu, C. Pradalier, and G. Zong, "Appearance-based monocular visual odometry for ground vehicles," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, 2011.

[13] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny, "Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking," *Journal of Sensors*, vol. 2021, 2021.

[14] S. Zhang, L. Zheng, and W. Tao, "Survey and Evaluation of RGB-D SLAM," *IEEE Access*, vol. 9, pp. 21367–21387, 2021.

[15] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, 2006.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, 1981.

[17] E. Olson, J. Leonard, and S. Teller, "Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates," in *ICRA*, 2006.

[18] F. Fraundorfer and D. Scaramuzza, "Visual odometry : Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics Automation Magazine*, vol. 19, no. 2, pp. 78–

90, 2012.

[19] T. C. Dong-Si and A. I. Mourikis, "Consistency analysis for sliding-window visual odometry," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012.

[20] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization," 2016.

[21] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, pp. 1309–1332, 12 2016.

[22] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, 2007.

[23] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the National Conference on Artificial Intelligence*, 2002.

[24] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2006.

[25] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, 2013.

[26] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, 2018.

[27] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[28] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *International Journal of Robotic Research - IJRR*, vol. 31, pp. 216–235, 05 2012.

[29] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G²o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, 2011.

[30] S. Agarwal, K. Mierle, and Others, "Ceres solver." http://ceres-solver.org.

[31] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph SLAM," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012.

[32] L. Carlone, A. Censi, and F. Dellaert, "Selecting good measurements via l1 relaxation: A convex approach for robust estimation over graphs," in *IEEE International Conference on Intelligent Robots and Systems*, 2014.

[33] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph SLAM," *International Journal of Robotics Research*, vol. 32, no. 14, 2013.

[34] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," in *Robotics: Science and Systems*, vol. 8, 2013.

[35] D. Sabatta, D. Scaramuzza, and R. Siegwart, "Improved appearance-based matching in similar and dynamic environments using a vocabulary tree," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010.

[36] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, 2006.

[37] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *International Journal of Robotics Research*, vol. 27, no. 6, 2008.

[38] F. Fraundorfer, C. Engels, and D. Nistér, "Topological mapping, localization and navigation using image collections," in *IEEE International Conference on Intelligent Robots and Systems*, 2007.

[39] F. Fraundorfer, C. Wu, J. M. Frahm, and M. Pollefeys, "Visual word based location recognition in 3D models using distance augmented weighting," in *4th International Symposium on 3D Data Processing, Visualization and Transmission, 3DPVT 2008 - Proceedings*, 2008.

[40] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification," *New York: John Wiley, Section*, vol. 10, 2001.

[41] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for IDF," *Journal of Documentation*, vol. 60, no. 5, 2004.

[42] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006.

[43] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012.

[44] K. L. Ho and P. Newman, "Loop closure detection in SLAM by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, 2006.

[45] D. Scaramuzza and Z. Zhang, "Visual-Inertial Odometry of Aerial Robots," tech. rep., Springer Encyclopedia of Robotics, 2019.

[46] L. HaoChih and D. Francois, "Loosely coupled stereo inertial odometry on low-cost system," 2017.

[47] S. Sirtkaya, B. Seymen, and A. A. Alatan, "Loosely coupled kalman filtering for fusion of visual odometry and inertial navigation," in *Proceedings of the 16th International Conference on Information Fusion*, pp. 219–226, 2013.

[48] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 2007.

[49] 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)*, pp. 298–304, 2015.

[50] K. Tsotsos, A. Chiuso, and S. Soatto, "Robust inference for visual-inertial sensor fusion," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, 12 2014.

[51] A. Martinelli. 2013.

[52] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 158–176, 2014.

[53] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Transactions on Robotics*, vol. PP, 08 2016.

[54] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, 2012.

[55] C. Chen, B. Wang, C. X. Lu, A. Trigoni, and A. Markham, "A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence," *ArXiv*,

vol. abs/2006.12567, 2020.

[56] S. Wang, R. Clark, H. Wen, and A. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2043–2050, 2017.

[57] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 963–968, 2011.

[58] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[59] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video in 3 d," 2018.

[60] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham, "A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence," 6 2020.

[61] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," 2020.

[62] L. Von Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[63] L. Han, Y. Lin, G. Du, and S. Lian, "Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints," 2019.

[64] 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.

[65] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Eklt: Asynchronous photometric feature tracking using events and frames," *International Journal of Computer Vision*, vol. 128, pp. 1–18, 03 2020.

[66] J. Delaune, R. Hewitt, L. Lytle, C. Sorice, R. Thakker, and L. Matthies, "Thermal-inertial odometry for autonomous flight throughout the night," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1122–1128, 2019.

[67] B. Silva, R. Xavier, T. Nascimento, and L. Gonçalves, "Experimental evaluation of ros compatible slam algorithms for rgb-d sensors," pp. 1–6, 11 2017.

[68] A. Quattrini Li, A. Coskun, S. M. Doherty, S. Ghasemlou, A. S. Jagtap, M. Modasshir, S. Rahman, A. Singh, M. Xanthidis, J. M. O'Kane, and I. Rekleitis, "Experimental Comparison of Open Source Vision-Based State Estimation Algorithms," in *Springer Proceedings in Advanced Robotics*, vol. 1, pp. 775–786, Springer Science and Business Media B.V., 2017.

[69] B. Joshi, S. Rahman, M. Kalaitzakis, B. Cain, J. Johnson, M. Xanthidis, N. Karapetyan, A. Hernandez, A. Q. Li, N. Vitzilaios, and I. Rekleitis, "Experimental Comparison of Open Source Visual-Inertial-Based State Estimation Algorithms in the Underwater Domain," 4 2019.

[70] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, J.-Y. Ertaud, J.-Y. Ertaud Bench, s. Nicolas Ragot, and t. Romain Rossi, "Benchmark of Visual SLAM Algorithms: ORB-SLAM2 vs RTAB-Map mark of Visual SLAM Algorithms: ORB-SLAM2 vs RTAB-Map Benchmark of Visual SLAM Algorithms: ORB-SLAM2 vs RTAB-Map*," 2019.

[71] M. Filipenko and I. Afanasyev, "Comparison of various slam systems for mobile robot in an indoor environment," 09 2018.

[72] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[73] B. Gao, H. Lang, and J. Ren, "Stereo Visual SLAM for Autonomous Vehicles: A Review," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 2020-October, pp. 1316–1322, Institute of Electrical and Electronics Engineers Inc., 10 2020.

[74] B. Bodin, H. Wagstaff, S. Saeedi, L. Nardi, E. Vespa, J. H. Mayer, A. Nisbet, M. Luján, S. Furber, A. J. Davison, P. H. J. Kelly, and M. O'Boyle, "SLAMBench2: Multi-Objective Head-to-Head Benchmarking for Visual SLAM," 8 2018.

[75] M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *Robotics, IEEE Transactions on*, vol. 29, pp. 734–745, 06 2013.

[76] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

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

[78] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[79] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2502–2509, 2018.

[80] J. Engel, T. Schoeps, and D. Cremers, "Lsd-slam: large-scale direct monocular slam," vol. 8690, pp. 1–16, 09 2014.

[81] C. Chen, H. Zhu, M. Li, and S. You, "A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives," 8 2018.

[82] D. Schlegel, M. Colosi, and G. Grisetti, "Proslam: Graph slam from a programmer's perspective," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[83] S. Sumikura, M. Shibuya, and K. Sakurada, "OpenVSLAM: A versatile visual SLAM framework," in *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, pp. 2292–2295, Association for Computing Machinery, Inc, 10 2019.

[84] 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, 02 2014.

[85] T. Whelan, S. Leutenegger, R. Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," 07 2015.

[86] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011.

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

[88] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, 2017.

[89] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[90] C. Campos, R. Elvira, J. J. Rodriguez, J. M. Jose, and J. D. Tardos, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE*

Transactions on Robotics, 2021.

[91] R. Elvira, J. Tardos, and J. Montiel, "Orbslam-atlas: a robust and accurate multi-map system," pp. 6253–6259, 11 2019.

[92] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016.

[93] Z. Zhang, C. Forster, and D. Scaramuzza, "Active exposure control for robust visual odometry in hdr environments," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[94] D. S. Giovanni Cioffi, "Tightly-coupled fusion of global positional measurements in optimization-based visual-inertial odometry," in *IEEE Int. Conf. Intel. Robots and Syst. (IROS)*, 2020.

[95] 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*, pp. 1280–1286, 2013.

[96] B. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society A*, vol. 4, pp. 629–642, 04 1987.

[97] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[98] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7244–7251, 2018.

[99] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7244–7251, 2018.

[100] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, 01 2016.

[101] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J.-Y. Ertaud, "Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map*," *2019 Eighth International Conference on Emerging Security Technologies (EST)*, pp. 1–6, 2019.

[102] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," pp. 801–808, 05 2016.