Visual Inertial Odometry for Mobile Home Robots

Paulo André Pereira Nogueira
Instituto Superior Técnico / UTL, Lisbon, Portugal
khakogcho@gmail.com

ABSTRACT

Robots aware of their surrounding, and able to roam freely, are expected to become ubiquitous in future homes. Visual inertial odometry is an enabling technology which is possible nowadays due to novel low-cost cameras and inertial measurement units (IMUs). In this work we analyze how the system-state initialization and dynamics propagation of IMU readings, involving intrinsic bias, has to match the visual motion captured by the camera. Then we analyze the synergies between these two sensors in the context of pose estimation.

A small wheeled mobile robot is developed based on one Raspberry Pi, one camera and one IMU mounted on an Arduino. The calibration of the system, namely the rigid transformation between camera and IMU is performed with a free, public domain, tool. The visual and inertial data is processed by means of an Unscented Kalman Filter with Lie group embedding for state representation. We propose a state initialization for this filter that enables matching the integrated IMU readings with the tracked visual features.

Experiments and results show the importance of successfully matching the feature tracks in the first images with the starting integration of IMU readings, as significant errors in initial bias estimations may preclude sensors fusion and filter convergence. Results show also that the operation of the fusion filter allows synergies between the two sensors, while the IMU provides instantaneous and reliable estimations of translation and rotation speeds, the visual component provides IMU bias correction.

I. INTRODUCTION

Visual Inertial Odometry combines imaging (video camera) with inertial (IMU) information to estimate the pose of a robot. The IMU specializes in tracking movement and orientation of objects. It is often used in navigation applications like aerial vehicles. However, its new found accessibility in current times has facilitated its integration in low-cost systems.

Combining cameras and IMUs in the context of pose estimation has multiple advantages, the main of which being the possibility of creating synergies between these sensors. In practice this means the visual information can help calibrate the IMU (bias), while the IMU provides motion scale to the visual information. With this in mind, we set out to develop a low-cost system that can perform visual inertial odometry, so we can critically analyze this synergy between sensors, and understand what are the key issues in using inexpensive equipment.

A. Related Work

Works as [11], that built navigation systems with low-cost IMUs, have shown it is critical handling properly the IMU bias. In general, research and development, bias is always considered an important aspect when working with IMUs, and there are multiple works like [16] that approach the issue of estimating and removing bias from the IMU readings, which are relevant to all IMUs independently of their price tag.

Simultaneous Location and Mapping (SLAM) is a set of problems that overlaps with the problem of pose (position and orientation) estimation. Works like [14] have compared the performance of visual SLAM (vSLAM) and visual inertial SLAM (viSLAM) concluded that typically viSLAM have better performances. Problems like the natural difficulty of vSLAM systems to capture motion scale and general robustness and accuracy are the main reasons.

Works as [3] have set out to implement already proven algorithms low-cost ground robots. They show it may be necessary to compromise filter performance due to the lack of computational power of low-cost robots. In an opposite direction, of accepting computational complexity, [15] proposes a state of the art versatile monocular visual inertial state estimator by means of a nonlinear optimization-based method. This approach is based on the alignment of a visual structure obtained by means of a structure from motion strategy with the results of a IMU preintegration.

These works show that it is worthwhile to pursue the idea of creating a low-cost visual inertial system dedicated to odometry. They also indicate that the process of IMU bias removal has a significant impact on the overall odometry performance. In this thesis we study inertial sensors combined with cameras, focusing on IMU bias estimation and removal, and vision / IMU synergies.

Visual inertial odometry, as a base component of mobile robots, has not only much research, but also real, interesting and relevant, datasets. The EuRoC MAV datasets [6] offer visual inertial measurements with accurate ground truth. These datasets are widely used in this area, whether to test new state of the art approaches or to establish benchmarks to compare algorithms. They will be used in this work along with our own datasets to test our solutions within a variety of scenarios.

B. Problem Formulation

Given a mobile robot the main objective consists of estimating its motion using onboard sensors. We consider using a camera and an IMU, and integrating the respective measurements, taking into account the lack of motion scale associated to a monocular camera and the bias of the IMU.

We set out to continue the work of [8] as it developed a low-cost mobile robot based on a Raspberry Pi that is able to acquire visual information for pose estimation. As referred, we add one IMU. The IMU allows us to introduce linear acceleration and angular velocity measurements in the system, which capture motion scale. These measurements are offset by bias, which needs to be removed as to avoid erroneous estimations.

In summary, in this work we propose developing a low-cost mobile robot that can acquire visual and inertial data, and then process this data to obtain pose estimates using an Unscented Kalman Filter (UKF) based on Lie groups. While processing this data, we evaluate how the sensors cooperate in the estimation.

Section II presents the engineering and mathematical background necessary to understand the concepts mentioned and/or proposed in this work. Section III describes the developed mobile robot and how it is able to acquire visual and inertial data. Section IV exposes the pose estimator filter with all the proposed modifications that enables it to meet our goals. Section V provides an overview of the different experiments performed as well as the results attained and a critical analysis of these results. Lastly, Section VI summarizes the work performed and highlights the main achievements in this work. Moreover, this section proposes further work to extend the activities described in this document.

II. Background and State of the Art

In this section we will cover several topics that serve as a theoretical basis for this work. We will also mentions works that had an impact on ours or serve as comparison to it.

A. Camera Model

A camera model defines the relation between 3D points in the real world and 2D points in an image. A common model is the pinhole model which is a perspective projection model. In this model the projection matrix that makes this transformation is defined as $P = K[R\ t]$ where R and t are the extrinsic parameters of the camera (i.e., the rigid transformation that map its referential to the world frame), and K is the intrinsic parameter's matrix defined as

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

where f is the focal length, c is the principal point, and s is the skew. With this projective matrix P we can maps the world points to the image plane using homogeneous coordinates (scale factor λ) as $\lambda \mathbf{x} = P\mathbf{X}$.

B. Inertial Measurement Unit (IMU)

One of the goals of this work is to incorporate an Inertial Measurement Unit (IMU) into a preexisting system. To achieve this, it is important that have an understanding of how this sensor works and what it provides.

Concept wise, an IMU is a combination of an accelerometer and a gyroscope (and a magnetometer if available, which for our IMU it is). All these three sensors measure their respective properties in three spacial axes. Let us consider the coordinate system of the IMU shown in Figure 3. These axes are applicable to the accelerometer and gyroscope. The magnetometer has the same axes, except z and y are inverted.

1) Accelerometer: It measures proper acceleration. This is the acceleration felt by an object. In practice, it is the acceleration of a body relative to a free falling state. This means, an accelerometer in free fall will measure zero acceleration for all axes. In contrast, a resting accelerometer on the surface of the Earth will measure, approximately, the gravity vector, $9.81\,\mathrm{m/s^2}$, upwards.

2) Gyroscope: Depending on the type, a gyroscope can measure angular velocity and/or orientation. Similarly to the accelerometer, we will consider a MEMS gyroscope, which measures angular velocity in degrees per second $^\circ/\mathrm{s}$ or radians per second $\mathrm{rad/s}$.

3) Magnetometer: Typically magnetometers are used to measure Earth's magnetic field. However magnetometers are susceptible to disturbances. These can take the form of hard (b) and soft (A) iron effects, and can be removed form the measurements as $m_{calib} = A(m - b)$. The first is caused by magnetic materials like magnets or electrified coils. The second is caused by ferromagnetic materials like iron, cobalt and nickel.

4) Measurement Model: Now that we know how the sensors in an IMU work and what they measure, it is time to talk about the mathematical models of the measurements. We will first touch upon bias and noise and how they effect estimations.

a) Noise and Bias: Every sensor is projected to measure the true magnitude of a property. Different sensors are affected by different types of phenomenons. We will focus on the two specific types of phenomenons that often cause issues to IMU sensors: (white) noise and bias.

White noise is a form of additive noise. It is a random signal with equal intensity at different frequencies. Bias is an offset from the real value the signal should have. This property is more often associated with IMU sensors. Part of Figure 1 shows a data signal (for example the gyroscope's angular rate) and what happens to it when affected by noise and bias.

b) IMU Sensors: We can represent the IMU measurements as angular rate $\omega = [\omega_x\ \omega_y\ \omega_z]^T \in \mathbb{R}^3$, linear acceleration $a = [a_x\ a_y\ a_z]^T \in \mathbb{R}^3$, and magnetic field $m = [m_x\ m_y\ m_z]^T \in \mathbb{R}^3$. In Equation 3 we have (embedded) the complete measurement model of the accelerometer and gyroscope measurements.

C. Sensor Fusion for Orientation Estimation

Sensor fusion is when there is a combining of multiple data sources as to generate a better understanding of the system. This can mean having average of two sensors for noise reduction, avoid single point failure, estimate unmeasured states or increase sensing coverage. Let us consider a 9 DoF IMU like the one presented in Section II-B. In this section we will discuss how we can use sensor fusion to estimate orientation using the considered IMU.

1) Estimating Orientation: We will now study two different ways of estimating orientation. One using both the accelerometer and the magnetometer, and one using the gyroscope.

a) Accelerometer and Magnetometer: These sensors can be used to estimate an absolute (does not depend on initialization) estimate of the orientation by performing cross products between the measurements (gravity vector and Earth's magnetic field) in order to estimate the north, east and down vectors. Through these it is possible to build a rotation matrix that represents the body's orientation.

One of the biggest drawbacks from this orientation estimation approach is how it expects the accelerometer to only measure the gravity vector. The accelerometer will measure all linear acceleration and even acceleration based on rotations not centered in the accelerometer. Every translation or rotation that generates linear acceleration will throw off the estimation of where the down axis is.

b) Gyroscope: By integrating the gyroscope's measures in a certain time interval we can obtain the rotation performed over that amount of time. By doing this in small steps and applying the estimate to the previous orientation we can estimate the current orientation. One of its flaws is the fact we need an initial estimate of orientation, due to this being a relative estimation. This can be provided by ground truth or making an initial estimation with an absolute method, like the one described with the accelerometer and gyroscope. However there is a bigger problem here: integrating the bias present in the measurements will cause the orientation value to drift over time, like shown in Figure 1. The only way to combat this is to estimate the bias value at each time step and remove it from the measurements. In Section IV-B1c we propose a method to estimate this value.

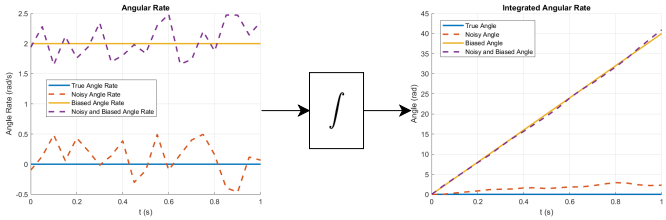


Fig. 1: Graphical representation of integration of biased angular velocity measurements and the resulting drift.

2) Sensor Fusion Filters: Having two estimates for orientation, the goal now is how to combine them as to mask each other's flaws. There are several filters that can do this job: Complementary, Kalman, or even dedicated filters like Madgwick or Mahony. From a conceptual standpoint they all typically do a similar job: they estimate orientation at each time step with the gyroscope measurements and use the accelerometer (and magnetometer if available) for the initial estimation and bias correction.

Since it was introduced in [10], the Kalman Filter has dominated all problems with linear systems due to its best characteristic: it is a statistically optimal filter. In our IMU sensor fusion scenario, this would mean this filter would calculate the perfect weight factor between the two orientation estimates every time. However there is a catch here: this IMU system is not linear, and therefore, it is not possible to use the Kalman Filter to estimate orientation. The EKF is a nonlinear version of the KF. The main assumption behind the KF is that all models are linear (which leads to Gaussian probability distributions). The EKF solves this problem by locally linearizing the function with its first order derivative. Although this is a simple extension to the KF which allows its usage with nonlinear systems, it is still sub-optimal since it does not capture the full system model and the noise on the system.

D. Visual Odometry

Odometry is the use of methods to estimate the position of an agent over time. From this we can derive Visual Odometry (VO) which is performing this task using visual information. Typically this also implies estimating the orientation (which when combined with the position forms the pose) and both linear and angular velocities that facilitate these estimations. It is usually performed by tracking specific points in a camera frame (features). It is common to use Kalman Filters to engage this problem, modeling them as online problems, i.e., problems where the estimation and acquisition occur in parallel.

Visual Odometry is an important concept in the context of Simultaneous Location and Mapping (SLAM, or in this case vSLAM), because, as the name implies, part of SLAM problems is to estimate position of an agent. The main goal here is to estimate the rigid transformation (3D position and rotation) that maps each camera frame into the world frame, while building a map of 3D points.

Another set of problems related to VO and vSLAM is Structure from Motion (SfM). These problems focus on reconstructing the 3D environment from 2D visual information. In practice, this reconstruction means estimating the rigid transformations that consistently map each camera frame in a considered world frame, as done in, for example, [9]. Although not perfectly, these sets of problems overlap, and so it is not uncommon that certain strategies are common to these problems.

E. State of the Art

We would like to highlight is [4], which provides the filter that was the software basis of our work. From here on out we will address this filter as Fusion18. This filter performs VIO with a monocular camera and an IMU. The reason why we chose to highlight this filter was its integration of Lie groups with a Square-Root Unscented Kalman Filter (SR-UKF). This approach is the culmination of several works that saw the potential Lie groups had in SLAM. It mainly builds upon two elements. First, the Lie group structure of SLAM advocated in the field of invariant filtering (as seen in [2, 1, 18]). Secondly, the UKF on Lie Groups (UKF-LG), whose general methodology has been introduced in [5].

III. Mobile Robot and System Architecture

In this section we will build upon the hardware and software setup of [8]. This work provided us with a mobile car

capable of live streaming video from its camera, and a PC user interface capable of giving movement commands to the robot and acquiring images from the video feed. Along with maintaining (and upgrading) the previous system's features, our two main objectives are: integrate an IMU in the system and enable the acquisition of formatted visual-inertial datasets.

A. Global System

In terms of hardware, this system can be divided in two main parts that serve distinct purposes: the mobile car and the user's PC. The first has the job to acquire visual-inertial data and make it available, while the latter has to be able to receive the data, format it and store it locally. To achieve this we propose the system represented in Figure 2.

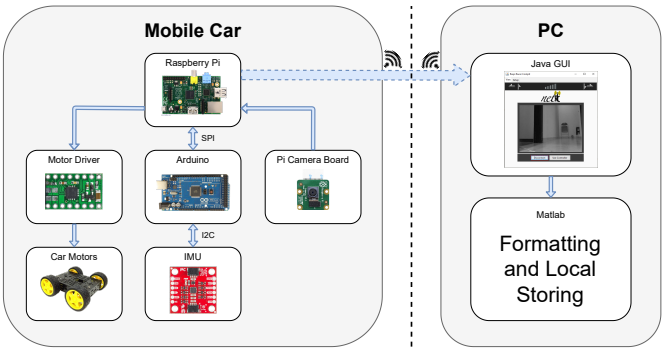


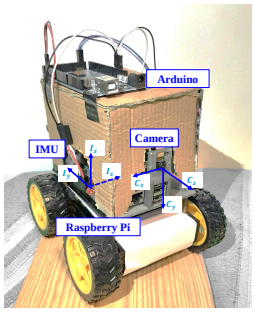
Fig. 2: Acquisition system architecture.

B. Mobile Robot

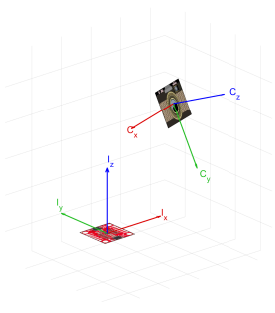
Our starting point was a mobile robot that had a Raspberry Pi Model B (R2) as the central processing component, a Pi Board Camera for image acquisition, a DRV8833 Dual Motor Driver Carrier with Multi-Chassis 4WD Kit (4 DC Motors) to enable movement and a USB battery box 4x18650 Coolook PB-2000 to power the system. With this setup, our first objective was to incorporate an IMU sensor. We used a SparkFun 9DoF (Degrees of Freedom) IMU Breakout ICM-20948 (Qwiic) SEN-15335 along with an Arduino Mega 2560 R3 that acted as middle man between the Raspberry Pi and the IMU. All connections are straight forwards according to the communication protocols available (which we will mention bellow).

In Figure 3 we can see the achieved mobile robot along with each sensor's coordinate system. This car has a 3D printed platform on which is has the Raspberry Pi and motor driver. We had to incorporate a cardboard box since we had no where to place the Arduino. However we made sure the IMU stayed in the printed platform since it oscillation less than the cardboard box when the robot is moving.

Now let us go through the main components of the car: what they do and how they interact with other components or outside elements.



(a) Main components



(b) Sensors

Fig. 3: Wheeled mobile robot.

1) IMU and Arduino: We chose this IMU due to its price-quality ratio and its Digital Motion Processor (DMP). This IMU allows us to get raw values from its sensors - accelerometer, gyroscope and magnetometer - but with the help of the DMP we can obtain processed values, like calibrated angular rate and 9DoF orientation quaternions.

The only available library that can access the DMP is written for Arduino, and as such, we need to use one as the middle man between the IMU and the Raspberry PI. These communications between the IMU and the Arduino are done with the I2C protocol.

2) Raspberry Pi: This is the central component of the mobile robot. The Raspberry Pi makes all the sensor data - visual and inertial - available to a PC client. It also receives commands that control the motor wheels.

The Pi is connected to both sensors, the Pi board camera and the Arduino-IMU, and to the actuators, the dual motor driver. To communicate with a PC, the Pi has three servers: a visual server that provides the clients with video feed from the Pi board camera; an inertial server that gets the inertial data from the Arduino and provides it to the clients, and a motor server that accepts commands for the DC motors (wheels).

The Arduino and Raspberry Pi are communicating over the SPI protocol with an ACK based system of messaging.

C. PC Interface

The PC interface is mainly characterized by the Java GUI and by the MATLAB scripts. The first serves as a graphical hub that allows the user to send motion commands and set network configurations. The second coordinates the acquisition from the PC side and collects the data, formatting it in the process.

D. Trajectories Ground Truth

One of the most important parts of this process is estimating ground truth. These are the values that represent truth in a dataset. The reference values that will enable us to evaluate the results of pose estimation. Our general strategy was to record the car and capture key positions in the trajectory. From these we interpolated the rest of the trajectory.

IV. Visual Inertial Odometry with State Initialization

In this section we will detail all the concepts behind our proposed Visual Inertial Odometry system. We will use Fusion18 [4] as a base and arrange it to meet our objectives. We set out to develop a system that could, at least, run with the datasets acquired in Section III, but ideally could run with many other quirky datasets.

A. Mobile Robot System and Measurements Model

a) System Frames: Because our camera and IMU do not have their coordinate systems aligned (as shown in Figure 3), we considered 3 frames: the world frame W, the body frame B and the camera frame C. W is the fixed global frame on which we want to estimate our key variables on. B is a mobile frame that follows and is centered on the IMU. We named it the body frame instead of IMU frame since we will consider this frame to represent the entire car. Finally, C is the camera's mobile frame.

Although B and C are mobile, for our hardware setup (again, as seen in Figure 3), they are fixed in relation to each other. Using Kalibr framework[1], we can get the extrinsic parameters $(\mathbf{R}_{C\to B}, \mathbf{t}_{C\to B})$ and $(\mathbf{R}_{B\to C}, \mathbf{t}_{B\to C})$ that map C to B and B to C, respectively.

Now all we need is a transformation that can map B to W. That is the transformation our whole problem means to estimate: the body's pose.

b) State Space: The state contains all the variables of interest that we want the filter to estimate. This includes the body's pose which is formed by its position $\mathbf{x} \in \mathbb{R}^3$ and orientation $\mathbf{R} \in SO(3)$, plus its velocity $\mathbf{v} \in \mathbb{R}^3$, the IMU biases $b_\omega \in \mathbb{R}^3$ and $b_a \in \mathbb{R}^3$, and the 3D position of p landmarks $\mathbf{p}_1, \ldots, \mathbf{p}_p \in \mathbb{R}^3$ in W. The state will be represented by the pair (χ, b), with χ being defined as matrix

$$\chi = \begin{bmatrix} \mathbf{R} & \mathbf{v} & \mathbf{x} & \mathbf{p}_1 \cdots \mathbf{p}_p \\ \mathbf{0}_{(p+2)\times 3} & & \mathbf{I}_{(p+2)\times(p+2)} & \end{bmatrix} \tag{2}$$

with size $(3+2+p)\times(3+2+p)$ and with $\mathbf{0}_m$ and $\mathbf{I}_m$ as, respectively, zero matrix and identity matrix with sizes m.

Finally, we can define the bias vector as $b = [b_\omega^T \; b_a^T]^T \in \mathbb{R}^6$.

c) Dynamic Model: Even though our hardware setup is different, our models are based on the ones present in [4]. This model is advantageous because, for our case, it allows us to anticipate scenarios like floors with inclinations. Therefore, we will consider a grounded body navigating on flat earth (with the possibility of a sloping floor) equipped with an IMU. We can model the system as

$$\text{body state} \begin{cases} \dot{\mathbf{R}} = \mathbf{R}(\omega - b_\omega + n_\omega)_\times \\ \dot{\mathbf{v}} = \mathbf{R}(a - b_a + n_a) - g \\ \dot{\mathbf{x}} = \mathbf{v} \end{cases}, \tag{3}$$

$$\text{IMU biases} \begin{cases} \dot{b_\omega} = n_{b_\omega} \\ \dot{b_a} = n_{b_a} \end{cases}, \tag{4}$$

$$\text{landmarks} \left\{ \dot{\mathbf{p}}_i = 0, i = 1, \ldots, p \right. , \tag{5}$$

[1] https://github.com/ethz-asl/kalibr

where $(\omega)_\times$ portrays the skew-symmetric matrix related with the cross product with vector $\omega \in \mathbb{R}^3$. We can group the multiple noises as $n = [n_\omega^T \; n_a^T \; n_{b_\omega}^T \; n_{b_a}^T]^T \sim \mathcal{N}(0, Q)$.

Using the Euler method we can discretize Equations (3) to (5), not including rotation. For a small time step Δt, we get

$$\begin{aligned} &\mathbf{R}_{t+\Delta t} = \mathbf{R}_t \mathbf{exp}\left[(\omega_t - b_{\omega,t})\Delta t + \mathbf{Cov}(n_\omega)^{1/2} g\sqrt{\Delta t}\right]_\times \\ &\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + (\mathbf{R}_t(a_t - b_{a,t}) - g)\Delta t, \quad \mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \mathbf{v}_t \Delta t. \\ &b_{\omega,t+\Delta t} = b_{\omega,t}, \quad b_{a,t+\Delta t} = b_{a,t}, \quad \mathbf{p}_{i,t+\Delta t} = \mathbf{p}_{i,t} \end{aligned} \tag{6}$$

d) Measurement Model: Along with the IMU, we are also considering the system to have a calibrated monocular camera that provides visual information and observes and tracks p landmarks in the visual scene. The camera observes landmark $\mathbf{p}_i$ through the standard pinhole model and respective projection model (Section II-A) as $\mathbf{y}_i = [y_u^i \; y_v^i]^T + n_\mathbf{y}^i$, where y_i is the result of projection:

$$\lambda \begin{bmatrix} y_u^i \\ y_v^i \\ 1 \end{bmatrix} = K[\mathbf{R}_{B\to C}^T \; (\mathbf{R}^T \; (\mathbf{p}_i - \mathbf{x}) - \mathbf{t}_{B\to C})], \tag{7}$$

with λ as the scale factor. This is a transformation from frame W to current image plane using the state's pose, $\mathbf{R}$ and $\mathbf{x}$ ($W \to B$), the IMU-camera extrinsic parameters, $\mathbf{R}_{B\to C}$ and $\mathbf{t}_{B\to C}$ ($B \to C$), and the camera intrinsic parameters, K ($C \to$ image plane).

The filter will compare this projection to the expected position of the landmark in the image. This process will invalidate the 3D landmarks that are too distant to the respective expected 2D feature.

e) Lie Groups: Something we did not mention right away is that the system dynamics form a Lie group. The state was arranged to form χ (2) which belongs to the special Euclidean group $SE_{2+p}(3)$. This representation also allows for a more compact and sturdy mathematical formulation that does not require variable conversions (which often leads to a decrease in accuracy and numerical consistency). This is particularly relevant for orientation as it has multiple forms of representation with different benefits, like quaternions, rotation matrices, angle-axis, and others.

The Special Euclidean group $SE_{2+p}(3)$ is an extension of the group $SE(3)$. It is defined by the same operator $*$, but the elements are slightly different. An element contains $1+p$ extra 3D vectors that, in our case, will be the linear velocity and positions of p landmarks in the world frame. Regarding the operation that defines this group, it represents a propagation. This means that if a system suffers a change χ in relation to the prior state χ_{t-1}, we can obtain the new system state as $\chi_t = \chi * \chi_{t-1}$.

B. Self Initialization and Feature Management

Now that we know the models used for this filter , let us talk about what changes needed to be made for us to meet our objectives. The two components that need intervention are: the initialization process and the feature management, with the latter being part of the filtering algorithm.

1) Initialization: Although the Fusion18 software has a robust and interesting filter, it lacks one key element: self initialization. Instead, this filter assumes the user will manually provide initialization values for the biases and 3D landmarks. As an example, the software comes with the results of running ORB-SLAM [13] to initialize the filter for a specific dataset.

First of all, let us clarify what is it that the initialization stage does. The initialization is responsible for establishing observation times, the initial state and a bank of landmarks. Aside from the observation times, which is an auxiliary variable to the algorithm, in a perfect world, the initialization should set the state and features to their correct values in the world. This means, we should manually measure these values and feed them to the algorithm. Aside from the fact we cannot always perform such measurement, this is not a very practical approach.

We will initialize the pose and velocity the same way the Fusion18 algorithm: by setting it as the ground truth. The main reason for this is so we can compare the results to the ground truth. This choice only implies that our starting pose will not be aligned with the world frame W origin, but other from that it does not have any real significance. In terms of how the filter works, all referential transformations are present where they need to be, which allows for any given initial pose. The filter will estimate pose starting on the initial one, regardless of its actual value.

a) Observation Times: The IMU and camera have different rates of acquisition. In a typical scenario, the IMU's rate is higher (as is with ours). This makes it so there usually are multiple IMU samples to be processed before each consecutive image. Considering Fusion18 processes an IMU sample at each iteration, one of the inputs of the algorithm is at which iterations there will be an images to process. This is what we define as observation times.

For datasets that have a rigorous sample rate for both sensor measurements, we can get these observation times by simply using the visual and inertial acquisition frequencies. However, if, for a given dataset, the time interval between samples changes, this equation cannot be used. The datasets we were able to acquire have this unfortunate property. As such we removed images that predated the first IMU sample or came after the last IMU sample. We also removed consecutive images (except for the least old) that came temporally in between two consecutive IMU samples, as the filter does not foresee this possibility.

b) Landmark Initialization: There are two types of features we need to estimate in the initialization process. First we need to estimate the 3D position of landmarks for the initial state. Second, we need to get 2D features for latter replacement of the state's landmarks. These last features can be obtained with by using a corner detection algorithm like minimum eigenvalue (which is Fusion18's preferred feature extraction algorithm) in the first image frame.

To estimate the 3D positions of detected landmarks to initialize the state, we convert the pixels, u and v, to metric coordinates, X and Y, as $X = (u - c_x)/fs_x$ and $Y = (v - c_y)/fs_y$. However, with just one view, there is not enough information to estimate depth. Our solution was to set Z to the same value for all landmarks. With this transformation we have estimated the 3D landmarks in the camera frame C. Now all that is left to do is to transform them to the world frame W. This can be done using the the camera-IMU extrinsic parameters, $\mathbf{R}_{C\to B}$ and $\mathbf{t}_{C\to B}$ ($C \to B$), and the initial state pose, $\mathbf{R}$ and $\mathbf{x}$ ($B \to W$).

To prevent a collective invalidation of the initial landmarks, we removed all, but one, IMU samples that came temporally before the first image, as to avoid multiple integration of bias, which would lead to a deviated pose, which would cause the landmark invalidation (Equation 7).

c) Gyroscope Bias Estimation: Our strategy to estimate this bias was inspired by [15]. To perform this estimation we are assuming an offline setup, i.e., we first need to acquire all data and only afterwards will we estimate the values. Let us consider I and J as the total number of images and IMU samples in the dataset, respectively. The first step is to estimate the orientation of all image frames, q_i (quaternions with $i = 1, \ldots, I$), using only the visual information. To achieve these orientation estimates we use the MonoSLAM algorithm of [7].

Secondly (or in parallel), we preintegrate the inertial data in the camera frame C in order to estimate the rotation matrices between each image frame, γ_i, $i = 1, 2, \ldots, I$. We consider the starting bias to be null $[0\ 0\ 0]^T$. However, if the user already has an initial bias estimation, it can be inputted to this method. For j as a discrete moment corresponding to a IMU sample within $[t_i, t_{i+1}]$, we can discretely integrate the gyroscope measurements as

$$\gamma_{j+1} = \gamma_j \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\omega_j - b_{\omega_j})\delta t \end{bmatrix}, \tag{8}$$

with δt as the time interval between IMU samples j and $j+1$. In order to get the rotation between image frames $i-1$ and i, we need to accumulate the rotations within that interval by doing the quaternion multiplication between consecutive rotations following the correct order.

We now have all the orientations at each image frame coming form the visual data and all rotations between each consecutive image frame coming from the inertial data. In a perfect world, for a arbitrary consecutive frames i and $i+1$, if we rotated the frame's orientation q_i over γ_{i+1}, we should get q_{i+1}. However, due to bias and noise, this will not happen. Because of this, we can build a cost function to estimate the bias:

$$\min_{\delta b_\omega} \sum_{i \in I} \left\| q_{i+1}^{-1} \otimes q_i \otimes \gamma_{i+1} \right\|^2, \quad \gamma_{i+1} \approx \gamma_{i+1} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_\omega}^{\gamma} \delta b_\omega \end{bmatrix}. \tag{9}$$

$J_{b_\omega}^{\gamma}$ is a part of the first-order Jacobian, J_{i+1}, of the covariance matrix, P_{i+1}, obtained in the IMU preintegration (see [15]).

By initializing both starting orientations (MonoSLAM and preintegration) as identity quaternion (null rotation) we are guaranteeing both methods to estimate orientation in the same reference since we have both body B and camera C frames fixed in relation to each other. Because we are only interested in rotations, and the whole systems rotates as one, all estimations stay coherent. With that done, we have now estimated the initial bias. It is possible to repropagate the

rotations between frames as (9) with the estimated bias and repeat the process.

2) Feature Management: By default, the filter will only try to replace the state's landmarks after processing 120 IMU samples. This is a measure put in place to ensure the filter can robustly initialize the 3D position of new landmarks. Not only is this a limitation on the size of a dataset, it also means that, in this interval with the initial landmarks, it is imperative the state is not too far off from reality. If the state's pose causes a collective invalidation of landmarks in this initial period, that would cause the filter to run solely with the inertial data without any possibility of bias estimation until the 120-th iteration. This would result in filter divergence. To combat this, we changed this iteration threshold to 30 and made all logic related changes to the original code necessary for this change to work properly.

C. Estimated Pose Error Metrics

Let us now consider two error metrics: RMSE metric that evaluates the residuals in a point to point basis, and a Procrustes based metric that aligns the estimates to the ground truth using the Procrustes problem to find the rigid transformation (with scale) that most closely performs this alignment. The first metric will be used to evaluate all orientation estimates and trajectories of public datasets, while the second will be used for trajectories estimated with our acquired datasets.

V. Experiments and Results

In this section we will present all the experimental results that lead us from acquiring an IMU to running Fusion18 with our own datasets.

A. Hardware Setup Testing and Calibration

a) Testing IMU Orientation Estimation : To visualize the orientation in real time, we used the software provided in [17]. It receives quaternion information from a serial communication port and represents it visually using a rectangular cuboid.

A really positive notion was the fact the cuboid reacted with an high level of accuracy and speed to each and every movement, regardless of its speed. The most relevant issue noticed was horizontal drift. It was apparent rotation movements caused the most drift, although we could notice a slight drift even if not moving.

b) IMU-Camera Extrinsic Parameters: One of the key parts of this work was to estimate the extrinsic parameters that performed the rigid transformation between the camera frame and the IMU frame, $(\mathbf{R}_{C\to B}, \mathbf{t}_{C\to B})$. Without these parameters we cannot have any form of synergy between the camera and IMU since they are representing data in different coordinate systems. To estimate these parameters we used the Kalibr toolbox[2]. In Figure 3 we can see the results of this calibration (not proportional to real life). In these figures it is hard to evaluate, but the results are highly accurate in comparison to the real life hardware.

[2]https://github.com/ethz-asl/kalibr

B. IMU and Video Datasets

Throughout this work we used datasets from two sources: EuRoC MAV [6] and our own mobile car. The main dataset used from [6] was the same dataset that came default with the Fusion18 filter: *V1_02_medium* (20 Hz and 200 Hz for the cameras and IMU, respectively). The Fusion18 software also came with initialization values for this dataset. This allowed us to test certain scenarios without having to worry much about the quality of the dataset.

With our mobile car we were able to acquire several visual inertial datasets with a stable visual sample rate, and varying inertial sample rate. All these datasets start with a few seconds of no motion.

Movement Forward "*mf_04*" This first dataset had the objective of having a simple trajectory with no rotations. Therefore, we acquired a it with the car making a straight line trajectory at a constant speed. The acquisition frequencies are 13 Hz (average) and 5 Hz for the inertial and visual data, respectively.

Movement Curve "*mc_01*" This dataset was meant to introduce rotations. More specifically, this dataset introduced rotations over the IMU's z axis. We were able to capture it at the frequency rates of 10 Hz (average) and 5 Hz for the inertial and visual data, respectively.

C. Effect of One IMU Sample Error

In order to see how the filter handles wrong IMU data we thought of a simple test: insert one noisy IMU sample. The filter uses IMU samples to make small changes to the state (Equation (6)), while also propagating its uncertainty. When the filter has an image to process, it then makes an update to the state based on that uncertainty using both the previous state estimated with IMU data and the image data. We wanted to see how robust the filter was against IMU noise. We inserted a IMU sample with a certain percentage of noise (size factor of original sample) in the middle of execution and compared the results to a no error run (normal run).

Let us consider the experiment with $400\,\%$ error in the sample. Although the caused offset in RMSE is not significantly big, the orientation could not be corrected, which affected the position and velocity estimates, even with them showing almost instant recoveries to the initial offset. This implies a good recovery must be achieved by the whole state. If it is just partial, then the bad estimate will influence the others.

Now let us focus on the experiment with $900\,\%$ error. These results came in contrast to last experiments'. All three estimates make a swift recovery which causes the last two seconds of estimates to have approximately the same RMSE as a normal run.

In the experiments with $1600\,\%$ noise, we can see the filter starting to fall apart under the impact of the noisy sample. The RMSE shape for each estimate is unrecognizable and although there is an effort for correction that lowers the RMSE, ultimately the filter cannot get a stable state and it falls in the loop of the sensors being hindrances to each other.

$3000\,\%$ noise is the final transition point for the filter. It is where is completely breaks without being able to recover. The

offset in the estimates is so big that neither the inertial or the visual data can help to make a correction. As mentioned on Section IV, all features will be invalidated if the state's pose is too distant from the real pose. With other noise percentages, although the sample caused an immediate invalidation of landmarks which negatively affects the state's estimation, the pose was still plausible which meant new features could facilitate a recovery. However, this last sample makes it so all transformations using the state's pose are implausible, and therefore cause the filter to diverge.

It is true that to converge, the filter must estimate a state that enables cooperation between sensors (as depicted in Section IV), but what this experiment showed is that once that state is reached, the filter becomes robust. It was necessary to introduce a sample with $1600\,\%$ noise in order to make the filter not converge.

D. Mobile Robot Navigation

Before talking about the final results of running Fusion18 with our datasets, let us go through some results from our modifications and experiments.

1) IMU Bias: To validate the gyroscope bias estimation we setup a small experiment using dataset *mf_04*. We will be estimating two values with the samples: angle and trajectory. In both we will consider a case with the original raw values and another with the IMU samples after subtracting the estimated gyroscope bias. This will allow us to see if removing the estimated bias will prevent or reduce the estimates from drifting away from the correct values over time.

To estimate the angle we integrated the angular rate over time. As for the trajectories, we used the same propagation as the Fusion18 filter. For the angle, although slight, the effect of drift is noticeably smaller when removing the estimated gyroscope bias, since the line has a smaller slope. As for the trajectories, we should note they were aligned with our Procrustes metric present in Section IV-C. But this alignment was not equal for both. We aligned the trajectory with less bias and then used that transformation to align the biased trajectory. This allows us to see that, in fact, the subtraction of bias helps in reducing the effect of drift, which in the results, it did.

Although this experiment explicitly shows removing the estimated initial bias helps in the immediate estimations, it implies something more important. Estimating an initial bias that helps reduce the measurement's bias means we are facilitating the filters initialization process. Even if the bias is not showing a huge impact right away, what matters is that we have a better initial estimate than a null bias. This will help the filter get to a better bias value faster, and, as mentioned before, will help the IMU estimate a good pose, which will help in the observation of landmarks, which will help the bias estimation, and so on.

2) First Frames fusion with IMU Readings: One of the most broad problems we faced in this work, in terms of dependencies, was the handling of the initialization period. The filter only uses landmarks that, when projected into the image frame (see Equation 7), have a small enough distance to the tracked features ($20\,\text{pixel}$). Without features, the IMU biases will not be estimated, which causes a worse estimation of pose. And this same pose is used in the projection that can result in landmark invalidation. Also, as mentioned in Section IV-B2, the filter has an initial period on which it will not replace the state's 3D landmarks, even if they all get invalidated. If we add of all this together we can see just how important this initialization period is. Most of the times this period is the difference maker between the filter converging or not. So we have to make sure the first image frames can fuse with the IMU readings in order to create a positive work flow between these two sensors, where each sensor helps estimate properties for the other.

One measure we put in place to fight this problem was the removal of all (but one) IMU samples that came temporally before the first image. To show the impact of this measure we plotted the real and estimated features (with and without the measure) in the first 5 frames that are used in the filter for observation. The results are in Figure 4. For simplicity, we will refer to these scenarios by the color of their features. The reason why the blue features only appear in the first frame is because they were invalidated due to their error (with regards to the red features). The filter rendered these features as invalid so they would not deteriorate the state estimation. Hijacking the filter and demanding it to further estimate the state with these invalid landmarks resulted in reprojections outside the bounds of the image.

It is clear to see that the number of propagations heavily influences the first projection (observation) of 3D landmarks. This is because each time we propagate the state, we are effectively integrating the bias present in the measurements. As explained in Section II-B4a, integrating bias results in a drift over time from the correct values, even if in a motionless scenario. This means the higher the number of times we propagate the state without a good bias estimate (to reduce the measurement bias), the higher the drift from reality.

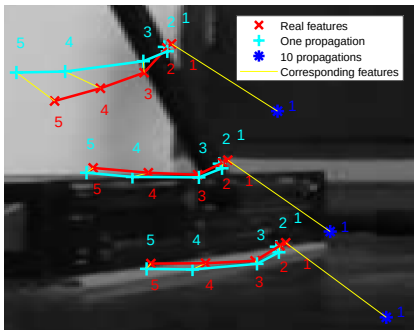


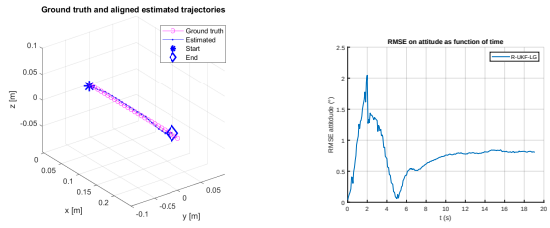
Fig. 4: Feature tracking on first 5 frames with and without IMU sample removal. Red features are the real (tracked) features. Cyan features are the estimated features with one initial propagation (our measure). Dark blue features are the estimated features with 10 initial propagations.

Without this IMU sample removing strategy, and also the changes in feature management described in Section IV-B2, all initial landmarks would get invalidated and then, because Fusion18 was set to only replace the state's landmarks after 120 IMU samples, the filter would run solely with the IMU.

Due to bias, by the time the filter replaced the landmarks, the pose would already be too far off from reality and the observation step would be unable to use any visual information, ultimately resulting in the filter diverging.

Our solution ensures the first landmark observation will not cause a general invalidation of features, which will provide us with an immediate start in IMU bias estimation. Although far from perfect, this provides us with adequate results that enable the filter to converge, as shown in Figure 4.

3) Navigation Assessment: Let us finally talk about the results of running Fusion18 with our own datasets. We considered the Procrustes error metric when running our own datasets. The results for dataset *mf_04* are in Figure 5.



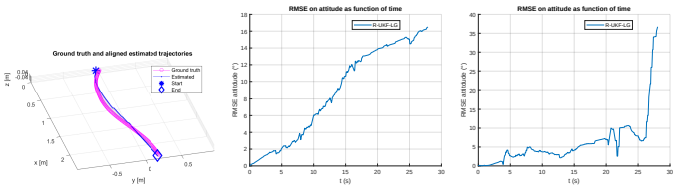
(a) Trajectories (b) Orientation RMSE over time

Fig. 5: Procrustes aligned estimated positions and ground truth positions point clouds (a) along with orientation RMSE function (b) and when running the *mf_04* dataset.

It is clear by looking at Figure 5b that the filter had an initialization period of 2 s before it had an estimation of the state that enabled sensor cooperation. This goes to show how important the first seconds of a dataset are. Which is why our datasets have an initial static period. It greatly helps the filter find a stable state that can create synergies.

The reason why the results with dataset *V1_01_medium* have such a seamless initialization period is that the filter started with half of the state (pose and linear velocity) coming from ground truth values and the other half (bias and 3D landmarks with corresponding features and backup features) coming from the result of running ORB-SLAM. This pretty much means the filter starts with values as close to reality as possible, and therefore, already has a state able to enable the synergy between both sensor's data.

When running dataset *mc_01* we experimented with the process noise covariance, which is related to the IMU measurements, and by default resulted from the IMU calibration (IMU intrinsic parameters). Our obtained Fusion18 results are shown in Figure 6. Trajectories are similar for both versions of the process noise. In Figure 6b we can notice the filter is not being able to arrange the synergy between sensors. To combat the early negative effects of the IMU bias, we increased the process noise covariance. As shown in Figure 6c, although not very robust, the synergy between sensors is present. However, around $t = 25\,\mathrm{s}$ the RMSE has an almost vertical increase. This is due to the fact during the visual capture of dataset *mc_01* we were unable to have a stable luminosity , which leads to detection and tracking of multiple feature in shadows. This will lead the vision portion of the filter to perceive an nonexistent movement and because we increased the process noise covariance, the filter will trust the vision more than the IMU. This is why this lighting problem does not appear to be present in Figure 6b.



(a) Angled view of the trajectories (b) Orientation RMSE over time (c) Orientation RMSE over time for higher process noise

Fig. 6: Procrustes aligned estimated positions and ground truth positions point clouds from and angled view (a) along with orientation RMSE function for the calibrated process noise (b) and for an increased process noise (c) when running the *mc_01* dataset.

Overall, these results are pretty positive since they validate our initialization estimation and show it is possible to run our own acquired datasets with plausible results (even if in need of alignment). But it left us with one final question: what if we starting the filter with a stronger initialization? What if we ran the filter up to a point and then started it again with the results of the first run? This is the final experiment we will present.

The steps for this experiment are: run Fusion18 on half of a dataset; save its final state along with all detected features and landmarks; run second half of the dataset but using this previous data to initialize it. The specific variables we are considering in this initialization are both sensor biases, the 3D landmark estimations and the reserve features along with their sighting in past images. We will still initialize the pose and linear velocity using ground truth.

We will consider the last dataset: *mfpf_02*. This dataset was specifically designed for this experiment. It was acquired in the exact same conditions as *mf_04*. In terms of movement, this dataset is also very similar to *mf_04*, except it is doubled. It follows the pattern: pause, forward, pause (moment of the filter reset), forward, pause. In sum we will run the filter on two similar trajectories that are separated by resting phase, on which we will make the filter reset. As expected, we will also perform this experiment using the initialization established in Section IV-B1 to initialize the state after the reset. This will allow us to compare our generic and (mostly) online initialization to this calibration like approach.

We will focus on the orientation results in Figure 7, since they are more explicit than the trajectories. The orientation RMSE's show disparity in the results. The run with our initialization has a linearly increasing error which is the result of a inadequate initialization process. The run with the Fusion18 initialization answers our question: a stronger initialization results in a short, or almost non existing initialization period. It is similar to the results with *V1_02_medium*, which possesses a strong initialization due to the fact it uses ground truth and estimated landmarks and bias from running ORB-SLAM [13].

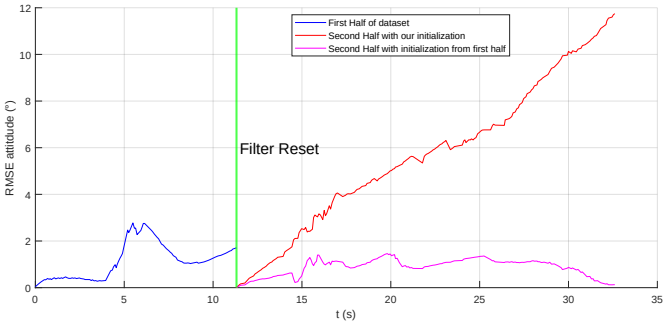


Fig. 7: Orientation RMSE when running the two halves of dataset *mfpf_02* with a reset in the middle. Red uses our initialization after the reset. Magenta uses as initialization the final state of the last half.

VI. Conclusion and Future Work

A. Conclusions

In this work we set out to study develop a low-cost mobile robot able to acquire visual and inertial data, and to study the synergies between those types of data in the context of pose estimation. Not only were we able to integrate and valid a low-cost IMU into this system, we were able to acquire visual inertial datasets. We proposed a state initialization for a Unscented Kalman Filter based on Lie groups that experimentally showed promising results.

The most crucial point of our work was how we experimentally saw the process that made the inertial and visual information synergize. We saw how important it was for the filter to quickly estimate the IMU bias so its measurements would not cause the state to drift from reality. Without this taking place we verified the landmark observations were too far from reality, which caused the filter to invalidate them. This would start a cycle of deterioration between both sensors. However, we were also able to experimentally get results that showed the opposite: a cycle of cooperation between sensors, where the bias estimation coming from the visual information would make the landmark projection more accurate, which in term would improve the bias estimation.

Ultimately we were able to advance (even if just a little) towards the goal of having low-cost mobile home robots that can autonomously roam and perform task.

B. Future Work

Starting with the IMU, there is great potential in inexpensive IMU's. Traditionally, IMU's are used in systems to provide measurements, and any estimation is performed outside the IMU. However, we witnessed the quality of the onboard orientation estimation.This opens plenty of new paths. Personally we think it would be interesting to develop a system where, instead of modeling the system around the IMU measurements, we could model it around the orientation coming directly from the IMU.

The topic of IMU sensor fusion for orientation estimation is something that, in our opinion, should be further explored. Filters like [4] have a lot of merit for the results they display. However they opt for a IMU-camera sensor fusion approach. We believe this concept could be improved by first fusing each IMU sensor with a separated filter and only then fuse inertial and visual data. Overall, the current accessibility to IMU's should be capitalized on. Be it with new filter, like the relatively recent Madgwick filter [12] that is currently one of the most used in filters for orientation estimation in digital motion processor, or with integration of existing technology.

References

[1] Axel Barrau and Silvère Bonnabel. "Invariant Kalman Filtering". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (May 2018).

[2] Axel Barrau and Silvere Bonnabel. *An EKF-SLAM algorithm with consistency properties*. 2016.

[3] Liu Bo, Lin Li, and Hengzhu Liu. "SoC Implementation of Visual-inertial Odometry for Low-cost Ground Robots". In: *Journal of Physics: Conference Series* 1453 (Jan. 2020), p. 012091.

[4] M. Brossard, S. Bonnabel, and A. Barrau. "Unscented Kalman Filtering on Lie Groups for Fusion of IMU and Monocular Vision". In: *International Conference on Robotics and Automation (ICRA)* (2017).

[5] M. Brossard, S. Bonnabel, and J. Condomines. "Unscented Kalman Filtering on Lie Groups". In: *IROS 2017, EEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE/RSJ. Vancouver, Canada, Sept. 2017.

[6] M. Burri et al. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* 35 (Jan. 2016).

[7] J Civera, A J Davison, and J M M Montiel. "Inverse Depth Parametrization for Monocular SLAM". In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 932–945.

[8] João Cruz. "Wireless Mobile Camera". MA thesis. Instituto Superior Técnico, 2015.

[9] A. Fitzgibbon and A. Zisserman. "Automatic Camera Recovery for Closed or Open Image Sequences". In: *European Conference on Computer Vision* (July 1998).

[10] Rudolph Emil Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[11] Yufei Liu, Noboru Noguchi, and Kazunobu Ishii. "Development of a Low-cost IMU by Using Sensor Fusion for Attitude Angle Estimation". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 4435–4440.

[12] S. Madgwick. "An efficient orientation filter for inertial and inertial/magnetic sensor arrays". In: *Report x-io and University of Bristol (UK)* 25 (2010), pp. 113–118.

[13] R. Mur-Artal, J. Montiel, and J. D. Tardós. "ORB-SLAM: a Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[14] S. M. Potirakis et al. "Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking". In: *Journal of Sensors* (Feb. 2021).

[15] T. Qin, P. Li, and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[16] Glauco Garcia Scandaroli and Pascal Morin. "Nonlinear filter design for pose and IMU bias estimation". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4524–4530.

[17] ZaneL. *3D Node.js quaternion sensor*. Version 1.0. May 2020. URL: https://github.com/ZaneL/quaternion_sensor_3d_nodejs.

[18] Teng Zhang et al. *Convergence and Consistency Analysis for A 3D Invariant-EKF SLAM*. 2017.