ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Smart Vehicular Systems

LOST IN GATES: ENHANCING STATE ESTIMATION IN HIGH SPEED AUTONOMOUS DRONE RACING

CANDIDATE

Flavio Pinzarrone

SUPERVISOR

Prof. Giovanni Pau

Academic year 2022-2023

Session 3rd

Abstract

This thesis delves into the integration of Artificial Intelligence (AI) within the realm of autonomous racing drones. Traditionally, the racing industry has led the way in advancing resilient and streamlined systems, with recent emphasis shifting towards the implementation of autonomous driving mechanisms. The intricacies of this field present complex challenges, demanding the development of precise control and perception systems that operate with minimal reaction times and constrained resources.

The research, conducted within the Drone Racing team at the Autonomous Robotics Research Center of the Technology Innovation Institute, primarily focuses on advancing perception and state estimation systems for autonomous racing drones.

Central to the study is the introduction of a novel high-speed autonomous drone racing multimodal dataset and an innovative map-based, perceptually aware state estimation technique.

This work is instrumental in pushing the boundaries of autonomous drone racing technology, offering valuable insights and solutions that contribute to the broader advancements within the field of autonomous systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world of racing has long been a driving force for the advancement of safer and more efficient technologies. The harsh conditions that vehicles face on the racetrack magnify challenges that are relevant to everyday transportation. This continuous push for progress has not only transformed competitive racing but also played a pivotal role in reshaping technology in our daily lives.

In recent years, the increasing interest in self-driving systems has given rise to autonomous vehicle competitions that span various aspects, including obstacle avoidance, path planning, and races involving a diverse array of vehicles, such as cars and drones. These latter competitions introduce even more intricate challenges, demanding the creation of exceptionally precise control and perception systems with rapid response capabilities.

This master's thesis delves into research conducted at the Autonomous Robotics Research Center of the Technology Innovation Institute. It primarily concerns the enhancement of perception and state estimation systems for autonomous racing drones. The research involves the creation of a high-speed dataset and the introduction of an innovative state estimation approach for autonomous racing drones. The findings are intended to provide valuable support for future research efforts aimed at advancing learning-based control and perception systems.

Drone racing

Drone racing, an electrifying and rapidly evolving sport, has surged in popularity in recent years, captivating a diverse audience of enthusiasts and spectators. This thrilling competition centers around the piloting of small, agile unmanned aerial vehicles, commonly known as drones, through complex and high-speed courses filled with obstacles and challenges. It seamlessly blends the precision of remote-controlled flight with the rush of competitive racing, resulting in a captivating fusion of technology and competition.

In the realm of drone racing, skilled pilots make use of specialized first-person-view (FPV) goggles that grant them an immersive and real-time perspective from the drone's perspective. This unique technology allows them to deftly navigate the racecourse, threading their way through gates, tunnels, and various other obstacles, all while racing against the clock.

The fundamental objective of drone racing is simple yet deceptively challenging: complete the designated course in the shortest time possible, all the while demonstrating extraordinary agility and precision in piloting the drones. With racecourses that often feature hairpin turns, long straights, and technical challenges, drone racing demands exceptional control and agility, pushing pilots and their equipment to their limits.

Beyond being a compelling and visually captivating sport, drone racing has steadily amassed a fervent global following. It not only underscores the potential of unmanned aerial systems for recreational enjoyment but also acts as a catalyst for technological innovation, pushing the boundaries of what drones can achieve in terms of speed, agility, and precision.

An illustrative example of this evolution is Autonomous drone racing, where researchers strive to replace human pilots with sophisticated software. This software is designed to execute all the sensory and manual actions that a human pilot would undertake, including perceiving surroundings, making rapid decisions, and taking actions autonomously.

Chapter 2

Background

Estimating the position and attitude of an autonomous racing drone on a racing track has been an open problem since the beginning of research in this field and has become more and more challenging with the increasing agility and speed that state-of-the-art technology and hardware have enabled. To solve this problem there has been the joint effort from both the robotics and the AI research communities to release datasets and develop new techniques. In this section I will first chronologically review the datasets that are publicly available for UAV purposes, highlighting what were the critical points that suggested the need for a new and more suited one. Then I will go through the different approaches to localization and state estimation in aggressive flights.

2.1 Review of Autonomous Drone Racing Datasets

In the realm of AI-driven perception systems for autonomous racing drones, the availability of suitable datasets stands as a cornerstone for advancing research and development. Over time, a range of datasets has emerged, each tailored to specific facets of this dynamic field. In this section, I want to cover all existing datasets in an extensive exploration, unveiling their distinctive attributes, and emphasizing the contributions of our newly introduced dataset.

Early Datasets

Among the earlier datasets aimed at Visual-Inertial Odometry (VIO) and Simultaneous Localization and Mapping (SLAM), we find the 2016 EuRoC dataset [6] and the 2017 Zurich Urban micro aerial vehicle MAV dataset [29]. The main aim of these dataset was to provide a benchmark for the emerging VIO and SLAM techniques, to promote their early adoption. For these reasons, they were primarily characterized by lower speeds and low image capture frequencies. Such limitations made this datasets unsuitable for addressing the rigorous demands of drone racing, where high-speed maneuvering and real-time perception are paramount.

Advent of Racing-Oriented Datasets

In recent years, there has been a notable surge in the development of datasets tailored to meet the specific demands of drone racing. For example, in 2018, the UPenn dataset [37] was introduced with the primary goal of validating stereo Visual-Inertial Odometry (VIO) methods for fast autonomous flight. However, the absence of a racing track with gates in the scene rendered it unsuitable for conducting comprehensive benchmarking of racing scenarios. The identification of gates through a monocular image is a critical perceptual step, even in human-piloted drone racing, as it allows the pilot to accurately localize themselves on the track and navigate it effectively at high speeds. In 2019, Lockheed Martin contributed to this field by releasing an image dataset for Test 2 of its AlphaPilot challenge's virtual qualifiers [15]. While this dataset served its purpose, it lacked essential drone state information, making it unsuitable for comprehensive dynamic perception tasks. In that same year, UZH introduced the UZH FPV Dataset [10]. This dataset marked a significant advancement in visual-inertial odometry benchmarking, being one of the most aggressive datasets available up to that date. It encompassed a variety of flight sequences, both indoor and outdoor, involving the navigation through gates.

These sequences were captured using a first-person-view (FPV) drone racing quadrotor equipped with sensors and flown aggressively by an expert pilot. However, it's worth noting that this dataset was characterized by relatively lower speeds and featured images with lower resolution and frequency compared to the growing demands of drone racing.

The Blackbird Dataset

The Blackbird dataset [1], introduced in 2020, was developed to facilitate comprehensive perception and control research in agile indoor flight scenarios. A notable feature was its fusion of real-world inertial data with high-resolution photorealistic images generated within the FlightGoggles [20] simulation environment. The dataset also included precise motion caputure ground truth of all the available trajectories.

Recent Contributions

Recent additions to the dataset landscape encompass [33] and [2], both specialized datasets catering to drone racing and aggressive multi-rotor flight. Additionally, [16] introduced an open-source, open-hardware racing drone, enhancing accessibility and reproducibility in this research domain.

A Novel Racing Dataset

Despite these commendable efforts, a dataset encompassing the entire spectrum of high-speed autonomous flights, featuring high-resolution, high-frequency RGB mono-camera images akin to those relied upon by human pilots, and offering comprehensive annotations, including precise gate corner labels, was absent until the creation of our dataset. Our dataset distinguishes itself from [10] not only by including piloted flights but also autonomous high-speed flights, thus reflecting the diverse operational modes of racing drones. Furthermore, it provides higher-resolution images captured under varying light conditions,

making it more representative of the challenging real-world scenarios encountered in drone racing. Most significantly, our dataset is meticulously annotated with high-frequency motion capture data, ensuring precise and comprehensive ground truth information for research purposes.

Distinctive Features of Our Dataset

In comparison to the existing literature [21], our dataset excels in several key dimensions. First, it represents the fastest autonomously flown dataset, pushing the boundaries of speed and maneuverability. Second, it offers high-resolution, high-frequency image data, capturing the subtleties of different lighting conditions, a vital factor in real-world racing scenarios. Lastly, our dataset is fully annotated at the granularity of individual gate corners. This empowers researchers to explore various facets of autonomous drone racing, including Visual-Inertial Odometry (VIO), gate pose estimation [11, 24] and and end-to-end control. In summary, our dataset provides a robust foundation for advancing research in AI-powered perception systems for high-speed autonomous racing drones.

2.2 Review of state estimation techniques

When it comes to state estimation techniques to localize autonomous aerial vehicles in the 3-dimensional space Visual-Inertial Odometry (VIO) stands out as the most common technique for the application, thanks to the fact that it relies only on onboard sensing and computing and that it represents a favorable trade-off between accuracy and computational requirements. Visual-Inertial Odometry (VIO) is a technique used in drone navigation, combining data from cameras and Inertial Measurement Units (IMUs) to estimate the drone's position, orientation, and velocity. While IMU measurements are integrated to provide short-term relative estimates rapidly, they accumulate significant errors over longer periods due to factors such as scale errors, axis misalignment,

and biases. On the other hand, camera measurements, albeit at a lower rate of around 30 Hz (in common low speed applications), offer rich environmental information but are susceptible to conditions like poor illumination, textureless scenes, and motion blur, which degrade their quality for state estimation. As a result, the combination of camera and inertial measurements is the standard choice for accurately estimating the state of flying vehicles.

2.2.1 Classical VIO methods

Classical VIO algorithms typically consist of two main components: the frontend and the backend.

The frontend utilizes camera images to estimate sensor motion. There are two primary approaches: direct methods [12], [4] and feature-based methods [30], [25], [35]. Direct methods work directly with raw pixel intensities, tracking image patches and estimating camera motion by minimizing photometric errors [12]. Feature-based methods extract visual features or keypoints from raw image pixels and estimate motion by tracking these points across images. While feature-based methods are more mature and robust, direct methods excel in low-texture environments. Hybrid methods, combining keypoints and pixel patches, also exist [17].

The backend combines frontend output with inertial measurements. There are two categories of methods in the literature: filtering methods and fixed-lag smoothing methods. Filtering methods, based on Extended Kalman Filters (EKFs), propagate the system state using inertial measurements and fuse camera measurements during updates. The pioneering filter-based VIO algorithm is the Multi-State Constraint Kalman Filter (MSCKF) [30], with various versions developed since [19]. Fixed-lag smoothing methods [25], [35], also known as sliding window estimators, solve nonlinear optimization problems involving recent robot states. Their cost functions include residuals from visual, inertial, and past states marginalization. Fixed-lag smoothing methods

accumulate less linearization error but are computationally demanding.

In the realm of drone racing, the combination of high speeds and vibrations generated by the quadcopter's motors poses a significant challenge to VIO systems. These conditions often lead to difficulties in handling blurred images and coping with substantial noise patterns present in the IMU data, making VIO systems alone unsuitable for the task due to their high drift and loss of feature tracking.

2.2.2 Drone racing specific approaches

Taking a step back to consider the human aspect of drone racing, it's important to highlight that pilots heavily depend on certain scene understanding cues when it comes to locating themselves within the racing track. Pilots possess prior knowledge about the appearance, shape, and dimensions of the gates, and they typically receive a visual representation of the track layout. Additionally, it's worth noting that they are able to navigate the track at high speeds through one single RGB image, often received through a noisy analog channel.

Hence, researchers in the autonomous drone racing domain began exploring solutions for gate detection and pose estimation challenges. Both the information regarding a bounding box encompassing the gate and the relative pose of the gate concerning the drone hold potential as input for subsequent state estimation, planning and control modules. One of the first notable works [22] for gate detection obtained the victory at the first autonomous drone race at IROS 2016, and was based on a modified Single Shot Detector (SSD) [28] Convolutional Neural Network CNN. After the detection of only the closest gate in the form of a bounding box (BB), the center of the bounding box was taken as a the next target point and the control module would act with with a Line of Sight (LOS) proportional guidance control, that aligned the center of the image plane to the center of the BB. Nonetheless, a limitation of this

method arises from the significant assumption that the center of the bounding box aligns precisely with the actual center of the gate. This assumption remains valid only under specific circumstances, such as when the camera's optical axis is orthogonal to the gate, and there is no presence of lens distortion. In practical real-world scenarios, these conditions are often not satisfied. Moreover, the approach relies on the strong assumption that at least one gate is always in sight. A significant milestone in the field of autonomous racing drones was achieved at IROS 2018 when Kaufmann et al. [24] secured victory using the first end-to-end image-to-pose approach. They employed a CNN with two separate Multi-Layer Perceptron (MLP) heads. One head was dedicated to regressing the relative pose of the closest gate, while the other provided an estimate of uncertainty. These two pieces of information were particularly valuable, as they were first used to build a gobal track layout representation with coarse gate locations during a single demonstration flight. At test time, a CNN predicted the poses of the closest gates along with their uncertainty. These predictions were incorporated by an EKF to maintain optimal maximum-aposteriori estimates of gate locations. The drone was then controlled using model predictive control (MPC), given the estimated gate poses.

The same authors built up another solution [15] starting from [24] which earned them the second position at the biggest autonomous drone racing competition ever hosted, the *Alphapilot Challenge* in 2019. In this case the gate detection step involved the identification of gates corners through a CNN and the association of them with the correct gate through the use of Part Affinity Fields (PAFs). Regarding the state estimation module, the output of the gate detection and VIO are fused together with the measurements from the downward-facing laser rangefinder (LRF) using an EKF. The EKF estimates a global map of the gates and, since the gates are stationary, uses the gate detections to align the VIO estimate with the global gate map, i.e. compensates for the VIO drift.

The winning team at the *Alphapilot Challenge* [11] secured victory with a similar approach in terms of gate detection and corner extraction, using GateNet, a fully convolutional network trained on a segmentation task in a supervised manner. GateNet produced as ouputs the gate masks which were then processed by a variant of the lightweight snake algorithm [26] to perform the next gate's corner extraction. Regarding the state estimation module De Wagter et al. drew inspiration from human pilots who focus greatly on the gates, while combining their observations with their knowledge of the drone's responses to control inputs and an approximate map of the track. For this reason, no state-of-the-art method for VIO was used, both because of their computational complexity and speed contraints (8 m/s was the top speed of the runners up team [15] due to VIO limits). Instead they combined the drone's pose estimation, obtained by solving a perspective-n-point (PnP) problem on the next gate corners positions and the approximative apriori known gate positions, with model-based predictions from a dynamic model fitted on flight data through a random sample consensus (RANSAC) based Moving Horizon Estimator (MHE) from a previous work [27].

While ongoing research continues on looking for fascinating learning based end-to-end solution for the perception-to-action loop, these examples highlight how in real world scenarios it is still often needed to have a customizable enough system to enable perception and control experts to adapt the autonomous modules to different racing scenarios.

Chapter 3

The Open-design Quadrotor and Autonomous Drone Racing Dataset

This thesis introduces a unique dataset, which is detailed in [5]. The dataset provides a versatile platform for comprehensive evaluations of Autonomous Drone Racing, encompassing various aspects such as perception systems, control, and dynamics. To acquire this dataset, we designed a custom quadrotor (shown in Figure 3.1). Our quadrotor design is centered around a 5" carbon-fiber frame with a diagonal propeller-to-propeller span of 215mm. The fully-assembled drone, including the battery, weighs approximately 870g and can achieve a top speed of 179km/h, enabling the aggressive maneuvers essential for drone racing. Importantly, our design seamlessly transitions between autonomous and human-piloted First-Person-View (FPV) racing, serving as an authentic benchmark for comparing autonomous drone performance against human pilots. The dataset encompasses both autonomous and piloted flights.

3.1 Platform Overview

The quadrotor comprises three primary sub-systems: (i) quadrotor electronics, (ii) the autonomous module, and (iii) the FPV system. These components are integrated using the frame and fasteners. The system incorporates two cameras: a digital camera connected to the autonomous module and an analog camera used by human pilots in the FPV system. The two cameras share a mount, with the FPV camera positioned above the digital one (as depicted in Figure 3.1).

Quadrotor Electronics

The quadrotor electronics include (i) the electronic speed controller (ESC), (ii) the Kakute H7 v1 flight controller unit (FCU), (iii) the radio controller (RC) receiver, and (iv) the battery. These components are mounted beneath the frame, protected by aluminum standoffs connecting the frame and a custom 3D-printed battery cage. The FCU features an STM32H7 microcontroller capable of running various firmware, including Ardupilot and PX4.

Autonomous Module

The autonomous module consists of (i) an NVIDIA Orin NX, hosted on the A203v2 carrier board with SSD and a wireless card, (ii) a battery eliminator circuit (BEC) to power it, and (iii) an Arducam RGB camera. These components are situated above the frame and are secured by two 3D-printed plates connected with aluminum standoffs. The top plate serves as the camera mount, and an MIPI CSI-2 ribbon cable links the companion board to the Arducam. The FCU connects via a serial port, employing a shielded cable for both drone control and sensor data retrieval.

FPV System

Operating independently from the autonomous module, the FPV system comprises an FPV analog camera, a video transmitter, and associated antennas, all mounted above the frame.

3.2 Sensors

Our quadrotor features a range of sensors to support autonomous and human-piloted aggressive flight:

InvenSense MPU6000 IMU

Embedded within the quadrotor electronics, the InvenSense MPU6000 IMU serves dual purposes. It provides real-time tri-axis angular rate sensor (gyroscope) data and accurate tri-axis accelerometer data. The autonomous companion computer retrieves raw IMU data through a demand/response exchange using the Multiwii Serial Protocol (MSP) [32].

Arducam IMX219 8MP RGB Bayer Camera

Part of the autonomous module (Section 3.1), the Arducam IMX219 captures 640x480 pixel frames at 120Hz, offering a diagonal field-of-view (FOV) of 175°. This camera, widely used for lightweight embedded applications, is fully supported by NVIDIA, with dedicated MIPI CSI-2 drivers. The image YUV frames are captured in NV12 format, converted to BGR for processing, and eventually saved as JPEG using the NVIDIA GStreamer plugin on the companion computer.

Foxeer T-Rex Mini 1500TVL

This low-latency (6ms) camera forms part of the FPV system. During data collection, it is used by human pilots in conjunction with a pair of 1280x960

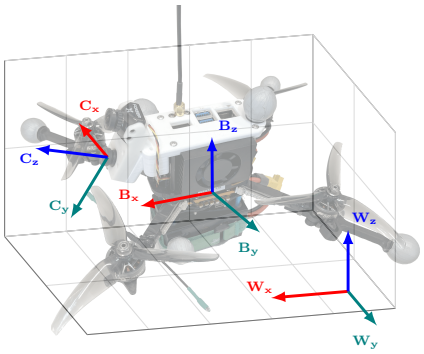


Figure 3.1: The drone platform used to record the dataset, the body frame B has its origin at the FCU's IMU location, the camera frame C is located where the bottom lens is (the top lens being the one of the FPV system).

OLED Fat Shark HDO2 goggles.

3.3 Software

Quadrotor Electronics Software

The quadrotor electronics (Section 3.1) run Betaflight 4.3.1 [3], featuring a tuned proportional-integral-derivative (PID) controller for flight. The companion computer employs MSP to send commands to the flight controller and read sensor data. Betaflight's `MSP_OVERRIDE` feature is activated to bypass RC controller commands while maintaining safety, allowing human supervisors to disarm the drone with an RC controller.

Autonomous Module Software

The autonomous module (Section 3.1) runs NVIDIA JetPack 5.1.1, encompassing Jetson Linux 35.3.1 Board Support Package (BSP) with Linux Real-Time Kernel 5.10 and an Ubuntu 20.04-based root file system with CUDA 11.4 support. The Robot Operating System 2 (ROS2) Humble LTS distribution serves as the middleware for communication among perception, planning, and control modules on the Orin NX module.

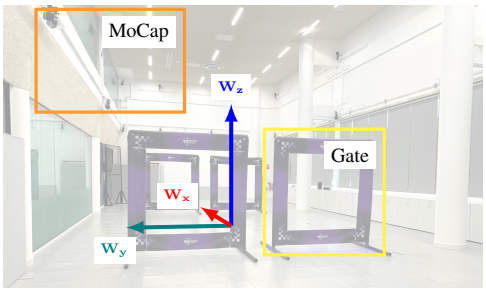


Figure 3.2: The 25×9.7×7 meters indoor arena, instrumented with 32 Qualisys MoCap cameras and equipped with four 5×5 feet racing gates used to record the dataset.

3.4 Data Collection

Flight Arena, Racing Gates, and Motion Capture System

The dataset was collected in an indoor flying arena measuring 25 meters in length, 9.7 meters in width, and 7 meters in height. The arena is equipped with a 32-camera Arqus A12 Qualisys Motion Capture (MoCap) system, capable of tracking 6-degree-of-freedom (6DoF) poses of defined rigid bodies with millimeter accuracy at a rate of 275Hz. To facilitate tracking, our quadrotor design (Section 3.1) is equipped with six 25mm markers defining a single rigid body. These markers are strategically placed on the top plate, battery cage, and 48.2mm arm extensions, ensuring they remain visible even during propeller motion (as depicted in Figure 3.1). The origin of the quadrotor's rigid body aligns with the FCU's IMU location, as illustrated in Figure 3.1. The IMU undergoes calibration using RC prior to each take-off [3]. Within the indoor arena, four racing gates (Figure 3.2) are constructed using PVC pipes covered by printed fabric banners. These gates measure 7 feet by 7 feet (213.36 cm) with an internal opening of 5 feet by 5 feet (152.4 cm), conforming to standards used in major drone racing leagues [31]. Each racing gate is defined by four markers placed at its inner corners.

Table 3.1: Summary of the flights recorded in the dataset

Control	Shape	Top Speed	Time	Distance
Autonomous	Ellipse†	**21.83 m/s**	149.32 s	526.15 m
	Lemniscate†	10.22 m/s	155.32 s	480.67 m
Piloted	Ellipse‡	9.50 m/s	575.62 s	3355.67 m
	Lemniscate‡	8.93 m/s	**594.60 s**	**3577.65 m**

†Flown twice in 6 flights (3 brightness × 2 camera settings). ‡Flown as many times as possible in 6 flights (3 brightness × 2 camera settings).

Flight Program

The dataset comprises a total of 24 flights (Table 3.1), evenly split between *human-piloted* and *autonomous* flights. For each category, two different flight shapes (*ellipse* and *lemniscate*) are executed six times, employing consistent gate configurations for each run. These six repetitions account for variations in illumination conditions and camera settings.

Brightness Levels

Data collection incorporates three levels of brightness: *high*, achieved with both natural and artificial light; *medium*, with controlled light by turning on artificial lights and blocking natural light using blinds; and *low*, attained by deactivating most arena lights and keeping blinds down.

Camera Settings

Two distinct camera settings are employed: *auto* exposure time and gains, and *fixed* exposure time, analog gain, and digital gain set to 2.5ms, 2, and 1, respectively. The *auto* setting produces brighter images but may introduce motion blur, while the *fixed* setting yields darker but less blurred images.

Human-piloted and Autonomous Control

Human-piloted flights range from 84 to 108 seconds, during which the pilot aims to maximize the number of laps on a single battery charge. In contrast, autonomous flights consist of precisely two laps, averaging a duration of 25.38 seconds. Autonomous flights employ a position controller based on [14], generating trajectory references using geometric formulas for position, heading, linear velocity, and linear acceleration. The motion capture system provides real-time quadrotor pose data to the controller via WiFi. The position controller applies a proportional-derivative (PD) control loop to calculate desired acceleration, thrust, and attitude. Subsequently, a proportional (P) control loop converts attitude into desired body rates. The human pilot employs a camera angle of 30° for both the FPV camera and the recorded Arducam. Autonomous flights utilize camera angles of 40° for lemniscate trajectories and 50° for ellipse trajectories. The choice of camera angles for autonomous mode ensures gate visibility at high speeds, similar to how FPV pilots adjust camera angles based on their planned speed.

Image Labeling

Gates play a crucial role in racing environments for relative localization and next-waypoint detection. This dataset provides image labels in the form of bounding boxes and keypoints corresponding to the inner corners of visible gates. Leveraging the quadrotor's inertial data and ground truth from the motion capture system, the dataset enables replication and benchmarking of state-of-the-art gate pose estimation [34] and the development of new methods. The labeling process initially involves automated labeling using a top-down keypoints detector [7, 9], trained on a synthetic dataset and fine-tuned with 5,000 manually labeled images. Subsequently, all images undergo iterative manual review and re-training. Finally, all labels receive manual verification. Bounding boxes and corner positions are provided for partially occluded

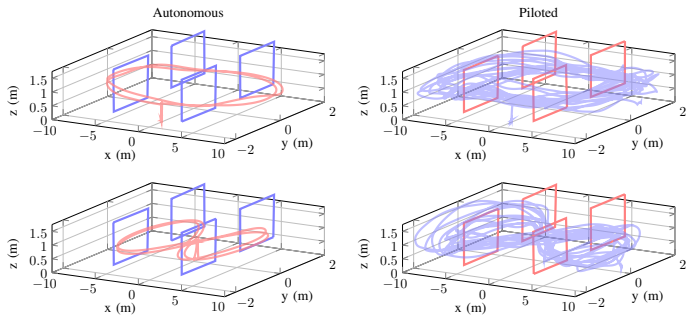


Figure 3.3: Examples of recorded trajectories on a 4-gate track: an autonomous ellipse `flight-01a-ellipse` (top-left); a piloted ellipse `flight-01p-ellipse` (top-right); an autonomous lemniscate `flight-07a-ellipse` (bottom-left); and a piloted lemniscate `flight-07p-ellipse` (bottom-right).

gates. However, no distinction is made between occluded and fully visible keypoints; instead, the dataset follows the COCO format definition [8], designating visibility values of 0 (outside image boundaries) and 2 (inside image boundaries) for keypoints. Gates are not labeled when no visible corners are present.

Time Synchronization

Data recording encompasses three separate streams: *(i)* a `rosbag` containing FCU readings and autonomous control setpoints, *(ii)* on-board camera images, and *(iii)* Qualisys motion capture measurements. For FCU data, custom ROS2 messages and the Real-Time Kernel are employed to minimize sensor reading jitter. The GStreamer pipeline saves images and timestamps them based on frame acquisition time. Synchronization involves two clocks: the real-time clock of the drone's companion computer and the clock of the Qualisys workstation. Both clocks are synchronized with a Network Time Protocol (NTP) server within the facility before each flight. Clock offsets relative to the NTP server for both machines are recorded before and after each flight to compute offset corrections and jitter. The drone achieves microsecond clock accuracy with Chrony, while the Qualisys workstation records millisecond accuracy.

The total jitter throughout a trajectory never exceeds 3ms.

Data Post-processing

Motion capture data are converted to CSV format, with onboard computer clock offsets removed. ROS2 bags are also converted to CSVs. All data are trimmed to eliminate pre-take-off and post-landing records. Data alignment is achieved using an open-source script that produces user-friendly comprehensive CSVs. Alignment involves linear interpolation for most fields, with spherical linear interpolation [36] applied to rotation matrices.

Chapter 4

Map-based state estimation

As discussed in Section 2.2, many different techniques have been used to enable drones to localize themselves in the 3d space and estimate their attitude, but let's take a step back and delve into how the perception-to-action loop works and what are the crtical challenges that need to be faced in a racing scenario. While flying through a racing track a quadrotor needs to perform all the steps that a human pilot would subconsciously perform explictly on its on board PC, based solely on the on board sensors, which in our case only consisted in a camera and an IMU. The main control perception-to-action loop comprises:

- Perception
- State estimation
- Planning
- Control

The primary hurdle in optimizing the racing setup lies in the critical factor of total execution time within the control loop. The demanding velocity requirements for the drone necessitate that the control loop operates within extremely tight timeframes, typically in the order of milliseconds. Adding to

this challenge is the limitation imposed by the hardware's constrained computing resources where these computations must be performed. In this and the next chapter, we will delve into the state estimation step of the loop, which involves gathering data from the onboard sensors to calculate the drone's current position and orientation in three-dimensional space.

4.1 Problem definition

This chapter is dedicated to the challenge of reconstructing the drone's position based solely on an image, a problem often referred to as Visual Odometry (VO). In recent years, researchers have approached this problem through various methods. Initially, geometric methods, as presented in [18] and [13], were employed. Later, learning-based approaches, such as [39] and [38], gained traction. However, these approaches struggled to handle the demanding conditions unique to drone racing, including high speeds, motion blur, and rapid trajectory variations.

It's important to emphasize that in the context of drone racing, the race track is predefined and known to the pilots in advance. Therefore, for the purpose of this study, we assume the availability of a map that provides precise information about the positions and orientations of the gates at all times during the flights.

Referring to Figure 3.1, our problem involves three distinct reference frames: the world reference frame **W**, which is the frame we aim to use for estimating the drone's position; the body reference frame of the drone **B**; the camera reference frame **C**; and the gates' reference frames **G**. It's crucial to note that the transformation matrix between the body reference frame and the camera reference frame, denoted as $\mathbf{H}_B^C$[1], is fixed and known apriori, whereas our goal is to estimate $\mathbf{H}_B^W$.

[1]From this point onward, all transformation matrices $\mathbf{H}$ and rotation matrices $\mathbf{R}$ will be denoted in accordance with the bottom-up convention, i.e., $\mathbf{H}_B^C$ signifies the rototranslation of the body frame B with respect to the camera reference frame C.

4.2 Estimation pipeline

The estimation pipeline commences with the capture of an image using the drone's camera. This image is subsequently subjected to processing by a convolutional neural network, specifically YoloV8, which is responsible for extracting the four inner corners of the gates. It's important to clarify that the scope of this work is focused on the pipeline beginning with the pixel coordinates of the gate's corners as the initial input. These obtained corner coordinates are then utilized as input for a Perspective-n-Point (PnP) algorithm. This algorithm computes the rototranslation of the gate in relation to the camera's reference frame, denoted as $\mathbf{H}_G^C$.

4.2.1 Perspective and Point

PnP, or Perspective-n-Point, is a computer vision algorithm used to determine the 3D pose of an object based on its 2D image projections. In our case, the object of interest is a gate on the racing track, and the 2D image projections are the pixel coordinates of the gate's corners in the image captured by the drone's camera.

The core principle behind the PnP algorithm lies in the knowledge of both 2D image points and the size (or scale) of the object in 3D space. These correspondences, along with the intrinsic and distortion camera parameters, allow PnP to compute the transformation between the gate's reference frame and the camera's reference frame.

Let's break down the PnP process step by step:

1. **Image Points**: The first step involves identifying and extracting the pixel coordinates of the gate's corners in the image. This is typically achieved through computer vision techniques, such as the gate corner extraction performed by YoloV8 in our pipeline.

2. **Size Information**: For the PnP algorithm to work, we need to know the

size (scale) of the gate in the 3D world. This size information is critical for accurate pose estimation and navigation. Fortunately, in drone racing scenarios, the size of the gates is typically known in advance (ca. 1.5x1.5m in our case).

3. **Camera Parameters**: Additionally, we need to be aware of the intrinsic and distortion parameters of the camera used on the drone. These parameters describe the camera's internal characteristics, such as its focal length, principal point, and lens distortion.

4. **PnP Computation**: With the 2D image points, the gate's size information, and intrinsic camera parameters in hand, the PnP algorithm computes the transformation matrix $\mathbf{H}_G^C$, which represents the rototranslation of the gate with respect to the camera reference frame (**C**). This matrix includes both the rotation and translation information, allowing us to precisely determine the gate's pose in 3D space with respect to the camera reference frame.

Following the completion of the computation, a crucial initial filtering step is implemented to mitigate the impact of potentially inaccurate estimates. The three-dimensional points corresponding to each gate undergo a reprojection onto the image plane based on the estimated rototranslation. These reprojected points are then subjected to a filtering process guided by a predetermined threshold, effectively enhancing the precision of the results by eliminating noisy estimates. Having this information, the pipeline can proceed with the identification of the gates in sight and the subsequent global position estimation.

Figure 4.1: Examples of poses estimated via PnP on dataset images.

4.2.2 Gate identification

To accurately reconstruct the drone's current position on the race track, it is essential to correctly identify the gates in the image by matching them with the map of the race track. This matching step assumes the availability of the last known rototranslation of the drone with respect to the world reference frame, denoted as $\mathbf{H}_B^W$, which will be recursively estimated. At the beginning of the flight, it can be assumed that the starting position and orientation of the drone are known a priori. The initial phase of the matching process entails transforming the gate's position and orientation into the world reference frame using the following transformation:

$$\mathbf{H}_G^W = \mathbf{H}_B^W \mathbf{H}_C^B \mathbf{H}_G^C$$

Here, $\mathbf{H}_G^C$ is the result of the PnP computation and $\mathbf{H}_C^B$ is a fixed transformation defined by the physical structure of the drone. With this transformation result, we can now compare it with the exact gate locations provided in the map of the race track. The objective is to find the closest matching gate, based on a minimum Euclidean distance criterion, considering a specified threshold. Moreover, the relative rotation between the estimated orientation $\mathbf{H}_G^W$ and the

ground truth one $\mathbf{H}_{\hat{G}}^{W}$ is computed as:

$$\mathbf{H}_{\hat{G}}^{\hat{G}} = \mathbf{H}_{W}^{\hat{G}}\mathbf{H}_{G}^{W}$$

This is used to filter out wrongly oriented detections, by applying a threshold to the angle represented by the rotational part of $\mathbf{H}_{\hat{G}}^{\hat{G}}$, enhancing the quality of the subsequent reconstruction. Moreover, to address the scenario where the drone navigates through gates in varying orientations, we apply this filtering considering a $180°$ modulo of the aforementioned angle.

4.2.3 Final position reconstruction

Having identified each gate in sight we proceed by deriving their ground truth rototranslation with respect to the camera, namely $\mathbf{H}_{\hat{G}}^{C}$, through the following relation:

$$\mathbf{H}_{\hat{G}}^{C} = \mathbf{H}_{W}^{C}\mathbf{H}_{\hat{G}}^{W}$$

Where $\mathbf{H}_{\hat{G}}^{W}$ represents the ground truth position and orientation of the gate in the world frame, extracted from the map. Subsequently, we can proceed with the final position reconstruction. For this we are going to use the estimated positional offset of the gate in the camera reference, expressed in homogeneous coordinates $t_g \in \mathbb{R}^4$, and the rotational part of $\mathbf{H}_{\hat{G}}^{C}$, expressed as $\mathbf{R}_{\hat{G}}^{C} \in \mathbb{R}^{3\times3}$. The transformation matrix that combines t_g with $\mathbf{R}_{\hat{G}}^{C}$ is obtained through the following:

$$\mathbf{H}_{C}^{\hat{G}} = \begin{bmatrix} \mathbf{R}_{\hat{G}}^{C^T} & 0^{3\times1} \\ 0^{1\times3} & 1 \end{bmatrix} \begin{bmatrix} -t_g \end{bmatrix},$$

Finally, for each gate in sight we obtain an estimate of the drone position given by

$$\mathbf{H}_{B}^{W} = \mathbf{H}_{\hat{G}}^{W}\mathbf{H}_{C}^{\hat{G}}\mathbf{H}_{B}^{C}$$

The primary source of measurement error arises from the uncertainty in detecting gate corners in the gate detector. This error, occurring in the image

plane, leads to a pose error when the PnP algorithm is applied. To address this, we have adopted a sampling-based approach to estimate the pose error based on the known average uncertainty in gate corner detection. For each gate, the PnP algorithm is employed not only on the nominal gate observation but also on 20 perturbed estimates of gate corners. The resulting distribution of pose estimates is then utilized to approximate the measurement covariance associated with the gate observation, which will be used later in the pipeline.

Chapter 5

Kalman Filter for VIO Drift Correction

When it comes to estimating the state of autonomous drones, researchers face a crucial trade-off between accuracy and the need for a fast and agile platform. Achieving high accuracy often requires more sensors, increasing the weight and volume of the drone and impacting its maneuverability. This trade-off is particularly challenging in the context of drone racing, where small and lightweight drones are essential. Drone racing demands drones that are not only swift and agile but also capable of navigating intricate courses with precision. Achieving the necessary speed and agility while maintaining accuracy in state estimation poses a formidable task. An additional challenge arises from the inherent noise in sensor data. In the high-speed, high-vibration environment of drone racing, sensors are susceptible to noise and interference, often resulting from intense vibrations and magnetic disturbances generated by the drone's motors. The noisy sensor data significantly affects the accuracy of state estimation, emphasizing the need for sophisticated and robust estimation techniques.

In this chapter, we delve into the application of the Kalman Filter (KF) for state estimation in racing drones. The Kalman Filter is a recursive Bayesian estimation method that combines sensor data with a dynamic model to estimate

the drone's state, incorporating uncertainty and correcting errors in measurements. Our exploration commences with an exploration of the foundational principles of the Kalman Filter (KF), a robust methodology tailored for precise state estimation in dynamic environments. We delve into its algorithmic underpinnings, accentuating the predictive and corrective steps. The KF's distinctive capability to iteratively enhance state estimates, accounting for uncertainties, positions it as a dependable choice for navigating dynamic scenarios such as drone racing.

In our pursuit of heightened accuracy and resilience in state estimation, we delve into the integration of diverse data sources. A deliberate choice was made to avoid the implementation of a complete Extended Kalman Filter, opting instead for efficiency and process streamlining. This chapter focuses on the amalgamation of Visual-Inertial Odometry (VIO) data and gate-based estimation within a Kalman Filter framework. This integration aims to estimate and rectify VIO drift, following the approach proposed by Kaufmann et al. [23].

Given the infrequent occurrence of gate detections and the precision of orientation estimates from VIO, our attention centers on refining the translational components of VIO measurements. The Kalman Filter is harnessed to estimate both the translational drift p_d (position offset) and its derivative, the drift velocity v_d. These estimates are iteratively employed to rectify the present state estimate, contributing to an overall enhancement in accuracy throughout the state estimation process.

5.1 Formulation

The Kalman Filter (KF) iteratively refines state estimates, accounting for uncertainty in measurements and predictions. It operates by generating an approximation of the system's state through a weighted average of the predicted

state and the latest measurement. This weighted average is determined by assigning weights based on the quality of the estimated uncertainty, giving more trust to values with smaller uncertainties. The weights are derived from the covariance, a metric indicating the anticipated uncertainty in predicting the system's state. The outcome of this weighted average yields a refined state estimate positioned between the predicted and measured states, characterized by a lower estimated uncertainty compared to either state in isolation.

This iterative process unfolds at each time step, with the updated estimate and its covariance influencing the subsequent prediction in a recursive fashion. Notably, the Kalman filter relies solely on the most recent "best guess" rather than the entire historical record of a system's state to iteratively compute a new state.

5.1.1 Prediction and Update Steps

The KF consists of two fundamental steps: prediction and update.

Prediction Step

In the prediction step, the KF uses the dynamic model of the system to estimate the value of the state vector at the next time step. In our case, the state vector is $\mathbf{x} = [\mathbf{p}_d^T, \mathbf{v}_d^T]^T \in \mathbb{R}^6$, and the prediction consists of estimating the amount of drift and its velocity at the next timestep, as per the following equation:

$$\mathbf{x}_{k+1} = F\mathbf{x}_k,$$

$$F = \begin{bmatrix} \mathbb{I}^{3\times3} & \mathbf{d}t\mathbb{I}^{3\times3} \\ 0^{3\times3} & \mathbb{I}^{3\times3} \end{bmatrix},$$

which merely express the fact that drift velocity $\mathbf{v}_d$ is the derivative of the translational drift $\mathbf{p}_d$

However, as real-world systems are often subject to unpredictable disturbances, the prediction also accounts for process noise according to the following equation:

$$P_{k+1} = FP_kF^T + Q,$$

$$Q = \begin{bmatrix} \frac{1}{4}\mathbf{d}t^4 & \frac{1}{2}\mathbf{d}t^3 \\ \frac{1}{2}\mathbf{d}t^3 & \mathbf{d}t^2 \end{bmatrix} \sigma_a^2$$

Where P represents the state's covariance, which is initialized to zero, as done for the state and σ_a^2 is the covariance of the noise process characterizing the acceleration of the drift.

Update Step

In the update step, the KF considers measurement noise and the likelihood of the observed measurements, aligning the predicted state with the actual measurements. The KF's update step is crucial for continuously refining the state estimate based on new data. In this case, for each measurement $\mathbf{z}_k$, representing the measured drift between a pose estimate obtained from a gate detection and a VIO estimate, the predicted VIO drift $\mathbf{x_k}^-$ is adjusted to the corrected estimate $\mathbf{x}_k^+$ using the Kalman filter equations:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1},$$

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + K_k(\mathbf{z}_k - H(\mathbf{x}_k^-)),$$

$$P_k^+ = (I - K_k H_k) P_k^-,$$

in which K_k is the Kalman gain, R is the measurement covariance matrix, estimated via Monte-Carlo sampling as described in the previous chapter, and H_k is the measurement matrix. When several gates are detected in a single camera frame, all relative pose estimates are stacked and processed in the same Kalman filter update step.

5.2 Deployment

During the deployment phase of this research, the code for the state estimation pipeline had to be encapsulated in a ROS 2 C++ node. This step was essential to align with the remainder of the drone's software stack, which operates on an Nvidia Jetson Orin NX embedded on the drone. The input sources for the state estimation node are two different topics: the VIO topic and the synchronized image-VIO topic. With the data coming from these two topics the state estimation node implements the following pipeline:

- undistortion of the image;
- gate detection and corner extraction, through the use of a separate library implementing the inference via TensorRT;
- solve PnP and compute drift estimates;
- perform drift estimation KF prediction and updates, through the use of a separate library implementing the KF formulation, described in Section 5.1;
- publish the corrected state estimate on a topic, which is then used by the controller.

As is common within the ROS framework, the management of various data streams is performed utilizing callback functions. Specifically, in handling the disparate frequencies of data from the two sources, prediction steps are performed each time a new VIO message is available, while updates are only triggered when a synchronized VIO-image message is delivered, in order to maintain synchronization and enhance the overall processing efficiency. This approach serves to maintain a stable and reliable frequency for the control system, enabling it to operate effectively even in scenarios where detection messages may experience delays.

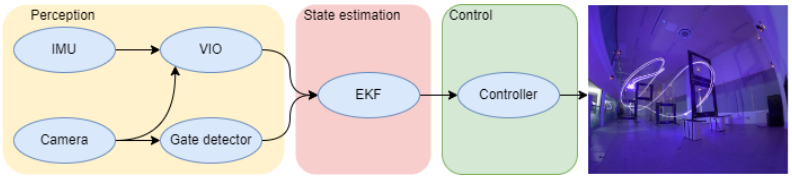


Figure 5.1: Pipeline overview.

5.3 Experiments

Throughout the development of the aforementioned module, the dataset presented in Chapter 3 played a pivotal role. This dataset facilitated essential preliminary tests using ground truth data, encompassing both image labels and the drone's position and orientation obtained through motion capture. To assess the performance of the state estimation pipeline, a series of test flights were conducted using an alternative platform, distinct from the one outlined in Section 3.1. This modified platform retained core components such as the flight controller and autonomous model but underwent significant alterations. Both the FPV camera, Arducam, and video transmitter were removed, and in their place, an Intel Realsense T265 stereo camera was integrated. The integration of this camera, facilitated by the Realsense SDK and ROS wrapper, enabled the streaming of Visual-Inertial Odometry (VIO) estimates via a ROS topic. Subsequently, these estimates were employed by the state estimation module, as described earlier. It is noteworthy that the use of the Intel Realsense T265 introduced additional challenges. The camera captures highly distorted fish-eye grayscale images, which posed complexities during Yolo model training and resulted in reduced accuracy in the Perspective-n-Point (PnP) estimates.

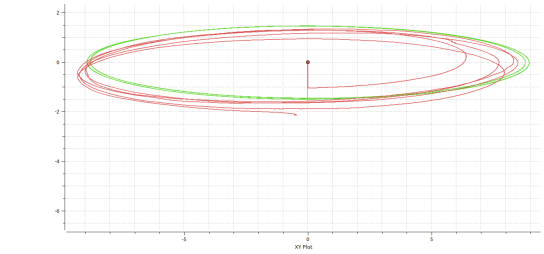
The test flights were carried out within the indoor arena detailed in Section 3.4, where gates were arranged to form a track. Before each flight, the precise positions of the gates were meticulously measured using the Qualisys motion capture system, which provided millimeter-level accuracy. These measurements were then stored as a race track map, serving as a reference for the state estimation module during flight operations. While flying, using a Model

Predictive Control (MPC) algorithm as our control algorithm, we recorded rosbags containing data from the following topics:

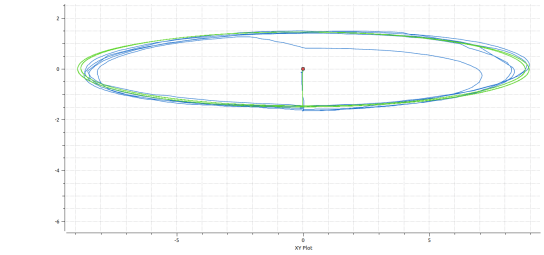
- Qualisys motion capture,
- image topic,
- VIO,
- drift corrected state estimation.

5.3.1 Test flight 1

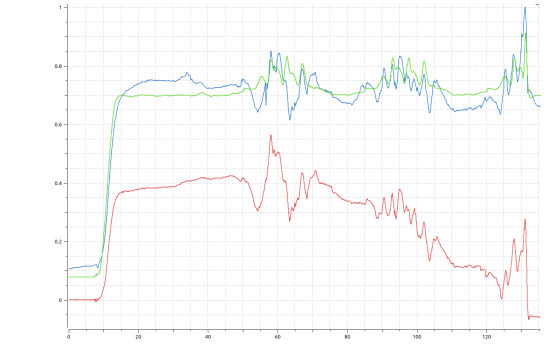
This test flight involved the drone flying through gates in a simple ellipse trajectory, with a top speed of 10 m/s. Being it a preliminary test flight the control module was still relying on the motion capture ground truth data and the state estimation computation has been performed offline on a laptop. As you can see from the plots in 5.2 the VIO estimate is largely affected by drift on all the axes, especially at the beginning of the flight. Moreover, it is also noticeable how the drift doesn't evolve regularly across time. As a consequence, the dynamic model present in the KF was unable to adequately fit the data. Subsequent to this observation, a decision was made to deactivate the drift velocity estimate in the filter, leading to higher precision tracking. Indeed, after just half a lap, the drift-corrected state published by the state estimation module almost totally compensates for the VIO offset, approaching the ground truth. This delay is attributed to the necessity of some initial movement for the drift estimator to commence convergence to a reasonable estimate, a factor unknown at the time this recording was made.



(a) VIO vs Mocap



(b) VIO vs KF



(c) **z** axis

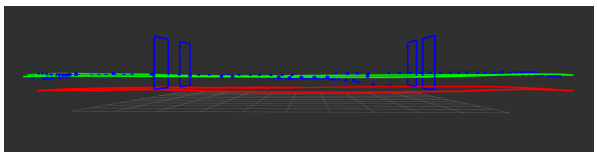
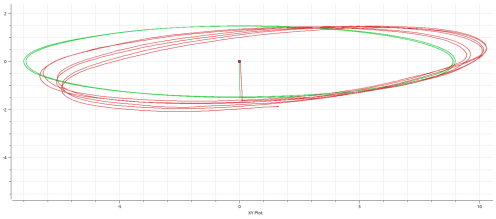


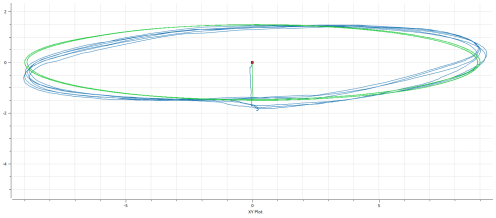
Figure 5.2: Flight 1. Red represents VIO data, blue drift corrected state, green Mocap ground truth data

5.3.2 Flight 2

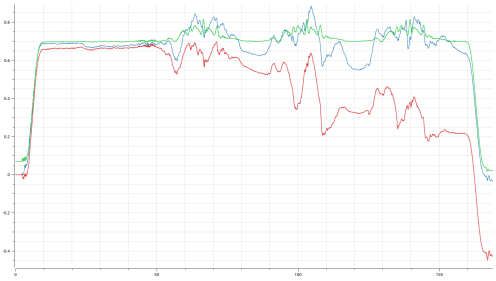
In the second flight, the drone followed the same trajectory as in Flight 1 but achieved a higher top speed of 12 m/s. As illustrated in the plots in Figure 5.3, VIO drift was more pronounced in both the x and y axes, although it exhibited less severity along the z-axis. Similar to the preceding flight, the controller depended on mocap position tracking, and the state estimation node was executed offline on a laptop.



(a) VIO vs Mocap



(b) VIO vs KF



(c) **z** axis

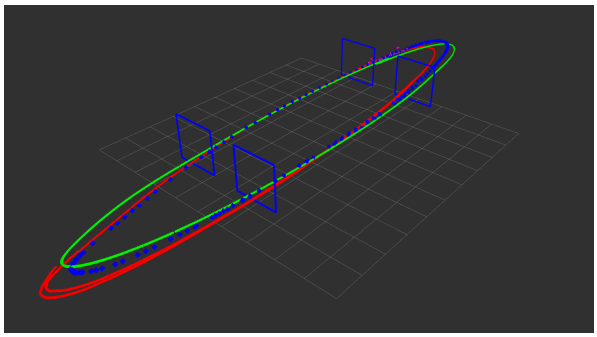
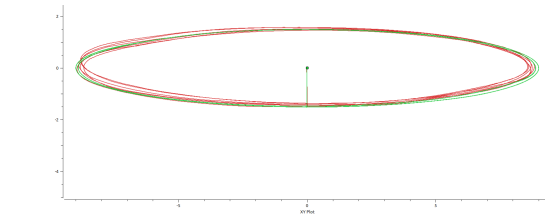


Figure 5.3: Flight 2. Red represents VIO data, blue drift corrected state, green Mocap ground truth data

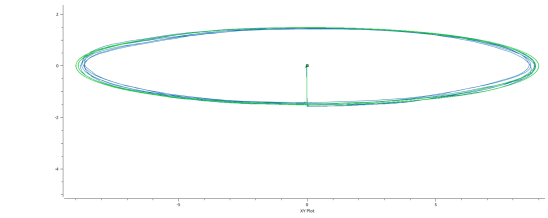
5.3.3 Flight 3

In this flight, the setup remains consistent with the previous flights, with the exception that the state estimation computation is now performed on the Nvidia Orin NX mounted on the drone. Meanwhile, the controller still relies on motion capture data. As evident from the plots in 5.4, the correction concerning

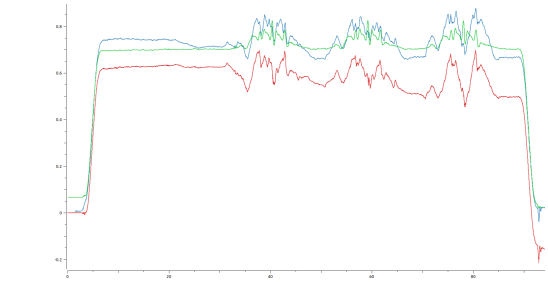
the VIO is highly effective across all axes, with notable improvement, particularly along the z-axis. With all conditions met to enable a fully vision-based flight, we proceeded to implement this in Flight 4.



(a) VIO vs Mocap



(b) VIO vs KF



(c) **z** axis

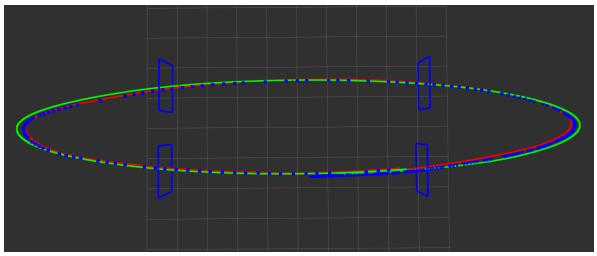
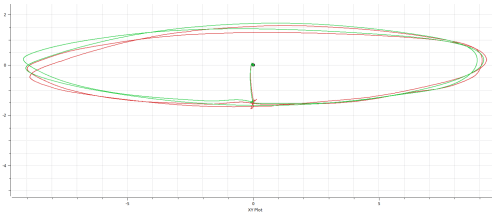


Figure 5.4: Flight 3. Red represents VIO data, blue drift corrected state, green Mocap ground truth data

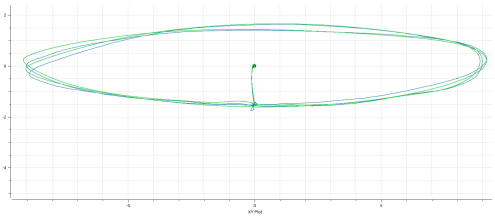
5.3.4 Flight 4

This final flight marked our inaugural fully vision-based flight on a real track with gates, following an elliptical trajectory at 8 m/s. Despite the VIO data exhibiting greater consistency in this run due to the lower speed, the discernible

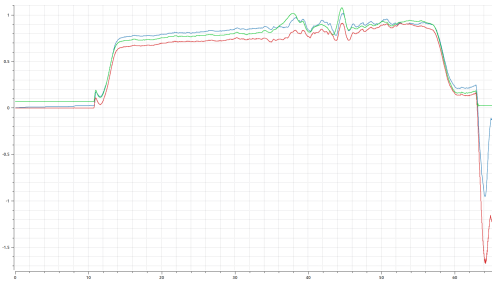
impact of the drift-corrected state estimation is evident from the plots in Figure 5.5, particularly on the z-axis.



(a) VIO vs Mocap



(b) VIO vs KF



(c) **z** axis

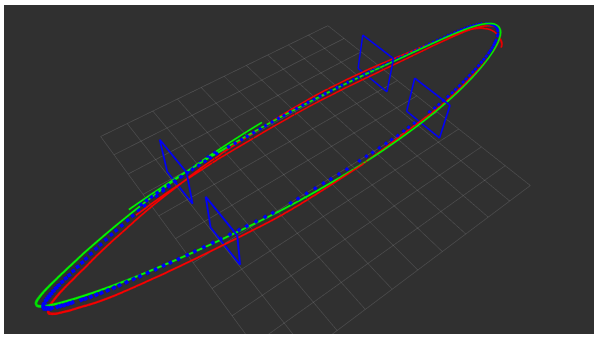


Figure 5.5: Flight 4. Red represents VIO data, blue drift corrected state, green Mocap ground truth data

Table 5.1: Summary of the flights

Flight	Max speed	Estimate	ATE	Max Error	Min Error
Flight 1	10 m/s	VIO	0.71 m	2.61 m	0.08 m
		KF	0.24 m	1.97 m	0.02 m
Flight 2	12 m/s	VIO	0.90 m	2.06 m	0.04 m
		KF	0.28 m	0.65 m	0.04 m
Flight 3	12 m/s	VIO	0.18 m	0.44 m	0.06 m
		KF	0.11 m	0.34 m	0.01 m
Flight 4	8 m/s	VIO	0.17 m	1.73 m	0.05 m
		KF	0.11 m	1.04 m	0.02 m

From Table 5.1, it is evident that the drift was successfully corrected in each flight, leading to a notable reduction in Absolute Trajectory Error (ATE) compared to using VIO alone.

In conclusion, as of the writing of this thesis, experiments are still in progress, involving incremental increases in speed and track complexity. The objective is to determine the true limitations of this system and eventually deploy it in a real autonomous drone race.

Chapter 6

Conclusion and Future Work

In this master's thesis, the primary focus centered around the domain of autonomous drone racing, specifically targeting perception and state estimation systems. The complex challenges within this context necessitate rapid, efficient, and precise solutions to optimize performance, ensuring agile drone control throughout the race track and maximizing speed.

The initial phase of the research involved the creation of a comprehensive, multimodal dataset tailored for high-speed drone racing. This meticulously annotated dataset is expected to play a pivotal role in advancing research within the field. Overcoming challenges in recording multimodal data required the development of an open-source accessible platform and the synchronization of an external motion capture system with onboard sensor data from the camera and IMU. An innovative aspect of the dataset is the inclusion of recordings from both piloted and autonomous flights.

The second major contribution of this thesis pertains to the design and implementation of a state estimation module. This module empowers the drone to autonomously navigate using solely onboard sensors and computation, eliminating reliance on external motion capture systems. Overcoming the limitations of state-of-the-art Visual-Inertial Odometry (VIO) systems, particularly in the extreme conditions encountered in drone racing, was a notable achievement. This was accomplished by leveraging prior knowledge of

the gates and track layout. Additionally, in the interest of reliability and efficiency, a simplified Kalman Filter for VIO drift estimation and correction was developed, instead of employing a full-state Extended Kalman Filter for sensor fusion. The described state estimation system underwent initial validation using motion capture data in lieu of VIO on the dataset. Subsequently, it achieved successful deployment for fully autonomous flights, marking a significant advancement in the realm of autonomous drone racing.

While the current work provides a solid foundation, several avenues for future research and development can be explored. One promising direction is the exploration of end-to-end perception-to-action training, allowing the drone to learn directly from raw sensor inputs to execute complex maneuvers. This approach may further enhance the adaptability of the system to unforeseen racing conditions. Additionally, further improvements to the state estimation system could involve the integration of advanced machine learning techniques to enhance the robustness and accuracy of position and orientation estimation. Investigating the potential of reinforcement learning for refining control policies in dynamic racing environments is another avenue worth exploring. Moreover, expanding the dataset to include diverse racing scenarios, environmental conditions, and hardware setups would contribute to the generalizability and applicability of the developed systems. The inclusion of more challenging racing tracks and varying lighting conditions can aid in pushing the boundaries of system performance. In summary, this thesis establishes the groundwork for perception-aware state estimation in autonomous drone racing. The potential for future work in these directions not only holds promise for pushing the boundaries of what is achievable in this exciting and challenging field but also underscores the applicability of the findings in diverse environments beyond racing scenarios.

Bibliography

[1] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman. The blackbird dataset: a large-scale dataset for uav perception in aggressive flight. In J. Xiao, T. Kröger, and O. Khatib, editors, *Proceedings of the 2018 International Symposium on Experimental Robotics*, pages 130–139, Cham. Springer International Publishing, 2020. ISBN: 978-3-030-33950-0.

[2] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza. NeuroBEM: hybrid aerodynamic quadrotor model. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, July 2021. DOI: 10.15607/rss.2021.xvii.042.

[3] Betaflight. The betaflight open source flight controller firmware project. https://github.com/betaflight/betaflight.

[4] 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)*, pages 298–304, 2015. DOI: 10.1109/IROS.2015.7353389.

[5] M. Bosello, D. Aguiari, Y. Keuter, E. Pallotta, S. Kiade, G. Caminati, F. Pinzarrone, J. Halepota, J. Panerati, and G. Pau. Race against the machine: a fully-annotated, open-design dataset of autonomous and piloted high-speed flight, 2023. arXiv: 2311.02667 [cs.RO].

[6] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets.

The International Journal of Robotics Research, 35(10):1157–1163, 2016. DOI: 10.1177/0278364915620033.

[7] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. Mmdetection: open mmlab detection toolbox and benchmark, 2019. arXiv: 1906.07155 [cs.CV].

[8] COCO - Common Objects in Context - Data format. URL: https://cocodataset.org/#format-data. [Accessed 30. Aug. 2023].

[9] M. Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020.

[10] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6713–6719, 2019. DOI: 10.1109/ICRA.2019.8793887.

[11] C. de Wagter, F. Paredes-Vallés, N. Sheth, and G. C. de Croon. The artificial intelligence behind the winning entry to the 2019 ai robotic racing competition. *ArXiv*, abs/2109.14985, 2021. URL: https://api.semanticscholar.org/CorpusID:238227128.

[12] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry, 2016. arXiv: 1607.02565 [cs.CV].

[13] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018. DOI: 10.1109/TPAMI.2017.2658577.

[14] M. Faessler, A. Franchi, and D. Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2018. DOI: 10.1109/LRA.2017.2776353.

[15] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza. Alphapilot: autonomous drone racing. *Autonomous Robots*, 46:307–320, 2020. URL: https://api.semanticscholar.org/CorpusID:218889286.

[16] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza. Agilicious: open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 7(67):eabl6259, 2022. DOI: 10.1126/scirobotics.abl6259. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abl6259.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014. DOI: 10.1109/ICRA.2014.6906584.

[18] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017. DOI: 10.1109/TRO.2016.2623335.

[19] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang. Openvins: a research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020. DOI: 10.1109/ICRA40945.2020.9196524.

[20] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman. FlightGoggles: photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality. In *2019 IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS). IEEE, November 2019. DOI: 10.1109/iros40897.2019.8968116. URL: https://doi.org/10.1109%2Firos40897.2019.8968116.

[21] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Pěnika, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza. Autonomous drone racing: a survey. *ArXiv*, abs/2301.01755, 2023. URL: https://api.semanticscholar.org/CorpusID:255440725.

[22] S. Jung, S. Hwang, H. Shin, and D. H. Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, 2018. DOI: 10.1109/LRA.2018.2808368.

[23] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, August 2023. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06419-4. URL: https://doi.org/10.1038/s41586-023-06419-4.

[24] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Beauty and the beast: optimal methods meet learning for drone racing. *2019 International Conference on Robotics and Automation (ICRA)*:690–696, 2018. URL: https://api.semanticscholar.org/CorpusID:53114276.

[25] 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*, 34, February 2014. DOI: 10.1177/0278364914554813.

[26] S. Li, M. M. O. I. Ozo, C. de Wagter, and G. C. de Croon. Autonomous drone race: a computationally efficient vision-based navigation and control strategy. *Robotics Auton. Syst.*, 133:103621, 2018. URL: https://api.semanticscholar.org/CorpusID:52284283.

[27] S. Li, E. van der Horst, P. Duernay, C. D. Wagter, and G. C. H. E. de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. *CoRR*, abs/1905.10110, 2019. arXiv: 1905.10110. URL: http://arxiv.org/abs/1905.10110.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg. Ssd: single shot multibox detector. In *European Conference on Computer Vision*, 2015. URL: https://api.semanticscholar.org/CorpusID:2141740.

[29] A. L. Majdik, C. Till, and D. Scaramuzza. The Zurich urban micro aerial vehicle dataset. *The International Journal of Robotics Research*, 36(3):269–273, 2017. DOI: 10.1177/0278364917702237. eprint: https://doi.org/10.1177/0278364917702237.

[30] 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*, pages 3565–3572, 2007. DOI: 10.1109/ROBOT.2007.364024.

[31] MultiGP. Multigp drone racing gate. https://www.multigp.com/product/drone-racing-gate-bundle/.

[32] MultiWii. Multiwii serial protocol. http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol.

[33] C. Pfeiffer and D. Scaramuzza. Human-piloted drone racing: visual processing and control. *IEEE Robotics and Automation Letters*, 6(2):3467–3474, 2021. DOI: 10.1109/LRA.2021.3064282.

[34] H. X. Pham, H. I. Ugurlu, J. L. Fevre, D. Bardakci, and E. Kayacan. Deep learning for vision-based navigation in autonomous drone racing. *Deep Learning for Robot Perception and Cognition*, 2022.

[35] T. Qin, P. Li, and S. Shen. VINS-mono: a robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, August 2018. DOI: 10.1109/tro.2018.2853729. URL: https://doi.org/10.1109%2Ftro.2018.2853729.

[36] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 245–254, New York, NY, USA. Association for Computing Machinery, 1985. ISBN: 0897911660. DOI: 10.1145/325334.325242.

[37] 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*, 3(2):965–972, 2018. DOI: 10.1109/LRA.2018.2793349.

[38] S. Wang, R. Clark, H. Wen, and N. Trigoni. DeepVO: towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017. DOI: 10.1109/icra.2017.7989236. URL: https://doi.org/10.1109%2Ficra.2017.7989236.

[39] W. Wang, Y. Hu, and S. Scherer. Tartanvo: a generalizable learning-based vo, 2020. arXiv: 2011.00359 [cs.CV].

Acknowledgements

I express my biggest gratitude to my supervisor, Prof. Giovanni Pau, and to Technology Innovation Institute for providing me with the chance to participate in this internship.

I'm very grateful to all my former and current team members at TII: Enrico, Davide, Sara, Gyordan, Michael, Davide, Junaid, KeiLong, Lessing, Aaesha and Yvo.