Aerial Robot Formations for Dynamic Environment Perception

Aerial Robot Formations for Dynamic Environment Perception

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Dipl. Inf. Jörn Eric Price

aus Friedrichshafen

Tübingen

2025

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	02.10.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Michael J. Black
2. Berichterstatter:	Jun.-Prof. Dr.-Ing. Aamir Ahmad
3. Berichterstatter:	Prof. Dr.-Ing. Hendrik P. A. Lensch
4. Berichterstatter:	Prof. Dr. Pedro Lima

Abstract

Perceiving moving subjects, like humans and animals, outside an enclosed and controlled environment in a lab is inherently challenging, since subjects could move outside the view and range of cameras and sensors that are static and extrinsically calibrated. Previous state-of-the-art methods for such perception in outdoor scenarios use markers or sensors on the subject, which are both intrusive and unscalable for animal subjects. To address this problem, we introduce robotic flying cameras that autonomously follow the subjects. To enable functions such as monitoring, behaviour analysis or motion capture, a single point of view is often insufficient due to self-occlusion, lack of depth perception and coverage from all sides. Therefore, we propose a team of such robotic cameras that fly in formation to provide continuous coverage from multiple view-points. The position of the subject must be determined using markerless, remote sensing methods in real time. To solve this, we combine a convolutional neural network-based detector to detect the subject with a novel cooperative Bayesian fusion method to track the detected subject from multiple robots. The robots need to then plan and control their own flight path and orientation relative to the subject to achieve and maintain continuous coverage from multiple view-points. This, we address with a model-predictive-control-based method to predict and plan the motion of every robot in the formation around the subject. A preliminary demonstrator is implemented with multi-rotor drones. However, drones are noisy and potentially unsafe for the observed subjects. To address this, we introduce non-holonomic lighter-than-air autonomous airships (blimps) as the robotic camera platform. This type of robot requires dynamically constrained orbiting formations to achieve omnidirectional visual coverage of a moving subject in the presence of wind. Therefore, we introduce a novel model-predictive formation controller for a team of airships. We demonstrate and evaluate our complete system in field experiments involving both human and wild animals as subjects. The collected data enables both human outdoor motion capture and animal behaviour analysis. Additionally, we propose our method for autonomous long-term wildlife monitoring. This dissertation covers the design and evaluation of aerial robots suitable to this task, including computer vision/sensing, data annotation and network training, sensor fusion, planning, control, simulation, and modelling.

Kurzfassung

Die Erfassung sich bewegender Subjekte, wie Menschen und Tiere, außerhalb einer geschlossenen und kontrollierten Umgebung in einem Labor ist von Natur aus schwierig, da sich die Subjekte aus dem Sichtfeld und der Reichweite von ortsfesten und extrinsisch kalibrierten Kameras und Sensoren bewegen können. Bisherige, dem Stand der Technik entsprechende Methoden für eine solche Wahrnehmung im Freien verwenden Marker oder Sensoren am Subjekt, die sowohl invasiv als auch für tierische Subjekte nicht skalierbar sind. Um dieses Problem zu lösen, führen wir fliegende Roboterkameras ein, die den Subjekten autonom folgen. Um Funktionen wie Monitoring, Verhaltensanalyse oder Bewegungserfassung zu ermöglichen, ist ein einziger Blickwinkel aufgrund von Selbstausschluss, fehlender Tiefenwahrnehmung und Erfassung von allen Seiten oft nicht ausreichend. Daher schlagen wir ein Team aus derartigen Roboterkameras vor, die in Formation fliegen, um eine kontinuierliche Erfassung aus mehreren Blickwinkeln zu ermöglichen. Die Position des Subjekts muss mit markierungsfreien Fernerkundungsmethoden in Echtzeit bestimmt werden. Um dies zu bewerkstelligen, kombinieren wir einen Detektor, der auf einem neuronalen Faltungsnetzwerk basiert, um das Objekt zu erkennen, mit einer neuartigen kooperativen Bayes'schen Fusionsmethode, um das erkannte Objekt von mehreren Robotern aus zu verfolgen. Die Roboter müssen dann ihre eigene Flugbahn und Ausrichtung relativ zum Subjekt planen und regeln, um eine kontinuierliche Erfassung aus mehreren Blickwinkeln zu erreichen und beizubehalten. Hierfür verwenden wir eine auf modellprädiktiver Regelung basierende Methode zur Vorhersage und Planung der Bewegung jedes Roboters in der Formation um das Subjekt. Ein erster Demonstrator wurde mit Multirotor-Drohnen umgesetzt. Drohnen sind jedoch laut und für die beobachteten Subjekte potenziell gefährlich. Um dieses Problem zu lösen, führen wir nicht-holonome, autonome Luftschiffe (Blimps) als Roboter-Kamera-Plattform ein, welche leichter als Luft sind. Diese Art von Robotern erfordert dynamisch beschränkte Umlaufformationen, um eine omnidirektionale visuelle Abdeckung eines sich bewegenden Objekts auch bei Wind zu erreichen. Hierfür stellen wir einen neuartigen modellprädiktiven Formationsregler für ein Team von Luftschiffen vor. Wir demonstrieren und evaluieren unser komplettes System in Feldexperimenten mit Menschen und Wildtieren als Subjekten. Die gesammelten Daten ermöglichen sowohl Motion-Capture von Menschen im Freien als auch Verhaltensanalysen von Tieren. Des Weiteren schlagen wir unsere Methode zur autonomen Langzeitüberwachung von Wildtieren vor. Diese Dissertation befasst sich mit dem Entwurf und der Evaluierung von Flugrobotern, die für diese Aufgabe geeignet sind, einschließlich Computer Vision/Sensorik, Datenannotation und Netzwerktraining, Sensorfusion, Planung, Steuerung, Simulation und Modellierung.

Acknowledgements

Our deep gratitude goes to everyone who made this dissertation possible, including everyone at Max Planck and IFR who helped with conducting experiments, building hardware, sharing ideas and brainstorming or was just there for a fun time. We thank everyone at the Flight Robotics and Perception Group who helped with experiments, everyone who stood outside in both burning sun, freezing cold, wind, and rain to pilot and supervise drone experiments. We specifically acknowledge the work of Guilherme Lawless for coding ROS nodes and help with the implementation of the C++ abstraction layer for coordinate system projection for the detector part, Rahul Tallamraju for his help in implementation of the multi-copter simulation environment in Gazebo, Yu-Tang Liu for modelling the individual components of our blimp as 3D objects available in the Gazebo simulator, without which the simulation of the vehicle would not have been possible and Pascal Goldschmid for his input and feedback regarding motion model derivation. A deep thanks also go to everyone who supervised and supported this dissertation, especially Aamir Ahmad for being a great mentor and deeply involved in the project from the very beginning. His vast ROS experience and previous work formed both a foundation and inspiration for our setup, and of course, Michael J. Black, whose experience and foresight initiated (and funded) this whole endeavour and directed it towards a successful path. Further thanks goes to Heinrich. H. Bülthoff, who supported the project with hardware and facilities. And last but most definitely not least our thanks goes to Noa Price, who sacrificed time and nerves to make this dissertation possible, while she was left alone with the kids (often until late at night and across weekends), and yet was still willing to help with proofreading papers and spotting all the typos.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Observing the world is the very foundation of all natural sciences. Measuring and recording are the crucial first steps, followed by understanding, forming of hypothesis, modelling, the latter of which allows meaningful predictions. Observation through measuring and recording is again crucial to validate those predictions. Only this knowledge allows us to foresee the very consequences of our own actions as humans and scientists.

Measuring and observation within the controlled environment of an indoor laboratory is a surmountable task. Cameras and other sensors can be placed in advance to record all relevant aspects of an experiment. Disturbances can be minimized and environmental parameters such as lighting can be controlled easily. Shortly after its inception in 2011, the Max Planck Institute for Intelligent Systems in Tübingen started designing a 4D scanner to measure shape and motion in its laboratory (Figure 1.1). This installation was used to measure and model the shape and deformation of the human body in motion, leading to groundbreaking models such as SMPL [1], SMPLR [2] and SMPL-X [3] for human body shape. These models could also be generalized to quadruped animal body pose and shape, such as SMAL [4] and SMALR [5]. The device also provided important results for medicine, allowing the modelling of body fat distribution and diabetes risk from mere images [6, 7]. Its 66 high-resolution cameras allow a complete reconstruction of the 3D body surface, 60 times a second [8]. However, the recording volume is constrained to a height of approximately 2 meters over a 1.5m^2 platform. This limits both the nature of movements and the kind of subjects that could be recorded. These limitations lead to the following questions:

1. Can we create a system that approximates this observation capability outdoors in unconstrained environments?

2. Is it possible to observe, measure and record human activities outside a controlled laboratory?

3. Can we also observe wild animals in their natural habitat, unconstrained and non-invasively?

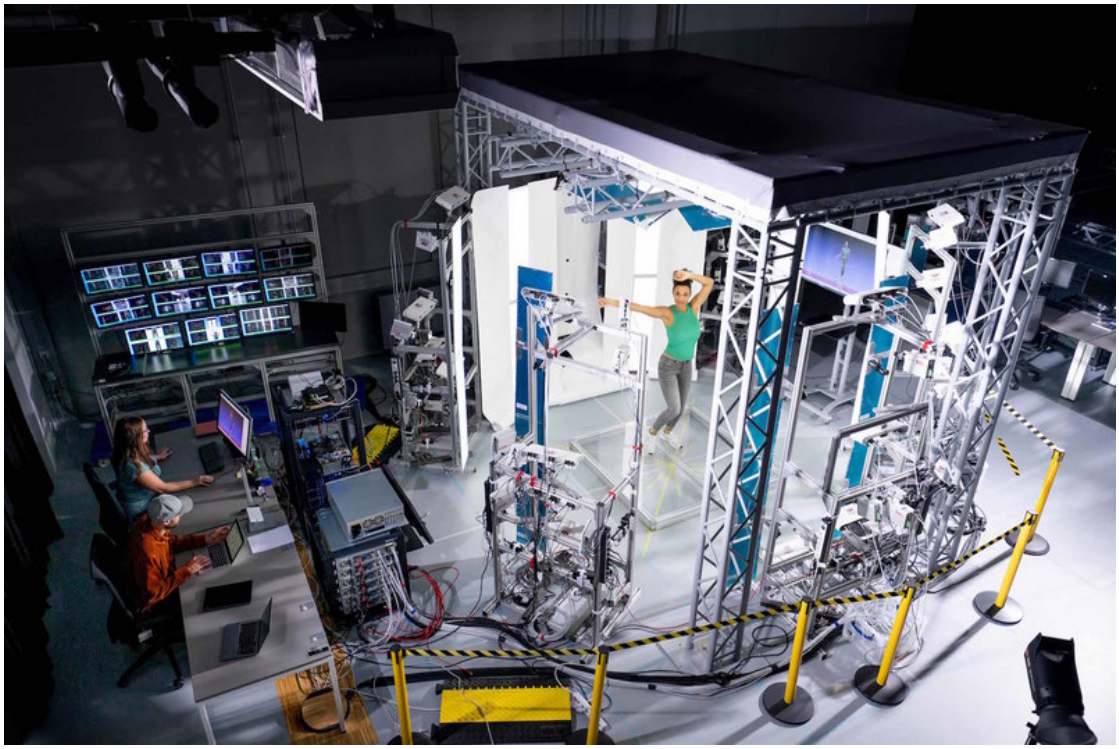


Figure 1.1: 4D Scanner, The 4D Motion scanner at the Max Planck Institute for Intelligent Systems. Image source: MPI IS, 2024 `https://ps.is.mpg.de/pages/4d-capture`

If yes, doing so would allow studies beyond mere shape and form. We could then observe the behaviour and interaction of animals[9, 10], which also enables model creation of their interaction, social relationships and how these change in relation to their environment and human interference. As with the indoor-scanner, we would need more than one camera in more than one location. In the indoor 4D-scanner, cameras are placed all around the subject to be able to see all sides, including non-convex surfaces. This is also needed to triangulate every point or vertex on the surface to estimate the depth, as single cameras can only measure a 2D projection. For complete coverage of the horizontal plane, a minimum of three cameras around the subject is required, distributed around the observed subject to continuously cover all sides. This avoids visual ambiguities of the 2D-projection, e.g. between the left and right leg of an animal. In recent years, drones have become increasingly popular for wildlife observation and behaviour study [11]. However, despite their popularity, these vehicles are noisy and not without risks [12], their presence can affect and frighten the subjects we would want to study [13]. Furthermore, multi-copters remain airborne using propulsive lift, which is inefficient and significantly limits their flight time [14]. We also have to consider the effects our system would have on its environment and how the subjects could respond to, while being

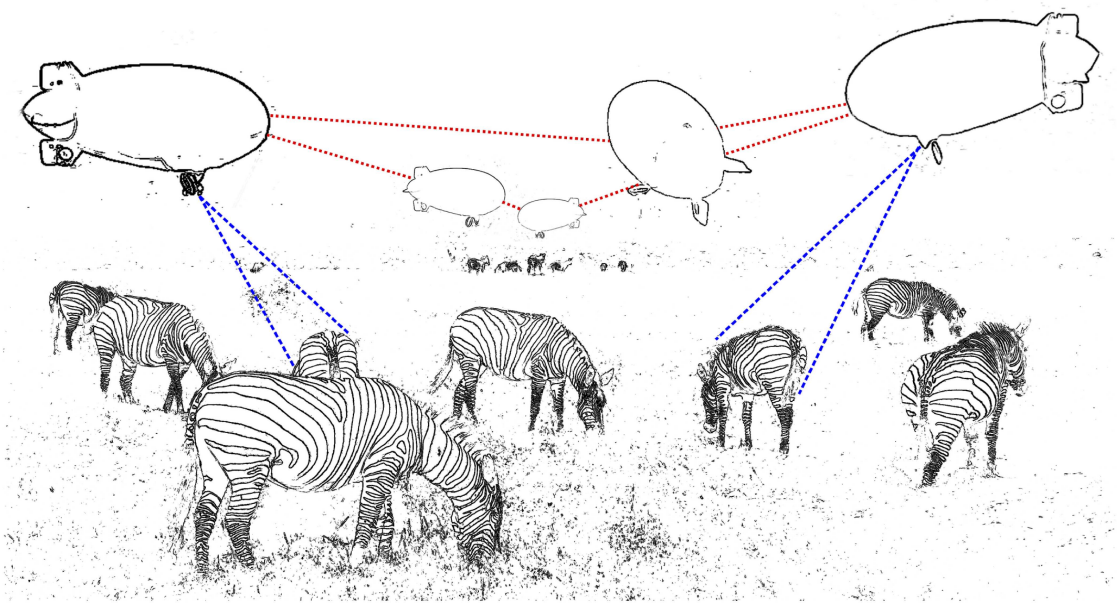


Figure 1.2: Blimp Formation, A formation of blimps envisioned to study Grévy's Zebras. The image is an edge-enhanced montage of images of our early blimp prototype in flight, overlaid over edge enhanced drone footage of Grévy's Zebras taken in Kenya, 2022. Image source: Aamir Ahmad

observed.

1.2 Proposed solution

The solution we propose involves a team of robotic airships [15] (Figure 1.2), which fly in such a formation to observe, track and follow the subject [16]. To achieve this, we follow the sense-think-act paradigm [17] of robotics (Figure 1.3). We design a closed-loop system involving robotic hardware, perception algorithms, planning algorithms and control algorithms to interact with the real world represented by the robotic environment in general and the to-be-observed subject in particular. In our case the robotic hardware is an entire formation of robots, which sense, think and act as a team. In principle there are two paradigms to achieve cooperative operation across multiple robots: i) centrally controlled, where a single computer processes the measurements of all robots, plans, and controls the actions, which are then sent to the individual robots and ii) distributed control, where each robot makes its own computation and receives only the necessary data from the other robots needed to come to a joint decision. Both approaches require communication. A central approach requires sensor data and actuations to be transferred, while a distributed approach requires part of the state estimate to be communicated. Throughout this dissertation, we chose the distributed approach, for two reasons. i)

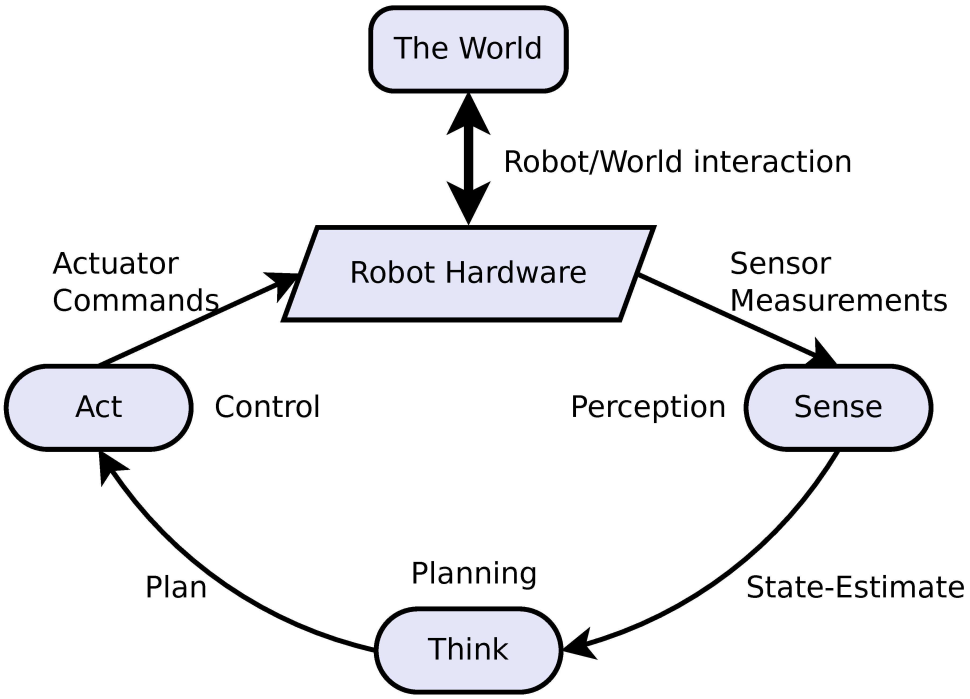


Figure 1.3: Sense-Think-Act paradigm by which robots interact with the world. Sensors on board of the robots provide measurements that are interpreted by perception algorithms. This results in state estimates that inform planning algorithms that create plans — in our case in the form of planned optimal trajectories. Control algorithms compute actuator commands that allow the robots to act in the world according to these plans. The effect of this interaction is then again measured and perceived, etc.

Safety: A centralized approach relies both on a central computer node and communication links between it and all robots. Any failure in these leads to partial or complete loss of control of these robots. In a distributed system, each robot is independently capable of sensing, thinking and acting in the world. In case of a communication failure or failure of a different robot, it merely possesses less information about the state. There is no central single point of failure. ii) Communication. A fully centralized approach requires transmission of all sensor data and all actuator commands wirelessly. As we show in Chapter 2, this does not scale, whereas a distributed approach not only minimizes the required communication bandwidth but also maximizes robustness to communication failures. Our approach involves distributed, cooperative perception, distributed cooperative planning, for a team of identical robots, which we design and for which we develop control algorithms. Furthermore, we design simulation methods for airships and data annotation methods for the used data-driven (learned) perception algorithms, which are described below.

1.2.1 Cooperative Perception

We use a team of multiple unmanned aerial vehicles (UAV) equipped with fixed, frame-mounted cameras, which are calibrated intrinsically and extrinsically with-respect-to (w.r.t.) the UAV body frame. A tracker, implemented as a distributed Kalman-filter (KF), running on every UAV tracks the position of the subject in the world frame. An Extended Kalman-filter (EKF), running on every vehicle as part of the flight controller (FC) tracks the position of each UAV relative to a reference coordinate (Home Position).

State-of-the-art detectors [18] are too slow for real-time detection on the full resolution camera images. On down-sampled images, the subject might be too small for detection. We solve this by running the detector on a down-sampled region of interest (ROI). The expected position of the subject is projected into each UAV's camera image frame, and a ROI is calculated around it, based on the projected size and position uncertainty of the subject, large enough to include the complete subject within 3 sigma likelihood. At a real-time frequency of approximately 4 Hz, this ROI in the most current video frame is downsampled to the size of a deep convolutional neural network single shot detector (SSD-Multibox) [19] and processed by an onboard GPU. Any detection bounding box above a threshold is then projected first as a 3D point in the camera frame, estimating depth based on the apparent size of the subject at the current observation angle and the expected average size of a subject. These detections are represented with Gaussian uncertainty as mean plus covariance matrix, where the detector's average localization accuracy dominates x and y uncertainty, while depth uncertainty is dominated by a comparably large uncertainty in the size distribution.

These detections are then projected into the global world coordinate system, combining the uncertainty of the detection with the uncertainty of the UAV position. These detections are sent to all UAVs and used to update the distributed KF. False positives could skew the tracker's estimate. Therefore, any detection at a position with less than 5 sigma likelihood considering the current mean and combined uncertainty of state and detection is discarded as a likely false positive. For multiple detections in the same video frame, only the detection that is closest to the tracker mean position is used.

The state estimate of each UAV is informed by a Global Satellite Navigation System (GNSS) receiver, typically called Global Positioning System (GPS). GPS coordinates are often biased depending on the set of satellites in view. This bias is not reflected in the UAV's EKF self pose estimate. To compensate for potential coordinate system offsets that could affect fusion of detections between different UAVs, each vehicle's KF also tracks a 3D self pose bias estimate, informed by the offset between its own subject detections and the global shared estimate. Any systematic error in the detector and projection is absorbed by this bias estimate to ensure a consistent shared estimate across the whole fleet, as well as bias corrected pose estimates of all UAVs relative to each other. The latter is crucial for collision avoidance and consistent formation control. This work is described in Chapter 2.

1.2.2 Cooperative Planning

The above-mentioned novel neural-network-based cooperative tracking framework has been published [20] as state-of-the-art in 2018. In our initial implementation, this was combined with a simple formation controller [21]. It maintained a preset angular distance and angular separation between UAVs, with every UAV oriented towards the subject to implicitly enforce visual coverage in the camera field of view (FOV).

This approach was later extended by [22] with a model predictive controller to optimize the formation in real time, continuously minimizing the tracking uncertainty, while simultaneously maintaining a safe distance between UAVs, the subject, and known static obstacles in the world. Both formation controllers send trajectory waypoints to the UAV's FC, which flies towards the latest waypoint using cascaded PID controllers. This design has the added advantage that a malfunction in the formation controller or the computer it runs on has the FC fly to the last assigned waypoint and stay there. A human operator can then take over remote control with a Radio-Control (RC) transmitter and safely land the affected vehicle, while the teammates autonomously avoid its last known position.

For airships, a different control strategy is needed. Airships use static buoyancy to overcome gravity instead of lift, which allows for much greater endurance, higher efficiency, less noise and overall a very low mass density, which alleviates the risk of injury in collision. Airships are, however, more difficult to control. Unlike a drone which can move omnidirectionally and also hover on the spot, a blimp is a non-holonomic vehicle that needs to point into the direction it is flying regarding the surrounding air — affected by wind — and it needs a minimum airspeed to maintain manoeuvrability. Combined with a limited camera field of view (FOV), either due to a static body frame mounted camera, or camera gimbal constraints, these restrictions pose a challenging problem for the camera-formation control. We can overcome this problem with an orbiting formation strategy.

In [22] it was established, that an optimal formation for perception has equal angular spacing. It follows that airships should also maintain the same formation objective. Since airships need to remain in motion to maintain manoeuvrability, the resulting formation is an orbit around the subject, where all airships maintain equal angular velocity w.r.t. the subject regardless of their current distance and flight velocity. These orbits are also possible if the subject is moving and if there is wind, however this skews the orbits, which are then no longer centred around the subject. Airships must control their airspeed, to ensure their angular rotation rate and velocity matches the orbital trajectory to maintain the subject in the centre of the frame of a perpendicularly oriented on-board camera. If the camera is also angled downwards, the altitude must be adjusted based on the changing distance to the subject. Finally, manoeuvres such as altitude changes or curved flight cause changes in pitch and roll of the airship, affecting the camera. The airship trajectory must be adjusted accordingly. We achieve this with a model predictive formation controller (MPC). It optimizes the trajectory of all airships relative to the subject to keep the subject's projection, as seen by each camera, as close to the centre of the camera frame as

possible. Simultaneously, the MPC controls the airship's distance to the subject and the angular separation between airships. The formation controller can also be extended to avoid collisions with static obstacles. However, especially with changing wind or similar external disturbances, collision avoidance cannot be guaranteed. We implemented this formation controller and demonstrated it on a formation of simulated airships, as well as on our real airship. [16]. This work is described in Chapter 5

1.2.3 Training and Data Annotation

The previously mentioned deep convolutional detector network requires training on annotated data to detect objects in video-images. Initially, we used SSD-Multibox [19], pre-trained on VOC-PASCAL [23] and MS-COCO [24] datasets , which was sufficient for human detection due to the availability of large amounts of annotated human images in these datasets. However, for animals in the wild, availability of suitable datasets is not guaranteed. We therefore had to annotate data collected in-situ from drone videos. Manually annotating video-frames is very time-consuming. To alleviate this bottleneck, we developed our own annotation tool, Smarter-labelme [25], with a method for accelerated assisted annotation. To speed up annotation, we exploit the temporal and spacial consistency in aerial video data, by pre-annotating data based on both detections and 2D tracking data in the video. For this, we combine state-of-the-art detection and tracking networks, particularly the same SSD-Multibox for detection and Re³ [26] — a Siamese recurrent 2D tracking network. 2D trackers for arbitrary objects often suffer from drift. This can be compensated by correcting the drift with per-frame detections, if available. To our knowledge, this approach [27] has not previously been used for video annotation. We show that this approach improves annotation speed, and we also show how re-training the detector on partially annotated data improves the automatic annotation performance. Once annotated real-world data was available, we re-trained SSD-Multibox. However, training on data of a single species in a single habitat alone does not yield acceptable performance, while fine-tuning a pre-trained network exhibits a disappointing trade, giving much worse generalization for marginal improvements on the fine-tuning dataset. Our solution was to combine datasets, adding machine augmented variants of our manually annotated data to the MS-COCO training data until our examples comprised approximately 10% of the complete training dataset. The resulting network has acceptable performance on our own validation data, without significant precision decrease when tested on MS-COCO. This work is described in detail in Chapter 3.

1.2.4 Modelling, Simulation, Robotic Hardware and Control

We designed both multi-copter and airship-based robots on a common Robot Operating System (ROS) [28] architecture using a distributed multi-master layout [29]. Each robot runs a main computer for perception and planning and a low-level flight controller (FC) (OpenPilot Revolution) [30]. While the IMU state-estimation via EKF and PID

control algorithms for the multi-copters are Off the Shelf (OTS) open-source implementations [31] running on an OpenPilot Revolution FC [30], the formation controller, subject tracker and probabilistic projection are implemented in ROS on an Ubuntu main computer. The neural network is running on an on-board embedded NVIDIA Jetson TX1 GPU [32]. A software-in-the-loop (SITL) simulation has been used, based on the Gazebo simulator [33] and RotorS [28], which includes existing models for multi-copter-simulation. The hardware architecture of the multi-copters is described in section 2.4
Airships are more difficult to control, and their motion is heavily influenced by aerodynamics. To use airships, it was first necessary to model, simulate and control an individual airship. We simulate the airship and its aerodynamics based on the sum force of individual components, which are non-rigidly connected. Wind is simulated using a Dryden-based turbulence frequency distribution [34]. We equipped a 5m blimp with an FC, an airspeed sensor and a fixed frame-mounted on-board camera and computer, which we also modelled in the "Gazebo" physics simulator [33]. Based on this, we implemented a cascaded PID-based flight controller that uses thrust vectoring and aerodynamic control surfaces to control airspeed, climb and sink rate and flight direction. This controller was then evaluated both in simulation and real-world flights [15]. This work is described in Chapter 4

1.2.5 Field Evaluation

We evaluated our methods in the wild under real-world conditions. An airship optimized for realistic field-conditions has been built and flown at the Pentezug reserve in the Hortobágy National Park in Hungary, a UNESCO World Heritage Site [35]. We demonstrated successful detection, tracking, and vehicle control under field conditions (Chapter 6).

1.3 Development, Experiments, and Results

1.3.1 Early multi-copter flights

Research for this dissertation started in 2016 at the Perceiving Systems department of the Max Planck Institute for Intelligent Systems, with a project aiming towards Aerial Outdoor Motion Capture (AirCap) [36]. Initial work involved finding suitable hardware and software for a fleet of octo-copters to test and evaluate suitable algorithms for aerial motion capture within the AirCap project (Figure 1.4). We settled on octo-copter frames by German manufacturer Mikrokopter (Hi Systems), which were previously used in the flight laboratory at the Max Planck Institute for Biological Cybernetics. An informal comparison of flight control hardware and software revealed that the stability and reliability of the avionics firmware was higher than competing products available at the time. Therefore, we switched to an OpenPilot Revolution flight controller [30]. We selected

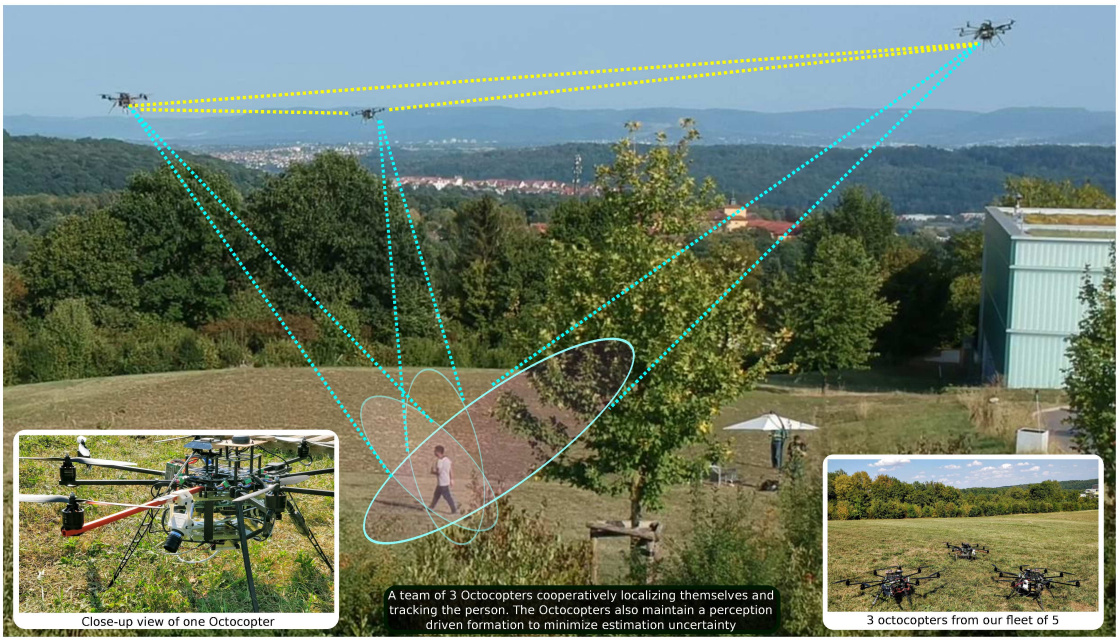


Figure 1.4: AirCap Multi-copters tracking a person in formation. The main image in the montage shows our formation of 3 octo-copters in a flight over the Max Planck Institute campus in Tübingen, Germany, during a motion capture experiment conducted in 2018 as part of the AirCap project.

to use multi-master Robot Operating System (ROS) [29] and Wi-Fi communication for the initial demonstrator. We evaluated several methods for visual object detection. The initial plan was to start with a simple colour-detector and later iterate to a deep method for object detection, but SSD-Multibox was more reliable in detecting humans than a traditional colour-detector in combination of orange hard hats to be worn by the person, so we equipped the octo-copters with a dedicated NVIDIA Jetson TX1 embedded GPU [32] to run the detector network, back then implemented in Caffe [37]. We implemented the distributed sensor fusion as described in Chapter 2 and flew initial test flights with 1 and later with 2 octo-copters. The initial Model Predictive Formation controller by Ahmad et al. [21] had a basic point-mass model, that spread out the vehicles across a circle around the subject with pre-defined altitude and radius. It had no anti-collision constraints and avoided collisions with a potential field based simulated force approach, which converged to the desired evenly spread formation. The low-level control used cascaded PID loops, that were part of the OpenPilot/LibrePilot flight firmware [31]. Our main contribution to LibrePilot at that point was the ROS integration [38], which allowed to access the EKF based state estimate of the flight controller and to send waypoint commands to the autopilot. In 2017, we improved the copter hardware as well as the Kalman Filter for sensor fusion to also model the GPS bias of each copter. This significantly improved the accuracy and allowed flights with 3 copters converged around the same subject. These

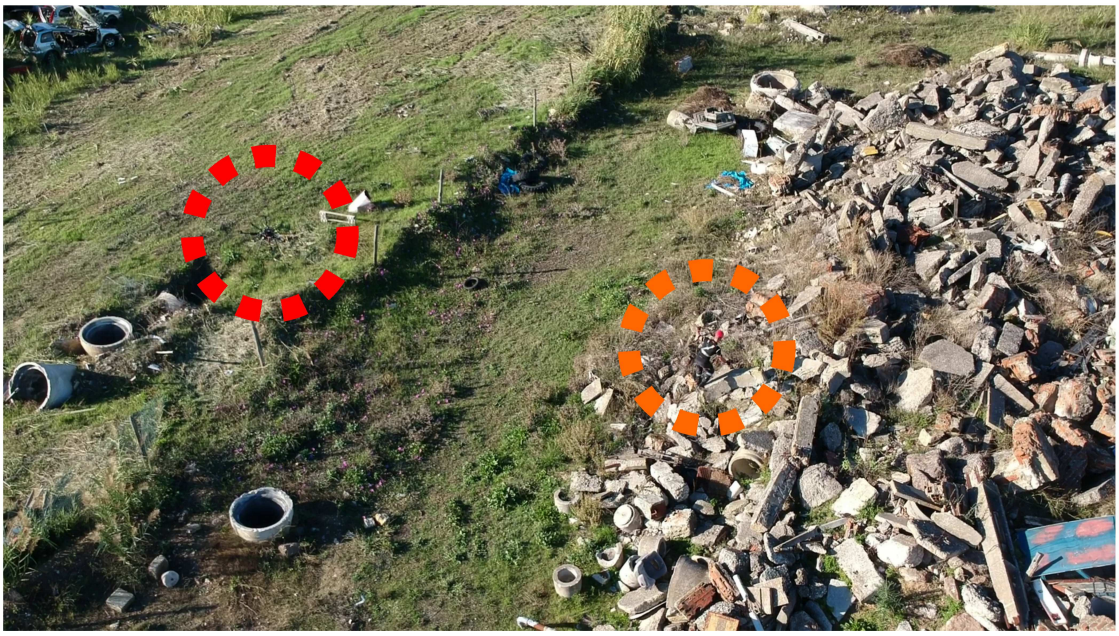


Figure 1.5: Multi-copter observing simulated rescue operation. Visually distinguishing objects in this environment is extremely challenging, both for humans and for computer vision. Highlights: There is a flying octo-copter (top left) and a person (centre right) in this image. This photo of two of our octo-copters was taken from a third drone during flight experiments with the Lisbon Fire fighting school over their search and rescue training ground, Portugal, 2018.

results were published and presented at NVIDIA GTC 2018 and IROS 2018 [20].

1.3.2 Operational flights with multi-copters

In parallel to the work described in Subsection 1.3.1, [22] worked on an improved MPC formation controller with anti collision constraints and guarantees. This did not change the shape of the formation, but this work shows that for 3 or more copters, an evenly spaced formation leads to the least uncertain state estimate. We then took our fleet of multi-copters to field tests in Portugal (Figure 1.5) for evaluation for search and rescue related applications in cooperation with the Lisbon fire fighting school.

1.3.3 Attempts at different neural network architectures

This exposed inherent weaknesses in the detector, especially when operating in highly cluttered environments that were unlike the visual environments the detector was trained on. Although it was clear that this could be alleviated by retraining on video material closer to the target domain, we also investigated alternate detector architectures based on

recurrent neural networks and LSTMs. The idea was to train a neural network to learn the current visual environment on the fly and use attention-based feature weighting to improve visual detection. Although initial results were promising, we were unable to match state-of-the-art single shot detector performance, and this work was abandoned and did not get published. Meanwhile, [39] worked with the recorded data from our flight experiments on recovering complete 3D pose and shape of humans from the recorded aerial data.

1.3.4 Lighter than air vehicles

After successfully demonstrating our methods on multi-rotors, we transitioned to lighter than air vehicles due to the shortcomings of multi-rotor drones mentioned earlier. We built our first Blimp-prototype in 2019. However, we had no autopilot or control algorithm to fly this vehicle, nor a simulation environment with which to realistically develop control methods. This was needed for a reinforcement learning-based method for blimp control [40] in development at the time, so we focussed on blimp simulation and control [15]. We modelled the blimp in ROS/Gazebo [28], including non-rigid connections between fins and hull, which allowed simulating deformations of the not-completely inflated vehicle. Aerodynamic forces were simulated using an independent component-force estimation. The model was tuned by hand to replicate the behaviour of our prototype as observed in manually flown test flights. We then implemented a simple but effective PID-based controller, which was tuned and tested in simulation and then demonstrated to fly the real vehicle. The work was presented at Intelligent Autonomous Systems 16 in 2020 [15]. This work is described in Chapter 4.

1.3.5 Real-time pose estimation

In 2022, [41] worked on a distributed real-time artificial neural network which could estimate full body human pose and shape from drones with minimal communication overhead. While the work on neural network architecture and training is not part of this dissertation, this method relies on the subject state estimate provided by our method. Furthermore, we contributed to [41] with the ROS-based communication and synchronisation code that allowed to demonstrate the network on actual flight hardware. To allow the extra computation, the aforementioned octo-copters were equipped with additional embedded GPUs, this time an NVIDIA Jetson TX2's [42].

1.3.6 Formation control with airships

The logical next step in our work was to implement AirCap on airships. For this, a new Model Predictive Formation Control (MPC) algorithm was needed, as the existing controllers by [21, 22] expected the vehicles to accelerate independently of their yaw-orientation, as common for drones. Airships behave more like fixed-wing aircraft

because they are typically non-holonomic vehicles that can only fly in the direction they are facing, albeit with a lateral angle of attack. We modelled this behaviour of airships in curved flight, and then designed an MPC formulation that directly optimizes the trajectories of an entire airship formation to centre the tracking subject in each vehicle's camera view, while also maintaining collision avoidance constraints [16]. This work is covered in detail in Chapter 5.

1.3.7 Recording real-world animal data

In 2022, while the work on airships was in progress, we took several field trips to study wild animals in Kenya and the Hortobágy national park in Hungary, where Grévy's zebras and Przevalski's horses were recorded in aerial footage. In the first round, all drones were flown manually. This yielded a wealth of training data, with which the neural network for real-time detection could be improved. For this, however, the data needed to be annotated.

1.3.8 Smart Data annotation

Our group undertook multiple annotation efforts, before we decided to adapt an open-source tool for video annotation and accelerate the process by integrating deep-learning-based detectors and trackers. This significantly sped up the annotation process and allowed us to retrain SSD-Multibox to detect horses and zebras, respectively. The results of this work are presented in Chapter 3.

1.3.9 Real-world evaluation

In a second trip to Hungary in the winter of 2022/2023, we successfully tracked a horse with a flying multi-rotor drone prototype, equipped with a Jetson TX2 and this retrained neural network. Although technical difficulties with the prototype in the form of engine-caused vibrations prevented collection of useful data, this demonstrated AirCap on wild animals (WildCap) for the first time in the wild.

In the summer of 2023 we went to Hungary again, this time with a new airship prototype (Figure 1.6). This was the first test of the new formation controller [16] in the loop with an airship and animal detection and tracking [43]. Due to time constraints and high winds, the algorithm was only able to track the horse for a few seconds, but this was sufficient to demonstrate convergence of the tracker and successful Model Predictive Control of airships for observation of animals.

We presented our work [27] on Smarter-labelme at IAS-18 in Korea, and the blimp formation MPC [16] at IROS-2023 in Detroit. The field evaluation was presented at DELTAs2024 [43] in Mumbai.

The drone-footage on animals also allowed further behaviour studies on Zebras and Horses [44]

Figure 1.6: Our airship, autonomously observing horses. This photo shows our autonomous blimp flying over a pair of Przevalski's horses with their foal, which have separated from the main herd. Picture taken in the Hortobágy National Park in Hungary during flight experiments conducted in 2023.

1.4 Primary Contributions

The main contributions in this thesis are

- In Chapter 2, a novel, closed loop method for cooperative distributed neural-network based detection and Bayesian tracking, published in IEEE Robotic Automation Letters in 2018 as "Deep Neural Network-Based Cooperative Visual Tracking Through Multiple Micro Aerial Vehicles" [20].

- In Chapter 3, an improvement to machine assisted object instance bounding box annotation in video, published in Intelligent Autonomous Systems 18 in 2023 as "Accelerated Video Annotation Driven by Deep Detector and Tracker" [27].

- In Chapter 4, a novel framework for airship simulation and control, published in Intelligent Autonomous Systems 16 in 2022 as "Simulation and Control of Deformable Autonomous Airships in Turbulent Wind" [15].

- In Chapter 5, a novel nonlinear model predictive control strategy for airship formations, published in IEEE Robotic Automation Letters in 2023 as "Viewpoint-Driven Formation Control of Airships for Cooperative Target Tracking" [16].
- In Chapter 6, the combination of all the above methods for the purpose of autonomous wild animal observation, presented at Design and Engineering of Lighter-Than-Air-Systems (DELTAs) in 2024 as "Airship Formations for Animal Motion Capture and Behaviour Analysis" [43].
- Chapter 7 concludes the thesis with a summary of key insights provided by this research.

Chapter 2

Cooperative Visual Tracking

This chapter forms the foundation for this thesis in terms of cooperative, distributed object detection and tracking, necessary to provide the formation controller with the relevant subject trajectory data. It also forms the foundation for the work of our research group on project AirCap [36], based on which Tallamraju et al. focused on improving the formation control for multi-rotor craft [22], Saini et al. worked on the reconstruction and motion tracking [39, 41] and this thesis pursues formation control with lighter than air vehicles [15, 16].

For this chapter, we operate under the assumption that a formation controller [16, 22] exists that keeps multiple robots' cameras pointed such that the subject is within their field of view. When this chapter was published, this was ensured by a basic baseline formation controller that maintained a constant distance to the subject in a present altitude above the subject, evenly spaced on a circle.

The first key insight of this chapter is that deep-neural-network-based object detectors are both capable enough and fast enough to visually detect and track subjects, only if the detector is run on a relatively low-resolution image (in our case 300px $\times$ 300px). By using a Bayesian filter to track the subject, a region of interest (ROI) can be calculated based on the distance, size, and uncertainty of the subject which, when down-sampled to the native neural network resolution, has the subject large enough for reliable detection.

The second key insight is that all robots have the tracking subject in view, which provides a common reference point in the world frame, even if the subject moves. As such, errors in self-position-estimation of all involved robots and their cameras, stemming from GPS bias and other sensing errors, can be corrected in a distributed fashion based on this common time-variant reference point.

For this chapter, a fleet of multi-copters were developed, built, and later improved. This fleet consists of octo-copters with a weight of approximately 4.5kg (Figure 2.1). Our initial approach was to capture raw video from global-shutter cameras with no compression artefacts. A powerful Intel I7 server board was chosen, alongside a state-of-the-art high IO bandwidth solid-state disk to store 2 Megapixel video losslessly with up to 60fps. The architecture has a Neural Network for detection running on a dedicated on-board GPU (NVIDIA Jetson TX1 [32]), connected to the main computer via Ethernet using a short CAT5 cable. A particular design challenge, not mentioned in the published

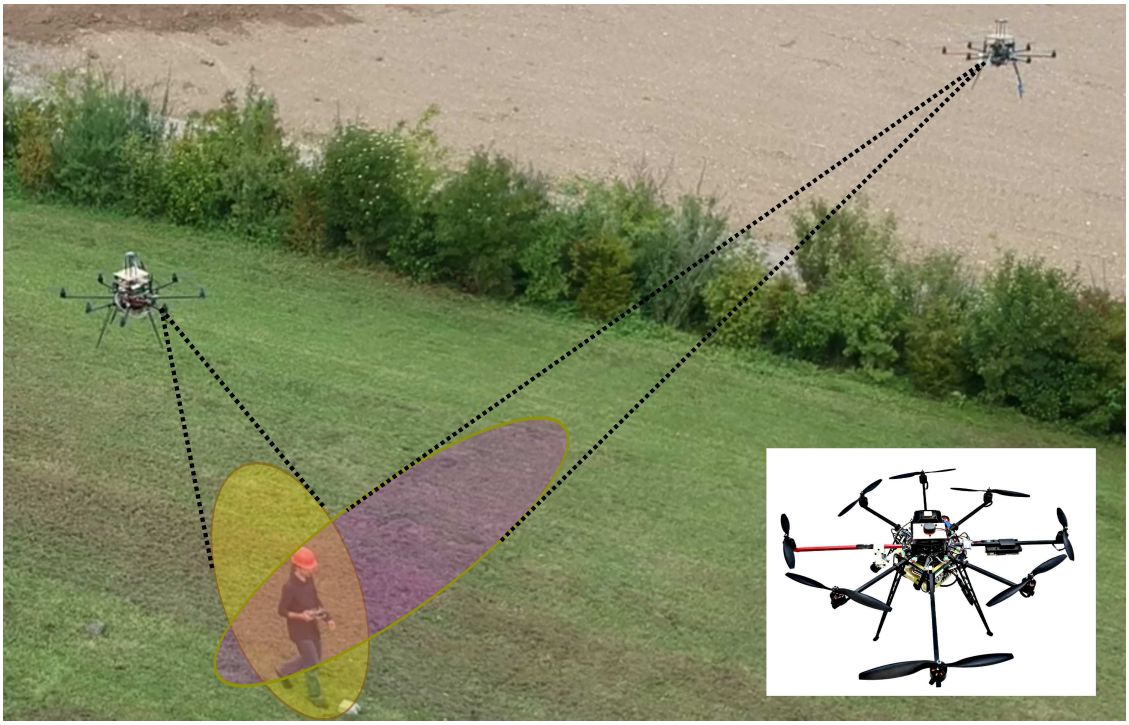


Figure 2.1: Two of our self-designed octo-copters cooperatively tracking a person while maintaining a perception-driven formation. (In box) Closer view of one octo-copter. The main image shows two of our earlier octo-copter prototypes in flight at the Max Planck Institute in Tübingen in 2016. The box shows a newer, improved and retrofitted 2018 design of the same vehicle.

paper, was the electromagnetic interference of the USB3 camera with the GPS receiver. We eventually solved this problem by exploiting the main battery as an EM shield by mounting the GPS receiver just above the Lithium Polymer battery, which is effectively a solid block of metal, well suited for this task.

2.1 Introduction

Human/animal pose tracking, and full-body pose estimation/reconstruction in outdoor, unstructured environments is a relevant and challenging problem. Its wide range of applications includes search and rescue [45], coordinating outdoor sports events [46] and facilitating animal conservation efforts in the wild [47]. In indoor settings, similar applications usually make use of body-mounted sensors, artificial markers and static cameras. While such markers might still be usable in outdoor scenarios, dynamic ambient lighting conditions and the impossibility of having environment-fixed cameras make the overall problem difficult. On the other hand, body-mounted sensors are not suitable for some

kinds of subjects (e.g., animals in the wild or large crowds of people). Therefore, our approach to the aforementioned problem involves a team of unmanned aerial vehicles (UAVs), tracking subjects by using only on-board monocular cameras and computational units, without any subject-fixed sensor or marker. Among the several challenges involved in developing such a system, multi-robot cooperative detection and tracking (MCDT) of a subject's 3D position is one of the most important. It is also the main focus of this chapter.

The key component of our MCDT solution is the person detection method suitable for outdoor environments and marker/sensor-free subjects. Deep convolutional neural network-based (DNN) person detection methods are, unarguably, the state-of-the-art. DNNs consistently outperform traditional techniques for object detection [48]. Consequently, they have gained immense popularity in various robotics-related fields, such as, autonomous driving [49] and detection and identification of pedestrians [50]. However, there are only few works that exploit the power of DNNs for visual object detection on board UAVs.

The main limitation in using DNNs on-board UAV is the computational requirement of these networks. One alternative is transmitting raw images to a ground-station to offload their processing. Unfortunately, this is infeasible due to communication bandwidth constraints when multiple UAVs are used. Using the light-weight GPU (Jetson TX1 [32]), Wei Liu et al.'s single shot multi-box detector (SSD-Multibox) achieves ~ 4 fps [19] in sequential processing. This hardware is suitable for airborne use, as its performance is just at the lower boundary of real-time applicability. The 4 fps performance of SSD is achieved only with low resolution (low-res) images. Vanilla SSD on Jetson TX1 cannot achieve real-time on-board processing of high resolution (high-res) images. Thus, it cannot exploit the richness of the high-res images provided by most modern cameras. A direct approach when using high-res images is to down-sample them before performing object detection. This is suboptimal due to the reduced information content. Such a strategy causes most deep-neural-network (DNN) based detectors, like SSD, to often fail at detecting objects far away from the camera, which is a typical characteristic of a scenario with aerial robots.

The mutual world knowledge about the tracked person is jointly acquired by our multi-UAV system during cooperative person tracking. Leveraging this, we introduce a method that actively selects the relevant region of interest (ROI) in images from each UAV that supplies the highest information content. Our method not only reduces the information loss incurred by down-sampling the high-res images, but also increases the chance of the tracked person being completely in the field of view (FOV) of all UAVs.
The core contributions of this chapter are as follows.

- A real-time, continuous and accurate DNN-based multi-robot cooperative detection and tracking method, which is designed and evaluated rigorously for a team of UAVs operating in outdoor environments with only on-board camera perception & computation.

- A method for statistically characterizing the DNN-based detection measurement noise.
- Fully open source ROS-based implementation of our method.

In the next section, we situate our work within the state-of-the-art. This is followed by a description of our system design, theoretical details of our proposed MCDT approach and characterization of detection measurement noise in Section 2.3. Section 2.5 and 2.5.4 present our experimental results in real and simulated scenarios, respectively. Section 2.6 concludes the chapter.

2.2 State of the Art

Motivated by several applications in computer graphics and computer vision, e.g., virtual reality, sports analysis, pose monitoring for health purposes and crowd management, the full-body pose and motion estimation problem has attracted widespread research attention [51]. Most state-of-the-art research in this field is focused on indoor scenarios, e.g., [52]. However, recent works have started to address the challenges of performing full-body motion capture in unstructured everyday environments [53] and even in unknown outdoor scenarios [54, 55]. Authors in [54] use body-fixed sensors (IMUs) to estimate the full-body pose. Although they achieve a high degree of accuracy, the setup itself may be cumbersome or even infeasible for certain subjects (e.g., animals). To avoid this, as well as any body-fixed markers, the alternative approach is to perform motion capture using multiple flying cameras. In [55] authors use depth sensors (Xtion™) mounted on UAVs and stream depth images to a ground station for processing. Most depth sensors are quite noisy in outdoor environments and in sunlight. Furthermore, off-board processing limits their system's real-time applicability. In contrast, our approach relies on UAV-mounted high-res RGB cameras and only on-board processing on the UAVs.

In [56], the authors presented a method to extract the skeletal motion of multiple closely interacting people by using multiple cameras, without using body-fixed markers or sensors. However, their solution assumes a scenario where cameras can be fixed to the environment and the ambient light is controlled. In unknown outdoor environments, these assumptions do not hold. Hence, our approach to motion capture in outdoor scenarios involves multiple UAVs, each with an on-board camera. In this chapter, we address the key issue of multi-robot cooperative detection and tracking (MCDT) involved in developing such an outdoor motion capture system. Through a cooperative perception-driven formation, we not only ensure that the tracked subject is almost always in the FOV of all UAVs, but also prevent the UAVs from colliding with each other.

Multi-robot cooperative tracking has been researched extensively over the past years [57, 58, 59]. While the focus of most of these methods is to improve the tracked target's pose estimate by fusing information obtained from teammate robots, recent methods, e.g. [58],

simultaneously improve the localization estimates of the poorly localized robots in addition to the tracked target's pose, within a unified framework. However, it is difficult to find any cooperative target tracking method that directly facilitates detections through cooperation among the robots. In this chapter, we do so by sharing independently obtained measurements among the robots, regarding a mutually observed object in the world. Using these shared measurements, our MCDT method at each UAV allows its detector to focus only on the relevant and most informative ROIs for future detections.

Cooperative tracking of targets using multiple UAVs has also attracted attention recently [60, 61, 62]. While some work addresses the problem of on-board visual detection and tracking using UAVs [63], they still rely on hand-crafted visual features and traditional detection approaches. Moreover, cooperation among multiple UAVs using on-board vision-based detections is not well addressed. In this chapter, we address both of these issues by using a DNN-based person detector that runs on board the UAVs and sharing the detection measurements among the UAVs to perform MCDT.

Deep CNN-based detectors currently require the parallel processing power of GPUs. For UAVs, light-weight GPUs or embedded solutions are critical requirements. In [64], Rallapalli et al. present an overview of the feasibility of deep neural network-based detectors for embedded and mobile devices. Moreover, several recent works now consider airborne applications of GPU-accelerated neural networks for computer vision tasks. However, they mostly evaluate their performance in offline scenarios, without a flight capable implementation [65]. On the other hand, there are some networks suitable for real-time detection and localization of arbitrary objects in arbitrary poses and backgrounds. These include networks, such as, YOLO [66] or Faster R-CNN [67], which are both outperformed in speed and detection accuracy by the SSD-Multibox [19]. Hence, in our work we use the latter. Furthermore, we ensure its real-time operation for a camera of any given resolution. We achieve this through our cooperative approach involving active selection of the most informative ROI, a method that is novel to the best of our knowledge.

2.3 Methodology

2.3.1 System Overview

Our multi-UAV system does not consist of a central computational unit. Each of our UAVs is equipped with an on-board CPU and GPU to perform all computations. Although our architecture does not depend on a centralized communication network, the field implementation is done through a central Wi-Fi access point. Each UAV runs its own instance of the following software modules (blue blocks in Figure 2.2).

- Low-level position and yaw controller module.

- Self-localization module using on-board GPS, IMU & barometer.

- Cooperative detection and tracking (CDT) module, which is the core contribution of this chapter and described in subsection 2.3.3.
- MPC-based formation controller and obstacle avoidance module that allows us to maintain a perception-driven formation, described in subsection 2.3.4.

Figure 2.2 details the flow of data among these modules. The data shared among UAVs consists of their self-pose estimates and the detection measurements of the tracked person.

2.3.2 Preliminaries

Let there be K UAVs $R_1, ..., R_K$ tracking a person P. Let the 6D pose of the k_{th} UAV in the world frame at the time t be given by $[\mathbf{x}_t^{R_k}\ \Phi_t^{R_k}] \in \mathbb{R}^6$, obtained using a self-localization system. $\mathbf{x}_t^{R_k}$ denotes the 3D position and $\Phi_t^{R_k}$ represents the 3 orientation angles. The uncertainty covariance matrix associated with the UAV pose is given as $\mathbf{\Sigma}_t^{R_k} \in \mathbb{R}^{6\times6}$. Each UAV R_k has an on-board, monocular, perspective camera, rigidly attached to the UAV's body frame. These cameras do not independently pan, tilt or physically zoom, i.e., they are not attached using a gimbal.

Let the position of the tracked person in the world frame at the time t be given by $\mathbf{x}_t^P \in \mathbb{R}^3$. We assume that the person is represented by the position of its centroid in the 3D Euclidean space. The uncertainty covariance matrix associated with it is given by $\mathbf{\Sigma}_t^P \in \mathbb{R}^{3\times3}$. Note that all variables are in the world frame coordinates, unless a left-superscript is used (e.g., I for image frame).

2.3.3 DNN-based Cooperative Detection and Tracking

Algorithm 1, which is a recursive loop, outlines our CDT approach for UAV R_k. Each UAV runs an instance of this algorithm in real time. The algorithm is based on an EKF where the inputs are the tracked person's 3D position estimate $\mathbf{x}_{t-1}^P$ at the previous time step $t-1$, the covariance matrix $\mathbf{\Sigma}_{t-1}^P$ associated to that estimate and $\mathbf{ROI}_{t-1}^{R_k}$, also computed at $t-1$. The other input is the image $\mathbf{I}_t^{R_k}$ at t. Lines 1–11 correspond to the iteration of the algorithm at t.

Line 1 of Algorithm 1 performs the person detection using a DNN-based detector on a ROI $\mathbf{ROI}_{t-1}^{R_k}$ provided to it from the previous time step $t-1$ and the image $\mathbf{I}_t^{R_k}$ at the current time step t. Using raw detection measurements on the image (explained below), the detection measurement in the world frame is computed as a mean $\mathbf{z}_t^{P,R_k}$ and a noise covariance matrix $\boldsymbol{Q}_t^{P,R_k}$. The detection is performed on the latest available image and uses only the GPU. If this resource is busy processing a previous image, the detection (and therefore the EKF update, Line 6) is skipped. Note that the detection computation time of our DNN is independent of the size of the ROI. It operates on a fixed input size of

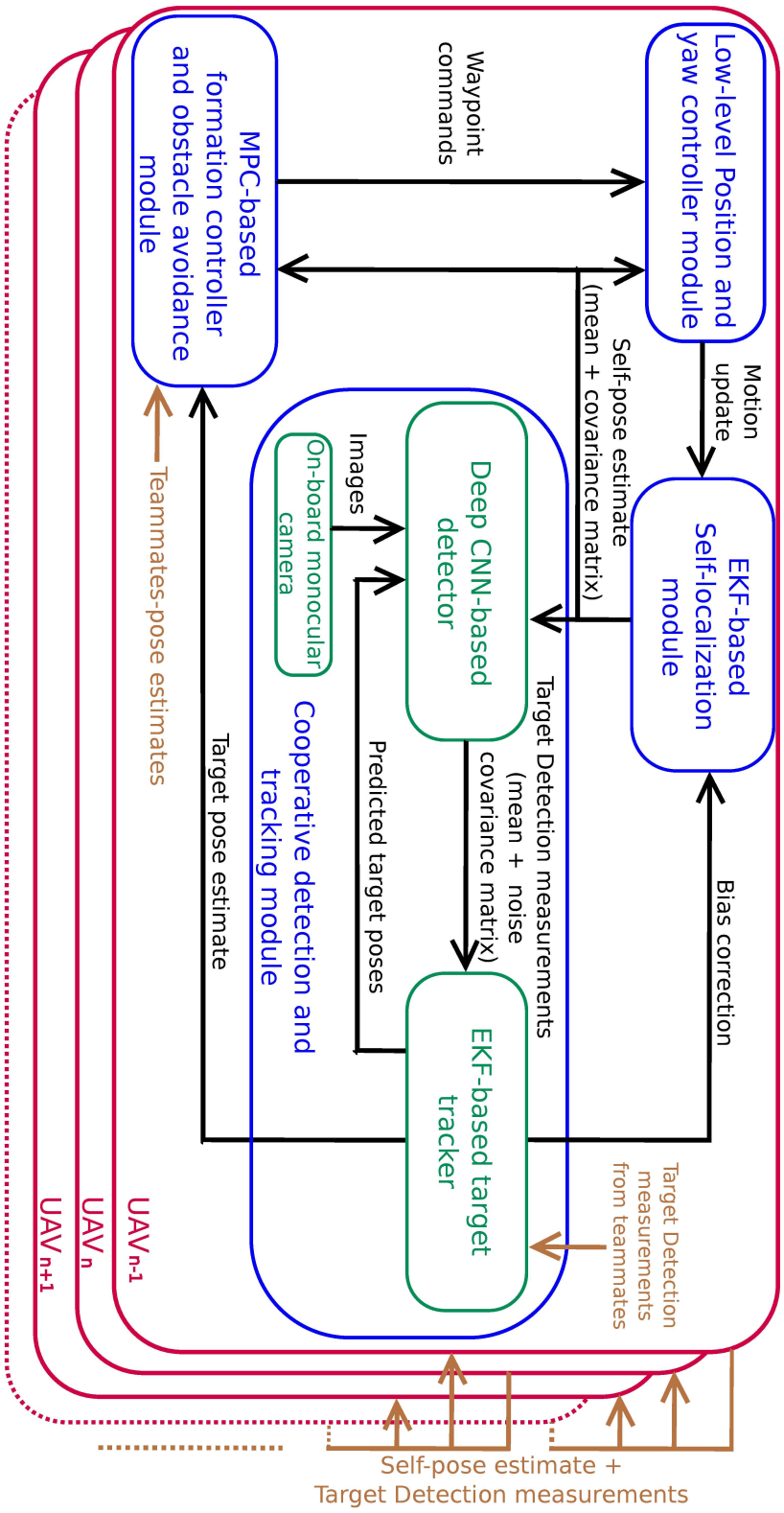


Figure 2.2: Overall architecture of our multi-UAV system.

Algorithm 1 Cooperative Detector and Tracker (CDT) on UAV R_k with inputs $\{\mathbf{x}^P_{t-1}, \mathbf{\Sigma}^P_{t-1}, \mathbf{ROI}^{R_k}_{t-1}, \mathbf{I}^{R_k}_t\}$

$\{\mathbf{z}^{P,R_k}_t, \boldsymbol{Q}^{P,R_k}_t\} \leftarrow$ DNN person detector $(\mathbf{I}^{R_k}_t, \mathbf{ROI}^{R_k}_{t-1})$.
Transmit $\{[\mathbf{x}^{R_k}_t \ \Phi^{R_k}_t]$, $\mathbf{\Sigma}^{R_k}_t$, $\mathbf{z}^{P,R_k}_t$ and $\boldsymbol{Q}^{P,R_k}_t\}$ to all UAVs R_m; $m = 1,...,K$; $m \neq k$
Receive $\{[\mathbf{x}^{R_m}_t \ \Phi^{R_m}_t]$, $\mathbf{\Sigma}^{R_m}_t$, $\mathbf{z}^{P,R_m}_t$ and $\boldsymbol{Q}^{P,R_m}_t\}$ from all other UAVs R_m; $m = 1,...,K$; $m \neq k$
$\{\bar{\mathbf{x}}^P_t, \bar{\mathbf{\Sigma}}^P_t\} \leftarrow$ EKF Prediction $\{\mathbf{x}^P_{t-1}, \mathbf{\Sigma}^P_{t-1}\}$ using exponentially decelerating state transition model.
for $m = 1$ to K **do**
    $\{\mathbf{x}^P_t, \mathbf{\Sigma}^P_t\} \leftarrow$ EKF Update $\{\bar{\mathbf{x}}^P_t, \bar{\mathbf{\Sigma}}^P_t, \mathbf{z}^{P,R_m}_t, \boldsymbol{Q}^{P,R_m}_t\}$
end for
$\{\hat{\mathbf{x}}^P_{t+1}, \hat{\mathbf{\Sigma}}^P_{t+1}\} \leftarrow$ Predict for next ROI $\{\mathbf{x}^P_t, \mathbf{\Sigma}^P_t\}$
$\mathbf{ROI}^{R_k}_t \leftarrow$ Calculate next ROI$\{\hat{\mathbf{x}}^P_{t+1}, \hat{\mathbf{\Sigma}}^P_{t+1}\}$
Update self-pose bias$\{\mathbf{z}^{P,R_k}_t, \boldsymbol{Q}^{P,R_k}_t, \mathbf{x}^P_t, \mathbf{\Sigma}^P_t\}$
return $\{\mathbf{x}^P_t, \mathbf{\Sigma}^P_t, \mathbf{ROI}^{R_k}_t\}$

300×300 pixels and takes approximately 250ms for one detection on our dedicated onboard hardware. To enable this, we scale down the ROI (obtained through our approach, as explained further in the algorithm) to 300×300 pixels regardless of the ROI's original size before the DNN takes it as input.

Raw detection measurements on the image consist of a set of rectangular bounding boxes with associated confidence scores and a noise covariance matrix. To obtain a model of this noise, we performed a characterization of the DNN-based detector, explained in Section 2.3.3. The raw detection measurements are transformed first to the camera coordinates and finally to the world coordinates. This transformation incorporates the noise covariances in the raw detection measurements and the UAV's self-pose uncertainty covariance. Note that the raw measurements are in the 2D image plane, whereas the final detection measurements are computed in the 3D world frame. For this, we make a further assumption on the height of the person being tracked. We assume that the person's height follows a distribution $H \sim \mathcal{N}(\mu_H, \sigma^2_H)$. While we assume that the person is standing or walking upright in the world frame, the model can be adapted to consider a varying pose (e.g., sitting down), for instance, by increasing σ^2_H. Finally, the overall transformation of detections from image frame to world frame measurements also takes σ^2_H into account. For mathematical details regarding the transformations that include propagating noise/uncertainty covariances, we refer the reader to [68]. Note that we also do not assume that the tracked person is on a flat surface. The terrain could be

uneven or sloped. As the 3D world coordinate refers to the GPS and barometer-derived world coordinate system used by the UAV to self-localize, the assumption on the human height distribution suffices to compute the 3D position of the person in the GPS world coordinate system.

In Lines 2–3 of Algorithm 1 we transmit and receive data among the robots. This includes self-pose estimates (used for inter-robot collision avoidance) and the detection measurements, both in the world frame. Line 4 performs the prediction step of the EKF. Here, we use an exponentially decelerating state transition model. When the algorithm continuously receives measurements allowing continuous update steps, predictions behave similar to a constant velocity model. However, if the subject is not detected for a prolonged time, the exponential decrease in velocity allows the tracked person's position estimate to become stationary. This is an important property of our CDT approach, as the ROI is calculated from the tracked person's position estimate (Line 8). In the case of having no detection measurements, the uncertainty in the person's position estimate would continuously grow, resulting in a larger ROI. This is further clarified in the explanation of Lines 8–9.

In Lines 5–7 we fuse measurements from all UAVs (including self-measurements). Since these measurements are in the world frame, fusion is done by simply performing an EKF update for each measurement.

In Lines 8–9 of Algorithm 1 lies the key novelty of our approach. We actively select a ROI ensuring that future detections are performed on the most informative part of the image, i.e., where the person is, while keeping the computational complexity independent of camera image resolution. As the computational complexity of DNN-based detectors grows very fast with the image resolution, using our approach we can still use a DNN-based method in real-time and with high detection accuracy. The ROI is calculated as follows. First (Line 8), using a prediction model similar to the EKF prediction and the estimates at the current time step ($\mathbf{x}_t^P$ and $\mathbf{\Sigma}_t^P$), we predict the state of the person in the following time step $t+1$ as $\{\hat{\mathbf{x}}_{t+1}^P, \hat{\mathbf{\Sigma}}_{t+1}^P\}$. Then, in Line 9, using the predicted 3D position of the person, we calculate the position and associated uncertainty of the person's head $\{\hat{\mathbf{x}}_{t+1}^{P_h}, \hat{\mathbf{\Sigma}}_{t+1}^{P_h}\}$ and feet $\{\hat{\mathbf{x}}_{t+1}^{P_f}, \hat{\mathbf{\Sigma}}_{t+1}^{P_f}\}$. For this, we assume the height distribution model for a person, as introduced previously, and that the person is in an upright position. $\hat{\mathbf{x}}_{t+1}^P$, $\hat{\mathbf{x}}_{t+1}^{P_h}$ and $\hat{\mathbf{x}}_{t+1}^{P_f}$ are back-projected onto the image frame along with the uncertainties and are denoted as $\{{}^I\hat{\mathbf{x}}_{t+1}^P, {}^I\hat{\mathbf{\Sigma}}_{t+1}^P\}$ (person's centre), $\{{}^I\hat{\mathbf{x}}_{t+1}^{P_h}, {}^I\hat{\mathbf{\Sigma}}_{t+1}^{P_h}\}$ (head) and $\{{}^I\hat{\mathbf{x}}_{t+1}^{P_f}, {}^I\hat{\mathbf{\Sigma}}_{t+1}^{P_f}\}$ (feet). The Centre-top pixel of the ROI is now given by

$$\mathbf{ROI}_t^{R_k}\{\textbf{centre, top}\} = ({}^I\hat{x}_{t+1}^P, {}^I\hat{y}_{t+1}^{P_h} + 3\,{}^I\hat{\sigma}_{\boldsymbol{y}_{t+1}^{P_h}}) \tag{2.1}$$

and the Centre-bottom pixel is given by

$$\mathbf{ROI}_t^{R_k}\{\textbf{centre, bottom}\} = ({}^I\hat{x}_{t+1}^P, {}^I\hat{y}_{t+1}^{P_f} - 3\,{}^I\hat{\sigma}_{\boldsymbol{y}_{t+1}^{P_f}}), \tag{2.2}$$

where ${}^{I}\hat{x}^{P}_{t+1}$ is the x-axis component of ${}^{I}\hat{\mathbf{x}}^{P}_{t+1}$. ${}^{I}\hat{y}^{P_h}_{t+1}$ and ${}^{I}\hat{y}^{P_f}_{t+1}$ are the y-axis components of ${}^{I}\hat{\mathbf{x}}^{P_h}_{t+1}$ and ${}^{I}\hat{\mathbf{x}}^{P_f}_{t+1}$, respectively. ${}^{I}\hat{\boldsymbol{\sigma}}_{\mathbf{y}}{}^{P_h}_{t+1}$ and ${}^{I}\hat{\boldsymbol{\sigma}}_{\mathbf{y}}{}^{P_f}_{t+1}$ are the standard deviations in the y-axis computed directly from ${}^{I}\hat{\boldsymbol{\Sigma}}^{P_h}_{t+1}$ and ${}^{I}\hat{\boldsymbol{\Sigma}}^{P_f}_{t+1}$, respectively. Lastly, the left and right borders of the ROI are calculated to match a desired aspect ratio. We chose the aspect ratio $(4:3)$ that corresponds to the majority of the training images in [23] for optimal detection performance.

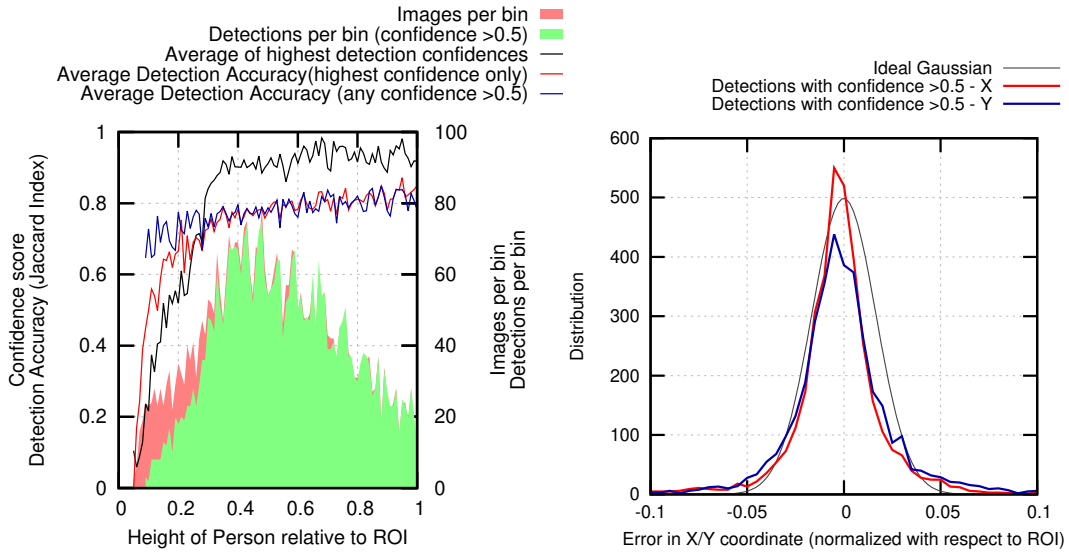
Line 10 of Algorithm 1 performs the bias update of the UAV self-pose. Self-pose estimates obtained using GPS, barometer, and IMU sensors (as is the case of our self-localization system) often have a time-varying bias [69]. The self-pose biases of each UAV cause mismatches when fusing detection measurements in the world frame, which are detrimental to our MCDT approach because i) the calculated ROI for person detection will be biased and will often not include the person, and ii) the person's 3D position estimate will be a result of fusing biased measurements. To truly benefit from our MCDT approach, we track and compensate the bias in each UAV's localization system (see 2.3.1). We track these biases using an approach based on [69]. The difference between a UAV's own detection measurements and the tracked estimate (after fusion) is used to update the bias estimate. Our approach is a simplified form of [69] as we only track position biases, which is sufficient if the bias in orientation is negligible.

Noise Quantification of a DNN-based Object Detector

Our MCDT approach depends on a realistic noise model of the person detector. To this end, we perform its noise quantification. Here we describe the procedure followed for the pre-trained SSD-Multibox detector[19], used in our work.

The output of the detector is, for each detection, a bounding box, a class label and a confidence score. The input image size is defined at training time. We used a pre-trained 300×300 pixel network (SSD300 trained on the PASCAL VOC 2007+12 dataset [19]) in this work. We quantify the detection noise w.r.t. the size of the detections, i.e., smaller and distant, or larger and closer to the camera. To this end, we created an extended test set from the PASCAL VOC 2007 dataset [23] with varying levels of downscaling and upscaling of the detected person, similar to SSD's training data augmentation in the learning phase. We selected images with a single, non-truncated person to avoid the detection association problem.

We perform statistical analysis using the pre-trained person detector on our extended test. As the test set has images of different sizes, we calculate all measures relative to their image size. Figure 2.3 (a) shows the detection accuracy, using the Jaccard index, relative to the person's height. It is evident that even though the detection accuracy for a given minimum confidence threshold is nearly constant w.r.t. the analysed ROI, the absolute error decreases with a smaller ROI. The chance of successful detection falls significantly for relative sizes below 30% of the ROI and goes down to zero at 10%, thus forming an upper boundary for the desired ROI size. This analysis clearly justifies the



(a) (b)

Figure 2.3: (a) Detection accuracy w.r.t. the relative person height in the ROI. Images are divided into bins according to the height. (b) Error distribution of SSD-Multibox detections in X and Y components.

necessity of our MCDT approach with active selection of ROIs.

Further analysis is presented in Figure 2.3 (b), where we show the relative error in the detected person's position over all test images. The error is shown to be well described by a Gaussian distribution. We perform similar analysis on the errors of each detection for the top, bottom, left and right-most points of the detection bounding box at different relative sizes. We find all these errors to be similarly well described by Gaussian distributions, without significant correlations between them. Thus, the person-detection noise can be well approximated by a constant variance model. We obtain the following variances for the detection noise of SSD-Multibox, measured for each side of the detection bounding boxes and normalized w.r.t the size of the ROI: $\sigma^2_{top} = 0.0014$, $\sigma^2_{bottom} = 0.0045$, $\sigma^2_{left} = 0.0039$ and $\sigma^2_{right} = 0.0035$.

2.3.4 MPC-based formation controller and obstacle avoidance module

This subsection describes Aamir Ahmad's contribution to [20]. A key requirement for a vision-based outdoor human-motion capture system is to not lose the tracked target from the field of view of the cameras of all UAVs. To this end, a perception-driven formation controller (FC) and obstacle avoidance module, similar to [70, 71] has been implemented. Although this FC is not a novel research contribution in itself, it is briefly

described in this subsection, to facilitate the understanding of the proposed overall system architecture and how the novel MCDT approach fits within it.

The FC is designed as follows. The objective of each UAV is to maintain i) a pre-specified horizontal (normal to gravity direction) distance d_{per} to the tracked person, ii) a pre-specified altitude h_{fix} (in world frame) and iii) its yaw orientation towards the tracked person. Additionally, each UAV must adhere to the constraints of i) maintaining a pre-specified collision distance threshold d_{col} from every teammate UAV and ii) staying within a pre-specified bounding box (e.g., legally permitted flyable zone).

Algorithm 2 outlines the FC strategy. Each UAV R_k runs an instance of Algorithm 2 at every time step t. In line 1, the UAV R_k computes its desired pose $\check{\mathbf{x}}_t^{R_k}$ using simple trigonometry. In line 2, an MPC based planner similar to [70] solves a 3D optimal control problem (OCP) with receding time horizon of size N. We consider nominal accelerations $\left[\mathbf{u}_t^{R_k}(0)\cdots\mathbf{u}_t^{R_k}(N)\right]^\top$ as the input vector of the OCP describing the translational motion of the UAV. The discrete-time state of the OCP consists of the UAV R_k's position $\mathbf{x}_t^{R_k}(n)$ and velocity $\dot{\mathbf{x}}_t^{R_k}(n)$. The discrete-time state-space equations at the n^{th} MPC step (different from time step t) with sampling time Δt are given by

$$\mathbf{u}_t^{R_k}(n) = \ddot{\mathbf{x}}_t^{R_k}(n) \tag{2.3}$$

$$\left[\mathbf{x}_t^{R_k}(n+1)\ \dot{\mathbf{x}}_t^{R_k}(n+1)\right]^\top = \mathbf{A}\left[\mathbf{x}_t^{R_k}(n)\ \dot{\mathbf{x}}_t^{R_k}(n)\right]^\top + \mathbf{B}\mathbf{u}_t^{R_k}(n) \tag{2.4}$$

where $\mathbf{A} = \begin{bmatrix}\mathbf{I}_3 & \Delta t\mathbf{I}_3\\ \mathbf{0}_3 & \mathbf{I}_3\end{bmatrix}$, $\mathbf{B} = \begin{bmatrix}\frac{\Delta t^2}{2}\mathbf{I}_3\\ \Delta t\mathbf{I}_3\end{bmatrix}$.

The OCP is described by the following convex quadratic cost function.

$$\begin{aligned} J_{\text{OCP}} = &\underset{\mathbf{u}}{\mathbf{arg\,min}}\ \Big(\big(\sum_{n=0}^{N}(\|\mathbf{u}_t^{R_k}(n)\|^2_{\Omega_C})\big)+\\ &\|\left[\mathbf{x}_t^{R_k}(N+1)\ \ \dot{\mathbf{x}}_t^{R_k}(N+1)\right]^\top - \left[\check{\mathbf{x}}_t^{R_k}\ \ \mathbf{0}\right]^\top\|^2_{\Omega_{term}}\Big), \end{aligned} \tag{2.5}$$

subject to the dynamics mentioned above and the following state and input constraints.

$$\mathbf{u}_{\min} \le \mathbf{u}_t^{R_k}(n) \le \mathbf{u}_{\max} \tag{2.6}$$

$$\mathbf{x}_{\min} \le \mathbf{x}_t^{R_k}(n) \le \mathbf{x}_{\max},\ \ \dot{\mathbf{x}}_{\min} \le \dot{\mathbf{x}}_t^{R_k}(n) \le \dot{\mathbf{x}}_{\max}. \tag{2.7}$$

Ω_{term} and Ω_C are the weight matrices for terminal state and input cost. The terminal state is set to be the computed desired position $\check{\mathbf{x}}_t^{R_k}$ and zero velocity. The MPC solver results in the optimal control inputs $\left[\mathbf{u}_t^{R_k}(0)\cdots\mathbf{u}_t^{R_k}(N)\right]^\top$ and the corresponding trajectory $\left[\mathbf{x}_t^{R_k}(1)\ \dot{\mathbf{x}}_t^{R_k}(1)\cdots\mathbf{x}_t^{R_k}(N+1)\ \dot{\mathbf{x}}_t^{R_k}(N+1)\right]^\top$ towards the desired position (except the desired yaw). We use the first step position component $\mathbf{x}_t^{R_k}(1)$ of the output trajectory

as the input to the low-level flight controller (line 9). In line 3, the desired yaw angle is calculated using simple trigonometry with the next 3D position of the UAV and the current pose of the person.

The MPC solver considers obstacle-free space. This is done to avoid non-convexity introduced by considering obstacles directly within the MPC formulation. Subsequently, to account for dynamic obstacles in the environment (teammate UAVs), we use the classical potential-field based obstacle avoidance algorithm on top of the MPC solution. Lines 3–5 check the presence of teammate UAVs within a distance threshold d_{col} of the UAV R_k to activate the avoidance and re-compute $\mathbf{x}_t^{R_k}(1)$ as $\hat{\mathbf{x}}_t^{R_k}(1)$. Consequently, the desired yaw angle is also recomputed in line 6. Recall that UAVs communicate their self-pose to each other over wireless network. Generalization of the avoidance method to static obstacles in the environment would be straightforward, but would require every UAV to have an obstacle detection sensor and algorithm. An alternative approach would be to enforce a formation geometry that makes the UAVs keep a distance between each other. However, we did not employ that approach for two main reasons. First, its formulation leads to non-convex constraints and requires a non-convex MPC-based (NMPC) approach. NMPC has also been addressed in the literature, including previous work by Ahmad et al. [71]. While some methods handle non-convexity by providing only locally optimal solutions [72], others solve it using a mixed integer program, which can suffer computationally with the increase in the number of robots [73]. Second, by following a fixed geometry we could be limiting the viewpoints of the UAVs from where the tracked person is visible, e.g., due to static obstacles between the UAV and the tracked person.

Finally, either in line 7 or 9, way-point commands consisting of the next time-step pose and desired yaw angle are sent to the low-level flight controller. The latter is essentially assumed to be a position and yaw controller. Notice that we do not specify a UAV formation geometry to the formation controller. The MPC and potential field combination of Algorithm 2 naturally results in a dynamic formation that keeps the tracked person within the field of view of all UAVs.

Algorithm 2 MPC-based formation controller and obstacle avoidance on UAV R_k with inputs $\{\mathbf{x}_t^P,\ \mathbf{x}_t^{R_m}; m = 1:K\}$

$\{\check{\mathbf{x}}_t^{R_k}\} \leftarrow$ Compute Destination Pose $\{\mathbf{x}_t^P, \mathbf{x}_t^{R_k}, d_{\text{per}}, h_{\text{fix}}\}$
$\{\mathbf{x}_t^{R_k}(1)\} \leftarrow$ Solve MPC for $\{\check{\mathbf{x}}_t^{R_k}, \mathbf{x}_t^{R_k}\}$
$\{\gamma_t^{R_k}(1)\} \leftarrow$ Compute Desired Yaw $\{\mathbf{x}_t^{R_k}(1), \mathbf{x}_t^P\}$
if $\mathbf{x}_t^{R_k}(1)$ within collision distance threshold (d_{col}) for any $\{\mathbf{x}_t^{R_m}; m = 1:K, m \neq k\}$ **then**
    $\{\hat{\mathbf{x}}_t^{R_k}(1)\} \leftarrow$ Potential field avoidance $\{\mathbf{x}_t^{R_k}(1)\}$
    $\{\hat{\gamma}_t^{R_k}(1)\} \leftarrow$ Re-compute Desired Yaw $\{\hat{\mathbf{x}}_t^{R_k}(1), \mathbf{x}_t^P\}$
    Transmit $\hat{\mathbf{x}}_t^{R_k}(1), \hat{\gamma}_t^{R_k}(1)$ to Low-level Controller.
else
    Transmit $\mathbf{x}_t^{R_k}(1), \gamma_t^{R_k}(1)$ to Low-level Controller.
end if

2.4 Hardware and System design

2.4.1 Multimaster ROS

Our robotic system is based on ROS [28]. When the work was started in 2016, this was ROS "Kinetic Kame" released in May 2016, and it was iteratively upgraded to each new ROS version up to "Noetic Ninjemys", the last ROS1 version released in 2020. ROS is a network-based system in which programs called "nodes" connect to each other via TCP/IP networking. Data is exchanged on a message-based protocol. Each node can subscribe and publish messages under so-called topics. A master process keeps track of all nodes, and which messages they publish. Each publication has a related TCP server, operated by the node, that subscribers can connect to. The master process will tell each node under which network address and port they can find the publishers they wish to subscribe to. For a flying formation, a single master process would be a single point of failure, potentially disrupting the entire network if the communication link is unreliable, even for messages originating and received on the same physical computer on the same robot. For this, we used an implementation called "FKIE-Multimaster" [29] by Fraunhofer, which establishes a distributed network of master processes, one on each robot. This ensures that on-robot message coordination is ensured even in the presence of communication problems, and at the same time, inter-robot bandwidth is minimized.

Each robot is equipped with a main computer running Ubuntu Linux and an FKIE ROS master process, communicating via Wi-Fi with IPv4 multicast networking set up. A base-station laptop is used to monitor and control the formation, it also runs ROS nodes for visualization and its own FKIE master process.

In addition to ROS, we run NTP processes for time synchronization. The base-station laptop runs an NTP master server, to which all other robots time-synchronize. This ensures all clocks are running synchronous and absolute timestamps between robots are directly comparable.

2.4.2 Networking

We use 5 GHz Wi-Fi (IEEE 802.11ax) for the physical network layer. The 5 GHz band is used to keep the 2.4 GHz band free for radio remote control of the UAVs for manual flight, respectively manual override. This ensures sufficient bandwidth over ranges up to 100m. The base-station can act as a Wi-Fi Access point to facilitate the managed network. We later used a dedicated Wi-Fi repeater as a physical (layer 2) network coordinator.

2.4.3 Flight Hardware

The base for our multi-copter robots is a Mikrokopter "Octo" frame. It consists of 8 metal arms joint to a modular central assembly which houses batteries, power electron-

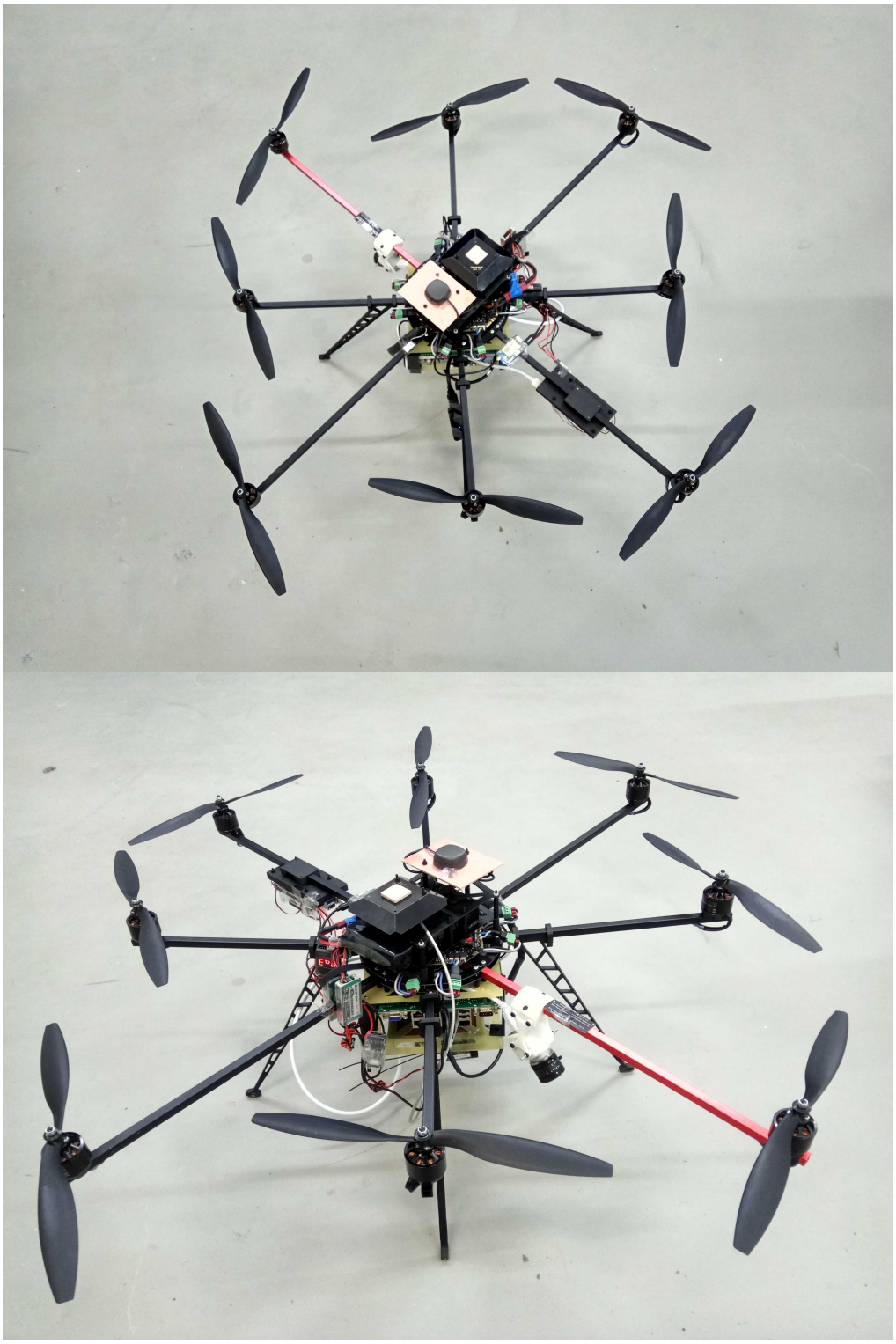


Figure 2.4: View of a single octo-copter robot. These photos show the improved and retrofitted 2018 design with GPS receivers on top and shielded by the main battery. The photos were taken in our lab at MPI Tübingen, Germany in 2019.

ics and computers. 4 of the arms have landing legs attached. The forward-facing arm has a main camera attached, pointing forward and 45° down. The rearward facing arm holds an NVIDIA Jetson TX 1 GPU on an Auvidea J120 embedded carrier board, as seen in Figure 2.4. It is connected via Ethernet cable to the main computer, which is an Intel Micro ATX Server board. We control the robot with an Openpilot Revolution [31] flight controller, which is based on a STM32F4 microcontroller for processing, running the FreeRTOS [74] real time operating system. Due to voltage differences between the Mikrokopter electronic speed controller and the signal level used by the OpenPilot Revolution, we added our own signal converter board which converts I2C [75] signals from 5V to 3.3V signal level and back. A Graupner HoTT™ receiver for manual flight control, connected to the OpenPilot Revolution, completes the flight electronics. Power is supplied from a single 4S LiPo, which is connected to the engine controller, supplies the main computer and the Jetson GPU via two redundant 12V switching voltage converters, and the flight controller and receiver with a dedicated 5V switching voltage converter.

The camera is connected to the main computer using USB3. Usage of USB3 with a high data rate creates noise ion the 1.5 GHz spectrum, which affects GPS signals. For that reason, both the main GPS receiver for flight control and a differential GPS receiver used for ground-truth positioning are mounted on top of the main battery, which is thereby exploited for EM shielding. Shielded data and power wires supply the GPS receivers and connect the flight control GPS receiver to the flight controller. The flight controller is mounted at the far bottom of the main electronics assembly, furthest from high current lines. This minimizes magnetic disturbances, which affect the compass, used to determine orientation of the vehicle by the flight controller.

The flight controller is connected via USB to the main computer, which allows autonomous control through ROS and makes the flight data available to ROS in real-time.

The whole system, including battery, has a mass of approx 4.5 kg and a thrust-to-weight ratio (with full batteries) exceeding 2.0.

2.4.4 Flight Software

An Extended Kalman Filter (EKF) on the flight controller fuses data from its inbuilt IMU, compass, an integrated barometric pressure sensor and the attached GPS receiver to determine a state estimate and uncertainty covariance matrix including orientation of the vehicle in space, represented as an orientation quaternion, position, and velocity in a NED coordinate system relative to a stationary reference geoposition.

The flight controller controls the engines using a constant allocator matrix from 4 pseudo-control-channels for roll, pitch, yaw, and thrust. These pseudo-control-channels receive their values from a cascaded PID controller, controlling orientation. The thrust is controlled open-loop.

During manual, human-controlled flight, the set point of this cascaded PID controller and thrust are fed directly from the stick positions on the remote control, allowing direct attitude control by a remote human pilot.

When in autonomous flight, the set points for the attitude PID control come from PID velocity control loops for speed in northern and eastern direction — rotated by the current vehicle orientation. The thrust is determined by a vertical velocity control loop.

A position control loop, which is a simple proportional term on the position error with bounds on the maximum allowed velocity, sets the set point for velocity based on the current 3D position of the vehicle and a "waypoint"

When controlled by ROS, this "waypoint" and the yaw-orientation is fed via USB from ROS. For this, we developed a software module on the flight controller and a matching ROS node that feeds this data to the flight controller, and also provides the EKF state estimate and other runtime data such as remote control stick position and auxiliary channels to ROS. This allows the remote control to be used as a human-machine interface for robot control if desired.

A desired property of this setup is, that a fault in the main computer — which means no more data is being sent to the flight controller — results in the robot maintaining the last received waypoint until normal operation is restored or a human intervenes. This is inherently safe behaviour.

The flight firmware code is open-source and can be found in [38].

2.4.5 ROS architecture

Each component in Figure 2.2 in subsection 2.3.1 is implemented in separate ROS nodes. This includes.

- Camera: A ROS node for reading data from the camera. Since the camera (various BASLER and PtGrey (now FLIR) models) uses proprietary control and data transfer protocols, camera-specific ROS nodes were developed, which read data from the camera, save losslessly compressed raw video and metadata to an SSD disk for later analysis, and then provide the video feed to other ROS nodes in the form of timestamped "image" messages. The timestamp is computed from the shutter timestamp of the camera and reflects when the image was taken, not when it was processed.

- Camera-Info: A ROS node that provides metadata for the camera such as resolution and — previously calibrated — intrinsics, focal point and distortion parameters.

- Transformations: A ROS node that provides static coordinate information that is robot-specific, such as geometry (position and orientation of the camera relative to the flight controller/IMU, etc.)

- Librepilot Bridge: A ROS node that reads data from the flight controller via USB and makes the state estimate available as ROS messages. This ROS node adds an offset to the copter's position to compensate for known GPS errors, which it

subscribes to from the tracking Extended Kalman Filter ROS node. The Librepilot bridge also forwards waypoints and desired orientation to the flight controller, which performs low-level flight control.

- Projection Model: A ROS node that projects 2D camera coordinates with uncertainty into 3D world coordinates with uncertainty, as well as the other way around. It computes depth based on a crude size-based model and adds a lot of uncertainty. This ensures that a 2D detection in a camera frame is available as a 3D detection to the tracking filter, and a 3D subject state estimate is available as a 2D region of interest in the current camera image.

- Neural network detection: This node crops the current region of interest from the current camera image, and sends this image, scaled to the native neural network input layer size, over a dedicated TCP connection to the neural network server. In the case of the octo-copters, the neural network server runs on the NVIDIA Jetson TX1 module connected via Ethernet. Any detection results that are above a detection confidence threshold and of the right class are published as ROS messages, which are then projected into 3D by the projection model node.

- Target Tracker: This ROS node implements an extended Kalman filter that fuses detections by all robots (Algorithm 1). Due to the latency of the camera and the delay caused by neural network processing, these detections are delayed between 0.2 and 0.4 seconds. The preserved timestamp of the shutter time of the corresponding camera, which is part of all ROS messages such as detections and 3D detections, is used to sort updates and re-compute the state estimate asynchronously, which supports backtracking in case messages arrive out of order. The Kalman filter also computes the self-pose GPS offset based on the discrepancy between a robot's own detections and the fused state estimate involving all robots.

- Formation Controller: The MPC ROS node computes the optimal trajectory for all robots in the formation based on corrected self pose estimates of the robot itself and pose information received over Wi-Fi from the peers, as well as the subject tracking estimate (Algorithm 2). From this trajectory, a waypoint is calculated, taking the low-level flight controller's position control P coefficient into account, to indirectly send a velocity command to the flight controller. This indirect approach maintains the inherent safety of waypoint based low-level control, while still giving the MPC fine control over the robot velocity.

The ROS code is open-source and can be found in [21], including implementations for SITL and HITL simulation.

2.4.6 Software in the loop simulation.

When simulating, typically we run only a single ROS master process on the simulating computer. The Neural Network Server can serve multiple neural network detector ROS nodes given a sufficiently fast desktop GPU for processing, and the detection speed is throttled to achievable in-flight speed by software-delays.

Librepilot is replaced by a software emulated copter in the Gazebo simulator. Wrapper-nodes transform all relevant ROS messages into compatible messages.

The camera image is provided by a simulated camera, which renders its image based on the Gazebo environment. This allows software in the loop simulation of the whole formation.

2.5 Experiments and Results

2.5.1 Hardware, Software and the Experimental Setup

To evaluate our approach, we conducted real robot experiments on a team of two self-designed 8-rotor UAVs as described in Section 2.4 (see Fig.2.1) tracking a person. We use the flight controller's position and yaw controller, as well as its GPS and IMU-based self-pose estimation (localization) functionalities (see the first two modules of our system architecture in Figure 2.2). Self-pose estimates and IMU sensor data are updated and logged at $100Hz$.

We also mounted an Emlid Reach differential GPS receiver on each UAV and 2 such receivers on either shoulder of the tracked person to obtain the ground truth (GT) position estimates of the UAVs as well as the person. Note that the data from this differential GPS was not used online for flight control or target state estimation during the experiments.

Each UAV continuously captures and stores images from the camera at $40Hz$. The on-board GPU runs SSD-Multibox [19]. As described in Section 2.3, ROIs of images down-sampled to 300×300 pixel are sent to the GPU. The detection frame rate achieved is $3.89Hz$. Using these detections, each UAV runs Algorithms 1 and the FC, described in the previous section, to achieve a perception-driven formation to track and follow a person walking on the ground with variable speeds and in varying trajectories. In all flight experiments we set $d_{\text{col}} = 5\text{m}$, $d_{\text{per}} = 6\text{m}$ and $h_{\text{fix}} = 6\text{m}$ above the origin of the experiment field.

As described in Section 2.4, our UAVs know the position of their teammates by communicating their self-pose estimates over the wireless network. To prevent inter-UAV collisions, teammate positions are used in the potential field-based method as described previously. Moreover, to compensate for their self-pose inaccuracies, the UAVs need to avoid each other with a high enough distance margin. It is therefore necessary that at least nearby robots can communicate with enough bandwidth to avoid colliding with each other. Detection measurements of the person and the UAVs self-pose estimates are very-low-bandwidth data points. We can therefore assume that congestion will not occur

	Twilight		Noon	
	Mean (m)	Var (m^2)	Mean (m)	Var (m^2)
3D error	1.36	0.19	2.06	0.30
2D error	0.91	0.23	1.67	0.47

Table 2.1: Tracked person's world-frame estimation errors w.r.t. GT.

	Twilight		Noon	
	Mean (m)	Var (m^2)	Mean (m)	Var (m^2)
3D error	0.46	0.03	1.02	0.06
2D error	0.37	0.04	0.58	0.16

Table 2.2: UAV 1 self-localization Errors w.r.t. GT.

	Twilight		Noon	
	Mean (m)	Var (m^2)	Mean (m)	Var (m^2)
3D error	1.06	0.23	1.27	0.21
2D error	0.89	0.24	0.77	0.30

Table 2.3: UAV 2 self-localization Errors w.r.t. GT.

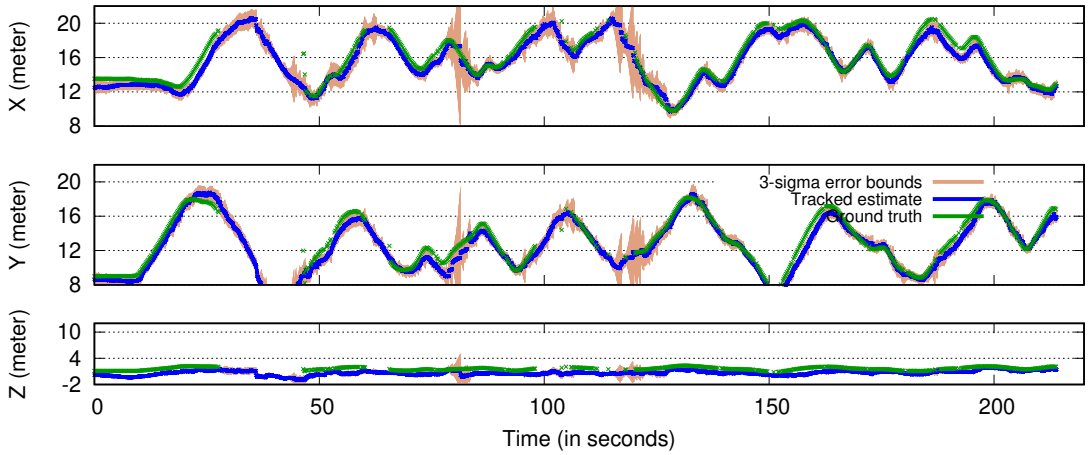
even in a relatively larger team of UAVs, and at least a percentage of all sent messages reach their destination at all times. Later in the subsection on simulation (see Sec.2.5.4) we show that there is no significant divergence in the tracked state estimate even if up to 80% of all messages are lost.

We performed 2 separate flight experiments. While the first flight experiment was performed at twilight (Twilight Experiment) and lasted 220s, the second was performed at noon (Noon Experiment) for 180s, both under clear skies.

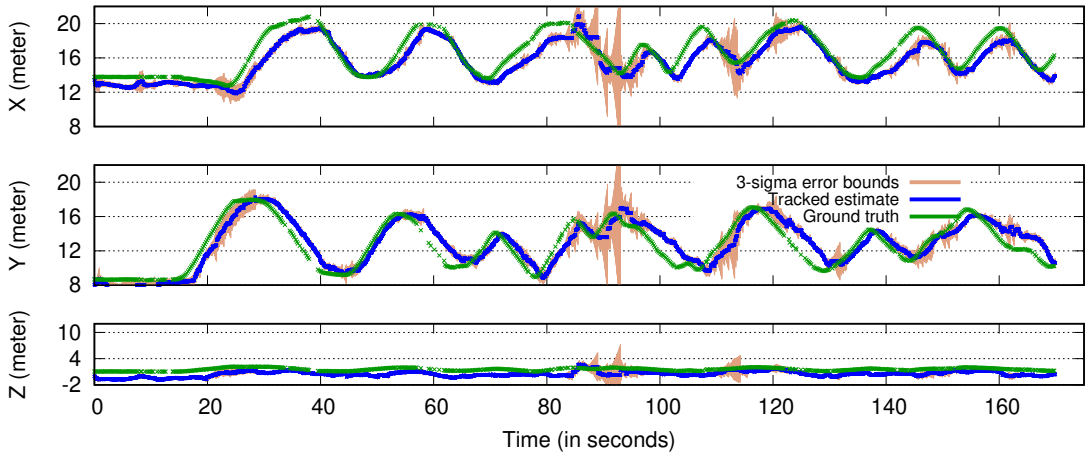
Note: All UAV independently estimate and record the position and trajectory of the tracked person in their on-board logs, based on data perceived by both on board sensors and received data from the other UAVs. If significant communication packet-loss were to occur, these logs could diverge and post-processing methods to fuse or reconcile the conflicting data would be needed. In our experiments, no divergence was observed, and the trajectories were identical within the accuracy displayed in the results. Therefore, no post-processing step was required and for simplicity, we only display and discuss the person trajectory estimate recorded by the first UAV.

2.5.2 Results and Comparisons w.r.t. Ground Truth

Results of the tracked position estimate of the person from both flight experiments are presented in Table 2.1, Figure 2.5 and in the accompanying video and multimedia material (see footnote on next page). The error is calculated as the difference between the position estimate of our proposed online method and the corresponding Ground Truth (GT) obtained by the differential GPS at each time-step. Let the errors in each dimen-



(a) Twilight Experiment



(b) Noon Experiment

Figure 2.5: Tracked person's trajectory comparison for both experiments.

sion of the person's position estimate be e_x, e_y and e_z. In Table 2.1, we present the mean and the variance of i) 3D Euclidean distance error ($\sqrt{e_x^2+e_y^2+e_z^2}$) and ii) 2D Euclidean distance error ($\sqrt{e_x^2+e_y^2}$). The supplementary videos display the tracked person's GT (in green) and the estimated trajectories (in blue) with 3-sigma error bounds (in pink).

From Tabs. 2.2 and 2.3, it is evident that the error in the self-pose estimate of the UAVs is high. This is mainly because during flight experiments, the UAVs compute a (biased) self pose estimate based on their regular on-board GPS receiver and IMU, both comparable to those used in consumer grade drones and smartphones. This regular GPS receiver contributes to the majority of the self localization error, which can be several meters and different for different GPS modules. Its accuracy also depends on how many

satellites it receives data from. Hence, it can vary over the course of a day. This is evident from the different self-pose accuracies of the same UAVs for noon and twilight experiments.

From Tab. 2.1, we see that the mean error in the tracked estimate is on the order of decimetres. This occurs mainly due to a noticeable high bias in the z-component of the estimate. Contributing factors to this are i) deviation between the actual and the assumed height of the person ii) on-board GPS being less accurate in the vertical than in the horizontal direction and iii) air pressure fluctuations caused by wind, degrading the UAV's self-localization by affecting the barometric altimeters. While the first reason is likely responsible for the constant part of the bias, the other two reasons contribute to drift in the bias over time.

When interpreting the tracking results from Tab. 2.1, it should be noted that all estimates and errors are in the world reference frame. As UAVs have significant self-localization errors of $\sim$ 1m (see Tabs. 2.2 and 2.3), the tracked person's world-frame estimates are heavily affected by them. This also implies that the person's tracked position estimate has a significantly lower error in the UAVs' local frames. Furthermore, the tracked estimate (see Tab. 2.1) has a low variance of 0.19 m^2 for the twilight experiment and 0.30 m^2 for the noon experiments in the 3D Euclidean errors. This shows that our method is quite precise in jointly tracking the target and keeping it in the field of view of both UAVs for the whole duration of the experiment, except for a few frames. This can be visualized in the image streams of both UAV's cameras[1]. The precision of the tracked estimate is also visible in the trajectory plots in these videos.

There is a visible time-lag between the person's GT and the estimated trajectory. This lag is more pronounced in the noon experiment. It is caused mostly by false negative detections, which happened much more in the noon experiment (see experiment footage). Processing an image frame on-board a UAV takes approximately 250ms, even if the detection is not successful. For example, algorithm 1 will perform an EKF update only $F + 1$ times 250ms after F consecutive unsuccessful detections followed by a successful one, assuming that meanwhile no teammate UAV performs successful detection. The prediction model of EKF, which models the motion dynamics of the tracked person, copes up with this lag to some extent, as the predictions are made at every image frame. However, the mismatch between this model and actual person motion, e.g., sudden change in person's walking direction, results in very visible estimation lag (especially noticeable in the video of the noon experiment and the corresponding trajectory plots in Figure 2.5 (b)). One of our future directions of research is to learn the motion model of the tracked person on-the-fly. Finally, it must be noted that we do not intend to have a swarm of UAVs as a practical method of providing centimetre-accurate absolute position tracking of the target. As described in the motivation of this chapter, our main

[1]Complete footage of the experiment:
Noon Experiment `https://youtu.be/_ZOsw5MWZiQ`
Twilight Experiment `https://youtu.be/LV_82bT25Bc`
Simulation `https://youtu.be/NHhyl9KnvRQ`

intent is to have a system that is suitable for human motion capture, for which we require the UAVs to have good relative localization w.r.t. the tracked target and most importantly have convergence and stability in the tracked target estimate. This is demonstrated from the reduced variance in the error using our approach as compared to the single vehicle baseline, evaluated in Subsection 2.5.3 (Figure 2.6) and 2.5.5 (Figure 2.7).

2.5.3 Results and Comparisons w.r.t. Baseline Methods

Using the recorded UAV camera images and the self-localization pose logs of both UAVs, we ran our MCDT method offline under two different situations. In the first situation, we considered only single UAVs and used the recorded images from each UAV, separately. Boxes in green and pink in Figure 2.6 correspond to this situation. In the second situation, we switched off only the self-pose bias correction (SPBC) feature (see Section 2.3.3) of our MCDT method (yellow boxes in Figure 2.6). The blue boxes represent the estimates from the actual online runs. Pattern-filled boxes summarize the localization errors of each UAV obtained during the online run and used during the aforementioned offline runs. The box-plots in Figure 2.6 show that both the cooperative aspect and the self-pose bias correction features of our method significantly reduce the tracking errors and make the estimator more precise. Most importantly, UAV 2 can keep a good person tracking estimate, despite being poorly localized. Note that the active ROI selection feature of our method is not directly comparable by running offline experiments on the recorded images and turning off the active ROI selection feature. This is because the images were recorded in the actual online run with that feature enabled, which is mainly responsible for keeping the person in the FOV of all UAVs. An offline run without active ROI selection on these images is biased, since the person is present in nearly all recorded images. Moreover, our preliminary flight tests without the active ROI selection resulted in UAVs not being able to detect and track the person most of the time.

2.5.4 False Detections and Other Limitations

Sparse false detections do not affect our method, since the EKF is generally robust to such sensor noise. Any potential false-positive in the environment outside the actively selected ROI would not be detected. However, multiple individuals within the ROI will create ambiguity and could lead to actual false positives. Therefore, within the ROI, we associate detections by discarding all but one, based on a comparison with the tracked estimate. Nevertheless, some failure cases are possible, e.g., i) if multiple individuals are present in the FOV before the EKF has converged on a stable estimate, or ii) if a UAV does not detect the true person for a prolonged period while continuously having false positive detections. In our approach, a UAV in such a situation can still recover to the correct person's estimate if enough other teammates have the true positive detections. Extending our approach to multiple-person pose tracking will involve more sophisticated data association techniques.

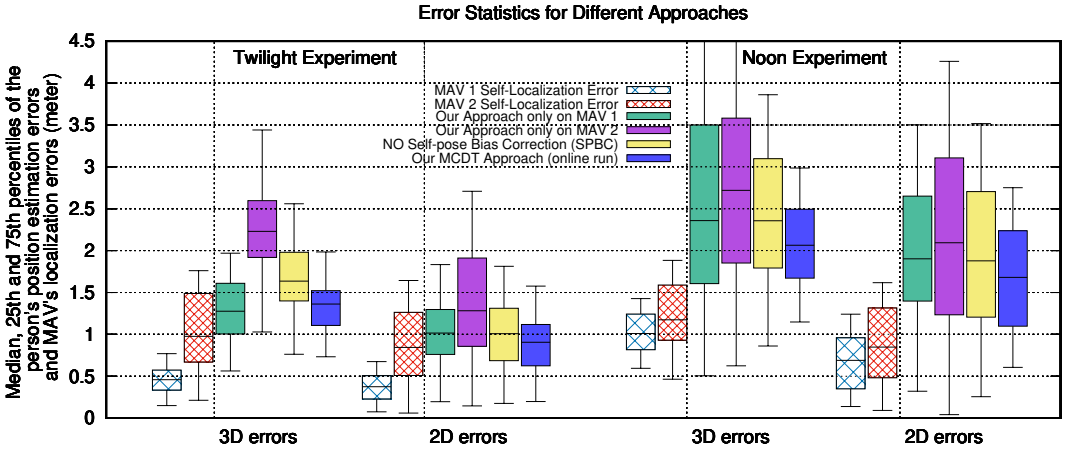


Figure 2.6: Box plots of the errors in different situations. Refer Section 2.5.3 for details. Note that the blue boxes of the online run also correspond to the plots in the accompanying videos.

2.5.5 Simulation Experiments Setup

Simulations were conducted as a software-in-the-loop setup on a single computer with a 12-core Intel® Core™ i7-3970X CPU @ 3.50GHz with an NVIDIA GK110 (GeForce GTX 780). All software components are ROS nodes, which communicate over TCP/IP networking. The detector runs for each simulated UAV, including delay logic to simulate the detection delay of the real-UAV hardware. Physics simulation was handled by Gazebo. The simulated UAVs were equipped with virtual cameras. Their parameters, e.g., frame rate, resolution, etc., were set to the same as the real UAVs' cameras. We achieved a simulation real-time factor of 0.15 with 16 UAVs. GPS inaccuracy and drift were simulated by superimposing a random-walk offset on the GT position of each simulated UAV to match the slowly drifting behaviour of the real UAVs GPS and IMU-based position estimate. During the experiment, we recorded each UAV's target position and self-pose estimate, as well as the corresponding GT directly available from Gazebo.

We define a simulation run as the following sequence: i) spawning of K simulated UAVs and a human actor (at the origin) in Gazebo, ii) waiting for all UAVs to detect the person, assume formation and converge on a stable estimate ($\sim$ 20s), iii) the simulated human actor starts walking on a predefined random trajectory within a 10×10m perimeter, iv) simulation and recording are stopped after 120 seconds.

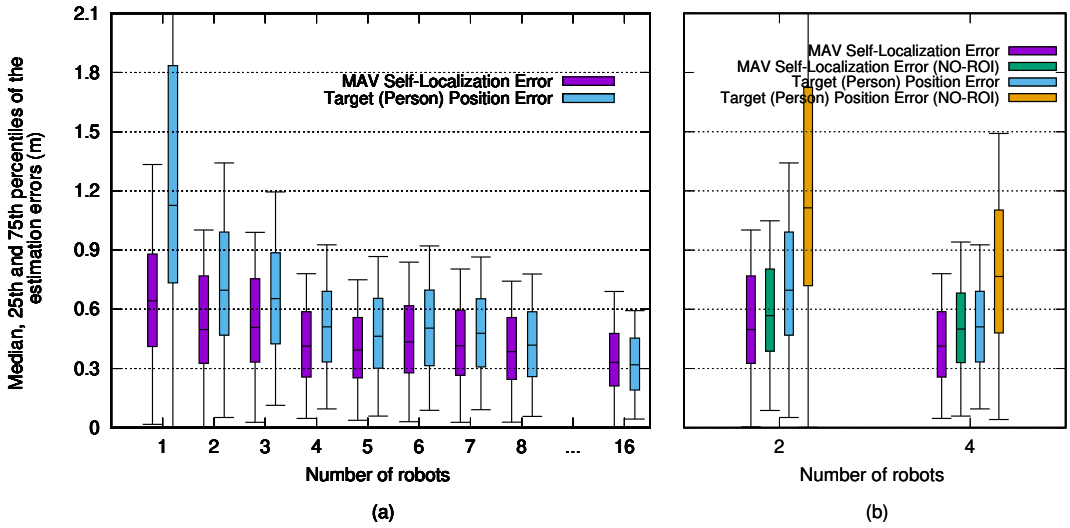


Figure 2.7: Simulation experiment 1 and 2. (a) scalability w.r.t. number of UAVs. (b) contribution of active ROI selection.

2.5.6 Simulation Experiment 1 — Scalability w.r.t. the number of UAVs

The goal of this experiment is to demonstrate the scalability of the approach with the growing number of UAVs. We conducted runs with K= $[1,...,8,16]$ UAVs and perfect networking, i.e., with a detection message loss rate of 0%. Since the behaviour is affected by the random GPS drift behaviour, each run was repeated 9 times and the results averaged over all repetitions and all UAVs in a run.

The Results of this experiment (see Figure 2.7 (a)) show that the largest improvement in both self-pose and target 3D position estimation is achieved when using 2 UAVs, rather than a single UAV. Accuracy continues to improve, up to 16 UAVs in a team. Beyond that, we could not conduct the experiments because the UAVs would violate the safety threshold distance while maintaining the required distance to the person.

2.5.7 Simulation Experiment 2 — Contribution of active ROI selection

The goal of this experiment is to demonstrate that active ROI selection in the image has a significant impact on the target tracking accuracy. Experiments are conducted with 2 and 4 UAVs, 9 times each, in which the region of interest is always the entire camera image. The results are compared to the corresponding runs in simulation experiment 1 (since those already have the active ROI selection enabled).

The results of this experiment (see Figure 2.7 (b)) show an increase of 113% mean error and 60% median error in the target position estimation when the entire image is

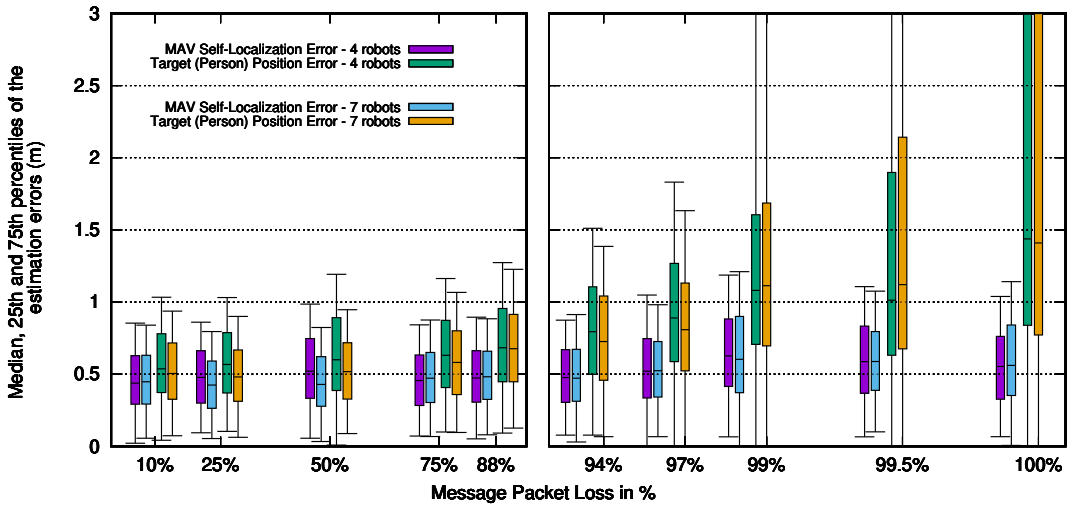


Figure 2.8: Simulation experiment 3 — robustness w.r.t. communication loss.

used instead of an actively selected ROI in the 2 robot case. For the 4-robot case, the increase is 60% and 50% in mean and median errors, respectively. The main reason for such an increase in estimation error is the reduced number of successful person detections if the person's projected size on the image is significantly smaller w.r.t. the processed image size.

2.5.8 Simulation Experiment 3 — Robustness w.r.t. communication loss

The goal of this experiment is to demonstrate the robustness of the approach w.r.t. network packet loss. We conducted runs with K$=[4,7]$ UAVs and loss rates of $[10,25,50,75,88,94,97,99,99.5,100]$%. Each run was repeated 6 times and the results averaged over all repetitions and all UAVs in a run. With a simulated loss rate of 100%, each UAV will only receive its own detections, but not from any other UAVs. UAVs were still receiving position information from the other UAVs to prevent collisions.

The results of this experiment (see Figure 2.8) show that only around 20% of detection measurements need to be exchanged between UAVs to ensure convergence of the entire swarm to a high accuracy estimate (relative to the UAV self-poses). If the loss rate is higher than 80%, the error in both target and self-pose estimates increases exponentially. It is important to note that a network outage in our real-UAV scenario would also prevent the UAVs from receiving their teammates positions. In such a situation, collision avoidance cannot be guaranteed, unless the UAVs are equipped with sensors for teammate detection and tracking that do not rely on radio communication.

2.6 Conclusions and Outlook

In this chapter, we presented a novel method for real-time, continuous DNN-based multi-robot cooperative detection and tracking. Leveraging cooperation between robots in a team, our method can harness the power of deep convolutional neural network-based detectors for real-time applications. Through real robot experiments involving only on-board computation and comparisons with ground truth and baseline approaches, we demonstrated the effectiveness of our proposed method. We also showed the feasibility of real-time person detection and tracking with high precision from a team of UAVs that maintain a perception-driven formation. Additionally, we performed noise quantification of the DNN-based detector that allowed us to use it within a Bayesian filter for person tracking. Through extensive simulation experiments, we verified that our approach is scalable w.r.t. the number of robots, as well as robust to communication failures and gaps.

The work in this chapter forms the foundation, not just for this dissertation, but the majority of the work of the flight robotics and perception group at Max Planck and the Institute for Flight Mechanics and Control at University of Stuttgart (IFR), including optimal model predictive formation control [22], markerless outdoor human motion capture [39], on-board real-time cooperative human motion capture [41], reinforcement-learning based formation control to minimize pose reconstruction error [76] and tracking with non-holonomic aerial vehicles [16].

Still open research goals at this point include heterogeneous teams of UAVs with different sensing and manoeuvring capability, and successful cooperative tracking of non-humans.

Reviewers of this work correctly identified the robustness against communication failures as a critical aspect of this work, which we addressed with additional experiments presented in subsection 2.5.8. At the point of conducting the experiment, we were ourselves surprised how well the method copes with up to 80% packet loss. The underlying principle is the ratio of the rate at which uncertainty in the model grows, reflected in the Bayesian state estimate, versus the rate at which correction information is provided by robot peers and represents the steady state of the Kalman Filter [77]. As bandwidth improves and the rate at which measurements from peers arrive increases, the state uncertainty asymptotically approaches a maximum. However, further increases in bandwidth yield diminishing results past a certain point. However, this additional bandwidth improves resilience to communication failures. The high resilience we measured basically indicates that our actual communication bandwidth is significantly beyond the bandwidth required to maintain the steady state. This resilience is only present if the communication failures are stochastic and each update has an equal chance of "making it through". If communication failures are systematic, for example due to hardware failure of one robot, this robot would receive no information from its peers and its state estimate would degrade to the uncertainty it can achieve based on its own sensors and the model alone.

As mentioned in subsection 2.5.8, communication failures have a more severe effect

on formation control. The MPC expects up-to-date information about the peer robots. Although the MPC predicts the motion of peer robots over the time horizon, it does not model the uncertainty in these estimates. Doing so would require stochastic model predictive control, which has significantly higher computation requirements. Furthermore, a non-communicating team member might exhibit non-stochastic behaviour. The communication failure could be the results of a systematic failure (for example main-computer-crash) in which case the robot would no longer follow the predicted trajectory but maintain position — or in the worst case, fall out of the sky. In either way, collision avoidance can only be guaranteed if the robots know the position of their peers with higher accuracy than the collision avoidance safety margins, a condition that will be violated if communication with a single robot is interrupted for more time than this robot requires crossing the distance of said safety margin.

With multi-copters, a safe fallback behaviour in this case could be to "stop". This is achieved by reducing the absolute velocity to zero and then maintaining the current position until a human pilot intervenes. This, as we will see in future chapters, is not possible for robots that require constant velocity to maintain manoeuvrability.

Both the resilience w.r.t. communication failures and the scalability of this method are only possible because processing of image data and computer-vision is performed on-board. A "detection" within an image is on the order of 12 numeric values, 3 dimensions of position and 3×3 for the uncertainty covariance matrix. With double precision float values (64bit) this requires 96 bytes of data to be transferred for every detection, which happens up to 4 times a second. If we assume 4 object detections per frame, that is in the order of 10 kilobits per second. Compared to $300 \times 300 \times 3$ bytes for just the input dimensionality of the single shot detector, this is a reduction factor of 200. If the UAVs were to transfer raw video from their cameras for off-board processing, even with modern video compression such as H264 or H265, the bandwidth would be exceeding 10 megabits per second per UAV, a factor of 1000 increase. This is a strong argument to conduct computer vision on-board, as this conserves bandwidth and allows more independent operation, as the UAVs only need to communicate with each other and not with a central external node to operate.

2.7 Publication

Work of this chapter has been presented at the NVIDIA GTC 2018 in San Jose. We then added additional experiments with differential GPS ground truth measurement for accurate estimation of tracking accuracy. This work was then submitted to IEEE Robotic Automation Letter. Reviewers suggested additional simulation experiments to verify robustness w.r.t. communication failures, which are found in Section 2.5.5. The chapter was published in IEEE Robotic Automation Letters Vol. 3 Number 4 in 2018 [20] and presented during the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2018 in Madrid, under the title "Deep Neural Network-based Cooperative

Visual Tracking through Multiple Micro Aerial Vehicles"

2.8 Improvements beyond the published work

While the published work in this chapter is still the foundation of our multi-vehicle formation research, certain key improvements have been implemented that extend this capability and are part of the public open-source code, but did not warrant a publication on their own.

2.8.1 False positive suppression based on detection probability

During our initial experiments we occasionally had problems with false positive detections by individual robots, either due to the presence of additional subjects that were not the tracking subject, or due to detector failures. To prevent these from upsetting the entire formation, especially in the presence of correct detections by other robots, we implemented a 5-sigma heuristic. Based on the subject position estimate and the combined uncertainty of the subject position, self pose uncertainty and the detector accuracy, for detection $z_t^{R_k}$ we calculate the probability $P\left(z_t^{R_k}, \bar{x}_t\right)$ assuming a Gaussian probability distribution around $\bar{x}_t$. If this probability is less than a threshold, which we set to a conservative $P \leq \frac{1}{1.7\times10^6}$ or 5σ, then we can safely assume the detection is a false positive and simply discard it. This does not affect the initial search for the subject in uncertain position, nor detections by robots with uncertain self pose estimate, since these uncertainties are already considered.

2.8.2 Interactive subject selection

The published paper does not go into a lot of detail how the subject position is originally initialized. In the original implementation, the position of the subject was assumed at the coordinate system origin, with a sufficiently large uncertainty to have the robots consider any successful detection as valid detection of the subject and then converge on this position. This is unsuited in a situation where multiple subjects are present, and the operator needs to be given a tool to select which subject to track. Since we already used Rviz [78] as the visualization tool to show detection and projected position in 2D camera frames, the straightforward solution was to make this visualization interactive. Parts of this interface can be seen on the left in Figure 2.9.

By clicking on a subject in this displayed 2D camera image, a position is calculated based on the current known camera orientation in the world frame that corresponds to a very elongated Gaussian probability distribution corresponding to the ray-cast of the selected pixel with a configured mean-depth. This manually selected pose estimate is then broadcast via ROS to all robots in the formation, and their EKF are updated with this corresponding probability distribution for the subject pose. This causes the robots to

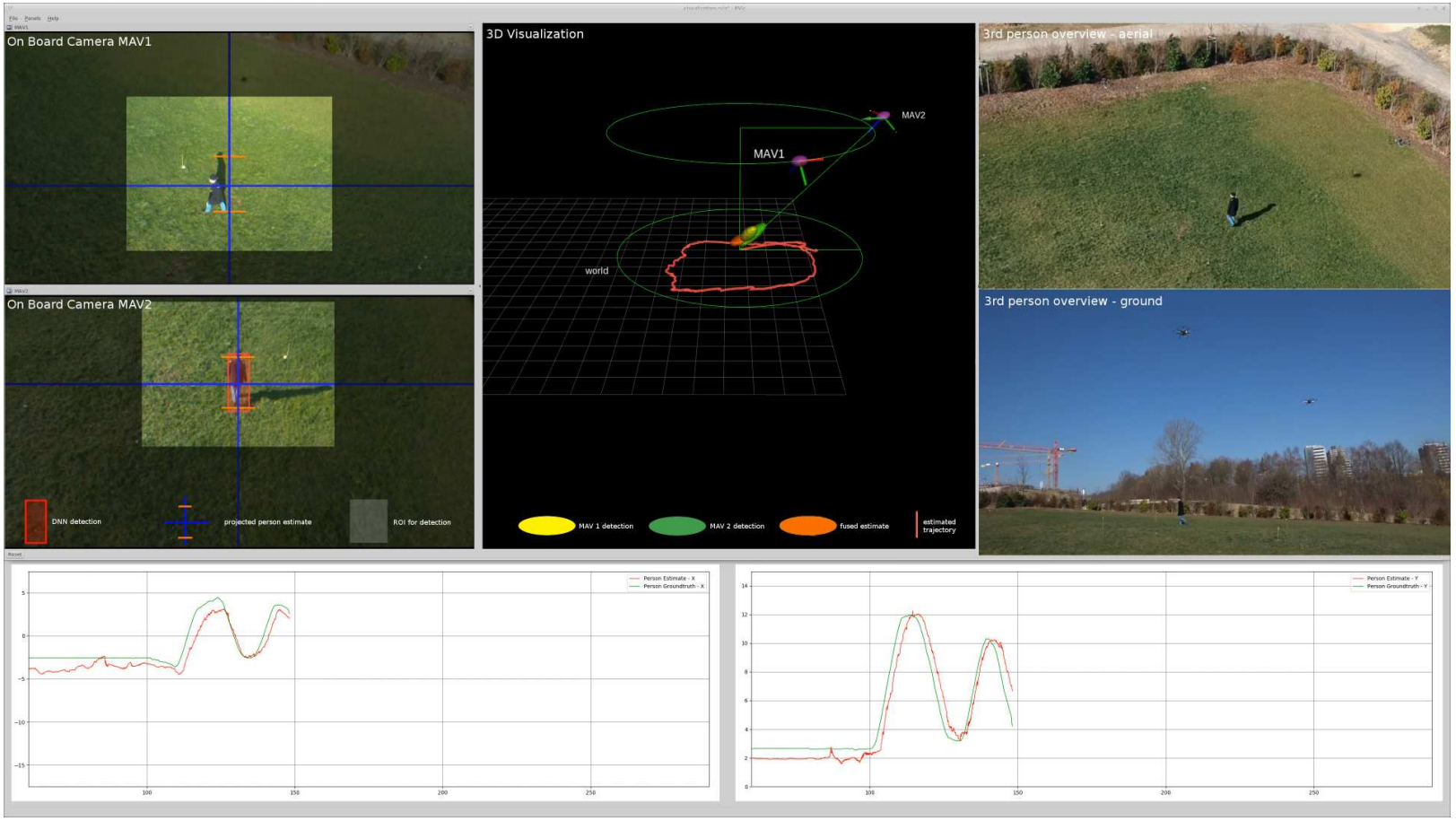


Figure 2.9: The noon experiment, camera visualization, 3D visualization and external cameras.

look for the subject according to the selected distribution, and, if successful detections are made (red rectangle in Figure 2.9) will converge on the selected subject simply by clicking on it.

This approach results in a simple, yet powerful solution to direct an entire formation towards a visible target-subject with a single touch gesture or mouse click.

2.8.3 Using the projection for follow-on analysis

The solution in this chapter forms the basis for several follow-up works. Aside from the adaptation on lighter than air vehicle formations in Chapter 5, the region of interest also was used to provide AirPose [41] with data for real-time human pose estimation.

AirPose works, by estimation of human pose and shape parameters with a distributed neural network in multiple stages. While the distributed network design is beyond the scope of this dissertation, the way it is applied onboard real vehicles is relevant here. At first, the ROI is sent to AirPose, running onboard in the same way it is used as input for SSD-Multibox. The network then provides an output with a crude, single-vehicle estimation, which needs to be shared with the other robot or robots observing the same subject. This data is synchronized and exchanged using ROS, ensuring the next network stage is executed within a pre-defined time window, after all ROS messages have been exchanged. This process is repeated 3 times, after which all robots converge to the same shared pose and shape estimate in SMPL parameters [1] — this is not possible without time synchronization, a region of interest, and a shared communication layer, all of which are provided by the work presented in this chapter.

A second follow-on work is the model predictive formation controller (MPC) by Tallamraju et al. [22]. In that work, the MPC described in Subsection 2.3.4 has been replaced with a significantly more capable controller for the same task, which has since been used in all further multi-copter experiments with AirCap.

Both implementations are open-source and can be found at `https://github.com/robot-perception-group`.

Chapter 3

Annotation of Training Data

This chapter deals with annotation and training of the detector network (Figure 3.1). Initially, we used pre-trained SSD-Multibox [19] on our robots, which was sufficient for human detection in benign environments. However, as we transitioned to animal detection, pretrained network weights proved increasingly unreliable, prompting us to annotate our own training data set.

Several attempts at improving detection were made, including the unfruitful attempts of creating a network capable of "learning to learn" the appearance of the tracked object and the background, as outlined in the Introduction chapter. As part of this work, we also evaluated competing network architectures for single shot detection available at that time. The result of that evaluation (Figure 3.2) was, that standard SSD-Multibox [19] still performed competitively on large objects — compared to the ROI —- although its architecture had known limitations when dealing with small, distant objects. The latter has been addressed in newer architectures by routing high-resolution information to deeper network layers — for example with a feature fusion approach [79] — which obviously comes with a performance penalty. This penalty can be avoided when the size of the to-be-detected object is sufficiently large, which is ensured both by our on-line foveating approach presented in the last chapter, and by a multiscale optical pyramid, presented in this chapter. Both approaches make the detection scale-agnostic.

Therefore, we stuck to SSD-Multibox as our detector of choice. Retraining was first attempted by fine-tuning, but this led to a sudden and drastic reduction in generalization capability of the network, while the performance on the freshly annotated data only increased slowly and marginally. Training on only the small newly annotated dataset also did not yield competitive results, so we settled on re-training the network from scratch on a combined dataset of MS-COCO [24] with our own data added in such a way that our training examples were approximately half the training examples of the class in question and 10% of the overall dataset.

Annotating is however a tedious manual process, which prompted us to pursue automation to assist and speed up this work. Here, we can exploit spatio-temporal correlation of objects in video data along with visual similarity.

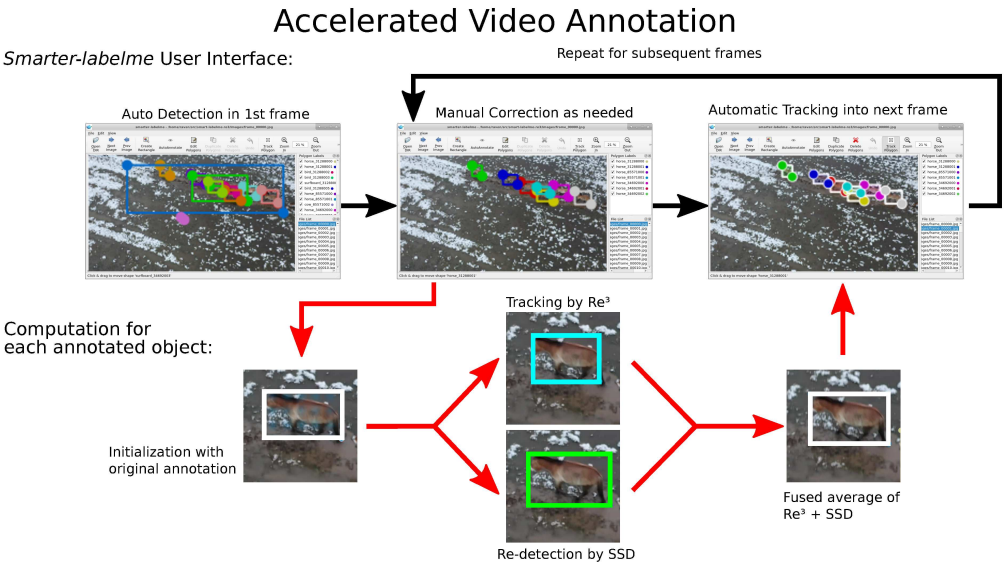


Figure 3.1: Method of *Smarter-labelme* using SSD-Multibox and Re^3 to auto-annotate and track objects. In this example, moving Przevalski's horses.

3.1 Introduction

Object annotation in videos is critical to generating ground-truth data for various machine-learning endeavours [84]. Reliable ground-truth object annotation requires the human annotator to solve two main tasks per video frame. The first involves detecting new relevant object instances in each frame, and the second entails tracking previously identified object instances in subsequent frames. An effective tool in assisting annotators with these tasks is a Graphical User Interface (GUI), which presents relevant information and allows users to add and modify annotations interactively. At present, various video annotation tools packaged as GUIs are available, providing a range of capabilities from manual to semi-automatic annotation [85].

Manual annotation tools are often lightweight, easy to use, and can provide web-based interfaces that minimize installation roadblocks while facilitating easy adoption [86, 87]. However, manual annotation approaches are challenging to scale, especially for large video datasets, and can quickly lead to high annotation costs. Tools with semi-automatic annotation capabilities provide an attractive alternative, where the object of interest can be pre-annotated using visual detection on the fly or as a preprocessing step [85, 88]. Such an approach has been widely employed in proprietary and open-source tools; however, for semi-automatic approaches to result in real-world gains (reduced annotation cost) they must reliably track the identified object instances across video frames. Failure to do so would lead to the annotator spending more time and effort manually correcting, re-annotating, and, if necessary, re-identifying object instances across frames.

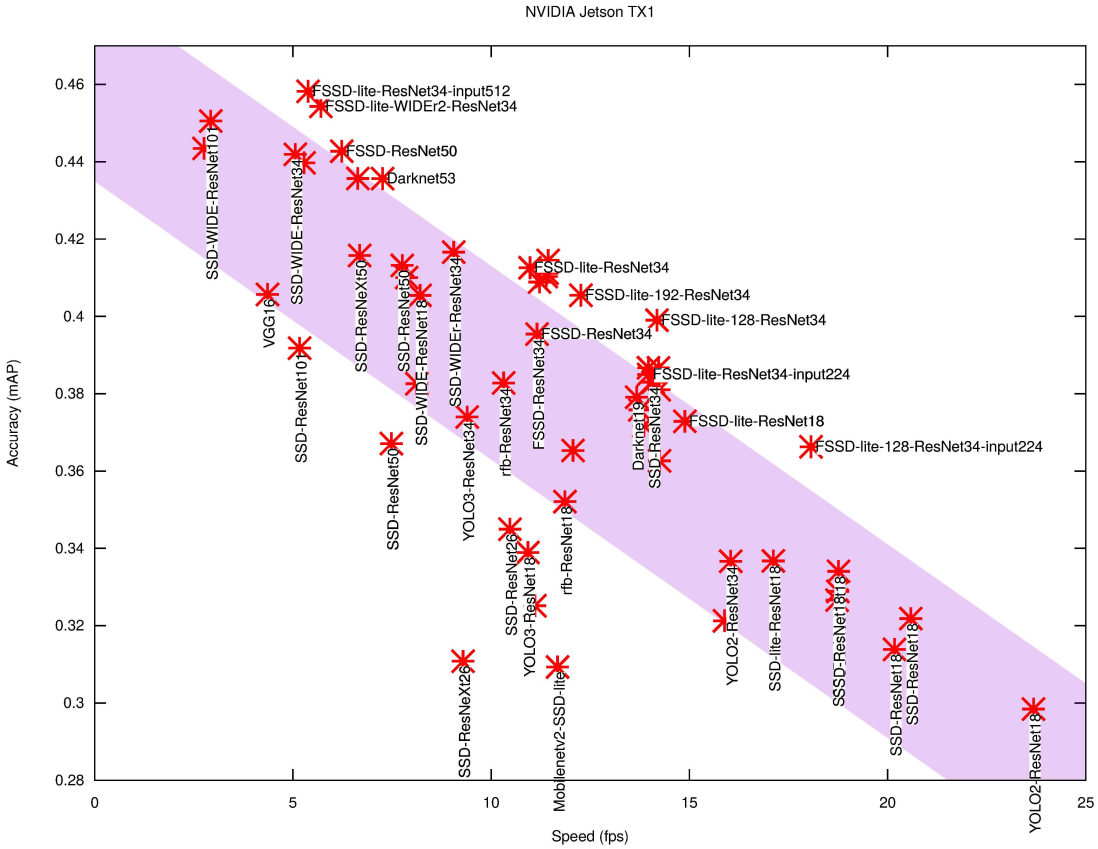


Figure 3.2: Comparison of Single Shot Detector network architectures trained and evaluated on MSCOCO [24]. There is a general tradeoff between mean average precision (mAP) and detection speed. Most of the tested networks lie on the same envelope. A detector can be made more accurate and slower, or faster and less accurate either by changing the feature extraction architecture (ResNet variants [80, 81], darknet [82], VGG [80]) the actual detection backend (YOLO2, YOLO3, SSD [19, 83]) or the input layer size (image size). Neither modification offers significant advantages, that is, comparable desired mAP and speed can be achieved with any network architecture by only changing one of these parameters, e.g. input layer size. In our test, only lightweight feature fusion architectures (FSSD and FSSD-lite) stood out slightly, but this is primarily due to their improved detection of very small objects.

At present, widely used semi-automatic video annotation tools such as *Smart-labelme* [89], Label Studio [90], and CVAT [91] employ both classic and deep tracking methods to track already annotated objects across frames, typically in the form of plug-ins. Classical tracking methods such as AdaBoost or Kernelized Correlation Filters, as well as deep methods such as Siamese- and transformer-based trackers can often suffer from "drift" as the tracked object changes appearance, for example due to rotation. What the tracker considers a part of the object or not might then change in every frame, e.g., a tracker initialized to track a person as a whole might drift, eventually tracking only their head or a similar visually easy-to-distinguish part. Another approach involves tracking the object instances using linear interpolation between key frames, which can fail if the object changes its shape or direction of motion between key frames. Overall, addressing these tracking issues is essential to reduce the annotation cost further and facilitate the generation of large ground-truth datasets, which are a key requirement to train generalized models for machine learning tasks.

Here, we present a new annotation approach, *Smarter-labelme*, that combines a deep learning-based tracker with a deep learning-based detector that can compensate for drift and changes in object motion/shape/size in real-world scenarios. We furthermore aid the tracker by initialising its search area by estimating the global camera motion. Rotation of the camera causes all objects in the image to shift between consecutive frames. For distant small objects, this shift can be several multiples of the object size, which is a significant challenge for most visual trackers that have a limited search area around the object. While such techniques have been used for visual tracking methods, to the best of our knowledge, they have not been introduced as part of an open-source video annotation tool. We test our tool on groups of various animate and inanimate object classes in the real world. Our results strongly suggest that *Smarter-labelme* can increase the annotation speed while keeping the processing overhead low, increase tracking accuracy by accounting for global camera motion, and maintain annotation accuracy over multiple consecutive frames. Altogether, these results demonstrate the effectiveness and versatility of our tool in lowering annotation costs to expedite the generation of reliable ground-truth datasets for various object detection and tracking tasks. The main novel contributions in this chapter are the extension of a video annotation program such that it

- pre-annotates objects in video frames based on the already annotated labels in the previous frame using on a recurrent state-of-the-art object tracker that remembers and updates the object's appearance
- uses a single shot object detector to re-detect the object and — with that — correct the drift of the recurrent tracker using class-agnostic object detection/localisation
- uses global camera motion estimation to initialize the tracker's search area, which allows tracking of small distant objects despite significant camera rotation

in a single tool and unified semi-automated workflow. The contribution of each of these methods to the overall accuracy of the automated tracking is evaluated in an ablation

study. An evaluation dataset and code to reproduce the automated evaluation results are provided [25]. The tool itself is open-source and available for the benefit of the community [92].

In the latest version, and to facilitate a more streamlined operation, Smarter-labelme can track fully automatically at the fastest computation speed the hardware allows, controlled by a start-stop button. In this mode, the program can use already annotated key frames to re-initialize the tracker, which allows fully automated annotation of videos that have previously been manually annotated at a lower frame rate. This is also supported in a "headless" mode, in which the tool is evoked from the command-line without a GUI and exits when the video is fully annotated. We employ this mode for the accuracy evaluation presented in Subsection 3.4.2.

3.2 State of the Art

Our tool is an extension of *Smart-labelme*, an open-source video annotation tool based on *labelme* [93], that can automatically track objects using a traditional computer vision technique provided by OpenCV [88, 94]. We enhance the capabilities of *Smart-labelme* by incorporating deep learning-based methods for both object detection and tracking. Previous work has demonstrated the effectiveness of deep convolutional networks for visual object detection [18, 95]. In our approach, bounding box detection of objects is performed using SSD-Multibox [19], known for its favourable speed-accuracy trade-off. While SSD-Multibox's performance diminishes on very small objects, we mitigate this by optimizing the scaling of detection regions for tracking, inspired by [20] (Chapter 2) and within a multiscale sliding window object detection approach as demonstrated in [96]. Furthermore, instead of parallelizing the detection on specialized hardware, we run the detection neural network sequentially on image regions and only pool the bounding boxes. Since we use a fast detector network, execution speed is limited by memory throughput. Therefore, sequential execution only slightly increases runtime but drastically reduces memory usage, allowing the algorithm to run efficiently on both GPU and CPU.

The SSD-Multibox detector used in our tool is pre-trained on the MSCOCO dataset [24] that contains a diverse set of common object classes. This allows us to address the challenge of detecting new, previously non-annotated objects. The initial detection in our tool is facilitated by applying the SSD-Multibox detector in a hierarchical overlapping sliding window technique with global, exclusive, non-maximum suppression. This technique avoids re-detecting already-annotated object instances, truncated objects or multiple detections per object. The detected instances are assigned a globally unique ID consisting of the detected class name and a unique random numeric identifier.

For tracking the identified object instances (bounding boxes), we surveyed previous work on visual object tracking in videos [97, 98] and opted for a fast, light-weight Re^3 [26] network that utilizes a recurrent Siamese approach. Moreover, we assist the

tracking of object instances by first accounting for the global camera motion between consecutive frames, making tracking of object instances more robust to changes in camera movement during video recording. We calculate the shift of the entire image in pixel-space and apply that to the search area for each bounding box. The resulting search area is used by Re^3 to pre-calculate probable bounding box annotations in the previous or next untracked frame. At this point, the SSD-Multibox detector is employed again to correct the predicted Re^3 bounding box for drift if detections with sufficiently high confidence and Jaccard overlap are found, as shown in Figure 3.1. Finally, the annotations are stored in per-video-frame JSON files with consistent instance labels over time. This allows post-processing of annotations, including fine-tuning of the networks that use the annotated data.

The likely most popular competing tool, CVAT [91], employs a modular approach, which also allows integration of state-of-the-art detector methods for object detection such as YOLOv8 [83] and also the latest state-of-the-art trackers such as SiamMask [99] and Transformer Tracking [100]. This modularity, however, prevents the combined application of these methods. CVAT currently does not employ a detector for tracker drift correction. Similarly, CVAT does not initialize the tracker search area based on larger scale motions (global camera motion estimation)

3.3 Methodology

Subsections 3.3.1-3.3.4 cover notation and the previous state-of-the-art, upon which this chapter is based. Subsections 3.3.5-3.3.8 cover our novel contributions.

3.3.1 Notation

A Video is a sequence of F image frames $\boldsymbol{i}_f$. We write

$$\text{Video} = \left(\boldsymbol{i}_f\right)_{0 \le f < F}. \tag{3.1}$$

The Video is to be annotated with Annotations

$$\text{Annotations} = \left(\hat{\boldsymbol{a}}_f\right)_{0 \le f < F}. \tag{3.2}$$

Each image frame i_f is an array of pixels with width X, height Y holding a 3 dimensional pixel vector $\boldsymbol{p} = [p_R\ p_G\ p_B]^\top$ with pixel brightness values for Red, Green, and Blue (RGB)

$$\boldsymbol{i}_f = \left(\boldsymbol{p}_{x,y}\right)_{0 \le x < X, 0 \le y < Y} \tag{3.3}$$

where $[x = 0\ y = 0]^\top$ is the upper-left corner.

Each annotation vector $\hat{\boldsymbol{a}}_f$ at frame f consists of A_f annotations

$$\hat{\boldsymbol{a}}_f = \left(\boldsymbol{a}_{a,f}\right)_{0 \leq a < A_f}. \tag{3.4}$$

Each annotation $\boldsymbol{a}_{a,f}$ includes a label $\boldsymbol{l}_a$ and location vector $\boldsymbol{b}_{a,f}$

$$\boldsymbol{a}_{a,f} = \left\{\boldsymbol{l}_a, \boldsymbol{b}_{a,f}\right\}. \tag{3.5}$$

While *labelme* [93] supports polygonal annotations for semantic segmentation, both *Smart-labelme* [89] and our solution track instances by their bounding box. Polygonal annotations are scaled in the x- and y-directions based on the bounding box size and shape. Therefore, each $\boldsymbol{b}$ is a vector with

$$\boldsymbol{b} = \left[x_b \; y_b \; X_b \; Y_b\right]^\top \tag{3.6}$$

where $\left[x_b \; y_b\right]^\top$ encodes the upper-left corner and $\left[X_b \; Y_b\right]^\top$ encodes the width and height of the rectangular bounding box in pixels.

The label $\boldsymbol{l}$ is a textual representation, which can be used to store information about the annotated object instance, such as its type, identity, etc.

A crop $\boldsymbol{C}()$ is a sub-image that is a part of a larger image $\boldsymbol{i}$. We define $\boldsymbol{C}()$

$$\boldsymbol{C}(\boldsymbol{i}, \boldsymbol{b}) = (\hat{\boldsymbol{c}}_{x_c, y_c}) = \left(\boldsymbol{p}_{(x_c + x_b), (y_c + y_b)}\right)_{0 \leq x_c < X_b, 0 \leq y_c < Y_b} \tag{3.7}$$

as the tensor that maps the sub-matrix of a frame $\boldsymbol{i}$ defined by the bounding box coordinates of $\boldsymbol{b}$. While $\boldsymbol{C}()$ is only defined within the dimensions of $\boldsymbol{i}$, we define $\boldsymbol{C_0}()$ such that pixels outside the dimensions of $\boldsymbol{i}$ are simply considered black ($p_R = p_G = p_B = 0$).

Similarly, a scale $\boldsymbol{S}()$ is an image similar to another image $\boldsymbol{i}$, which has its resolution changed to $\left[X_s \; Y_s\right]^\top$. This can be defined by a remapping of pixel coordinates with a tensor $\boldsymbol{S}()$

$$\boldsymbol{S}\left(\boldsymbol{i}, \left[X \; Y\right]^\top, \left[X_s \; Y_s\right]^\top\right) = (\hat{\boldsymbol{s}}_{x_s, y_s}) = \left(\boldsymbol{p}_{(\lfloor x_s \frac{X}{X_s} \rfloor), (\lfloor y_s \frac{Y}{Y_s} \rfloor)}\right)_{0 \leq x_s < X_s, 0 \leq y_s < Y_s}, \tag{3.8}$$

where $\lfloor \; \rfloor$ rounds to the smaller integer. For both tensors, there are matching functions $c(), s()$ and their inverse $c^{-1}(), s^{-1}()$ to map pixel coordinates $\left[x \; y\right]^\top$ or a bounding box $\boldsymbol{b}_0 = \left[x_{b0} \; y_{b0} \; X_{b0} \; Y_{b0}\right]^\top$ between the image and the crop/scale with

$$\left[x_c \; y_c\right]^\top = c\left(\left[x \; y\right]^\top, \boldsymbol{b}\right) = \left[(x - x_b) \; (y - y_b)\right]^\top, \tag{3.9}$$

$$\boldsymbol{b}_c = c(\boldsymbol{b}_0, \boldsymbol{b}) = \left[(x_{b0} - x_b) \; (y_{b0} - y_b) \; X_{b_0} \; Y_{b_0}\right]^\top, \tag{3.10}$$

$$\left[x_s \; y_s\right]^\top = s\left(\left[x \; y\right]^\top, \left[X \; Y\right]^\top, \left[X_s \; X_s\right]^\top\right) = \left[\left(x \frac{X_s}{X}\right) \; \left(y \frac{Y_s}{Y}\right)\right]^\top, \tag{3.11}$$

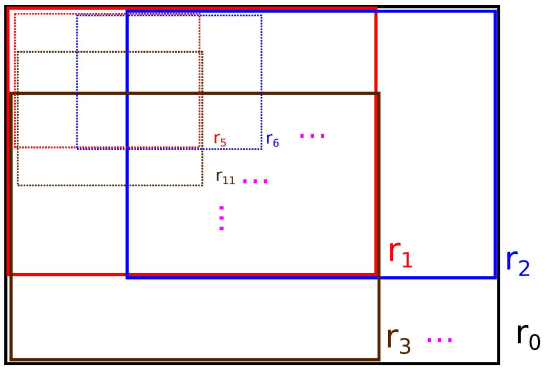


Figure 3.3: Tiling an image into an array of overlapping tiles at different scales.

$$\boldsymbol{b}_s = s\left(\boldsymbol{b}_0, [X\ Y]^\top, [X_s\ X_s]^\top\right) = \left[\left(x_{b0}\frac{X_s}{X}\right)\ \left(y_{b0}\frac{Y_s}{Y}\right)\ \left(X_{b0}\frac{X_s}{X}\right)\ \left(Y_{b0}\frac{Y_s}{Y}\right)\right]^\top. \quad (3.12)$$

Note: In practice, for $\boldsymbol{S}()$, we use interpolating methods implemented in both OpenCV and PyTorch to avoid aliasing effects.

3.3.2 Multi Scale Tiling Approach

A common approach in computer vision is to divide the image $\boldsymbol{i}$ systematically into overlapping tiles of different size. This is used among others by [19, 96] and also the majority of Deep Convolutional Neural Networks (CNN).

A visual operation, for example feature or object detection, can then be executed in parallel on all these tiles, across the image at both large and small scale. The approach is also called an "Optical Pyramid" and can be considered a manifestation of the "divide and conquer" paradigm often found in computer science. If a tile $\boldsymbol{r}$ has size $[X_r\ Y_r]^\top$, then $\boldsymbol{r_0}$ would typically represent the whole image with

$$\boldsymbol{r_0} = \boldsymbol{S}\left(\boldsymbol{i}, [X\ Y]^\top, [X_r\ Y_r]^\top\right). \quad (3.13)$$

All other tiles are based on a cropped tile area $\boldsymbol{b}_n = [x_{b_n}\ y_{b_n}\ X_{b_n}\ Y_{b_n}]^\top$, with

$$\boldsymbol{r_n} = \boldsymbol{S}\left(\boldsymbol{C}(\boldsymbol{i}, \boldsymbol{b}_n), [X_{b_n}\ Y_{b_n}]^\top, [X_r\ Y_r]^\top\right). \quad (3.14)$$

The tiling, shown in Figure 3.3, is typically done with a recursive algorithm that calculates all $\boldsymbol{b}_n$ of different sizes to cover the whole image space. This can be either overlapping or non-overlapping, depending on the use case. Typically, every layer of tiles has approximately 4 times the tiles as the previous at half the tile dimensions, while the number of layers depends on the resolution of the original image and the minimum tile

size. The latter is often identical to $[X_r\ Y_r]^\top$ and yields an array of R tile boxes $\boldsymbol{T}$ with

$$\boldsymbol{T} = (\boldsymbol{b}_r)_{0 \le r < R} = \text{Tile}\left([X\ Y]^\top, [X_r\ Y_r]^\top\right). \tag{3.15}$$

3.3.3 Manual Annotation

Labelme [93] is optimized to streamline the process for human annotators. One video frame $\boldsymbol{i}_f$ is always displayed prominently on the screen. The user can navigate between frames with the arrow keys on the keyboard and in the spatial dimension with the keyboard + scroll wheel, which allows very fast navigation. Similarly, with the Ctrl key + scroll wheel the user can zoom in or out to magnify areas in high-resolution images, while the Alt key or no key is used to pan horizontally or vertically. Navigating to a specific f can be done quickly by mouse-clicking on a timeline that highlights all frames already annotated. The user can add annotations by entering the appropriate mode (Ctrl+R) and then clicking diagonally opposite corners of a rectangle. A prompt box will appear where the user can select or type a label for the new annotation. The previous annotation is pre-selected, which is very fast if many identical labels are given. However, modification of the label is necessary if every instance is supposed to receive distinct identifiers. For the latter case, an experienced and well-rested annotator can achieve a speed of under 5 seconds per annotation. For identical labels, a speed of under 4 seconds is possible. These values are approximate, evaluated in self-test, and can increase drastically depending on the visual difficulty of the individual annotation task.

3.3.4 Assisted Tracking

Smart-labelme [89] adds the ability to track objects between subsequent frames using a method for parametric image alignment [88], which is implemented in the OpenCV library [94]. Using this to track objects works if these are visually distinctive and do not change appearance drastically. The method is not well suited to tracking organic structures that undergo morphological changes, such as a bird flapping its wings or a horse moving its legs while walking. The process involves selecting one or more existing annotations and then clicking “track-polygon”. The user then navigates to a different video frame that does not have an annotation with the same labels, where the tracking algorithm is applied to find the corresponding bounding box in the new frame and create new object annotations. Even if the tracking is imperfect, the human annotator only needs to adjust bounding box corners for existing annotations, with no need to type. The algorithm runs sequentially on the CPU and is relatively slow. The annotations in the first video frame $\hat{\boldsymbol{a}}_0$ must be made manually, as described in Subsection 3.3.3.

3.3.5 Assisted Tracking with Re^3

Re^3 [26] is a recurrent Siamese network, utilizing Long-term-Short-term memory (LSTM) nodes to remember the appearance of tracked object instances, while simultaneously observing two cropped images. $\boldsymbol{C}\left(i_f, \tilde{\boldsymbol{b}}_{a,f}\right), \boldsymbol{C}\left(i_{f+1}, \tilde{\boldsymbol{b}}_{a,f}\right)$. Since Re^3 has internal memory, for each annotation $\boldsymbol{a}_{a,f}$ we also need to store internal memory $\boldsymbol{R}_{a,f}$, which will be updated by the tracker in every tracking transition. We can represent Re^3 as a function, based on a crop region $\tilde{\boldsymbol{b}}_{a,f}$, as

$$\left\{\hat{\boldsymbol{b}}_{a,f+1}, \boldsymbol{R}_{a,f+1}\right\} = \text{Re}^{\mathbf{3}}\left(\boldsymbol{C_0}\left(\boldsymbol{i}_f, \tilde{\boldsymbol{b}}_{a,f}\right), \boldsymbol{C_0}\left(\boldsymbol{i}_{f+1}, \tilde{\boldsymbol{b}}_{a,f}\right), \boldsymbol{R}_{a,f}\right). \tag{3.16}$$

Re^3 provides the new coordinates $\hat{\boldsymbol{b}}_{a,f+1}$ for the tracked objects in the new cropped region corresponding to an object that occupies the exact centre in the old cropped region. For any annotation $\boldsymbol{a}_{a,f} = \left\{\boldsymbol{l}_a, \boldsymbol{b}_{a,f}\right\}$ and $\boldsymbol{b}_{a,f} = [x_b\ y_b\ X_b\ Y_b]^\top$ the crop region is always twice as large as the to-be-tracked object instance, with

$$\tilde{\boldsymbol{b}}_{a,f} = \left[x_{\tilde{b}}\ y_{\tilde{b}}\ X_{\tilde{b}}\ Y_{\tilde{b}}\right] = \left[\left(x_b - \left\lfloor\frac{X_b}{2}\right\rfloor\right)\ \left(y_b - \left\lfloor\frac{Y_b}{2}\right\rfloor\right)\ 2X_b\ 2Y_b\right]^\top, \tag{3.17}$$

which leads to annotation $\boldsymbol{a}_{a,f+1}$ based on $\hat{\boldsymbol{b}}_{a,f+1} = \left[x_{\hat{b}}\ y_{\hat{b}}\ X_{\hat{b}}\ Y_{\hat{b}}\right]^\top$ and $\tilde{\boldsymbol{b}}_{a,f}$

$$\boldsymbol{a}_{a,f+1} = \left\{\left[\left(x_{\hat{b}} + x_{\tilde{b}}\right)\ \left(y_{\hat{b}} + y_{\tilde{b}}\right)\ X_{\hat{b}}\ Y_{\hat{b}}\right]^\top, \boldsymbol{l}_a\right\}. \tag{3.18}$$

If $\boldsymbol{R}_{a,f}$ is not known, we first initialize the tracker with

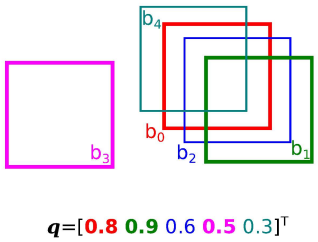
$$\left\{\ldots, \boldsymbol{R}_{a,f}\right\} = \text{Re}^{\mathbf{3}}\left(\boldsymbol{C_0}\left(\boldsymbol{i}_f, \tilde{\boldsymbol{b}}_{a,f}\right), \boldsymbol{C_0}\left(\boldsymbol{i}_f, \tilde{\boldsymbol{b}}_{a,f}\right), \boldsymbol{0}\right). \tag{3.19}$$

This allows tracking equivalent to *Smart-labelme* but using a modern, fast, recurrent deep convolutional method. Re^3 suffers from drift, but drift only manifests itself after some time, at which point the user might have to adjust the bounding box manually to correct for the error.

3.3.6 Assisted Detection with SSD-Multibox

SSD-Multibox is a Deep Convolutional Neural Network (CNN) for object detection. It operates with a fixed input size $[X_r\ Y_r]^\top$ of 300px $\times$ 300px. This input image is divided using tiling function Tile() (3.15) into $T_{\text{SSD}} = 8732$ overlapping receptive fields, called prior boxes. For each such box the network outputs $\boldsymbol{o}_n$ with $\boldsymbol{b}_n = [x_{b_n}\ y_{b_n}\ X_{b_n}\ Y_{b_n}\ ]^\top$ for a potential object instance candidate, as well as K confidence scores $\boldsymbol{k}_n$ used for classification into K object classes, so

$$(\boldsymbol{o}_n)_{0 \le n < T_{\text{SSD}}} = \text{SSD}(\boldsymbol{i}) \tag{3.20}$$



$\boldsymbol{D}$:

	$\boldsymbol{b}_0$	$\boldsymbol{b}_1$	$\boldsymbol{b}_2$	$\boldsymbol{b}_3$	$\boldsymbol{b}_4$
$\boldsymbol{b}_0$	1	0	1	0	1
$\boldsymbol{b}_1$	0	1	1	0	0
$\boldsymbol{b}_2$	1	1	1	0	0
$\boldsymbol{b}_3$	0	0	0	1	0
$\boldsymbol{b}_4$	1	0	0	0	1

$\hat{\boldsymbol{D}}$:

	q_0	q_1	q_2	q_3	q_4
$\boldsymbol{q_0}$	**0.8**	0	0.6	0	0.3
$\boldsymbol{q_1}$	0	**0.9**	0.6	0	0
q_2	0.8	**0.9**	0.6	0	0
$\boldsymbol{q_3}$	0	0	0	**0.5**	0
q_4	**0.8**	0	0	0	0.3

Figure 3.4: Matrices $\boldsymbol{D}$ and $\hat{\boldsymbol{D}}$ for example detections $\boldsymbol{b}$ with confidence scores $\boldsymbol{q} = [0.8\ 0.9\ 0.6\ 0.5\ 0.3]^\top$. NMS is performed by selecting only those b_n for which q_n (underlined) equals the row-maximum (**bold**).

with

$$\boldsymbol{o}_n = \{\boldsymbol{b}_n, \boldsymbol{k}_n\}. \tag{3.21}$$

Typically non-maximum suppression (NMS) is applied to only report the most confident of any overlapping array of detections for each class. However, we defer this step. When analysing high-resolution video, it is impractical to downscale each frame to the network resolution. Instead, we apply Tile() to divide each image frame $\boldsymbol{i}_f$ into D overlapping tiles $\boldsymbol{b}_d = \left[x_{b_d}\ y_{b_d}\ X_{b_d}\ Y_{b_d}\right]^\top$, then run SSD on each so

$$\boldsymbol{i}_{f,d} = \boldsymbol{S}\left(\boldsymbol{C}\left(\boldsymbol{i}_f, \boldsymbol{b}_d\right), \left[X_{b_d}\ Y_{b_d}\right]^\top, \left[X_r\ Y_r\right]^\top\right) \forall \boldsymbol{b}_d \in \text{Tile}\left(\left[X\ Y\right]^\top, \left[X_r\ Y_r\right]^\top\right). \tag{3.22}$$

In each array of detections SSD $\left(\boldsymbol{i}_{f,d}\right)$, we discard any detections that are close to the edges of $\boldsymbol{i}_{f,d}$, since these detections are likely truncated. Since tiles are overlapping at different scales, there exists another tile that covers the whole object instance. We apply $c^{-1}()$ and $s^{-1}()$ to transform the remaining bounding boxes into the coordinate frame of $\boldsymbol{i}_f$ to compute $\hat{\boldsymbol{o}}_f$ with

$$\hat{\boldsymbol{o}}_f = \bigcup_{0 \le d < D} c^{-1}\left(s^{-1}\left(\text{remove\_truncated}\left(\text{SSD}\left(\boldsymbol{i}_{f,d}\right)\right)\right)\right). \tag{3.23}$$

Non-Maximum Suppression

Storing $\hat{\boldsymbol{o}}_f$ in a PyTorch tensor, we loop through each of the K object classes. For each class, we sort $\hat{\boldsymbol{o}}_f$ by confidence score and truncate the $L = 5000$ highest confidence detections. This puts an upper bound on the computational effort. We then have L bounding boxes $\boldsymbol{b} = (\boldsymbol{b}_l)_{0\leq l<L}$ and L confidence scores $\boldsymbol{q} = (q_l)_{0\leq l<L}$. We calculate $\boldsymbol{J} = (j)_{m,n}$, which is an $L \times L$ matrix of the intersection over union (IOU) of these bounding boxes. We derive $\boldsymbol{D}$ as a boolean $L \times L$ matrix with

$$\boldsymbol{D} = \left(d_{m,n} = \begin{cases} 1 \iff j_{m,n} > 0.5 \\ 0 \iff j_{m,n} \leq 0.5 \end{cases} \right)_{0\leq m<L, 0\leq n<L}, \tag{3.24}$$

from which we calculate $\hat{\boldsymbol{D}}$ by per column multiplication with the vector $\boldsymbol{q}$. Therefore,

$$\hat{\boldsymbol{D}} = \left(\hat{d}_{m,n} = q_n d_{m,n}\right)_{0\leq m<L, 0\leq n<L}, \tag{3.25}$$

as illustrated in Figure 3.4. Maxima are those $\{\boldsymbol{b}_l \; q_l\}$ for which the maximum of the lth row-vector of $\hat{\boldsymbol{D}}$ is identical to q_l itself, i.e. the row-maximum is on the diagonal of $\hat{\boldsymbol{D}}$.

Crucially, this entire algorithm can be expressed in PyTorch tensor operations and computed in a parallel on GPU with constant computational runtime. This is substantially faster than typical iterative looping algorithms for NMS, especially when implemented in Python.

For automatic annotation, we extend NMS by also excluding any detections that have an IOU ≥ 0.5 with existing $\boldsymbol{a}_{a,f} \in \hat{\boldsymbol{a}}_f$. We add any remaining detections, computing a new $\boldsymbol{l}_a$ using the SSD class k and a unique number computed from the system time and a counter.

3.3.7 Drift-compensation with SSD-Multibox and Re^3

For each Re^3 invocation, we run SSD on each $\tilde{\boldsymbol{b}}_{a,f}$. If a detection is found ((3.21)) that has an IOU with $\hat{\boldsymbol{b}}_{a,f+1}$ as computed by Re^3 ((3.16)) of 0.8 or higher, regardless of class, then we assume SSD found the object instance. We then correct $\boldsymbol{b}_{a,f+1}$ using a weighted average of both SSD and Re^3 prediction and copy the label to compute $\hat{\boldsymbol{a}}_{f+1}$ as also shown in Figure 3.1.

3.3.8 Global Camera Motion

In many real-world situations, the recorded video is coupled with camera motion, which can lead to significant jumps in the pixel position of the object of interest. We address the camera motion-induced problem by first identifying a transformation matrix for the whole image, which describes the shift of the entire image in pixel-space. This correction matrix is then applied to the coordinates of the search area of each object before

employing the Re^3 tracker. To identify this transformation matrix, we use a parametric image alignment algorithm [88] on the down-sampled (to 100px $\times$ 100px) versions of the previous and the current image frames, $\boldsymbol{i}_{f-1}$ and $\boldsymbol{i}_f$, respectively. This provides two benefits. First, although not GPU accelerated, the method is sufficiently fast on small images and needs to be executed only once for each pair of frames. Second, the influence of small moving objects in the scene on the algorithm is minimized by down-sampling, while the global image motion is maintained.

3.4 Experiments and Results

We perform two sets of experiments to test the effectiveness of *Smarter-labelme*; the first evaluates the annotation speed (annotations per minute), and the second quantifies annotation tracking accuracy. We also test *Smarter-labelme*'s versatility by performing unsupervised detection and tracking on the GMOT-40 dataset [101] which contains ground-truth annotations for multi-object tracking of 40 different object classes.

3.4.1 Annotation Speed Evaluation

The goal of this experiment is to evaluate the annotation speed with increasing degrees of machine assistance. We use three tool variants: *labelme*, *Smart-labelme*, and *Smarter-labelme* that corresponded to no tracking assistance, OpenCV tracking, and our tracking approach.

Method Variants

Baseline. The baseline variant does not aid the annotator with any computer vision technique. Annotations $\hat{\boldsymbol{a}}_0$ need to be annotated by hand. In the original *labelme* [93], no tracking variant exists, and labels need to be manually copied to the next frame. However, this is needless overhead and would not be suitable to evaluate the impact of machine accelerated tracking. Therefore, we implement a pseudo-tracker that copies the current annotations to the next frame with $\hat{\boldsymbol{a}}_{f+1} = \hat{\boldsymbol{a}}_f$ when engaged. This keeps the workflow identical to the variants with a real tracker. However, the user needs to then manually correct each bounding box in all subsequent frames, unless the object instance has remained stationary.

OpenCV. The OpenCV variant is *Smart-labelme* [89] which, as described in Section 3.3.4. It uses parametric image alignment to track object instances.

Ours. Our variant, *Smarter-labelme*, uses Re^3 with SSD-Multibox drift compensation as described in Section 3.3.7 to track. The Auto-Annotation function described in Section 3.3.6 is available to the annotator to help annotate $\hat{\boldsymbol{a}}_0$ at the annotator's discretion.

Experiment Setup

We set up a computer with the method variants and allowed 5 annotators to perform a video annotation task with each. For a fair comparison, all experiments are conducted on the same computer equipped with an AMD Ryzen Threadripper 3960X 24-Core CPU, an NVIDIA RTX 2080Ti GPU, and a 4K monitor. The annotated video, shown in Figure 3.5 is a short 4K aerial clip of 8 horses walking on a snowy plane, shot using one of our aerial robots at the Hortobágy National Park in Hungary. Each annotator is provided with an introduction and demonstration of the labelling tool, followed by a 5-minute practice period to familiarize themselves with the tool and the annotation process. Then all annotations are reset and the program restarted. Each annotator is given 10 minutes to annotate the video for each of the 3 variants (i.e., 30 min in total). After ten minutes, the variant is switched and all annotations reset. The order in which variants are evaluated was randomized to minimize bias introduced by exhaustion or increased proficiency. The annotators are instructed to annotate with consistent accuracy throughout the experiments. The number of successfully annotated object instances are counted for each method variant and annotator. Since the purpose of human annotation is to provide "ground-truth", no ground truth data is available to evaluate human annotation accuracy. Therefore, for this experiment, the metric for evaluation is the annotation speed.

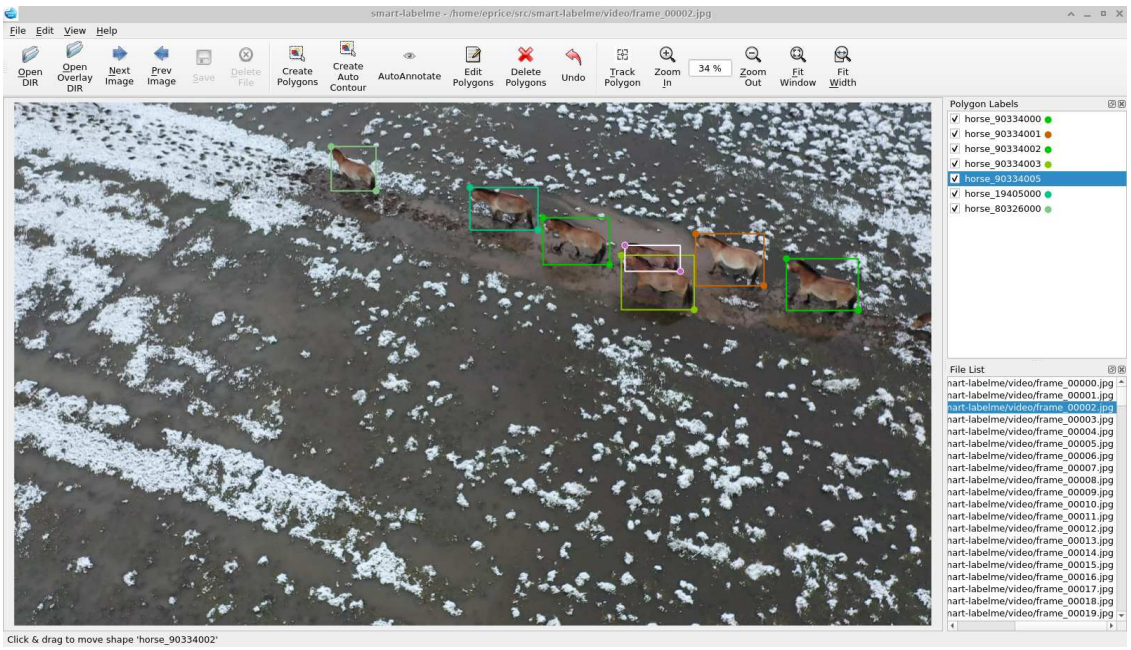


Figure 3.5: Screenshot of the experiment. Annotation of a video with walking horses.

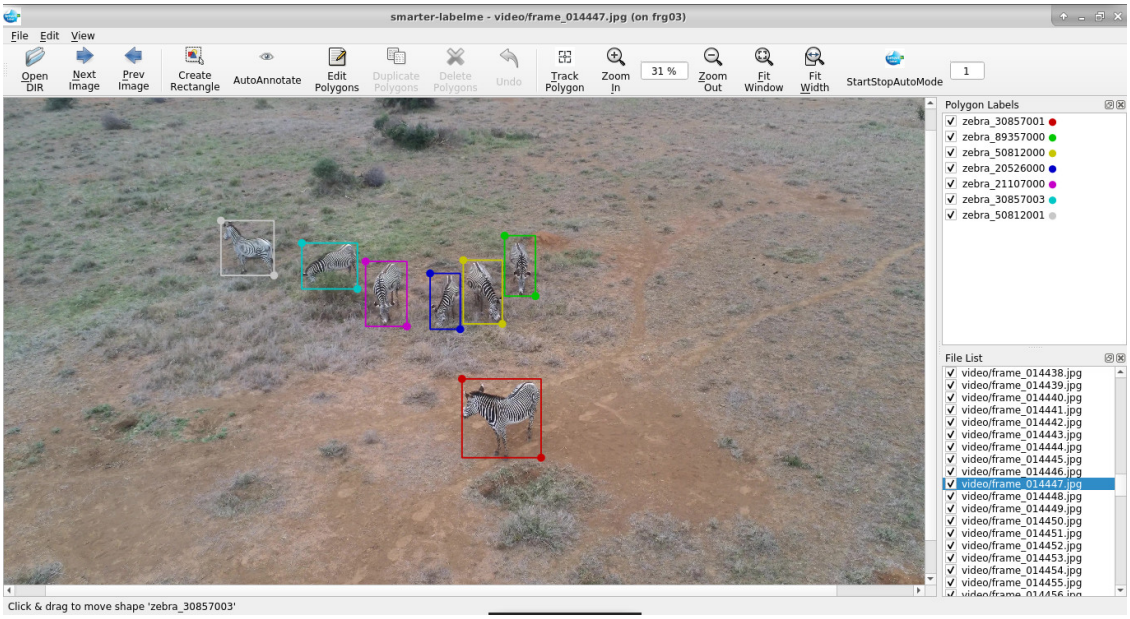


Figure 3.6: A group of Grévy's zebras, annotated with *Smarter-labelme*.

3.4.2 Annotation Accuracy Evaluation

However, we can compare the accuracy of fully automated annotation with existing ground-truth datasets. To evaluate the tool's accuracy, we investigate three main factors. The first is the influence of fine-tuning the SSD detector to the desired object class. The second explores the accuracy with incrementally increasing the influence of the SSD detector compensating for Re^3 drift, and the third explores the effect of spatial overlap of SSD detection with Re^3 prediction for drift compensation.

We test each of these factors in separate sub-experiments by comparing ground-truth and auto-annotated labels in a single fully annotated video. The video used is a 19-min duration on-board video of a drone in-flight, recorded at the MPALA reserve in Kenya in 2022. The video records a group of Grévy's zebras at 24 fps and consists of 27,790 frames in 4K resolution of which 24,432 frames have been manually annotated as part of our group's research on animal behaviour [44], Figure 3.6. The annotations contain 156,575 individual bounding box annotations of 9 individual Grévy's zebras, 7 of which are visible in more than 20,000 frames. The data has been released publicly for the benefit of the community [1].

The metric for accuracy estimation is the average overlap, as defined in [102]. For this, we provide *Smarter-labelme* with ground-truth annotations in every nth frame and evaluate the tracking accuracy over the next $n-1$ frames, averaged over the whole video. For all our experiments except the long-term tracking stability test, we set $n = 32$ since

[1] http://tinyurl.com/SLRDa1

it covers sufficient tracking duration, at which point intervention by the annotator is expected.

Effect of retraining the SSD detector network

We set $n = 32$ and $\alpha = 0.8$. We evaluate the influence of retraining the SSD detector by comparing the accuracy of two SSD network variants on the test video. The first variant, which we call the **Baseline-SSD** detector, uses SSD-Multibox trained on MSCOCO [24] and horses as used in our annotation experiment in Subsection 3.4.1. The second SSD variant is called **Zebra-SSD**. This SSD-Multibox variant uses the weights from [44], which was **Baseline-SSD** retrained with the addition of 979 annotated drone video frames of 34 individual zebras from 5 different drone videos from the same location and time of year, and does not include the test video.

Effect of the weight of drift compensation on accuracy

The weighted average between Re^3 and SSD-Multibox detection is described in Subsection 3.3.7. We compute the average with $\text{Track} = \alpha\text{SSD} + (1-\alpha)\,\text{Re}^3$. In this experiment, we vary α between $\alpha = 0.0$, which means no drift compensation is performed, showing the stability of Re^3 only, and $\alpha = 1.0$, which simulates tracking by detection (in this case, Re^3 only has an effect if SSD fails to detect anything). We run Smarter-labelme with the Zebra-SSD detector and $n = 32$.

Effect of spatial sensitivity of drift compensation on accuracy

Previously, in Subsection 3.3.7 we set the minimum Jaccard overlap between detection and Re^3's tracking estimate to $\text{IOU} = 0.8$. This value had been determined as "good-enough" through trial and error. In this experiment, we systematically evaluate the accuracy of varying Jaccard overlaps between $0.0 \leq \text{IOU} \leq 1.0$. The experiment uses the Zebra-SSD detector with $n = 32$ and $\alpha = 0.8$.

Long-term tracking stability

In this experiment, we evaluate the long-term tracking stability over 1 min, that is, n=1440 (24 fps x 60 s) frames. The goal of this exercise is to explore the tracking accuracy beyond 32 frames and its rate of decrease with an increase in the number of consecutive frames.

Effect of global camera motion estimation

Estimating the global camera motion helps accuracy in case of large camera motions and low frame rates. With $n = 32$ and $\alpha = 0.8$ with Zebra-SSD, we compare the accuracy with and without global camera motion estimation enabled.

Binary search tracking

The ability to cope with larger camera motions allows an alternate annotation scheme. Instead of tracking the objects over $n-1$ consecutive frames, each of which can cause the tracker to lose the object, we can annotate every frame with at most $\log_2(n)$ consecutive tracking steps in a binary tree approach. We track from 0 to $\frac{n}{2}$ in a single step, then use this new starting point to $\frac{3n}{4}$ as well as the original starting point to track to $\frac{n}{4}$, and so forth. We compare the accuracy of this approach to the standard sequential tracking.

Performance on various object classes

The GMOT-40 dataset [101] provides various visually challenging multi-object tracking image sequences with instance bounding box annotation. We convert the GMOT-40 ground truth into *Smarter-labelme* JSON annotations. We then calculate the average accuracy of *Smarter-labelme* on this data, when initialized on the first frame of every 32 frame snippet, and when initialized with ground truth on the first frame of the sequence only.

3.4.3 Results of the Annotation Speed Evaluation

Deep tracking can lead to annotation speed gains

We conducted experiments as described in Section 3.4.1 with 5 different human annotators. The results are shown in Table 3.1. On that video, the annotators achieved on average 13% faster annotation speeds with our method than the baseline. *Smart-labelme* [89] using OpenCV was on average 3% slower than the baseline. As seen in Figure 3.7, there were noticeable differences between the annotators. Annotator 3 reported, that they spent a lot of time re-labelling and correcting auto annotations made with SSD-Multibox in the first frame, which had a comparably large impact due to their slow annotation speed. All annotators did manual corrections to the tracker's labels in every frame as

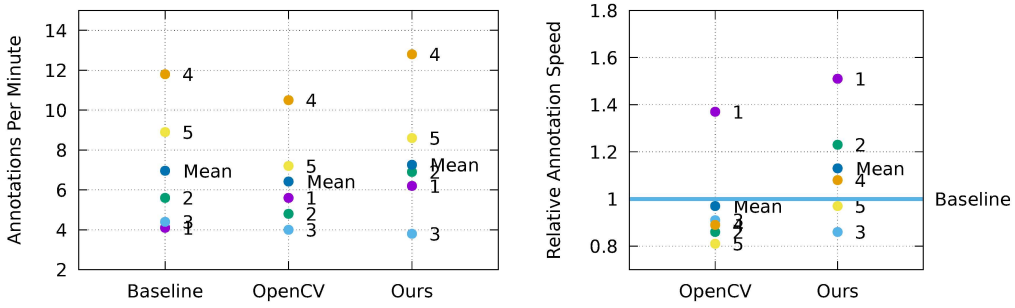


Figure 3.7: Visualized experiment results. The absolute and relative annotation speed of the respective annotators are displayed for each method, compared to the baseline.

Table 3.1: Results of the annotation experiments.

Annotator	Seq.	Exp.	Frames	Annotations	Speed-up
1	1	Baseline	6	41	1.0
	2	**Ours**	**8**	**62**	**1.51**
	3	OpenCV	7	56	1.37
2	1	OpenCV	6	48	0.86
	2	**Ours**	**8**	**69**	**1.23**
	3	Baseline	7	56	1.0
3	1	**Baseline**	**6**	**44**	**1.0**
	2	OpenCV	5	40	0.91
	3	Ours	5	38	0.86
4	1	**Ours**	**16**	**128**	**1.08**
	2	Baseline	15	118	1.0
	3	OpenCV	14	105	0.89
5	1	OpenCV	9	72	0.81
	2	**Baseline**	**12**	**89**	**1.0**
	3	Ours	11	86	0.97
Mean		OpenCV	8.2	64.2	0.97
		Baseline	9.2	69.6	1.0
		Ours	**9.6**	**72.6**	**1.13**

needed, but consistently reported that this process took longer if the tracking error was larger. This explains why the baseline method, which simply copies the previous frame's labels to the following, outperformed OpenCV, which had a tendency to occasionally make large tracking errors. Another contributing factor was the long processing time of OpenCV, which adds several seconds to every annotated frame. The deep methods track significantly faster since they use GPU on top of being more accurate than OpenCV. On the other hand, the baseline method has low accuracy, but does not spend any time processing between frames.

Speed gains are annotator and task dependent

The experiments show that using our approach, we can accelerate the annotation process while keeping the processing overhead low. Secondly, humans are faster with a trivial method (copying bounding boxes and manually adjusting them) than correcting a bad automatic label. During tests, we also found cases where, in a complex scene, many objects were tracked correctly, while some objects, due to their visual appearance, upset the tracker and resulted in consistently bad labels. The annotation tool should, therefore, allow the annotator to disable the accelerated tracker for individual objects if they notice such a malfunction, and revert to the trivial method of copying bounding boxes between frames. We have therefore added this functionality to our open-source implementation.

The benefit of our approach depends greatly on the required annotation quality. In our experiment, some annotators aspired to make pixel-exact annotations, which required

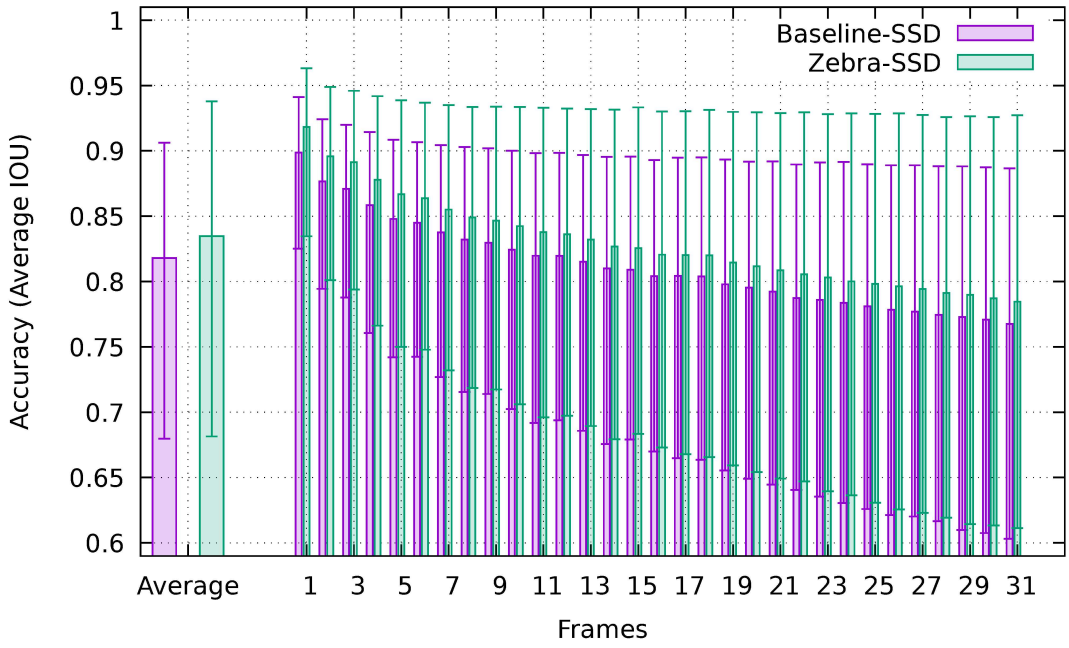


Figure 3.8: Tracking-Accuracy (average overlap) improvement over 32 consecutive frames averaged over 747 32-frame windows with 4775 zebra annotations, after retraining the detector network on in-domain annotations. The small bars show the positive and negative mean deviation across all labels for that frame.

careful monitoring and corrections to every bounding box in every frame. If rough annotations are sufficient, both the validation process and the correction — if needed — can be significantly faster. The performance of the detector and tracker might often be sufficient for the task. In this case, the processing time of the tracker becomes a major bottleneck for annotation and additional effort is required to optimize the software for speed. On the other hand, if very high-quality annotations are needed, the overall speed can increase if the quality of machine accelerated annotations is improved. This can be done by fine-tuning the detector on already annotated data. We discuss the benefit of fine-tuning the detector in the following Subsection 3.4.4.

3.4.4 Results of the Annotation Accuracy Evaluation

Fine-tuning SSD improves accuracy

The results in Figure 3.8 show that re-training the detector on in-domain data noticeably improves the tracking accuracy consistently across frames. This is especially relevant for annotation tasks that involve large video datasets – retraining SSD-Multibox with a small number (979 annotations in our case) of relevant in-domain ground-truth annotations can

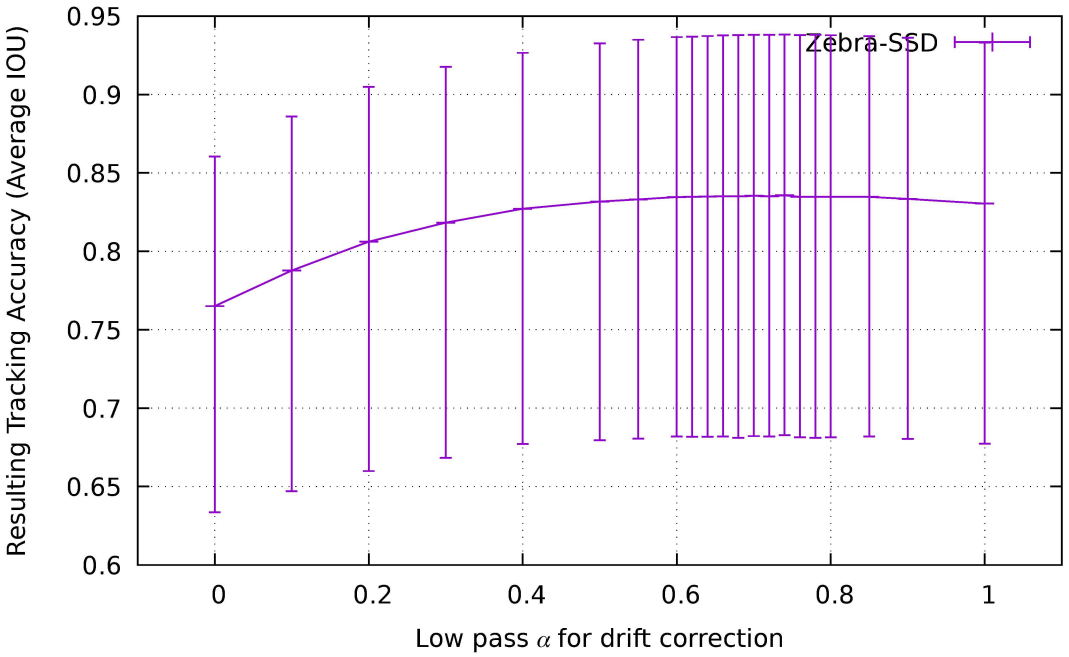


Figure 3.9: Tracking-Accuracy (average overlap) over 32 consecutive frames averaged over 747 32-frame windows with 4775 zebra annotations, using Zebra-SSD with different α for drift compensation. The bars show the positive and negative mean deviation across all labels. Good results are achieved with $0.68 \leq \alpha \leq 0.8$. The default value is $\alpha = 0.8$

lead to significant real-world benefits in terms of reducing annotation cost for the user. Furthermore, the accuracy of greater than 0.8 is maintained up to 25 frames, suggesting that an even higher annotation speed can be achieved for annotation tasks that do not require pixel-exact annotations.

Relying more strongly on the detector can improve Re^3 drift compensation

The results in Figure 3.9 show that the highest accuracy is achieved with the combination of both (SSD-Multibox detection and Re^3 tracking) methods around $\alpha = 0.74$, with very similar results between $0.68 \leq \alpha \leq 0.8$. Our experiment considers tracking over 32 frames, which is a sufficiently long time for the Re^3 tracker to show significant drift. This might explain the finding of a high α value to achieve 0.8 average IOU. Annotators dealing with shorter snippets of consecutive frames might want to explore using a smaller α value and test for its accuracy. Nonetheless, it should be kept in mind that the tracker accuracy is also dependent on the object's motion. In our case, zebras walking across the frame led to lower tracking performance compared to zebras walking towards or away from the camera, irrespective of a high α value.

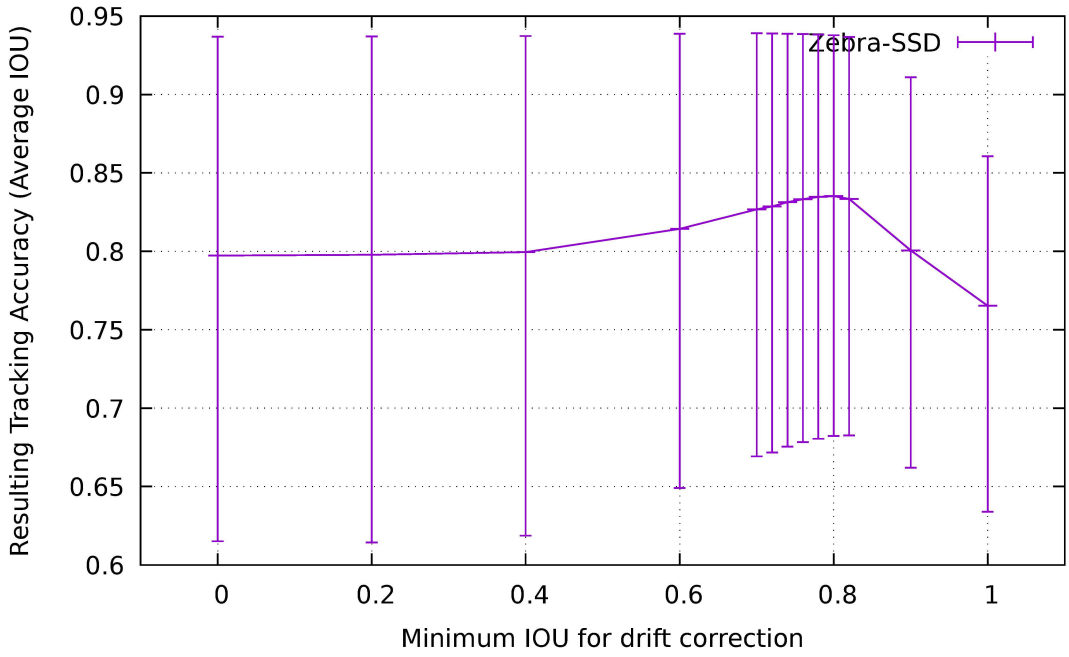


Figure 3.10: Tracking-Accuracy (average overlap) over 32 consecutive frames averaged over 747 32-frame windows with 4775 zebra annotations, using Zebra-SSD with different minimum overlap required for detections to be used for drift compensation. The small bars show the positive and negative mean deviation across all labels. The peak performance is with IOU $= 0.82$. The default value is IOU $= 0.8$

Requiring high spatial overlap between detector and tracker for drift compensation can improve accuracy

The results in Figure 3.10 show the highest accuracy is indeed achieved with a spatial sensitivity of IOU $= 0.8$, which was the default value previously chosen based on trial and error. Though not surprising, our results suggest good spatial agreement between the SSD-Multibox detector and the Re^3 tracker leads to improved accuracy.

Accuracy drops over time when unsupervised

Figure 3.11 shows that once the tracker loses an object, it is very unlikely to recover. As a result, the long-term trend of the accuracy is monotonously approaching zero as more and more labels representing objects are lost. How fast this happens depends both on the tracker performance and the difficulty of the tracking task. A detector capable of detecting the tracked objects can delay this decrease. Small and fast-moving objects, rapid camera movement and object occlusions contribute to a quicker decline.

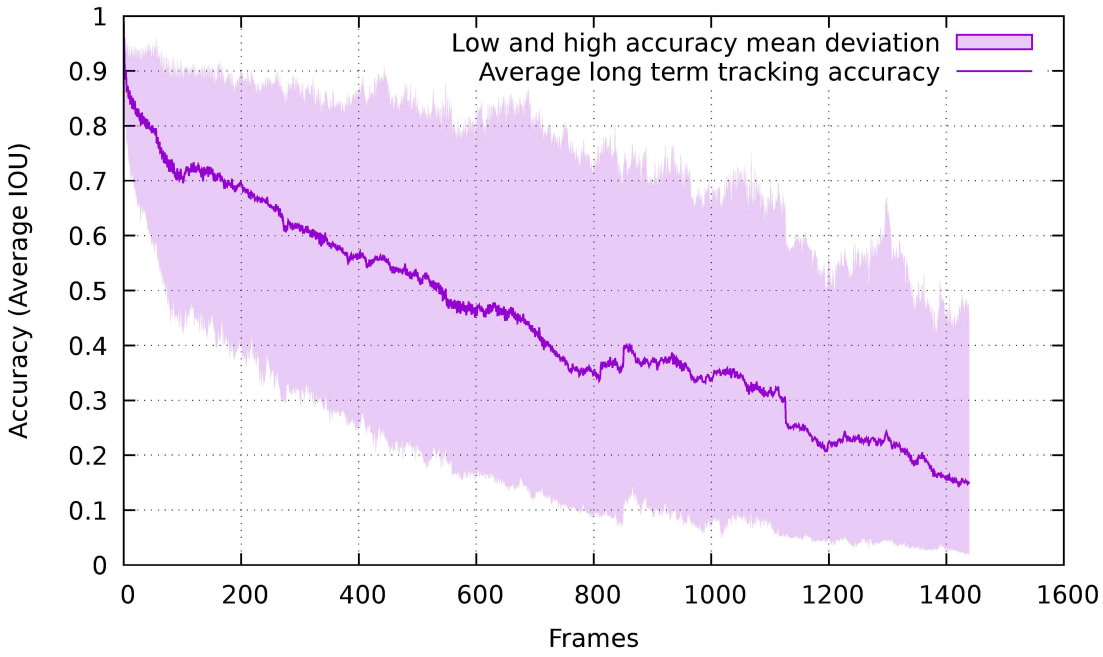


Figure 3.11: Tracking-Accuracy (average overlap) over 1440 consecutive frames, averaged over 17 videos and 108 individual zebra annotations, using Zebra-SSD. The plot includes positive and negative mean deviation across all labels.

Compensating for global camera motion improves tracking accuracy

Accounting for the camera motion updates the search area for each annotated object with the shift in pixel-space before performing the Re^3 tracking prediction. This reduces drastic jumps in object position in pixel-space caused by camera motion. Figure 3.12 compares the tracking accuracy with and without global camera estimation enabled. We observe consistently improved tracking by accounting for global camera motion throughout 32 frames when tested at 24 fps. The advantage of accounting for global camera motion becomes more pronounced with lower frame rates, as described below.

Figure 3.13 shows the comparison with $n = 32$, but at a much lower frame rate (1 fps instead of 24 fps). Although global camera motion tracking does improve the tracking accuracy, this is still a very challenging example with low overall accuracy. The tracker has difficulty following the objects across 1 second time-jumps, since the objects themselves can traverse significant distances during this time.

Binary tracking can improve annotation speed while maintaining accuracy

The binary tracking approach lined out in 3.4.2 is inferior if the initial step is too long. If objects have moved too far between two frames, as it happens when either the camera

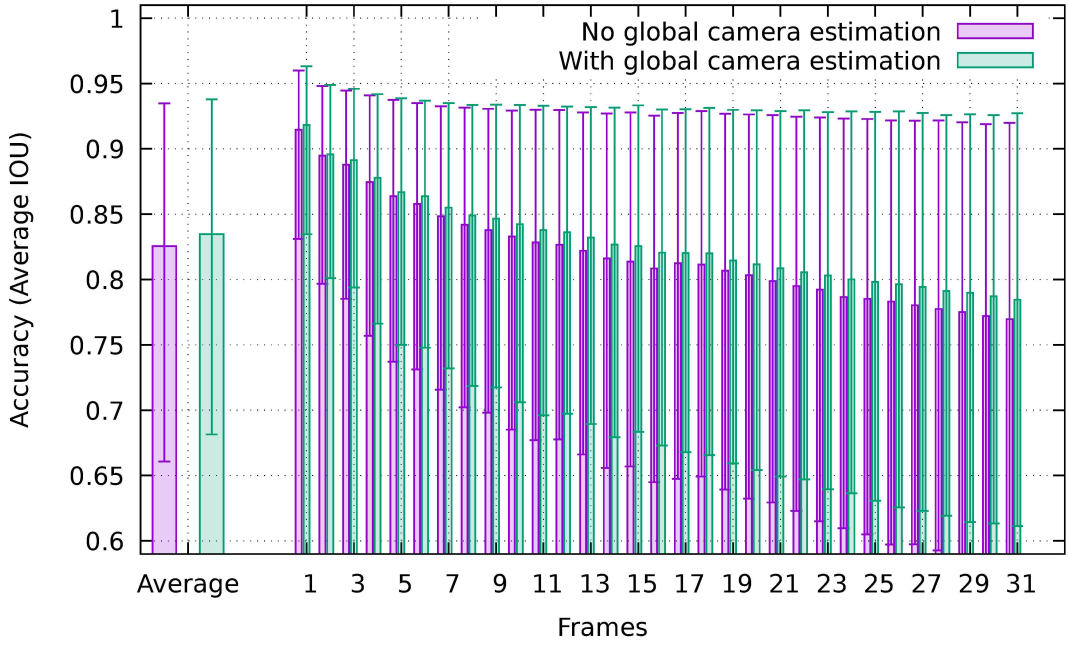


Figure 3.12: Tracking-Accuracy (average overlap) over 32 consecutive frames averaged over 747 32-frame windows with 4775 zebra annotations, using Zebra-SSD at 24fps, including positive and negative mean deviation across all labels. The plot compares accuracy with and without global camera motion estimation.

or the object or both are moving, the tracking problem becomes much harder and overwhelms the capability of the tracking algorithm. As seen in Figure 3.14, a single large step over $\frac{n}{2} = 16$ frames leads to noticeably worse accuracy compared to 16 consecutive small steps. However, for smaller steps over $\frac{n}{4} = 8$ frames or less, the binary approach outperforms the linear tracking result when global camera motion estimation is available.

This behaviour depends primarily on the size and speed of to-be-tracked objects. The faster objects move relative to their size, the smaller the time steps need to become to still allow tracking. A binary approach is only beneficial below this minimum time.

Tracking accuracy is dependent on object class and video quality

We test the accuracy of *Smarter-labelme* on the GMOT-40 dataset that contains 40 different object classes, with each video sequence corresponding to one object class and a varying number of object instances. Two tracking approaches are evaluated. First, when *Smarter-labelme* is initialized on the first frame of every 32 frame snippet, and second, when initialized with ground truth on the first frame of the sequence only. We evaluate both cases with Baseline-SSD, as depicted in Figure 3.15. The accuracy is highly dependent on the visual difficulty of the sequence, as apparent in Figure 3.16.

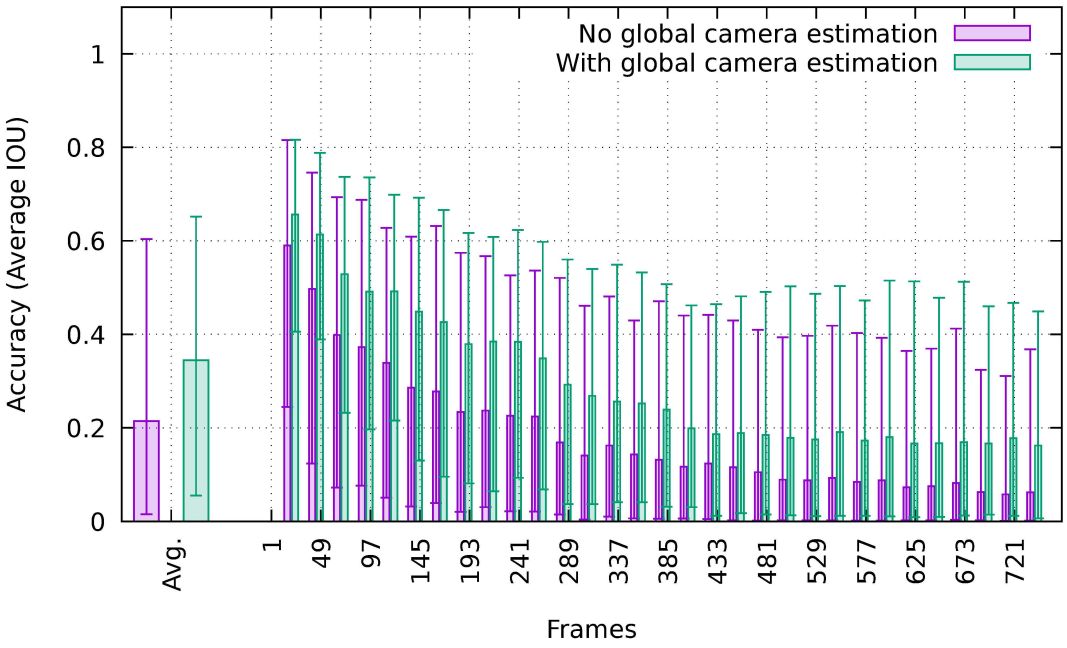


Figure 3.13: Tracking-Accuracy (average overlap) over 32 consecutive frames averaged over 747 32-frame windows with 4775 zebra annotations, using Zebra-SSD at 1fps, including positive and negative mean deviation across all labels. The plot compares accuracy with and without global camera motion estimation.

Smarter-labelme performs best if the object's rate of motion relative to their size per frame is low, so the tracker can find the moved instance within the vicinity of their original location. Small changes in appearance from frame to frame lead to better results than large changes. Good foreground-background contrast and sharpness in the image is also helpful, while motion blur and fuzzy object-boundaries are harmful to the performance. All of these are correlated with video quality. Good illumination, focus, short shutter-timings and high frame rate are beneficial for tracking. Furthermore, the performance is negatively affected by overlapping objects and occlusions. The class/type of the object and the scene seems to have only a secondary effect on the tracking performance. Visual conditions appear to have a much higher impact on tracking success than detector performance and in-domain retraining.

Note on comparison to GMOT-40 leaderboard.

It is not feasible to compare Smarter-labelme's performance to the leaderboard of the GMOT-40 challenge[103]. Although the overall performance of 16% MOTA looks at first glance competitive to state-of-the-art algorithms over the whole sequences, there is a significant difference in the way GMOT algorithms are initialized. For GMOT Evalu-

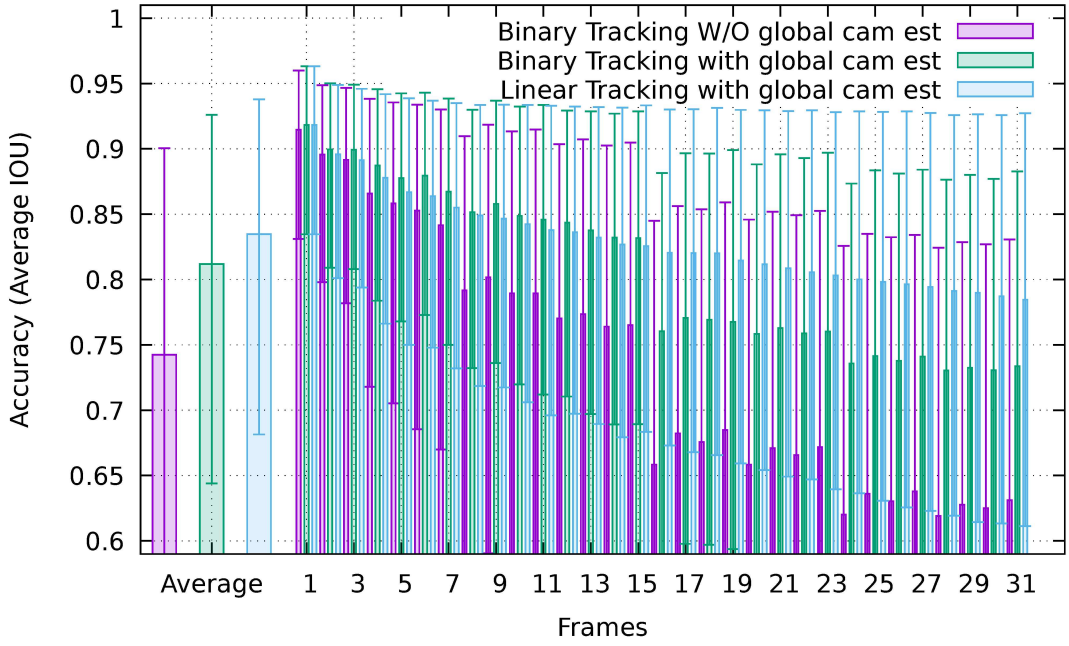


Figure 3.14: Tracking-Accuracy (average overlap) over 32 consecutive frames averaged over 747 32-frame windows with 4775 zebra annotations, using Zebra-SSD at 24fps, including positive and negative mean deviation across all labels. Shown is the performance of a binary search tracking with and without global camera motion estimation compared to sequential tracking. The binary search tracking yields superior results for $n < 16$ frames but, due to the large tracking error in a single 16 frames wide time-jump, results for $n > 16$ are worse.

ation, only one object instance in the first frame is given to the GMOT algorithm, which then needs to identify and track all other instances in the video on its own. In contrast, our method initializes all instances with annotated ground-truth, since Smarter-labelme is not designed to identify similar instances by visual appearance. It also cannot find new instances of this type in subsequent frames unprompted. This gives Smarter-labelme an initial evaluation advantage in the first frame, as well as a disadvantage as new object instances appear later in the video.

3.5 Conclusion and Outlook

We demonstrated a novel approach for accelerating instance label annotation in videos. Our method combines deep networks for detection and tracking, while compensating for camera motion across frames. Altogether, this leads to reduced tracking drift across frames, increases annotation speed, and significantly reduces manual supervision. We

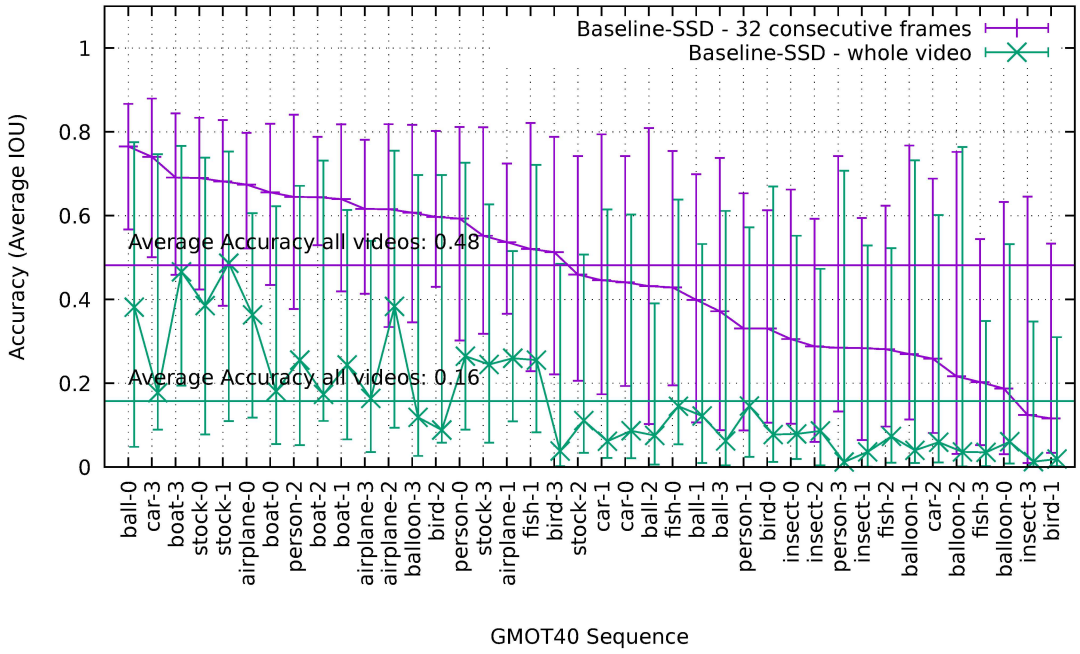


Figure 3.15: Tracking-Accuracy (average overlap) over 32 consecutive frames (blue +) respectively the whole video sequence (cyan *) for each video sequence in GMOT-40, as well as overall average overlap, using Baseline-SSD including positive and negative mean deviation across all labels.

validated the increase in annotation speed by comparing it to baseline methods previously used for annotation tasks. Our experiments on annotation accuracy indicate that multiple factors can influence annotation accuracy. For example, retraining the detector with in-domain data or modifying the deep detector and tracker parameters can lead to further increase in annotation accuracy. Our exploration of these factors should inform users how best to tune them and maximize the benefits gained from our approach for their use-case. We also demonstrated the versatility of our tool by testing its multi-object tracking on the GMOT-40 dataset, which includes 40 object classes covering animate and inanimate objects. Finally, our light-weight implementation can work on CPU and GPU alike, making it widely accessible to the research community. We provide the open-source code, instructions, and the annotated dataset used in our testing for the benefit of the community [25, 92].

Future work may involve online training of the detector on the objects being tracked in real time, using the already labelled annotations as training data. We also foresee higher modularity, to allow different methods for tracking and detection to be employed, while simultaneously preserving the ability to combine tracking and detection for improved results.

Meanwhile, several new tools have become available, such as Meta's "Segment Any-

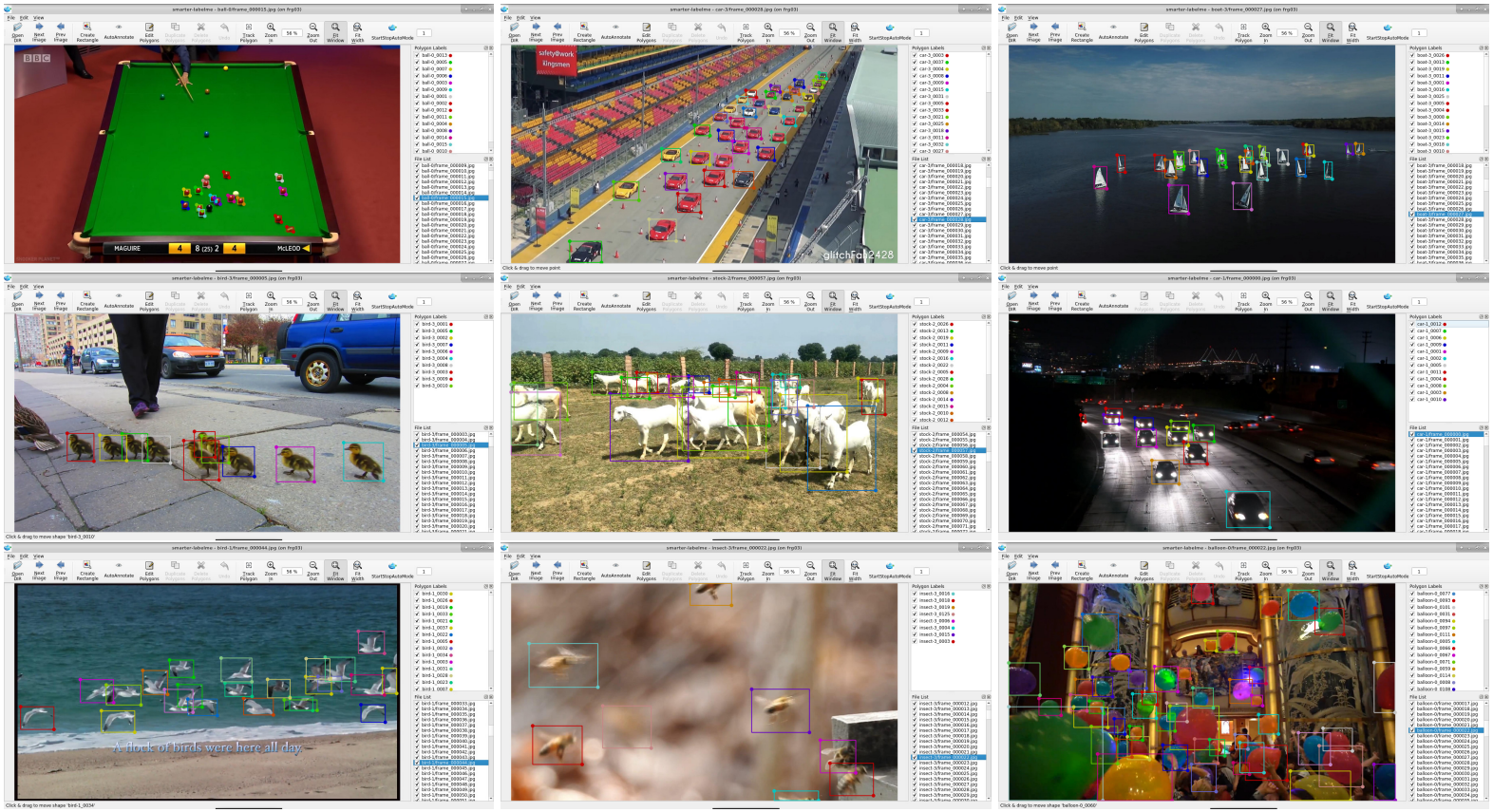


Figure 3.16: Examples of ground truth annotation in GMOT-40 top 3 (top row), mid-range (centre row) and worst 3 (bottom row) video sequences. Smarter-labelme performs best if the objects are high contrast, distinctly separated, have fixed geometry and do not move much between frames. The mid-tier videos show visually challenging light conditions, significant overlap between instances, and increased appearance changes over time. The bottom most sequences have fast motions resulting in large motion in each frame, significant motion blur, drastic morphologic changes in the objects in each frame (such as wing flapping), challenging foreground-background contrast, and severe overlap.

thing" Model (SAM) [104] that perform extremely well on machine-annotating a wide range of data due to the vast amount of training data their authors had available. However, when applied to rare data domains such as aerial videos of wildlife, their performance drops, and it becomes apparent that there is still need for manual data annotation and domain-specific re-training.

Current benchmarks for visual object tracking focus on more complex tasks. The VOT challenges in 2023 and 2024 [105] combine both single and multi object tracking. Most improvement is seen in combined methods that perform detection, segmentation, and tracking simultaneously.

The same is true for detectors. For example, YOLO11 [106] is no longer just a single shot detector. Although the network still operates on a single frame, it unifies bounding box detection, classification, instance segmentation, key point detection and pose estimation in a single method.

Training data for such networks similarly needs to become semantically more rich, which goes beyond the capabilities of what Smarter-labelme can currently do. What would be needed is a machine assisted "annotate everything" tool that employs semantically rich tracking methods for all the data-modalities. DeepLabCut [107] is a step in this direction regarding key points and body-poses, but still lacks the interface for frame-based segmentation or boundary annotation. Other tools such as CVAT [91] cover all the types of annotations, but do not have the level of sophisticated multi-modal machine-assistance. Therefore, more research and development of more powerful, faster and smarter annotation tools is still required.

3.6 Publication

The work lined out in this chapter has been submitted, peer reviewed and presented at the 18th International Conference on Intelligent Autonomous Systems (IAS) in 2023 [27] with the title "Accelerated video annotation driven by deep detector and tracker". Among selected publications, the paper was also chosen for submission to Robotics and Autonomous Systems (RAS) with new content. This additional content includes re-training and re-evaluation on Zebras and evaluation of the tracking-accuracy of the method on the GMOT-40 dataset, which has not been published yet.

3.7 Annotation of data

During a research excursion to the Hortobágy National Park in Hungary in the spring of 2022, several hours of UAV footage were collected, flying commercial off-the-shelf (COTS) multi-copters by hand. The subject was a herd of wild Przewalzki's horses [108] in cooperation with the park operators. This data allowed us to evaluate the detection performance of the pretrained SSD-Multibox detector qualitatively, which was poor when

only trained on MSCOCO.

To apply our cooperative detection and tracking method to these animals, we had to improve the detector to the point, where horses were reliably detected. To this effect, we randomly selected 100 video frames from the entirety of the footage, excluding frames with no horses. This was done by extracting the video into individual images with a very low frame rate, and then randomly sampling frames, manually discarding those without animals. These were then manually annotated using another open-source annotation tool. Since up to 200 individuals were visible in some camera frames, annotating just 100 images resulted in the order of magnitude of 5,000 annotated horse bounding boxes. Fine-tuning SSD-Multibox on this data did not yield satisfying results. The detector quickly lost its ability to detect other classes, as became apparent in a rapidly decreasing test score on the MSCOCO evaluation set, while the detection of horses was only improving marginally. We then opted for training SSD-Multibox on a combined dataset. MSCOCO contains about 120,000 images. To avoid data imbalance and simultaneously reducing the image resolution from 4k to the resolution used in MSCOCO, typically 640×480px, we extracted 20,000 random crops from the 100 annotated images and scaled them to that size. SSD-Multibox was then re-trained on a dataset containing both the original MSCOCO training data and our 20k horse images. The resulting detector showed comparable performance on the MSCOCO evaluation set to the original detector. The mAP scores when evaluating on a second set of random crops on the same annotated horses were surprisingly poor, however when we did a qualitative test, attempting to detect horses in the recorded aerial footage, we achieved very satisfying results.

A second trip to Hortobágy in November 2022 was used to test detection and tracking of horses with this re-trained detector. Although mechanical problems with our copters prevented us from performing autonomous tracking with multiple-vehicles, the detector performed excellently, despite the horses having grown their winter fur, and the weather and flora being seasonably distinct from our first data collection. We successfully tracked a horse with a single copter, following it autonomously for around 50m, before the experiment was aborted.

3.8 Additional Work

The Smarter-labelme annotation and training pipeline also formed the basis for a behaviour classifier network, which attempts to detect the current action performed by any annotated object, as well as the means to annotate such behaviour. This has been made public on BioArxiv as “A Framework for Fast, Large-scale, Semi-Automatic Inference of Animal Behavior from Monocular Videos” [44].

Chapter 4

Airship Simulation and Control

While individual control of a multi-copter can be considered — for the most part — a solved problem, and simulation of these kinds of vehicles could be implemented using readily available ROS components originally developed by ETH Zürich [28], this was definitely not the case for Airships. Although previous work on Airship control has been published since the early 20th century, when these vehicles were used extensively for passenger travel, the use of airships in a cooperative multi UAV environment is mostly uncharted territory.

When we made the decision in our group to use Airships for motion capture, we had no way of simulating an airship, no electronics designed to control it, and very little to go by. Since airship designs are on the market for remote controlled flight, we could relatively easily model and import the individual physical components such as gondola, thrusters and the inflated hull into the Gazebo/ROS physics simulator, which we were using for multi-copters. However, we had no aerodynamic model. Therefore, how aerodynamics and actuators would interact was unclear.

The approach pursued in this chapter uses first principles to estimate aerodynamic forces on each individual part of the blimp based on its approximate geometric shape. We then use Gazebo to calculate the resulting motion of the entire vehicle subject to the sum of these forces.

Blimps, i.e. airships based on a non-rigid gas filled pressure hull, are subject to deformation when exposed to internal and external forces, especially if this pressure is low. We encountered these effects during flight tests, when changes in temperature on the experiment day had a profound effect on the pressure and shape of our vehicle. This prompted us to include non-rigidity in the modelling and evaluate the effects this had on controllability and resulting trajectories.

Although we performed all the work in this chapter on a single, specific vehicle, all the methods and algorithms are designed as generic as possible, to support the majority of possible airship designs and make them as easily accessible for robotic research as multi-copters. It is this architectural choice that facilitates the work in Chapter 6 where the same methods are applied to a significantly more optimized airship design.

Figure 4.1: Our flying blimp. An autonomous deformable lighter-than-air vehicle (LTAV). This image taken from a drone shows our blimp prototype suring flight experiments over fields in the Neckar valley near Tübingen, Germany in 2022.

4.1 Introduction

In the last decade, the term Unmanned Aerial Vehicles (UAV) has become near-synonymous with so called "drones" — small lightweight multi-rotor craft, kept stable by active control using cheap lightweight MEMS sensors that became ubiquitous as a by-product of the smartphone revolution [109]. Their popularity is mirrored in the field of aerial robotic research. The vast majority of work covers multi-rotor craft, and to a lesser degree, fixed wing aircraft and helicopters. Thus, flight electronics, control software and simulation tools are readily available for these types of vehicles [110, 111]. This allows researchers to build up a suitable autonomous flying platform for any research task with ease, using commercial off the shelf (COTS) components and open-source software.

In contrast, lighter-than-air vehicles (LTAVs) such as rigid and non-rigid dirigibles (Figure 4.1) have fallen out of fashion, despite their superiority for some applications [112]. LTAVs are uniquely suited as mobile aerial communication relays, as well as for monitoring and wildlife conservation tasks. They produce little noise and are characterized by high energy-efficiency and long flight times, display benign collision and crash characteristics, pose low danger and cause little environmental impact. However, their comparably high handling complexity, size, lifting gas requirements and cost create an entry barrier for researchers. Unlike heavier-than-air "drones", there have been no COTS flight controllers that support autonomous dirigible flight. Therefore, guidance and control algorithms have to be implemented for each vehicle — even though various suitable control strategies can be found in the literature [112, 113, 114, 115, 116, 117, 118]. Similar to both rotor craft and fixed wing UAVs, dirigibles come in many types of actuator arrangements: fixed or vectoring main thrusters, differential thrust, different tail fin arrangements and auxiliary thrusters, single or double hull, etc. Thus, a control algorithm

for a specific vehicle might not always be applicable to others. Development of control algorithms is further complicated by the lack of realistic simulation. Existing robotic simulation environments [110] are often ill-suited for the characteristics of LTAVs. As LTAVs are highly susceptible to wind and turbulence, it is crucial to accurately model both aerodynamic forces and the effects these have on an inherently non-rigid vehicle. To our knowledge, no existing real-time robotic simulation environment addresses this, although analytic studies of these effects have been conducted in the past [119].

In this chapter, we attempt to solve the aforementioned issues in the context of LTAVs and provide researchers in the field of aerial robotics with the tools they are accustomed to when working with fixed wing and rotor craft. Our contributions, for which all source code is provided[1], are:

1. A robotic simulation framework, using ROS/Gazebo [33, 110] for real-time hardware in the loop (HITL) and software in the loop (SITL) simulation of easily customizable airship models in a realistic virtual environment.
   a) Realistic simulation of wind, turbulence [34] and aerodynamic forces on a custom shaped, modular, non-rigid deformable air frame.
   b) Simulated deformation of the air frame in response to aerodynamic, control, thrust and collision forces.
   c) Simulation of buoyancy variation, changes in rigidity and shape, depending on hull pressure and structural parameters.
2. A baseline guidance and control algorithm for autonomous navigation of airships using hierarchical PI controllers. We employ a generic design, to operate on a large subset of possible airship shape and actuator configurations.
   a) Implemented based on the Librepilot [31, 110] open-source flight control tool chain, to integrate with existing SITL & HITL frameworks and available COTS flight control hardware as well as the above-mentioned simulation framework.
   b) Full integration with ROS [33]. This facilitates easy integration of lighter-than-air vehicles into existing and future robotic research projects for higher level tasks.
3. Through simulation experiments, we show the effect of changes in rigidity and wind turbulence on the vehicle's controllability, including the introduction of oscillation modes not present in rigid models.
4. Through real flight experiments, we demonstrate the flight behaviour of a 5m autonomous blimp UAV, with control coefficients previously determined in our simulation, thus validating our simulation results.

[1] Source code: https://github.com/robot-perception-group/airship_simulation

In Section 4.2, we compare this chapter to the state-of-the-art. We explain our methodology in Section 4.3. Experiments and results are shown in Section 4.4, followed by a remark on limitations in Section 4.4.4 and our summarizing conclusions in Section 4.5.

4.2 State of the Art

The seminal work involving the AURORA airship [112, 118, 120] describes the development of a complete system, including novel on board electronics, communication, control algorithms and a flight simulator, both for control evaluation and human operator training. The hierarchical PI control algorithm described there is similar to our implementation, but it does not utilize dynamic thrust vector control. It also makes many simplifications, including flight at constant airspeed. Although a separate hover control scheme is mentioned, which does use thrust vector control, this is not described in detail. Their simulation assumes rigidity of the air frame. Aerodynamics are modelled based on existing wind tunnel data from a different airship with similar proportions. In contrast, we use modern COTS UAV components and freely available open-source software, which greatly enhances reproducibility and the ability to integrate with common robotic components [33]. We also model non-rigid deformation effects in our simulation. Our aerodynamic approach is more generic and allows modelling arbitrary dirigibles, even when no detailed aerodynamic measurements are available — based on geometric shape.

The literature review paper [119] covers modelling of airships, including analysis of deformation and its effect on aerodynamics, spanning 100 years of research, from wind tunnel tests in the early 20th century [121] to computational fluid dynamics (CFD) analysis in the 2000s [122]. We take inspiration from some presented methods, especially [123], which we apply to real-time simulation of aerodynamically relevant components. We consider each of these components rigid internally, but allow non-rigid interactions on joints between them. Our simulation does not take aerodynamic cross-interactions between individual components into account. However, we argue that these can be approximated by artificially adjusting lift and drag coefficients to minimize the overall modelling error. This could be done, for example, by comparison to CFD analysis of the rigid shape. However, this is not covered in this chapter.

More recent work on airships often involves coupling control with perception and computer vision, such as [113, 117, 124, 125]. All of which make simplifying assumptions, such as frame rigidity and disregard of wind turbulence.

A novel airship guidance and control scheme was presented in [116], using a backstepping control strategy. Its performance is mathematically proven utilizing a highly simplified analytical plant model under the assumption of rigidity. No real-world experiments have been made. This has inspired some more recent theoretical work on various airship guidance schemes [126, 127]. However, none of it seems to have been validated in real-world flight. In contrast, we base our model on observed real-world flight behaviour and validate our results in additional real-world flights.

The airship control problem can also be addressed with a model free or learned-model approach [114, 115]. These techniques are well suited to deal with the dynamics of a deforming vehicle under changing conditions from a control perspective, and can be directly applied to the real world. However, a simulation, as presented in this chapter, is still required to validate and compare the performance of learned algorithms under controlled, reproducible conditions.

4.3 Methodology

4.3.1 Simulation

Gazebo is a physics simulator often used in robotics, which seamlessly integrates with the Robot Operating System (ROS) [33]. Gazebo models rigid physical objects (primitives) based on shape, mass, and moments of inertia. Using an XML/URDF description file, these component objects can be described and linked together into hierarchical compound objects. Rigid joints are simulated, using very stiff springs with high damping. These joint parameters can be altered for both linear and rotational freedom. Motion limits, stiffness, and dampening properties can be set to simulate non-rigid compound objects. The simulator employs a solver engine to calculate object motions under Newtonian physics, taking all joint forces, collisions, and gravity into account. This can be extended through plugins that add additional forces to component objects or joints to model actuators or thrusters. The simulator models time in discrete time steps with configurable temporal resolution and can handle most scenarios in real time with sufficient accuracy. Simulated sensors, which provide simulation data measurements for IMU and positioning sensors as ROS messages, are also provided by plugins attached to their respective component objects.

We model a blimp (Figure 4.2) as a flexible compound object, consisting of rigid primitive component objects for all actuated and non-actuated components. These are fins, rudders, the gondola, thrusters, ballast weights, etc. We model the main hull with at least two separate components, to allow flexibility of the shape as well as higher granularity of forces during rotations. Plugins are added to these primitives to model buoyancy, aerodynamic forces, control actuation and thrust. This setup allows the physical properties of the blimp to be modelled as the sum of its physical components. Mass, dimensions and inertial moments of homogeneous basic shapes can be easily determined by first principles or by measuring the corresponding part on real hardware.

Buoyancy

Buoyancy is modelled as a single upward force on sections of the hull based on their volume and coefficients. Buoyancy coefficients can be altered at run-time during the simulation, to experiment with the effects of changing buoyancy.

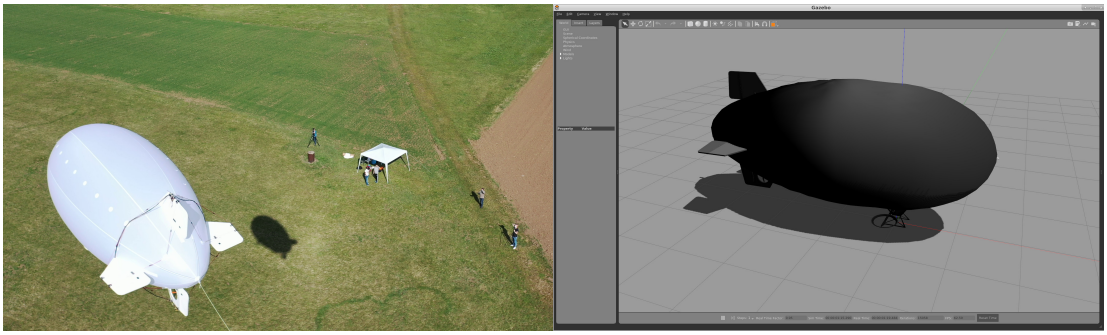


Figure 4.2: Our real-world blimp with our team and the corresponding simulated object. The image on the left shows more drone footage from flight experiments near Tübingen, Germany in 2022. The screenshot on the right shows the simulated airship in the Gazebo/ROS simulator.

Wind

We model wind as a turbulent flow field, using the Dryden turbulence model (MIL-F-8785C) [34], the implementation of which is from the FlightGear open-source flight simulator [111]. It must be noted that this frequency model simulates both the spatial and temporal variability of turbulence in a single, moving point only. As such, we can only calculate a single, time variable global flow vector, which we assume constant in space; the same assumption that was made by FlightGear [111]. For small airships, we deem this is an acceptable limitation. This flow vector is published as a ROS message, so it is available to other plugins at any time.

Lift and drag

For all primitive objects that are relevant from an aerodynamic point of view, including hull sections, we calculate the local lift and drag forces based on the orientation and motion of the object through the flow field, defined by the current wind flow vector. These are then integrated over by the physics solver to model the effect on the whole vehicle as well as its deformations. We model two types of basic lifting primitives, "quasi-planar" and "quasi-cylindrical". Let $\overline{f}$ be the flow vector relative to the primitive, consisting of orthogonal components, f_x from the front, f_y from the left and f_z from above. For quasi-planar types, f_y is ignored. Quasi-cylindrical objects are rotated around their x (forward) axis. This yields a rotated flow ${}_r\overline{f}$ such that

$$ {}_r\overline{f} = \left[f_x, 0, \left|f_y + f_z\right|\right] \ . \tag{4.1} $$

From that point on, quasi-cylindrical sections are treated the same as quasi-planar shapes. We calculate the angle of attack α, the normalized drag vector $\hat{f}_d$, normalized lift vector

$\hat{f}_l$ and dynamic pressure q_d as

$$\alpha = \arctan\left(\frac{f_z}{f_x}\right) , \tag{4.2}$$

$$\hat{f}_d = \frac{\overline{f}_d}{|\overline{f}_d|} \text{ for } \overline{f}_d = [f_x, 0, f_z] , \tag{4.3}$$

$$\hat{f}_l = \hat{f}_d \times \left[0, \frac{\alpha}{|\alpha|}, 0\right] , \tag{4.4}$$

$$q_d = k\left(f_x^2 + f_z^2\right) . \tag{4.5}$$

Let A be the relevant area of the object, given coefficients c_{l_0}, c_{d_0}, c_{d_1} and α_{stall}. We calculate forces for lift $\overline{F}_l$ and drag $\overline{F}_d$, with

$$\overline{F}_l = q_d A \hat{f}_l c_l \ , c_l = c_{l_0} \begin{cases} \frac{|\alpha|}{\alpha_{stall}} & \text{if } |\alpha| \leq \alpha_{stall} \\ \frac{\pi - 2|\alpha|}{\pi - 2\alpha_{stall}} & \text{if } |\alpha| > \alpha_{stall} \end{cases} , \tag{4.6}$$

$$\overline{F}_d = q_d A \hat{f}_d c_d \ , c_d = c_{d_0}\left(1 - \frac{2|\alpha|}{\pi}\right) + c_{d_1}\frac{2|\alpha|}{\pi} . \tag{4.7}$$

Both functions are highly simplified linear approximations (Figure 4.3) of the true lift and drag curves. For each primitive object, only 4 aerodynamic coefficients are needed. The approximate drag coefficients for basic primitive shapes, both for frontal and sideways flow, are commonly known, as are the lift coefficients of thin rectangular plates or cylinders. Care has to be taken to adjust the coefficients when nearby primitives interact with each other on a compound object. For example, only the first and last section of the hull would have a high c_{d_0} due to pressure drag, while any intermediate sections would only experience friction drag, which typically is an order of magnitude lower. Due to the linear contribution, assuming identical A for n hull sections and frontal flow drag coefficient ${}_H c_d$ for the basic shape, we observe

$$\sum_{k=1}^{n} {}_k c_{d_0} = {}_H c_d , \tag{4.8}$$

and set the coefficients for the hull sections accordingly, weighted by their expected drag contribution.

As another example, fins attached to the hull might encounter increased lift due to increased dynamic pressure of the airflow around the hull, which can be modelled by artificially inflated lift coefficients for these surfaces.

If an aerodynamic model exists for a specific vehicle, it should be possible to solve for a "best fit" of the primitive coefficients. However, our approach has the advantage that a roughly approximated aerodynamic model can be created based on geometric shape

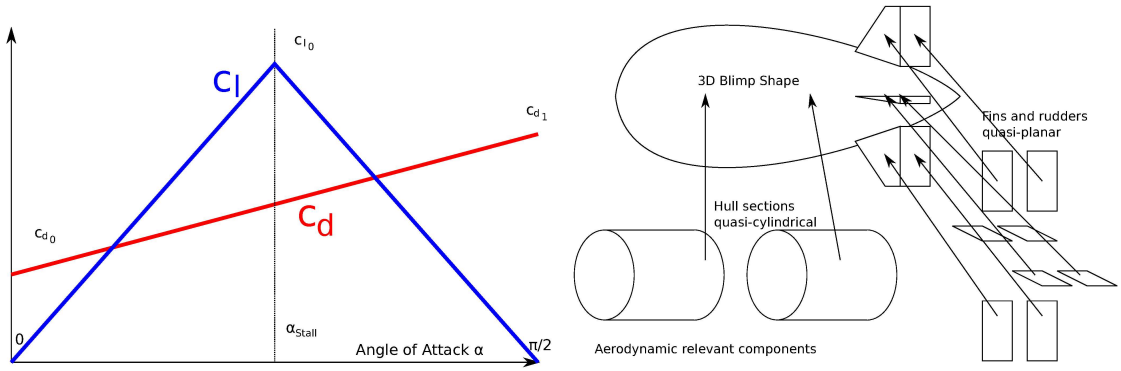


Figure 4.3: Functions for lift and drag coefficients for a simulated object component. These are calculated for each primitive component of the blimp for which aerodynamic forces are relevant.

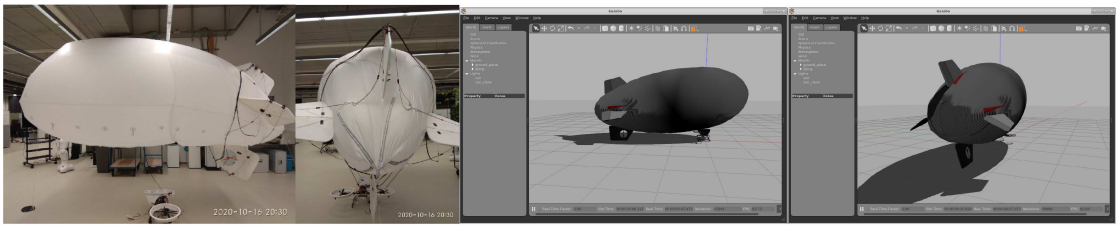


Figure 4.4: Deflation of the blimp affects its shape, both in reality and simulation. The hull is visually rendered rigid, but aerodynamically it is simulated in sections. Under their weight, the fins sink into the hull and drag the tail down until the bottom fin carries part of the blimp's weight. These images show severe deflation for better visibility. Flight in this state would not be possible. The images were taken inside the lab at MPI Tübingen, Germany in 2022.

only, which we argue is a valuable property for evaluating new vehicle designs.

Non-rigidity

Gazebo joint properties in compound objects can be manipulated through a ROS service while a simulation is running. We wrote a script that can simultaneously adjust both the rotational and linear spring stiffness, as well as the range of free rotation for all joint connections between the blimp hull and other components such as fins and gondola around all axes of rotation. The same program also adjusts the buoyancy of the blimp. This way, the effects of a drop or increase in lifting gas pressure and volume can be evaluated at any time, both regarding flight behaviour and controllability (Figure 4.4).

4.3.2 Control

We employ a cascaded PI controller design similar to the controller presented in [112, 118], with extensions. Librepilot [31, 110] offers a vehicle-agnostic control hierarchy based on virtual control axes, which are mapped to physical actuators through a configurable matrix. This maps a "Pitch" virtual axis unto the elevator servos, the "Yaw" virtual axis on both rudder servos and a yaw thruster, as well as a "Thrust" virtual axis on both main thrusters. We decided not to employ main thruster differential thrust for control, due to the low torsional moment and high strain inflicted between gondola and hull. A simple PI control loop controls the yaw rate, while a hierarchical controller with an outer loop proportional term controls the pitch angle based on an PI inner loop controlling pitching rate. Implementations for these already exist and are utilized both for fixed wing and multi-rotor control.

In extension, we implemented a novel path follower, which determines the set points for the lower level control: pitch P, yaw rate $\dot{Y}$, thrust T and thrust vector angle $0 \leq \gamma \leq \frac{\pi}{2}$. Its inputs are a set point velocity vector in 3D space $\overline{v}_S$, the airspeed v_I, the velocity in 3D space $\overline{v} = [v_x, v_y, v_z]$ and the current attitude of the vehicle in Euler angles $q = [q_r, q_p, q_y]$. All the state estimates come from the flight controller's state estimation using an extended Kalman filter (EKF). If no airspeed sensor is available, v_I is assumed to be the forward component of $\overline{v}$. We compensated the wind to calculate motion relative to the flow field. Let $q(v_I)$ be a function that rotates v_I into the world-frame. We then calculate the estimated wind flow vector $\overline{f}_l$ and the corrected velocity in air $\tilde{\overline{v}}$, as

$$\overline{f}_l = \overline{v} - q(v_I) \ , \tag{4.9}$$

$$\tilde{\overline{v}} = [\tilde{v}_x, \tilde{v}_y, \tilde{v}_z] = \overline{v} - \overline{f}_l \ . \tag{4.10}$$

For each vehicle a minimum controllable airspeed v_{min} and a maximum achievable airspeed v_{max} are defined. We solve for a wind-corrected set point $\tilde{\overline{v}}_S$, with

$$\tilde{\overline{v}}_S = \begin{bmatrix} \tilde{v}_{sx} \\ \tilde{v}_{sy} \\ \tilde{v}_{sz} \end{bmatrix} = b\overline{v}_s - \overline{f}_l \ , \begin{cases} b > 0, \ \left| b\overline{v}_s - \overline{f}_l \right| = v_{min} & \text{if } \left| \overline{v}_S - \overline{f}_l \right| < v_{min} \\ b = 1 & \text{if } v_{min} \leq \left| \overline{v}_S - \overline{f}_l \right| \leq v_{max} \\ b > 0, \ \left| b\overline{v}_s - \overline{f}_l \right| = v_{max} & \text{if } \left| \overline{v}_S - \overline{f}_l \right| > v_{max} \ . \end{cases} \tag{4.11}$$

If the wind is stronger than the vehicle's maximum speed, there might be no solution with $b > 0$. In this case, the solution with the highest b is chosen, which minimizes the positional error accumulated under these conditions.

Separate controllers are employed for direction C_d, airspeed C_v, climb rate $C_{\dot{h}}$ and thrust vector C_γ. Let $\sphericalangle\left(\overline{a}, \overline{b}\right)$ be the signed angular difference between $\overline{a}$ and $\overline{b}$ while $\int_{\langle \pm l_i \rangle}$ shall denote an integral accumulator limited by $\pm l_i$. Then

$$\dot{Y} = C_d\left(\tilde{\overline{v}}_S, \tilde{\overline{v}}\right) = {}_{\dot{Y}}k_p \sphericalangle\left(\left[\tilde{v}_{sx}, \tilde{v}_{sy}\right], \left[\tilde{v}_x, \tilde{v}_y\right]\right) \ , \tag{4.12}$$

$$P = C_h\left(\tilde{\bar{v}}_S, \tilde{\bar{v}}\right) = {}_P k_p\left(\tilde{v}_{sz} - \tilde{v}_z\right) + \int_{\langle \pm_P l_i \rangle} \left({}_P k_i\left(\tilde{v}_{sz} - \tilde{v}_z\right) dt\right) , \tag{4.13}$$

$$T = C_v\left(\tilde{\bar{v}}_S, \tilde{\bar{v}}\right) = {}_T k_p\left(\left|\tilde{\bar{v}}_S\right| - \left|\tilde{\bar{v}}\right|\right) + \int_{\langle \pm_T l_i \rangle} \left({}_T k_i\left(\left|\tilde{\bar{v}}_S\right| - \left|\tilde{\bar{v}}\right|\right) dt\right) + {}_{P\times T} k_p P , \tag{4.14}$$

$$\gamma = C_\gamma\left(\tilde{\bar{v}}_S, \tilde{\bar{v}}\right) = {}_\gamma k_p P . \tag{4.15}$$

The main addition to [112, 118] is the thrust vector term C_γ which, in combination with the proportional climb speed cross term ${}_{P\times T}k_p P$ in C_v, employs the main thrusters for altitude control. For a vehicle without thrust vector control, the corresponding coefficients ${}_\gamma k_p$, ${}_{P\times T}k_p$ would remain 0.

4.3.3 Middleware

The flight control algorithm is implemented in C/C++ and executed either on a physical embedded flight controller (Openpilot Revolution) [110] on the LTAV, both in flight or for hardware in the loop (HITL) simulation, or compiled as a computer program for software in the loop (SITL) simulation. The Librepilot Ground Control Software (GCS) connects to either, using telemetry or networking. The GCS offers a HITL/SITL interface, in which the flight controller's sensor measurements are overridden with simulated data and actuator commands are sent to the simulator. We extended this interface, to connect with ROS, and as such our Gazebo simulation. We also wrote additional software that interfaces with the flight controller directly (via USB or Networking) and allows HITL/SITL without the GCS. Its main purpose, however, is to send sensor and state estimate data to ROS components running on board and in flight. In turn, ROS can send the flight controller guidance set points. This facilitates high level autonomous control, such as vision or perception-based algorithms, through ROS. Of course, these components are useful beyond their application for airships.

4.3.4 Robotic Hardware

We equipped a commercial 5m blimp (Figs. 4.2,4.5,4.6) (CloudMedia Sopot, Sopot, Poland), designed for advertising and aerial photography, with an Openpilot Revolution flight controller (including an IMU, 3 axis magnetometer, barometer, and a processor), a GPS receiver, a digital airspeed sensor, and an onboard computer for data logging and wireless networking. All sensors and the flight controller have been installed on the top vertical tail fin, to avoid electromagnetic interference. The main computer, an NVIDIA Jetson TX2, is installed in the gondola and connected with a lightweight USB cable. Telemetry is relayed to the ground via 5 GHz Wi-Fi using the main computer's inbuilt transceiver. The blimp can be flown manually using a Graupner flight control transmitter (TX), a matching receiver is connected to the flight controller. Autonomous flight modes are engaged using switches on the TX. The blimp has 8 independently controllable actuators: i) Bottom fin lateral thruster (propeller with forward and reverse thrust),

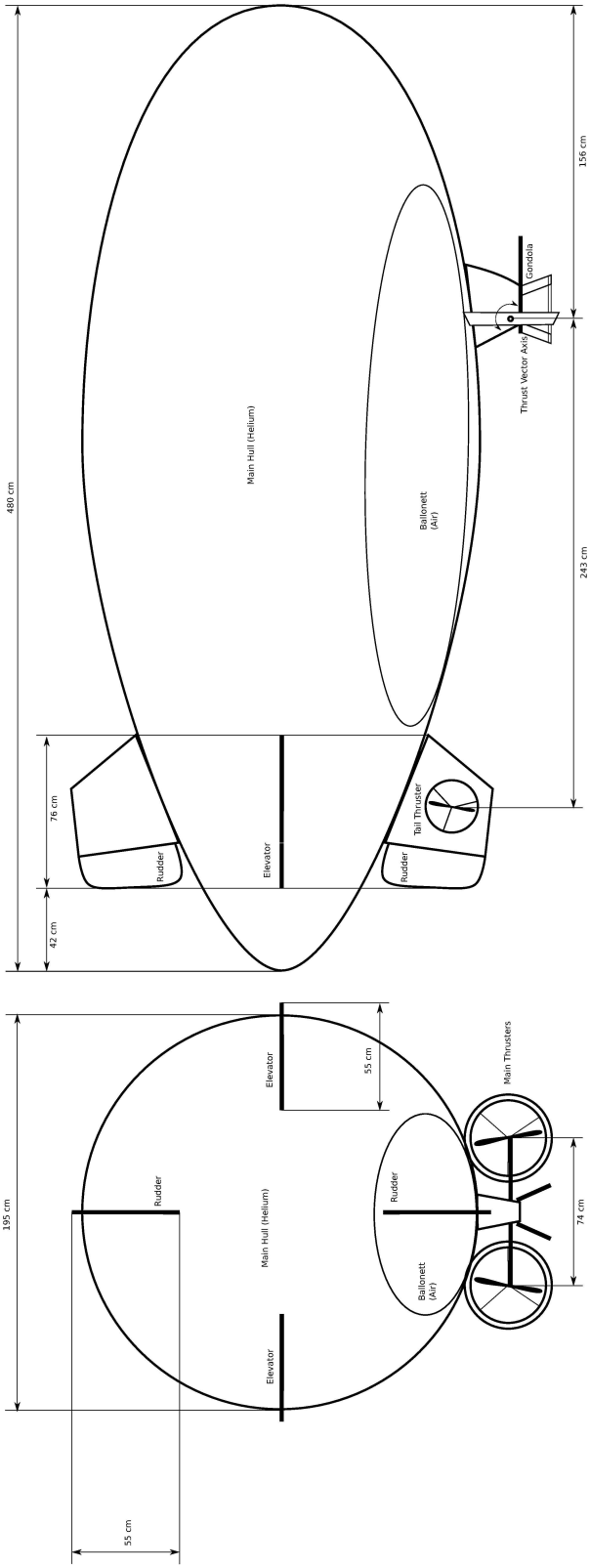


Figure 4.5: Our blimp prototype, physical layout and dimensions.

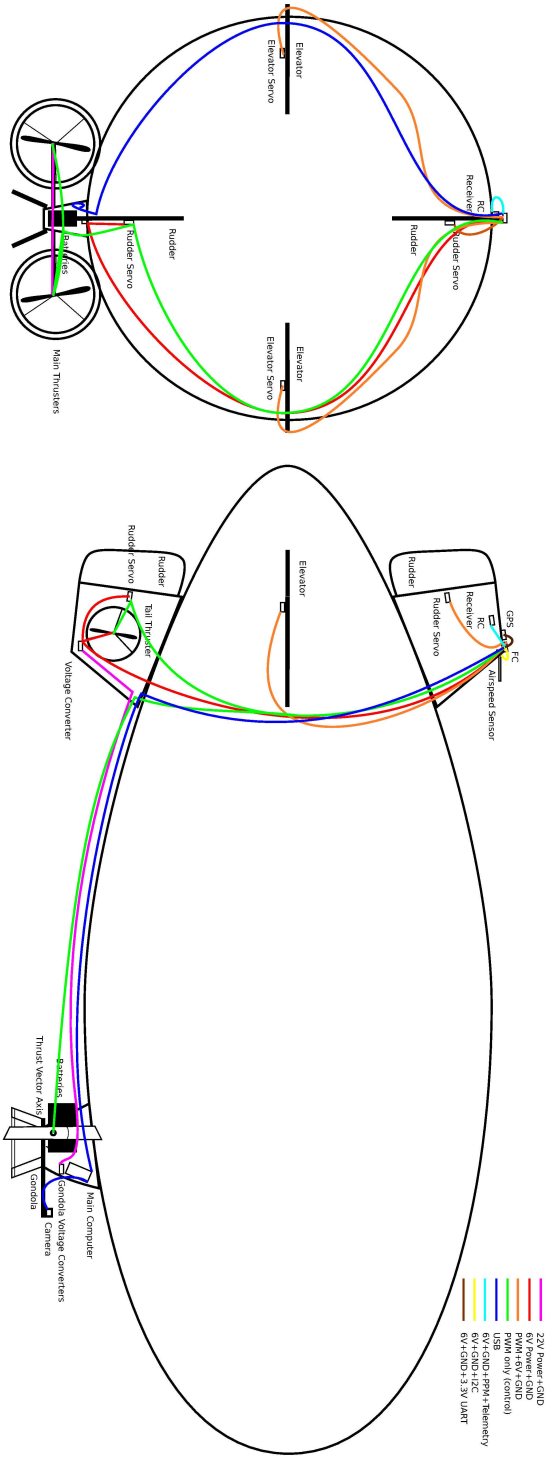


Figure 4.6: Our blimp prototype, electronics placement and wiring diagram.

ii) Top rudder (servo), iii) Bottom rudder (servo), iv) Left elevator (servo), v) Right elevator (servo), vi) Thrust vector control axis (servo), vii) Left main thruster (forward and reverse thrust), viii) Right main thruster (forward and reverse thrust). This vehicle has a mass of 10kg, including a payload or ballast of approximately 1kg. During our experiments, it became apparent that the original 2 layer hull became leaky after some time, which resulted in unsustainable Helium losses. We replaced the original polyethylene hull with a custom-made metal coated Helium retention hull by German company Windreiter. This change resulted in approx 400g additional weight savings, which allowed us to install a camera on the gondola as well as a ballonet (see Schematics, Fig 4.5). A ballonet is an air-filled bag alongside the Helium filled main gas envelope, located inside the outer envelope at the bottom of the vehicle. This second hull can be inflated or deflated to compensate for changes in air pressure and for minor adjustments in buoyancy, without any change in Helium mass.

The main difference to the multi-copter setup in Chapter 2, Section 2.4 is that we run ROS on an NVIDIA Jetson TX2. The embedded board in this case is still capable of running Neural Networks on its integrated GPU, but it also runs Ubuntu Linux and ROS. As such, we have a single main computer system instead of 2, which saves significant payload mass. The flight controller is identical. However, we do not need voltage conversion, as the airship actuators are directly compatible and controlled via analogue pulse width modulated (PWM) signals.

Physically, the main battery, engine controllers and the main computer are located in the gondola. The gondola also houses a side-mounted camera, which will be relevant for the next chapter. The blimp batteries are 6S 22V LiPos. A 12V switching voltage converter is used to power the main computer.

The flight controller, GPS receiver and Pitot probe [128] with differential pressure sensor are located on the top tail fin. This arrangement places the flight controller close to the control surfaces, and far from sources of electromagnetic interference. This arrangement, however, requires significant wire distance to connect the flight controller to the main computer.

For this, a custom 3 wire cable connects the flight controller to the USB bus. The USB-2 standard technically requires 4 wires, Ground, Data+, Data- and +5V. In our case, with modifications to the flight controller USB driver to not rely on the 5V signal for connection verification, the 5V line can be skipped. Instead, we route a high voltage 22V line to the tail assembly, which reduces current draw and wire thickness. On the tail, a 6V switching power regulator connected to this 22V bus supplies servos, the tail thruster and the flight controller. 3 PWM signal wires are routed from the flight controller back to the gondola to control the thrust vector servo and the two main thruster engine controllers.

An additional external GND connection had to be made between the flight controller USB port shield and the GND bus connection at the tail to avoid GND current flow through the flight controller while ensuring common GND level despite the long lines and potential high load. This is needed to avoid damage to the USB logic on both sides.

This means a total of 8 signal wires connect the tail with the gondola:

- 22V+
- Main GND
- Signal Left Thruster PWM
- Signal Right Thruster PWM
- Signal Thrust Vector Control PWM
- Data+ USB
- Data- USB
- GND USB

In addition, the 4 tail rudder servos are connected directly to the flight controller on the top fin for signal and power.

For the Pitot tube [128], a Sensirion SDP33 differential pressure sensor was connected to the flight controller, communicating via I2C [75]

A Graupner flight control receiver on the tail allows for manual flight control, identical to those used on the octo-copters. Manual control of the airship involves direct control of horizontal and vertical rudders, left and right thrusters coupled in a thrust channel, and an auxiliary channel for manual thrust vector control.

4.3.5 HITL

In our HITL setup, all flight control computation is done on board the embedded flight computer on robotic hardware. Servos are physically actuated, while thrusters are left unpowered for safety reasons. The value of each actuation channel is also sent via USB telemetry to the Gazebo simulator and simultaneously applied to virtual actuators in the simulation. Simulated sensor data generated by the simulation is sent back over the same telemetry link and supersedes all on board sensors, including IMU and GPS. The measured time delay induced by sending sensor data over USB is less than 0.5ms and therefore negligible. The physical sensors on the flight controller are read out at a fixed rate of 500Hz. For accurate HITL simulation, this rate must be matched by the simulator's physics engine for the simulated sensors. With Gazebo, this is possible on sufficiently fast hardware.

4.4 Experiments and Results

We conducted several experiments, both in simulation and in real-world validation.

4.4.1 Simulation experiments

Simulation experiment 1 — manual simulated flight

— Based on data from physical measurements and observations of a tethered real-world test flight[2], the simulation model is configured and tested in manual simulated flight. Unknown coefficients are adjusted until simulation and reality are in qualitative agreement[3].

Simulation experiment 2 — loitering

— The control algorithm is implemented, and control parameters are tuned in simulation, starting with the lowest level control loop. Given manual set point inputs, P and I rate control coefficients are determined for pitch and yaw rate, followed by P coefficient for the pitch control loop. When these are satisfactory, the C_d control loop is tuned, followed by C_h, C_v and C_γ. We then follow an iterative tuning procedure, while the simulated blimp is constantly loitering around a position P_0. Given current position estimate P, the velocity set point is calculated as

$$\overline{v}_S = k\,(P_0 - P)\ . \tag{4.16}$$

With the simulated blimp loitering, the performance in each controlled domain is monitored and control coefficients for whichever parameter (speed, altitude, course) is deemed worse is adjusted until satisfactory results are achieved (Figure 4.7).

Simulation experiment 3 — trajectory following

— We employ the path planner in Librepilot to follow a waypoint sequence with 4 waypoints. We denote $\hat{p} = \frac{\overline{p}}{|\overline{p}|}$ as the notation for a normalized vector. For any pair of consecutive waypoints P_0, P_1 and desired velocity v_0, we determine $\overline{v}_S$ via

$$\overline{p}_a = P - P_0\ , \tag{4.17}$$

$$\overline{p}_b = P_1 - P_0\ , \tag{4.18}$$

$$\overline{v}_S = v_0\hat{p}_b + k\,(\overline{p}_b\,(\overline{p}_a \cdot \hat{p}_b) - \overline{p}_a)\ . \tag{4.19}$$

In equation (4.19), the term $v_0\hat{p}_b$ moves the vehicle along the desired path vector, while the second term corrects any orthogonal deviance from the path, as $\overline{p}_b\,(\overline{p}_a \cdot \hat{p}_b)$ projects $\overline{p}_a$ on $\overline{p}_b$. This guidance scheme allows observations of the control behaviour in steady state, while the vehicle follows a reproducible trajectory repeatedly (Figure 4.8).

[2] Video, test flight: https://youtu.be/ZTCQNS4tmqo

[3] Video, simulation experiment 1: https://youtu.be/LNybtK3HNp8

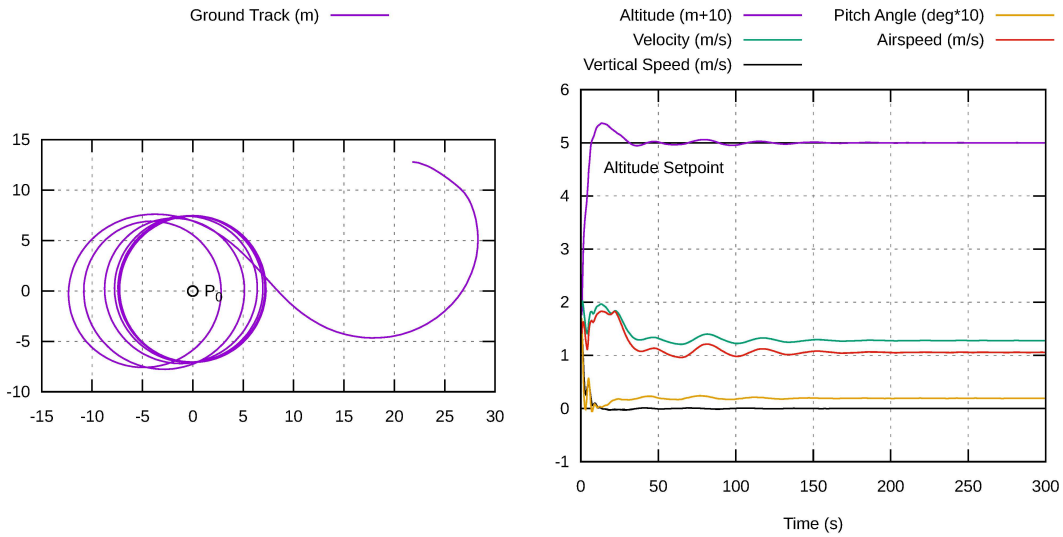


Figure 4.7: Simulation experiment 2 — The simulated blimp loiters around a given position. Without wind, the steady state becomes a circle at airspeed $v_{min} = 1\text{m/s}$ and at the minimum possible turn radius. The turn rate throughout all experiments is limited to $\pm 10^{\circ}/\text{s}$.

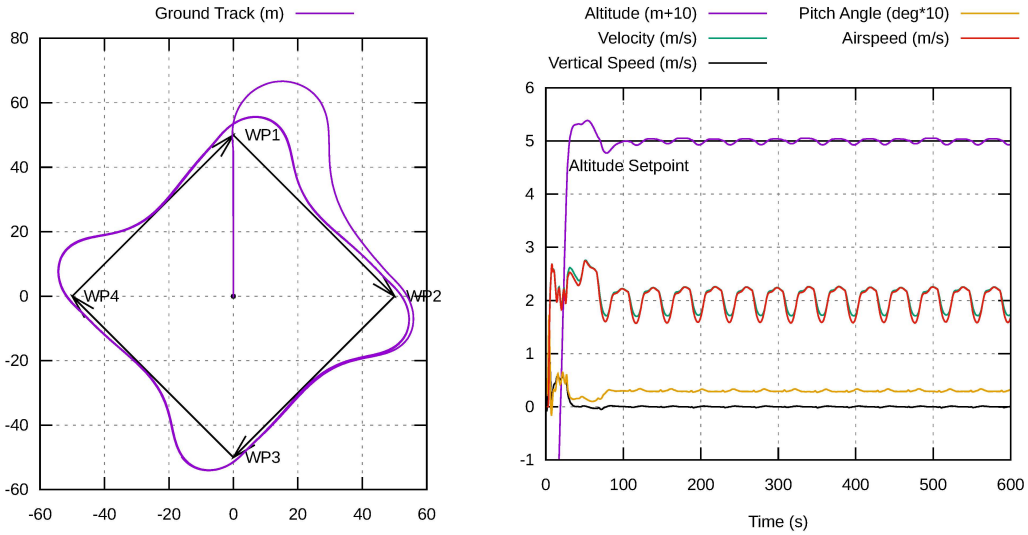


Figure 4.8: Simulation experiment 3 — The simulated blimp follows a waypoint sequence at a commanded speed of 2m/s.

Simulation experiment 4 — wind and turbulence

— We add wind and turbulence to situations from both simulation experiments 2 and 3 and observe the effects. Wind velocities between v_{min} and v_{max} allow the simulated blimp to hover on the spot with good accuracy. We note altitude excursions of several meters due to vertical wind gust components. During the trajectory following, wind from the rear poses a challenge if v_{min} is set too low, as the blimp is almost stationary in relation to the air-stream. Consequently, the control surfaces have very limited authority. This leads to a noticeable overshoot of waypoints due to the time it takes for the blimp to reorient. This can be alleviated by commanding a higher v_{min} (Figure 4.9).

Simulation experiment 5 — deflation

— We re-conduct both simulation experiments 2 and 3. During the experiment, we adjust the rigidity of the simulated blimp to simulate Helium loss and observe the effects. Simulated loss of hull pressure and the resulting decrease in rigidity reduces the controllability of the blimp. A reduction in altitude accuracy and oscillations in both pitch, altitude and also in course are observed. The ability of the fins to flex and twist in response to control actuation not only adds additional oscillation modes, especially for the top fin which houses the IMU (both in simulation and on the real-world blimp), it also reduces the control effects, since the fins rotate in response to the control forces until a new force equilibrium is reached. This significantly reduces the overall control force enacted on the hull, since the angle of attack of the fin and attached rudder negate each other. Nevertheless, the blimp remains controllable, as long as sufficient buoyancy is present to maintain altitude (Figure 4.10).

4.4.2 Real-world experiment

Loitering outdoor with wind and deflation

— We configured the real-world blimp with the control coefficients determined in simulation and engaged position hold mode. Due to a previous ground test incident, the hull had developed significant leakage. As such, we were able to reproduce the effects in simulation experiment 5 in the real-world, with the blimp's internal pressure far below optimum and dropping. Although we were able to maintain buoyancy by removing ballast weights, the blimp's flight performance was below optimal. As predicted in simulation experiment 5, a high-frequency oscillation developed in the yaw axis. The rudder actuation imparted momentum and flexion on the top fin, which was picked up by the IMU gyroscope. The real-world blimp was flown with identical control coefficients to those used in simulation experiments. However, due to changes on the electronic speed controller (ESC), the blimp produced higher thrust than anticipated. This might have contributed to oscillations in pitch and altitude, which were similar, but more persistent and of higher frequency than those observed in simulation experiment 5. Our airspeed

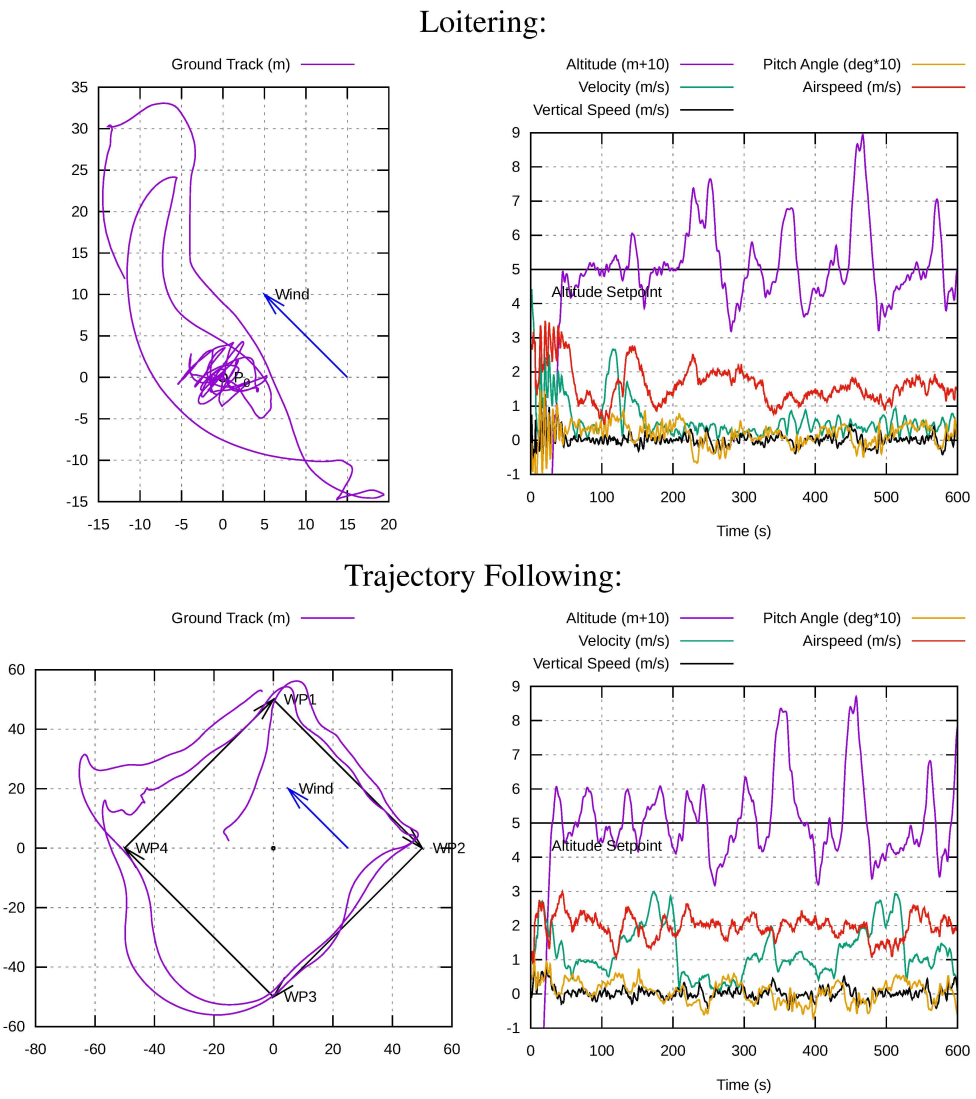


Figure 4.9: Simulation experiment 4 — Navigating in the wind poses a challenge for the simulated blimp, as the simple action of turning around after an overshoot leads to significant drift downwind. However, with the wind speed between $v_{min} = 1$m/s and $v_{max} = 2$m/s, the blimp manages to hover in close proximity to the reference point. The "random walk" is an effect of turbulent fluctuations in wind speed and direction. When navigating a waypoint sequence, the ground speed is severely increased on the downwind leg, even at an airspeed of v_{min}, resulting in large turn radii and overshoot. On the upwind leg, the ground speed is greatly reduced, coming to a near halt during strong gusts. The gusty wind poses challenges to the blimp's ability to maintain altitude, vertical gust components cause severe altitude excursions of several meters which the blimp then compensates to the best of its abilities. These are reproducible regardless of the guidance mode, leading to similar excursions at the same time in each experiment, whenever the same wind gusts are encountered. The simulated wind is coming from south-east (135°) with an average velocity of 1.5m/s, simulated turbulence for the Dryden model was set to magnitude 3.

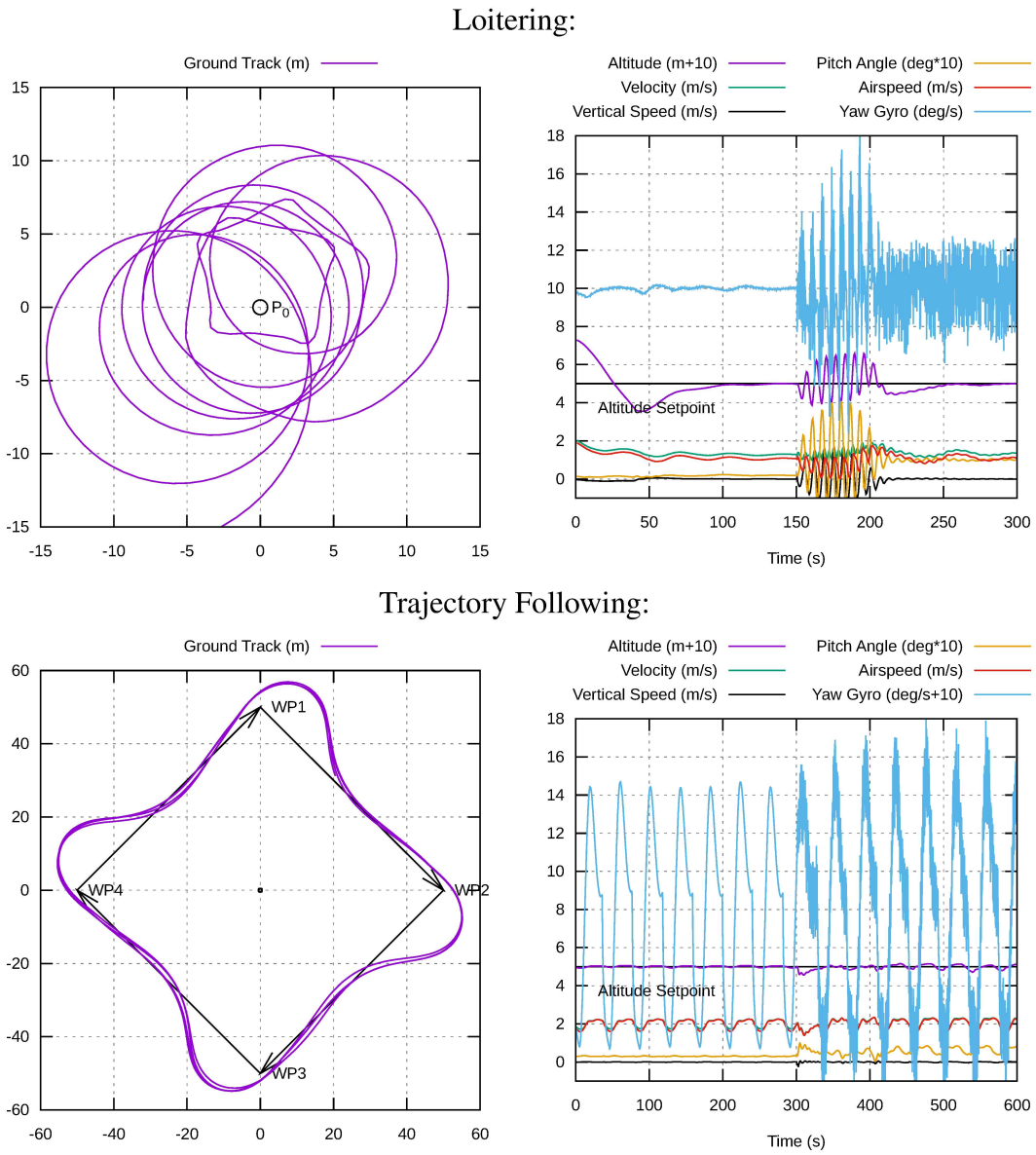


Figure 4.10: Simulation experiment 5 — The simulated blimp was partially deflated at the half-time mark of each experiment ("deflate_blimp.sh /blimp 8 0 0.2 .95"). In the loiter experiment, this exposes a severe oscillation instability in altitude and pitch that was not present when fully inflated, although the blimp can still control itself. In both situations, the altitude accuracy is noticeably decreased. In this plot, the yaw gyroscope signal is plotted, revealing a high-frequency oscillation in yaw, as the actuation of the rudder now moves the fin that the simulated gyroscope is mounted to, relative to the blimp.

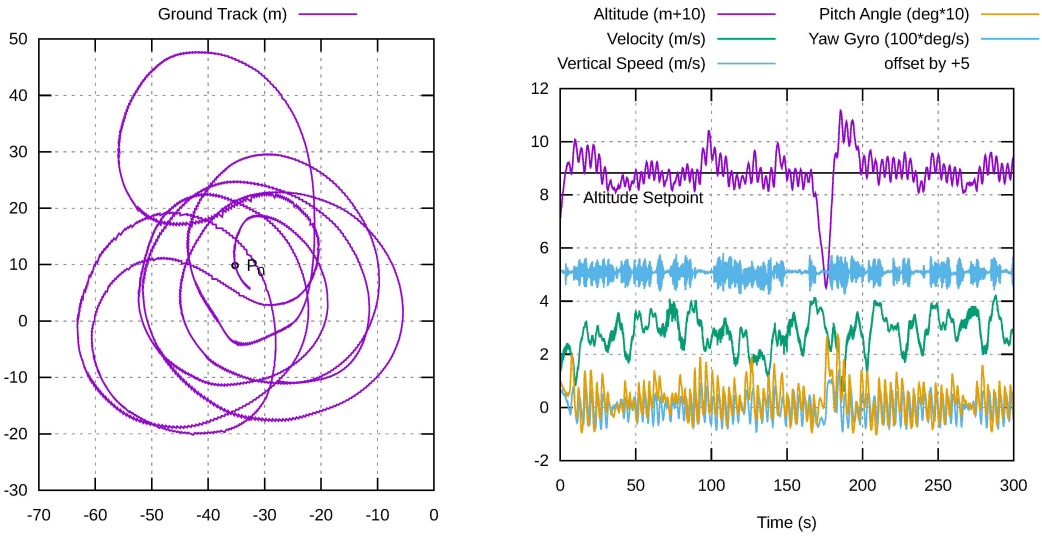


Figure 4.11: Real-world experiment- The blimp loiters around a given position in the real world, while losing Helium. The winds were benign, but a single gust at the 160s mark resulted in a noticeable drop in altitude which the control subsequently corrected, similar to those seen in simulation experiment 4. Due to lack of functional airspeed sensor and higher than tuned-for thrust, the ground speed and turn radius are larger than in the simulation. As predicted in simulation experiment 5, the blimp displays pitch-altitude oscillations throughout the flight. A high-frequency oscillation in the yaw axis was also present (drawn down-scaled by factor 100), as predicted.

sensor failed to work and had to be deactivated, leading to v_I estimated based on ground speed, but we expect minor impact due to the benign winds that day ($<$ 1m/s average). The altitude and position were controlled reliably for more than 5 minutes of continuous flight time. One observed wind gust caused an altitude excursion, as predicted by simulation experiment 4. No intervention by the pilot was required at any time (Figure 4.11)[4].

4.4.3 Discussion

Our experiments have confirmed that changes in rigidity, as simulated in our framework, have significant effects on behaviour and control of airships, which is well documented by the literature [119]. Modelling of wind gusts has also exposed notable control behaviour, not seen in non-turbulent air. We have shown that a simulation framework as presented here can be used to develop, tune and evaluate a control algorithm with results comparable to and representative of the real world, with limitations discussed below.

[4]Video, real-world experiment: https://youtu.be/KF4Dni7-Jnw

4.4.4 Limitations

Very large airships would benefit from wind and turbulence simulation, that accurately models local variations of the wind flow at the same time. This would require the wind model to be modified, to estimate a whole wind field and calculate the wind vector as a function of both space and time for multiple places. Due to our modular aerodynamics approach, it would be relatively straightforward to incorporate this data into a simulation with a sufficiently segmented hull, although this is beyond the scope of this chapter.

Our current simulation is not quantitatively accurate for our vehicle, and we cannot accurately estimate the simulation to real-world discrepancy. To achieve quantitative simulation accuracy, or to analyse the simulation error made by the model, it would be necessary to compare the calculated forces to those determined in wind tunnel tests or CFD analysis. The latter could be done with acceptable effort for the rigid case only. Based on this analysis, "best fitting" coefficients on all component objects could be determined, to minimize the simulation error. This would also show, if additional correction coefficients are needed to model aerodynamic interaction between different components, or if the current set of coefficients in combination with higher granularity is sufficient. Lacking ground truth aerodynamic data, we have to defer this to future work.

Our current aerodynamic model cannot model "prop wash" and the effects it has on impinged control surfaces and fins. This should be addressed by future work to improve the blimp simulation, although solving this would also be beneficial for other vehicles such as multi-copters and fixed wing.

Although our model can simulate deformations of the hull as a whole, it cannot model surface deformations of the skin, such as wrinkling or denting. Temporary changes of the geometric shape of a component could have significant effects on the drag coefficient, as mentioned in [119]. These have been observed to occur in our real-world experiment. Although we could dynamically adjust the aerodynamic coefficients of component objects in the simulation, this would require knowledge of the relationship between aerodynamic forces, internal pressure, shape, and effects on aerodynamics. This is a hard problem that has not been well researched yet, as noted in [119]. It justifies future work.

4.5 Conclusion and Outlook

We have created a solution for SITL and HITL simulation as well as a control algorithm, which allows researchers to set up and fly LTAVs with the same tools and flight electronics that have been used for years with fixed wing and multi-copters. The seamless integration with ROS and Gazebo incorporates airship support into popular robotic tool chains. As such, this chapter should facilitate a significant reduction of the gap between heavier- and lighter-than-air UAV robotics. Although LTAVs can be cumbersome to work with due to size and lifting gas handling, turning them into autonomous research platforms is now straightforward. Researchers can attach a COTS flight controller and

run freely available open-source tools on any flight hardware. This had previously been possible only for heavier-than-air craft.

We have shown that modelling both wind turbulence and deformation is crucial for realistic simulation of LTAVs. Rigid models in non-turbulent air cannot predict important aspects of the vehicle's behaviour.

We presented a reproducible approach for modelling non-rigidity in simulation, in combination with wind turbulence. This allows researchers to test, evaluate, or, in the case of learning algorithms, to "train" control algorithms under these conditions. The simulation environment is well suited for multi-vehicle experiments and allows research on mixed formations involving both ground robots, heavier-than-air and LTAVs. Although our current simulation models a specific blimp, it is straightforward to modify the configuration for other LTAVs. Only the robot XML/URDF configuration has to be changed.

The author considers this chapter to be the foundation for his future lighter-than-air research, and hopes the scientific community will find it useful as well.

4.6 Publication

The work in this dissertation chapter has been submitted, peer reviewed and presented at the 16th International Conference on Intelligent Autonomous Systems in 2021 [15]. The proceedings have been published in Lecture Notes in Networks and Systems, vol 412. in 2022 [15] under the title "Simulation and control of deformable autonomous airships in turbulent wind".

The work in this chapter facilitated significant follow-on research, including Chapter 5, Chapter 6 and Liu et al. [40].

Chapter 5

MPC-based Airship Formation Control

This chapter closes the overall sense-think-act control loop introduced in Chapter 1, Figure 1.3. While Chapter 2 covers detection and tracking of subjects as well as distributed state estimation and Chapter 4 models the motion and control of individual airships, this chapter ensures a formation is maintained in such a way that all cameras maintain line of sight with the subject.

As a corollary of Chapter 2 we can assume here, that the subject position and trajectory is already known, as long as all robots, i.e. airships, fly in such a way that the subject is maintained within the camera field of view at a close enough distance.

The key insight in this chapter is that both relative orientation w.r.t. the subject and relative angle between non-holonomic flying vehicles (i.e. airships) w.r.t. the subject can be maintained even if the subject is moving. Enforcing both constraints and forward motion, with a side-facing camera, naturally results in orbiting trajectories with variable radius. These can both be described analytically, with a few simplifying assumptions, and numerically when implemented with a nonlinear model predictive formation controller.

A second insight is that, ignoring the ground and visual perception, if we just model the relative motion, subject motion and wind are equivalent. The airships move relative to the airstream, and so does the subject. Whether the subject or the air or both are moving in relation to the ground or global reference frame is irrelevant w.r.t. the equations of motion, i.e. it represents nothing but a change of reference coordinate system.

Solving this problem for an assumed fixed camera yields a more general and versatile solution. For once, it allows implementation on vehicles where the camera is indeed fixed, which saves weight and reduces uncertainty in the camera orientation w.r.t. the airframe. On the other side, even if a camera is gimballed, there are sometimes gimbal limits or angles that need to be avoided due to self-occlusion. Therefore, modelling a movable camera as a fixed camera with a very wide field of view allows the application of this solution for all vehicles.

While this chapter handles airships specifically, the same fundamental principles apply to other non-holonomic vehicles such as fixed wing aeroplanes, submarines and ground vehicles, either using the same parametric formulas or with a few additional constraints. Although this has not been shown in experiment, the approach should therefore general-

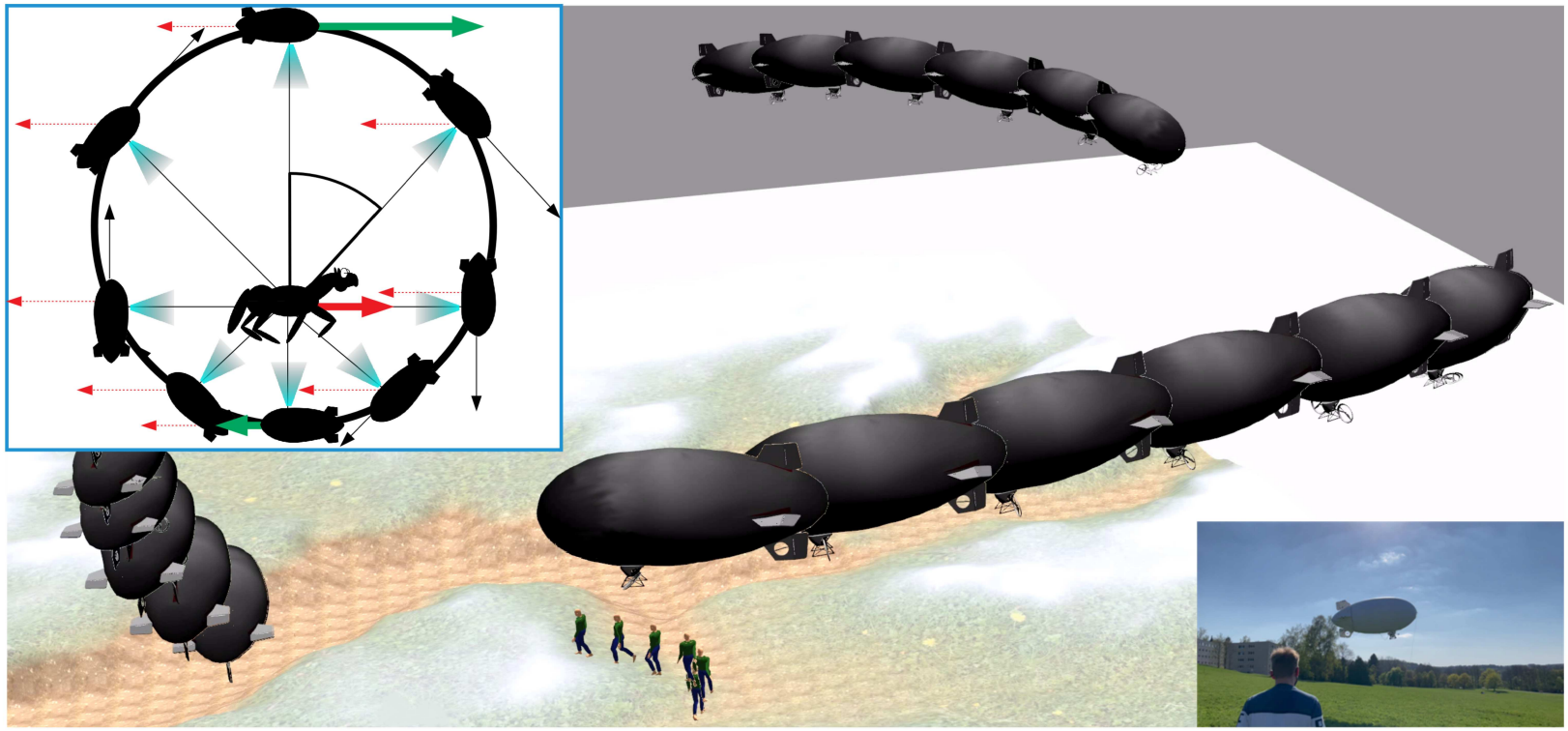


Figure 5.1: Multi-exposure visualization of 3 simulated airships in a formation, tracking a moving person. Inset Left: top-view illustration of an 8-blimp formation orbiting a subject. Red arrows: relative motion w.r.t. subject due to subject motion, resulting in periodic changes in orbit radius. Black arrows: airspeed. Green arrows: the airspeeds at the furthest and closest points, increased or decreased, respectively, by two times the subject velocity. Inset Right: Our real blimp during an experiment.

ize to any kind of non-holonomic vehicle.

5.1 Introduction

Aerial motion capture (MoCap) of human and animal subjects, i.e., estimating the trajectory of their 3D positions, skeletal poses and body shape, using a team of aerial vehicles with on-board cameras, is a challenging task in various monitoring applications. For optimal aerial MoCap, in our previous work, we developed an MPC-based [22] and a reinforcement learning-based [76] formation control method for multi-copters. For human pose and shape estimation from multiple aerial images, our previous works include [41] and [20]. In these works, we have shown that observation of an articulated subject, like a human, from multiple UAVs avoids self-occlusions in the subject and improves the accuracy of the MoCap estimates compared to single view or single camera observation. In [22] and [76], we also established that optimal MoCap estimates can be achieved if i) the subject remains centred in the camera images of all UAVs, ii) the UAVs maintain a threshold distance to the subject within safety limits, and iii) the UAVs maintain a formation-size-specific angular formation around the subject.

While these formation properties are agnostic to the vehicle-type, the controllers presented in both [22] and [20] assume multi-copter UAVs. These are not well-suited for some subjects, such as animals in their natural habitat, due to their loud noise, short flight time (battery) and safety-related concerns. Buoyant, Helium-based, lighter than air vehicles (or airships) [15, 40] can address these concerns. These are relatively safe in terms of collision-related damage. Compared to multi-copters capable of carrying similar payloads, airships can have significantly longer flight times and considerably lower noise levels.

Unfortunately, airship motion and control is fundamentally non-holonomic in nature and substantially different from that of the multi-copters. Therefore, the solutions developed in [22] and [20] are not directly applicable. Most airship designs need to remain in motion to be controllable. Their orientation and flight direction are also tightly coupled. For example, to climb, an airship typically needs to pitch up, which rotates the camera view around its lateral Y-axis, assuming a rigidly mounted camera. A directional change, on the other hand, triggers a roll around the airship's longitudinal X-axis. Most importantly, classical airship designs allow movement only in the forward direction through the surrounding air, offset by a lateral angle of attack. While the global camera orientation could be decoupled from flight direction using gimbal-mounted cameras, these add additional weight and control complexity. Therefore, a solution is required for airship formation control for the task of aerial MoCap that i) does not need camera gimbals, and ii) adheres to the previously mentioned optimal MoCap conditions. This is the core problem addressed in this chapter.

A possible solution for a stationary subject and a single airship, is to mount the camera perpendicular to the forward direction of the airship, tilted downward at an angle, and

let the airship orbit around the subject. For a formation of airships, maintaining the same angular velocity by all airships would be a solution. However, for subjects moving with arbitrary and changing velocities, these solutions are not directly applicable. The key contribution of this chapter is that we show, both theoretically and empirically, that subject-orbiting trajectories of airships can be generalized even to subjects in motion, while maintaining a perpendicular line-of-sight (LoS) to the subject and a prescribed angular separation between them. We achieve this by exploiting a periodic relationship between the airspeed of the airship and its distance to the subject. We also show how our method copes with wind, as both wind and subject motion result in equivalent relative velocity between the airship and the subject. Our further novel contributions include the following.

- We derive analytical solutions for a simplified 2D case (airships and the subject moving on the same plane) with simplified aerodynamics. From this, we derive boundary conditions on the subject velocity that allow possible orbit solutions under the constraints of maximum and minimum airship speeds.
- We then derive an approximate analytical solution in the full 3D case with realistic physics. Based on this, and a MoCap cost function, we introduce a numeric solver for the formations. Utilizing this solver, we implement a fixed-time model predictive controller (MPC) for the formation control problem.

We evaluate our MPC in simulation experiments (Figure 5.1) by varying parameters such as wind, subject motion, and the number of vehicles in the formation. Finally, we demonstrate it on our real airship. We opened the source code for the benefit of the community.

5.2 State of the Art

MPC has been extensively used for drone formation control [22, 129, 130, 131]. Most works on airship formation control, with [132] or without [133] MPC, cover traditional formation types with pre-specified geometry. However, none considers centring a subject in the on-board camera's image or angular separation constraints w.r.t. a tracked subject.

Since the non-holonomic motion constraints of airships and fixed-wing aircraft are similar enough to allow cross-application, we briefly overview some works in that field. Subject-relative formations have been well studied for observation with fixed-wings, but primarily with straight downward facing cameras, where the vehicles loiter high above the subject [134, 135, 136, 137]. This greatly simplifies the formation control problem, while simultaneously limiting the observation angles to a top-down bird's-eye view, which is not well suited for the MoCap task. In contrast, our approach can be employed for arbitrary observation angles and for subjects at the same altitude or even higher than the airship.

Orbiting trajectories for fixed-wing UAVs around a subject have been described in [138], where a Lyapunov-function-based vector field is designed to explicitly converge on an orbiting solution in a constant altitude plane. It maintains angular separation between vehicles and constant distance to the subject, but it does not consider vehicle orientation and camera viewing angles relative to the subject. The Lyapunov-function-based approach is sensitive to starting conditions and cannot easily be extended to maintain additional constraints. Our optimization-based approach not only converges on steady state solutions, but also optimizes the transition phases for optimal subject coverage, considers motion in 3D space and can enforce constraints on both state and control inputs. Other works [139, 140, 141] deal with subject-relative formation strategies for varying objectives, but do not consider body-frame limited viewing angles or non-holonomic motion constraints.

5.3 Methodology

5.3.1 Notations

In the world frame, the velocity of the airship $n \in [1 \dots N]$ at the time t is given as ${}^{n}\boldsymbol{v}_t = \left[{}^{n}v_{xt}, {}^{n}v_{y_t}, {}^{n}v_{z_t}\right]^\top$ and its horizontal speed is given as ${}^{n}v_{ht} = \left\| \left[{}^{n}v_{xt}, {}^{n}v_{y_t}\right] \right\|$. The wind/fluid vector is given as ${}^{F}\boldsymbol{v}$. The velocity of the subject is given as ${}^{S}\boldsymbol{v}$. The position of the airship n and subject S at the time t are given as ${}^{n}\boldsymbol{p}_t$ and ${}^{S}\boldsymbol{p}_t$, respectively. We use a left subscript to denote a reference frame different from the world frame, e.g., the velocity of airship n at the time t relative to the fluid F is given as ${}^{n}_{F}\boldsymbol{v}_t = {}^{n}\boldsymbol{v}_t - {}^{F}\boldsymbol{v}$ and the position of airship n relative to the subject S is given as ${}^{n}_{S}\boldsymbol{p}_t = {}^{n}\boldsymbol{p}_t - {}^{S}\boldsymbol{p}_t$ (Figure 5.2a).

All positions and velocities are oriented North-East-Down, unless specified otherwise. In airship body frame, we consider x forward, y rightward and z downward. The orientation of the airship n at the time t is given as ${}^{n}\boldsymbol{\Theta}_t = [{}^{n}\varphi_t, {}^{n}\theta_t, {}^{n}\psi_t]^\top$, with components roll ${}^{n}\varphi_t$, pitch ${}^{n}\theta_t$ and yaw ${}^{n}\psi_t$ as defined in [142, p. 147]. The horizontal motion direction of the airship n in the fluid is given as

$$ {}^{n}_{F}\chi_t = \operatorname{atan2}\left(\frac{{}^{n}_{F}v_{y_t}}{{}^{n}_{F}v_{xt}} \right). \tag{5.1} $$

We write ${}^{n}\dot{\psi}_t$ for the yaw rate and ${}^{n}_{F}\dot{\chi}_t$ for the turn rate of the airship n at the time t. We define the formation state of N airships at the time t as

$$ \boldsymbol{X}_t = \left[{}^{S}\boldsymbol{x}_t, {}^{1}\boldsymbol{x}_t, \dots, {}^{N}\boldsymbol{x}_t \right]^\top, \tag{5.2} $$

consisting of subject state ${}^{S}\boldsymbol{x}_t = \left[{}^{S}\boldsymbol{p}_t \right]$ and airship states ${}^{n}\boldsymbol{x}_t = [{}^{n}\boldsymbol{p}_t, {}^{n}\boldsymbol{v}_t, {}^{n}\boldsymbol{\Theta}_t]^\top$ for all airships $n \in [1 \dots N]$.

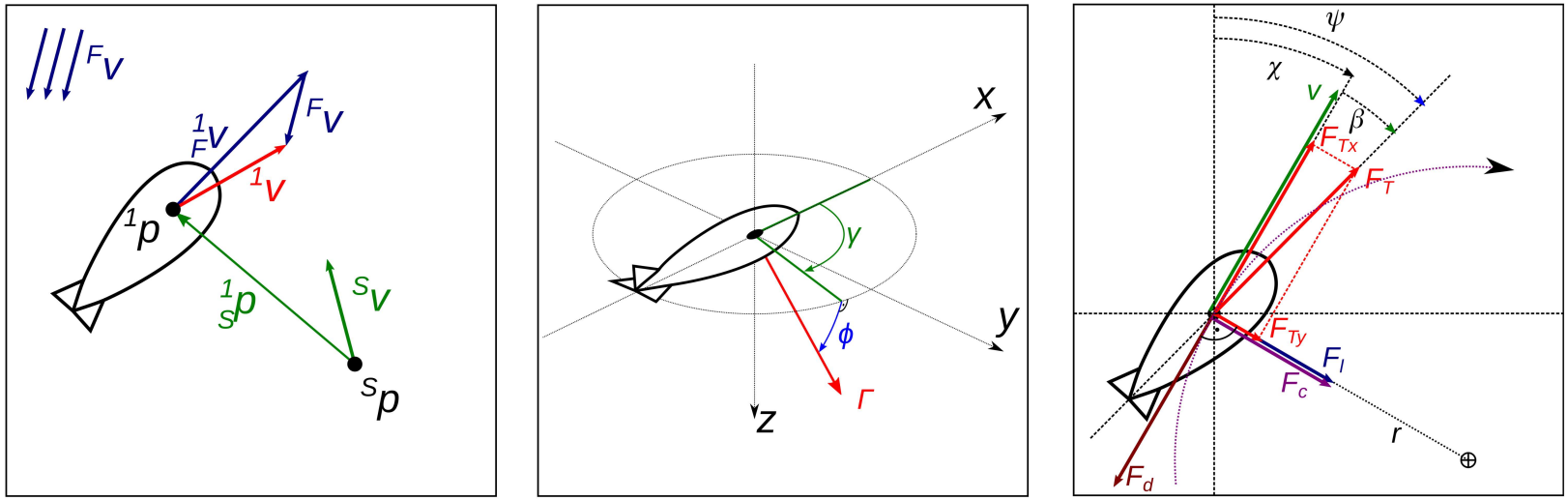


Figure 5.2: a) Absolute and relative position and velocity of airship 1 near subject *S*. b) Camera angle in the blimp body frame. c) Forces on an airship on a curved trajectory, subject to aerodynamic lift and drag.

5.3.2 Problem Statement

Let us assume a team of N airships, each with a body-fixed camera, tracking a moving subject S. Let us also assume that the camera's fixed azimuth and elevation angles w.r.t. the carrier airship are given as γ and ϕ, respectively, and represented jointly as $\mathbf{\Gamma} = [\gamma, \phi]$. Our goal is to control the airship's acceleration ${}^{n}_{F}\dot{v}$ and yaw rate ${}^{n}\dot{\psi}_t$, such that the following conditions are met.

1. S remains centred in the camera view for all $n \in [1 \dots N]$, a MoCap accuracy-specific requirement [76].
2. The angles subtended by any two airships n, m, w.r.t. the subject S, remain equal to a defined preset that is optimal for minimizing the joint state estimation uncertainty. In [22], it was shown that an angle of $\frac{2\pi}{N}$ for $N > 2$ and $\frac{\pi}{2}$ for $N = 2$ is optimal.
3. Other input and state constraints: we assume that to remain manoeuvrable, each airship must fly above a minimum airspeed $v_{\min}$ and because of drag and finite thrust, cannot exceed $v_{\max}$ such that

$$v_{\min} \leq \| {}^{n}_{F}\boldsymbol{v}_t \| \leq v_{\max}. \tag{5.3}$$

5.3.3 Planar Airship Orbits in 2D

For our primary analysis, we consider a system with the following simplifications: i) movement and rotation in a 2D X/Y plane, ii) movement only in the heading direction (${}^{n}\psi_t = {}^{n}_{F}\chi_t$). If the airship n moves at velocity $\| {}^{n}_{F}\boldsymbol{v}_t \| > v_{\min}$, while the subject is not moving relative to the fluid ($\left\| {}^{S}_{F}\boldsymbol{v} \right\| = 0$), then the only stable solution to keep the subject centred in camera view is i) to mount the camera perpendicular to the motion direction with $\gamma = \pm\frac{\pi}{2}$ and ii) to orbit on a curved, tangential path with constant radius

$${}^{n}_{S}r_t = \| {}^{n}_{S}\boldsymbol{p}_t \|. \tag{5.4}$$

This satisfies the equation of circular motion

$${}^{n}_{S}r_t = \frac{\| {}^{n}_{S}\boldsymbol{v}_t \|}{{}^{n}\dot{\psi}_t}, \tag{5.5}$$

with tangential speed $\| {}^{n}_{S}\boldsymbol{v}_t \|$ and rotation rate ${}^{n}\dot{\psi}_t$. Since ${}^{n}\dot{\psi}_t$ can be controlled, this can be done for arbitrary radii regardless of velocity constraints.

If the subject is moving ($\left\| {}^{S}_{F}\boldsymbol{v} \right\| \neq 0$) (Figure 5.1 inset), it is still possible to orbit around S while keeping it centred in the camera view. To illustrate this intuitively, we mirror and rotate the system until, without limiting generality, the subject is moving along the negative X axis with ${}^{n}\dot{\psi}_t > 0$ and $\gamma = \frac{\pi}{2}$. To keep the subject centred in the camera, the speed

of the airship n must change to match the projected subject velocity component perpendicular to the camera axis. Adding to the airship's orbital speed ${}^{n}_{S}r_t \, {}^{n}\dot{\psi}_t$, we calculate

$$\| {}^{n}_{F}\boldsymbol{v}_t \| = {}^{n}_{S}r_t \, {}^{n}\dot{\psi}_t - \cos({}^{n}\psi_t) \left\| {}^{S}_{F}\boldsymbol{v} \right\|. \tag{5.6}$$

Since by assumption (in this subsection only) the airship can only fly in the direction it is pointing, the distance to the subject, which is the current orbit radius, will change with

$${}^{n}_{S}\dot{r}_t = \sin({}^{n}\psi_t) \left\| {}^{S}_{F}\boldsymbol{v}_t \right\|. \tag{5.7}$$

By integration, we introduce a constant r_0 as a base radius. Thus, ${}^{n}_{S}\boldsymbol{r}_t$ can be given as

$${}^{n}_{S}r_t = r_0 - \cos({}^{n}\psi_t) \frac{\left\| {}^{S}_{F}\boldsymbol{v} \right\|}{{}^{n}\dot{\psi}_t}. \tag{5.8}$$

Inserting (5.8) in (5.6), we get

$$\| {}^{n}_{F}\boldsymbol{v}_t \| = {}^{n}\dot{\psi}_t \, r_0 - 2\cos({}^{n}\psi_t) \left\| {}^{S}_{F}\boldsymbol{v} \right\|, \tag{5.9}$$

from which we calculate minimal and maximal airspeed during the orbit as

$$\min(\| {}^{n}_{F}\boldsymbol{v}_t \|) = {}^{n}\dot{\psi}_t \, r_0 - 2 \left\| {}^{S}_{F}\boldsymbol{v} \right\| \tag{5.10}$$

and

$$\max(\| {}^{n}_{F}\boldsymbol{v}_t \|) = {}^{n}\dot{\psi}_t \, r_0 + 2 \left\| {}^{S}_{F}\boldsymbol{v} \right\|, \tag{5.11}$$

for a desired yaw rate ${}^{n}\dot{\psi}_t$ and base radius r_0, i.e., the distance to the stationary subject or the average distance to a moving subject over one orbit.

Multiple airships in a formation, i.e., $N > 1$, can maintain the angular constraints (2nd constraint in Subsection 5.3.2) only when ${}^{n}\dot{\psi}_t$ is the same for all airships. For physically achievable orbits (5.3), that satisfy all the required perception constraints (Subsection 5.3.2),

$$\min(\| {}^{n}_{F}\boldsymbol{v}_t \|) \geq v_{\min} \quad \text{and} \quad \max(\| {}^{n}_{F}\boldsymbol{v}_t \|) \leq v_{\max}, \tag{5.12}$$

must hold.

If two airships m, n are opposed by π on the same orbit, at ${}^{m}\psi_t = 0$ and ${}^{n}\psi_t = \pi$, the worst-case scenario is that the velocity of airship m, given by (5.10), approaches $v_{\min}$ and the velocity of airship n, given by (5.11), approaches $v_{\max}$, both simultaneously. Solving for the subject velocity using (5.10) and (5.11) at this point leads to

$$\left\| {}^{S}_{F}\boldsymbol{v} \right\| = \frac{(v_{\max} - v_{\min})}{4}. \tag{5.13}$$

Consequently, the maximum magnitude of the subject velocity for which the whole

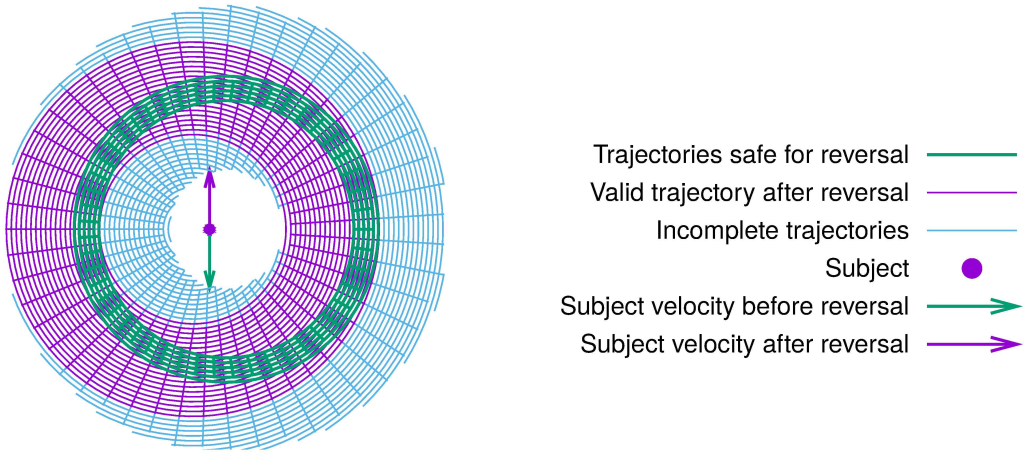


Figure 5.3: Subject velocity reversal: Orbital space around a moving subject for a fixed yaw rate and airship velocity constraints $v_{\min} \leq \|{}^{n}_{F}\boldsymbol{v}\| \leq v_{\max}$, with visualization of the effect of reversal of ${}^{S}_{F}\boldsymbol{v}$ direction. For any ${}^{n}\dot{\psi}_{t}$, the innermost valid trajectory reaches $v_{\min}$ when flying opposite to the subject, the outermost approaches $v_{\max}$ when flying parallel to the subject. A trajectory is safe under reversal (green), if every point on the trajectory lies on a closed and valid purple trajectory. Trajectories in blue cannot be followed without violating constraints (blank areas).

orbit satisfies all the required conditions (Subsection 5.3.2) is given as

$$\left\| {}^{S}_{F}\boldsymbol{v} \right\| \leq \frac{(v_{\max} - v_{\min})}{4}. \tag{5.14}$$

Feasibility under reversal

Although we consider the subject velocity as constant, we can also determine the robustness of this approach to changes in ${}^{S}_{F}\boldsymbol{v}$. For any given threshold ${}^{S}_{F}\boldsymbol{v}$, the worst-case change is the sudden reversal of the motion direction, since this inverts the offset of the trajectory relative to the subject, as shown in Figure 5.3. The airship might now be on an invalid (blue) orbit. Continuing on that trajectory would then violate (5.3). However, a family of orbits exists, for which the whole orbit including minimum and maximum, is always on a feasible trajectory after reversal. This is displayed in Figure 5.3 in green. Analogous to (5.14), it can be shown that under single reversals safe trajectories exist if

$$\left\| {}^{S}_{F}\boldsymbol{v} \right\| \leq \frac{(v_{\max} - v_{\min})}{8}. \tag{5.15}$$

In Summary, the analytic solution with the 2D assumptions for our problem statement (Subsection 5.3.2), is given by (5.8) subject to constraint (5.14).

5.3.4 Airship Orbits in 3D with Realistic Physics

A realistic airship model must take aerodynamic forces into account. The hull of an airship, as depicted in Figure 5.2c is subject to lift and drag forces [142, pp 31-59] and can be modelled as an airfoil. Below, we model the following physical parameters.

Lateral angle of attack

In a real airship, ${}^{n}\psi_t \neq {}^{n}_{F}\chi_t$ unless the airship flies in a straight line. To fly in a curve, a centripetal force F_c needs to act on the airship. This force is a combination of the lateral engine thrust vector F_{ty} and aerodynamic lift F_l, as shown in Figure 5.2c. Both force components are a function of the lateral angle of attack ${}^{n}_{F}\beta_t$, given as

$$ {}^{n}_{F}\beta_t = {}^{n}\psi_t - {}^{n}_{F}\chi_t \,. \tag{5.16} $$

The thrust component (Fig 5.2c) F_{ty} is given as

$$ F_{ty} = \tan({}^{n}_{F}\beta_t)\, F_{tx}. \tag{5.17} $$

To maintain airspeed, the thrust component F_{tx} must be equal to drag aerodynamic F_d with

$$ F_{tx} = F_d = c_1 A \, \| {}^{n}_{F}\boldsymbol{v}_t \|^2 \tag{5.18} $$

where c_1 is the vehicle-specific drag coefficient and A is the relevant area. On the other hand, aerodynamic lift for angle of attack ${}^{n}_{F}\beta_t < \beta_{\text{stall}}$ [142, p. 44, Figure 23] is

$$ F_l = c_2 A \, {}^{n}_{F}\beta_t \, \| {}^{n}_{F}\boldsymbol{v}_t \|^2 \,. \tag{5.19} $$

with aerodynamic lift coefficient c_2. Since for small angles ${}^{n}_{F}\beta_t < \beta_{\text{stall}}$, ${}^{n}_{F}\beta_t \approx \tan {}^{n}_{F}\beta_t$, we can combine both forces, divide by vehicle mass based on generic acceleration $a = \frac{F}{m}$ and approximate

$$ {}^{n}\boldsymbol{a}_{ct} \approx {}^{n}c_l \, \| {}^{n}_{F}\boldsymbol{v}_t \|^2 \, {}^{n}_{F}\beta_t \,. \tag{5.20} $$

where ${}^{n}c_l$ is a system-specific coefficient, representing the combined effect of the airship aerodynamic lift and drag and its mass in a single coefficient. We will later enforce small ${}^{n}_{F}\beta_t$ with control constraints $v_{\min}$ and $\dot{\psi}_{\max}$.

When the airship rotates in flight with ${}^{n}\dot{\psi}_t$, ${}^{n}_{F}\beta_t$ will steadily increase until the lateral acceleration asymptotically approaches the centripetal acceleration

$$ {}^{n}\boldsymbol{a}_{ct} \to \| {}^{n}_{F}\boldsymbol{v}_t \| \, {}^{n}\dot{\psi}_t \,. \tag{5.21} $$

At this point, the rotation rate equals the turn rate ${}^{n}_{F}\dot{\chi}_t = {}^{n}\dot{\psi}_t$ and the lateral angle of attack no longer changes, i.e. ${}^{n}_{F}\dot{\beta}_t = 0$. Thus, for a fixed yaw rate ${}^{n}\dot{\psi}_t$ and for sufficient

airspeed $\| {}^{n}_{F}\boldsymbol{v}_t \| \geq v_{\min}$ which implies ${}^{n}_{F}\beta_t < \beta_{\text{stall}}$, we can state

$$\| {}^{n}_{F}\boldsymbol{v}_t \| \, {}^{n}\dot{\psi}_t \approx {}^{n}c_l \, \| {}^{n}_{F}\boldsymbol{v}_t \|^2 \, {}^{n}_{F}\beta_t \tag{5.22}$$

and approximate

$${}^{n}_{F}\beta_t \approx \frac{{}^{n}\dot{\psi}_t}{c_l \, \| {}^{n}_{F}\boldsymbol{v}_t \|}. \tag{5.23}$$

Changes in velocity

If the airship is undergoing acceleration ${}^{n}_{F}a_s$ in the direction of motion ${}^{n}_{F}\chi_t$, on a curved trajectory with ${}^{n}\dot{\psi}_t \neq 0$ then ${}^{n}_{F}\boldsymbol{v}_t$ is changing in both direction and magnitude. We can accurately calculate changes in the magnitude

$$\| {}^{n}_{F}\boldsymbol{v}_t \| = \| {}^{n}_{F}\boldsymbol{v}_{t0} \| + t \; {}^{n}_{F}a_s \, . \tag{5.24}$$

based on the known acceleration ${}^{n}_{F}a_s$. To calculate the velocity vector ${}^{n}_{F}\boldsymbol{v}_t$, we exploit the known magnitude and estimate its direction with

$${}^{n}_{F}\boldsymbol{v}_t = \| {}^{n}_{F}\boldsymbol{v}_t \| \begin{bmatrix} \cos({}^{n}_{F}\chi_t) \\ \sin({}^{n}_{F}\chi_t) \end{bmatrix}. \tag{5.25}$$

We determine

$${}^{n}_{F}\chi_t = {}^{n}\psi_t - {}^{n}_{F}\beta_t \, , \tag{5.26}$$

as a function of ${}^{n}\dot{\psi}_t$, by plugging (5.23) into (5.26). Thereby, we obtain

$${}^{n}_{F}\chi_t \approx {}^{n}\psi_t - \frac{{}^{n}\dot{\psi}_t}{c_l \, \| {}^{n}_{F}\boldsymbol{v}_t \|} \tag{5.27}$$

respectively

$${}^{n}_{F}\boldsymbol{v}_t \approx \| {}^{n}_{F}\boldsymbol{v}_t \| \begin{bmatrix} \cos\left({}^{n}\psi_t - \frac{{}^{n}\dot{\psi}_t}{c_l \left\| {}^{n}_{F}\boldsymbol{v}_t \right\|} \right) \\ \sin\left({}^{n}\psi_t - \frac{{}^{n}\dot{\psi}_t}{c_l \left\| {}^{n}_{F}\boldsymbol{v}_t \right\|} \right) \end{bmatrix}. \tag{5.28}$$

Roll angle

Most airships are passively stable in roll and have their centre of mass underneath the centre of lift and buoyancy (Fig 5.4). Consequently, they self-orient according to the sum of gravity and centripetal acceleration. Although this steady state will not be achieved

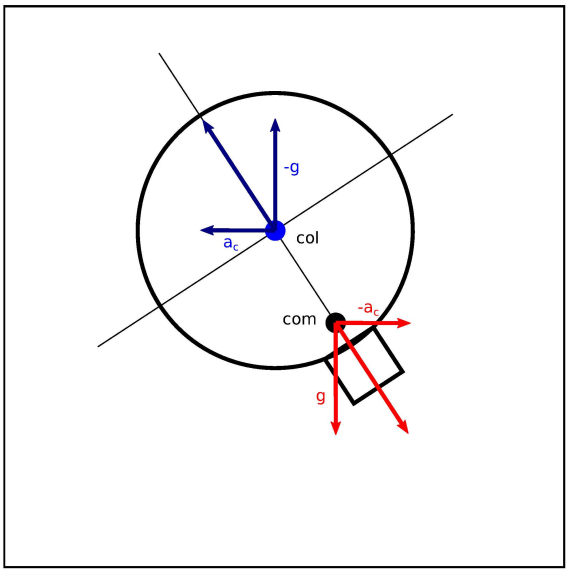


Figure 5.4: Airship roll angle with gravity, buoyancy, lateral acceleration and inertia in equilibrium.

immediately in dynamic situations, we can approximate the roll angle of the airship n at the time t with

$$ {}^{n}\varphi_t \approx \operatorname{atan}\left(\frac{{}^{n}_{F}\dot{\chi}_t \, \| {}^{n}_{F}\boldsymbol{v}_t \|}{g}\right), \tag{5.29}$$

where $g \approx 9.81 \frac{\mathrm{m}}{\mathrm{s}^2}$ is the earth's gravitational acceleration. In fixed wing aircraft, the same formula describes a *coordinated turn* [142, p. 153, Figure 63]. An important distinction is that a fixed wing aircraft typically actively assumes this angle for the most efficient turn, in which the lateral angle of attack - or slip - is minimized. The lift force both compensates gravity and provides lateral acceleration, since the aerodynamic lift to drag ratio of the wings is superior to that of the fuselage. An airship, on the other hand, typically has a rotation-symmetric envelope and the lift efficiency does not change with roll angle. The assumed roll angle is typically involuntary.

Altitude

For altitude changes, we assume that the airship can change its vertical speed arbitrarily and independently of its horizontal motion, within limits [15]. For the analytical solution, the necessary altitude $(-{}^{n}_{S}p_{zt})$ above S to maintain the subject centred in the camera view may be approximated based on radius ${}^{n}_{S}r_t$ and camera elevation ϕ (Figure 5.2b) as

$$ -{}^{n}_{S}p_{zt} \approx \frac{\tan({}^{n}\varphi_t - \phi)}{{}^{n}_{S}r_t}, \tag{5.30}$$

however, this analytical solution ignores the effect of the airships pitch angle ${}^{n}\theta_t$, which we explicitly consider in the numeric solution (Subsection 5.3.5).

Pitch angle

Airships can change their altitude in a number of ways, not all of which require the same attitude changes. Yet, assuming that the nose points in the direction of travel is a good approximation in most cases. Therefore, pitch angle ${}^{n}\theta_t$ is computed with

$$ {}^{n}\theta_t \approx \operatorname{atan}\left(\frac{-{}^{n}_{F}v_{zt}}{{}^{n}_{F}v_{ht}}\right). \tag{5.31} $$

In summary, one approximate analytical solution for airship orbits in 3D for our problem statement (Subsection 5.3.2) can be obtained by combining (5.8), (5.29) and (5.30), however the effect of changes in ${}^{n}\theta_t$ is not reflected in this analytical solution. In the numeric solution (Subsection 5.3.5), we empirically show that changes in ${}^{n}\theta_t$ make the orbits less eccentric than in (5.8).

5.3.5 Optimization Formulation for Numeric Solution

We can find an optimal trajectory for a formation starting state $\boldsymbol{X}_0$, by minimizing a cost function C given as

$$ C = \sum_{t,n} \left(k_c \, {}^{n}E_{ct} + k_f \, {}^{n}E_{f_t}\right), \tag{5.32} $$

subject to constraints

$$ v_{\min} \leq \left\| {}^{n}_{F}\boldsymbol{v}_t \right\| \leq v_{\max} \quad \text{and} \quad \left\| {}^{n}_{F}v_{zt} \right\| \leq v_{z\max}. \tag{5.33} $$

$k_c \, {}^{n}E_{ct}$ keeps the subject S centred in the camera images, while $k_f \, {}^{n}E_{f_t}$ maintains angular separation between the airships around S. k_f and k_c can be chosen to prioritize these goals during transition periods. This does not change the converged orbit, as both cost terms are near 0 for a fully converged trajectory. Here

$$ {}^{n}E_{ct} = \left\| \begin{bmatrix} k_d & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} d_c \\ 0 \\ 0 \end{bmatrix} - {}^{n}\boldsymbol{R}_t \left(-{}^{n}_{S}\boldsymbol{p}_t\right) \right) \right\|^2, \tag{5.34} $$

${}^{n}\boldsymbol{R}_t$ is the rotation matrix combining $\boldsymbol{\Gamma}$ and ${}^{n}\boldsymbol{\Theta}_t$ for a projection of S into the camera coordinate system of the airship n, k_d is a weight for the distance term, and d_c is the optimal camera distance. As shown in [22], a formation is optimal for minimizing the joint uncertainty in the state estimate when the vehicles are equally spaced for 3 or more

vehicles, but at 90° to each other for 2 vehicles. This is achieved with

$$
{}^{n}E_{f_t} = \begin{cases} \sum_{m=1, m\neq n}^{N} \left(\frac{\pi}{2} - \operatorname{acos}\left(|-{}_{S}^{n}\boldsymbol{p}_t| \cdot |-{}_{S}^{m}\boldsymbol{p}_t|\right)\right)^2 & \text{for } N = 2 \\ \sum_{m=1, m\neq n}^{N} \max\left(0, \frac{2\pi}{N} - \operatorname{acos}\left(|-{}_{S}^{n}\boldsymbol{p}_t| \cdot |-{}_{S}^{m}\boldsymbol{p}_t|\right)\right)^2 & \text{for } N \geq 3. \end{cases} \tag{5.35}
$$

For $N \geq 3$, ${}^{n}E_{f_t}$ penalizes airships that are separated by less than the desired angular distance for an evenly spaced formation $\frac{2\pi}{N}$. This has a repulsive effect and optimizes for equal spacing without enforcing any specific order.

We solve for control input $\boldsymbol{U}$ at the time t, consisting of vehicle controls ${}^{n}\boldsymbol{u}_t$, with

$$
\boldsymbol{U}_t = \left[{}^{1}\boldsymbol{u}_t, \ldots, {}^{N}\boldsymbol{u}_t\right] \quad \text{and} \quad {}^{n}\boldsymbol{u}_t = \left[{}^{n}\dot{\psi}_t, {}^{n}\dot{v}_{ht}, {}^{n}\dot{v}_{zt}\right]. \tag{5.36}
$$

We define the discrete time state transition consisting of subject state transition and state transitions for all airships, with

$$
\boldsymbol{X}_{t+\Delta t} = \left[{}^{S}\boldsymbol{x}_{t+\Delta t}, {}^{1}\boldsymbol{x}_{t+\Delta t}, \ldots, {}^{N}\boldsymbol{x}_{t+\Delta t}\right]^{\top}. \tag{5.37}
$$

The subject state transition is given as

$$
{}^{S}\boldsymbol{x}_{t+\Delta t} = \left[{}^{S}\boldsymbol{p}_t + \Delta t\, {}^{S}\boldsymbol{v}\right]. \tag{5.38}
$$

The airship state transition is given by

$$
{}^{n}\boldsymbol{x}_{t+\Delta t} = \left[{}^{n}\boldsymbol{p}_{t+\Delta t}, {}^{n}\boldsymbol{v}_{t+\Delta t}, {}^{n}\boldsymbol{\Theta}_{t+\Delta t}\right]^{\top}, \tag{5.39}
$$

where ${}^{n}\boldsymbol{p}_{t+\Delta t}$ is calculated as

$$
{}^{n}p_{t+\Delta t} = {}^{n}\boldsymbol{p}_t + ({}^{n}\boldsymbol{v}_t)\Delta t. \tag{5.40}
$$

${}^{n}\boldsymbol{v}_{t+\Delta t}$ is calculated as

$$
{}^{n}v_{t+\Delta t} = {}_{F}^{n}\boldsymbol{v}_{t+\Delta t} + {}^{F}\boldsymbol{v}, \tag{5.41}
$$

where

$$
{}_{F}^{n}\boldsymbol{v}_{t+\Delta t} = \begin{bmatrix} ({}_{F}^{n}v_{ht} + {}_{F}^{n}\dot{v}_{ht}\,\Delta t)\cos({}_{F}^{n}\chi_{t+\Delta t}) \\ ({}_{F}^{n}v_{ht} + {}_{F}^{n}\dot{v}_{ht}\,\Delta t)\sin({}_{F}^{n}\chi_{t+\Delta t}) \\ {}_{F}^{n}v_{zt} + {}_{F}^{n}\dot{v}_{zt}\,\Delta t \end{bmatrix}. \tag{5.42}
$$

${}^{n}\boldsymbol{\Theta}_{t+\Delta t}$ is computed as

$$
{}^{n}\boldsymbol{\Theta}_{t+\Delta t} = \begin{bmatrix} {}^{n}\varphi_{t+\Delta t} \\ {}^{n}\theta_{t+\Delta t} \\ {}^{n}\psi_t + {}^{n}\dot{\psi}_t\,\Delta t \end{bmatrix}. \tag{5.43}
$$

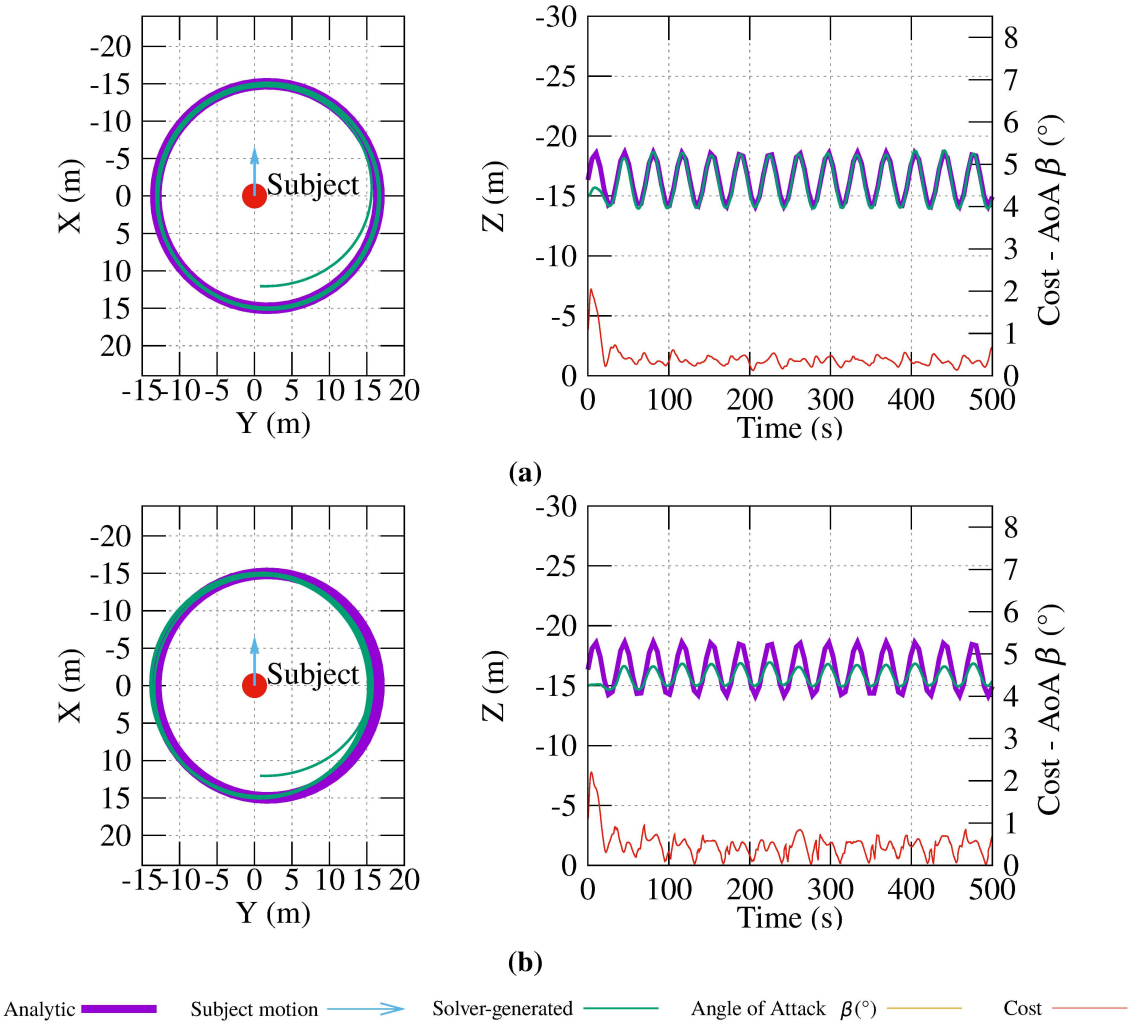


Figure 5.5: Orbital space around a moving subject for a fixed yaw rate ${}^{n}\dot{\psi}_t = \text{const}$ and velocity constraints on the airship $v_{\min} \leq \|{}^{n}_{F}\boldsymbol{v}_t\| \leq v_{\max}$. The analytic solution or closest approximation is shown in purple. The green trajectory is a solver-generated numeric solution, attempting to keep the camera centred on the subject. The red line is cost ${}^{n}E_{ct}$. **a)** 2D case. We assume $c_l \to \infty$, which implies ${}^{n}\psi_t = {}^{n}_{F}\chi_t$ and ${}^{n}_{F}\beta_t = {}^{n}\theta_t = 0$. The solver generated trajectory and the analytic trajectory are a close match. **b)** Analysis of the effect of ${}^{n}\theta_t$. AoA ${}^{n}_{F}\beta_t$ is assumed 0. The numeric solution (green) considers ${}^{n}\theta_t$, while the analytic solution does not. The resulting optimal trajectory has less altitude variation than the analytic solution, but shares the same minimum radius. However, the maximum radius is significantly reduced, implying that there could be viable orbits in 3D with $\left\|{}^{S}_{F}\boldsymbol{v}\right\| > \frac{1}{4}(v_{\max} - v_{\min})$.

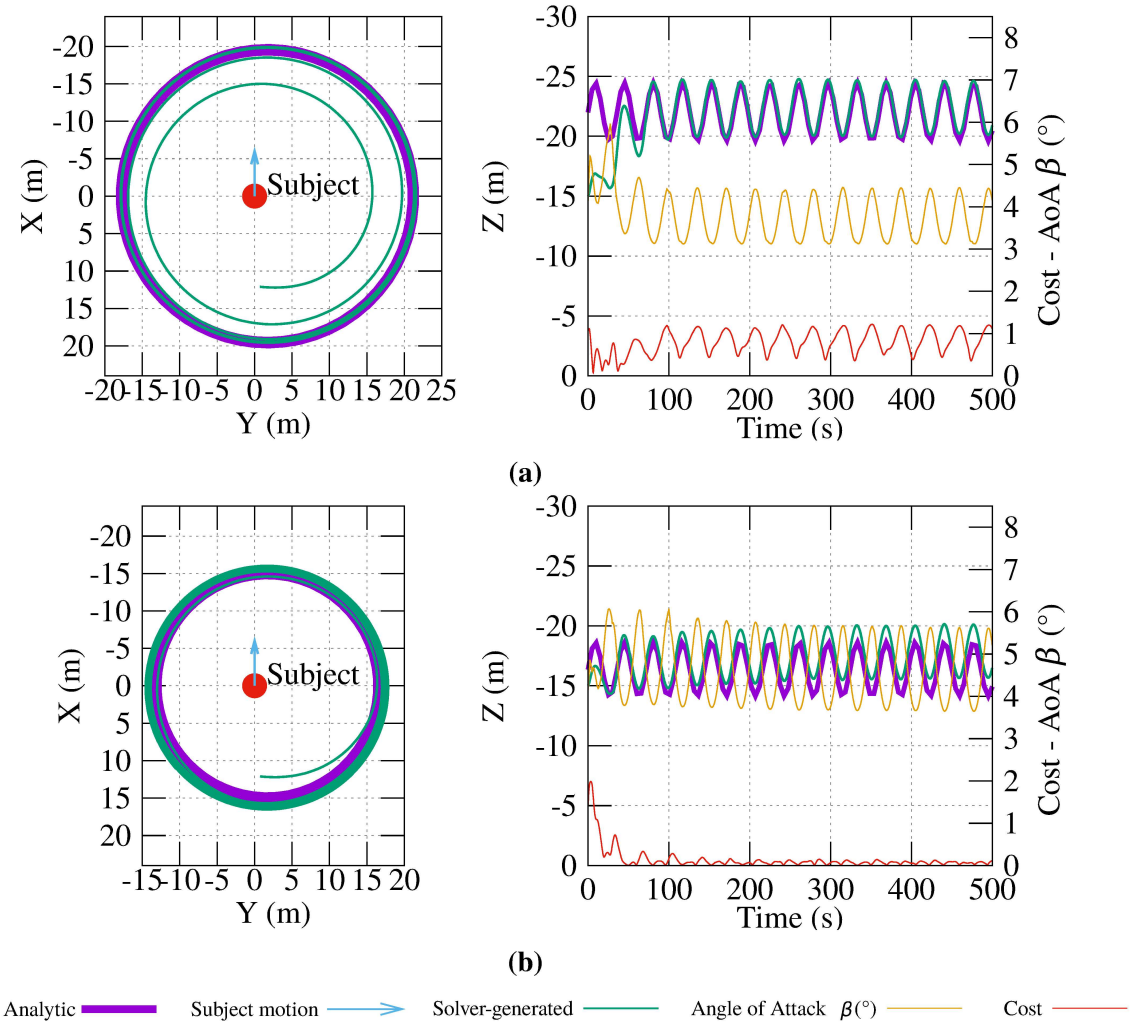


Figure 5.6: Orbital space around a moving subject for a fixed yaw rate ${}^{n}\dot{\psi}_t = \text{const}$ and velocity constraints on the airship $v_{\min} \leq \|{}^{n}_{F}\boldsymbol{v}_t\| \leq v_{\max}$. The analytic solution or closest approximation is shown in purple. The green trajectory is a solver-generated numeric solution, attempting to keep the camera centred on the subject. The red line is cost ${}^{n}E_{ct}$. Analysis of the effect of ${}^{n}_{F}\beta_t$ for **a)** $\gamma = \frac{\pi}{2}$ and **b)** $\gamma = 87°$. ${}^{n}\theta_t$ is assumed zero. At a sensor angle of $\gamma = \frac{\pi}{2}$, this angular offset forces the airship on an outward spiral. Once the maximum constraint-satisfying distance is reached (purple), the airship can no longer maintain the subject centred in the camera view, as seen in the non-zero cost (red). At a sensor angle of $\gamma = 87°$ the solver converges to a solution only slightly further out than $\min(\|{}^{n}_{F}\boldsymbol{v}\|) = v_{\min}$ (purple), at which the effects of AoA ${}^{n}_{F}\beta_t$ are fully compensated over a full orbit, allowing the subject to remain centred in the camera image within tight tolerances.

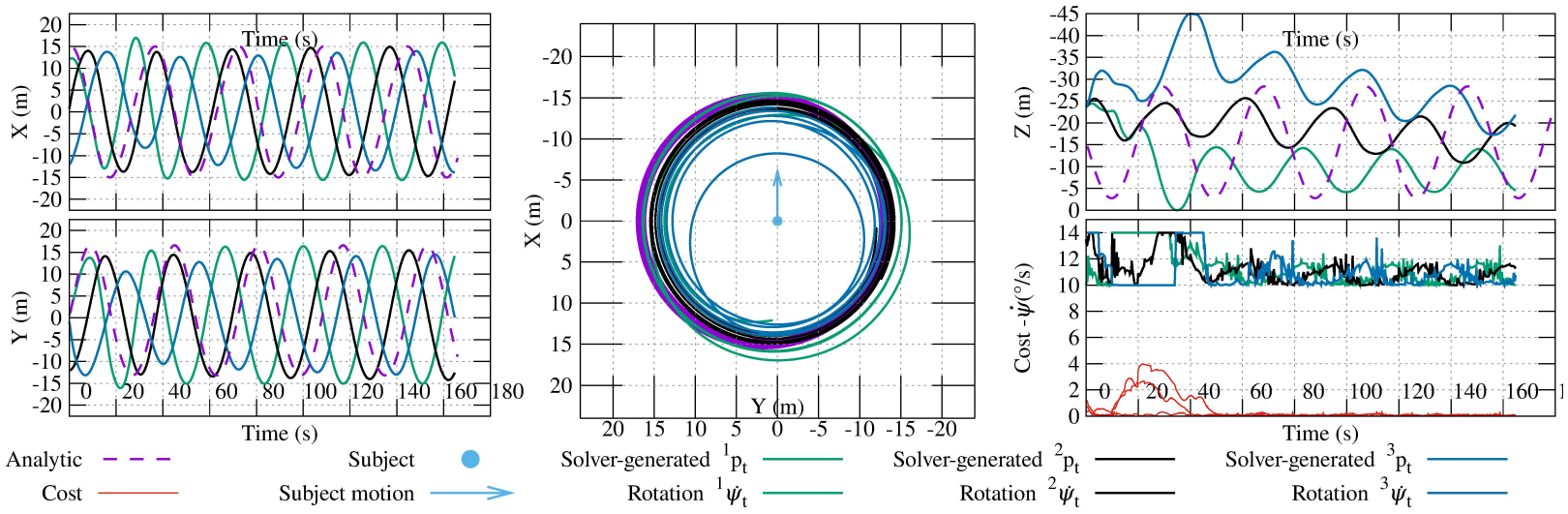


Figure 5.7: A numerically solved optimal formation of 3 airships around a moving subject. Trajectories have a non-optimal starting point, leading to some non-optimal subject centring in order to converge on a global optimal formation within the first $1\frac{1}{2}$ orbits. The red plot shows only the cost-term ${}^nE_{ct}$.

In (5.43), ${}^{n}_{F}\chi_{t+\Delta t}$, ${}^{n}\varphi_{t+\Delta t}$ and ${}^{n}\theta_{t+\Delta t}$ are calculated according to (5.27), (5.29) and (5.31), respectively.

5.3.6 Model Evaluation

We implemented the optimization problem described above (Subsection 5.3.5) using OpEn/PANOC [143, 144]. Additional common cost terms, omitted for brevity, are implemented to minimize control effort and add soft constraints around hard state constraint boundaries. To increase the accuracy at large Δt, we use 4th order Runge-Kutta to integrate over $\int {}^{n}_{F}\boldsymbol{v}_t\,\delta t$ to propagate ${}^{n}\boldsymbol{p}_t$ in (5.40).

A single airship trajectory, optimized using the model in Section 5.3.5 converges to the analytical optimal solution described in Section 5.3.3 as shown in Figure 5.5a), if ${}^{n}_{F}\beta_t$ and ${}^{n}\theta_t$ are set to 0 and ${}^{n}\dot{\psi}_t$ is set constant. If the starting state is not on the optimal trajectory, there is a short transitional phase, where the cost is minimal but not zero. k_d is set to 0 for this evaluation to compute the optimal trajectory, without regard to the distance between the airship and subject S.

As shown in Figure 5.6a), a non-zero lateral angle of attack ${}^{n}_{F}\beta_t$ results in a trajectory, where the subject is not centred in the camera, but is offset by approximately ${}^{n}_{F}\beta_t$. ${}^{n}_{F}\beta_t$ is smaller at higher airspeed, as such the numeric solution approaches the largest radius possible within constraints. To compensate for this effect, one solution is to adjust the camera azimuth γ (Figure 5.2). An intuitive correction would be $\gamma = \frac{\pi}{2} - \text{mean}({}^{n}_{F}\beta_t)$, so that the average would be zero. This mean depends not only on the model parameters but also on the mean velocity, which in turn depends on the orbit radius. We can estimate it by applying (5.23) to a circular orbit with the same parameters. In Figure 5.6a), ${}^{n}_{F}\beta_t$ oscillates between 3° and 4.5°.

Figure 5.6b) shows that setting $\gamma = 87°$ solves this problem and allows an optimal solution close to the minimum radius. Intuitively, a larger orbit results in higher velocities and a smaller ${}^{n}_{F}\beta_t$. When $\gamma > \frac{\pi}{2} - \text{mean}({}^{n}_{F}\beta_t)$, the airship needs to fly an inward spiral to keep the subject centred in view. Similarly, too tight an orbit will lead to a larger lateral angle of attack with $\gamma < \frac{\pi}{2} - \text{mean}({}^{n}_{F}\beta_t)$. Consequently, the formation will naturally converge on an orbit with $\gamma = \frac{\pi}{2} - \text{mean}({}^{n}_{F}\beta_t)$. This is not a limitation for our approach, as we still have ${}^{n}\dot{\psi}_t$ as a controllable variable to change the desired distance to the subject.

Figure 5.5b) shows the effect of ${}^{n}\theta_t$ with a camera angled $\phi = -45°$ downwards. Intuitively, when the nose of the airship raises in a climb, which is the case whenever the radius ${}^{n}_{S}r_t$ grows (5.30), the camera looks further ahead. To compensate for this and keep the subject S in view requires a sharper turn, diminishing the ${}^{n}_{S}r_t$ increase. Symmetrically, when descending during shrinking ${}^{n}_{S}r_t$, the camera looks behind. Both effects reduce the change in ${}^{n}_{S}r_t$. As the change in radius ${}^{n}_{S}r_t$ is now less pronounced as in the analytic 2D case in Subsection 5.3.3, the upper bound for the subject velocity (5.14) becomes relaxed.

As shown before, solutions to the 2D simplified orbiting problem are guaranteed to

exist if $\left\| {}^{S}_{F}\boldsymbol{v} \right\| \leq \frac{1}{4}(v_{\max} - v_{\min})$. However, with the change in ${}^{n}\theta_t$, solutions might exist for $\left\| {}^{S}_{F}\boldsymbol{v} \right\| > \frac{1}{4}(v_{\max} - v_{\min})$, but we have no analytical solution for the exact boundary, which depends on parameters γ, ϕ, ${}^{n}_{F}\beta_t$, c_l, ${}^{n}\varphi_t$ and ${}^{n}\dot{\psi}_t$.

Finally, we apply the solver to compute optimal formation trajectories for 3 airships. To allow airships to adjust their relative angles to the subject S, we no longer fix the yaw rate ${}^{n}\dot{\psi}_t$, but allow the solver to adjust ${}^{n}\dot{\psi}_t$ within new constraints $\dot{\psi}_{\min} \leq {}^{n}\dot{\psi}_t \leq \dot{\psi}_{\max}$ to converge to an optimal multi airship formation, shown in Figure 5.7. The constraints are identical for all airships. As cost weight $k_f > 0$ is now in effect, the optimal solution is a trade-off between ${}^{n}E_{ct}$ and ${}^{n}E_{f_t}$.

5.4 Experiments and Results

5.4.1 Implementation and Experimental Setup

We implement the model in Section 5.3.5 as a nonlinear MPC for real airships. Like [22], it is computationally centralized, but distributed from an information perspective as each airship runs its own instance of the formation controller based on the information it receives from other airships. We reduce the planning horizon to approximately $\frac{1}{3}$ orbit. This planning horizon is long enough to conduct a 180° turn if necessary, e.g., in case of suboptimal starting conditions or to avoid collisions. Additional control constraints are enforced to limit the change in ${}^{n}\dot{\psi}_t$ in each time step, and state constraints such as minimum-height to avoid ground contact. We also added additional non-convex constraints to enforce sufficient distances between airships at all times and prevent collisions between them and possible static obstacles. In the absence of static obstacles, any solution close to the optimum has the airships maximally spread out due to cost term (5.35), which naturally forms a large convex corridor in which optimal orbits reside.

We modify the open-source airship controller in [15] to take ${}^{n}\dot{\psi}_t$, ${}^{n}_{F}\dot{v}_{ht}$ and ${}^{n}_{F}\dot{v}_{zt}$ as direct control inputs. We can measure ${}^{n}_{F}v_h$ using a Pitot probe [128] and ${}^{n}\boldsymbol{x}_t$ and ${}^{n}\dot{\psi}_t$ using GPS and inertial sensors, which allows us to estimate the wind vector ${}^{F}\boldsymbol{v}_t$ by inverting (5.42) with a numerical approximation method. We employ the cooperative perception framework of AirCap [20] to estimate ${}^{S}\boldsymbol{x}_t$ from ${}^{1\ldots N}\boldsymbol{x}_t$ and visual detection in the camera images using a single shot detector neural network [19] and a distributed Bayesian filter. We assume ${}^{F}\boldsymbol{v}$ and ${}^{S}\boldsymbol{v}$ are constant over the planning horizon. We estimate the model specific parameters c_l for real airships by measuring ${}^{n}_{F}\beta_t$ during a constant rate turn (5.23). With this, we can use our MPC to control a real or simulated airship. With a horizon of 10 time steps and $\Delta t = 1.25$s, the OpEN engine [143] typically converges within ~ 20ms on a Jetson TX2 embedded computer, which is sufficiently fast for real-time applications.

For all experiments, we set $k_c = 1$, $k_d = 0.6$, $d_c = 15$m, $k_f = 100$ as optimization weights. Camera and vehicle parameters for the simulated airships were $\gamma = 82°$, $\phi =$

$-30°$, $c_l = 0.24$. The state constraints were set to $v_{\min} = 0.5\frac{\mathrm{m}}{\mathrm{s}}$, $v_{\max} = 4.0\frac{\mathrm{m}}{\mathrm{s}}$, $v_{z\max} = 0.5\frac{\mathrm{m}}{\mathrm{s}}$, $\dot{\psi}_{\min|\max} = \pm 18\frac{^\circ}{\mathrm{s}}$, allowing both left and right turns.

5.4.2 Experiment Description and Evaluation Metrics

Experiments 1-3 are done in simulation. In Experiment 1, we evaluate different formation sizes up to $N = 6$ airships around a stationary subject, with constant wind of $\left\| {}^F\boldsymbol{v} \right\| = 0.6\frac{\mathrm{m}}{\mathrm{s}}$. In Experiment 2, we investigate the effect of wind and wind gusts with $N = 3$ airships. In Experiment 3, we investigate the effect of subject motion, including sudden turns, with and without wind. Finally, in Experiment 4, we evaluate our MPC in an outdoor environment with a real airship but a simulated stationary subject.

In Experiments 1–3, we evaluate 4 different metrics.

1. The visual tracking accuracy, which measures how well the estimated subject position, given as ${}^S\boldsymbol{p}_t$, agrees with the ground-truth ${}^S_{\mathrm{GT}}\boldsymbol{p}_t$ in simulation, calculated as $\left\| {}^S\boldsymbol{p}_t - {}^S_{\mathrm{GT}}\boldsymbol{p}_t \right\|$.

2. Tracking uncertainty, measured as the trace of the covariance matrix of the subject position estimate.

3. Subject visibility, which is the percentage of video frames that have the subject in the field of view (FOV). The subject is considered in FOV if its projected centre point is less than half the camera resolution from the image centre point. For this, the camera is simulated as a pinhole camera with 640×480 pixel resolution and 90° camera FOV.

4. The proximity to the image centre, measured as the distance in pixels of the subject centre to the camera image centre.

In Experiment 4, since no tracking takes place, we only evaluate metrics 3) and 4), but in addition, we compare the actually flown trajectory to simulations under similar wind conditions.

5.4.3 Experiment 1 - Formation Size

Figure 5.8 summarizes the effect of formation size on the tracking formation with $N = 1,2,3,4,5$ and 6. All evaluations were conducted with wind speed $\left\| {}^F\boldsymbol{v} \right\| = 0.6\frac{\mathrm{m}}{\mathrm{s}}$. For $N = 1$, there is a larger tracking error and uncertainty, which is expected since a single airship cannot reliably estimate the depth from a single camera image. Consistent with [20, 22], for $N >= 2$, the tracking error and uncertainty becomes smaller with each additional airship in the formation. All formations managed to maintain sight of the subject 100% of the time, with the distance to the camera centre averaging around 30px and never exceeding 40px.

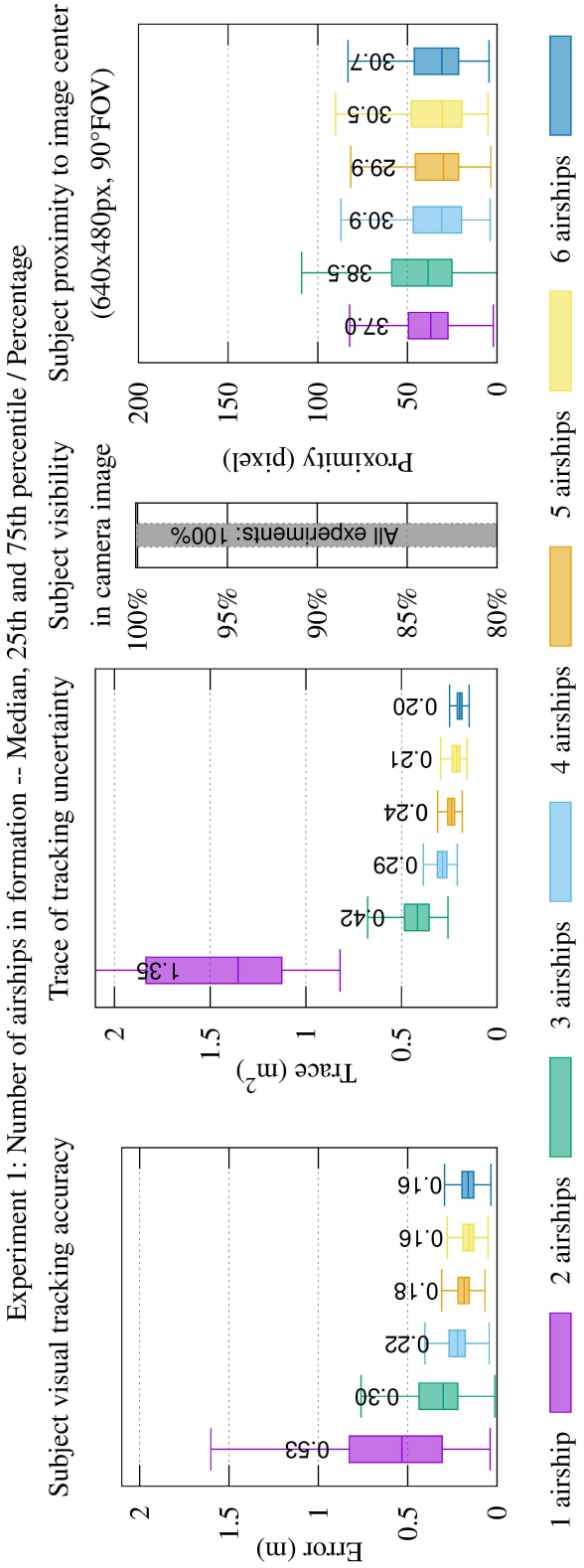


Figure 5.8: Experiment 1: Variation of Formation Size.

5.4.4 Experiment 2 - Wind Velocity

Figure 5.9 shows the effect of wind on the tracking formation with $\left\| {}^{F}\boldsymbol{v} \right\| = 0\frac{\mathrm{m}}{\mathrm{s}}, 0.3\frac{\mathrm{m}}{\mathrm{s}}, 0.6\frac{\mathrm{m}}{\mathrm{s}}, 0.9\frac{\mathrm{m}}{\mathrm{s}}, 1.4\frac{\mathrm{m}}{\mathrm{s}}, 2.2\frac{\mathrm{m}}{\mathrm{s}}$ and $3.0\frac{\mathrm{m}}{\mathrm{s}}$. All evaluations were conducted with $N = 3$ airships and a stationary subject. For wind gusts, we utilize the Dryden turbulence model at turbulence level "moderate". Up to $1.4\frac{\mathrm{m}}{\mathrm{s}}$ wind, there is a small but consistent decrease in tracking accuracy, however as the wind exceeds $\frac{v_{\max}-v_{\min}}{4}$ (from $\left\| {}^{F}\boldsymbol{v} \right\| = 0.9\frac{\mathrm{m}}{\mathrm{s}}$) the MPC no longer manages to keep the subject S well centred in camera images, which is consistent with our analysis on 2D orbits. With $\left\| {}^{F}\boldsymbol{v} \right\| >= 2.2\frac{\mathrm{m}}{\mathrm{s}}$, the airships could no longer maintain sight of the subject through the entire orbit. With stronger wind, there were increasingly large blind sectors during which S could not be observed. This also results in increased tracking error, although the subject was still visible in the cameras for close to 90% of all camera frames.

5.4.5 Experiment 3 - Subject Not Stationary

Figure 5.10 shows the effect of subject motion. For two of the trials, with no wind and $0.6\frac{\mathrm{m}}{\mathrm{s}}$ wind, the subject is stationary, while for the third and fourth, the subject follows a repeated predefined trajectory. This subject path consists of straight and curved segments with sharp direction changes. This simulates the worst case described in Subsection 5.3.3. In the trials with a moving subject, the subject velocity is $\sim 1\frac{\mathrm{m}}{\mathrm{s}}$, which is much larger than $\frac{v_{\max}-v_{\min}}{8}$. Consequently, keeping the subject centred through the reversals is not possible. However, the MPC still maintains the subject in the camera FOV for 96% and 98% of all video frames in those trials, with or without wind. Typically, only one out of three airships loses sight at a time and only for a very short time whenever an unanticipated subject direction change happens.

5.4.6 Experiment 4 - Real-World Experiment

We conducted a real-world experiment with one airship (Figure 5.11) to verify the applicability of our control method. To isolate evaluation of the control method, no tracking took place, the subject position was fixed at $[0,0,0]$, providing ground-truth for the subject location. The wind was measured between 1 and 2 m/s with gusts exceeding 4 m/s from East to North-East. ($^{S}_{F}\boldsymbol{v} > v_{\max}$). Under these conditions, stable equally spaced formations with multiple airships would not have been possible. In Figure 5.12, we show the real-world flight alongside two simulated flights at different mean wind speeds using the Dryden model with "severe" simulated turbulence intensity. Both the resulting trajectories and performance of the controller w.r.t. keeping the camera pointed at the subject in the real experiment are in line with the simulation results under equivalent wind conditions.

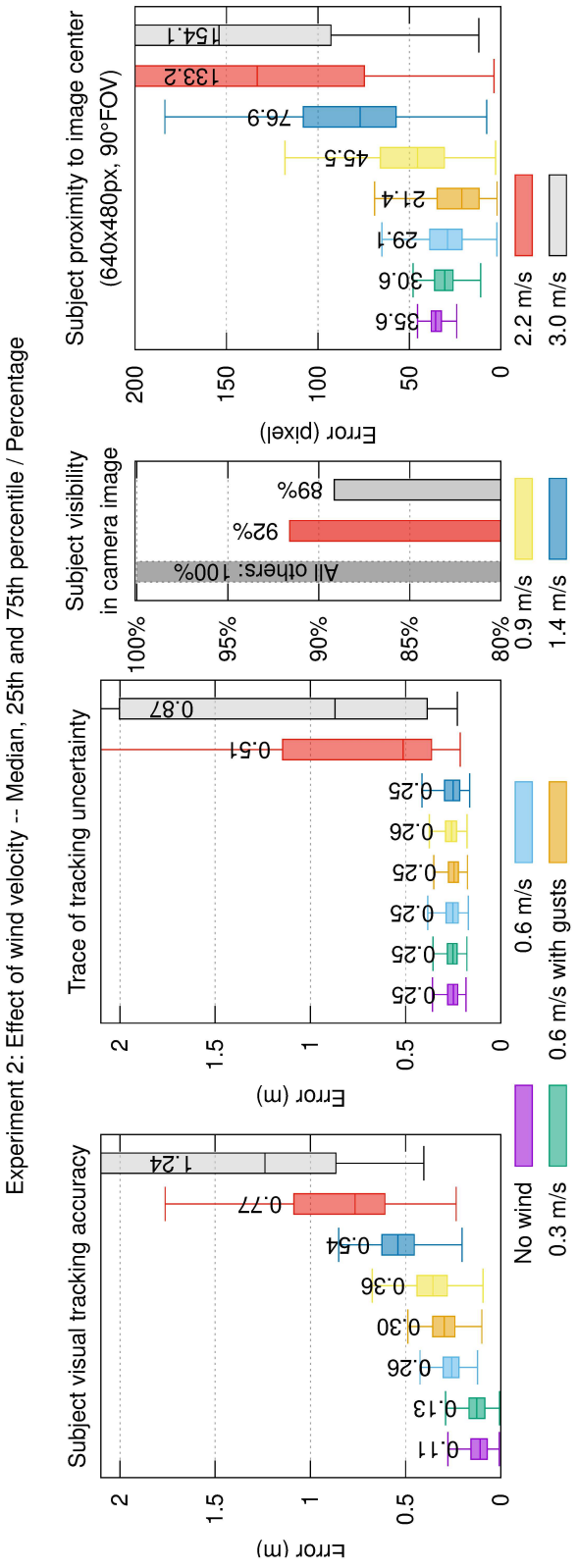


Figure 5.9: Experiment 2: Variation of Wind Velocity.

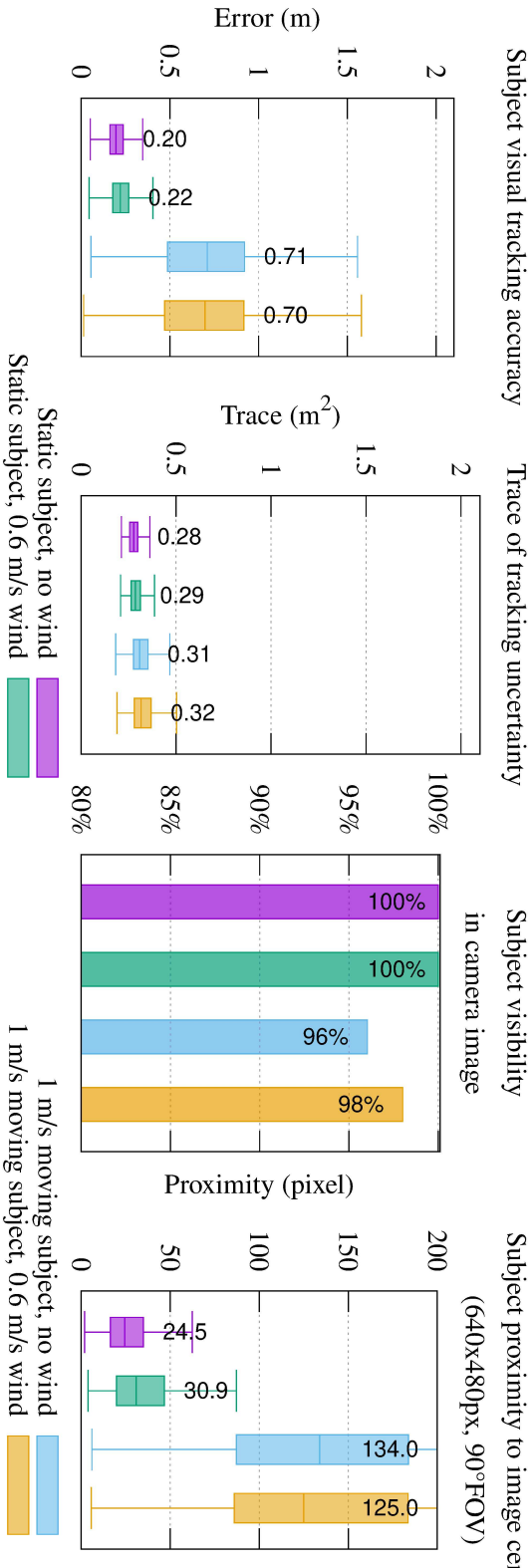


Figure 5.10: Experiment 3: Subject not Stationary

Figure 5.11: Experiment 4: Real-world flight. This video still taken during the flight experiments in 2023 shows our blimp prototype near the University Campus in Stuttgart, Germany.

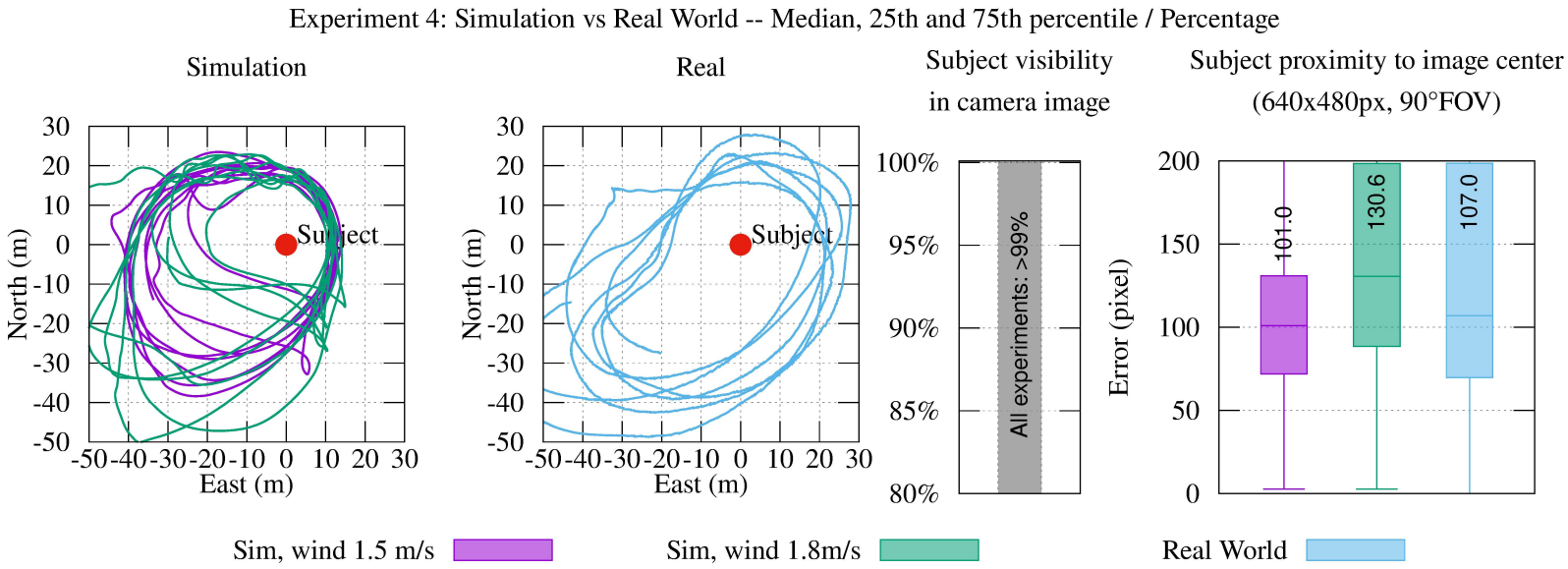


Figure 5.12: Experiment 4: Real-world flight. Comparison of real-world flight results to two simulated flights in similar wind conditions with 1.5 and 1.8 m/s average wind speeds, yielding comparable results.

5.5 Conclusion and Outlook

In this chapter, we presented a nonlinear MPC-based method for formation control of airships with frame-fixed cameras. The goal of the MPC is to keep a moving subject within the field of view of the cameras of all the airships simultaneously, while adhering to other motion and view constraints. We derived analytical solutions in a simplified 2D case for boundary conditions on the subject velocity. We also approximated analytical solutions for the realistic 3D case where possible, excluding the effect of the pitch angle, and averaging over the lateral angle of attack. We formulated our objective as an optimization problem and solved it using a numerical solver. This was followed by an MPC implementation for real-time application. Through extensive simulations, we show our method's efficacy, accuracy and reliability. Through a real-world experiment, we demonstrate its speed, stability and real-world applicability. In future work, we will investigate the problem of collision and obstacle avoidance within the context of airship formation control. Generalization to other non-holonomic vehicles, e.g., fixed-wing aircraft, is also planned.

Simulation and flight source code for this chapter is open-source and can be found in [145]

5.6 Publication

This chapter was published in IEEE Robotic Automation Letters Vol. 8 Number 6 in 2023 [16] under the title "Viewpoint-Driven Formation Control of Airships for Cooperative Target Tracking". It was presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2023 in Detroit. The publication subsequently won the University of Stuttgart, Faculty of Aerospace Engineering and Geodesy publication prize in 2024.

Chapter 6

Field Evaluation of an Airship System for Horse Observation

This chapter covers the field evaluation of a new airship design, which we designed in cooperation with the commercial partner "Windreiter" [146]. It addresses some shortcomings of the design presented in Chapter 4, especially the low top speed and high susceptibility to wind, which it achieves with a significantly more light-weight hull and a sleeker, more aerodynamic hull shape. We also mount the IMU in the payload gondola, which allows more accurate estimation of the camera orientation. Separating IMU and camera on a non-rigid vehicle is not necessarily a good idea, as there can be significant discrepancies in orientation when the airship is deformed. To address electromagnetic interference issues, we used USB-2 cameras that are not jamming GPS and designed for a separate payload and main gondola. This airship was deliberately designed for wildlife observation. Consequently, we performed the initial flight evaluation in-situ in the Hortobágy national park in Hungary, with the goal to combine all the methods presented in Chapter 2 up to Chapter 5 on this new airship. That is, to successfully detect, track, observe and follow horses. Considering it was a brand-new design that had not even flown outside a hangar before, with a method that had never been used on animals before, this was an extremely ambitious endeavour. Nevertheless, as shown in this chapter, the mission and research trip was a complete success.

6.1 Introduction

In this chapter, we evaluate the use of lighter than air UAVs for use in studying animals in the wild, especially grazing herds in open terrain. The aerial perspective allows observation at a scale and in a context that is not possible on the ground, and offers new insights into group behaviour. Airships seem like an obvious choice for wildlife field-studies. Unlike fixed wing and multi-copter systems, which suffer from limited flight time and have noise and safety concerns, airships are characterised by static buoyancy and low mass density. This makes them both inherently efficient and safe. However, airships can be challenging from a ground handling and control point of view. Airships can also be more affected by wind than "traditional" UAVs. In this chapter, we showcase our system

Figure 6.1: Our new blimp, observing horses. This photo shows our autonomous blimp flying over a pair of Przevalski's horses with their foal, which have separated from the main herd. Picture taken in the Hortobágy National Park in Hungary during flight experiments conducted in 2023.

of lighter than air vehicles designed to track, follow and visually record wild horses from multiple angles. We also show the results of our field evaluations. The following aspects are of particular importance for this airship application:

6.1.1 Airship Design

The design of an airship for animal studies is determined by the following requirements: i) Speed and efficiency: the airship must be capable of station-holding for extended times in open terrain, which frequently encounters significant winds. On many days during our experiments, the wind was in excess of $5\frac{\mathrm{m}}{\mathrm{s}}$ with gusts exceeding $10\frac{\mathrm{m}}{\mathrm{s}}$. An airship built for extended flight times must be designed for comparable sustained airspeeds, which requires a geometric high fineness ratio, low drag and efficient thrusters. ii) Manoeuvrability: being able to maintain stable camera tracks of moving subjects on the ground, especially in the presence of wind and turbulence, requires sufficient control authority in combination with passive stability. iii) Payload: in addition to actuation, on board computer vision and image processing are required to perform autonomous observation of

moving subjects. This means, capable embedded compute hardware with non-negligible weight and power consumption must be carried and supplied with power for an extended time. iv) Durability: operation "in the wild" for an extended time-frame requires that the hull, structure and payload are sufficiently rugged to maintain function and lift-gas retention in the presence of weather, turbulence and handling related strain. iv) Safety: operating a UAV in a national park and/or near wildlife requires high reliability of flight control and navigation, as well as vehicle electronics and mechanics with inbuilt fail-safes. v) Pressure: changes in air pressure and temperature over extended mission times cannot be ruled out. Hull pressure must be maintained either with sufficient margins or with active control (e.g. ballonets).

6.1.2 Sensing and Computer Vision

When studying wild animals and their behaviour, a critical criterion is to be non-invasive. We cannot rely on markers, transmitters or similar strategies requiring hardware on the animal. Instead, passive sensors must be used. Since the visual modality is most important for studying the subjects in question, using vision for real-time sensing and tracking is the obvious choice.

6.1.3 Autonomous Control

To navigate a formation of airships near animals, both formation and individual vehicle control needs to be addressed under consideration of both airship dynamics, wind and motion of the subject animals. To address the needs of directional sensors and computer vision, the control algorithm needs to keep the subject animals in the camera field of view. This should be done from multiple angles with multiple vehicles simultaneously, while also maintaining a safe distance to avoid collisions between all vehicles, terrain or disturbances to subjects.

6.1.4 Simulation

Complex autonomous algorithms and operations need to be extensively tested under realistically simulated conditions before application in the field. This should include realistic (flight) physics, sensor behaviour including sensor noise, actuator effects and disturbances from the environment including wind and turbulence. To minimize discrepancies between simulation and the real world, software and hardware in the loop simulation is desired, where sensing and control methods are agnostic w.r.t. the source of data, i.e. simulated vs. real sensors. Timing/sequencing should be as realistic as possible.

6.1.5 Field Experiments and Practical Considerations

Operating an airship under field conditions can be more complicated than operating small multi-copters, which can often be transported by a single person and, if necessary, hand launched. The main limitations are the need to inflate the airship with lift-gas and the significant volume and drag of the vehicle once inflated. Inflating and trimming the vehicle therefore should ideally happen under sheltered conditions. While making the vehicle flight-ready, it is very vulnerable to wind and turbulence, especially near obstacles. Having a way to arrest both the nose and the tail of the vehicle securely during handling is important. Takeoff and landing are another challenge, especially in vehicles that do not have vertical thrust. Thermal activity and detachment of hot air bubbles are common phenomena in open terrain, especially on sunny days. Typically, airship UAVs are trimmed heavier than air by up to 10% of their mass to ensure the vehicle safely returns to the ground in case of control-loss. This can be insufficient when the airship encounters thermal updrafts. Even high thrust in combination with a nose-down attitude might not be sufficient to maintain altitude. Similarly, microbursts with significant down-force can be encountered, which can threaten to drive an airship into the ground. A combination of navigation and control strategies, as well as human override capability, can be utilized to overcome these challenges. It is crucial that all operators of airship UAVs are familiar with these phenomena and how to handle them.

6.2 State of the Art

While the benefits of UAVs for wildlife research are manifold and undisputed [147, 148], multi-copters have known drawbacks, including disturbance of animals [13, 149] and limited flight time [150].

A recent literature study reported a significant increase in use of lighter than air vehicles for wildlife research [151] citing several advantages, including significantly higher efficiency, lower cost and longer flight times. However, established systems used in the field today are tethered aerostats without propulsion or autonomous mobility [152, 153].

Manned airships have been extensively used for aerial observation up to the 1930s, including geographic aerial surveys and military observation in World War I [154]. The use of airships for research expeditions in Africa had been proposed as early as 1908 [155]. Airships have been used for multiple research expeditions, mostly into the arctic [156]. However, modern manned designs are too expensive to operate to compete with unmanned solutions, especially in remote areas.

Using autonomous robotic airships for environmental monitoring and wildlife observation has been proposed as a general concept [148], as well as motivation for both ship design proposals [157] and airship component research [158]. However, to our knowledge, no prior work described the design of a robotic airship specifically for this purpose nor its evaluation for wildlife observation under realistic field conditions.

This work builds on an extensive foundation of airship simulation and modelling [15] (Chapter 4), computer vision and state estimation [20] (Chapter 2) and model predictive control for airships in formations relative to observation subjects [16] as presented in Chapter 5.

6.3 Methodology

6.3.1 Airship Design

Structure

In cooperation with a commercial partner, the German company "Windreiter" [146], we designed a 5.5m, 3.6m^3 single hull prototype. It has a metal coated polymer film envelope with a mass of 430g. To minimize drag, a configuration with 3 tail-fins was chosen. The propulsion gondola with batteries is located under the centre of mass. Additionally, the airship is equipped with a forward mounted payload gondola, to physically distance sensors such as the magnetometer and pressure sensors from propulsion disturbances. The fins, rudders and gondolas are made from Depron™ and plywood+Depron sandwich. Each fin has a mass of approx 200g. The total maximum payload capacity at sea level is 1800g (Figure 6.2, Fig. 6.3). A rope for ground handling has been attached to the nose of the vehicle.

Power Supply and Propulsion

The airship has been equipped with 2 4S 5500mAh Lithium Polymer batteries (125Wh), with the option of flying with a third battery for 186Wh. Thrust is generated by 2 fixed, brush-less propellers in pull configuration, attached to the propulsion gondola. Power and propulsion have been designed w.r.t the air-frame with the goal of a $12\frac{\mathrm{m}}{\mathrm{s}}$ top speed and sustained flight at $8\frac{\mathrm{m}}{\mathrm{s}}$ for at least one hour.

Payload and Avionics

We equipped the airship with 2 Logitech Brio 4k USB webcams [159], one of which is front-facing for navigation, while the other is pointing to starboard and 30 degrees down for subject observation. The main computer is an NVIDIA Jetson TX2 [42] on an Auvidea J-120 embedded board [160]. We utilize the TX2's inbuilt 5 GHz Wi-Fi for inter-vehicle communication and communication with a ruggedized ground-station laptop. A Graupner HoTT transceiver [161] is used for remote flight control and long-range telemetry operating on 2.4 GHz. The flight controller (FC) is an Openpilot Revolution STM32 based embedded board [30], equipped with an IMU, 3 axis magnetometer and barometric static pressure sensor. Additionally, we equip a GPS receiver and a dynamic pressure-based airspeed sensor consisting of a Pitot tube [128] and a differential pressure

Figure 6.2: Our blimp prototype in the air. This picture shows our airship in flight, picture taken during flight experiments in the Hortobágy National Park, 2023. Clearly visible is the rope attachment we added to the nose as a field modification, as well as the payload and propulsion gondolas, the latter with attached ballast.

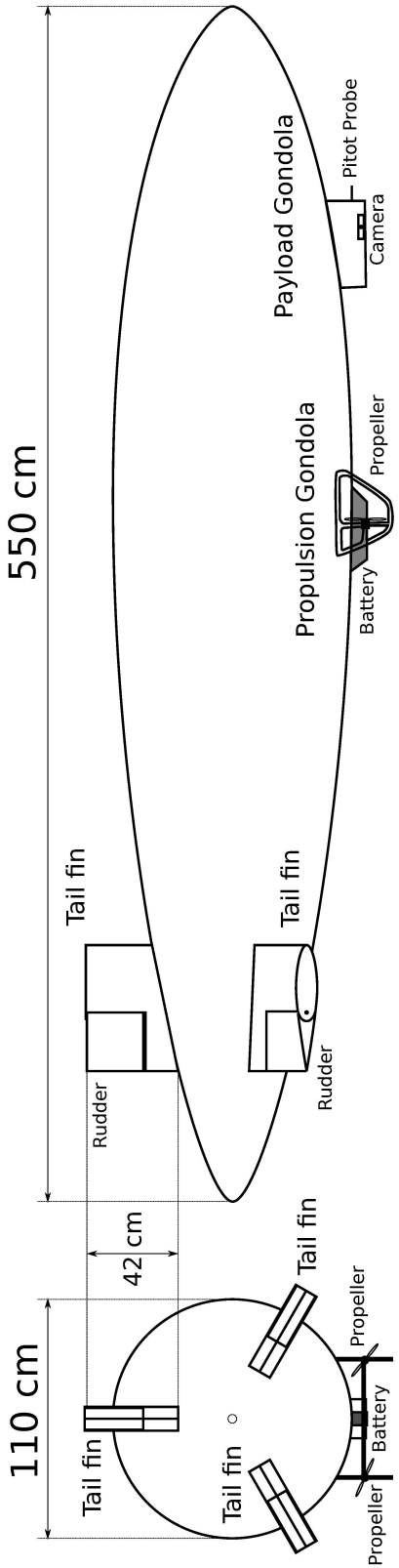


Figure 6.3: The dimensions and physical layout of the airship prototype, including rudders, propulsion and payload gondola.

sensor. The TX2 is equipped with a 1TB SSD drive for on board data storage, sufficient for several hours of recordings.

6.3.2 Sensing and Computer Vision

The flight controller (FC) runs our ROS-enabled [162] version of Librepilot [38]. It includes an Extended Kalman Filter (EKF) for state estimation which fuses GPS, IMU, barometer and magnetometer for self-pose estimate (position and orientation), which is made available to ROS running on the TX2 main computer. We use our own distributed cooperative Bayesian foveated visual tracker [20] (Chapter 2) across one or multiple vehicles to improve the self pose estimate and to track a subject, in this case an animal, on the ground. For visual detection in the relevant field of view (FOV), our approach uses a convolutional single shot multi-box detector (SSD) [19] running on the TX2.

6.3.3 Autonomous Control

All flight controls, i.e. all 3 rudders and the 2 main thrusters, are directly controlled by the FC. A cascaded PID controller is implemented on the FC, which allows basic low-level control, i.e. flying at a controlled airspeed and climb/sink rate on a straight or curved trajectory with controlled rotational rate. It is capable of position hold and way-point navigation, as described in [15] (Subsection 4.3.2). This low-level controller is also used to enforce a "sky-box", a geo-fenced area in which more sophisticated autonomous vision-based navigation is permitted. This ensures no ground collisions or fly-away conditions are encountered, even if there is a failure in the high-level control. Within the sky-box, when in autonomous mode, a Model Predictive Controller (MPC) [16] (Chapter 5) estimates the wind vector based on the airship's ground speed, dynamic pressure (i.e. airspeed), rotation rate and estimated angle of attack. The MPC knows the camera orientation on the airship. It calculates the optimal trajectory to keep the subject, which is tracked as described in Subsection 6.3.2, centred in the camera field of view, while maintaining equal angular separation w.r.t. the subject between all vehicles in the formation. For this, optimal trajectories for all vehicles are calculated. The MPC takes collision avoidance constraints into account for both moving vehicles, the subject and known static obstacles. It avoids direct overflight of the tracked subject for safety reasons. In the absence of obstacles and if winds are steady, the resulting trajectories resemble circular orbits around the subject, which are offset to one side based on the subject's velocity and/or wind [16]. In strong winds, alternative solutions are possible. A single airship can "hover" with its nose in the wind and/or fly parallel to a fast-moving subject. Manual override via the remote control transmitter is always possible for safety reasons.

Figure 6.4: The blimp takeoff procedure from a small "hill" to increase ground-clearance. Photo taken during initial flight experiments in the Hortobágy National Park in Hungary, 2023.

6.3.4 Simulation

We implemented an aerodynamic physics simulation of airship formations based on our previous work on "Simulation and Control of Deformable Autonomous Airships in Turbulent Wind" [15] using ROS and Gazebo [28] (Subsection 4.3.1). This includes hardware in the loop simulation, in which the low-level flight control algorithms are running on a physical or software emulated FC in real-time. Since our formations are subject-centred, we include a simulated subject, which the SSD detector can detect in simulated camera images as presented in our previous works [16, 20]. This allows real-time simulation of all components under diverse conditions, including different wind and turbulence regimes, as well as the loop-closure between perception, sensor fusion and formation control relative to the tracked subject. All our control and simulation code is available open source for ease of reproduction and the benefit of the community [145].

6.4 Field Experiments, Practical Considerations and Results

In August 2023, we conducted field experiments at the Hortobágy National Park in Hungary, which is a UNESCO world heritage property [35]. Our intent was to develop procedures and to assess the efficacy of airship and airship formations for motion capture and behavioural study of wild horses, particularly Przewalski's horses (Equus ferus przewalskii). We used manually annotated aerial drone footage from the same herd of

Przewalski's taken the previous year to train our SSD detector network, to reliably detect these horses in aerial video frames. We brought the previously un-flown airship prototype (Subsection 6.3.1) to the site in an off-road vehicle, along with two 50l 200Bar Helium lift-gas cylinders (sufficient for $20m^3$). The airship was assembled on-site. Although the use of a gazebo (tent) was considered, we opted instead for an empty farm-shed which was available on site to assemble and fill the airship. To protect the sensitive hull from dirt, a tarpaulin was placed on the ground. A net over the hull was used to prevent the hull from rising until attachments, batteries and sufficient ballast could be equipped. A pressure reducer valve and a hose were used to fill the airship with Helium. Metal screws in a plastic bag were attached to the main gondola as ballast, to trim the airship 300g heavier than air. Bringing the airship to the takeoff location was challenging due to gusty winds. It was not possible to maintain handling control over the airship using the gondola as a holding point due to excessive lateral wind forces. Handling became manageable using a rope attached to the nose. Takeoff and landing were performed by releasing and catching the airship by hand on the nose and front gondola. This was done on a small hill to increase ground clearance (Figure 6.4).

The vehicle maiden flight and initial tuning of the FC control coefficients were all conducted in situ. For safety reasons, the maximum throttle of the propellers was limited to 80%, at which an airspeed of $11\frac{m}{s}$ was achieved (Figure 6.5). We measured a peak current of 23A at sustained 80% throttle in flight with a battery voltage of 14.5V, resulting in a maximum sustained power consumption of 333W. Power consumption at 40% throttle (half the allowed) was 6.9A or 100W at approx $6\frac{m}{s}$. We did not measure engine RPM. The expected flight endurance is reciprocal with electric current and can be calculated based on our 10Ah battery capacity. Current and power are approximately proportional to the square of both the throttle setting and measured airspeed. Over several test-flights, the various components were evaluated, including the formation controller using a 1-airship formation around a simulated stationary subject. It was determined that a minimum airspeed of $4\frac{m}{s}$ was necessary to maintain sufficient vertical control authority in the presence of both strong thermals and microbursts. Once the control accuracy was sufficient, the vehicle was flown manually into the vicinity of wild horses. The subject tracking and formation control was then activated in flight for an autonomous formation consisting of 1 airship and one subject horse. The initial acquisition of the subject in the camera image was challenging, since the camera was mounted sideways and has a limited field of view. It was difficult for the pilot to estimate the relative position of the airship in the air and the horses on the ground, especially w.r.t. relative distance. We later added a forward facing navigation camera to the gondola to address this problem. Both cameras can be monitored from the ground in real time. The formation was monitored with a ruggedized ground-station laptop, connected to the airship via Wi-Fi. We achieved a practical Wi-Fi range of approximately 150m, which was a limiting factor but sufficient for our experiment. The operator could see a low-res visualization of the tracked subject, as seen by the on-board camera, on the laptop screen. We used the "Rviz" ROS tool with an interactive camera plugin to point and click on the correct horse to track, which

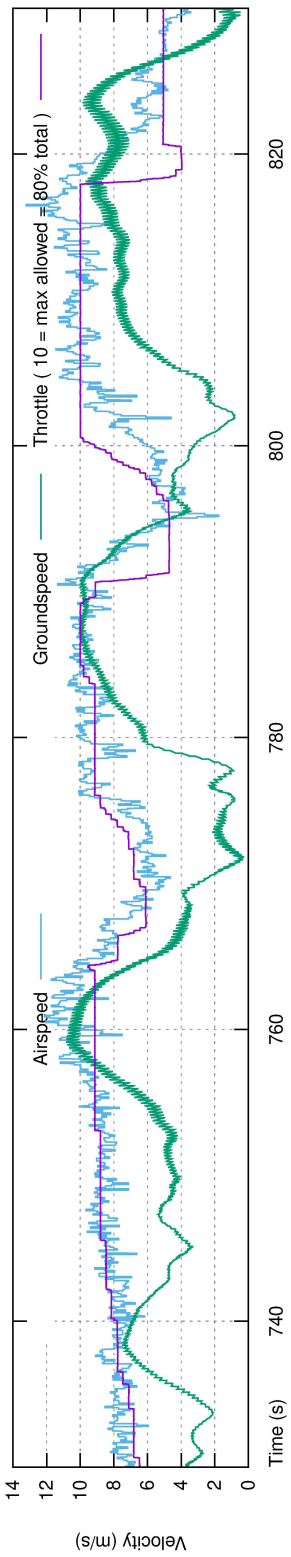


Figure 6.5: Logged airspeed vs groundspeed in back-and-forth flights on the same track (ROS). Peak power consumption was 333W at maximum throttle (not displayed).

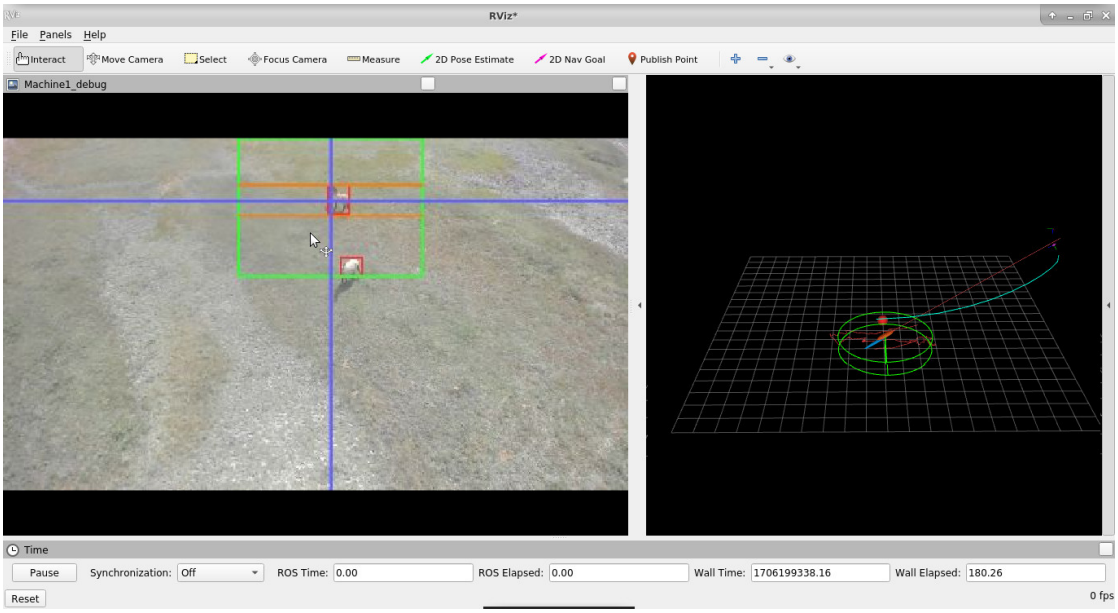


Figure 6.6: Our airship tracking a horse in strong winds, as seen from the base-station in Rviz. The left shows the on-board camera view with the projected position estimate (blue cross), region of interest (green) and detected horses (red). The right shows a 3D visualization of the airship and its predicted MPC trajectory (far right), as well as the safety exclusion zone around the horse position estimate (green circle).

re-initialized the tracker with the projected camera coordinate. This was sufficient to initialize the formation control once the animal was in sight. A 3D-visualization showed the airship relative to the subject, along with the estimated position uncertainty (Figure 6.6). We collected video data of the horses with the autonomous airship in very challenging conditions with approx $6-8\frac{\mathrm{m}}{\mathrm{s}}$ winds (Fig. 6.1). Video and telemetry data of the flight were stored on the on-board SSD, both in form of compressed raw-video, radio-telemetry log files and ROS-bag-files. The used video storage format was MJPEG compressed raw video data as it came from the camera, enriched with microsecond accurate frame timing information. Telemetry was stored as a telemetry stream-dump with timing information, allowing replay and analysis in the Librepilot GCS software. Recorded ROS data consisted of crucial airship-state information including position, velocity, IMU data, airspeed as well as remote control inputs and set points at a rate of 500Hz. ROS data also included low resolution video frames with annotations for debugging.

6.4.1 Discussion

The airship prototype exceeded our expectations. We went from maiden flight to successful autonomous flight tracking horses within only 4 days. The experiments indicated

that a vehicle with sufficient maximum airspeed and endurance has no problems tackling challenging wind conditions. Based on measured airspeed and power consumption, the calculated endurance of our airship at $8\frac{\mathrm{m}}{\mathrm{s}}$ would be 50min, which is technically below the targeted 1 hour. However, we attribute a large part of the high power consumption to drag caused by a foil-belt used for payload-gondola attachment, which started twisting and fluttering at high airspeeds. We flew with approximately 600g of ballast, which could be partially replaced by another 5000mAh battery, resulting in a 50% endurance increase. Unlike multi-copters, adding more batteries to airships does not result in higher power consumption. This indicates that minor modification to the existing design would result in both higher flight speeds and an increased endurance, meeting our design goals. The ultra-light metal-coated polymer hull is fragile in ground handling and can easily be scratched, punctured or otherwise damaged. We also found out that touching the hull with sweaty hands causes corrosion and discolouration of the metal coating. A single hull vehicle also does not allow for dynamic pressure compensation without adding lift-gas. A double hull, although more limiting on payload, would be better suited in the long term, and allow easy installation of an air filled ballonet. Ground handling was also the limiting factor w.r.t. the encountered wind speeds and gusts, since bringing the airship into a hangar, barn or shelter without colliding is very hard when there is lateral wind. During our experiments, ground handling mishaps were the only causes of damage to the vehicle, especially cracks in the Depron™ material, which could be repaired in situ. Small punctures in the hull, caused by thorny vegetation, were easily fixed with patches of sticky tape, without noticeable loss of lift gas. Thermals and down drafts were initially concerning, but could easily be overcome by ensuring the airship always maintained sufficient airspeed, i.e. vertical control authority. The system as a whole showed that airships are a feasible and useful tool for autonomous animal observation and research, matching and - in the aspect of endurance - easily exceeding the abilities of drones. The perceived propulsion noise of the airship was also significantly lower than the noise of a drone of comparable payload capacity. According to assessments by wildlife researchers observing our experiments, the presence of the airships did not disturb the horses.

6.5 Conclusions

We developed a system that allows a formation of airships to autonomously follow and record horses from an aerial view, enabling unmatched research opportunities such as motion capture and behaviour analysis. We showed that using airships for such studies in situ and in the wild is feasible. At a minimum, a large tent or similar shelter is needed to fill and fully assemble airships and/or keep them out of the elements between experiments. This overhead is not unreasonable compared to other aerial vehicles, and we determined the benefits outweigh the inconveniences. Takeoff and handling in the air are not harder than other drones, and the data collected is of equal quality or superior. The

main benefit of airships is the long flight time on the order of magnitude of hours, which allows for extended studies, not currently feasible with multi-rotor drones [14]. Control of airships, although more challenging than multi-copters capable of omnidirectional movement, can be solved with more sophisticated models and control methods, as we have shown. This work is based on both open and proprietary hardware designs which are commercially available, as well as publicly available open-source projects. To foster lighter than air research and use of airships especially for wildlife research, we have shared all our simulation and control code, developed in previous works [15, 16, 20], for the benefit of the community[1].

6.6 Publication

This chapter has been presented at the DELTAs 2024 airship conference in Mumbai under the title "Airship Formations for Animal Motion Capture and Behavior Analysis" [43] and was subsequently awarded "Best Paper".

[1] `https://github.com/robot-perception-group/Airship-MPC`

Chapter 7

Conclusion and Outlook

The work performed in this dissertation has led to several new insights.

1. Observing, measuring and recording a moving subject in unconstrained outdoor environments is both possible and technically feasible with a team of flying robots, as demonstrated. This is the main objective this work was set to accomplish.

2. Observation of wild animals is feasible with the techniques presented in this work, both with multi-copters and with airships. While each vehicle type has certain advantages and limitations (Section 7.2 below), the efficacy of each has been demonstrated.

3. Communication requirements are minimal. While communication is necessary between flying robots in a formation, our experiments in Chapter 2 have shown that the only information needed is the projected position of the subject whenever detected and the positions of all robots, along with the associated uncertainty of both. More so, the frequency at which this information must be available is quite low, making the system resilient and suitable for low bandwidth communication links. As we have shown, the bandwidth needed to achieve a given tracking accuracy is constant w.r.t. the number of robots.

4. Continuously observing formations are possible with robots constrained by non-holonomic motion — i.e. robots that need to face into the direction they are travelling — with fixed-frame mounted cameras or sensors — within known boundary conditions as shown in Chapter 5. Gimballed sensors and omnidirectional robot motion are beneficial, but not mandatory.

5. On board computation and convolutional neural networks are essential for detection, tracking and following of markerless subjects in the wild. Deep, learning-based methods are required to achieve the necessary generalization, while on board computation is needed to limit latency between perception and action and maintain acceptable communication overhead. Quite obviously, a system that relies on wireless communication of raw data would not scale, as outlined in Section 2.6.

6. Autonomous observation tasks can be addressed with iterative approaches for data collection and the training of learning-based methods. As we showed in Chapter 3, bootstrapping of detection methods is possible to speed up the data annotation process. This, in turn, enables better autonomous data-collection with learned detectors, which, in turn, provides more data for annotation and training in an operational feedback loop. This process allows new insights, new models and, in turn, informs new methods for data collection.

These points are further discussed in the following sections.

7.1 Technical feasibility

A limiting factor for the practical application so far is the lack of commercial drones with sufficiently flexible inter-drone communication. Many commercial applications focus on powerful single drones, which can be equipped with substantial payload and compute. These need to be programmed and controlled individually. Proprietary interfaces and locked-down, on-board compute hardware on affordable consumer drones prevent these vehicles from being used for the applications outlined in this dissertation, even though they have more than sufficient compute and sensing power. To protect each manufacturer's commercial interests and intellectual property, the drones deliberately lack open access and programmability. Even professional, expensive, ready-to-fly hardware often lacks interfaces to interfere with low-level flight control, only allowing waypoint-based navigation or switching preprogrammed behaviour. On the other hand, there are swarm applications, for example for drone-choreographies, which can consist of hundreds of drones, but these typically lack the sensors and computation power to be useful in a perception-drive formation. Typically, they follow a mostly pre-programmed choreography and are controlled centrally.

Although self-designed and self-built robots are sufficient for evaluation and proof-of-concept field studies, as demonstrated in this dissertation, the hardware needs to reach a higher level of technology-readiness. Technical and engineering difficulties such as vibration or electromagnetic interference create a hard entry barrier. These challenges are typically outside the field of experience of the researcher trying to collect the data, i.e. biologists.

Co-operations with research facilities and commercial vendors should be undertaken to bridge this gap, for example in the context of nationally or EU-funded research projects. Beyond that, only further practical experience informs the exact requirements and efficacy of robot and formation designs, especially regarding implementation details such as control interface, flight time, communication range, etc. Further field experiments need to be undertaken to iteratively improve robotic designs to be of greater use to scientists.

7.2 Interference with wildlife

A general problem in observing nature is that the act of observation can affect the outcome of the experiment. In the field of quantum physics, this is a fundamental law of nature. In biology or generally, when dealing with intelligent agents, capable of observing their environment themselves, the goal is to either become imperceptible by the subject, or, when this is not possible, to be perceptible but irrelevant to the outcome of the experiment. Imperceptibility is a hard constraint for flying vehicles, which can be easily perceived through both noise and vision over a long distance. Animals in particular are highly perceptive in the audible spectrum as well as to motions. Experiments [13] show that a response from wildlife is to be expected at distances outside the feasible perception ranges for many robots. If imperceptibility is not practically achievable, the goal must be to minimize the disturbance, that means to avoid being perceived as a threat and to generally minimize the change in behaviour caused by the robot. The exact relationship between robot behaviour and animal behaviour is extremely complex and challenging to research, especially since many animals-of-interest do remember encounters for extended times and sometimes even pass that information along [163]. However, it can be easier to determine if the behaviour in the presence of robots deviates from normal behaviour or shows signs of distress, at least if a species' typical distress behaviour is well known. As such, more research should be done to find out how animals have to be approached to not disturb them and observe them without affecting their behaviour noticeably. This needs to be established in future work, and results should inform robot designs as well as observation procedures.

7.3 Communication

The bandwidth required for communication between vehicles, required to observe a subject, is nearly constant. For a stationary or slow-moving subject, a single detection from 3 angles, once a second, is typically sufficient. For a subject that displays fast and erratic motion, more shared observations are necessary to accurately track it. This requirement for information does not increase with more robots: there are only more receivers, but no need for additional data. This means, with more robots in formation, each individual robot needs to make fewer detections and also requires less active communication. This changes in the presence of false positives or incorrect detections, as any false detection needs to be offset by a sufficient amount of correct detections to maintain shared knowledge of the truth through a Bayesian filter. Eliminating detections based on Bayesian estimation methods — like our implemented 5 sigma threshold — does not completely eliminate the problem of false positives, although it does limit the effect a single false positive has on the state estimate. Either way, a low but constant communication bandwidth must be guaranteed. If the environment inhibits communication completely for some periods of time, the state estimate is bound to diverge, down to the accuracy of

a single robot on its own. However, the communication bandwidth requirement grows linearly with the number of subjects tracked simultaneously, which is relevant if tracking groups, or herds of animals instead of individuals.

Another aspect is the bandwidth requirement and resilience against communication outages w.r.t. robot self pose estimation. Robots need to know where their peers are to i) optimize the formation around the subject and ii) avoid collisions. If a robot's position is not known to its peers, the formation will become suboptimal and collision avoidance can no longer be guaranteed. This is the case in a complete communication outage. Moreover, the bandwidth requirements increase linearly with the number of robots, as each robot needs to broadcast its position to all other robots. As with the subject information, the impact of missing peer position information depends on the length of the outage. If, as in our methods, the robots use a model predictive controller to predict their own and the peer positions, the peer information can be anticipated for some time. However, this will become increasingly incorrect over time, especially if the cause of the communication outage is not known. The non-communicating peer might be malfunctioning and be following a different trajectory than the model predicts.

Even under these considerations and when tracking multiple subjects with large formations, the communication requirements are well below the capabilities of existing medium-range wireless communication, e.g. Wi-Fi. Future work should focus on mitigation of communication failure, especially where relevant to collision-avoidance. The best course of action would be to passively sense robots — as well as other objects in the environment — that are on a potential collision course and avoid them through path planning. Methods such as simultaneous locating and mapping (SLAM) would furthermore increase awareness of the robot's surroundings and improve the self-pose estimate.

7.4 Non-holonomic formations

Formation control with highly motion-restricted and observation-direction-restricted aerial robots performed in this dissertation is easily generalizable to less restricted vehicles. Therefore, our methods can be applied beyond airships and multi-copters to very different vehicle types. Vehicles that have movement-restrictions very similar to airships include:

Fixed Wing Aircraft. Fixed wing aircraft typically control yaw-rate by bank angle in a coordinated turn, as opposed to airships, which turn with rudder and enter a bank as the result of aerodynamic centripetal forces, mass inertia and gravity, but the resulting motion is almost identical. Both move in 3D space, although fixed wing aircraft need to exert more energy to change their altitude, which limits vertical motion. This allows adaptation for fixed wing aircraft with reduced vertical motion, which is possible if the sensor or camera has a sufficient vertical field of view.

Helicopters. Helicopters have similarities with both multi-copters and fixed wing aircraft. Although helicopters can hover and move omnidirectionally at low speeds, at higher speeds their yaw-orientation self-corrects in flight-direction, and they can be approximated with a fixed-wing physical model. In either case, our approach is applicable.

Submarines. Submarine robots can be modelled as underwater airships. These vehicles, too, are buoyant and subject to very similar physics. The main challenges of underwater robotics are related to available sensors and communication, which justify further research.

Surface ships. Surface ships motion can be modelled as a special case of submarine motion, restricted to zero depth, although wave action can cause additional oscillation of their attitude. Since they cannot change their height or depth, adjustable sensors or cameras/sensors with a wide vertical field of view should be used. Doing so also compensates for wave motion and rolling motion when moving on a curved trajectory, which depends on the ship's hull-geometry.

Wheeled ground vehicles. The motion of wheeled ground vehicles, such as cars or rovers, is typically also non-holonomic and comparable to ships. Wheeled robots cannot adjust height, and they do not roll or bank, with the exception of motorcycles, which bank in the same way as airships. Nevertheless, wheeled ground vehicles also need height adjustable sensors or cameras/sensors with wide vertical field of view.

Each of these vehicle classes comes with their own inherent challenges regarding payload capacity, flight or mission time, vibration and disturbances, communication, speed, etc. These parameters typically offer trade-offs for their feasibility for subject observation under different circumstances and in different environments. For ad-hoc observation of animals in-situ, a robot should be lightweight and transportable, require a very short setup time and be VTOL capable. These requirements typically favour multi-copters. Long-term observation over a defined, fixed region often favours vehicles with high-efficiency and long mission durations, such as fixed wing aircraft and airships. As the transition from multi-copters to airships in this dissertation has shown, most perception aspects remain similar when switching platforms.

Future work should also look into mixed-mode observations, including heterogenous robot teams consisting of holonomic and non-holonomic vehicles. The challenge is to optimize the entire formation w.r.t. each vehicle's motion limitations. These can likely be addressed with an MPC-based approach, but there are open questions w.r.t. convexity of the optimization problem and convergence guarantees.

7.5 On-board computation

Hand-crafted feature detection, as common in computer vision before the deep learning revolution, still works reasonably well for well-defined, controlled markers, such as QR-codes or tags, as well as any active markers. For markerless detection, learning-based methods need to be used. All conclusions about communication and bandwidth in the previous section only hold if the raw sensor/camera data is processed on board. If the robot relies on off-board processing, raw data needs to be transferred wirelessly, processed, and the detection results either transferred back or broadcast to all robots. This requires a high-bandwidth low-latency communication link, and the bandwidth requirements scale linearly with the number of robots in formation. This is not feasible, as typical communication networks such as 5G or Wi-Fi share available communication bandwidth with all vehicles in an area. The available bandwidth becomes quickly saturated. Compression can alleviate this to a degree, but this, in turn, requires on board processing involving video encoder hardware. The computation power to compress raw video into a H264 or H265 stream is comparable to that of typical detection algorithms. As such, the only point that speaks in favour of off-board processing is that some commercial drones already come with optimized hardware for video compression and transmission to a ground station, for example, to be viewed by a human operator. Unfortunately, these drones typically cannot be freely programmed to autonomously run perception on-board. But the ability for long-range compressed video transmission can be exploited for a work-around. Using the existing video compression and transmission capabilities of commercially available consumer drones, for example "Parrot Anafi", the processing can be computed on a powerful ground station, which then also computes formation parameters and sends motion commands to the vehicles. This layout is practical for operation in close range to the base station and with limited formation sizes, and has the advantage that no specialized drone hardware is needed. It offers a work-around to apply some techniques covered in this dissertation with existing robotic hardware in the wild and justifies further evaluation. However, this should not distract from the notion that this approach is highly suboptimal based on underlying principles. On-board computation allows operation of robots with significantly lower bandwidth requirements and with much higher autonomy — that is, outside the range of a centralized high bandwidth base station. Any longer-term future work should focus on flying robots with on-board computation to avoid the bottlenecks of centralized computation, which we consider a stop-gap measure and suitable only for niche applications, where ubiquitous unlimited communication bandwidth is available. It should, however, be mentioned that the advent of high-bandwidth low-latency satellite internet [164] can greatly expand these niches. As such, open-mindedness towards potential off-board computation solutions should also be encouraged.

7.6 Iterative observation method improvements

Observing the environment is in itself an iterative learning process. The more we learn about the environment, the better we can model and predict. This in turn allows for better and more informed observation, which enables new insights. In other words, observation represents a chicken-and-egg problem, where knowledge about the environment needs to first be gained to effectively observe the environment, especially with autonomous methods. Throughout this dissertation, this paradigm has manifested through numerous aspects.

Visual detection. In Chapter 6 we detect horses with a trained convolutional neural network and use these detections for tracking and observation in formation as outlined in Chapters 2 and 5. To do so, the detector network needs to be bootstrapped. The method needs to be trained on annotated horse images which initially have to be collected manually, by human-piloted drones. As described in Chapter 3, about 15 minutes of manual human operated flight provide sufficient training data to enable autonomous detection and autonomous observation, which then allows collecting more data autonomously. The more training data becomes available, the more robust and reliable the detection becomes.

Data annotation. Once data is collected, ground-truth detection information must be annotated, as described in Chapter 3. While machine-assisted methods can accelerate the data-annotation process, this requires initial bootstrapping with fully manual annotation. The more annotation data becomes available, the more robust and reliable machine assisted annotation becomes.

Robotic hardware and equipment. As described in Chapters 2 and 6, collecting data with autonomous systems can iteratively be improved using experience and results collected in field experiments. Aspects such as recording equipment, sensors, sensor-fusion algorithms, but also Bayesian models for state estimation and prediction used in object tracking are informed and corrected using real-world data collected in field experiments. The more flight data becomes available, the more robust and reliable guidance and control hard- and software becomes.

Subject response and behaviour. As shown in [13, 149] animals we attempt to observe in turn perceive us, as well as our robotic systems. How animals respond and react to being observed is subject to ongoing and future research. Observation methods need to be improved and adjusted in turn to minimize interference while the interaction between observer and subject becomes better understood. The more subject response and behaviour data becomes available, the more robust and reliable observation strategies can become.

7.7 Closing remark

In addition to the concept of aerial robot formations outlined in this dissertation, there are, of course, a plethora of alternate observation methods, including classical tracking with GPS tracking-collars, remote sensing with aeroplanes and satellites, wildlife camera traps and other stationary sensors, manual observation by scientists in the field, etc., all of which inform and complement each other to maximize information gain. The methods presented in this dissertation offer only one route among many. Not one method can provide a complete picture of the world, and research always involves the attempt to piece together a complex picture from many small puzzle pieces, many of which might be missing, distorted, or yet incomprehensible. It is the author's hope that this work and the presented methods will help researchers to gain a few more puzzle pieces and piece together the model of nature a bit more. To facilitate that, all methods, source code and data described in this world have been made open-source and shared on our research group's repository on `https://github.com/robot-perception-group`.

Bibliography

[1] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: a skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), oct 2015. https://doi.org/10.1145/2816795.2818013.

[2] M. Madadi, H. Bertiche, and S. Escalera. Smplr: Deep learning based smpl reverse for 3d human pose and shape recovery. *Pattern Recognition*, 106:107472, 2020. https://doi.org/10.1016/j.patcog.2020.107472.

[3] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10967–10977, June 2019. https://doi.org/10.1109/CVPR.2019.01123.

[4] S. Zuffi, A. Kanazawa, D. W. Jacobs, and M. J. Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5524–5532, 2017. https://doi.org/10.1109/CVPR.2017.586.

[5] S. Zuffi, A. Kanazawa, and M. J. Black. Lions and tigers and bears: Capturing non-rigid, 3d, articulated shape from images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3955–3963, 2018. https://doi.org/10.1109/CVPR.2018.00416.

[6] M. C. Wong, B. K. Ng, I. Tian, S. Sobhiyeh, I. Pagano, M. Dechenaud, S. F. Kennedy, Y. E. Liu, N. Kelly, D. Chow, A. K. Garber, G. Maskarinec, S. Pujades, M. J. Black, B. Curless, S. B. Heymsfield, and J. A. Shepherd. A pose-independent method for accurate and precise body composition from 3d optical scans. *Obesity*, 29(11):1835–1847, Nov. 2021. https://doi.org/10.1002/oby.23256.

[7] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: a model of dynamic human shape in motion. *ACM Trans. Graph.*, 34(4), jul 2015. https://doi.org/10.1145/2766993.

[8] MPI IS. 4d dynamic scanner. http://web.archive.org/web/20230812024804/https://ps.is.mpg.de/pages/4d-capture, 2023. Accessed: 2023-08-12.

[9] S. Günel. *Using Animal Motion Capture to Learn Neural Representations*. PhD thesis, École polytechnique fédérale de Lausanne, Lausanne, 2022. https://doi.org/10.5075/epfl-thesis-9124.

[10] J. D. Marshall, T. Li, J. H. Wu, and T. W. Dunn. Leaving flatland: Advances in 3d behavioral measurement. *Current Opinion in Neurobiology*, 73:102522, 2022. https://doi.org/10.1016/j.conb.2022.02.002.

[11] T. Maeda and S. Yamamoto. Drone observation for the quantitative study of complex multilevel societies. *Animals*, 13(12), 2023. https://doi.org/10.3390/ani13121911.

[12] E. T. Campolettano, M. L. Bland, R. A. Gellner, D. W. Sproule, B. Rowson, A. M. Tyson, S. M. Duma, and S. Rowson. Ranges of injury risk associated with impact from unmanned aircraft systems. *Annals of Biomedical Engineering*, 45(12):2733–2741, Dec 2017. https://doi.org/10.1007/s10439-017-1921-6.

[13] E. Bennitt, H. L. A. Bartlam-Brooks, T. Y. Hubel, and A. M. Wilson. Terrestrial mammalian wildlife responses to unmanned aerial systems approaches. *Scientific Reports*, 9(1):2142, Feb 2019. https://doi.org/10.1038/s41598-019-38610-x.

[14] M. E. Lussier, J. M. Bradley, and C. Detweiler. *Extending Endurance of Multicopters: The Current State-of-the-Art*, In *AIAA Scitech 2019 Forum*. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan 2019. 0. https://doi.org/10.2514/6.2019-1790.

[15] E. Price, Y. T. Liu, M. J. Black, and A. Ahmad. Simulation and control of deformable autonomous airships in turbulent wind. In M. H. Ang Jr, H. Asama, W. Lin, and S. Foong, editors, *Intelligent Autonomous Systems 16*, pages 608–626, Cham, 2022. Springer International Publishing. https://doi.org/10.1007/978-3-030-95892-3_46.

[16] E. Price, M. J. Black, and A. Ahmad. Viewpoint-driven formation control of airships for cooperative target tracking. *IEEE Robotics and Automation Letters*, 8(6):3653–3660, June 2023. https://doi.org/10.1109/LRA.2023.3264727.

[17] M. Siegel. The sense-think-act paradigm revisited. In *1st International Workshop on Robotic Sensing, 2003. ROSE' 03.*, pages 5 pp.–, 2003. https://doi.org/10.1109/ROSE.2003.1218700.

[18] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, pages 1–20, 2023. https://doi.org/10.1109/JPROC.2023.3238524.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. https://doi.org/10.1007/978-3-319-46448-0_2.

[20] E. Price, G. Lawless, R. Ludwig, I. Martinovic, H. H. Bülthoff, M. J. Black, and A. Ahmad. Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles. *IEEE Robotics and Automation Letters*, 3(4):3193–3200, Oct 2018. https://doi.org/10.1109/LRA.2018.2850224.

[21] Robot perception Group. Git repository: Aircap. https://github.com/robot-perception-group/AirCap, 2018.

[22] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Bülthoff, M. J. Black, and A. Ahmad. Active perception based formation control for multiple aerial vehicles. *IEEE Robotics and Automation Letters*, 4(4):4491–4498, Oct 2019. https://doi.org/10.1109/LRA.2019.2932570.

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010. https://doi.org/10.1007/s11263-009-0275-4.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*. European Conference on Computer Vision, September 2014. https://www.microsoft.com/en-us/research/publication/microsoft-coco-common-objects-in-context/.

[25] E. Price. Smarter-labelme source code. https://github.com/robot-perception-group/smarter-labelme, 2023.

[26] D. Gordon, A. Farhadi, and D. Fox. Re^3: Real-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, April 2018. https://doi.org/10.1109/LRA.2018.2792152.

[27] E. Price and A. Ahmad. Accelerated video annotation driven by deep detector and tracker. In S.-G. Lee, J. An, N. Y. Chong, M. Strand, and J. H. Kim, editors, *Intelligent Autonomous Systems 18*, pages 141–153, Cham, 2024. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-44981-9_12.

[28] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016. https://doi.org/10.1007/978-3-319-26054-9_23.

[29] Fraunhofer FKIE. Fkie multimaster. https://web.archive.org/web/20240304042507/http://fkie.github.io/multimaster_fkie/, 2024.

[30] LibrePilot. Archived web page: Openpilot revolution. http://tinyurl.com/ARCHIVEDOPREVO, 2024.

[31] librepilot.org. Archived web page: Librepilot. https://web.archive.org/web/20201112014955/https://www.librepilot.org/site/index.html, 2015. Accessed: 2020-11-12.

[32] N. Otterness, M. Yang, S. Rust, E. Park, J. H. Anderson, F. D. Smith, A. Berg, and S. Wang. An evaluation of the nvidia tx1 for supporting real-time computer-vision workloads. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 353–364, 2017. https://doi.org/10.1109/RTAS.2017.3.

[33] C. Sciortino and A. Fagiolini. Ros/gazebo-based simulation of quadcopter aircrafts. In *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)*, pages 1–6, 2018. https://doi.org/10.1109/RTSI.2018.8548411.

[34] Y. Jessie C. Implementation and testing of turbulence models for the f18-harv simulation. Technical report, NASA, 1998. https://ntrs.nasa.gov/citations/19980028448.

[35] UNESCO. Archived web page: Hortobágy national park. https://web.archive.org/web/20240125130845/https://whc.unesco.org/en/list/474, 2024.

[36] A. Ahmad, E. Price, R. Tallamraju, N. Saini, G. Lawless, R. Ludwig, I. Martinovic, H. Bülthoff, and M. Black. Aircap–aerial outdoor motion capture. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Workshop on Aerial Swarms*, 2019. https://ps.is.mpg.de/publications/aircap2019aerialswarms.

[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 675–678, New York, NY, USA, 2014. Association for Computing Machinery. https://doi.org/10.1145/2647868.2654889.

[38] LibrePilot and Robot Perception Group. Git repository: Ros enabled librepilot source code for airships. https://github.com/robot-perception-group/LibrePilot/tree/airship_control_noetic, 2024.

[39] N. Saini, E. Price, R. Tallamraju, R. Enficiaud, R. Ludwig, I. Martinovic, A. Ahmad, and M. Black. Markerless outdoor human motion capture using multiple autonomous micro aerial vehicles. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 823–832, 2019. https://doi.org/10.1109/ICCV.2019.00091.

[40] Y. T. Liu, E. Price, M. J. Black, and A. Ahmad. Deep residual reinforcement learning based autonomous blimp control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12566–12573, 2022. https://doi.org/10.1109/IROS47612.2022.9981182.

[41] N. Saini, E. Bonetto, E. Price, A. Ahmad, and M. J. Black. AirPose: Multi-view fusion network for aerial 3D human pose and shape estimation. *IEEE Robotics and Automation Letters*, 7(2):4805 – 4812, Apr. 2022. Also accepted and presented in the 2022 IEEE International Conference on Robotics and Automation (ICRA). https://doi.org/10.1109/LRA.2022.3145494.

[42] NVIDIA. Data sheet: Jetson tx2. http://tinyurl.com/ARCHIVEDNVTX2DS, 2024.

[43] E. Price and A. Ahmad. Airship formations for animal motion capture and behavior analysis. In M. Tripathi and K. M. Kiran Babu, editors, *Lighter Than Air Systems*, pages 179–190, Singapore, 2025. Springer Nature Singapore. https://doi.org/10.1007/978-981-96-8101-3_11.

[44] E. Price, P. C. Khandelwal, D. I. Rubenstein, and A. Ahmad. A framework for fast, large-scale, semi-automatic inference of animal behavior from monocular videos. *bioRxiv*, 2023. https://doi.org/10.1101/2023.07.31.551177.

[45] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. von Stryk, S. Roth, and B. Schiele. Vision based victim detection from unmanned aerial vehicles. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1740–1747, Oct 2010. https://doi.org/10.1109/IROS.2010.5649223.

[46] M. Consortium. Multidrone: H2020-ict-2016-2017 h2020-ict-2016-1 - robotics and artificial intelligence project. https://multidrone.eu/multidrone-in-short/, 2008.

[47] G. A. M. d. Santos, Z. Barnes, E. Lo, B. Ritoper, L. Nishizaki, X. Tejeda, A. Ke, H. Lin, C. Schurgers, A. Lin, and R. Kastner. Small unmanned aerial vehicle system for wildlife radio collar tracking. In *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 761–766, Oct 2014. https://doi.org/10.1109/MASS.2014.48.

[48] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. https://doi.org/10.1016/j.neunet.2014.09.003.

[49] F. Falcini, G. Lami, and A. M. Costanza. Deep learning in automotive software. *IEEE Software*, 34(3):56–63, 2017. https://doi.org/10.1109/MS.2017.79.

[50] J. Hosang, M. Omran, R. Benenson, and B. Schiele. Taking a deeper look at pedestrians. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4073–4082, 2015. https://doi.org/10.1109/CVPR.2015.7299034.

[51] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693, 2014. https://doi.org/10.1109/CVPR.2014.471.

[52] F. Bogo, M. J. Black, M. Loper, and J. Romero. Detailed full-body reconstructions of moving people from monocular rgb-d sequences. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2300–2308, 2015. https://doi.org/10.1109/ICCV.2015.265.

[53] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović. Practical motion capture in everyday surroundings. *ACM Trans. Graph.*, 26(3), July 2007. https://doi.org/10.1145/1276377.1276421.

[54] T. von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll. Sparse inertial poser: Automatic 3d human pose estimation from sparse imus. *Computer Graphics Forum*, 36(2):349–360, 2017. https://doi.org/10.1111/cgf.13131.

[55] L. Xu, Y. Liu, W. Cheng, K. Guo, G. Zhou, Q. Dai, and L. Fang. Flycap: Markerless motion capture using multiple autonomous flying cameras. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2018. https://doi.org/10.1109/TVCG.2017.2728660.

[56] Y. Liu, J. Gall, C. Stoll, Q. Dai, H.-P. Seidel, and C. Theobalt. Markerless motion capture of multiple characters using multiview image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2720–2735, 2013. https://doi.org/10.1109/TPAMI.2013.47.

[57] S. S. Dias and M. G. S. Bruno. Cooperative target tracking using decentralized particle filtering and rss sensors. *IEEE Transactions on Signal Processing*, 61(14):3632–3646, July 2013. https://doi.org/10.1109/TSP.2013.2262276.

[58] A. Ahmad, G. Lawless, and P. Lima. An online scalable approach to unified multirobot cooperative localization and object tracking. *IEEE Transactions on Robotics*, PP(99):1–16, 2017. https://doi.org/10.1109/TRO.2017.2715342.

[59] Z. Wang and D. Gu. Cooperative target tracking control of multiple robots. *IEEE Transactions on Industrial Electronics*, 59(8):3232–3240, Aug 2012. https://doi.org/10.1109/TIE.2011.2146211.

[60] K. Hausman, J. Müller, A. Hariharan, N. Ayanian, and G. S. Sukhatme. *Cooperative Control for Target Tracking with Onboard Sensing*, In M. A. Hsieh, O. Khatib, and V. Kumar, editors, *Experimental Robotics: The 14th International Symposium on Experimental Robotics*, pages 879–892. Springer International Publishing, Cham, 2016. https://doi.org/10.1007/978-3-319-23778-7_58.

[61] W. Zheng-Jie and L. Wei. A solution to cooperative area coverage surveillance for a swarm of mavs. *International Journal of Advanced Robotic Systems*, 10(12):398, 2013. https://doi.org/10.5772/56801.

[62] M. Zhang and H. H. T. Liu. Cooperative tracking a moving target using multiple fixed-wing uavs. *J. Intell. Robotics Syst.*, 81(3-4):505–529, Mar. 2016. https://doi.org/10.1007/s10846-015-0236-9.

[63] C. Fu, R. Duan, D. Kircali, and E. Kayacan. Onboard robust visual tracking for UAVs using a reliable Global-Local object model. *Sensors (Basel)*, 16(9), Aug. 2016. https://doi.org/10.3390/s16091406.

[64] S. Rallapalli, H. Qiu, A. Bency, S. Karthikeyan, R. Govindan, B. Manjunath, and R. Urgaonkar. Are very deep neural networks feasible on mobile devices. *IEEE Trans. Circ. Syst. Video Technol*, 2016. https://api.semanticscholar.org/CorpusID:6621072.

[65] D. C. De Oliveira and M. A. Wehrmeister. Towards real-time people recognition on aerial imagery using convolutional neural networks. In *2016 IEEE 19th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 27–34, 2016. https://doi.org/10.1109/ISORC.2016.14.

[66] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. https://doi.org/10.1109/CVPR.2016.91.

[67] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis*

and Machine Intelligence, 39(6):1137–1149, 2017. https://doi.org/10.1109/TPAMI.2016.2577031.

[68] A. Ahmad, E. Ruff, and H. H. Bülthoff. Dynamic baseline stereo vision-based cooperative target tracking. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1728–1734, 2016. https://ieeexplore.ieee.org/abstract/document/7528092.

[69] M. Campbell and W. Whitacre. *Cooperative Geolocation and Sensor Bias Estimation for UAVs with Articulating Cameras*, In *AIAA Guidance, Navigation, and Control Conference*. https://doi.org/10.2514/6.2009-6220.

[70] Y. Liu, S. Rajappa, J. M. Montenbruck, P. Stegagno, H. Bülthoff, F. Allgöwer, and A. Zell. Robust nonlinear control approach to nontrivial maneuvers and obstacle avoidance for quadrotor uav under disturbances. *Robotics and Autonomous Systems*, 98:317 – 332, 2017. https://doi.org/https://doi.org/10.1016/j.robot.2017.08.011.

[71] P. U. Lima, A. Ahmad, A. Dias, A. G. Conceição, A. P. Moreira, E. Silva, L. Almeida, L. Oliveira, and T. P. Nascimento. Formation control driven by cooperative object tracking. *Robotics and Autonomous Systems*, 63:68–79, 2015. https://doi.org/10.1016/j.robot.2014.08.018.

[72] J. Alonso-Mora, S. Baker, and D. Rus. Multi-robot navigation in formation via sequential convex programming. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4634–4641, 2015. https://doi.org/10.1109/IROS.2015.7354037.

[73] H. Fukushima, K. Kon, and F. Matsuno. Model predictive formation control using branch-and-bound compatible with collision avoidance problems. *IEEE Transactions on Robotics*, 29(5):1308–1317, 2013. https://doi.org/10.1109/TRO.2013.2262751.

[74] F. Guan, L. Peng, L. Perneel, and M. Timmerman. Open source freertos as a case study in real-time operating system evolution. *Journal of Systems and Software*, 118:19–35, 2016. https://doi.org/10.1016/j.jss.2016.04.063.

[75] I. Susnea and M. Mitescu. *Using The I2C Bus*, In *Microcontrollers in Practice*, pages 61–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. https://doi.org/10.1007/3-540-28308-0_5.

[76] R. Tallamraju, N. Saini, E. Bonetto, M. Pabst, Y. T. Liu, M. Black, and A. Ahmad. Aircaprl: Autonomous aerial human motion capture using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6678 – 6685, Oct. 2020. Also accepted and presented in the 2020 IEEE/RSJ International Conference on

Intelligent Robots and Systems (IROS). https://doi.org/10.1109/LRA.2020.3013906.

[77] B. Friedland. Steady-state behavior of kalman filter with discrete- and continuous-time observations. *IEEE Transactions on Automatic Control*, 25(5):988–992, 1980. https://doi.org/10.1109/TAC.1980.1102474.

[78] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60(2):337–345, Oct 2015. https://doi.org/10.1007/s11235-015-0034-5.

[79] J. Yang and L. Wang. Feature fusion and enhancement for single shot multibox detector. In *2019 Chinese Automation Congress (CAC)*, pages 2766–2770, 2019. https://doi.org/10.1109/CAC48633.2019.8996582.

[80] M. Swapna, D. K. Sharma, and D. B. Prasad. Cnn architectures: Alex net, le net, vgg, google net, res net, Mar. 2020. https://doi.org/10.35940/ijrte.f9532.038620.

[81] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019. https://doi.org/https://doi.org/10.1016/j.patcog.2019.01.006.

[82] L. Yang, G. Chen, and W. Ci. Multiclass objects detection algorithm using darknet-53 and densenet for intelligent vehicles. *EURASIP Journal on Advances in Signal Processing*, 2023(1):85, Aug 2023. https://doi.org/10.1186/s13634-023-01045-8.

[83] M. Sohan, T. Sai Ram, and C. V. Rami Reddy. A review on yolov8 and its advancements. In I. J. Jacob, S. Piramuthu, and P. Falkowski-Gilski, editors, *Data Intelligence and Cognitive Informatics*, pages 529–545, Singapore, 2024. Springer Nature Singapore. https://doi.org/10.1007/978-981-99-7962-2_39.

[84] K. Alhazmi, W. Alsumari, I. Seppo, L. Podkuiko, and M. Simon. Effects of annotation quality on model performance. In *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 063–067, 2021. https://doi.org/10.1109/ICAIIC51459.2021.9415271.

[85] E. Gaur, V. Saxena, and S. K. Singh. Video annotation tools: A review. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 911–914, Oct 2018. https://doi.org/10.1109/ICACCCN.2018.8748669.

[86] A. Dutta and A. Zisserman. The via annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*,

MM '19, page 2276–2279, New York, NY, USA, 2019. Association for Computing Machinery. https://doi.org/10.1145/3343031.3350535.

[87] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, May 2008. https://doi.org/10.1007/s11263-007-0090-8.

[88] G. D. Evangelidis and E. Z. Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, Oct 2008. https://doi.org/10.1109/TPAMI.2008.113.

[89] B. Ajani. Smart-labelme: Video / Image Annotation (Polygon, Semantic mask, Classification) with Python. https://github.com/bhavyaajani/smart-labelme, 2020.

[90] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov. Label Studio: Data labeling software, 2020-2022. Open source software available from https://github.com/heartexlabs/label-studio.

[91] B. Sekachev, N. Manovich, M. Zhiltsov, A. Zhavoronkov, D. Kalinin, B. Hoff, TOsmanov, D. Kruchinin, A. Zankevich, DmitriySidnev, M. Markelov, Johannes222, M. Chenuet, a andre, telenachos, A. Melnikov, J. Kim, L. Ilouz, N. Glazov, Priya4607, R. Tehrani, S. Jeong, V. Skubriev, S. Yonekura, vugia truong, zliang7, lizhming, and T. Truong. opencv/cvat: v1.1.0, Aug 2020. https://doi.org/10.5281/zenodo.4009388.

[92] E. Price and et. al. Smarter-labelme dataset. https://tinyurl.com/SLRDa1, 2024.

[93] K. Wada. labelme: Image Polygonal Annotation with Python. https://github.com/wkentaro/labelme, 2016.

[94] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek. A brief introduction to opencv. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 1725–1730, May 2012. https://ieeexplore.ieee.org/document/6240859.

[95] X. Wu, D. Sahoo, and S. C. Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020. https://doi.org/10.1016/j.neucom.2020.01.085.

[96] H. Nakahara, H. Yonekawa, and S. Sato. An object detector based on multiscale sliding window search using a fully pipelined binarized cnn on an fpga. In *2017 International Conference on Field Programmable Technology (ICFPT)*, pages 168–175, Dec 2017. https://doi.org/10.1109/FPT.2017.8280135.

[97] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung. Handcrafted and deep trackers: Recent visual object tracking approaches and trends. *ACM Comput. Surv.*, 52(2), apr 2019. https://doi.org/10.1145/3309665.

[98] S. Javed, M. Danelljan, F. S. Khan, M. H. Khan, M. Felsberg, and J. Matas. Visual object tracking with discriminative filters and siamese networks: A survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2022. https://doi.org/10.1109/TPAMI.2022.3212594.

[99] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr. Fast online object tracking and segmentation: A unifying approach. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1328–1338, 2019. https://doi.org/10.1109/CVPR.2019.00142.

[100] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu. Transformer tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8122–8131, 2021. https://doi.org/10.1109/CVPR46437.2021.00803.

[101] H. Bai, W. Cheng, P. Chu, J. Liu, K. Zhang, and H. Ling. Gmot-40: A benchmark for generic multiple object tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6715–6724, 2021. https://doi.org/10.1109/CVPR46437.2021.00665.

[102] L. Čehovin, M. Kristan, and A. Leonardis. Is my new tracker really better than yours? In *IEEE Winter Conference on Applications of Computer Vision*, pages 540–547, 2014. https://doi.org/10.1109/WACV.2014.6836055.

[103] GMOT-40. Gmot-40 leaderboard, 2021. Accessed: 2024-01-17. https://web.archive.org/web/20240117130156/https://spritea.github.io/GMOT40/result.html.

[104] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, Oct 2023. https://doi.org/10.1109/ICCV51070.2023.00371.

[105] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, Nov 2016. https://doi.org/10.1109/TPAMI.2016.2516982.

[106] R. Khanam and M. Hussain. Yolov11: An overview of the key architectural enhancements, 2024. https://arxiv.org/abs/2410.17725.

[107] J. Lauer, M. Zhou, S. Ye, W. Menegas, S. Schneider, T. Nath, M. M. Rahman, V. Di Santo, D. Soberanes, G. Feng, V. N. Murthy, G. Lauder, C. Dulac, M. W. Mathis, and A. Mathis. Multi-animal pose estimation, identification and tracking with deeplabcut. *Nature Methods*, 19(4):496–504, Apr 2022. https://doi.org/10.1038/s41592-022-01443-0.

[108] V. Kerekes, I. Sándor, D. Nagy, K. Ozogány, L. Göczi, B. Ibler, L. Széles, and Z. Barta. Trends in demography, genetics, and social structure of przewalski's horses in the hortobagy national park, hungary over the last 22 years. *Global Ecology and Conservation*, 25:e01407, 2021. https://doi.org/10.1016/j.gecco.2020.e01407.

[109] L. Sahawneh and M. A. Jarrah. Development and calibration of low cost mems imu for uav applications. In *2008 5th International Symposium on Mechatronics and Its Applications*, pages 1–9, 2008. https://doi.org/10.1109/ISMA.2008.4648819.

[110] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz. A survey of open-source uav flight controllers and flight simulators. *Microprocessors and Microsystems*, 61:11 – 20, 2018. https://doi.org/https://doi.org/10.1016/j.micpro.2018.05.002.

[111] A. R. Perry. The flightgear flight simulator. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '04, page 31, USA, 2004. USENIX Association. https://www.usenix.org/conference/2004-usenix-annual-technical-conference/flightgear-flight-simulator.

[112] A. Elfes, S. S. Bueno, J. J. G. Ramos, E. C. de Paiva, M. Bergerman, J. R. H. Carvalho, S. M. Maeta, L. G. B. Mirisola, B. G. Faria, and J. R. Azinheira. Modelling, control and perception for an autonomous robotic airship. In G. D. Hager, H. I. Christensen, H. Bunke, and R. Klein, editors, *Sensor Based Intelligent Robots*, pages 216–244, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45993-6_13.

[113] T. Fukao, K. Fujitani, and T. Kanade. Image-based tracking control of a blimp. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 5, pages 5414–5419 Vol.5, 2003. https://doi.org/10.1109/CDC.2003.1272498.

[114] J. Ko, D. J. Klein, D. Fox, and D. Haehnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 742–747, 2007. https://doi.org/10.1109/ROBOT.2007.363075.

[115] A. Rottmann, C. Plagemann, P. Hilgers, and W. Burgard. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1895–1900, 2007. https://doi.org/10.1109/IROS.2007.4399531.

[116] F. Repoulias and E. Papadopoulos. Robotic airship trajectory tracking control using a backstepping methodology. In *2008 IEEE International Conference on Robotics and Automation*, pages 188–193, 2008. https://doi.org/10.1109/ROBOT.2008.4543207.

[117] Hong Zhang and J. P. Ostrowski. Visual servoing with dynamics: control of an unmanned blimp. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 618–623 vol.1, 1999. https://doi.org/10.1109/ROBOT.1999.770044.

[118] E. C. de Paiva, S. S. Bueno, S. B. V. Gomes, J. J. G. Ramos, and M. Bergerman. A control system development environment for aurora's semi-autonomous robotic airship. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 3, pages 2328–2335 vol.3, 1999. https://doi.org/10.1109/ROBOT.1999.770453.

[119] Y. Li, M. Nahon, and I. Sharf. Airship dynamics modeling: A literature review. *Progress in Aerospace Sciences*, 47(3):217 – 239, 2011. https://doi.org/https://doi.org/10.1016/j.paerosci.2010.10.001.

[120] J. J. G. Ramos, S. M. Maeta, M. Bergerman, S. S. Bueno, L. G. B. Mirisola, and A. Bruciapaglia. Development of a vrml/java unmanned airship simulating environment. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. (Cat. No.99CH36289)*, volume 3, pages 1354–1359 vol.3, 1999. https://doi.org/10.1109/IROS.1999.811668.

[121] R. Jones. The aerodynamical characteristics of the airship as deduced from experiments on models, with application to motion in a horizontal plane. *Journal of Royal Aeronautical Society*, 28(158):88–150, 1924. https://archive.org/details/sim_aeronautical-journal_1924_28_index.

[122] T. Lutz, P. Funk, A. Jakobi, and S. Wagner. Summary of aerodynamic studies on the lotte airship. In *4th International Airship Convention and Exhibition*, pages 28–31, 2002. https://www.yumpu.com/s/5JnLWrKLlHkfuZuS.

[123] S. P. Jones and J. D. DeLaurier. Aerodynamic estimation techniques for aerostats and airships. *Journal of Aircraft*, 20(2):120–126, 1983. https://doi.org/10.2514/3.44840.

[124] L. S. de Coelho, M. F. M. Campos, and V. Kumar. Computer vision-based navigation for autonomous blimps. In *Proceedings SIBGRAPI'98. International Symposium on Computer Graphics, Image Processing, and Vision (Cat. No.98EX237)*, pages 287–294, 1998. https://doi.org/10.1109/SIBGRA.1998.722762.

[125] T. Fukao, K. Fujitani, and T. Kanade. An autonomous blimp for a surveillance system. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, pages 1820–1825 vol.2, 2003. https://doi.org/10.1109/IROS.2003.1248908.

[126] S. Q. Liu, Y. J. Sang, and J. F. Whidborne. Adaptive sliding-mode-backstepping trajectory tracking control of underactuated airships. *Aerospace Science and Technology*, 97:105610, 2020. https://doi.org/10.1016/j.ast.2019.105610.

[127] L. Cheng, Z. Zuo, J. Song, and X. Liang. Robust three-dimensional path-following control for an under-actuated stratospheric airship. *Advances in Space Research*, 63(1):526–538, 2019. https://doi.org/10.1016/j.asr.2018.09.008.

[128] J. L. LIVESEY. The behavior of transverse cylindrical and forward facing total pressure probes in transverse total pressure gradients. *Journal of the Aeronautical Sciences*, 23(10):949–955, 1956. https://doi.org/10.2514/8.3695.

[129] L. Dubois and S. Suzuki. Formation control of multiple quadcopters using model predictive control. *Advanced Robotics*, 32(19):1037–1046, 2018. https://doi.org/10.1080/01691864.2018.1470572.

[130] K. Yamamoto, K. Sekiguchi, and K. Nonaka. Experimental verification of formation control by model predictive control considering collision avoidance in three dimensional space with quadcopters. In *2017 11th Asian Control Conference (ASCC)*, pages 1602–1607, Dec 2017. https://doi.org/10.1109/ASCC.2017.8287413.

[131] B. Lindqvist, P. Sopasakis, and G. Nikolakopoulos. A scalable distributed collision avoidance scheme for multi-agent uav systems. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9212–9218, 2021. https://doi.org/10.1109/IROS51168.2021.9636293.

[132] H. Fukushima, K. Kon, F. Matsuno, Y. Hada, K. Kawabata, and H. Asama. Constrained model predictive control: Applications to multi-vehicle formation and an autonomous blimp. In *2006 SICE-ICASE International Joint Conference*, pages 4515–4520, Oct 2006. https://doi.org/10.1109/SICE.2006.315039.

[133] E. Bicho, A. Moreira, S. Diegues, M. P. Carvalheira, and S. Monteiro. Airship formation control. In *3rd International Conference on Informatics in Control, Automation and Robotics - Workshop on Multi-Agent Robotic Systems (MARS 2006)*, pages 22–33, 2006. https://hdl.handle.net/1822/19247.

[134] Z. He, J.-X. Xu, S. Yang, Q. Ren, and X. Deng. On trackability of a moving target by fixed-wing uav using geometric approach. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 1572–1577, June 2014. https://doi.org/10.1109/ISIE.2014.6864849.

[135] S. lin Liao, R. ming Zhu, N. qi Wu, T. A. Shaikh, M. Sharaf, and A. M. Mostafa. Path planning for moving target tracking by fixed-wing uav. *Defence Technology*, 16(4):811–824, 2020. https://doi.org/10.1016/j.dt.2019.10.010.

[136] T. Z. Muslimov and R. A. Munasypov. Multi-uav cooperative target tracking via consensus-based guidance vector fields and fuzzy mrac. *Aircraft Engineering and Aerospace Technology*, 93(7):1204–1212, Jan 2021. https://doi.org/10.1108/AEAT-02-2021-0058.

[137] M. Zhang and H. H. T. Liu. Cooperative tracking a moving target using multiple fixed-wing uavs. *Journal of Intelligent & Robotic Systems*, 81(3):505–529, Mar 2016. https://doi.org/10.1007/s10846-015-0236-9.

[138] T. Z. Muslimov and R. A. Munasypov. Coordinated uav standoff tracking of moving target based on lyapunov vector fields. In *2020 International Conference Nonlinearity, Information and Robotics (NIR)*, pages 1–5, Dec 2020. https://doi.org/10.1109/NIR50484.2020.9290189.

[139] R. Takei, R. Tsai, Z. Zhou, and Y. Landa. An efficient algorithm for a visibility-based surveillance-evasion game. *Communications in Mathematical Sciences*, (7):1303–1327, 2014. https://doi.org/10.4310/CMS.2014.v12.n7.a7.

[140] Z. Zhou, J. R. Shewchuk, D. Stipanović, H. Huang, and C. J. Tomlin. Smarter lions: Efficient cooperative pursuit in general bounded arenas. *SIAM Journal on Control and Optimization*, 58(2):1229–1256, 2020. https://doi.org/10.1137/17M1152589.

[141] Z. Zhou, W. Zhang, J. Ding, H. Huang, D. M. Stipanović, and C. J. Tomlin. Cooperative pursuit with voronoi partitions. *Automatica*, 72:64–72, 2016. https://doi.org/10.1016/j.automatica.2016.05.007.

[142] T. Von Kármán. *Aerodynamics*. McGraw-Hill paperbacks : science, mathematics and engineering. McGraw-Hill, 1963. https://www.scribd.com/document/326488978/Theodore-Von-Karman-Aerodynamics-McGraw-Hill-Education-1963.

[143] P. Sopasakis, E. Fresk, and P. Patrinos. Open: Code generation for embedded nonconvex optimization. *IFAC-PapersOnLine*, 53(2):6548–6554, 2020. 21st IFAC World Congress. https://doi.org/10.1016/j.ifacol.2020.12.071.

[144] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos. A simple and efficient algorithm for nonlinear model predictive control. In *IEEE Conference on Decision and Control (CDC)*, pages 1939–1944, Dec 2017. https://doi.org/10.1109/CDC.2017.8263933.

[145] Robot Perception Group. Git repository: Simulation and control demo code. https://github.com/robot-perception-group/Airship-MPC, 2024.

[146] Windreiter. Archived web page: Windreiter. https://web.archive.org/web/20240125103634/https://www.windreiter.com/, 2024.

[147] S. Mazumdar. *Drone Applications in Wildlife Research—A Synoptic Review*, In *Environmental Informatics: Challenges and Solutions*, pages 237–257. Springer Nature Singapore, Singapore, 2022. https://doi.org/10.1007/978-981-19-2083-7_14.

[148] D. Chabot and D. M. Bird. Wildlife research and management methods in the 21st century: Where do unmanned aircraft fit in? *Journal of Unmanned Vehicle Systems Virtual Issue*, 01(01):137–155, 2016. https://doi.org/10.1139/juvs-2015-0021.

[149] M. Mo and K. Bonatakis. Approaching wildlife with drones: using scientific literature to identify factors to consider for minimising disturbance. *Australian Zoologist*, 42(1):1–29, 07 2021. https://doi.org/10.7882/AZ.2021.015.

[150] L. Bauersfeld and D. Scaramuzza. Range, endurance, and optimal speed estimates for multicopters. *IEEE Robotics and Automation Letters*, 7(2):2953–2960, 2022. https://doi.org/10.1109/LRA.2022.3145063.

[151] S. S. Bhat, S. G. Anavatti, M. Garratt, and S. Ravi. Review of autonomous outdoor blimps and their applications. *Drone Systems and Applications*, 12:1–21, 2024. https://doi.org/10.1139/dsa-2023-0052.

[152] H. B. Rinker, M. D. Lowman, and M. W. Moffett. Africa from the treetops. *The American Biology Teacher*, 57(7):393–401, 1995. http://www.jstor.org/stable/4450028.

[153] K. Adams, A. Broad, D. Ruiz-García, and A. R. Davis. Continuous wildlife monitoring using blimps as an aerial platform: a case study observing marine megafauna. *Australian Zoologist*, 40(3):407–415, 05 2020. https://doi.org/10.7882/AZ.2020.004.

[154] W. Cross. *Zeppelins of World War I*. Barnes & Noble Books, 1993. https://www.barnesandnoble.com/w/zeppelins-of-world-war-i-wilbur-cross/1004811394.

[155] L. Frobenius. *Die Möglichkeit einer Deutsch-Inner-Afrikanischen Luftflottenstation : (erster Bericht über die Studienergebnisse der Motorkommission der "Deutschen Inner-Afrikanischen Forschungs-Expedition" vorgetragen im Reichstage am 5. Februar 1913)*. Verlagsbuchhandlung Wilhelm Süsserott, Berlin, 1913. http://publikationen.ub.uni-frankfurt.de/frontdoor/index/index/docId/59274.

[156] S. B. Nelson. Airships in the arctic. *ARCTIC*, 46(3):278–283, 1993. https://doi.org/https://doi.org/10.14430/arctic1353.

[157] M. Gerke, U. Borgolte, I. Masár, F. Jelenciak, P. Bahnik, and N. Al-Rashedi. Lighter-than-air uavs for surveillance and environmental monitoring. In N. Aschenbruck, P. Martini, M. Meier, and J. Tölle, editors, *Future Security*, pages 480–483, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-33161-9_69.

[158] S. D. Ilcev and I. Skoryk. Automatic landing of animal tracking lighter-than-air systems (ltas) on the recharging platform. In *2014 24th International Crimean Conference Microwave & Telecommunication Technology*, pages 275–277, 2014. https://doi.org/10.1109/CRMICO.2014.6959391.

[159] Logitech. Data sheet: Brio web cam. https://web.archive.org/web/20240125095001/https://www.logitech.com/content/dam/logitech/en_us/video-collaboration/pdf/brio-datasheet.pdf, 2024.

[160] Auvidea. Data sheet: J120 carrier board. http://web.archive.org/web/20240125100516/https://auvidea.eu/download/manual/J120/J120_technical_reference_1.6.pdf, 2024.

[161] Graupner. Data sheet: Gr-16. https://web.archive.org/web/20240125100845/https://controlhobbies.s3-us-west-1.amazonaws.com/Manuals/GR_12_16_24_32_Receiver_Manual_EN.pdf, 2024.

[162] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009. https://www.andrewng.org/publications/ros-an-open-source-robot-operating-system/.

[163] H. N. Cornell, J. M. Marzluff, and S. Pecoraro. Social learning spreads knowledge about dangerous humans among american crows. *Proceedings of the Royal Society B: Biological Sciences*, 279(1728):499–508, 2012. https://doi.org/10.1098/rspb.2011.0957.

[164] F. Michel, M. Trevisan, D. Giordano, and O. Bonaventure. A first look at starlink performance. In *Proceedings of the 22nd ACM Internet Measurement Conference*,

IMC '22, page 130–136, New York, NY, USA, 2022. Association for Computing Machinery. https://doi.org/10.1145/3517745.3561416.