EPFL

Thèse n° 9186

Vision-based Flocking in Aerial Robot Swarms

Présentée le 25 mars 2022

Faculté des sciences et techniques de l'ingénieur
Laboratoire de systèmes intelligents
Programme doctoral en robotique, contrôle et systèmes intelligents

pour l'obtention du grade de Docteur ès Sciences

par

Fabian Maximilian SCHILLING

Acceptée sur proposition du jury

Prof. F. Mondada, président du jury
Prof. D. Floreano, directeur de thèse
Dr M. Saska, rapporteur
Prof. L. Gambardella, rapporteur
Prof. A. Alahi, rapporteur

École polytechnique fédérale de Lausanne

2022

For instance, on the planet Earth, man had always assumed that he was more intelligent than dolphins because he had achieved so much — the wheel, New York, wars and so on — whilst all the dolphins had ever done was muck about in the water having a good time.
But conversely, the dolphins had always believed that they were far more intelligent than man — for precisely the same reasons.

— Douglas Adams, "The Hitchhiker's Guide to the Galaxy"

To my grandparents...

Acknowledgments

I am incredibly grateful to all the people without whom this thesis would not have been possible. I would first like to express my sincere gratitude to my thesis director, Prof. Dario Floreano, for his continued trust, support, and guidance throughout my PhD. Thank you so much for welcoming me into your Laboratory of Intelligent Systems — a stimulating and diverse research environment that allowed me to meet and work together with so many wonderful and inspiring people. You gave me the freedom to explore different ideas, taught me how to distill scientific questions to their essence, and how to best present my work. I also would like to thank my thesis committee members, Prof. Alexandre Alahi, Prof. Luca Gambardella, Prof. Francesco Mondada, and Dr. Martin Saska, for presiding over my oral examination. Presenting my thesis to you was a true honor, especially since you have contributed to it in so many ways, e.g., your involvement in teaching activities, the organization of summer schools, and the articles that my work bases itself on. I am also very grateful to the Swiss National Science Foundation for providing the monetary resources to conduct this research.

I want to thank the reviewers who have taken the time to provide constructive criticism and feedback that has allowed me to improve my work. I would also like to thank the open source community for their relentless efforts to provide free, high-quality software to everyone. This thesis would not have been possible without your exceptional work.

I am very grateful to Michelle Wälti, the lab's administrative assistant, for providing her help with all sorts of organizational questions. I will fondly remember our interactions since they so often put a smile on my face. I would also like to thank Corinne Lebet and Sandra Roux, the administrative assistants of the EDRS doctoral program and the EPFL doctoral school, respectively. Your kindness and helpfulness are second to none.

A big thank you goes to Prof. Patric Jensfelt, my Master thesis supervisor, for his kindness and support. Your encouragements were key for my decision to pursue a PhD in robotics and machine learning. I would also like to thank Prof. Matthew Turk, for sparking my interest in computer vision with his course on mixed and augmented reality.

Another big thank you goes to my collaborators, co-authors, and friends, Dr. Julien Lecoeur, Dr. Fabrizio Schiano, and Enrica Soria. Thank you for the countless discussions in which you patiently explained things to me and gave me helpful feedback and advice. I have learned so much from each of you, both on a technical and personal level. I would also like to express my deepest gratitude to Olex, the mastermind behind the LeQuad hardware developments. This thesis could not have been completed without your technical expertise. In addition, working together with you was just a pleasure — thank you for making all the workshop hours and

rooftop sessions so much more enjoyable. Moreover, thank you to the students that I have supervised over four years. Each project was unique and I have learned a lot from every single one of you. I would also like to thank the numerous wonderful colleagues and friends that I have met in Lausanne over the years. I will fondly remember lunches and coffee breaks, as well as lake sessions and barbecues.

I am also forever thankful to my dear friends and lab brothers, Vivek, Enrico, and Vali. You have been the source of so much joy and the main reason I wanted to come to the lab even when the research was giving me headaches. Your presence and support made all the difference and you continue to inspire me every day. I will always remember our hikes and e-sport sessions during the lockdown, tea breaks in the 'headquarters', as well as conquering land, water, snow, and even the air together. Thank you so much to my housemates, Alex, Alice, and Blaise. You have been there for me every single day, seen me at my most stressed, and listened to my nerd talk. Thank you for transforming our apartment into a comfy home over so many shared meals, horror movie nights, joint workouts, and all the other shenanigans.

I would also like to thank my oldest and dearest friends, Felix, Jule, Marcel, Maze, and Timo. You have had such a grounding effect on me, it is hard to describe. Even though we do not spend every day together anymore, whenever we meet we go back to how things have always been — in the best possible way. It feels as though we are still in this together which I appreciate so much. I am forever grateful to my parents, Eva and Gerd, for their love and support for over three decades. Thank you for always believing in me and for giving me the feeling that anything is possible. I also want to thank my grandparents, Inge and Rudolf. Even though they are no longer among us, I am immensely thankful for their loving care and for sparking my technical curiosity from an early age. I am also incredibly grateful to Roeli, for her love, support, and understanding — and for being such an utter goofball. You make me laugh every single day, even on the few bad ones. You, your family, and your friends are such wonderful people and I am truly thankful to have you in my life.

Lausanne, September 29, 2021 Fabian Schilling

Abstract

Aerial robot swarms have the potential to perform time-critical and dangerous tasks such as disaster response without compromising human safety. However, their reliance on external infrastructure such as global positioning for localization and wireless networks for communication is still a limiting factor for many applications. Such infrastructure may not be available everywhere, increasing their chance of collisions in case of signal interruptions and limiting their robustness to failure. Moreover, agent-to-agent wireless communication can suffer from time delays and outages, preventing their scalability to large swarms that fly in dense configurations. Drones should have the autonomy to make their own decisions exclusively based on local sensory information to avoid scalability and robustness issues.

To address these limitations, we propose an entirely vision-based approach to swarm control inspired by flocking birds. In particular, we develop methods that enable drones to coordinate their motion by recognizing each other only with visual perception. We propose two distinct strategies that leverage the predictive power of convolutional neural networks: an end-to-end approach based on imitation learning and a modular system based on object detection and tracking. We test the algorithms using agent-based models and physics-based simulations with realistic sensor noise and validate them with a fleet of custom-built quadcopters in controlled indoor and challenging outdoor environments. The drones are equipped with an omnidirectional camera setup to avoid blind spots and onboard computation to process the images in real-time without requiring specialized hardware such as easy-to-recognize visual markers. Extensive simulations and real-world experiments show that vision-based swarms can perform collision-free and cohesive navigation while only relying on local visual information for control. We finally address the scalability of vision-based swarms in terms of group size and density.

Keywords: unmanned aerial vehicles, multi-robot systems, vision-based control, deep learning, collective motion, robot learning, object detection, flocking algorithms, imitation learning, multi-target tracking, sim-to-real transfer

Résumé

Les essaims de robots aériens ont la possibilité d'accomplir des tâches dangereuses et urgentes, telles que l'intervention en cas de catastrophe, le tout sans compromettre la sécurité humaine. Cependant, leur dépendance à l'égard d'une infrastructure externe, comme le système de positionnement global pour la localisation et les réseaux sans fil pour la communication, reste un facteur limitant pour de nombreuses applications. Ces infrastructures ne sont pas forcément disponibles partout, ce qui augmente les risques de collisions en cas d'interruption du signal et limite leur robustesse aux pannes. De plus, la communication sans fil entre agents peut souffrir de retards et de pannes, ce qui empêche son extensibilité à de grands essaims qui volent dans des configurations denses. Les drones devraient avoir l'autonomie nécessaire pour prendre leurs propres décisions en se basant exclusivement sur les informations sensorielles locales afin d'éviter les problèmes de scalabilité et de robustesse.

Pour remédier à ces limitations, nous proposons une approche entièrement basée sur la vision pour le contrôle d'essaims, inspirée des oiseaux en essaim. En particulier, nous développons des méthodes qui permettent aux drones de coordonner leurs mouvements en se reconnaissant mutuellement uniquement par la perception visuelle. Nous proposons deux stratégies distinctes qui exploitent le pouvoir prédictif des réseaux neuronaux convolutifs : une approche de end-to-end learning basée sur l'apprentissage par imitation et un système modulaire basé sur la détection et le suivi d'objets. Nous testons les algorithmes en utilisant des modèles basés sur des agents et des simulations basées sur la physique avec un bruit de capteur réaliste. Nous les validons avec une flotte de quadcoptères personnalisés dans des environnements intérieurs contrôlés et des environnements extérieurs difficiles. Les drones sont équipés d'un système de caméras omnidirectionnelles pour éviter les angles morts et d'un système de calcul embarqué pour traiter les images en temps réel sans nécessiter de matériel spécialisé tel que des marqueurs visuels faciles à reconnaître. Des simulations approfondies et des expériences dans le monde réel montrent que les essaims basés sur la vision peuvent effectuer une navigation sans collision et cohésive tout en se basant uniquement sur des informations visuelles locales pour le contrôle. Nous abordons enfin la scalabilité des essaims basés sur la vision en termes de taille et de densité de groupe.

Zusammenfassung

Flugroboterschwärme haben das Potenzial, zeitkritische und gefährliche Aufgaben wie Katastrophenhilfe zu übernehmen, ohne die Sicherheit von Menschen zu gefährden. Ihre Abhängigkeit von externer Infrastruktur wie der globalen Positionsbestimmung für Lokalisierung und drahtlosen Netzwerken für Kommunikation ist jedoch für viele Anwendungen noch immer ein limitierender Faktor. Solche Infrastrukturen sind möglicherweise nicht überall verfügbar, was die Wahrscheinlichkeit von Kollisionen bei Signalunterbrechungen erhöht und ihre Robustheit gegenüber Ausfällen einschränkt. Des Weiteren kann die drahtlose Kommunikation von Drohne zu Drohne unter Zeitverzögerungen und Ausfällen leiden, was ihre Skalierbarkeit für große Schwärme, die in dichten Konfigurationen fliegen, verhindert. Drohnen sollten die Autonomie haben, ihre eigenen Entscheidungen ausschließlich auf der Grundlage lokaler sensorischer Informationen zu treffen, um Probleme hinsichtlich der Skalierbarkeit und Robustheit zu vermeiden.

Um diese Einschränkungen zu beseitigen, schlagen wir einen vollständig visuellen Ansatz zur Schwarmkontrolle vor, der von Vogelschwärmen inspiriert ist. Insbesondere entwickeln wir Methoden, die es den Drohnen ermöglichen, ihre Bewegungen ausschließlich mittels visueller Wahrnehmung zu koordinieren. Wir schlagen zwei unterschiedliche Strategien vor, die sich die Leistungsfähigkeit von neuronalen Faltungsnetzen zunutze machen: einen Ende-zu-Ende Ansatz, der auf Imitation Learning basiert, und ein modulares System, das auf Objekterkennung und -verfolgung beruht. Wir testen die Algorithmen mit Hilfe von agentenbasierten Modellen und physikbasierten Simulationen mit realistischem Sensorrauschen und validieren sie mit einer Flotte von speziell angefertigten Quadcoptern in kontrollierten Innen- und anspruchsvollen Außenumgebungen. Die Drohnen sind mit omnidirektionalen Kameras ausgestattet, um tote Winkel zu vermeiden, und verfügen über genug integrierte Rechenleistung, um die Bilder in Echtzeit zu verarbeiten, ohne dass spezielle Hardware, wie z.B. leicht zu erkennende visuelle Marker, erforderlich sind. Ausführliche Simulationen und Experimente zeigen, dass bildverarbeitungsbasierte Drohnenschwärme, die nur auf lokale visuelle Stimuli reagieren, Kollisionen bei der Navigation erfolgreich vermeiden können, ohne dabei den Gruppenzusammenhalt zu beeinträchtigen. Zuletzt befassen wir uns mit der Skalierbarkeit von bildverarbeitungsbasierten Schwärmen in Bezug auf Gruppengröße und -dichte.

Table of contents

List of figures

List of tables

1 Introduction

1.1 Motivation

Aerial robots — commonly called drones — have enormous socio-economic potential and find applications in fields such as agriculture, mapping, construction, and delivery [1, 2, 3]. For instance, drones can be deployed to monitor crops, build detailed 3D maps, survey construction sites, and deliver medicine to inaccessible places — all without compromising the safety of human operators. Deploying drones in swarms — as opposed to a single robot — allows them to parallelize and complete time-critical tasks faster and more efficiently (Fig. 1.1). Collaboration between multiple drones can also enable entirely new applications such as cooperative transport [4] or mobile sensor networks [5, 6]. However, most drone swarms deployed today are far from autonomous since the robots cannot make their own decisions (i.e., they are remotely controlled) or rely on external infrastructure (e.g., satellite-based systems) to operate. Their lack of autonomy severely limits their applicability in many scenarios.

(a) Agriculture

(b) Delivery

(c) Firefighting

Figure 1.1 – Example applications of aerial robot swarms. Drone swarms can be used in (a) agriculture for tasks such as reporting on crop health, monitoring livestock, improving spraying accuracy, and mapping of farmland. They can also be used for (b) deliveries since they can quickly traverse densely populated areas and reach remote locations with limited road infrastructure. In (b) firefighting, they can be used to monitor the spread of a fire or to access burning buildings.

Most drone swarms deployed today fall into either of two categories: centralized or decentralized [7]. Centralized drone swarms rely on a ground computer that precomputes their trajectories and continuously monitors their positions and velocities [8, 9, 10, 11]. In centralized systems, the ground computer represents a single point of failure and its malfunction can have disastrous consequences since all drones rely on it to receive new motion commands. In contrast, decentralized drone swarms typically rely on the the wireless exchange of position information obtained from external localization systems to operate [12, 13, 14, 15]. Localization is usually achieved with optical motion capture for indoor deployments [8, 16, 9, 11] or satellite-based systems for outdoor applications [17, 13, 12, 14]. Optical motion capture systems are very precise but entirely centralized, difficult to deploy at a large scale, and costly. Satellite-based localization systems are unavailable indoors and suffer from inaccuracies in many outdoor environments where satellites may be occluded and signals reflected on obstacles, e.g., cities and forests. Communication-based approaches are inflexible since all agents must be localized in the same reference frame and adhere to the same communication protocol. Moreover, wireless communication often suffers from delays and outages, which increase the risk of collisions and thus prevent the drones from flying in dense configurations [14, 18].

Natural swarms such as flocks of birds are a great source of inspiration since they do not depend on any central infrastructure or explicit communication to fly in swarms and navigate their environment. In particular, they rely only on their local sensing capabilities to perceive other swarms members. For example, large flocks of starlings can fly in dense configurations while keeping a safe distance between each other to avoid collisions [19]. Faced with a predator attack, they swiftly split and reunite again to maintain group cohesion [20]. Their migration patterns show that birds can efficiently solve complex navigation problems in a completely self-organized manner [21]. The consensus among biologists is that visual perception is the essential sensory modality to give rise to these behaviors [22, 23, 24].

Giving drones the ability to perceive each other can remove their dependence on external infrastructure and wireless communication. The failure or malfunction of external localization infrastructure would no longer cause entire missions to fail. Delays and outages in wireless communication would no longer cause collisions since the drones are aware of each other. By only relying on local visual information, drone swarms can operate in a decentralized fashion and would be more flexible, scalable, and robust to failures [7]. Moreover, vision is arguably the ideal sensory modality for localization on aerial robots since cameras are small, lightweight, and provide high information density at comparatively low power consumption [25]. However, vision-based methods come with their own set of unique challenges. Visual recognition of drones with computer vision techniques is challenging because they are relatively small and can fly in environments with large amounts of background clutter and difficult lighting conditions [26, 27]. Additionally, the image processing algorithms must run onboard the drones, whose own motion may generate motion blur and whose small dimensions and lightweight mass can restrict computational capabilities. Furthermore, other perceptual factors such as visual occlusions become relevant for larger group sizes and swarms that fly in

dense configurations.

1.2 General approach

This thesis is concerned with the synthesis of vision-based aerial robot swarms in which the drones recognize each other only using visual perception. The use of local visual information will enable the drones to cooperate without external localization infrastructure or wireless communication. The overall goal is to deploy a vision-based group of drones that can perform navigation tasks without collisions or fragmentation into subgroups. Such collision-free and cohesive swarms can be an enabling factor for a variety of real-world applications (Fig. 1.1).

We take biological inspiration from natural collectives such as flocks of birds to develop vision-based controllers for aerial robot swarms. We expect many characteristics of natural swarms (e.g., scalability and flexibility) to translate to robotic swarms if they are built on the same principles (e.g., self-organization and decentralization). Biological research suggests that flocking birds rely predominantly on visual perception for motion coordination [19, 22, 23, 28, 24]. We argue that visual perception is sufficient to generate the desired behavior with robots, so we limit this thesis to strategies that rely on vision as a sensory modality for control. Notably, this excludes methods that are based on modalities such as sound [29, 30] or wireless signal strength [31, 32], for example.

This thesis focuses on methods that are based on deep learning [33, 34] due to its empirical successes in related fields such as computer vision [35], robotics [36], and biology [37]. Deep neural networks give machines an unprecedented ability to extract useful and actionable information from the world around them. In computer vision, convolutional neural networks have drastically lowered error rates of tasks such as object detection [38, 39, 35] and semantic segmentation [40, 41, 42], enabling computers to recognize and localize objects in images with pixel-level precision and human-level accuracy. In robotics, deep learning enables robotic arms to grasp novel objects [43], legged robots to walk over difficult terrain [44], and aerial robots to navigate without collisions [45]. In biology, convolutional networks have become an indispensable tool for detecting and tracking animals [37] to study the interactions between them [46]. Deep learning approaches — and convolutional neural networks in particular — seem to be suitable methods for vision-based control of aerial robot swarms.

We use agent-based modeling and physically realistic simulations to test and validate the developed methods. Agent-based models are very useful for the development of multi-agent systems and are widely used in fields such as robotics [47, 48, 49] and biology [50, 51, 52]. Simple kinematic agents allow us to gain insights into collective behaviors by rapid prototyping and to generate reproducible statistical results. Moreover, they enable the simulation of large group sizes without running into computational bottlenecks. Physics-based simulations increase the realism of agent-based models by considering the drone dynamics and sensors such as cameras and inertial measurement units [53, 54, 55]. Realistic simulators enable us to validate and run our algorithms like on real hardware without solving engineering problems

related to actuation, communication, or perception.

We implement and deploy the methods and algorithms on real drones and evaluate them in controlled indoor environments and challenging outdoor scenarios. The real-world results obtained in this thesis are based on a fleet of custom-built quadcopters — nicknamed leQuad — that were adapted and redesigned from the work of previous students [56, 57]. Each drone is equipped with an open-source autopilot [58, 59] and a powerful embedded computer that enables the onboard processing of images from an omnidirectional camera array in real-time.

1.3 Thesis outline

In the following, we provide a brief outline of the thesis and summarize the content of each chapter. The summaries are based on the abstracts of the publications mentioned in the respective chapters.

Chapter 2: Flocking algorithm for aerial robot swarms

We describe a flocking algorithm that serves as the basis for high-level control in subsequent chapters. We also define the metrics we use throughout the thesis to assess the performance of vision-based swarms.

Chapter 3: End-to-end vision-based flocking using imitation learning

We propose a method for vision-based swarm control based on end-to-end imitation learning. Each drone is controlled by a convolutional neural network that takes omnidirectional images as inputs and predicts velocity commands that match those computed by the flocking algorithm (Chapter 2). We train the neural network using a combination of real and simulated images and propose a task-specific unsupervised domain adaptation approach to facilitate the sim-to-real transfer. We test the approach in simulation with a vision-based flock of nine drones and in a motion tracking hall with two real quadcopters that perform several leader-follower experiments. The results show that end-to-end imitation enables collision-free and cohesive vision-based flight without an explicit spatial representation of the neighboring drones, e.g., range and bearing. The neural network implicitly learns to localize other agents, which we show with an attribution study that highlights the regions of the visual inputs with the most influence on the motion of an agent. The proposed method can be optimized end-to-end, does not require camera calibration, and works without visual fiducial markers to simplify the recognition. We thus remove the dependence on sharing positions among swarm members by taking only local visual information into account for control.

Chapter 4: Modular vision-based flocking using object detection and tracking

We propose a modular approach to vision-based flocking based on visual detection and multi-target tracking. Each drone employs a convolutional neural network to detect and localize its neighbors onboard in real-time. We train the neural network with automatically labeled images using background subtraction by systematically flying a quadcopter in front of a static camera. The method enables us to collect a diverse image dataset with precise bounding box annotations in different environments without manual labeling. We use a multi-target tracker to transform the predicted bounding boxes into estimates of relative positions and velocities, which are subsequently used by the flocking algorithm for high-level control (Chapter 2). We test the approach in simulation and validate it using a group of three real quadcopters in an outdoor environment with substantial background clutter and difficult lighting conditions. The results show that the drones can navigate safely and cohesively without relying on external localization, wireless communication, or visual fiducial markers. Moreover, the proposed method is modular and easy to debug since the performance of each component can be assessed individually.

Chapter 5: Scalable vision-based flocking in the presence of occlusions

We address the scalability of vision-based flocking with regard to group size and density. Vision-based swarms rely on the detection of neighbors but usually neglect mutual visual occlusions because they operate in small groups (Chapter 4). We extend the flocking algorithm (Chapter 2) with a realistic model of visual occlusions that discards agents if they are obstructed by closer ones. We evaluate the occlusion model with up to one thousand agents, showing that occlusions have adverse effects on the inter-agent distances and velocity alignment as the swarm scales up, both in terms of group size and density. In particular, we find that small agent displacements have considerable effects on neighbor visibility and lead to control discontinuities. The destabilizing effects of visibility switches, i.e., agents continuously becoming visible or invisible, can be mitigated if agents select their neighbors from adjacent Voronoi regions. We evaluate the Voronoi-based flocking algorithm with one hundred quadcopters in a physics simulation with realistic quadcopter dynamics and sensor noise. The results show that vision-based swarms can remain collision-free and cohesive across group sizes and densities despite visual occlusions.

Chapter 6: Conclusions

We summarize the thesis and conclude with a discussion of the implications, significance, limitations, and possible directions for future work.

Appendix A: Vision-based localization in dynamic environments

We propose a dataset and method for vision-based localization in dynamic environments. This task is highly relevant for vision-based swarms since each drone has to localize itself while neighboring robots are constantly in motion. However, these deployment scenarios are still challenging for most vision-based localization algorithms because they assume to operate in static scenes. We propose a dataset that captures the dynamicity of common environments as a benchmark for evaluating the robustness of vision-based localization algorithms. We also propose a method to mitigate the adverse effects of dynamic objects by taking semantic information into account.

Appendix B: Open-source software

We briefly describe the software packages that were developed during this thesis.

Appendix C: Publications

The work presented in this thesis is based on the following publications:

- F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," in *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4523–4530, Oct. 2019 [60].
- F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, Apr. 2021 [61].
- F. Schilling, E. Soria, and D. Floreano, "On the scalability of vision-based drone swarms in the presence of occlusions," in *IEEE Access*, vol. 1, no. 1, pp. 1–13, (**submitted**) Aug. 2021 [62].
- K. Minoda, F. Schilling, V. Wüest, D. Floreano, and T. Yairi, "VIODE: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1343–1350, Apr. 2021. [63]

2 Flocking algorithm for aerial robot swarms

In this chapter, we describe a simple and versatile flocking algorithm that we use to synthesize collective motion throughout the thesis. The purpose of the flocking algorithm is threefold. Firstly and most importantly, collisions among agents should be avoided. Secondly, the swarm should remain cohesive as a single unit without breaking into subgroups. Thirdly and finally, the swarm should be able to perform collective waypoint navigation.

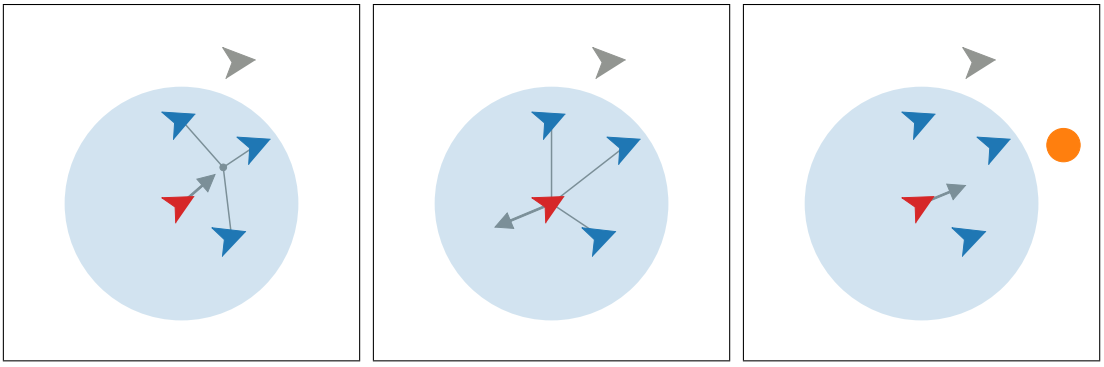
2.1 Preliminaries and notation

We consider a set of N homogeneous agents that are labeled by $i \in \mathcal{A}$, where $\mathcal{A} = \{1, 2, \ldots, N\}$ denotes the set of all agents and $|\mathcal{A}| = N$ its cardinality. We also define the set of all but the focal agent i as $\mathcal{A}_i = \mathcal{A} \setminus \{i\}$. The state of each agent i can be described by its position and velocity $\mathbf{p}_i, \mathbf{v}_i \in \mathbb{R}^m$. We are especially interested in the cases where $m \in \{2, 3\}$, i.e., planar or three-dimensional agent configurations, respectively. We denote the relative position of agent i with respect to j as $\mathbf{r}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ with distance $d_{ij} = \|\mathbf{r}_{ij}\|$ where $\|\cdot\|$ is the Euclidean norm.

We model the swarm of agents as a directed sensing graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of vertices $\mathcal{V} = \{1, \ldots, N\}$ denotes the agents and the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ contains the ordered pairs of agents $(i, j) \in \mathcal{E}$ if an agent i is adjacent to agent j, which we denote by $i \sim j$. The graph $\mathcal{G}$ can also be represented by an $N \times N$ adjacency matrix of the form A_{ij} with entries of 1 if $i \sim j$ and 0 otherwise.

2.2 Flocking algorithm

The objective of the swarm is to perform waypoint navigation while avoiding inter-agent collisions and staying together as a group (Fig. 2.1). We formulate this objective as an artificial potential field that is inspired by the Reynolds flocking algorithm [64]. The motion of an agent is composed of an attractive/repulsive potential that provides separation and cohesion between agents (Sec. 2.2), as well as a migratory potential responsible for goal-directed



(a) Cohesion (b) Separation (c) Migration

Figure 2.1 – The flocking algorithm is composed of three complementary terms: (a) cohesion, (b) separation, and (c) migration. The focal agent (red triangle) reacts only to the neighboring agents (blue triangles) within its perception radius (light blue disk) but not the ones outside of it (gray triangle). All agents are attracted equally by the migration point (small orange disk). The cohesion term (a) keeps agents together by steering them towards the average position of their neighbors. The separation term (b) prevents collisions among agents by repulsing them from each other. The migration term (c) introduces a navigation goal and steers the agents towards a waypoint. The sum of all terms produces collision-free and cohesive goal-directed navigation.

navigation (Sec. 2.2).

The motion of an agent is composed of a social term that captures agent-to-agent interactions and a migration term that introduces the navigation objective. The velocity command of an agent can be written as

$$\mathbf{v}_i = \mathbf{v}_i^{\text{soc}} + \mathbf{v}_i^{\text{mig}} \tag{2.1}$$

where $\mathbf{v}_i^{\text{soc}}$ and $\mathbf{v}_i^{\text{mig}}$ denote the respective social (Eq. 2.3) and migration terms (Eq. 2.5). In order to obtain a final velocity command that is feasible even under the actuation constraints of a physical robot, we limit its magnitude as

$$\tilde{\mathbf{v}}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|} \min(\|\mathbf{v}_i\|, v^{\max}) \tag{2.2}$$

where $v^{\max}$ denotes the maximum speed.

Separation and cohesion

Cohesion and collision avoidance can be achieved with an attractive/repulsive potential that keeps the agents at an equilibrium distance (Fig. 2.2). The cohesion term keeps the swarm together by attracting agents to the average position of their neighbors. The separation term leads to collision avoidance by repulsing nearby agents from each other. We can express these

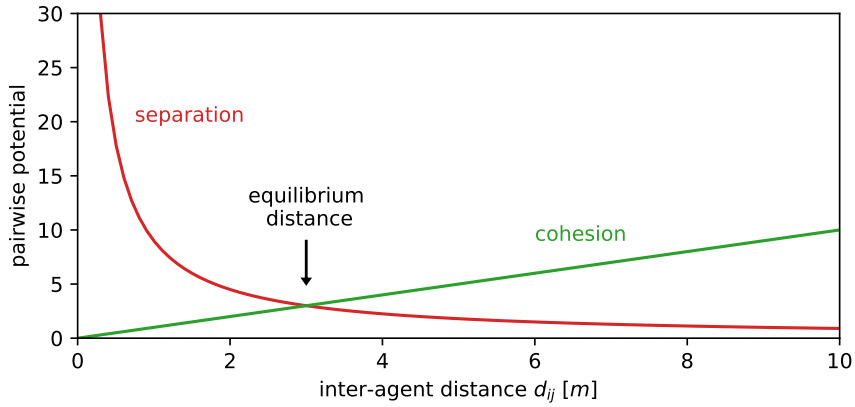


Figure 2.2 – Pairwise artificial potential of cohesion and separation terms as a function of inter-agent distance. The cohesion term is a linear function of the inter-agent distance, whereas the separation term is inversely proportional to the distance. The equilibrium distance is defined as the inter-agent distance at which the cohesion and separation terms balance.

rules more formally as

$$\mathbf{v}_i^{\text{soc}} = k^{\text{coh}} \underbrace{\frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{r}_{ij}}_{\text{cohesion}} - k^{\text{sep}} \underbrace{\frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^2}}_{\text{separation}} \tag{2.3}$$

where k^{coh} and k^{sep} are gains that regulate the strength of the attraction and repulsion, respectively.

Neighbor selection is an important consideration for all flocking algorithms since it introduces the notion of locality (e.g., in communication, perception, etc.) as opposed to all-to-all information transfer. Unless specified otherwise (e.g., in Chapter 5), we select neighbors from a metric perception radius defined by

$$\mathcal{N}_i = \{ j \in \mathcal{A}_i \mid d_{ij} < r^{\max} \} \tag{2.4}$$

where $r^{\max}$ denotes the maximum perception range.

Migration

The purpose of the migration term is to give the agents a navigation goal by steering them towards a waypoint. The migration term can be written as

$$\mathbf{v}_i^{\text{mig}} = k^{\text{mig}} \frac{\mathbf{r}^{\text{mig}}}{\|\mathbf{r}^{\text{mig}}\|} \tag{2.5}$$

where $\mathbf{r}^{\text{mig}}$ denotes the relative position of the migration point with respect to the focal agent, and k^{mig} the gain for modulating the migration speed.

Other considerations

In this section, we briefly explain our decisions on other modeling choices such the alignment term (Sec. 2.2), limited field of view (Sec. 2.2), and alternative flocking algorithms considered (Sec. 2.2).

Alignment

We do not make use of the alignment term (also: *velocity matching*) that is used in the original Reynolds flocking formulation [64]. We only consider the separation (also: *collision avoidance*) and cohesion (also: *flock centering*) terms for agent-to-agent interactions since they only depend on relative positions. Conversely, the alignment term depends on relative velocities which are difficult to infer without tracking agents over multiple time steps. Inferring the heading from the agent orientation is equally difficult for holonomic platforms such as quadcopters which are symmetric along both horizontal axes. However, the alignment of velocities is a byproduct and a direct consequence of adding the migration term for waypoint navigation.

Limited field of view

Similarly, we do not limit the field of view of the perception radius as is sometimes done in the literature [65]. Research on flocking algorithm with a limited field of view shows that lateral vision is crucial for collision-free collective motion [66, 67] and may explain why flocking birds have near omnidirectional vision [68]. For a robotic implementation, a limited field of view can be seen as a rather artificial constraint since omnidirectional cameras are available off-the-shelf [69] and more cameras can easily be added to achieve omnidirectional vision while limiting distortions [70].

Other flocking algorithms

The flocking algorithm literature is extensive and there exist many formulation with different assumptions and goals [47]. Most notably, we consider other algorithms such as Olfati-Saber [71] and Vásárhelyi flocking [14] for their desirable properties. The Olfati-Saber algorithm has the advantage that a precise inter-agent reference distances can be specified apriori. The agents form a lattice-like structure in which nearest neighbor distances converge to the desired value. The Vásárhelyi algorithm is specifically designed to dampen oscillations with a viscous friction-like term. This enables the agents to avoid collisions and align their velocities even in the presence of time delays and communication outages. However, both of these algorithms

are prone to fragmentation if they are not constrained to confined environments [72, 11]. In particular, the Olfati-Saber algorithm can only be used with a very limited perception radius [71] and the Vásárhelyi algorithm does not model cohesion at all [14]. Therefore, we find that Reynolds flocking is sufficient for our needs since it enables collision avoidance, group cohesion, and waypoint navigation in a simple formulation.

2.3 Flocking performance metrics

We briefly describe several complementary metrics to evaluate the performance of flocking algorithms. The most relevant metrics are the 1) minimum nearest neighbor distance d^{min}, 2) order ϕ^{order}, and 3) union ϕ^{union}. These metrics capture whether we have achieved 1) collision-free, 2) ordered, and 3) cohesive collective navigation, respectively.

Distance

The minimum nearest neighbor distance is arguably the most important metric since it captures whether or not the agents can effectively avoid collisions during migration. It is computed as

$$d^{\text{min}} = \min_{i \neq j} d_{ij} \tag{2.6}$$

and we say that a collision occurs whenever two agents get closer than twice their radius $d^{\text{min}} < 2r$. We also use the maximum inter-agent distance $d^{\text{max}} = \max_{i \neq j} d_{ij}$ to capture the dispersion between agents.

Order

The order metric measures the correlation of the velocity vectors of the agents within the swarm. It is computed as

$$\phi^{\text{order}} = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}. \tag{2.7}$$

An order value of one indicates that all agents are moving in the same direction in perfect alignment, whereas a value around zero means that the swarm is in a completely disordered state in which no two agents align their direction of motion.

Union

The union metric measures the cohesion of the swarm and expresses whether the swarm has split into subgroups. It is computed as

$$\phi^{\text{union}} = 1 - \frac{n^{\text{comp}} - 1}{N - 1} \tag{2.8}$$

where n^{comp} is the number of connected components of the neighbor adjacency matrix (Sec. 2.1). A union value of one indicates that the swarm is moving as a single cohesive unit. A value of zero represents the degenerate situation in which the swarm is split into N subgroups and the agents are unable to perceive any other agent.

3 End-to-end vision-based flocking using imitation learning

Most drone swarms deployed today either rely on sharing positions among agents or detecting swarm members with the help of visual markers. This chapter proposes a vision-based approach to coordinate drone swarms based on end-to-end imitation learning without the need for markers. Each agent is controlled by a convolutional neural network that takes omnidirectional images as inputs and predicts velocity commands that mimick those computed by a flocking algorithm with access to ground-truth positions. We train the neural network using a combination of real and simulated images and propose a simple yet effective unsupervised domain adaptation approach to facilitate the transfer to the real world. The neural network learns to avoid mutual collisions, stay cohesive to the group, and localize other drones in the input images, the latter without direct supervision. We demonstrate the behaviors with a swarm of nine agents in realistic physics-based simulation and with a group of two quadcopters performing leader-follower experiments in an indoor motion tracking hall. The neural network policy removes the need to share positions among agents by taking only local visual information into account for control.

The work presented in this chapter is adapted from [60][1]:

- F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," in *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4523–4530, Oct. 2019.

3.1 Introduction

The collective motion of animal groups such as flocks of birds is an awe-inspiring natural phenomenon that has profound implications for the field of aerial swarm robotics [1, 2]. Animal groups in nature operate in a completely self-organized manner since the interactions between them are purely local. By taking inspiration from decentralization in biological systems, we can develop powerful robotic swarms that are 1) robust to failure, and 2) highly

[1] Video: https://youtu.be/I9vFvPphfpU.

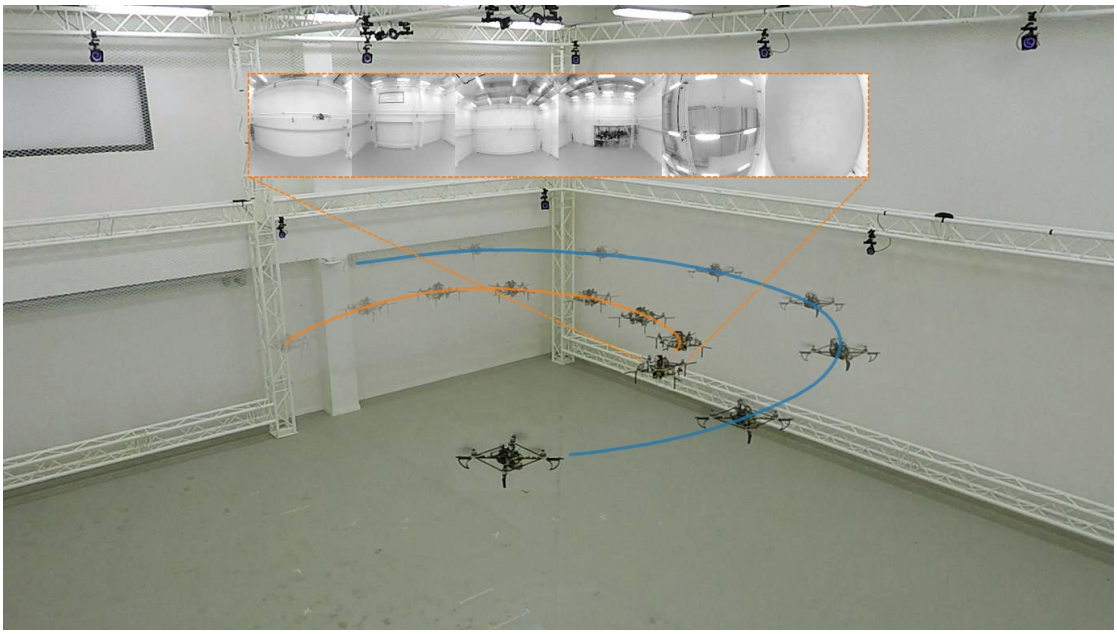


Figure 3.1 – Vision-based leader-follower flight in the motion tracking hall. The proposed visual controller operates fully decentralized and provides collision-free, coherent collective motion without the need to share positions among agents. The behavior of an agent only depends on its omnidirectional visual inputs (orange rectangle). Collision avoidance and cohesion between agents are learned entirely from visual inputs.

scalable since the number of agents can be increased or decreased dynamically depending on the workload of the task.

One of the most appealing characteristics of collective animal behavior for robotics is that decisions are made based on local information. Thus, the behavior of animal groups does not require extensive knowledge of the swarm state or a central coordinator. As of today, however, most multi-agent robotic systems rely on entirely centralized control [8, 9, 10] or wireless communication of positions [13, 12, 14], which are obtained either from a motion capture system or global navigation satellite system (GNSS). The main drawback of these approaches is the introduction of a single point of failure, as well as the use of unreliable data links, respectively. Relying on centralized control bears a significant risk since the agents lack the autonomy to make their own decisions in failure cases such as a communication outage. The possibility of failure is even higher in dense urban environments, where GNSS measurements are often unreliable and imprecise.

Vision is arguably the most promising sensory modality to achieve a maximum level of autonomy for robotic systems, particularly considering the recent advances in computer vision and deep learning [33]. Apart from being lightweight and having relatively low power consumption, even cheap commodity cameras provide an unparalleled information density with respect to sensors of similar cost. Their characteristics are specifically desirable for the deployment

of an aerial multi-robot system. The difficulty when using cameras for robot control is the processing of the visual information which this work addresses directly.

In this chapter, we propose a reactive control strategy based entirely on local visual information (Fig. 3.1). We formulate the swarm interactions as a regression problem in which we predict control commands as a nonlinear function of the visual input of a single agent. To the best of our knowledge, this is the first successful attempt to learn vision-based swarm behaviors such as collision-free navigation in an end-to-end manner directly from raw images.

Our contributions can be summarized as follows:

- We propose a data-efficient imitation learning approach to solve the problem of vision-based coordination of a swarm of drones. The control policy is trained incrementally by following the previous best policy and thus collecting relevant data from its failure cases. The proposed system generates high-level control commands from raw images in the form of velocity setpoints, whereas a classical cascaded feedback control architecture handles low-level control.
- We present a remarkably simple and effective task-specific unsupervised domain adaptation approach to transfer the image data obtained from simulation to the real world. To this end, we collect a dataset of unlabeled images from the target environment to serve as backgrounds for images generated in simulation.
- We implement the algorithm on a physical quadrotor platform and show that all computations (policy evaluation, state estimation, and control) can be run entirely onboard in real-time.
- We provide an evaluation of the system in simulation and experimental validation in a motion tracking hall to show that the control policy generalizes to coordinated multi-agent flights in the real world.

3.2 Related work

Decentralized swarms of drones such as quadrotors and fixed-wings are the focus of recent research in swarm robotics. Early work presents ten fixed-wing drones deployed in an outdoor environment [17]. Their collective motion is based on Reynolds flocking [64] with a migration term that allows the swarm to navigate towards the desired goal. Thus far, the largest decentralized quadrotor swarm consisted of 30 autonomous agents flying in an outdoor environment [14]. The underlying algorithm has many free parameters which are optimized using an evolutionary algorithm that relies on a fitness function that incorporates several swarm order parameters. The commonality of the mentioned approaches and others, for example, [13, 73], is the ability to share GNSS positions wirelessly among swarm members. However, there are many situations in which wireless communication is unreliable or GNSS

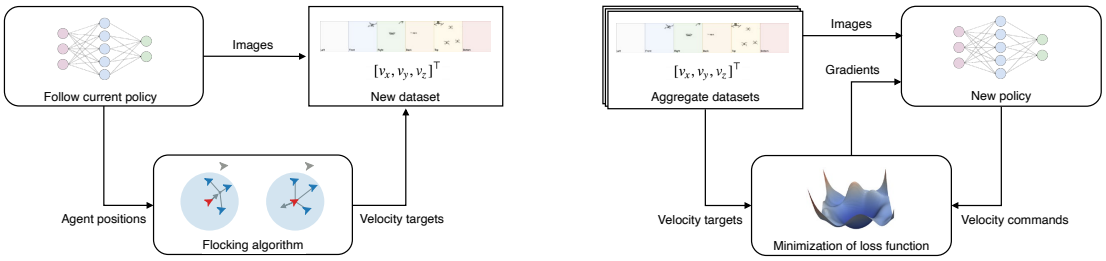
positions are too imprecise. We may not be able to tolerate position imprecisions in situations where the environment requires a small inter-agent distance, e.g., when traversing narrow passages in urban environments. In these situations, tall buildings may deflect the signal and communication outages occur due to the wireless bands being over-utilized.

Recent advances in the field of machine learning facilitate the vision-based control of flying robots. In particular, the controllers are based on three types of learning methods: imitation learning, supervised learning, and reinforcement learning. Imitation learning is used in [74] to control a drone in a forest environment based on human pilot demonstrations. The authors motivate the importance of following suboptimal control policies in order to cover more of the state space. A supervised learning approach [75] features a convolutional network that is used to predict a steering angle and a collision probability for drone navigation in urban environments. In contrast with the previous methods based only on supervised learning, an approach based on reinforcement learning [76] shows that a neural network trained entirely in a simulated environment can generalize to flights in the real world. The work described above and other similar methods, for instance, [45, 77], use a data-driven approach to control a flying robot in real-world environments. The probability of collision is learned by minimizing the binary cross-entropy of labeled images collected while riding a bicycle through urban environments. A shortcoming of these methods is that the learned controllers operate only in two-dimensional space which bears similar characteristics to navigation with ground robots. Moreover, the approaches do not show the ability of the controllers to coordinate a multi-agent system.

The control of multiple agents based on visual inputs is achieved with relative localization techniques [78] for a group of three quadrotors. Each agent is equipped with a camera and a circular marker that enables the detection of other agents and the estimation of relative distance. The system relies only on local information obtained from the onboard cameras in near real-time. Thus far, decentralized vision-based drone control has been realized by mounting visual markers on the drones [79]. Although this simplifies the relative localization problem significantly, the marker-based approach would not be desirable for the real-world deployment of flying robots. The used visual markers are relatively large and bulky which unnecessarily adds weight and drag to the platform; this is especially detrimental in real-world conditions. Another recently proposed approach is the use of active ultraviolet markers to identify the relative range and bearing to other agents [80]. However, the markers have to be placed in carefully chosen pre-defined locations, and the system is thus unable to detect markerless drones that do not precisely conform to these specifications.

3.3 Method

At the core of the proposed method lies the prediction of a velocity command for each agent that matches the velocity command computed by a flocking algorithm (Fig. 3.2a). We consider the velocity command from the flocking algorithm as the target for a supervised imitation



(a) Step 1: Collect new dataset from current policy

(b) Step 2: Train next policy on aggregate datasets

Figure 3.2 – Each iteration of the imitation learning algorithm follows two steps: (a) dataset collection and (b) policy training. In the (a) dataset collection step, we follow the current policy to obtain omnidirectional images and the corresponding velocity targets computed from a flocking algorithm with privileged access to the positions of the agents. In the (b) policy training step, we obtain a new policy by minimizing the loss function on all previously collected aggregate datasets. The new policy from (b) is then used in (a) as a current policy to collect a new dataset and so on. The current policy learns to recover from the failure cases of the previous policies which are contained in the aggregate datasets. The final policy no longer depends on the knowledge of the agent positions since it learns to imitate the flocking algorithm entirely from visual inputs.

learning problem (Fig. 3.2b). The main idea is to eliminate the dependence on the knowledge of the positions of other agents by taking only local visual information into account for control. Imitation learning presents a practical alternative to the approach in which separate modules are responsible for object detection, multi-object target tracking, and control, respectively (Chapter 4). The modular approach requires the labeling of large amounts of images with precise bounding box annotations. By using direct imitation, the control inputs can be calculated directly from the relative positions of other agents obtained either from simulation or a motion capture system.

Flocking algorithm

We use an adaptation of Reynolds flocking [64] to generate targets for the learning algorithm (Sec. 2.2). In particular, we only consider the *collision avoidance* and *flock centering* terms from the original formulation since they only depend on relative positions. We omit the *velocity matching* term since estimating the velocities of other agents is a challenging task given only a single snapshot in time. One would have to rely on either estimating velocities from several consecutive images or estimating the orientation and heading with relatively high precision in order to infer velocities from a single image.

In the formulation of the flocking algorithm, we use the terms *separation* and *cohesion* to denote collision avoidance and flock centering, respectively [81]. We further add an optional *migration* term that enables the agents to navigate towards a goal. An important consideration when modeling the desired behavior of the swarm is the notion of neighbor selection. It is

reasonable to assume that each agent can only perceive its neighbors in a limited range. We therefore only consider agents as neighbors if they are closer than the desired cutoff distance $r^{\max}$ which corresponds to only selecting agents in a sphere with a given radius. We do not make any restrictions on the field of view of the agents since limiting perception, specifically in the lateral direction, has been shown to have adverse effects on the flocking performance [67].

The separation term steers an agent away from its neighbors in order to avoid collisions, whereas the cohesion term can be seen as the antagonistic inverse since its purpose is to steer an agent towards its neighbors to provide coherence to the group (Sec. 2.2). For the implementation, the separation and cohesion terms are sufficient to generate a collision-free swarm in which agents remain together, given that the separation and cohesion gains are chosen carefully. We denote the combination of the two terms as the social velocity command $\mathbf{v}_i^{\text{soc}} = \mathbf{v}_i^{\text{sep}} + \mathbf{v}_i^{\text{coh}}$ which is later predicted by the neural network. Moreover, the addition of the migration term provides the possibility to give a uniform navigation goal to all agents (Sec. 2.2). The velocity command for an agent i is computed as a sum of the social terms, which is a combination of separation and cohesion, as well as the migration term, as $\mathbf{v}_i = \mathbf{v}_i^{\text{soc}} + \mathbf{v}_i^{\text{mig}}$. In general, we assume a homogeneous swarm, which means that all agents are given the same gains for separation, cohesion, and migration.

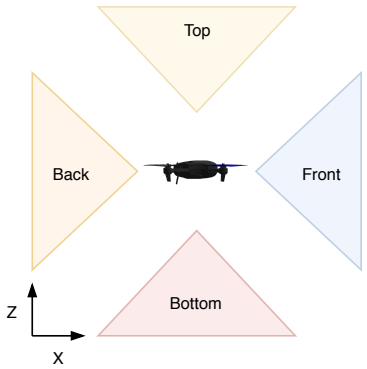
3.3.1 Drone model

We perform simulations in Gazebo with a group of nine quadrotor drones, each equipped with six simulated cameras to provide omnidirectional vision (Fig. 3.3). The cameras are positioned away from the center of gravity of the drone in order to have an unobstructed view of the surrounding environment, including the propellers. Each camera has a $135 \times 90°$ horizontal and vertical field of view and takes a grayscale image of 128×128 pixels with a refresh rate of 10 Hz. We concatenate the images from all six cameras along the horizontal axis to form a 128×768 pixels grayscale image.

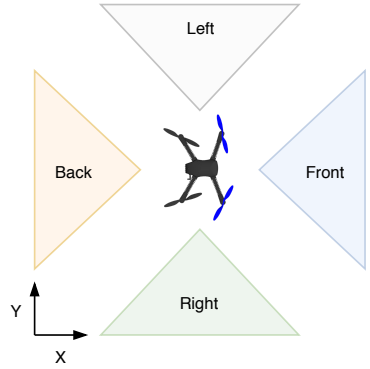
3.3.2 Imitation learning

We use an on-policy imitation learning approach to synthesize a purely vision-based control policy, denoted by $\hat{\pi}$, that matches the behavior of the position-based flocking policy, denoted by π^*, as closely as possible. More formally, we denote the learned policy $\hat{\pi}(\mathbf{o}_t) = \mathbf{a}_t$ as a mapping from observations to actions, where the observations $\mathbf{o}_t \in \mathbb{R}^{128\times768}$ are grayscale images and the actions $\mathbf{a}_t \in \mathbb{R}^3$ velocity commands for each time step $t \in T$. The expert policy $\pi^*(\mathbf{s}_t) = \mathbf{a}_t$, on the other hand, computes velocity commands from the state $\mathbf{s}_t$ of the system, which is represented by known relative positions $\mathbf{r}_{ij}$ to other agents (Sec. 3.3). The image observations can be seen as a lossy representation of the underlying system state (e.g., the relative positions of other agents) because of adverse factors such as limited resolution, occlusions, lens distortions, and inherent noise in the system. To learn the vision-based policy

(a) Concatenation of six orthogonal camera images



(b) Side view



(c) Top view

Figure 3.3 – The visual input of an agent is the (a) concatenation of six orthogonal camera images. We show the positioning of the cameras schematically as a (b) side view and (c) top view of the simulated drone. The images are combined into the full visual field of an agent (color-coded by camera direction). The cameras are positioned such that the visual field of an agent corresponds to an image cube map, i.e., each camera is pointing at a different face of a cube as seen from the center of the cube itself.

$\hat{\pi}$, we use the DAGGER imitation learning algorithm [82] (Alg. 1).

We collect data and train the policy in an iterative fashion, first in simulation and then in a motion tracking hall. We use a 80%/20% split between training $\mathcal{D}_{\text{train}}$ and validation data $\mathcal{D}_{\text{val}}$. In simulation, each iteration of the imitation learning algorithm proceeds as follows. The drones take off and assume random positions within a cube of side length 4 m, and with a minimum inter-agent distance of 1.5 m. The side length and minimum distance were chosen to resemble a plausible real-world deployment scenario in a confined environment such as the motion tracking hall. All agents then switch to vision-based control and use raw velocity commands generated by the learned policy (which is randomly initialized at first) from the visual inputs sampled at 10 Hz. Simultaneously, ground truth control commands are computed from the flocking algorithm and stored for post-processing. The iteration is considered complete as soon as 1) any two drones collide, 2) any two drones are too far away from each other, or 3) 200 observation-action samples are generated. We consider two agents too close if any pair of drones falls below a collision threshold of 1 m; similarly, we consider two agents as too far away when the distance between them exceeds a threshold of 7 m. The collision threshold follows the constraints of the drone model, and the dispersion threshold stems from the diminishing size of other agents in the field of view. For data collection in the

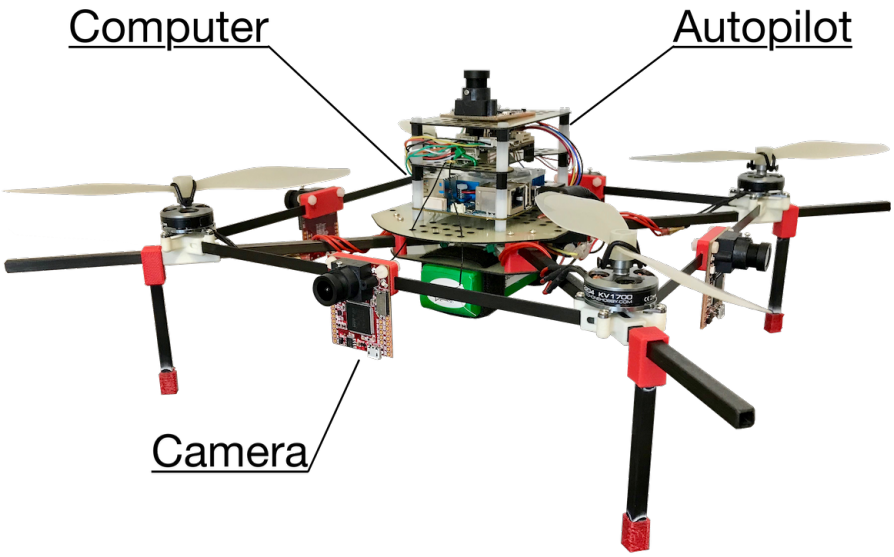


Figure 3.4 – The hardware implementation of the drone is based on the design in [73]. The drone uses six OpenMV Cam M7 with ultra-wide angle lenses for image acquisition, an Odroid XU4 onboard computer for image processing, and a PixRacer autopilot for state estimation and control.

real world, we relax the above requirements and stop an iteration as soon as the situation becomes subjectively too dangerous, for instance when the inter-agent distance becomes too small, or the drone starts to move too close to the walls of the motion tracking hall. A new policy is then trained using the collected image samples, the control commands generated by the learned policy, and the expert control commands computed from the flocking algorithm rules. Finally, the data collection process is repeated with the new policy.

3.3.3 Domain adaptation

One fundamental problem with the on-policy imitation learning approach (Alg. 1) is that the learned policy needs to be executed online in order to collect samples. Executing an untrained policy in a multi-agent setting with quadcopters in a confined space such as a motion tracking hall can be dangerous. A possible solution is to set an initial policy $\pi_i(\mathbf{o}_t) = \beta_i \pi^*(\mathbf{s}_t) + (1-\beta_i)\hat{\pi}_i(\mathbf{o}_t)$, i.e., a linear combination of the expert and learner's action and let the factor β_i decay from one to zero over time. While this approach would work, data generation in the real world is error-prone and collecting large datasets would require significant amounts of time.

To avoid the collection of a large real-world dataset, we propose a simple yet effective task-specific domain adaptation method to learn an initial vision-based policy from simulated and unsupervised images. To this end, we construct a simulated environment in which there is no visual clutter such that the drones appear in front of a uniform white background (Fig. 3.5a).

Algorithm 1: Multi-agent dataset aggregation.

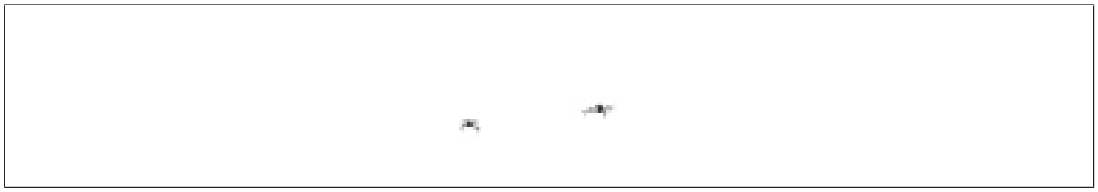
Initialize empty dataset $\mathcal{D} \leftarrow \emptyset$.
Initialize parameters of learned policy $\hat{\pi}_1$.
for $i \leftarrow 1$ **to** N **do**
  Sample trajectories from learned policy $\hat{\pi}_i$ *simultaneously for all agents.*
  Collect dataset $\mathcal{D}_i = \{(\mathbf{o}_t, \pi^*(\mathbf{s}_t))\}_{t=1}^{T}$ of observations from the learned policy $\hat{\pi}_i$ and actions given by expert π^* *for all agents.*
  Aggregate datasets $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$.
  Train new policy $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end
return Best policy $\hat{\pi}_i$ on hold-out validation set $\mathcal{D}_{\text{val}}$.

Next, we collect a $20k$-sample image background dataset from the six onboard cameras during a single-agent flight in the motion tracking hall (Fig. 3.5b). During the flight, we rotate the drone in several yaw configurations and cover as much of the space in the hall as possible to increase the variability in the image data. As a final step, we add the simulated drones onto the background in order to create a dataset that resembles actual drones flying in the motion tracking hall. The real and domain-adapted images are almost indistinguishable to the human eye at the resolution used by the control policy (Fig. 3.5c and 3.5d). The control actions corresponding to the images from the simulated dataset remain unchanged during this process.

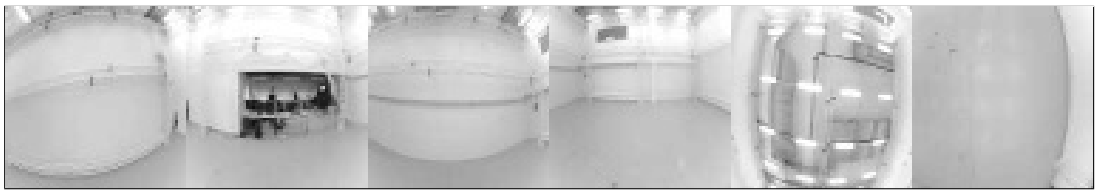
3.3.4 Visual policy

We formulate the vision-based imitation of the flocking algorithm as a regression problem that takes an image (Fig. 3.3) as an input and predicts a velocity command which matches the ground truth velocity command as closely as possible. To produce the desired velocities, we consider a small and efficient convolutional neural network [75] that is geared towards drone navigation. However, unlike [75], we opt for a single-head regression architecture to avoid convergence problems caused by different gradient magnitudes from an additional classification objective during training. This simplifies the optimization problem and the model architecture and thus the resulting controller.

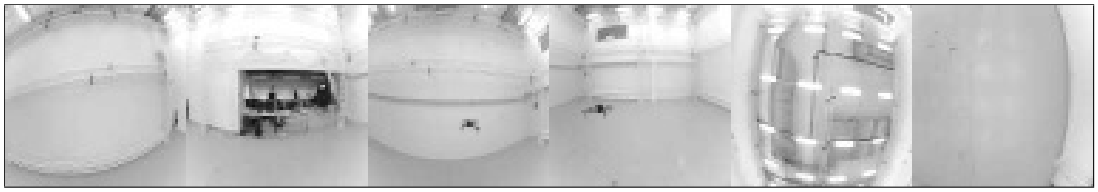
We use mini-batch stochastic gradient descent to minimize the regularized mean squared error loss between predicted and target velocity commands. We employ variance-preserving parameter initialization by drawing the initial weights from a truncated normal distribution [83]. The biases of the model are initialized to zero. The objective function is minimized using the Adam optimizer [84] and an initial learning rate of 10^{-3} which is decayed by a factor of 0.5 after 10 consecutive epochs without improvement on the hold-out validation set. We train the network using a mini-batch size of 128, a weight decay factor of $5 \cdot 10^{-4}$, and a dropout probability of 0.5. We stop the training process as soon as the validation loss plateaus for more than ten consecutive epochs.



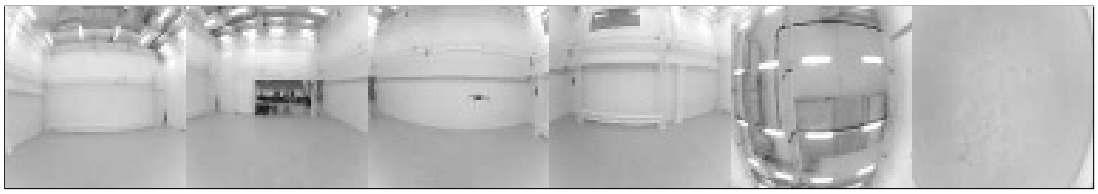
(a) Foreground



(b) Background



(c) Fake sample (foreground + background)



(d) Real sample

Figure 3.5 – Sim-to-real transfer with domain adaptation: simulated foreground + fake background. Example of unsupervised domain adaptation method in which (a) simulated foreground images and (b) real background images from the motion tracking hall are combined into (c) domain-adapted images. For visual comparison, we also show (d) a real sample from a two-agent flight in the motion tracking hall.

The raw images and velocity targets are pre-processed using feature standardization such that each input batch has a mean of zero and a standard deviation of one. For the velocity targets from the flocking algorithm, we perform a frame transformation from the world frame $\mathcal{W}$ into the drone's body frame $\mathcal{B}$ as $\mathbf{v}_i = \mathbf{R}^{\mathcal{B}}_{\mathcal{W}i}\mathbf{v}^{\text{soc}}_i$ where $\mathbf{R}^{\mathcal{B}}_{\mathcal{W}i} \in \text{SO}(3)$ denotes the rotation matrix from world to body frame for robot i and $\mathbf{v}^{\text{soc}}_i$ corresponds to the target velocity command. We perform the inverse rotation to transform the predicted velocity commands from the neural network back into the world frame. In terms of data augmentation, we randomly adjust the image brightness and contrast of each mini-batch by ±25%. Furthermore, we randomly rotate the image cube map and the control command in 90° increments around the body frame z-axis (yaw) such that the vision-based controller becomes invariant to the direction in which agents are predominantly present in the data. In practice, this is equivalent to shifting and

wrapping around the first four images (left, front, right, and back) in 128-pixel increments, as well as rotating the last two images (top and bottom) by 90° increments.

3.4 Simulation results

This section presents an evaluation of the learned controller as a comparison to the target flocking algorithm. We refer to the swarm operating on the learned controller (which relies on visual inputs) as *vision-based*. We refer to the swarm operating on the flocking algorithm (which relies on shared agent positions) as *position-based*. The results show that the proposed controller represents a robust alternative to communication-based systems in which the positions of other agents are shared with other members of the group.

The experiments are performed using the Gazebo simulator [53] in combination with the PX4 autopilot [59] for state estimation and control. The neural network is implemented in PyTorch [85]. We employ the same set of flocking parameters used during the training phase throughout the following experiments. We set the number of agents $N = 9$, the maximum perception radius $r^{\max} = 7\,\mathrm{m}$, and the maximum speed $v^{\max} = 2\,\mathrm{m\,s^{-1}}$. We set the separation, cohesion, and migration gain to $k^{\mathrm{sep}} = 7\,\mathrm{m\,s^{-1}}$, $k^{\mathrm{coh}} = 1\,\mathrm{m\,s^{-1}}$, and $k^{\mathrm{mig}} = 1\,\mathrm{m\,s^{-1}}$, respectively.

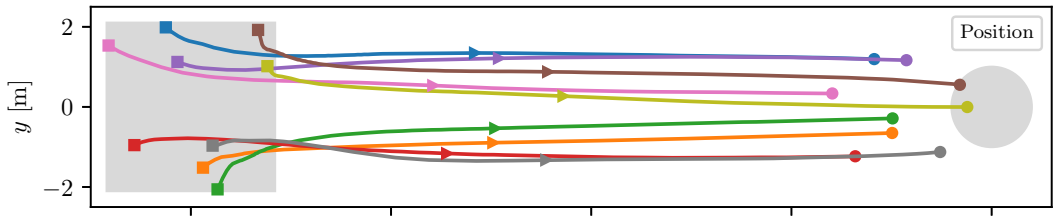
We report the results in terms of minimum and maximum inter-agent distances, two complementary metrics that describe the state of the swarm at a given time step. The minimum and maximum inter-agent distance are direct indicators for successful collision avoidance, as well as general segregation of the swarm, respectively. Two conditions are tested: a first one in which all agents share a common migration goal, and a second one in which a subset of the agents have an opposing migration goal.

3.4.1 Common migration goal experiment

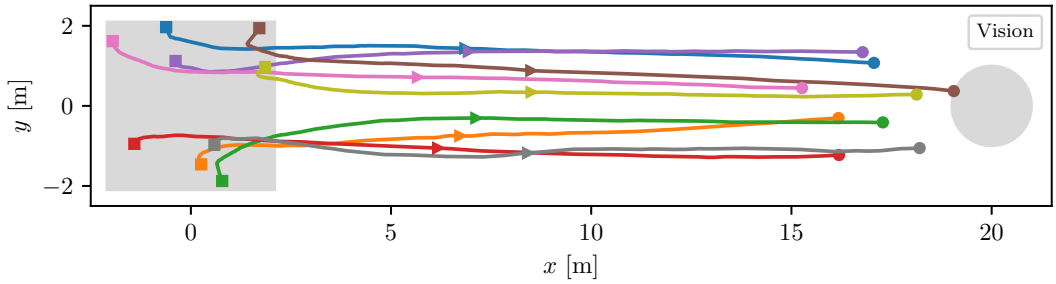
In the first experiment, we give all agents the same migration goal and show that the swarm remains collision-free during navigation. The *vision-based* and the *position-based* swarm exhibit remarkably similar behavior while migrating (Fig. 3.6a and 3.6b). For the *vision-based* controller, the velocity commands predicted by the neural network are sent to the agents in their raw form without any further processing. The *vision-based* swarm matches the *position-based* one very well since the inter-agent distances do not deviate significantly over the course of the entire trajectory (Fig. 3.6c). The minimum inter-agent distance remains larger than the collision threshold of 1 m, which indicates that the neural network policy has learned to keep a minimum inter-agent distance and thus to avoid collisions.

3.4.2 Opposing migration goals experiment

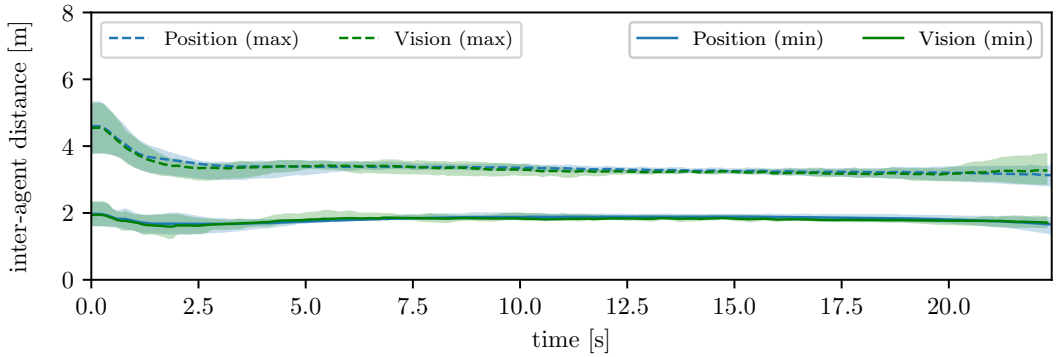
In this experiment, we assign different migration goals to two subsets of agents. The first group, consisting of five agents, is assigned the same waypoint as in Sec. 3.4.1. The second group,



(a) Paths taken using *position-based* flocking



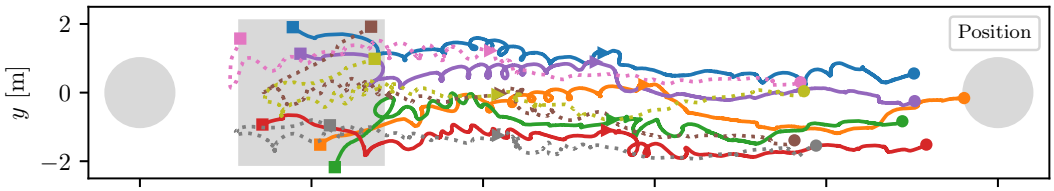
(b) Paths taken using *vision-based* flocking



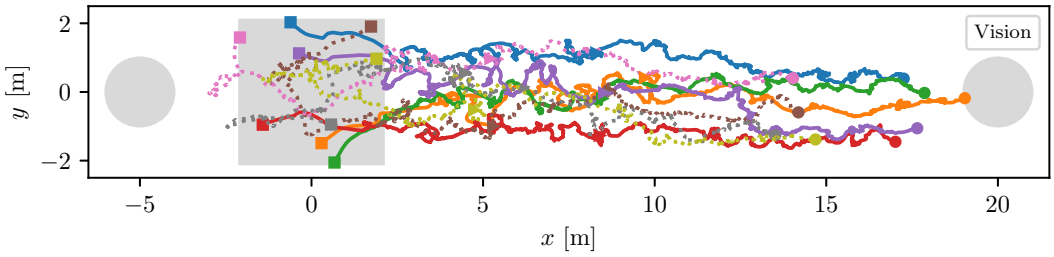
(c) Minimum and maximum inter-agent distances

Figure 3.6 – Position-based vs. vision-based flocking with a common migration goal in simulation. Top view of a swarm migrating using the (a) *position-based* and (b) *vision-based* controller, as well as their respective (c) minimum and maximum inter-agent distances over time. The path of each agent is shown in a different color. The colored squares, triangles, and circles show the agent positions during the first, middle, and last time step, respectively. The gray square and gray circle denote the spawn area and the migration point, respectively. The mean minimum distance between any pair of agents is denoted by a solid line, whereas mean maximum distances are shown as a dashed line. The colored shaded regions show the minimum and maximum distance between any pair of agents.

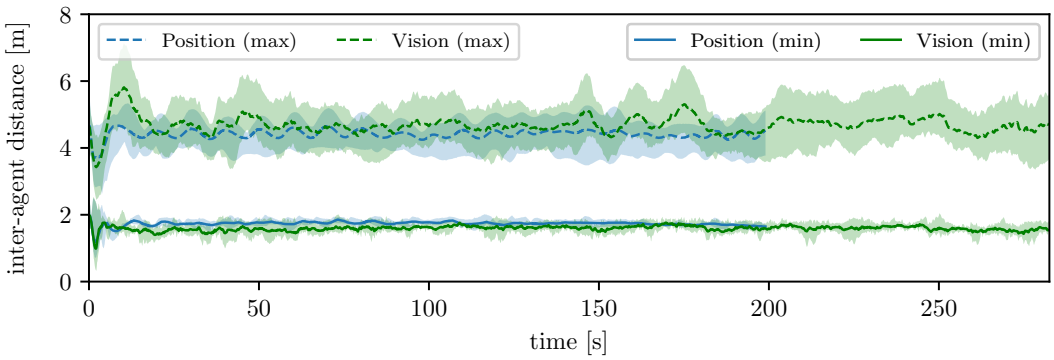
consisting of the remaining four agents, is assigned a migration point on the opposite side with respect to the first group. The *position-based* and *vision-based* swarm exhibit very similar migration behaviors (Fig. 3.7a and 3.7b). In both cases, the swarm cohesion is strong enough



(a) Paths taken using *position-based* flocking



(b) Paths taken using *vision-based* flocking



(c) Minimum and maximum inter-agent distances

Figure 3.7 – Position-based vs. vision-based flocking with opposing migration goals in simulation. Top view of a swarm migrating using the (a) *position-based* and (b) *vision-based* controller, as well as their respective (c) minimum and maximum inter-agent distances over time. The path of each agent is shown in a different color. The colored squares, triangles, and circles show the agent positions during the first, middle, and last time step, respectively. The gray square and gray circle denote the spawn area and the migration point, respectively. The waypoint on the right is given to a subset of five agents (solid lines), whereas the waypoint on the left is given to a subset of four agents (dotted lines). The mean minimum distance between any pair of agents is denoted by a solid line, whereas mean maximum distances are shown as a dashed line. The colored shaded regions show the minimum and maximum distance between any pair of agents. The *position-based* distances plot does not continue until the last time step since the agents reach the migration point faster than the *vision-based* agents.

to keep the agents together despite the diverging migration goals. Note that the *vision-based* swarm reaches its migration goal far later than the *position-based* swarm.

3.5 Real-world results

We propose three experiments involving two quadrotors to show that the learned controller can perform vision-based markerless flight in the real world. We conclude with an attribution study that visualizes the regions of the visual input that contribute the most to the neural network's predictions. All flights are performed in a motion tracking hall that is equipped with 26 OptiTrack cameras. Each drone receives its ground truth pose via Wi-Fi at a frequency of 100 Hz.

3.5.1 Circle experiment

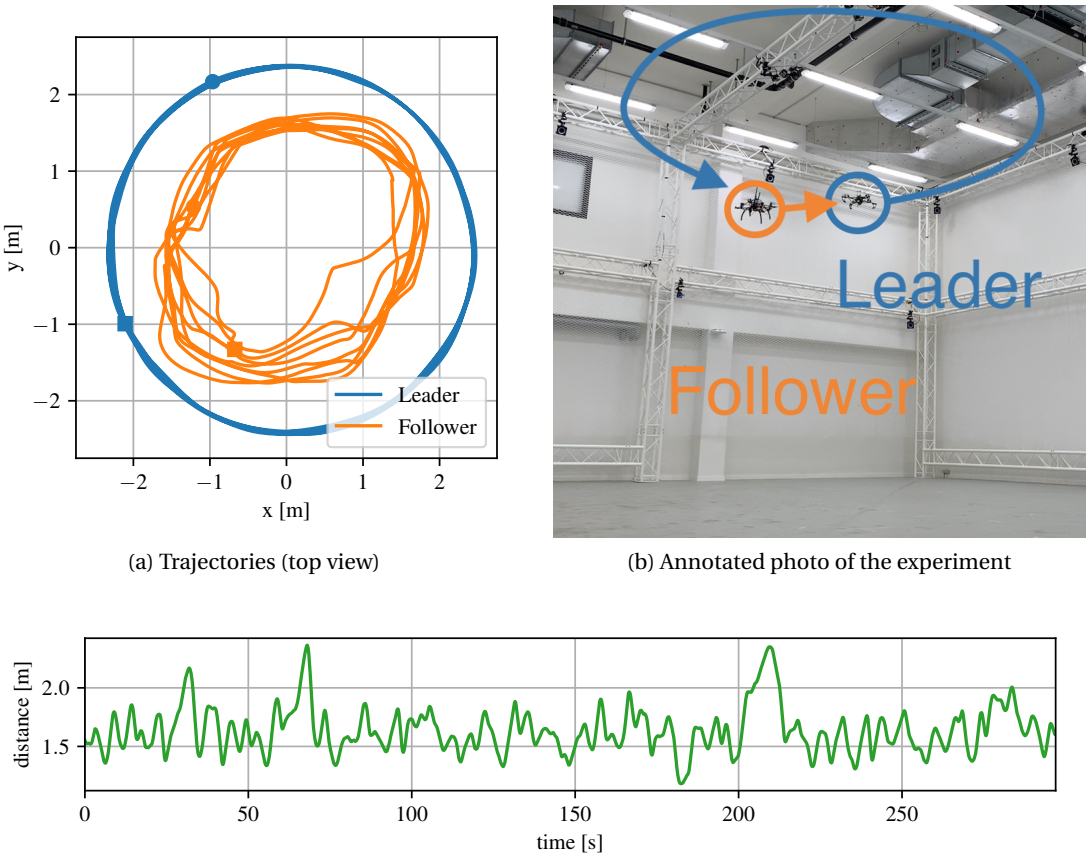
The circle experiment showcases the ability of the learned policy to maintain cohesion with another agent. To this end, the leader drone is given a circular trajectory, whereas the vision-based follower uses raw velocity commands generated onboard by the neural network. We set the radius of the circle as 2.5 m and the angular velocity along the circular trajectory to $10\,^{\circ}\,\mathrm{s}^{-1}$. The follower drone keeps a stable distance between itself and the leader drone during a representative 6 min flight (Fig. 3.8). One can observe that the visual policy can recover from small mistakes reliably, most notably after the 70 s and 210 s marks (Fig. 3.8c).

3.5.2 Carousel experiment

The carousel experiment can be seen as an extension of the circle scenario with the added difficulty that the altitude of the leader is now modulated by a sinusoid as well. The parameters of the circle trajectory remain the same, but we add a sinusoid component to the altitude tracked by the leader drone. The altitude component has an amplitude of 1 m and the same frequency as the horizontal components, which leads to a tilted circular trajectory (Fig. 3.9b). The vision-based follower thus needs the ability to operate in full 3D space in order to stay cohesive with the leader. The inter-agent distance in the carousel experiment increases slightly compared to the circle experiment, especially when the leader agent deviates the most from the average flight altitude. Nevertheless, the vision-based drone can maintain a steady cohesion with the leader (Fig. 3.9). Upon closer examination of the altitude of both agents over time, it is clear that the follower can modulate its altitude as a reaction to the leader. The extreme points in altitude of the follower are indeed aligned with the intersection points of the two curves (Fig. 3.9c).

3.5.3 Push-pull experiment

The motivation for the push-pull experiment is the validation of the separation ability of the vision-based flocking policy. In both the circle and carousel experiments, the follower drone is never on a direct collision course with the leader. In order to encourage collisions, we let the leader navigate between two waypoints and position the follower in the middle. We further set both the x and the z-component of the velocity command computed by the



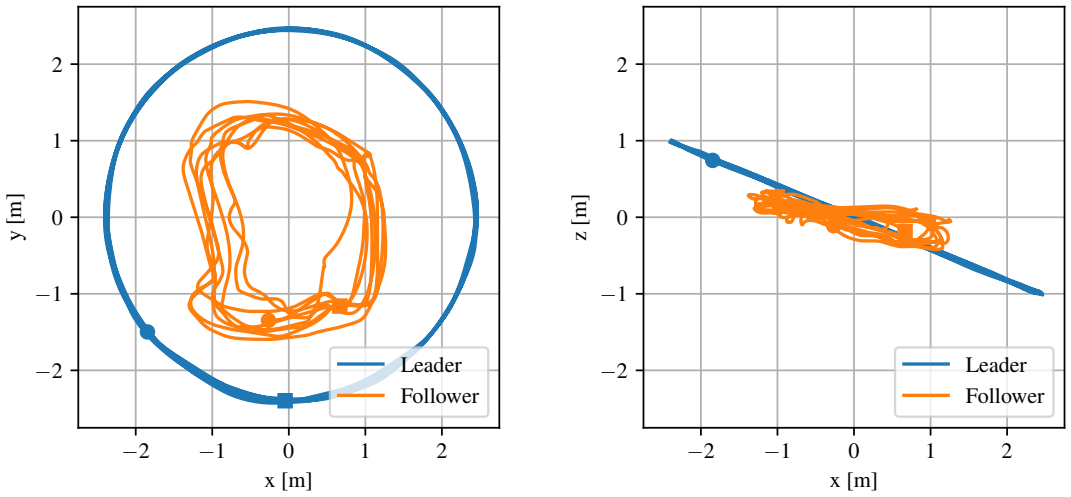
(a) Trajectories (top view)

(b) Annotated photo of the experiment

(c) Inter-agent distance over time

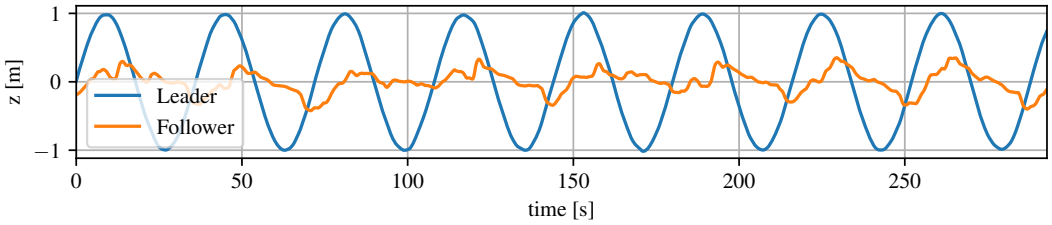
Figure 3.8 – Vision-based leader-follower flight with circular leader trajectory in reality. We show (a) the top view of the trajectories, (b) an annotated photo of the experimental setup, and (c) the inter-agent distance over time.

neural network to zero in order to fix the follower's degrees of freedom to a line defined by the two waypoints. This adjustment is necessary to show collision avoidance since leaving the control input unrestricted would degenerate into a cohesion-like scenario where collisions are not explicitly encouraged. Moreover, since the drones may get into situations where they are on top of each other, downwash may blur the lines between the separation due to the learned controller and the physical repulsion due to the airflow. The vision-based follower avoids collisions and maintains a constant equilibrium distance to the leader drone (Fig. 3.10). This behavior results directly from the spring-like dynamics of two agents in which one is following the flocking algorithm. The oscillations in the position may occur because distances smaller than equilibrium are penalized quadratically, while distances larger than equilibrium are penalized linearly by the flocking rules (Fig. 2.2). In addition, noise in the raw control command leads to small and sudden repulsive maneuvers that are quickly compensated (Fig. 3.10c).



(a) Trajectories (top view)

(b) Trajectories (side view)

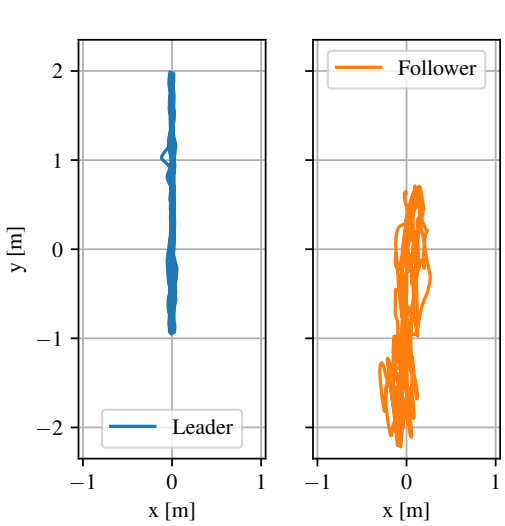


(c) Vertical position over time

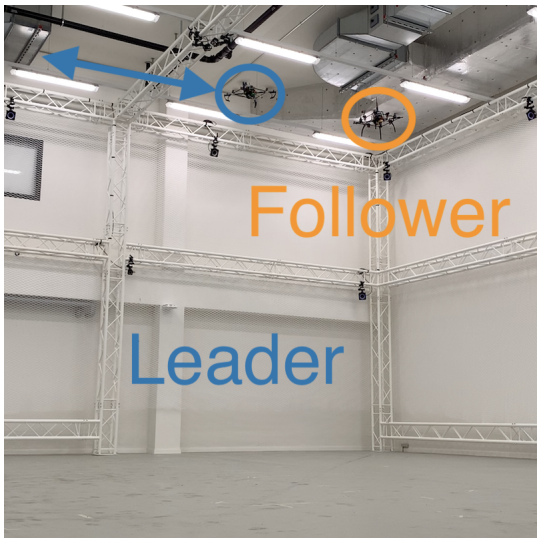
Figure 3.9 – Vision-based leader-follower flight with height-modulated circular leader trajectory in reality. We show (a) the top view, (b) the side view of the trajectories, and (c) the vertical position over time.

3.5.4 Attribution study

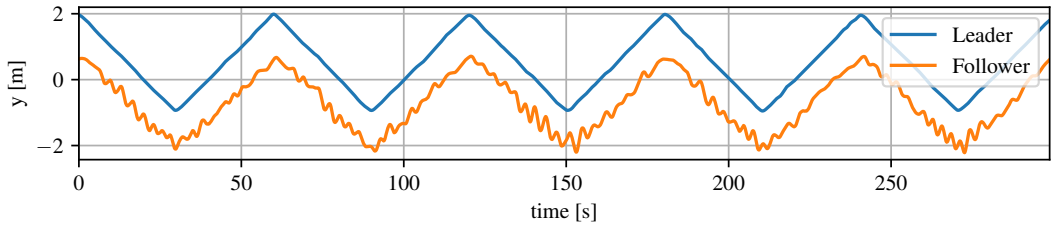
Since the *vision-based* controller provides a very tight coupling between perception and control, the need for interpretation of the learned behavior arises. To this end, we employ the Grad-CAM (gradient-weighted class activation mapping) attribution method [86], which shows how much influence each pixel in the input image has on the predicted velocity command (Fig. 3.11). In particular, we opt for the Grad-RAM (gradient-weighted *regression* activation mapping) formulation since we use a regression objective to predict velocity commands [87]. More specifically, we compute the gradients for the heat map with respect to the last convolutional layer of the neural network in which the individual feature maps have a spatial size of only 8×48 pixels. We then employ bilinear upsampling to increase the resolution of the resulting saliency map before we blend it with the original input image using a jet colormap for visualization purposes. The attribution map can be generated very efficiently using one forward and backward pass and could therefore serve as a valuable attention-like input for



(a) Trajectories (top view)



(b) Annotated photo of the experiment



(c) Positions over time

Figure 3.10 – Vision-based leader-follower flight with linear leader trajectory in reality. We show (a) the top view of the trajectories (separated for clarity), (b) an annotated photo of the experimental setup, and (c) the longitudinal position over time.

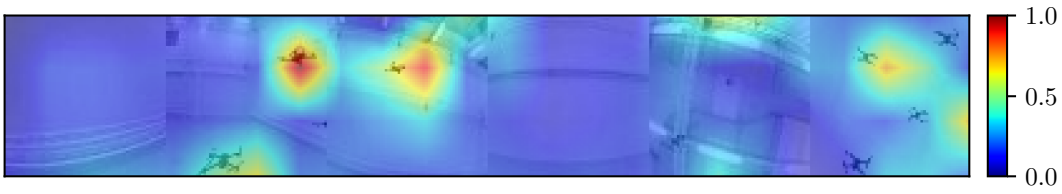
further real-time processing.

One can observe that the network is effectively localizing the other agent spatially in the visual input. However, the network is putting non-zero importance on regions with more visual clutter such as the control room in the backward-facing camera (Fig. 3.11). One may note that the most salient region is not perfectly matching the location of the visible agent, which can be attributed to the low spatial resolution of the activations generated at the last convolutional layer.

3.6 Conclusions

This chapter presented a machine learning approach to the problem of collision-free and coherent motion of a dense swarm of quadcopters. The agents learn to coordinate themselves

(a) Real images



(b) Fake images

Figure 3.11 – Attribution heat maps computed for (a) real images taken from a leader-follower flight and (b) fake samples composed of simulated foreground and real background images. The attribution heat maps visualize the relative importance of each pixel in the visual input of the drone towards its velocity command. Red regions contribute most to the magnitude of the control command, whereas blue regions contribute the least. Best viewed in color.

entirely via visual inputs in 3D space by mimicking a flocking algorithm. The learned controller removes the need for communication of positions among agents and thus presents the first step towards a fully decentralized vision-based swarm of drones.

The algorithm naturally handles navigation tasks by adding a migration term to the predicted velocity of the neural network policy. The method does not require camera calibration since the policy computes velocity commands directly from raw images. The absence of any intermediate spatial representation of the other agents means that the entire policy can be optimized end-to-end — without possibly occurring losses at module boundaries such as object detection or tracking.

While end-to-end learning has many desirable properties such as the ones mentioned above, it comes with several key limitations. Firstly, it is very hard to debug and find errors since the entire algorithm — from perception to control and everything in between — is encoded entirely in the neural network weights. To gain useful insights about which computations these weights are performing, attribution methods such as the one employed in this chapter (Sec. 3.5.4) can be extremely helpful. For example, during our initial experiments, we found that the neural network would attend to the cameras of the motion tracking hall because of their similarity to the drones at low resolution. This was a very useful indicator that the training data did not include enough objects that may confuse the policy. Unfortunately, collecting more training data or more data augmentation is often the only remedy for performance

issues with the method presented here.

Secondly, the lack of modularity is another key limitation. Due to the tight coupling between perception and control, we lose flexibility such as the ability to change parameters that affect the swarm behavior. For example, if the requirements change and the swarm should operate at larger inter-agent distances than encoded by then neural network weights, the entire policy needs to be re-trained with an increased separation gain. The method proposed here also prevents the use of dynamic swarm behaviors such as expansion and contraction since there is no way to adjust parameters during runtime. Another example is a scenario in which the quadcopters have a different visual appearance than the neural network policy is trained on. While domain adaptation and data augmentation can help, the policy still needs to be re-trained end-to-end for changes to take effect. Therefore, employing separate modules for perception and control can increase flexibility since parameters can be adjusted depending on the task.

Finally, the performance assessment is difficult and may be dangerous. The output of the neural network policy is a control command and, as such, requires a dynamical system in order to be properly assessed. In the case described here, it requires the interaction of several quadcopters with partially trained policies which may pose a security hazard. During the initial experiments, we employed a proxy for performance which consisted in comparing ground truth and estimated velocity commands in unseen agent configurations and environments. While this is useful to see if the neural network has learned useful representations, it is equivalent to the loss function used to train it and not an actual performance metric. Adopting a more modular approach can be beneficial since each module, e.g., perception and control, can be tested in isolation with performance metrics that are relevant for the task.

Another problem we identified during experiments is the strong effect of physical downwash in dense three-dimensional configurations (Sec. 3.10). For example, if a quadcopter hovers in close proximity above another one, its spinning propellers create an airflow that pushes the lower quadcopter, which in turn needs to increase its thrust in order to compensate for the downward force. Downwash compensation is not only energetically inefficient but also makes the evaluation of the system more difficult since the repulsion between drones can be caused by deliberate control or aerodynamic forces — or both. While methods exist that take the effect of downwash into account [88], they usually require the planning of trajectories as opposed to reactive control strategies. In the remainder of the thesis, we will therefore limit the swarm to horizontal planar configurations embedded in three-dimensional space.

Regarding future work, a natural subsequent step will be to scale up the real-world experiments with more vision-based drones, as well as the transfer to outdoor scenarios where ground truth positions will be obtained using RTK-capable GNSS receivers. However, the current drone hardware is not well-suited for this task since the camera resolution is too low to distinguish other drones in front of cluttered backgrounds and difficult lighting conditions. The transfer to outdoor scenarios therefore requires upgrading the cameras, whose larger resolutions will

also require more onboard computational power for image processing.

4 Modular vision-based flocking using object detection and tracking

This chapter aims to address the limitations of end-to-end learning and proposes a modular approach to vision-based swarm control based on object detection and multi-target tracking. Each drone uses a convolutional neural network to localize its neighbors in omnidirectional images. Rather than manually labeling a dataset to train the neural network, we automatically generate bounding box annotations using background subtraction by systematically flying a quadcopter in front of a static camera. We use a multi-target tracker to transform the predicted bounding boxes into estimates of relative positions and velocities used for high-level control. We evaluate the approach in simulation and with a group of three real quadcopters in a challenging outdoor environment. The results show that the drones can navigate safely and cohesively without relying on external localization, wireless communication, or visual fiducial markers. Instead, the approach relies entirely on local visual inputs that are processed onboard in real-time.

The work presented in this chapter is adapted from [61][1]:

- F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, Apr. 2021.

4.1 Introduction

Drone swarms have a large socio-economic potential and can serve in a variety of real-world applications [1]. For example, drones can be leveraged to automatically monitor crops, safely inspect confined spaces, or quickly deliver medicine to inaccessible locations. Operating these vehicles in swarms could bring increased robustness to failures, larger area coverage, and faster task completion times [7].

Despite this potential, decentralized control has been a limiting factor in the deployment

[1] Video: https://youtu.be/wU8-Wm9_YLs.

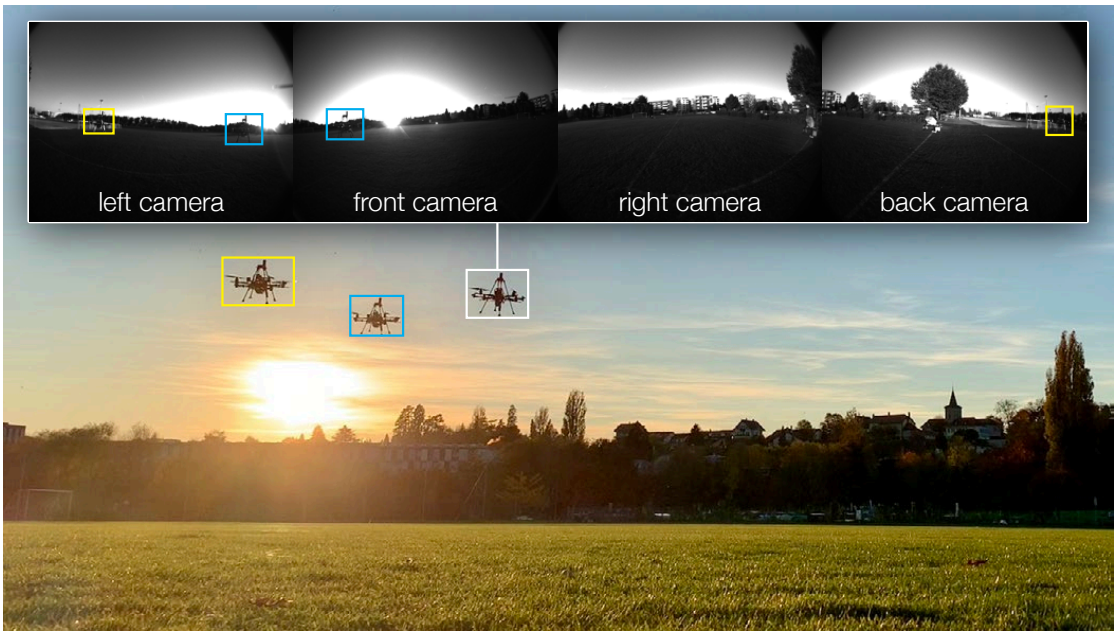


Figure 4.1 – Photo taken during outdoor experiments with detection annotations. The quadcopters avoid collisions with each other and remain cohesive as a group while performing a variety of navigation tasks. Each agent detects its neighbors in real-time from omnidirectional images. There is no communication of state information between agents, and relative positions and velocities are estimated onboard using local visual inputs.

of drone swarms. For instance, large groups of quadcopters can be used to perform awe-inspiring aerial choreographies in the night sky. These robotic light shows are a true feat of engineering but individual drones are far from autonomous: their motion is centrally controlled by a ground computer that precomputes their trajectories and continuously monitors their positions. Hence, the ground computer represents a single point of failure. Researchers attempted to remove the central computer by equipping drones with hardware that allows them to wirelessly communicate with each other. Notable examples of these decentralized swarms feature bearing-only formation control [16], exploration of unknown environments [15], as well as flocking with ten fixed-wing drones [17] or thirty quadcopters [14].

Scaling up decentralized drone swarms is complicated by the limitations of wireless communication. As the number of robots increases, the communication channels may become saturated and possibly jammed since the data transfer volume scales quadratically with the robot count [18]. Frequent retransmissions of messages can lead to delays that render the control of each drone extremely difficult. Researchers have experimented with different sensory modalities such as sound [30], but vision seems to be the most scalable approach to address the relative localization problem. Indeed, there is evidence to support that vision is the primary sensory modality that enables collective motion in animal groups [22].

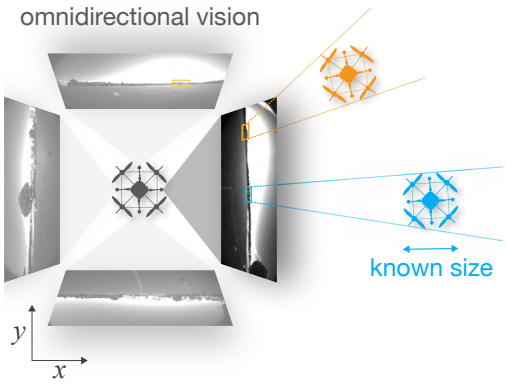
Visual detection of outdoor flying drones is challenging because they are relatively small

and can fly in environments with large amounts of background clutter and difficult lighting conditions [26]. Additionally, the visual detection algorithm must run onboard the drones, whose own motion may generate motion blur and whose small dimensions and lightweight mass can restrict computational capabilities. To simplify the relative localization problem, researchers have mounted different types of easily detectable visual markers on the drones [89, 80]. However, this approach is less general since specialized hardware has to be mounted on each drone, which can result in increased weight and drag, thus reducing the energetic autonomy of the drones.

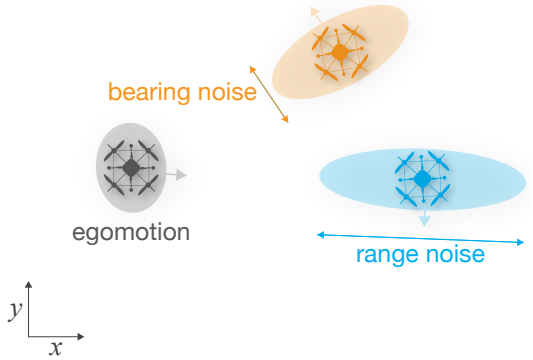
In recent years, markerless detection of drones has become an active research topic. In [90], the authors use a boosted cascade of classifiers in combination with visual tracking and finite set filtering to estimate the positions and velocities of markerless drones from a mostly static observer. However, the method is applied in post-processing and not validated in a sense-and-avoid setting. In [91], the authors propose a combination of stereo vision and convolutional detection to enable leader-follower flight. However, the to-be-localized agent is always visible in front of the clear sky which simplifies the detection problem and would be impossible to guarantee in a self-organized flock. Moreover, the limited field of view and processing delays in the proposed system are known to be problematic when flying in dense swarms [14, 67]. Other notable examples of markerless detection include approaches based on template matching and morphological filtering [92], as well as convolutional neural networks [93].

The previous chapter (Chapter 3) proposes a fundamentally different approach to visual flocking based on imitation learning. Rather than detecting neighboring agents, we predict flocking algorithm commands directly from omnidirectional visual inputs, which allow the drones to remain collision-free and cohesive. The approach is validated with leader-follower experiments in an indoor environment but its reliance on end-to-end learning means that the entire monolithic neural network has to be retrained each time the task and/or visual appearance of the drones change. Adopting a more modular approach can be beneficial since the flocking algorithm may easily be exchanged for another task-dependent controller, and only the detector would have to be retrained for different drone appearances or environmental conditions. In all of the above cases, the algorithms are validated on a single agent and not in a multi-robot control setting.

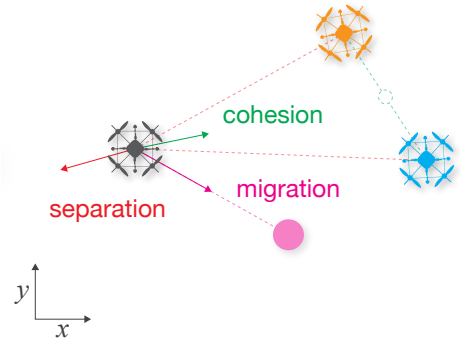
Here, we propose a modular detection and tracking algorithm that enables collision-free and cohesive navigation for drone swarms. We automatically label images of drones using background subtraction to generate a dataset for the drone detector. We show that the detector can localize other drones in the presence of background clutter from onboard a flying drone despite being trained on images from a static camera. We assess the method with a dense group of three real quadcopters that flock in planar configurations in an outdoor environment with difficult lighting conditions. The omnidirectional camera configuration of the drone is specifically designed to enable safe operation in swarms regardless of the agent configuration. The overall proposed flocking algorithm is modular since each component, i.e., detection, tracking, and control, is self-contained and can be evaluated independently. To the best of our



(a) Step 1: Real-time drone detection



(b) Step 2: Multi-agent state tracking



(c) Step 3: Potential-field-based control

Figure 4.2 – Overview of the processing steps of the vision-based flocking algorithm for a single time step: a) *detection*, b) *tracking*, and c) *control*. Step 1 (*detection*): we detect neighboring agents from omnidirectional images to estimate their relative range and bearing from the camera intrinsics and the drone's known physical size. Step 2 (*tracking*): we track the positions and velocities of the detected agents using a linearized observation model of range and bearing, as well as a state transition model that takes the focal agent's egomotion into account. Step 3 (*control*): we control the focal agent using a Reynolds-rules-based flocking algorithm that keeps the swarm collision-free and cohesive while following a navigation goal.

knowledge, this is the first entirely vision-based flock that does not depend on visual markers to simplify mutual detections.

4.2 Method

The proposed approach to vision-based flocking can be divided into the following steps: *detection*, *tracking*, and *control* (Fig. 4.2). Firstly, the *detection* module (Fig. 4.2a) takes grayscale images from an omnidirectional camera setup as inputs and outputs bounding boxes of nearby drones in real-time (Sec. 4.2.2). Secondly, the *tracking* module (Fig. 4.2b) transforms

the bounding boxes into range and bearing measurements using the known dimensions of the drones. Their relative positions and velocities are then estimated from the noisy measurements using a multi-agent state tracker (Sec. 4.2.2). Finally, the *control* module (Fig. 4.2c) applies a flocking algorithm to the relative positions to obtain high-level control commands that keep the drones collision-free and cohesive (Sec. 4.2.2). In the tracking and control steps, we assume that the agents are moving on a horizontal plane. We mainly introduce this constraint to be able to attribute their mutual repulsion to the flocking algorithm and avoid the effects of physical downwash that may be caused by nearby agents.

4.2.1 Real-time monocular drone detection

We first describe how we collect the drone image dataset and the procedure used to label the dataset using background subtraction (Sec. 4.2.1). We then outline how we train the detector to perform real-time drone detection (Sec. 4.2.1).

Automatic drone labeling with background subtraction

We use background subtraction to automatically generate a labeled image dataset for the object detector (Fig. 4.3). We record images from a stationary camera mounted on a tripod and manually fly a quadcopter within its field of view. The image data is recorded under varying lighting conditions in both indoor and outdoor environments that contain large amounts of background clutter (Fig. 4.4). For each scene, the camera location and orientation are carefully chosen such that the quadcopter is the dominant source of motion. The final dataset consists of 9 891 training and 1 931 testing examples.

In post-processing, we apply a nearest neighbor background subtraction algorithm [94] to the sequence of images to learn a background model of the scene. We extract a foreground mask of the moving parts of the image (and therefore the quadcopter) by computing the element-wise difference of the input image and the background, followed by a thresholding step. The ground truth label is obtained by filtering out the largest contour present in the foreground mask and enclosing it with an axis-aligned rectangular bounding box.

Training the real-time drone detector

We train a single-stage convolutional object detector on the automatically annotated image dataset to obtain drone detections. We opt for the YOLOv3-tiny architecture [95] due to its favorable tradeoff between detection accuracy and inference speed on embedded devices. The network architecture is comprised of a total of 13 convolutional layers that are interspersed with max-pooling operations and leaky rectified linear units (ReLUs). The final detections are computed independently at two different scales and subsequently filtered by non-maximum suppression to account for bounding box overlaps.

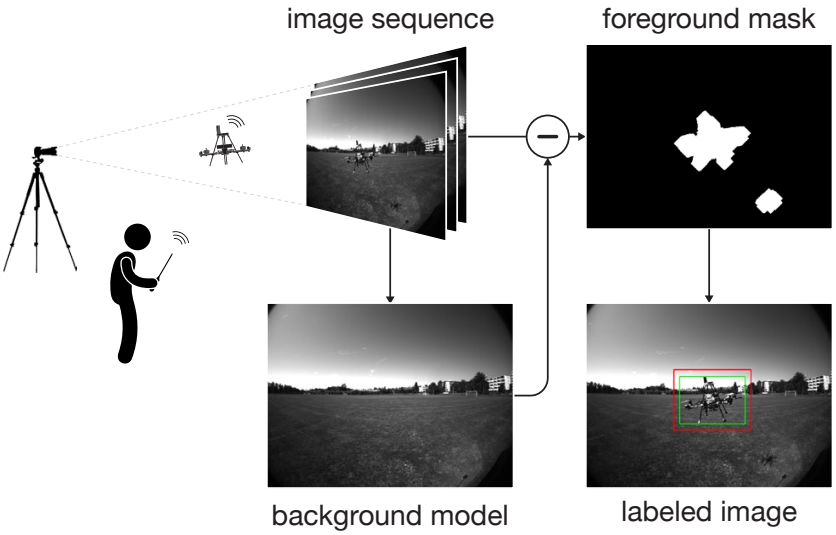
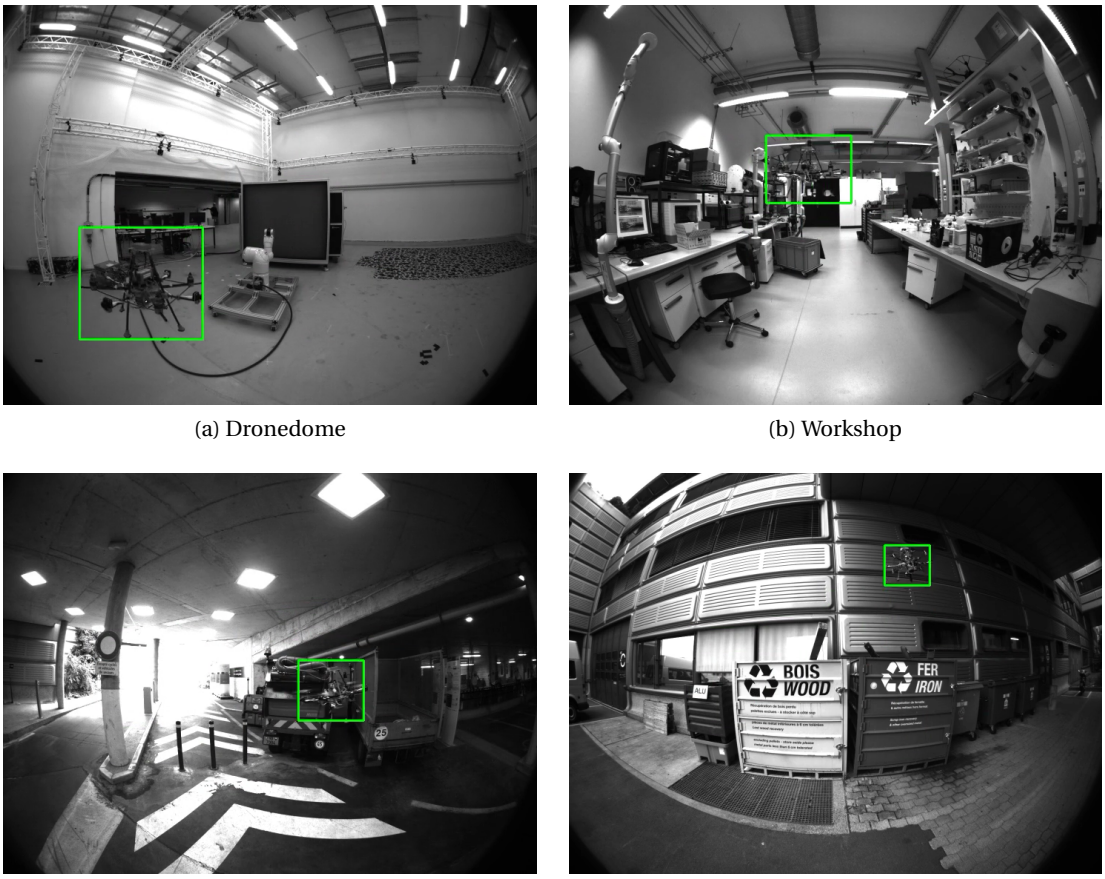


Figure 4.3 – Schematic overview of the background subtraction process for automatic generation of detection annotations. We manually fly a quadcopter in front of a static camera and use background subtraction to generate bounding box annotations for the drone. We find that the most precise bounding box labels are obtained by dilating the foreground mask since it eliminates discontinuities that occur due to the mechanical design of the drone. Enclosing the dilated mask with a bounding box (red rectangle) overestimates the size of the drone. We therefore scale the bounding box down (green rectangle) to obtain a precise label. We record six flights of roughly one-minute duration in the above target environment.

We make a few notable modifications to the training procedure of the original publication [95]. Firstly, we replace the mean-squared error localization loss with an objective that is based on the generalized intersection over union (GIoU) [96]. The GIoU loss is nonzero even if bounding box predictions have no overlap (therefore producing gradients) and directly optimizes the intersection over union (IoU) metric [97]. Secondly, we employ two recently popularized data augmentation techniques to improve detection accuracy: 1) multi-scale training and 2) mosaic augmentation [98]. In multi-scale training, each training batch is scaled at random by up to ±50% of its side length to make the detector invariant to object scales. Mosaic training refers to concatenating four random training samples along their spatial dimensions to obtain an image collage. The resulting four-image mosaic is subsequently cropped randomly in the center to obtain a new training sample. We use hold-out cross-validation to find suitable values for the most critical hyperparameters.

The parameters of the network are initialized with weights that are pre-trained on the COCO [99] dataset. The detector is then fine-tuned on a single *drone* class with stochastic gradient descent and Nesterov momentum of $\mu = 0.937$. We modulate the learning rate using a cosine annealing schedule [100] with an initial learning rate $\eta = 0.01$ and a final learning rate $\eta_f =$



(a) Dronedome (b) Workshop

(c) Parking (d) Passage

Figure 4.4 – Example images with bounding box annotations from the dataset generated using background subtraction. We record image data from a static camera in both indoor (top row) and outdoor (bottom row) environments. The environments are selected to maximize the variety of background clutter and lighting conditions. We record three flights of roughly one-minute duration in each of the above environments.

0.0005. We also employ a weight decay term of $\lambda = 0.0005$. The total loss is computed as

$$\mathcal{L} = k^{\text{bal}}\mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{obj}} \tag{4.1}$$

where $\mathcal{L}_{\text{loc}}$ denotes the GIoU-based localization loss and $\mathcal{L}_{\text{obj}}$ refers to the binary cross-entropy objectness loss. We set the hyperparameter $k^{\text{bal}} = 0.055$ to balance the losses and to account for their different magnitudes during training. Unlike the original article [95], we omit the classification loss term since we are only training on a single class.

Our highest-scoring model achieves an average precision (AP@0.5) of 98.9% [97] at a confidence threshold of $p^{\text{conf}} = 0.001\%$ on the hold-out test set after 77 epochs of training. The model can be trained in less than 1 h using a batch size of 64 on a recent GPU such as the

Nvidia GeForce RTX 2080Ti. We also evaluate the model at different image scales on the onboard computer (Sec. 4.3.1) to determine a reasonable speed/accuracy tradeoff of the detector. We find that performing inference at a resolution of 512×384 pixels (in batches of four, one image per camera) provides accurate detections at a frequency of around 5 Hz. For the experiments, we set the confidence threshold to $p^{\text{conf}} = 50\%$ and use a non-maximum suppression threshold of $p^{\text{nms}} = 60\%$.

We have also experimented with spatiotemporal detection architectures that use a sequence of images to detect drones [101]. The experiments were inspired by the combination of appearance and motion cues for the detection of flying objects using a single moving camera [102, 26]. The key idea was to use deformable convolutions [103] to spatially shift and fuse features from past frames to increase the detection accuracy of the current frame. While the approach outperforms the detector described here by a slight margin, it does so at a noticeable additional computational cost. Moreover, the computational overhead was higher on the embedded hardware, most likely because deformable convolutions are not well optimized (Sec. 4.3.1).

4.2.2 Multi-agent localization and tracking

Relative localization based on known physical size

We compute the relative location of the drone detections using their apparent size in the field of view and the camera parameters. The camera parameters are obtained using the Kalibr visual-inertial calibration toolbox [104]. We use the equidistant — or Kannala-Brandt — camera model for its compatibility with fisheye lenses and its resulting sub-pixel reprojection errors [105]. The projection function of the equidistant camera model can be formalized as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \theta_d \frac{x}{r} \\ f_y \theta_d \frac{y}{r} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{4.2}$$

where $[u, v]^\top$ are the pixel coordinates, $[x, y, z]^\top$ the camera coordinates, $[f_x, f_y]^\top$ the focal lengths, $[c_x, c_y]^\top$ the optical centers in horizontal and vertical direction, respectively, and

$$r = \sqrt{x^2 + y^2}, \tag{4.3}$$

$$\theta = \text{atan2}(r, z), \quad \text{and} \tag{4.4}$$

$$\theta_d = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 \tag{4.5}$$

denote the range, bearing, and polynomial that models distortions with the coefficients $[k_1, k_2, k_3, k_4]^\top$. Note that we use the conventions from the camera calibration literature to denote range r and bearing θ which differ from the notation in the rest of the section [105, 106].

To obtain the relative position estimate from the detection, we first compute the unit-norm bearing vector to the center $\boldsymbol{\beta}^{\text{ctr}}$ and one of the extreme points $\boldsymbol{\beta}^{\text{ext}}$ of the image bounding box

from the intrinsic camera parameters. We assume the drone can be enclosed by a bounding cube with side length l which is reasonable given its mechanical design (Fig. 4.5). We can then compute the approximate distance to the three-dimensional center of the detected object as

$$d = \frac{l/2}{\tan(\alpha)} + l/2 \tag{4.6}$$

where $\alpha = \cos^{-1}(\boldsymbol{\beta}^{\text{ctr}} \cdot \boldsymbol{\beta}^{\text{ext}})$ denotes the angle between the unit-norm bearing vectors. Note that the second term in the above equation accounts for the depth of the object.

Multi-agent state estimation using random finite sets

We use the Gaussian mixture probability hypothesis density (GM-PHD) filter [107] to filter out spurious false-positive detections and to estimate the positions and velocities of nearby agents over time. We briefly describe the workings of the filter but refer the reader to [107] for more details. We omit the subscript i to denote the dependence on the focal agent for notational brevity. The following steps are computed independently for each agent in a decentralized fashion.

Theory. The GM-PHD filter takes as input a set of relative localization measurements $\mathcal{Z}_k = \{\mathbf{z}_{k,1}, \ldots, \mathbf{z}_{k,M_k}\}$ and computes an output set of agent states $\mathcal{X}_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,J_k}\}$ for each discrete time step k. The states can be described as a single intensity that consists of a weighted sum of Gaussian components of the form

$$v_k(\mathbf{x}_k) = \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}(\mathbf{x}_k; \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}) \tag{4.7}$$

where $w_k^{(i)}$ denotes the weight associated with each of the J_k Gaussian components which are described by their mean $\mathbf{m}_k^{(i)}$ and covariance $\mathbf{P}_k^{(i)}$. Each of the Gaussian components is then propagated with a *prediction* and *update* step, similar to the Kalman filter.

The *prediction* step can be formalized as

$$v_{k|k-1}(\mathbf{x}_k) = \sum_{i=1}^{J_k} w_{k|k-1} \mathcal{N}(\mathbf{x}_k; \mathbf{m}_{k|k-1}^{(i)}, \mathbf{P}_{k|k-1}^{(i)}) + \gamma(\mathbf{z}_k) \tag{4.8}$$

with respective weight, mean, and covariance

$$w_{k|k-1}^{(i)} = p_{s,k} w_{k-1}^{(i)} \tag{4.9}$$

$$\mathbf{m}_{k|k-1}^{(i)} = \mathbf{F}_{k-1} \mathbf{m}_{k-1}^{(i)} + \mathbf{B}_{k-1} \mathbf{u}_{k-1} \tag{4.10}$$

$$\mathbf{P}_{k|k-1}^{(i)} = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^{(i)} \mathbf{F}_{k-1}^{\top} + \mathbf{Q}_{k-1} \tag{4.11}$$

where $\mathbf{F}_{k-1}$ is the state transition matrix and $\mathbf{Q}_{k-1}$ is the process noise covariance. To account

for the egomotion of the observing drone, we include a control input matrix $\mathbf{B}_{k-1}$ and its control input $\mathbf{u}_{k-1}$. We let $p_{s,k}$ denote the probability that a Gaussian component survives the prediction step. We assume an adaptive agent birth model $\gamma_k(\mathbf{z}_k)$ in which each observation generates a new Gaussian component with weight w_γ, mean $\mathbf{m}_\gamma$, and covariance $\mathbf{P}_\gamma$. We further assume that new agents cannot be spawned from existing ones and that there are no spontaneous births without associated measurement.

The *update* step can be formalized as

$$v_k(\mathbf{x}_k) = (1 - p_{d,k})\, v_{k|k-1}(\mathbf{x}_k) \tag{4.12}$$

$$+ \sum_{\mathbf{z}_k \in Z_k} \sum_{i=1}^{J_k} w_k^{(i)}(\mathbf{z}_k) \mathcal{N}(\mathbf{x}_k; \mathbf{m}_{k|k}^{(i)}(\mathbf{z}_k), \mathbf{P}_{k|k}^{(i)}) \tag{4.13}$$

with respective weight, mean, and covariance

$$w_k^{(i)}(\mathbf{z}_k) = \frac{p_{d,k} w_{k|k-1}^{(i)} q_k^{(i)}(\mathbf{z}_k)}{\kappa_k(\mathbf{z}_k) + \sum_{j=1}^{J_{k|k-1}} p_{d,k} w_{k|k-1}^{(j)} q_k^{(j)}(\mathbf{z}_k)} \tag{4.14}$$

$$\mathbf{m}_{k|k}^{(i)}(\mathbf{z}_k) = \mathbf{m}_{k|k-1}^{(i)} + \mathbf{K}_k^{(i)}(\mathbf{z}_k - \mathbf{H}_k \mathbf{m}_{k|k-1}^{(i)}) \tag{4.15}$$

$$\mathbf{P}_{k|k}^{(i)} = (\mathbf{I} - \mathbf{K}_k^{(i)} \mathbf{H}_k) \mathbf{P}_{k|k-1}^{(i)} \tag{4.16}$$

and

$$q_k^{(i)}(\mathbf{z}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{H}_k \mathbf{m}_{k|k-1}^{(i)}, \mathbf{H}_k \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^\top + \mathbf{R}_k) \tag{4.17}$$

$$\mathbf{K}_k^{(i)} = \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^\top + \mathbf{R}_k)^{-1} \tag{4.18}$$

where $\mathbf{H}_k$ is the measurement matrix and $\mathbf{R}_k$ is the measurement noise covariance. We let $p_{d,k}$ denote the probability that an agent is detected during the update step. We model false positive detections as clutter κ_k.

Since the number of Gaussian components increases at each filter iteration, the intensity quickly becomes computationally intractable. Therefore, we prune the components according to the following three conditions to guarantee fast tracking performance. Firstly, we discard components with a weight of less than the truncation threshold of T. Secondly, we merge components with Mahalanobis distance less than the merging threshold of U. Finally, we retain only the $J_{\max}$ components with the largest weights.

Implementation. We model the state of each agent as $\mathbf{x}_k = \left[\mathbf{p}_{x,k}, \mathbf{p}_{y,k}, \mathbf{v}_{x,k}, \mathbf{v}_{y,k}\right]^\top$ which consists of relative position $(\mathbf{p}_{x,k}, \mathbf{p}_{y,k})$ and velocity $(\mathbf{v}_{x,k}, \mathbf{v}_{y,k})$. The position and velocity components of the state are two-dimensional since the agents are assumed to fly in a planar configuration at approximately the same altitude. The control input $\mathbf{u}_k$ is defined as the linear velocity of the drone in the body frame which we obtain from the internal state estimate of the autopilot. Finally, the measurements $\mathbf{z}_k = [d_k, \beta_k]^\top$ consist of range and bearing, respectively.

The process and measurement noise covariances are modeled as

$$\mathbf{Q}_k = \sigma_v^2 \begin{bmatrix} \frac{\Delta_k^4}{4}\mathbf{I}_2 & \frac{\Delta_k^3}{2}\mathbf{I}_2 \\ \frac{\Delta_k^3}{2}\mathbf{I}_2 & \Delta_k^2\mathbf{I}_2 \end{bmatrix} \qquad \text{and} \qquad \mathbf{R}_k = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\beta^2 \end{bmatrix} \tag{4.19}$$

where Δ_k denotes the time elapsed since the last measurement and is computed as the difference between consecutive timestamps $\Delta_k = t_k - t_{k-1}$. We further let σ_v, σ_d and σ_β denote the standard deviation of the process, range, and bearing noise, respectively.

The state transition function follows a linear Gaussian model and is defined as

$$f(\mathbf{x}_{k-1}, \mathbf{u}_k) = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k \tag{4.20}$$

where

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{I}_2 & \Delta_k \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix} \qquad \text{and} \qquad \mathbf{B}_k = \Delta_k \mathbf{I}_2. \tag{4.21}$$

The observation model is nonlinear and consists of measurements of range and bearing

$$h(\mathbf{x}_k) = \begin{bmatrix} d_k \\ \beta_k \end{bmatrix} = \begin{bmatrix} \sqrt{\mathbf{p}_{x,k}^2 + \mathbf{p}_{y,k}^2} \\ \text{atan2}(\mathbf{p}_{y,k}, \mathbf{p}_{x,k}) \end{bmatrix} \tag{4.22}$$

where we use the two-argument function atan2 to avoid ambiguities in the conversion from cartesian to polar coordinates. We linearize the measurement model by computing the Jacobian with respect to the state as

$$\mathbf{H}_k = \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k} = \begin{bmatrix} \frac{\mathbf{p}_{x,k}}{\sqrt{\mathbf{p}_{x,k}^2 + \mathbf{p}_{y,k}^2}} & \frac{\mathbf{p}_{y,k}}{\sqrt{\mathbf{p}_{x,k}^2 + \mathbf{p}_{y,k}^2}} & \mathbf{0}_2 \\ -\frac{\mathbf{p}_{x,k}}{\mathbf{p}_{x,k}^2 + \mathbf{p}_{y,k}^2} & \frac{\mathbf{p}_{y,k}}{\mathbf{p}_{x,k}^2 + \mathbf{p}_{y,k}^2} & \end{bmatrix}. \tag{4.23}$$

We set the probability of detection to $p_d = 90\%$ to provide a slightly more conservative estimate of the detection performance during real-time inference than the results on our test dataset suggest (Sec. 4.2.1). We assume a probability of survival of $p_s = 100\%$ since agents that are detected once should not disappear. New Gaussian components are initialized directly from the measurements using an adaptive birth model with weight, mean, and covariance

$$w_\gamma = 10^{-5} \tag{4.24}$$

$$\mathbf{m}_\gamma = \left[d_k \cos(\beta_k), d_k \sin(\beta_k), 0, 0 \right]^\top \tag{4.25}$$

$$\mathbf{P}_\gamma = \text{diag}\left(\left[\sigma_{\mathbf{p}}^2, \sigma_{\mathbf{p}}^2, \sigma_{\mathbf{v}}^2, \sigma_{\mathbf{v}}^2 \right] \right) \tag{4.26}$$

where $\sigma_{\mathbf{p}}$ and $\sigma_{\mathbf{v}}$ are the standard deviation of the position and velocity which we set to 1 m and $10\,\mathrm{m\,s^{-1}}$, respectively. The mean is computed by converting the raw measurement from polar to cartesian coordinates, assuming zero initial velocity. We model false positive detections by assuming that we observe one clutter return per time step and therefore set $\kappa_k = 1/a^2$ where $a = 10\,\mathrm{m}$ is the side length of the virtual arena. Although the bearing measurements are precise to a single degree, we set its standard deviation to a slightly larger value of $\sigma_\beta = 3°$ to account for factors such as calibration inaccuracies or imprecisions in the detections due to background clutter. Since we experimentally determined that the range noise varies with the distance to the observer, we model its standard deviation as a function of the measured distance itself (Sec. 4.5.1). Finally, we model the truncation threshold as $T = 10^{-5}$, the merging threshold as $U = 0.5$, and the maximum number of components to $J_{\max} = 100$.

Flocking algorithm

The method described so far could be leveraged by any flocking algorithm. In this work, we use a control algorithm based on the Reynolds flocking rules [64] to compute high-level velocity commands from the relative position estimates of nearby drones [60]. The weighted velocity commands are: 1) a repulsive *separation* term to steer nearby drones away from each other, 2) a *cohesion* term to keep the drones close to each other, and 3) a *migration* term that provides a navigation goal to the swarm (Sec. 2.1).

During the experiments, we set the maximum speed to $v^{\max} = 0.5\,\mathrm{m\,s^{-1}}$, and the separation, cohesion, and migration gains to $k^{\mathrm{sep}} = 7$, $k^{\mathrm{coh}} = 1$, and $k^{\mathrm{mig}} = 1$, respectively. The gains are chosen such that the agents converge to an equilibrium distance of approximately 2 m during migration.

4.3 Experimental setup

4.3.1 Hardware

We use a custom-built quadcopter named LeQuad for all experiments (Fig. 4.5). Each quadcopter features four FLIR Firefly S global-shutter cameras mounted at a right angle from each other to obtain omnidirectional visual inputs. Each camera is equipped with an OpenMV ultra-wide angle lens which provides a horizontal and vertical field of view of 166° and 116°, respectively. We operate the cameras over a powered USB 2 hub at a binned resolution of 720×540 to obtain grayscale images at a frequency of 10 Hz. We refrain from using USB 3 to avoid electromagnetic interference with the Drotek F9P RTK-GNSS receiver, which provides centimeter-accurate absolute positions. We use the Nvidia Jetson TX2 mounted on a ConnectTech Orbitty carrier board as an onboard computer and the Holybro Pixhawk 4 as an autopilot.

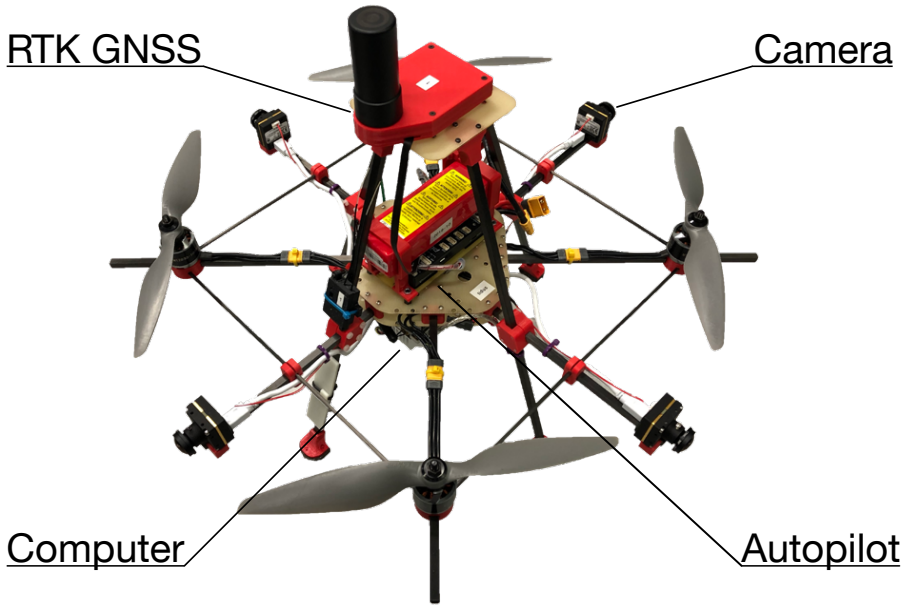


Figure 4.5 – Physical drone hardware for outdoor experiments. The LeQuad drone features omnidirectional vision via four cameras, a high performance onboard computer with embedded GPU for real-time inference, and a RTK-enabled GNSS to obtain centimeter-accurate ground-truth positions.

4.3.2 Software

The onboard computer runs Ubuntu 18.04 bundled with the Linux4Tegra (L4T) distribution, and we use ROS Melodic [108] as a robotics middleware. The autopilot runs PX4 [59] and is responsible for hardware-triggering the cameras to provide the resulting images with IMU-synchronized timestamps. The neural networks are trained and evaluated using PyTorch [85].

4.4 Simulation results

We evaluate the proposed approach using the Gazebo simulator [53]. We create a simple scene that resembles the outdoor environment in which we spawn the simulated quadcopters (Fig. 4.6). The drone detector is trained analogously to real-world experiments, albeit with simulated images and labels obtained using background subtraction (Sec. 4.2.1). Due to the simplicity of the scene and minimal background clutter, the detection model achieves an average precision (AP@0.5) of 100.0% at a confidence threshold of $p^{\text{conf}} = 0.001\%$ on the hold-out test set already after 12 epochs of training.

To further increase the realism, we additionally model misdetections in terms of false negatives and positives, as well as processing delays. In particular, we model false negatives as a Bernoulli random variable by discarding detections with a probability of 10%. We also model false

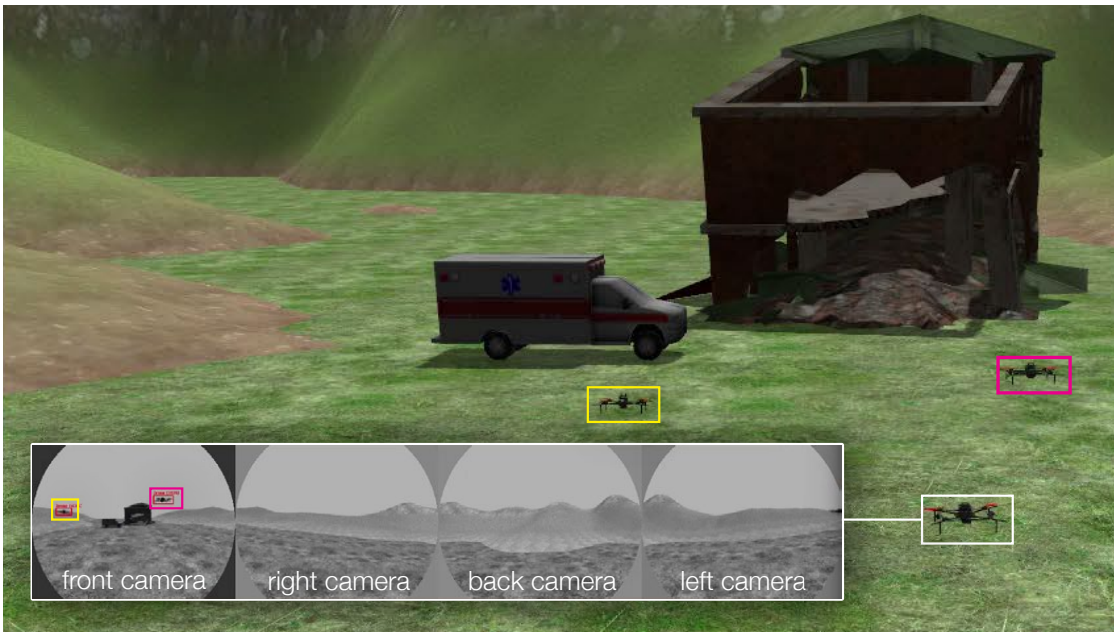
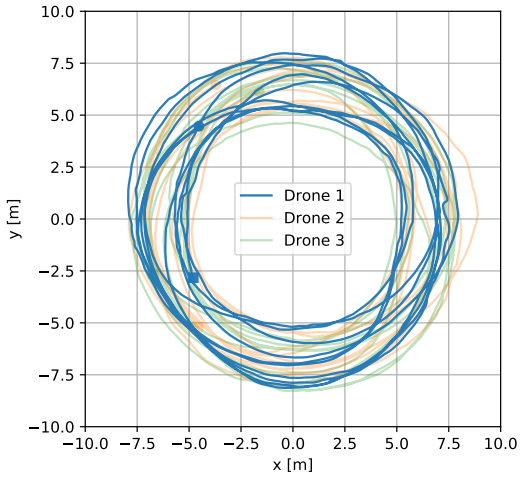


Figure 4.6 – Annotated screenshot of vision-based flocking experiments in a simulated outdoor environment. We simulate three quadcopters operating in a fictitious disaster scenario. We show the visual inputs from the four orthogonal cameras (white inset rectangle) of the focal agent (white rectangle). Here, the focal agent detects two other drones (yellow and pink rectangle) in its field of view.

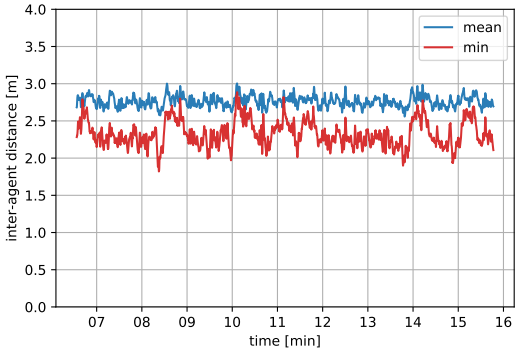
positives as a random variable that is Poisson-distributed in time, i.e., as clutter (Sec. 4.2.2), and uniformly distributed in space over the perception radius of an agent which we set to $r^{\max} = 5\,\text{m}$. We finally add a processing delay of 200 ms to account for the inference time of the drone detector (Sec. 4.2.1). We note here that it is difficult to realistically model misdetections in simulation since their distribution highly depends on environmental conditions such as visual clutter. We therfore provide real-world experiments in outdoor environments with substantial background clutter and difficult lighting conditions (Sec. 4.5.2)

4.5 Real-world results

We report the visual relative localization errors of drones in a controlled indoor environment with access to millimeter-accurate position information (Sec. 4.5.1). We show vision-based flights in outdoor environments with three real quadcopters performing several navigation tasks (Sec. 4.5.2) and free flocking, i.e., self-organized flocking without a navigation goal (Sec. 4.5.3). We finally show examples of predicted detection bounding boxes to provide a qualitative overview of success and failure cases (Sec. 4.5.4).



(a) Paths



(b) Distances

Figure 4.7 – Vision-based flocking in simulated environments with circular migration. We show (a) the paths and (b) inter-agent distances of the drones over the course of the migration experiment.

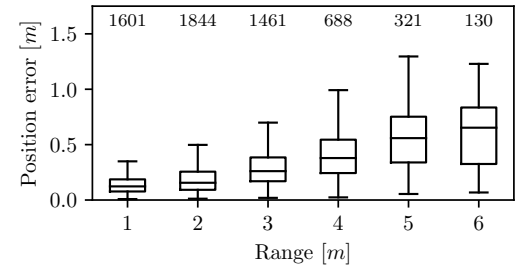
4.5.1 Visual relative localization

We show results on the theoretical performance of the visual relative localization system to test its operational bounds and to find suitable values for the range and bearing noise parameters σ_d and σ_β (Eq. 4.19). To this end, we employ a setup similar to the one used for automatic labeling (Sec. 4.2.1) except that we additionally obtain ground truth poses of the observing camera and drone from a motion capture system. After transforming the visual detections into the frame of the motion capture system, we can directly compare the drone's true position with its estimate obtained using vision in metric space. We find that the relative localization error varies considerably as a function of the distance to the drone, whereas the error caused by bearing variations is negligible (Fig. 4.2.2).

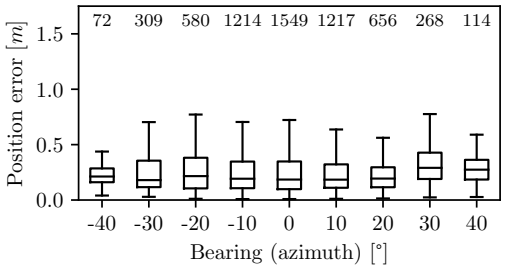
4.5.2 Collective outdoor navigation

We report results for three different navigation scenarios: *linear*, *rectangular*, and *circular* migration. Before each flight, we place the drones at roughly 2.5 m distance from each other and wait for their RTK-GNSS receivers to converge to a fixed solution which provides centimeter-accurate absolute positions at 10 Hz. These measurements are only used to provide a reliable ground-truth for the evaluation of the experiments.

After the RTK fix is obtained, we let all agents take off simultaneously and reach a height of 2 m above the ground before we let the vision-based flocking algorithm take over the control of their motion. The agents are given the same list of migration points depending on the type of navigation scenario. We switch from one migration point to the next as soon as an agent

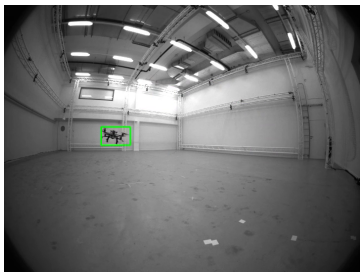


(a) Range error



(b) Bearing error

(c) Distance: 1 m



(d) Distance: 3 m

(e) Distance: 5 m

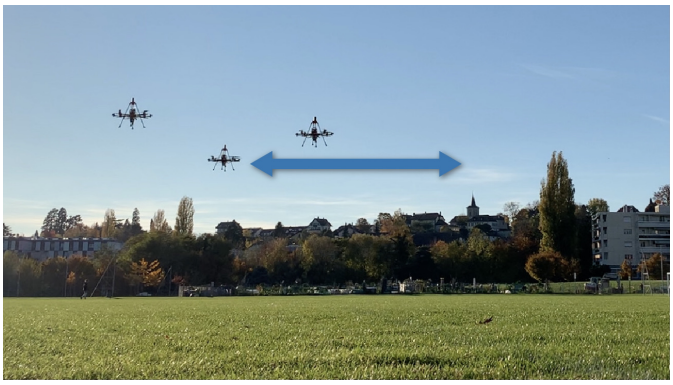
Figure 4.8 – Comparison of vision-based relative localization errors as a function of (a) *range* and (b) *bearing*. We show example images of the experimental setup at (c) 1 m, (d) 3 m, and (e) 5 m relative distance to the observer. The bearing estimates are near-constant over the field of view, whereas the range errors increase with distance from the observer. Millimeter-accurate ground-truth positions are obtained at 100 Hz using a motion capture system. The counts above the boxes indicate the number of measurements used to calculate their statitics.

enters an acceptance radius of $r^{\text{acc}} = 3\,\text{m}$. As the list of migration points is exhausted, we repeat the procedure from the first waypoint. We stop the experiment as soon as the battery level of one of the agents reaches a critical capacity of 15%.

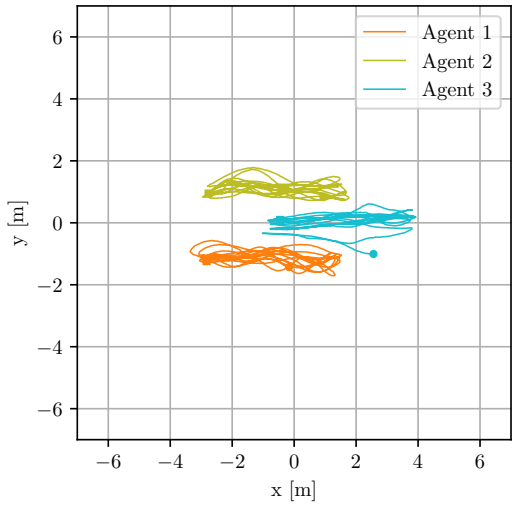
The height of the drones is individually regulated using a proportional controller to constrain their motion to a horizontal plane. However, the planar constraint may be lifted by mounting additional cameras that point to the top and bottom, or by equipping the existing cameras with lenses that provide a larger field of view (Chapter 3).

During the linear migration experiment, the agents fly between two waypoints that are located 10 m apart from each other (Fig. 4.9b). Over a total flight duration of around 2.5 min, the minimum inter-agent distance the agents reach is 1.82 m and the overall mean 2.42 m (Fig. 4.9c).

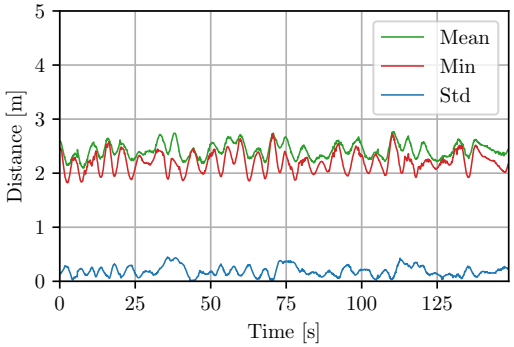
The rectangular migration experiment defines four waypoints that are located at the corners of a square with side length 10 m (Fig. 4.10b). The total flight time is around 3.3 min and the overall minimum and mean inter-agent distances are 1.45 m and 2.36 m, respectively (Fig. 4.10c).



(a) Photo



(b) Paths

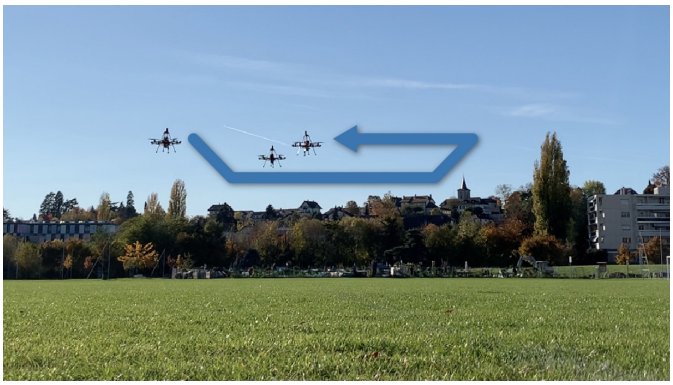


(c) Distances

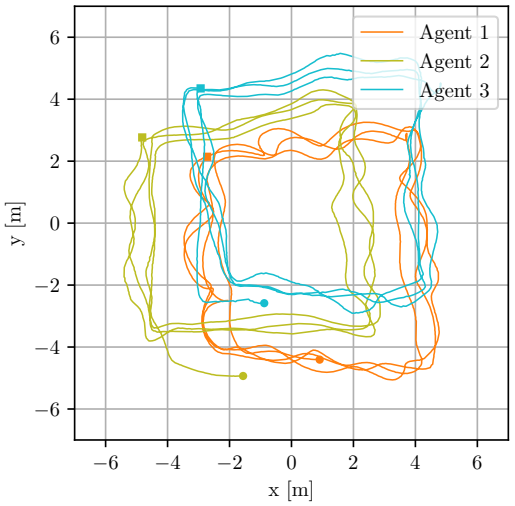
Figure 4.9 – Vision-based linear migration in outdoor environments. We show (a) an annotated photo, (b) the paths, and (c) inter-agent distances of the drones over the course of the migration experiment. The drones remain collision-free and cohesive using only local visual information, which is processed onboard in real-time. Centimeter-accurate ground-truth positions are obtained at 10 Hz using RTK-enabled GNSS receivers mounted on the drones. These positions are not shared between the drones during the experiments and only serve to analyze the inter-agent distances.

Finally, the circular migration experiment leads the agents through a series of twelve waypoints that are linearly spaced around a circle with 10 m diameter (Fig. 4.11b). During an overall flight time of around 5 min, the agents remain collision-free while reaching a minimum inter-agent distance of 1.37 m and an overall mean distance of 2.32 m (Fig. 4.11c).

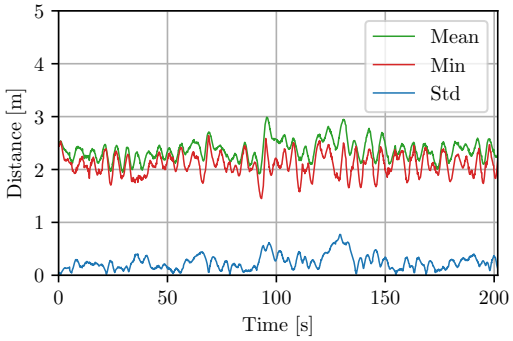
The above experiments are three representative flights taken from a total of 30 min of collision-free experimental recordings. The progressively lower inter-agent distances across linear, rectangular, and circular migration experiments can be explained by examining the distribution and/or density of waypoints. During the rectangular migration experiment, the lowest



(a) Photo



(b) Paths



(c) Distances

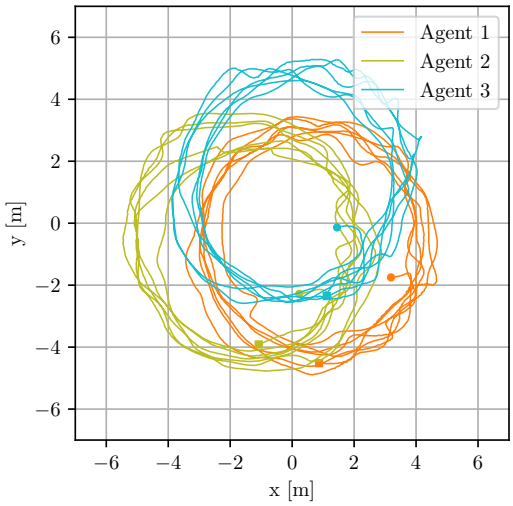
Figure 4.10 – Vision-based rectangular migration in outdoor environments. We show (a) an annotated photo, (b) the paths, and (c) inter-agent distances of the drones over the course of the migration experiment.

inter-agent distances are reached close to the corners where directional changes occur. In the case of the circular migration experiment, the larger number and density of waypoints have a cohesive effect since the agents simultaneously approach points that are more densely spaced.

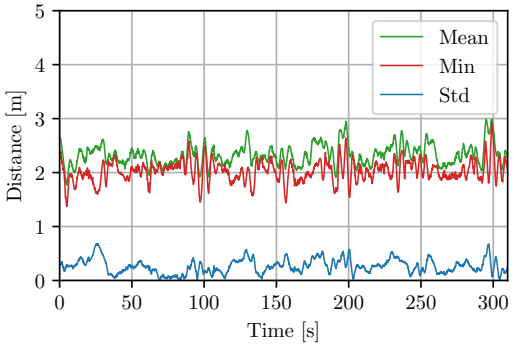
4.5.3 Outdoor free flocking

We perform free flocking experiments analogously to the collective navigation experiments (Sec. 4.5.2) except that we do not add a migration term to the flocking algorithm (Sec. 4.2.2). Hence, we let the drones self-organize their collective motion entirely based on agent-to-agent interactions. The absence of waypoints leads to some interesting behaviors such as spontaneous migration (Fig. 4.12) and reconfiguration (Fig. 4.13).

(a) Photo



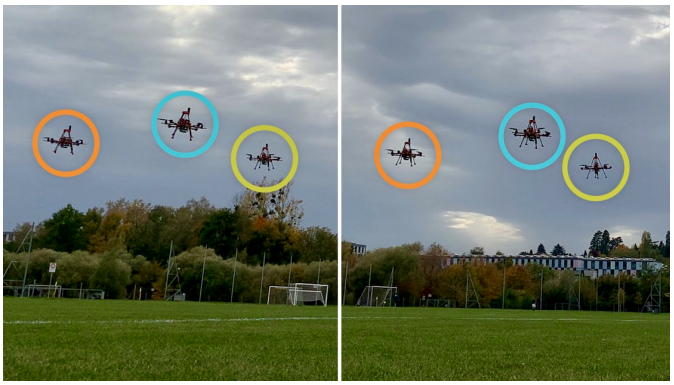
(b) Paths



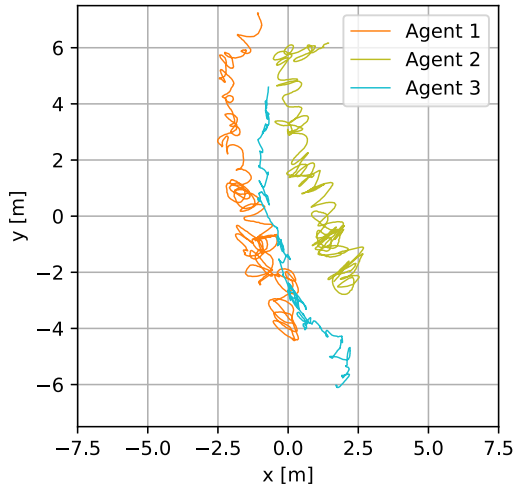
(c) Distances

Figure 4.11 – Vision-based circular migration in outdoor environments. We show (a) an annotated photo, (b) the paths, and (c) inter-agent distances of the drones over the course of the migration experiment.

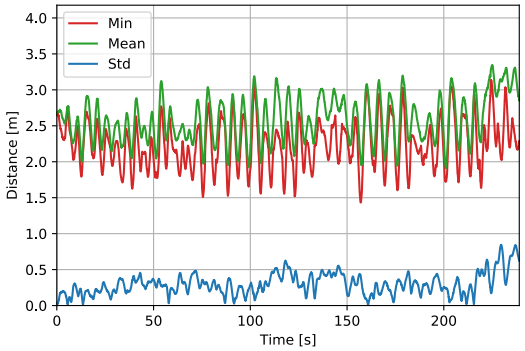
During the spontaneous migration behavior, the first agent (orange line; Fig. 4.12b) mistakenly detects a drone in the background clutter of the far side of the football field. The majority of false positives occur due to the line-like features of the poles in front of the bushes, which are falsely detected as drones. The bounding boxes are relatively small, suggesting the detected neighbor is far away, thus causing the drone to decrease its relative distance. The other drones start to follow the — in this case misinformed — drone to keep their inter-agent distances at equilibrium. On the one hand, this behavior highlights a failure case of the modular flocking system since the detector and tracker cannot reliably reject false positives. On the other hand, the result shows that a single agent, representing only a small proportion of the group, can steer the other drones towards a goal. This behavior could be exploited for situations in which the drones need to navigate based on local information (e.g., by detecting an object of interest in their environment) which may not be visible from the perspective of



(a) Photo (left: $t = 100\,\mathrm{s}$, right: $t = 0\,\mathrm{s}$)



(b) Paths

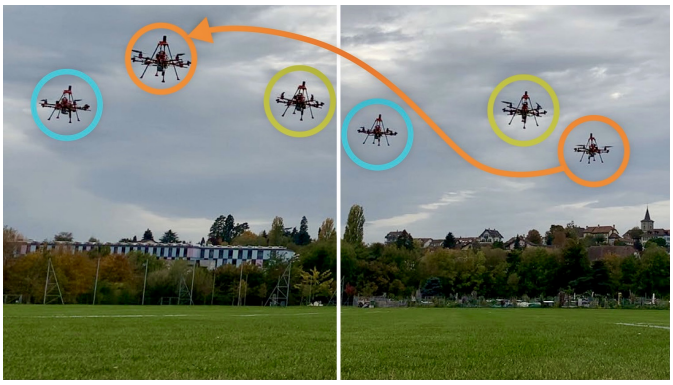


(c) Distances

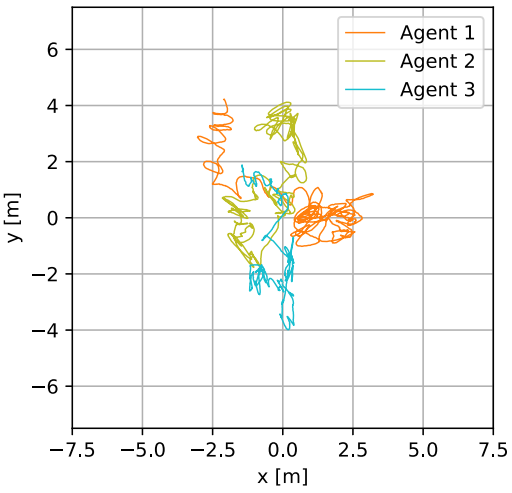
Figure 4.12 – Vision-based spontaneous migration in outdoor environments. We show (a) an annotated photo, (b) the paths, and (c) inter-agent distances of the drones over the course of the free flocking experiment.

every agent within the swarm (e.g., due to visual occlusions of other group members or the environment). The group would therefore have to rely on an informed subset of agents to steer them. The trajectories and inter-agent distances suffer from noticeably higher levels of oscillations around the equilibrium distance (Fig. 4.12b and 4.12c) However, the agents remain collision-free throughout the experiment, while reaching a minimum inter-agent distance of 1.41 m and an overall mean distance of 2.35 m (Fig. 4.12c).

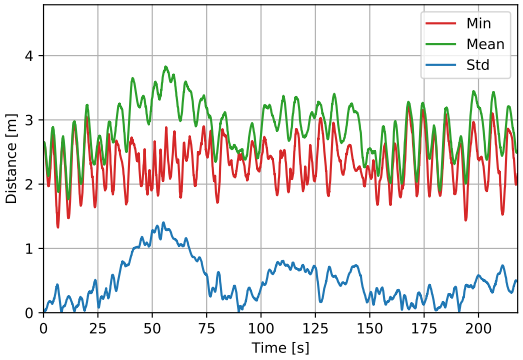
During the spontaneous reconfiguration behavior, the first agent (orange line; Fig. 4.13) squeezes through a gap in between the other two drones. The three drones temporarily form a linear formation at around $t = 55\,\mathrm{s}$ after which they return to their regular triangular equilibrium. The maneuver is caused by a combination of false negative and false positive detections on all drones and occurs roughly in between $t = 25\,\mathrm{s}$ and $t = 75\,\mathrm{s}$. Some of the false



(a) Photo (left: $t = 100\,\mathrm{s}$, right: $t = 0\,\mathrm{s}$)



(b) Paths



(c) Distances

Figure 4.13 – Vision-based spontaneous reconfiguration in outdoor environments. We show (a) an annotated photo, (b) the paths, and (c) inter-agent distances of the drones over the course of the free flocking experiment.

positives are valid detections caused by a spare drone that was not appropriately covered at the beginning of the experiment. Although the drones reach the smallest minimum inter-agent distances of 1.29 m, the group can remain collision-free and cohesive (Fig. 4.13c). The overall mean distance of 2.64 m is larger than during the previous experiments due to the reconfiguration maneuver (Fig. 4.13c).

4.5.4 Qualitative detection results

During the experiments, a variety of objects that can potentially confuse the detector are present in the background (Fig. 4.14). In descending order of frequency, the most common objects aside from drones are trees, buildings, cars, people, fences, traffic signs, tables, and

even dogs. The other drones are generally detected despite background clutter and adverse lighting conditions (Fig. 4.14a and 4.14b). False-negative detections occur most frequently when another drone is flying far away and/or in front of the ground control station, i.e. a camping table with the experimental equipment covered by a sun umbrella (Fig. 4.14c). False-positive detections appear most often in image regions that contain many line-like features since they resemble the mechanical design of the drone (Fig. 4.14d). They are mostly present for the duration of a single frame and the tracker can reliably reject them. False tracks are occasionally created if false-positive detections are present for more than one frame. However, the false tracks do not cause instabilities that lead to collisions.

4.6 Conclusions

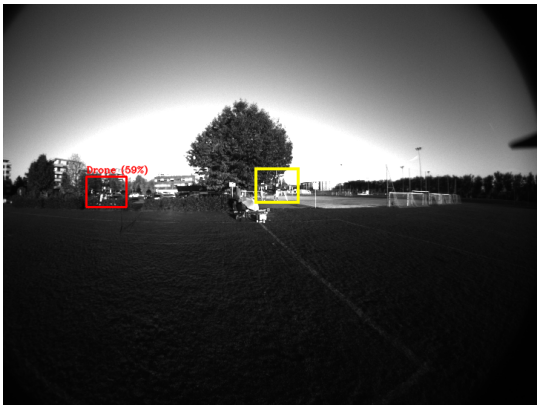
We presented a vision-based detection and tracking algorithm that enables dense groups of drones to fly cohesively and without mutual collisions. The proposed approach does not depend on visual markers or inter-agent communication and is thus suitable for flocking operation in GNSS-denied environments or in situations where wireless links are unreliable. The approach is fully decentralized since each agent relies exclusively on onboard processing of local visual information to estimate the positions and velocities of neighboring drones. The outdoor navigation experiments show that the system is robust to background clutter and enables collision-free flight even in demanding lighting conditions.

The approach to multi-agent tracking used in this chapter (Sec. 4.2.2) is well-suited for reducing the noise from the visual relative localization module (Sec. 4.2.2) but is rather ineffective against misdetections. In general, the GM-PHD filter is more effective at rejecting false positives (Fig. 4.14d) than it is at filling in missing detections due to false negatives (Fig. 4.14c). During the experiments, false negative detections would cause the filter to lose track after a single time step, whereas new tracks would be created after two consecutive false positive detections if they occur in close proximity to each other. The main issue here is that misdetections — whether false positives or false negatives — tend to occur in bursts that last several time steps, which makes them difficult to properly reject. In particular, false positive detections are usually not uniformly distributed in space but occur at the same location because the detector is identifying background clutter as a drone. We generally recommend lower detector confidence thresholds that favor occasional false positives over false negatives. This approach is also safer since it more likely to avoid collisions at the expense of jitter in the trajectories.

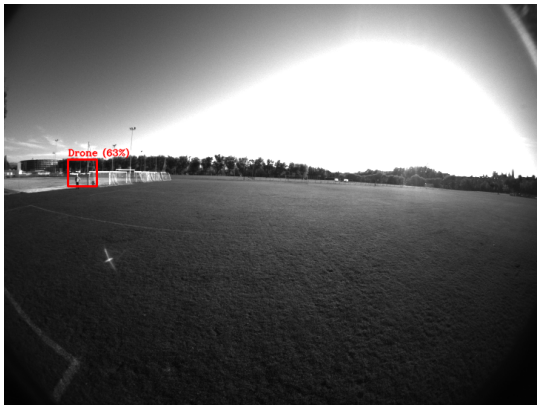
The experiments presented here should be considered as minimal validation conditions of vision-based flocking algorithms without explicit communication or localization infrastructure. Further experiments are needed to ensure the scalability of the proposed system to larger numbers of vision-based agents. However, large-scale real-world experiments with custom quadcopter platforms (Sec. 4.5) are both prohibitively expensive and time-consuming. Moreover they require additional logistic infrastructure and human resources to manage. We will therefore limit the study of scalability to simulation for the remainder of the thesis.

(a) High confidence

(b) Low confidence



(c) False negative



(d) False positive

Figure 4.14 – Qualitative examples of detection results categorized by confidence score (high and low) and type of error (false negative and false positive). (a) High confidence: the drone is easily detected in front of the grass texture despite direct sunlight. (b) Low confidence: the drone blends in with the background due to the line features and lighting conditions. (c) False negative: the right drone (yellow, manually labeled) can not be reliably distinguished from the background clutter. (d) False positive: this type of spurious detection occurs relatively frequently but is filtered out by the multi-agent state tracker.

5 Scalable vision-based flocking in the presence of occlusions

This chapter addresses the scalability of vision-based drone swarms in terms of group size and density. Vision-based swarms rely on detecting neighbors but usually neglect perceptual factors such as mutual visual occlusions because they operate in small groups. To study the impact of occlusions on the scalability of the swarm, we propose a simple but perceptually realistic visual neighbor selection model that discards obstructed agents. We evaluate the visibility model with up to one thousand point mass agents, showing that occlusions have adverse effects on the inter-agent distances and velocity alignment as the swarm scales up, both in terms of group size and density. In particular, we find that small agent displacements have considerable effects on neighbor visibility and lead to control discontinuities. We show that the destabilizing effects of visibility switches, i.e., agents continuously becoming visible or invisible, can be mitigated if agents select their neighbors from adjacent Voronoi regions. We validate the resulting flocking algorithm using up to one hundred agents with quadcopter dynamics and subject to sensor noise in a high-fidelity physics simulator. The results show that Voronoi-based interactions enable vision-based swarms to remain collision-free, ordered, and cohesive in the presence of occlusions. These results are consistent across group sizes and agent densities.

The work presented in this chapter is adapted from [62][1]:

- F. Schilling, E. Soria, and D. Floreano, "On the scalability of vision-based drone swarms in the presence of occlusions," in *IEEE Access*, vol. 1, no. 1, pp. 1–13, (**submitted**) Aug. 2021.

5.1 Introduction

Aerial robot swarms have a vast socio-economic potential and are used for numerous real-world applications in industries such as agriculture, mapping, and construction [1, 3, 7]. Drone swarms can be deployed to monitor crops, create maps, and survey sites much faster than a single drone since they can solve tasks cooperatively and in parallel. Larger group sizes

[1] Video: https://youtu.be/2MZ-TN7MnYw.

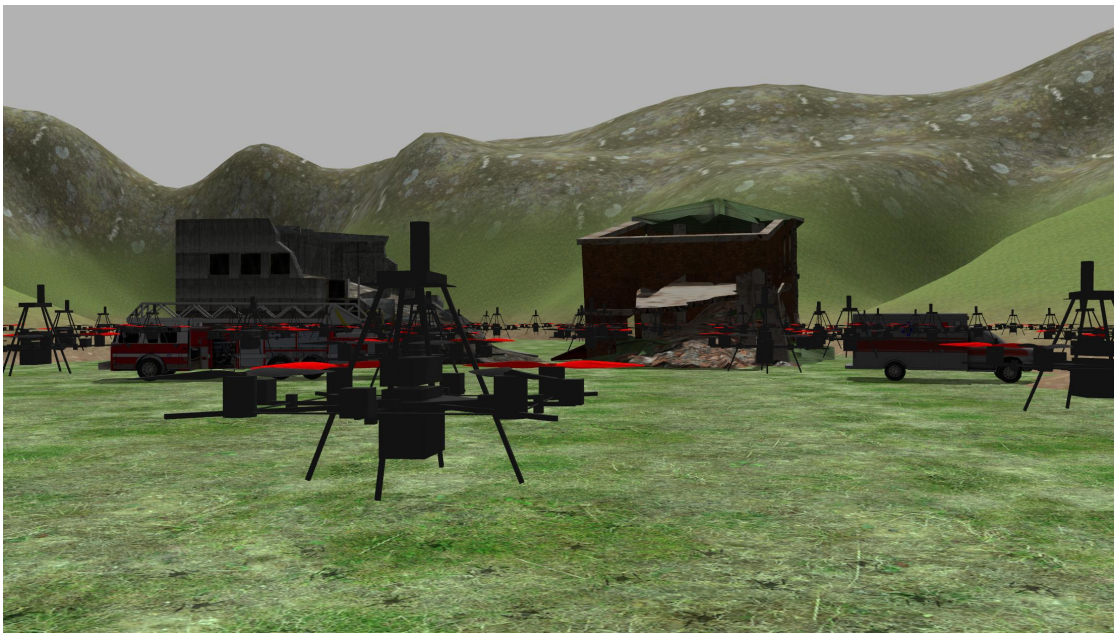


Figure 5.1 – Screenshot of a collective search and rescue mission with a swarm of one hundred quadcopters in the Gazebo simulator. During the mission, the quadcopters take off from the ground and navigate towards a fictitious disaster scenario.

can further decrease task completion times and operating swarms in compact formations can enable new applications in confined spaces such as buildings. However, most drone swarms deployed today rely on external localization and wireless communication, both of which represent major limiting factors towards their scalability in terms of group size and swarm density.

Localization in drone swarms is usually achieved with satellite-based systems for outdoor applications or optical motion capture for indoor deployments. The drones are typically equipped with wireless communication devices that enable the exchange of state information such as positions and velocities with each other [17, 11]. While this approach has enabled successful deployments of impressive aerial swarms, it comes with several key limitations. Firstly, wireless communication suffers from inherent scalability issues since the bandwidth requirement scales quadratically with the number of agents [18]. In practice, this leads to compounding delays whose durations are difficult to estimate and thus require dampening and interpolation [13, 14]. Secondly, the approach lacks flexibility since the agents must adhere to the same communication protocol and need to be localized in the same frame of reference. Thirdly, the exclusive use of an external positioning system represents a single point of failure and its malfunctioning can have disastrous effects.

Vision-based relative localization methods rely entirely on local information to detect other agents, thus removing the dependence on external localization systems and additional communication infrastructure. Moreover, vision is arguably the ideal sensory modality for lo-

calization on aerial robots since cameras are small, lightweight, and provide extremely high information density at comparatively low power consumption [25]. Multi-robot systems that use a vision-based approach to mutual localization have recently emerged in the form of leader-follower formations [93, 60, 91, 109] and the first aerial flocks [78, 110, 61]. Important perceptual factors such as visual occlusions, i.e., agents that are obstructed by others, are usually neglected in these swarms because of their small group size. However, these factors become a deterrent for larger swarms, especially when they have to fly in dense configurations (Fig. 5.1).

While some swarm roboticists explicitly make use of visual occlusions to solve collaborative transport problems [111] and robotic shepherding tasks [112], the most thorough treatment of visibility constraints can be found in the collective motion literature. Using computer vision techniques, researchers are able to reconstruct the poses and visual fields of individual animals and show that visual perception best explains how information about food sources and predators transfers within the group [20, 22, 23, 24]. How individuals select and react to their neighbors is one of the fundamental questions in the study of collective motion and agent-based flocking models provide an indispensable tool to test and verify different hypotheses [113, 114, 115, 116]. Notable examples of neighbor selection methods include *metric* (i.e., within a metric radius) [71], *topological* (i.e., the set of n nearest neighbors) [117], or *voronoi*-based (i.e., from adjacent Voronoi regions) [118] interactions. Recently, different forms of *visual* neighbor selection have gained popularity due to their biological plausiblity [20, 22, 23, 24]. For example, research on flocking models with a limited field of view shows that lateral vision is crucial for collision-free collective motion [66, 67] and may explain why flocking birds have almost omnidirectional vision [68]. Simulations of large schools of fish show that visual obstructions lead to more realistic group shapes and densities than purely metric interactions [65]. Simulations of large vision-based flocks show that bird density can be regulated effectively if individuals only react to the projection of their neighbors [28]. Other researchers show that many natural behaviors such as milling and polarized flocking emerge from purely visual interactions even in the absence of a spatial representation of neighbors [52]. Although these models offer interesting collective behaviors, they often make modeling choices that are geared towards a particular species or result in undesirable behavior for robotic swarms since they lead to frequent collisions.

In this chapter, we tackle visibility constraints arising from occlusions from a robotics perspective with the goal of synthesizing large and compact vision-based drone swarms. In particular, we study the effect of occlusions on the performance (i.e., collision avoidance, cohesion, and velocity alignment) of vision-based swarms as they scale from low densities and a handful of agents to high-density swarms with thousands of individuals. To this end, we propose a *visual* neighbor selection model that offers a perceptually plausible alternative to the ubiquitous but unrealistic *metric* selection of neighbors, i.e., methods that assume agents can sense all neighbors within a given radius. We simulate vision-based swarms of up to one thousand point mass agents and program them to perform collective waypoint navigation using a simple attractive/repulsive flocking algorithm. The results show that swarms in which agents react to

all *visible* neighbors perform poorly, especially at high densities and as the group size increases beyond tens of agents. However, by limiting visual interactions to their Voronoi neighbors, we can successfully synthesize collision-free, cohesive, and ordered vision-based swarms. A comparison of Voronoi interactions with other common neighbor selection methods (i.e., metric and topological) reveals their superiority in large, high-density swarms. We validate the scalability of the resulting flocking algorithm at different densities and group sizes with quadcopter dynamics using a simulator with realistic physics and noise levels. The analysis shows that visually-constrained Voronoi interactions are both perceptually plausible and highly effective for the coordination of large aerial robot swarms in which agents rely purely on local visual information for control.

5.2 Method

We aim to synthesize a vision-based swarm that remains as compact as possible and collision-free while performing collective waypoint navigation. We define this objective since it enables many practical applications such as cooperative mapping, aerial deliveries, and search & rescue. In the following, we restrict ourselves to swarms that operate in two-dimensional planar configuations.

We first describe a simple attractive/repulsive flocking algorithm that provides collision avoidance and cohesion, as well as a navigation capability to the swarm (Sec. 5.2). To obtain a flocking algorithm that is plausible for vision-based swarms, we define the notion of agent visibility in the form of a neighbor selection strategy that is based on a realistic occlusion model (Sec. 5.2.1). Since vision-based detection is an inherently stochastic process, we further model sensing noise on the range and bearing measurements (Sec. 5.2.2).

The motion of each agent can be described by single-integrator dynamics of the form

$$\mathbf{p}_i^{k+1} = \mathbf{p}_i^k + \mathbf{v}_i^k \Delta t \tag{5.1}$$

where k denotes the index of the discrete time step with duration Δt.

In the remainder of the section, we skip the dependence on the discrete time step k for notational brevity and clarity. However, all computations in this section are performed at every time step without exception.

Flocking algorithm

The objective of the swarm is to perform waypoint navigation while avoiding inter-agent collisions and staying together as a group (Sec. 2.2). We formulate this objective as an artificial potential field that is inspired by the Reynolds flocking algorithm [64]. The motion of an agent is composed of an attractive/repulsive potential that provides separation and cohesion between agents (Sec. 5.2), as well as a migratory potential responsible for goal-directed

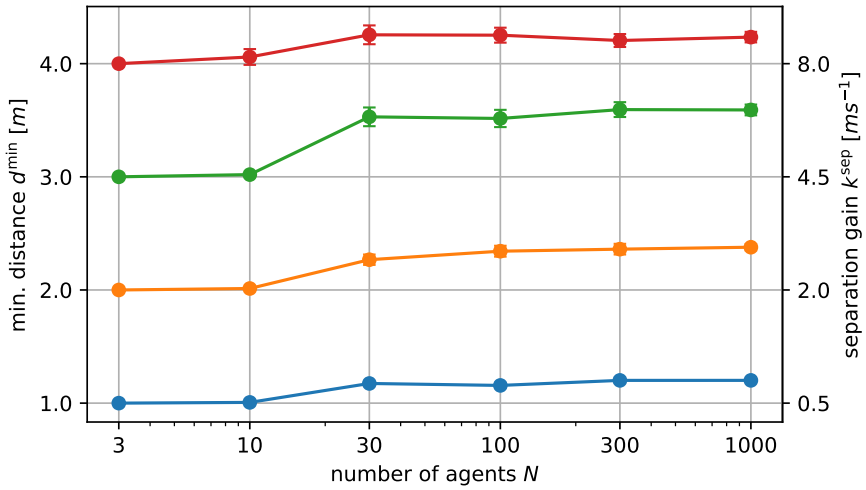


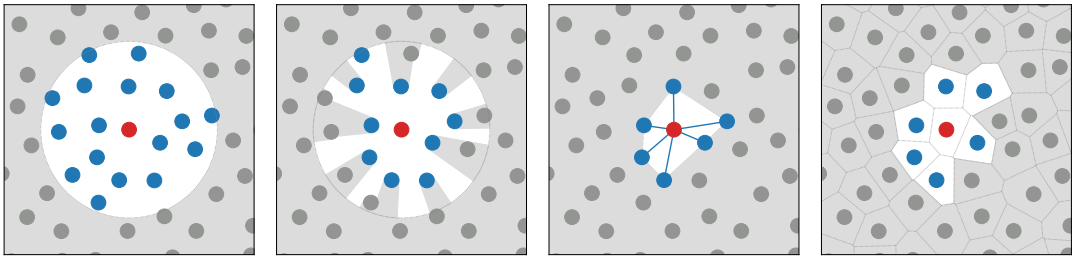
Figure 5.2 – Scalability of minimum nearest neighbor distances to increasing numbers of agents using the baseline *metric* neighbor selection model, i.e. agents within the perception radius are detected irrespective of whether they are occluded. Each line represents the minimum equilibrium distance between nearest neighbors obtained from different separation gains as the swarm size increases (mean and std. dev. over ten trials, all other parameters constant). Aside from a noticeable increase of inter-agent distances between ten and thirty agents that occurs due to the saturation of the perception range with agents, the inter-agent distances remain constant across different group sizes (note the logarithmic scale).

navigation (Sec. 5.2).

Separation and cohesion

Cohesion and collision avoidance can be achieved with an attractive/repulsive potential that keeps the agents at an equilibrium distance (Sec. 2.2). The cohesion term keeps the swarm together by attracting agents to the average position of their neighbors. The separation term leads to collision avoidance by repulsing nearby agents from each other.

Note that we do not scale the separation velocity command by the number of agents. This formulation has the advantage that minimum inter-agent distances remain quasi-constant as the group size increases and thus reduces the need for readjusting the control gains (Fig. 5.2). We further use the analytical solution to the above equations for three agents as a first approximation of the desired inter-agent distance d^{ref}. This allows us to express an approximate reference distance by using a separation gain of the form $k^{\text{sep}} = (d^{\text{ref}})^2/2\,\text{m}\,\text{s}^{-1}$ and keeping the cohesion gain fixed at $k^{\text{coh}} = 1\,\text{m}\,\text{s}^{-1}$. Note that in general, the separation gain slightly overestimates the reference distance for larger swarms since it does not take the number of neighbors into account. It is nevertheless a useful approximation that spares us the tedious task of finding the reference distance empirically for each agent swarm scale separately.



(a) Metric (b) Visual (c) Topological (d) Voronoi

Figure 5.3 – Schematic visualization of different neighbor selection strategies: (a) metric, (b) visual, (c) topological, and (d) Voronoi-based. We take the perspective of a focal agent within a swarm (central red disk) that selects agents (blue disks) and discard others (gray disks) depending on the following selection criteria: (a) *metric* selects all agents within a metric perception radius, (b) *visual* selects all visible agents within a metric radius, i.e., all agents that appear large enough and are not occluded by others, assuming agents are equally sized and have an omnidirectional camera at their center, (c) *topological* selects only the n closest agents (here $n = 6$), irrespective of their distance, and (d) *voronoi* selects only those agents that belong to a neighboring Voronoi region.

Migration

The purpose of the migration term is to give the agents a navigation goal by steering them towards a waypoint (Sec. 2.2).

5.2.1 Neighbor selection

Neighbor selection is an important consideration for all flocking algorithms since it introduces the notion of locality (e.g., in communication, perception, etc.) as opposed to all-to-all information transfer. In the following, we denote the neighbors of agent i as a set $\mathcal{N}_i$ where $\mathcal{N}_i \subseteq \mathcal{A}_i$.

Note that the adjacency matrix is not necessarily symmetric and the resulting graph may be directed. The metric and voronoi neighbor selection mechanism we define in the following sections (Sec. 5.2.1 and 5.2.1) result in an undirected graph and a symmetric adjacency matrix, whereas visual and topological neighbor selection (Sec. 5.2.1 and 5.2.1) are generally asymmetric and directed. In other words, the visibility between a pair of agents $i \sim j$ does not imply that the inverse relationship $j \sim i$ is true.

Metric: distance-based neighbor selection

Metric neighbor selection keeps only those agents that fall within a radius $r^{\max}$ centered around the focal agent (Fig. 5.3a). We can formalize metric neighbor selection as the set

$$\mathcal{N}_i^{\text{metric}} = \{j \in \mathcal{A}_i \mid d_{ij} < r^{\max}\} \tag{5.2}$$

where $r^{\max}$ denotes the maximum perception range.

Defining the set of neighbors based on a metric range is the most popular means of neighbor selection in the literature [64, 119, 71, 47]. Metric neighbor selection is a simple and effective method to introduce locality in the interactions and can be interpreted as a perception radius for vision-based swarms or a communication range for swarms that can exchange information via wireless links, for example. With the assumption that all agents are homogeneous and equally sized, we can use the metric perception range to represent visual acuity, i.e., the minimum size that another agent spans on the retina of the focal agent before it can no longer be perceived. We therefore encourage the reader to think about the perception range as the equivalent of the minimum subtended angle that another agent spans on the retina of the focal agent.

Visual: occlusion-based neighbor selection

Visual neighbor selection keeps only those agents that appear large enough and are not occluded by closer ones as seen from the perspective of the focal agent (Fig. 5.3b). The set of visible agents can be formalized as

$$\mathcal{N}_i^{\text{visual}} = \left\{j \neq k \in \mathcal{N}_i^{\text{metric}} \mid \neg\left(\|\mathbf{u}_{ij} - \mathbf{u}_{ik}\| < \hat{r}_{ij} + \hat{r}_{ik} \wedge d_{ij} < d_{ik}\right)\right\} \tag{5.3}$$

where $\mathbf{u}_{ij} = \mathbf{r}_{ij}/d_{ij}$ and $\hat{r}_{ij} = r/d_{ij}$ are the projections of the agent position and radius onto the unit circle, respectively.

In other words, we represent each agent as an opaque disk with a given radius centered around its position. To compute if an agent occludes another one, we project their relative positions and radii onto the unit circle of the observing agent and check whether their projected radii intersect each other. We consider the first agent occluded if there is an intersection and its relative distance is smaller.

Note that by combining metric and visual neighbor selection, we obtain a model of visibility that takes into account both visual acuity and occlusions. We consider this model plausible for vision-based swarms since it captures the information that is de facto available to an individual that operates purely on visual perception.

The above definition of visibility contains two key assumptions. The first assumption is that agents can distinguish individuals from each other. Note that this assumption does not require

identities to be maintained over time. The second assumption is that partially occluded agents are considered invisible, i.e., only the closest set of agents with an uninterrupted line of sight are contained in the visible set. This assumption is reasonable for monocular vision since the relative distance to other agents can only be reliably estimated if all of their spatial extent is visible.

Topological: n-nearest neighbor selection

Topological neighbor selection keeps only the n nearest neighbors of the focal agent (Fig. 5.3c). We can write the set of nearest neighbors as

$$\mathcal{N}_i^{\text{topo}} = \left\{ n\text{-}\underset{j \in \mathcal{A}_i}{\arg\min}\, d_{ij} \right\} \tag{5.4}$$

where the n-argmin operator selects at most the n nearest neighbors.

Topological neighbor selection is a popular method due to its explanatory success in natural swarms [20, 120] and is often used in models of collective motion to maintain group cohesion [47, 117].

Voronoi: spatially balanced nearest neighbor selection

Voronoi neighbor selection keeps only those agents whose Voronoi regions share a border with the focal agent (Fig. 5.3d). We can write the set of Voronoi neighbors as

$$\mathcal{N}_i^{\text{voronoi}} = \left\{ j \in \mathcal{A}_i \mid V_i \cap V_j \neq \emptyset \right\} \tag{5.5}$$

where $\emptyset$ denotes the empty set and V_i the Voronoi region of agent i which can be defined as

$$V_i = \left\{ j \in \mathcal{A}_i, \mathbf{q} \in \mathbb{R}^m \mid \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\| \right\}. \tag{5.6}$$

In other words, the Voronoi region of an agent can be described as the set of all points that are closer to itself than to any other agent.

Neighbor selection based on the Voronoi tessellation can be seen as topological interactions that are parameter-free and automatically balanced in space [117]. Moreover, it can be shown that the average number of Voronoi neighbors is at most six for the planar case we are considering here [121].

5.2.2 Sensing noise

We model the visual relative localization inaccuracies in two independent components: range and bearing. We model range noise as a function that varies linearly with relative distance

from the observer whereas the bearing noise is constant over the field of view [89, 122, 91, 61]. More formally, we define the noisy version of range and bearing with which agent i detects agent j as

$$\hat{d}_{ij} = d_{ij}(1+\omega_d), \qquad \omega_d \sim \mathcal{N}(0,\sigma_d) \tag{5.7}$$

$$\hat{\beta}_{ij} = \beta_{ij} + \omega_\beta, \qquad \omega_\beta \sim \mathcal{N}(0,\sigma_\beta) \tag{5.8}$$

where ω_d and ω_β are independent and identically distributed white noise with zero mean and standard deviation of σ_d and σ_β, respectively. The noisy relative position can then be constructed from polar coordinates as

$$\hat{\mathbf{r}}_{ij} = \begin{bmatrix} \hat{d}_{ij}\cos(\hat{\beta}_{ij}) \\ \hat{d}_{ij}\sin(\hat{\beta}_{ij}) \end{bmatrix} \tag{5.9}$$

where $\hat{\mathbf{r}}_{ij}$ can serve directly as an input to the social term of the flocking algorithm (Eq. 2.3). The exact values for range and bearing noise depend on several factors such as camera resolution, lens quality, calibration accuracy, and target deformation.

5.3 Experimental setup

We briefly describe the experimental setup and parameters (Sec. 5.3.1), as well as the simulation environments that are used to obtain the experimental results (Sec. 5.3.2).

5.3.1 Experimental parameters

We perform ten repeated runs of migration experiments to make statistical statements about the scalability of the swarm using different neighbor selection methods, group sizes, swarm densities, agent dynamics, and noise levels.

The specific parameter values we use are informed by our previous experiments with real vision-based quadcopters in indoor [60] and outdoor environments [61], as well as the literature on vision-based drone localization [89, 78, 122, 80, 90, 26, 92, 123, 93]. We choose the radius of an agent as $r = 0.25\,\text{m}$ since it reflects a common physical size of quadcopter platforms used in robotic experiments. The perception radius $r^{\max} = 10\,\text{m}$ is chosen as the distance at which other drones were no longer reliably detected during outdoor experiments. The time delta $\Delta t = 100\,\text{ms}$ is chosen as a reasonable amount of time to solve the visual perception, state estimation, and control problems in real-time. The desired inter-agent distance is set to $d^{\text{ref}} = 1\,\text{m}$ to generate the most compact formation that simultaneously provides enough safety margin against potential collisions.

In order to provide a fair comparison of the visual neighbor selection methods, we choose parameter values that result in comparable numbers of neighbors as the group size increases (Fig. 5.4d). In particular, we set the maximum number of agents for topological neighbor

Name	Set notation
metric	$\mathcal{N}_i^{\text{metric}}(r^{\max})$
visual	$\mathcal{N}_i^{\text{visual}}(r^{\max})$
visual + myopic	$\mathcal{N}_i^{\text{visual}}(2d^{\text{ref}})$
visual + topological	$\mathcal{N}_i^{\text{visual}}(r^{\max}) \cap \mathcal{N}_i^{\text{topo}}(n)$
visual + voronoi	$\mathcal{N}_i^{\text{visual}}(r^{\max}) \cap \mathcal{N}_i^{\text{voronoi}}$

Table 5.1 – Neighbor selection methods used during the experiments.

selection to $n = 6$ since it reflects the average number of Voronoi neighbors for planar configurations [121]. We further let $r^{\max} = 2d^{\text{ref}}$ for myopic interactions since it approaches an average number of six neighbors as the group size increases. We provide an overview of the neighbor selection methods used during the experiments in Tab. 5.1.

At the beginning of each experiment, the agents are spawned randomly within a circular region. The initial positions are sampled uniformly in a non-overlapping fashion using rejection sampling such that no pair of agents are closer than their desired reference distance d^{ref}. The area of the circular region is chosen such that the agent number density ρ_N remains constant for different numbers of agents. The agents exhibit no motion at the beginning of the experiment, i.e., their initial velocities are set to zero. The agents are given a constant navigation direction $\mathbf{r}^{\text{mig}} = [1,0]^\top$ along the horizontal axis which can be seen as a migratory route along the magnetic field [21]. We let the swarm develop its collective motion for a total of $T = 200\,\text{s}$ composed of 2000 isochronous discrete time steps k with duration $\Delta t_k = 0.1\,\text{s}$. At each time step, the agents select their neighbors according to the indicated neighbor selection function (Fig. 5.3) and compute their motion command (Sec. 5.2). We set the separation and cohesion gains to $k^{\text{sep}} = 1\,\text{m}\,\text{s}^{-1}$ and $k^{\text{coh}} = 1\,\text{m}\,\text{s}^{-1}$ to provide an approximate nearest neighbor distance of $d^{\text{ref}} = 1\,\text{m}$. The separation gain is set to $k^{\text{mig}} = 0.5\,\text{m}\,\text{s}^{-1}$ which provides goal-directed motion without overpowering the attractive/repulsive commands. We set the maximum speed an agent can sustain to $v^{\max} = 1\,\text{m}\,\text{s}^{-1}$. A concise overview of the experimental parameters is provided in Tab. 5.2.

In order to provide a fair comparison across vastly different group sizes, we compute the metrics over the last quarter of the simulation, i.e. considering only the final 500 time steps. Particularly for large swarm sizes, we avoid computing metrics during an initial transient period in which agents have not yet aggregated to their final configuration. We refer to the time range during which we compute the metrics as the equilibrium period for convenience.

We report the minimum nearest neighbor distances as a minimum over time over the equilibrium period since it reveals whether collisions occur. For the order and union metrics, we report time averages over the equilibrium period. The mean and standard deviations are computed over the ten independent runs with random initial conditions.

Description	Notation	Value
Agent radius	r	0.25 m
Reference distance	d^{ref}	1 m
Perception radius	$r^{\max}$	10 m
Bearing noise	σ_β	1°
Range noise	σ_d	0.05 m
Maximum topological neighbors	n	6
Maximum speed	$v^{\max}$	$1\,\text{m}\,\text{s}^{-1}$
Separation gain	k^{sep}	$1\,\text{m}\,\text{s}^{-1}$
Cohesion gain	k^{coh}	$1\,\text{m}\,\text{s}^{-1}$
Migration gain	k^{mig}	$0.5\,\text{m}\,\text{s}^{-1}$
Time delta	Δt	0.1 s
Simulation duration	T	200 s

Table 5.2 – Parameters used during the experiments.

5.3.2 Simulation environments

We employ two different simulation environments that serve complementary purposes. The simulation environment with point mass dynamics allows us to rapidly prototype algorithms and quickly generate statistical results with up to one thousand agents without running into time or computational constraints.

The Gazebo simulator, on the other hand, provides more physical realism and allows us to obtain an approximation of how an algorithm would behave on real hardware. However, by default, Gazebo, ROS, and PX4 run asynchronously, meaning that messages are exchanged on a best-effort basis given the computational load. To provide a fair comparison at different group sizes, we must ensure that the number of agents does not have any adverse effects on the simulation fidelity by lockstepping all of its software components. In practice, this means we run Gazebo and PX4 in their respective lockstep modes and additionally pause the simulation at each time step, compute the velocity commands for all agents in parallel, and resume the simulation. Unfortunately, even with lockstepping, Gazebo reaches its computational limits at around one hundred agents, after which the real-time factor decreases considerably and spawning additional agents becomes unreliable. We therefore limit the experiments with quadcopter dynamics to one hundred agents.

5.4 Results

We report results on three sets of complementary simulation experiments: 1) we compare several neighbor selection methods with increasing numbers of agents to show their performance for different swarm sizes (Sec. 5.4.1), 2) we evaluate the neighbor selection methods for increasing inter-agent distances to show the effect of varying agent number densities on the swarm performance (Sec. 5.4.2), and 3) we validate the highest-performing neighbor selection

method (across group sizes and densities) with quadcopter dynamics and realistic sensing noise to show its performance under real-world conditions (Sec. 5.4.3).

5.4.1 Performance across swarm sizes

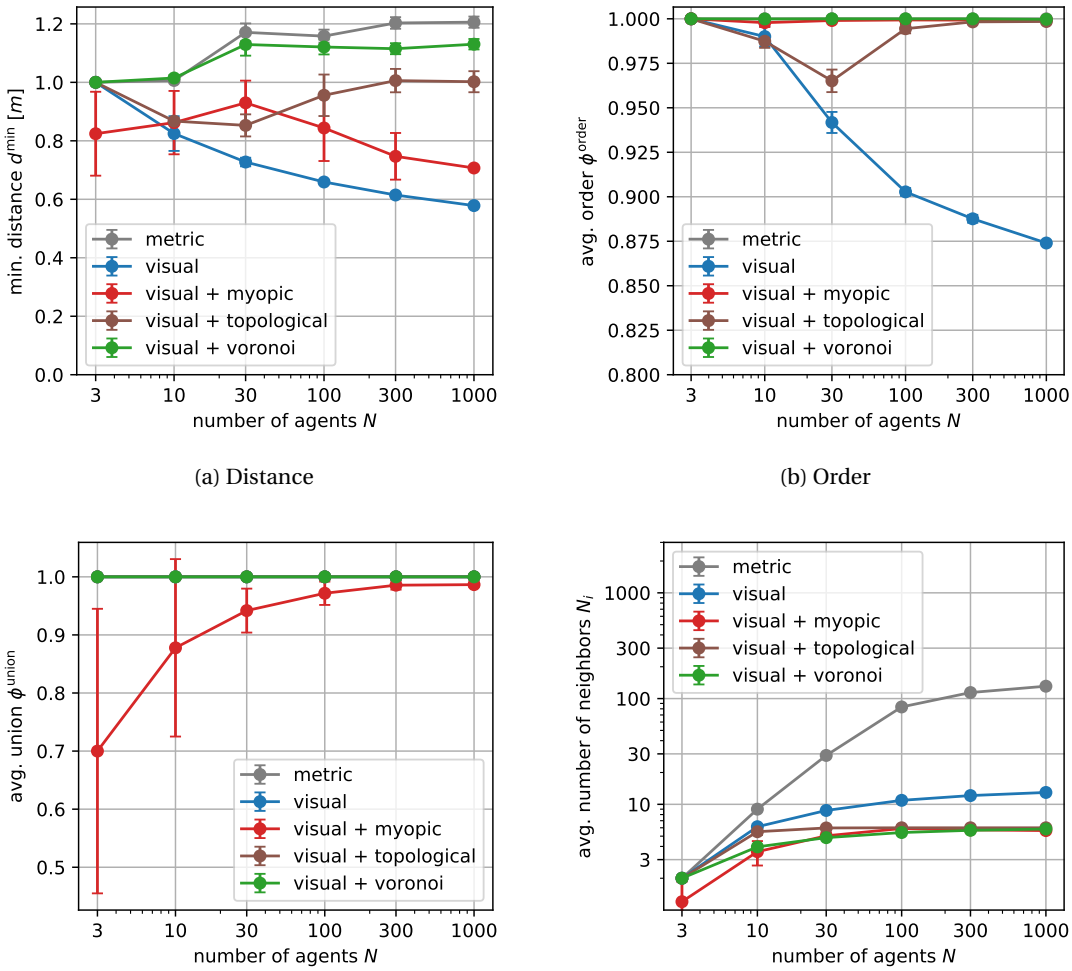
We assess the performance of the swarm for all neighbor selection methods and six levels of increasing group size $N \in \{3, 10, 30, 100, 300, 1000\}$. We set the reference distance $d^{\text{ref}} = 1\,\text{m}$ constant throughout the experiments to keep the agent number density fixed and to allow a direct comparison of the effect of group size.

Visual neighbor selection

Purely *visual* neighbor selection shows the overall lowest performance as the group size increases. There is a considerable performance penalty in the distance and order metrics (Fig. 5.4a and 5.4b). The minimum distance is tracked well only for a group size of 3 agents ($d^{\text{min}} = 1.0 \pm 0.0\,\text{m}$; Fig. 5.4a). The distance gradually approaches the collision threshold of $2r = 0.5\,\text{m}$ and reaches its minimum at 1000 agents ($d^{\text{min}} = 0.58 \pm 0.0\,\text{m}$; Fig. 5.4a). The order metric shows a similar trend since the agents start out perfectly ordered for 3 agents ($\phi^{\text{order}} = 1.0 \pm 0.0$; Fig. 5.4b). However, for larger group sizes, the order metric decreases monotonously until reaching its minimum at 1000 agents ($\phi^{\text{order}} = 0.87 \pm 0.0$; Fig. 5.4b). The swarm stays cohesive as a single unit across all group sizes ($\phi^{\text{union}} = 1.0 \pm 0.0\,\text{m}$; Fig. 5.4c). Generally, using *visual* neighbor selection, the swarm performance decreases as soon as occlusions start to emerge (Fig. 5.4d). There is no performance penalty for 3 agents using *visual* neighbor selection since they predominantly occur in equilateral triangle formations in which there are no occlusions (i.e., $N_i = 2$). For larger group sizes, an increasing number of agents within the perception radius is occluded (32% occluded for $N = 10$; up to 90% occluded for $N = 1000$).

Qualitatively, the trajectories of agents using purely *visual* neighbor selection are jittery (Fig. 5.5a). The agents migrate with considerable deviations from the optimal linear trajectory in the migration direction. In particular, the relative positions of the agents within the swarm are not fixed but rather subject to frequent topology switches. For instance, agents that initially belong to the swarm periphery move towards the swarm center (Fig. 5.5a; blue line) and vice versa.

The topology switches can be explained by considering that an agent within the swarm is exposed to constant changes of its neighbor set (Fig. 5.7). Small agent displacements result in considerable changes of perspective that cause neighbors to appear and disappear from the visible set (Fig. 5.7a and 5.7b: 11 agents appear and 4 disappear, for example). Here, the focal agent is exposed to a total of 32 visibility switches (8 ± 1.22 switches per timestep) over the course of four consecutive seconds of the experiment.



(a) Distance

(b) Order

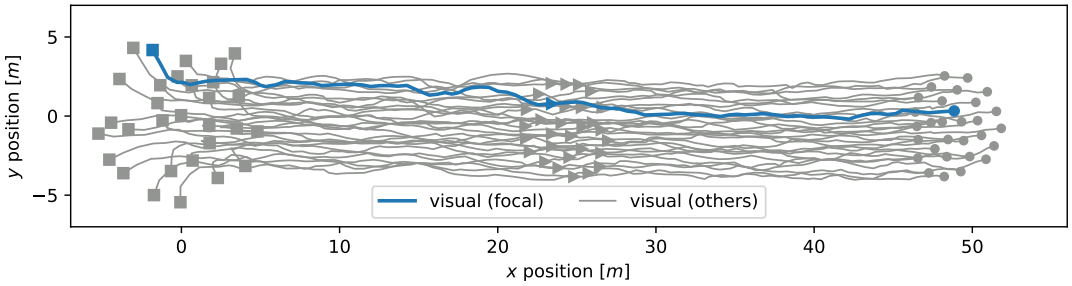
(c) Union

(d) Neighbors

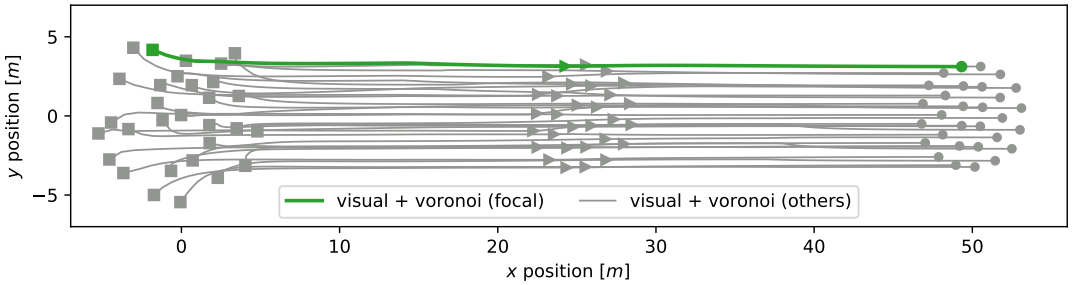
Figure 5.4 – Swarm performance during the collective migration experiment for different neighbor selection methods and reference distance $d^{\text{ref}} = 1\,\text{m}$. We show the effect of different neighbor selection methods on the (a) minimum nearest neighbor distance d^{min}, (b) average order ϕ^{order}, (c) average union ϕ^{union}, and (d) the average number of neighbors N_i, expressed as a function of the number of agents N (note the logarithmic scale). The neighbors are selected as follows: 1) *metric* selects all agents within the perception radius $r^{\text{max}} = 10\,\text{m}$, 2) *visual* selects all visible agents, 3) *visual + myopic* selects all visible agents within a smaller radius $r^{\text{max}} = 2\,\text{m}$, 4) *visual + topological* selects the $n = 6$ topologically closest visible neighbors, and 5) *visual + voronoi* selects the neighbors from adjacent Voronoi regions. The Voronoi neighbor selection method scales most predictably with the number of vision-based agents, i.e., distance, order, and union remain quasi-constant as the swarm size increases.

Alternatives to purely visual neighbor selection

Neighbor selection based on the Voronoi tesselation shows the highest performance of all neighbor selection methods across group sizes. The minimum distance, order, and union



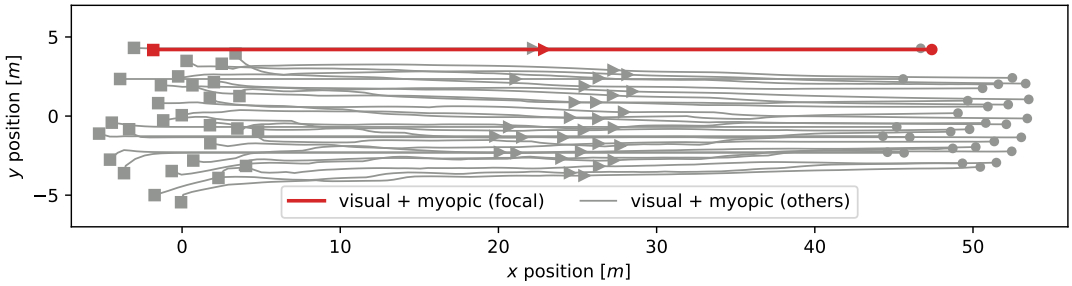
(a) Visual



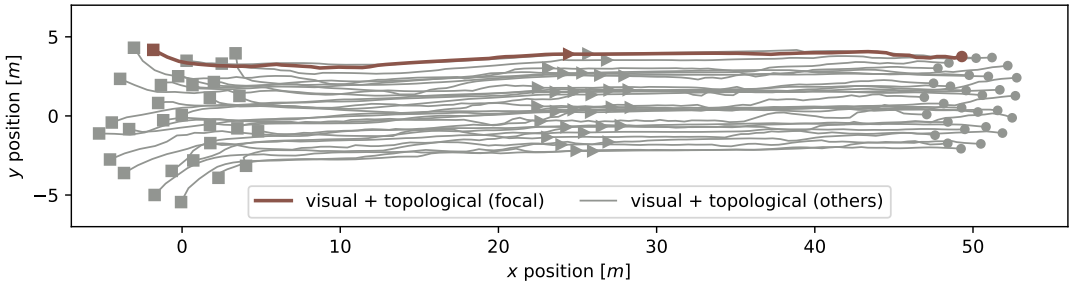
(b) Visual + voronoi

Figure 5.5 – Example trajectories of a swarm of thirty agents during a single run of the collective migration experiment using (a) visual and (b) Voronoi neighbor selection mechanisms. We use the same random seed to create equal initial conditions and highlight an arbitrary focal agent (colored, thick line) to reveal its motion among the other agents (grey, thin lines). The agents start from their initial positions (solid squares) on the left and migrate along the horizontal axis (solid triangles) to the right side of the virtual arena (solid disks). (a) Visual neighbor selection leads to control discontinuities and disorder; agents frequently change positions inside the swarm. (b) Visual and Voronoi neighbor selection together result in collision-free, ordered, and cohesive migration (see Fig. 5.6 for continuation).

metrics show performance comparable to *metric* neighbor selection (Fig. 5.4a, 5.4b, and 5.4c). In particular, the minimum distance is tracked even closer to the reference distance of $d^{\text{ref}} = 1\,\text{m}$ for increasing group size (for 1000 agents: $d^{\text{min}} = 1.13 \pm 0.02\,\text{m}$ for *visual* and $d^{\text{min}} = 1.21 \pm 0.02\,\text{m}$ for *metric*, for example; Fig. 5.4a). This can be explained by considering that *metric* swarms have a significantly larger number of neighbors compared those based on *visual* + *voronoi* neighbor selection for group sizes $N > 3$ (Fig. 5.4d). For example, at $N = 1000$ agents, the *metric* neighbor set contains around 22 times the number of agents than it does for *visual* + *topological* neighbor selection (on average 11.2 ± 8.5 times the number of neighbors for all group sizes; Fig. 5.4d). Recall that the flocking algorithm computes the separation term as a sum of reciprocal distances (Sec. 5.2). Therefore, each neighbor has an additive contribution towards the repulsion (albeit a very small one for distant agents) that explains the slightly larger distances. The agents are perfectly ordered and cohesive for all group sizes ($\phi^{\text{order}} = 1.0 \pm 0.0$ and $\phi^{\text{union}} = 1.0 \pm 0.0$, respectively; Fig. 5.4b and 5.4c). Qualitatively, the paths taken by *visual* + *voronoi* swarms are generally linear and smooth (Fig. 5.5b). The swarm



(a) Visual + myopic

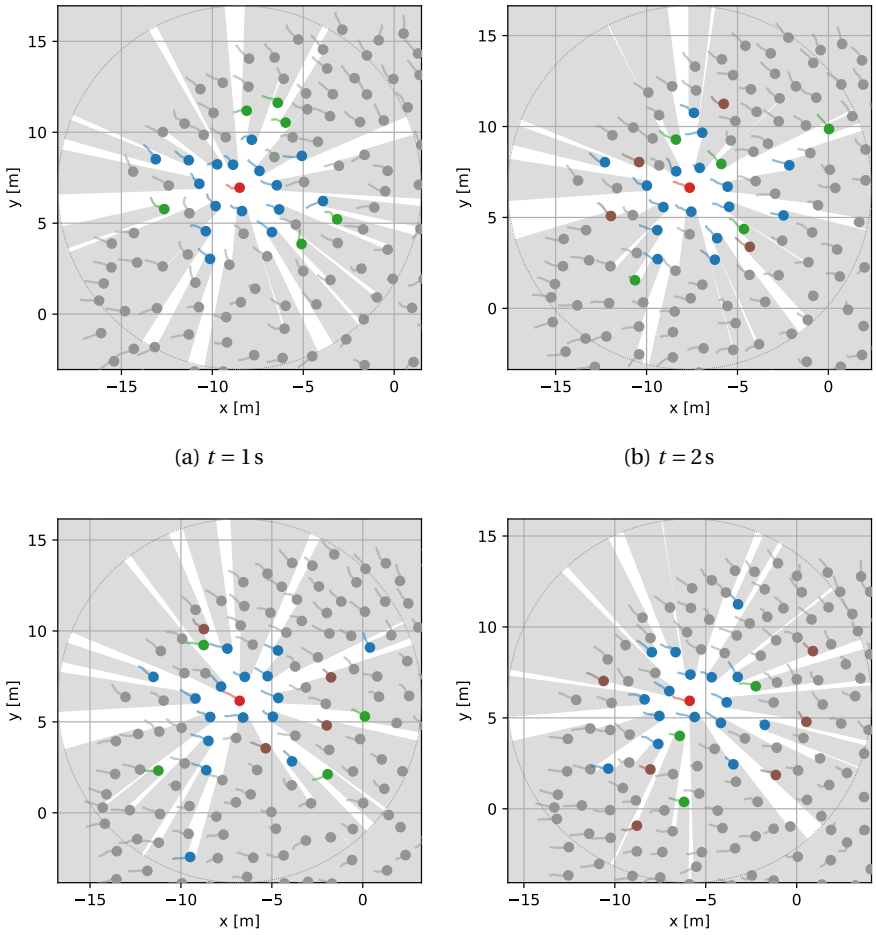


(b) Visual + topological

Figure 5.6 – Example trajectories of a swarm of thirty agents during a single run of the collective migration experiment using (a) myopic and (b) topological neighbor selection mechanisms (see Fig. 5.5 for description). (a) Myopic visual interactions mitigate the discontinuities but lead to fragmentation. (b) Visuo-topological interactions mitigate strong discontinuities but swarms are not well-ordered, especially for peripheral agents.

performs collision-free, ordered, and cohesive collective migration. Switches in the neighbor set do occur but are infrequent and do not lead to unsafe situations or disorder (e.g., changes in neighbor configuration at $x \approx 23\,\text{m}$; Fig. 5.5b).

Swarms that use *visual* + *myopic* or *visual* + *topological* neighbor selection do not perform as well as those using *visual* + *voronoi* selection for different group sizes. Generally, *visual* + *myopic* swarms exhibit low cohesion and easily fragment into several subgroups (Fig. 5.4c). Fragmentation occurs because agents that exit the perception radius are usually found within small subgroups or entirely isolated due to their limited perception range (see subgroups and isolated agent; Fig. 5.6a). The fragmentation phenomenon also skews the minimum distance metric towards lower values with large standard deviations compared to other neighbor selection methods (average of $d^{\text{min}} = 0.82 \pm 0.12\,\text{m}$ across group sizes; Fig. 5.4a). This occurs because isolated agents are usually far away from any other agent (see isolated agent; Fig. 5.6a). We verified that minimum distances to nearest neighbors are usually well-tracked within subgroups of at least three agents. The union metric is always below $\phi^{\text{union}} < 1$ which indicates that fragmentation occurs for all group sizes (Fig. 5.4c). Cohesion is lowest for small groups and approaches, but never reaches, a value of $\phi^{\text{union}} = 1$ that would indicate a single-unit cohesive swarm ($\phi^{\text{union}} = 0.7 \pm 0.25$ for $N = 3$, up to $\phi^{\text{union}} = 0.98 \pm 0.0$ for $N = 1000$; Fig. 5.4c).



(a) $t = 1\,\mathrm{s}$ (b) $t = 2\,\mathrm{s}$

(c) $t = 3\,\mathrm{s}$ (d) $t = 4\,\mathrm{s}$

Figure 5.7 – Visual representation of the switching topologies caused by occlusions during a collective migration experiment. We show the perspective of an arbitrary focal agent (central red disk) over the course of four isochronous time steps $t \in \{1\,\mathrm{s}, 2\,\mathrm{s}, 3\,\mathrm{s}, 4\,\mathrm{s}\}$. The focal agent uses *visual* neighbor selection and therefore perceives only agents within its perception radius that are in a direct line of sight (blue disks), whereas occluded agents are invisible (grey disks). We further highlight visibility switches, i.e., when an agent that has been occluded since the previous time step becomes visible (green disks) and when a previously visible agent becomes occluded (brown disks). A total of 32 visibility switches occur over the course of four seconds.

Note that larger groups exhibit higher union performance since the metric is normalized by group size, i.e., larger groups consist of fewer subgroups relative to the overall group size. Swarms with *visual* + *myopic* neighbor selection are effectively ordered ($\phi^{\mathrm{order}} = 1.0 \pm 0.0$; Fig. 5.4b) Qualitatively, apart from fragmentation, larger subgroups tend to have irregular shapes that are less circular compared to other neighbor selection methods (see the largest subgroup; Fig. 5.6a).

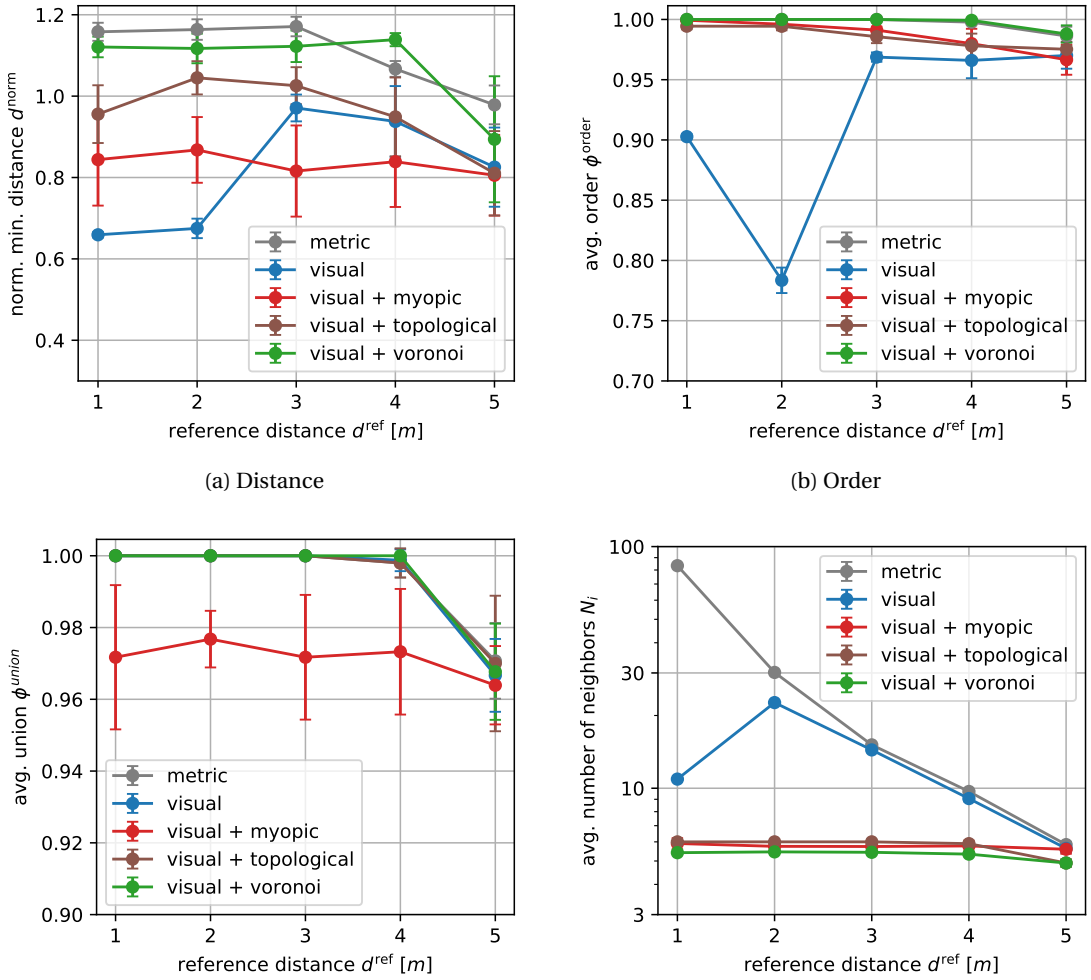
Swarms that use *visual* + *topological* neighbor selection do not exhibit consistent performance across swarm sizes. Especially for intermediate group sizes of 10, 30, and 100 agents, both minimum distances and order metrics suffer a decrease in performance (Fig. 5.4a and 5.8b, respectively). For the respective distances and order metrics, the minimum performance occurs at 30 agents ($d^{\text{min}} = 0.85 \pm 0.04\,\text{m}$ and $\phi^{\text{order}} = 0.97 \pm 0.01$; Fig. 5.4a and 5.4b, respectively). We can explain this behavior by considering that agents *always* select the six closest visible neighbors, irrespective of where they are located. Agents that belong to the swarm center tend to have six neighbors that are spaced around them at approximately equal angles from each other. Conversely, agents on the periphery consider only neighbors in one direction which are subject to occlusions. This leads to similar visual switching topologies as for the purely *visual* neighbor selection, albeit less severe since even the most distant nearest neighbor for $n = 6$ is usually in close proximity. The effect of occlusions is mostly mitigated for larger swarm sizes $N > 100$ since a smaller proportion of agents is located on the periphery relative to the swarm center. We do not observe fragmentation with *visual* + *topological* neighbor selection for any group size ($\phi^{\text{union}} = 1.0 \pm 0.0$; Fig. 5.4c). Qualitatively, *visual* + *topological* interactions generate paths that are not perfectly straight (Fig. 5.6b). We also observe swarms that exhibit rotations, as well as ones that periodically switch between a set of recurring configurations.

5.4.2 Performance across swarm densities

We evaluate the swarm performance for all neighbor selection methods and for five levels of increasing inter-agent distances $d^{\text{ref}} \in \{1\,\text{m}, 2\,\text{m}, 3\,\text{m}, 4\,\text{m}, 5\,\text{m}\}$. We let $N = 100$ to fix the group size and to enable a direct comparison between agent number densities. We define the normalized minimum nearest neighbor distance as $d^{\text{norm}} = d^{\text{min}} / d^{\text{ref}}$ to make the minimum distances more easily comparable for different agent densities.

Visual neighbor selection

Purely *visual* neighbor selection does not show consistent performance for different swarm densities. The performance penalty in distance and order is especially severe for agents in high-density configurations with small reference distances (Fig. 5.8a and 5.8b, respectively). The normalized distance is much lower than the desired reference of $d^{\text{norm}} \geq 1$ and has its minimum for $d^{\text{ref}} \in \{1\,\text{m}, 2\,\text{m}\}$ ($d^{\text{norm}} = 0.66 \pm 0.01$ and $d^{\text{norm}} = 0.67 \pm 0.02$, respectively; Fig. 5.8a). For larger reference distances, $d^{\text{ref}} \in \{3\,\text{m}, 4\,\text{m}\}$, the normalized distance stabilizes again to larger values ($d^{\text{norm}} = 0.97 \pm 0.03$ and $d^{\text{norm}} = 0.94 \pm 0.09$, respectively; Fig. 5.8a) Note that the minimum distance, order, and union metrics for large reference distances $d^{\text{ref}} = 5\,\text{m}$ decrease for all neighbor selection methods. A reference distance of $d^{\text{ref}} = r^{\text{max}}/2 = 5\,\text{m}$ effectively renders all neighbor selection methods *myopic* and fragmentation starts to occur. The union metric indicates that this is indeed the case for $d^{\text{ref}} = 5\,\text{m}$ since all neighbor selection methods show comparable mean performance to *myopic* swarms (average of all neighbor selection methods $\phi^{\text{union}} = 0.97 \pm 0.01$; Fig. 5.8c). The order metric reaches its minimum at $d^{\text{ref}} = 2\,\text{m}$ ($\phi^{\text{order}} = 0.78 \pm 0.01$; Fig. 5.8b). The minimum order coincides with the maximum of



(a) Distance

(b) Order

(c) Union

(d) Neighbors

Figure 5.8 – Swarm performance during the collective migration experiment for different neighbor selection methods and group size $N = 100$. We show the effect of different neighbor selection methods on the (a) normalized minimum nearest neighbor distance d^{norm}, (b) average order ϕ^{order}, (c) average union ϕ^{union}, and (d) average number of neighbors N_i, expressed as a function of the reference distance d^{ref}. The neighbors are selected as follows: *metric* selects all agents within the perception radius $r^{\max} = 10\,\text{m}$, *visual* selects all visible agents, *visual + myopic* selects all visible agents within a smaller radius $r^{\max} = 2\,\text{m}$, *visual + topological* selects the $n = 6$ topologically closest visible neighbors, and *visual + voronoi* selects the neighbors from adjacent Voronoi regions. With the exception of *myopic* conditions (at $d^{\text{ref}} = 5\,\text{m}$), the Voronoi neighbor selection method scales most predictably with the density of the vision-based swarm and the performance remains quasi-constant as the reference distance increases.

the average number of visible neighbors at $d^{\text{ref}} = 2\,\text{m}$ ($N_i = 22.62 \pm 0.04$). This indicates that order follows an inverse relationship with the number of visible neighbors: if more agents

are visible, the likelihood of visual topology switches that lead to disorder increases (Fig. 5.7). The neighbor graph also highlights that the effect of occlusions is maximized at intermediate densities. At high densities, the nearest neighbors occlude most agents in all directions (87% occluded for $d^{\text{ref}} = 1\,\text{m}$; Fig. 5.8d). Conversely, the effect of occlusions diminishes at lower densities since the agents are not large enough to break the line of sight (5% occluded for $d^{\text{ref}} = 3\,\text{m}$, for example; Fig. 5.8d).

Alternatives to purely visual neighbor selection

The Voronoi-based neighbor selection provides the highest and most consistent performance across different group densities. The distance, order, and union metrics remain stable for all but the lowest density level ($d^{\text{ref}} = 5\,\text{m}$) at which interactions are rendered *myopic* (Fig. 5.8a, 5.8b, and 5.8c; Sec. 5.4.2 for discussion of *myopic* interactions). The normalized distance, order, and union remain stable for high and intermediate swarm densities (average $d^{\text{norm}} = 1.12 \pm 0.03\,\text{m}$, $\phi^{\text{order}} = 1.0 \pm 0.0$, and $\phi^{\text{union}} = 1.0 \pm 0.0$; Fig. 5.8a, 5.8b, 5.8c, respectively).

Swarms with *visual* + *myopic* and *visual* + *topological* interactions perform comparatively poorly to *visual* + *voronoi* neighbor across group densities. The *visual* + *myopic* neighbor selection method shows consistently low performance in terms of distance and union metrics (Fig. 5.8a and 5.8c). Myopic interactions effectively reduce the negative impact of occlusions. However, they also induce low distances and fragmentation (average $d^{\text{norm}} = 0.84 \pm 0.02$ and $\phi^{\text{union}} = 0.97 \pm 0.01$ across reference distances; Fig. 5.8a and 5.8c, respectively). Swarms with *visual* + *topological* interactions can avoid the fragmentation issues but their minimum distances fluctuate for different densities (e.g., $d^{\text{norm}} = 1.05 \pm 0.04$ for $d^{\text{ref}} = 2\,\text{m}$ and $d^{\text{norm}} = 0.95 \pm 0.10$ for $d^{\text{ref}} = 4\,\text{m}$; Fig. 5.8a).

5.4.3 Validation in realistic conditions

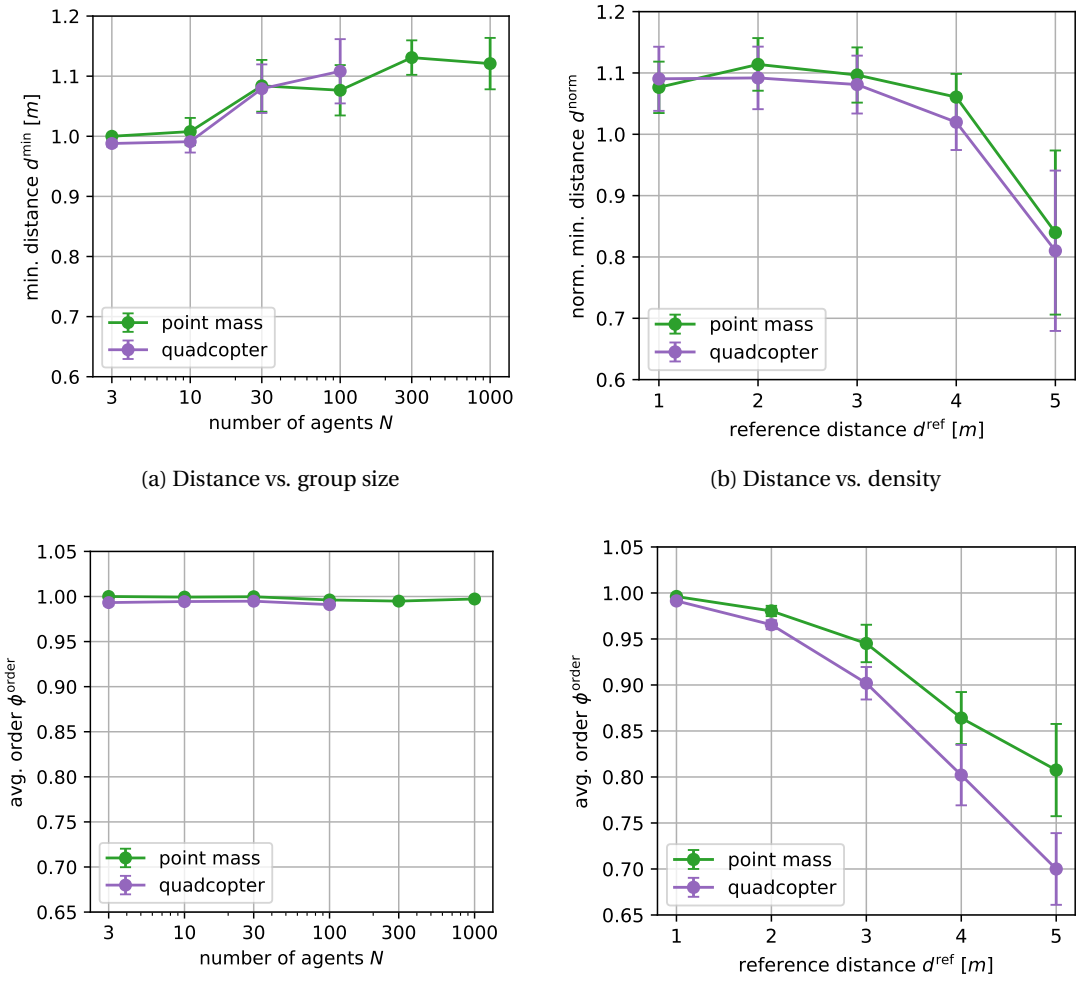
We finally assess the performance of the most promising *visual* + *voronoi* neighbor selection method in more realistic conditions. This is done to evaluate whether the performance transfers to agents with quadcopter dynamics and more realistic sensor noise. Analogous to the previous experiments, we vary the reference distance $d^{\text{ref}} = \{1\,\text{m}, 2\,\text{m}, 3\,\text{m}, 4\,\text{m}, 5\,\text{m}\}$ while keeping the number of agents $N = 100$ fixed to show the effect of agent number density on the flocking performance. Similarly, we vary the number of agents $N \in \{3, 10, 30, 100\}$ ($N \leq 100$ due to the limitations of the physics simulation; Sec. 5.3.2) while keeping the reference distance $d^{\text{ref}} = 1\,\text{m}$ constant to show the effect of group size on the performance metrics. We replace the single-integrator dynamics (Eq. 5.1) with a cascaded PID controller [59] that uses the velocity commands from the flocking algorithm as inputs (Eq. 2.1). We further set the range and bearing noise to $\sigma_d = 0.05\,\text{m}$ and $\sigma_\beta = 1°$, respectively. The specific values are informed by our previous experiments in indoor [60] and outdoor environments [61] and resemble estimates from visual relative localization using object detection with a multi-target state tracker [107].

The *visual + voronoi* neighbor selection method shows comparable performance with point mass agents and quadcopters operating with realistic sensor noise. The swarm performance generally degrades more with increasing reference distances than it does for increasing group size, regardless of the simulation realism. We omit an analysis of the union since the swarms remained cohesive as a single unit during all experiments without exception ($\phi^{\text{union}} = 1.0 \pm 0.0$). We further omit the neighbor statistics since we did not observe any discernable differences. The only noticeable difference between *point mass* and *quadcopter* simulations is the divergence of the average order for decreasing density ($\phi^{\text{order}} = 0.81 \pm 0.05$ for *point mass* and $\phi^{\text{order}} = 0.70 \pm 0.04$ for *quadcopter*; Fig. 5.9d). This difference can largely be attributed to the range noise that increases linearly with distance (Eq. 5.7). The effect of the noise for *quadcopter* dynamics can also be observed in the slightly lower normalized distances compared to *point mass* dynamics (Fig. 5.9a) Interestingly, the more realistic simulation also results in slightly larger minimum distances for 100 agents than would be expected with decreases due to noise ($d^{\text{min}} = 1.07 \pm 0.04\,\text{m}$ for *point mass* and $d^{\text{min}} = 1.11 \pm 0.05\,\text{m}$ for *quadcopter*; Fig. 5.9a). However, these effects are too small to be significant and could have occurred due to chance.

5.5 Conclusions

Methods for multi-agent coordination often make unrealistic assumptions about the information that is available to the individual agent. One of the most pervasive simplifying assumptions is that vision-based agents can sense the state of *all* surrounding neighbors within a metric perception radius, even if they are obstructed by closer ones. Here, we break this common assumption and construct a simple yet realistic model of visibility that selects neighbors only if 1) they appear large enough in the field of view, and 2) are not occluded by other agents. Extensive flocking simulations with the visual occlusion model show that perfectly ordered metric-based swarms become disordered and unsafe when agents react to all of their visible neighbors. These adverse effects can be attributed to small perspective changes that continuously influence the set of visible neighbors, thus causing the agents to move in reaction to the new neighbor configuration. We show that this interplay between visibility constraints and collective motion can lead to severe instabilities for vision-based swarms, especially for large numbers of agents and high swarm densities.

Selecting a subset of visible neighbors from adjacent Voronoi regions significantly improves the swarm performance (i.e., collision avoidance, velocity alignment, and group cohesion) across group sizes and densities. Controlled experiments with subsets of the visual neighbors show that Voronoi-based interactions are a more effective countermeasure against occlusions than metric and topological ones. The main drawback of metric and topological neighbor selection methods is their dependence on specific parameters, namely the perception range and the number of nearest neighbors, respectively. Choosing favorable values for these parameters that provide high performance at all group sizes and densities may be impossible for vision-based swarms. In particular, swarms that select too many neighbors suffer from the adverse effects of occlusions and selecting too few neighbors inevitably leads to fragmentation.



(a) Distance vs. group size

(b) Distance vs. density

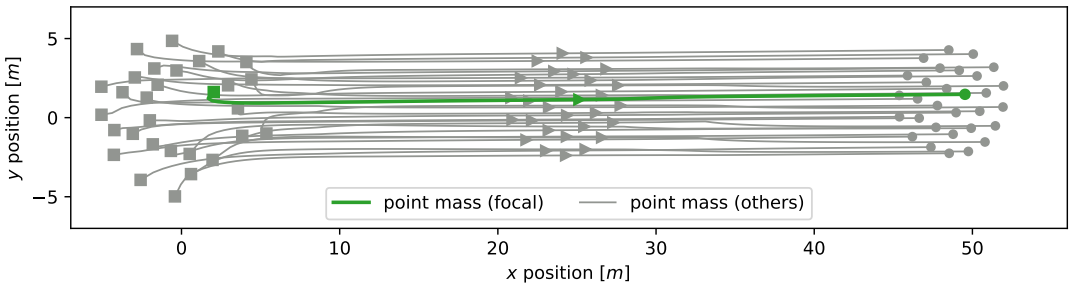
(c) Order vs. group size

(d) Order vs. density

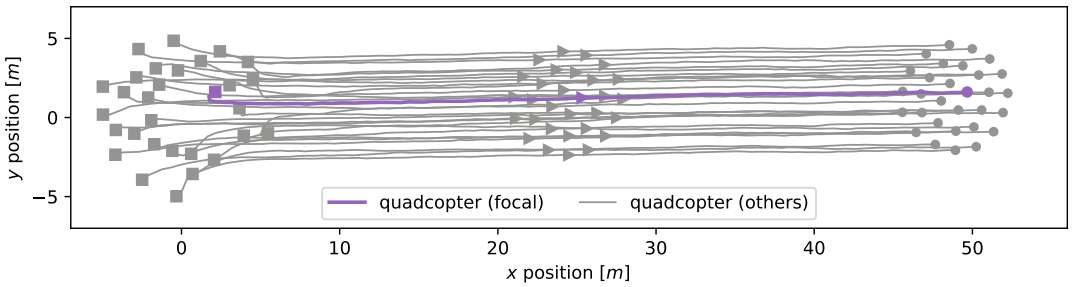
Figure 5.9 – Quantitative results from the collective migration experiment comparing point mass dynamics (green) and with quadcopter dynamics and realistic noise (purple). We show the effect of the simulation realism on the (a, b) minimum (normalized) nearest neighbor distances d^{min} (d^{norm}) and (c, d) average order ϕ^{order} metrics. The metrics are shown as a function of (a, c) the number of agents N (note the logarithmic scale) and (b, d) the reference distance d^{ref}. The metrics follow very similar trends under point mass and quadcopter dynamics with realistic noise. For quadcopter dynamics, we limit the analysis to one hundred agents since simulations with larger group sizes proved to be too unreliable for statistical analysis.

Voronoi-based interactions provide an elegant solution to this problem since they are both parameter-free and spatially balanced [117].

The occlusion model presented here is undoubtedly useful but it neglects an important aspect of vision-based relative localization: errors due to misdetections. False positives (i.e., detecting an agent that is not there) and false negatives (i.e., not detecting an agent that is defacto there)



(a) Point mass dynamics without noise



(b) Quadcopter dynamics with noise

Figure 5.10 – Example paths taken by a swarm of thirty agents during a single run of the collective migration experiment using (a) point mass dynamics without noise and (b) quadcopter dynamics with realistic noise. We use the same random seed to create equal initial conditions and highlight an arbitrary focal agent (colored, thick line) to reveal its motion among the other agents (grey, thin lines). The agents start from their initial positions (solid squares) on the left and migrate along the horizontal axis (solid triangles) to the right side of the virtual arena (solid disks). Apart from the effect of noise, there is no discernable qualitative difference between the point mass and quadcopter swarms.

inevitably occur in real-world conditions but are notoriously difficult to model. The main difficulty is that the distribution of misdetections depends not only on the used hardware and detection algorithm but also on environmental conditions such as background clutter and lighting conditions. Multi-target filtering algorithms can alleviate errors due to sensing noise and false positive detections to some extent but are largely ineffective against false negatives [61].

We argue that occlusions should not be neglected when designing algorithms for vision-based swarms since they are comparatively easy to model. We consider the occlusion model presented here (Eq. 5.3) simple enough to be a drop-in replacement for algorithms that would otherwise default to purely metric interactions (Eq. 5.2). Simple agent-based simulations can thus prevent significant hardware damage by considering occlusions early in the algorithm design and before they are implemented on real robots. The validation presented here is specifically geared towards drones but we expect the results to translate well to other types of

vision-based robots.

6 Conclusions

In this thesis, we have developed and implemented methods for the synthesis of drone swarms in which the robots perceive each other only using visual perception. We have validated the vision-based approach to collective motion using extensive simulations and experiments with real drones in indoor and outdoor environments. The results show that collision-free and cohesive motion of multi-robot systems can be achieved without relying on external infrastructure, wireless communication, or visual markers. The vision-based approach has the potential to increase the safety, flexibility, and scalability of robot swarms.

In the following, we address the limitations of the vision-based approach presented in this thesis (Sec. 6.1). We qualitatively compare the two strategies, i.e., end-to-end learning and detection-based control, and discuss their advantages and limitations (Sec. 6.2). We finally briefly discuss interesting directions for future work (Sec. 6.3), as well as the importance of open-source software and hardware in research (Sec. 6.4).

6.1 Limitations of the vision-based approach

The vision-based approach to multi-robot coordination presented in this thesis has several important limitations. Possibly the most obvious shortcoming compared to, e.g., a communication-based approach is that it is limited to conditions with adequate visibility. Counterexamples are conditions in which there is no visible light (e.g., outdoors at night, indoors without a light source) or external factors reduce visibility (e.g., fog, heavy rain, snow, dust, smoke). However, possible remedies for some low-visibility conditions include the use of external light sources [124] or techniques that are based on light outside of the visible spectrum, e.g., infrared [125] or ultraviolet [122, 80].

Another shortcoming is that vision-based swarms are limited to relatively dense configurations whose inter-agent distances do not exceed their perception range. Therefore, vision-based approaches may not be suitable for missions that involve extensive area coverage such as establishing ad-hoc communication networks [126]. The maximum perception radius itself

depends on many factors, such as the camera resolution and computational constraints and environmental factors such as the low-visibility conditions described above. Communication-based swarms notably suffer from the same limitation since their communication range is also limited, albeit typically to larger distances than for visual perception.

In real-world decentralized swarms, wireless communication can complement visual perception to enable higher efficiency or applications that require cooperation. For instance, wireless communication enables the robots to share not only their current state but also their planned trajectories with each other [88, 127, 11]. This approach can lead to faster navigation and more effective collision avoidance since the robots have access to a future projection of their neighbors instead of relying purely on local visual information. Moreover, wireless communication can also remove the need for external localization since drones can collaboratively build a shared map of the environment in which they self-localize [128, 129, 18]. However, as previously discussed, a fully decentralized implementation with agent-to-agent communication incurs large bandwidth requirements and suffers from inherent scalability limitations. An alternative approach that reduces the required bandwidth is to estimate relative distances using the received signal strength of the wireless transmitters [31].

6.2 End-to-end learning vs. the modular approach

In this thesis, we proposed two different approaches to synthesize vision-based drone swarms: an end-to-end learning-to-control approach based on visual imitation (Chapter 3) and a modular approach based on object detection and tracking (Chapter 4). In the following, we discuss the inherent advantages and disadvantages of the two methods.

The imitation learning approach solves the multi-agent coordination problem using a single neural network that optimizes perception and control jointly. End-to-end learning can be advantageous for performance since the neural network can learn an internal representation that is most suitable to solve a given task. It also has the potential to reduce performance losses at the module boundaries. For example, imposing a specific representation such as object detection for the perception module can discard useful information such as the pose of the drone (Sec. 4.2.1). Subsequent modules, e.g., tracking or control, can not recover this information which may limit the performance of drone swarms. An interesting side effect of end-to-end learning is that camera calibration is no longer required since image observations directly produce control commands. The absence of the transformation to metric space begs the question of whether such intermediate representations, e.g., relative positions, are even necessary for vision-based flocking. The results presented here and others suggest it is not [52, 28, 130].

The lack of intermediate representations in end-to-end learning also has several key limitations that make it hard to recommend for engineering problems. In the learning-to-control framework, the predicted command is a complex nonlinear function of the input image. However, given an input image that produces a command, it is not immediately clear what

causes a specific behavior and which features in the input image are responsible. Attribution methods (Sec. 3.5.4) can be helpful to answer these questions but are far from an exact science since different methods can produce vastly different results [131, 86]. Another problem is that end-to-end learning is inflexible since it couples perception and control too tightly. For example, suppose we move a camera to make space for additional hardware components on a drone. In that case, the policy may no longer work since the old camera configuration is encoded in its weights. A modular approach that decouples perception and control improves flexibility since a frame transformation can easily be adjusted. Moreover, since there are no individual modules in end-to-end learning, components cannot be tested and evaluated in isolation, making it challenging to predict whether a method will work before trying it in the control loop. Controlled simulation experiments also show that end-to-end policies based on raw images perform suboptimally compared to those that have access to standard computer vision representations, e.g., depth estimation and semantic segmentation [132]. These results suggest that the intermediate representations obtained from end-to-end learning may not be the most suitable or most efficient to solve a given task after all.

In summary, end-to-end learning has great potential to optimize perception and action jointly but it is difficult to justify for real-world applications that demand modularity, safety, flexibility, and interpretability.

6.3 Possible directions for future work

The method for detection and tracking presented in this thesis does not explicitly take the robotic platform and its dynamics into account (Sec. 4.2). However, observing the orientation of the platform in addition to its relative range and bearing can give very useful additional information for predicting its future state. In the case of quadcopters, there exists a strong coupling between the rotational and translational dynamics due to their under-actuated nature [133]. For example, a nonzero pitch angle will inevitably lead to a translation along the longitudinal axis, i.e., induce a forward or backward motion. Therefore, detecting the full pose of the platform and considering it in the motion model of the tracking phase may lead to more accurate state estimates. Similar arguments can be made for other types of robotic platforms such as fixed-wing drones [134].

More sophisticated control algorithms such as model predictive control (MPC) can be used to improve the flocking performance of vision-based swarms. In particular, they can replace the combination of cascaded PID controllers and potential-field-based flocking algorithms used in this thesis to improve the speed, order, and safety of the swarm [11]. MPC algorithms use a dynamic model of the drones to predict how their motion will develop in the future. Hence, they have an advantage over purely reactive algorithms used in this thesis since they can optimize for the best command to issue based on a projection of the future state of the swarm. The detection and tracking methods developed in this thesis are suitable for integration with MPC algorithms since they provide relative positions and velocities of neighboring agents

(Chapter 4).

The main focus of this thesis has been the motion coordination of drone swarms with vision-based agent-to-agent interactions and did not consider obstacles. However, vision-based swarms need to perform autonomous navigation in unstructured and unknown environments such as cities or forests to be useful for many applications (Sec. A). To this end, the detect-and-track approach presented in this thesis (Sec. 4) can be combined with vision-based methods for obstacle detection and avoidance [7, 110].

6.4 Closing remarks

This thesis would not have been possible without relying on third-party software tools and libraries and the importance of free software in research — especially in fields such as robotics — cannot be overstated. Notable examples include the Robot Operating System (ROS) [108], the Gazebo physics simulator [53], the RotorS drone simulation framework [54], the Kalibr visual-inertial calibration toolbox [104], the PX4 open-source drone autopilot [59], SwarmLab [48], the MRS UAV system [55], and many others. Over the course of this thesis, we attempted to follow these examples and contribute open-source software to the community (Appendix B).

This thesis also resulted in the development of a custom quadcopter platform that required solutions to a myriad of technical problems. Fortunately, we could also — at least partly — rely on open hardware platforms such as the Pixhawk autopilot [58] and the PX4Flow smart camera [135]. The world of open hardware is — compared to open-source software — still in its infancy, but the work of many dedicated researchers is slowly improving the situation.

A Vision-based localization in dynamic environments

This chapter proposes both a dataset and method for vision-based localization in dynamic environments. Dynamic environments such as multi-robot deployments in urban areas are still challenging for popular visual-inertial odometry (VIO) algorithms. Existing datasets typically fail to capture the dynamic nature of these environments, therefore making it difficult to quantitatively evaluate the robustness of existing VIO methods. To address this issue, we propose three contributions: firstly, we provide the VIODE dataset, a novel benchmark recorded from a simulated drone that navigates in challenging dynamic environments. The unique feature of the VIODE dataset is the systematic introduction of moving objects into the scenes. It includes three environments, each of which is available in four dynamic levels that progressively add moving objects. The dataset contains synchronized stereo images and IMU data, as well as ground-truth trajectories and instance segmentation masks. Secondly, we compare state-of-the-art VIO algorithms on the VIODE dataset and show that they display substantial performance degradation in highly dynamic scenes. Thirdly, we propose a simple extension for visual localization algorithms that relies on semantic information. The results show that scene semantics are an effective way to mitigate the adverse effects of dynamic objects on VIO algorithms.

The work presented in this chapter is adapted from [63][1]:

- K. Minoda, F. Schilling, V. Wüest, D. Floreano, and T. Yairi, "VIODE: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1343–1350, Apr. 2021.

A.1 Introduction

Vision-based localization in dynamic environments is an important and challenging task in robot navigation. Camera images can provide a rich source of information to estimate the pose of a robot, especially in environments such as indoor and urban areas where GNSS information

[1] Video: https://youtu.be/LlFTyQf_dlo.

may be unreliable or entirely unavailable. In such environments, however, dynamic objects such as humans, vehicles, or other mobile robots often coexist in the same workspace. These dynamic objects can be detrimental to vision-based algorithms since they commonly use the assumption of a static world in which they self-localize. Breaking the static-world assumption may result in large estimation errors since the algorithms cannot distinguish between dynamic objects and static ones.

One possible way to remove these errors is to introduce additional proprioceptive information, such as data from inertial measurement units (IMUs), into the estimation pipeline. Unlike a camera that measures the environment, proprioceptive sensors measure the internal state of the robot, which is usually not affected by the environment. Thus, visual-inertial odometry (VIO) has the potential to perform more robustly compared to purely visual odometry (VO). However, as we show in this study, adding inertial information does not necessarily guarantee robustness, and further information is required to obtain a reliable state estimation algorithm.

Despite the importance of visual-inertial fusion for robust localization in dynamic scenarios, there are no publicly available datasets that are suitable as a benchmark of VIO algorithms in dynamic environments. While a considerable amount of VIO datasets are proposed, most of them contain only a few dynamic objects. While there are VIO algorithms that aim to perform robustly in dynamic environments [136, 137], it is difficult to quantitatively compare them due to the lack of a benchmark that addresses object dynamicity.

This chapter presents a novel dataset called VIODE (VIO dataset in Dynamic Environments) — a benchmark for assessing the performance of VO/VIO algorithms in dynamic scenes. The environments are simulated using AirSim [138], which is a photorealistic simulator geared towards the development of perception and control algorithms. The unique advantage of VIODE over existing datasets lies in the systematic introduction of dynamic objects at increasing numbers and in different environments (Fig. A.1). In each environment, we use the same trajectory of the aerial vehicle to create data sequences with an increasing number of dynamic objects. Therefore, VIODE users can isolate the effect of the dynamic level of the scene on the robustness of vision-based localization algorithms. Using VIODE, we show that the performance of state-of-the-art VIO algorithms degrades as the scene gets more dynamic.

The VIODE dataset further contains ground-truth instance segmentation masks, since we believe that semantic information will be useful for researchers and engineers to develop robust VIO algorithms in dynamic scenes. In order to assess the effects of semantic information in algorithms operating on the VIODE dataset, we develop a method incorporating semantic segmentation into VINS-Mono [25]. We refer to this method as VINS-Mask. VINS-Mask masks out the objects which are assumed to be dynamic. Using the proposed dataset, we demonstrate that VINS-Mask outperforms the existing state-of-the-art algorithms. Though VINS-Mask uses ground-truth segmentation labels provided by AirSim, these results indicate the potential of the utilization of scene semantics in dynamic environments.

In summary, the main contributions of this work are:

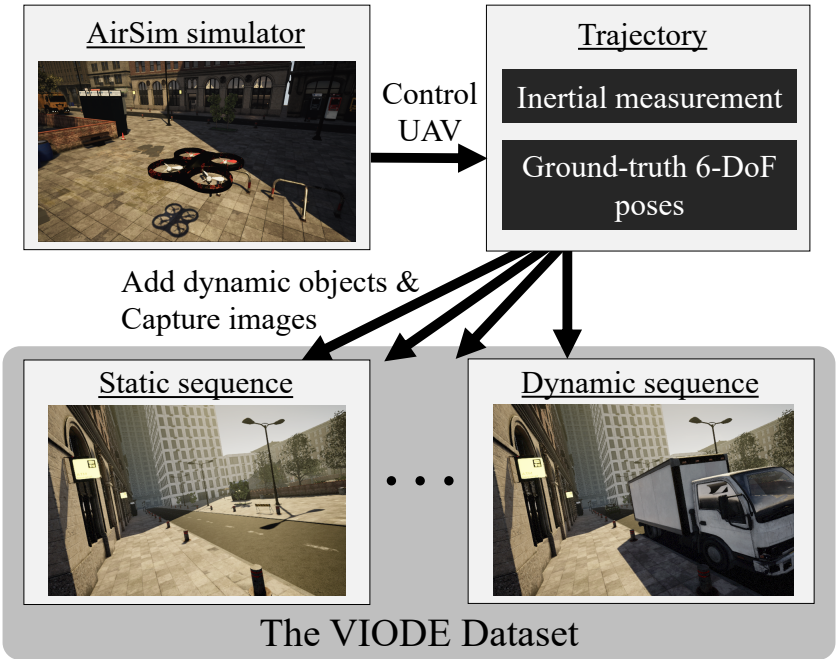


Figure A.1 – The VIODE dataset is created using a photorealistic drone simulator to which we systematically add moving objects while keeping trajectory, lighting conditions, and static objects the same — a setup that would be prohibitively difficult to achieve with real-world data. Each environment in the dataset contains four sequences with the same trajectory and IMU data, but with increasing levels of dynamic objects.

- We propose a novel simulated visual-inertial dataset (VIODE) where we systematically add dynamic objects to allow benchmarking of the robustness of VO and VIO algorithms in dynamic scenes.

- Using VIODE, we experimentally show performance degradation of state-of-the-art VIO algorithms caused by dynamic objects.

- We show that incorporation of semantic information, implemented as a state-of-the-art VIO algorithm with semantic segmentation, can alleviate the errors caused by dynamic objects.

A.2 Related work

In this section, we briefly outline related work in three areas. Firstly, we summarize existing VIO datasets and their shortcomings. Secondly, we discuss the state-of-the-art of VIO algorithms. Finally, we highlight methods for robust vision-based localization in dynamic environments.

VIO datasets

Existing VIO datasets are diverse, as they span various carriers, environments, and sensor setups. However, they are not sufficient to assess the robustness of VIO in dynamic environments on 6-DoF trajectories.

Most of the datasets that are recorded on drone or handheld rigs contain only a few moving objects. For instance, widely used VIO benchmarks such as the EuRoC MAV dataset [139] only contain a small number of people moving in indoor scenes and thus the scenes are mostly static. Thus they are not suitable for benchmarks for systematically studying performance in environments with moving objects.

Outdoor datasets tend to contain more dynamic objects. The TUM VI dataset [140] and the Zurich Urban dataset [141] contain outdoor sequences in which moving vehicles and people appear sporadically. KITTI [142], Urban@CRAS [143], and KAIST Urban [144] are recorded from driving cars in urban areas. These datasets, especially the latter two, contain several moving vehicles. However, the field of view in these datasets still contains mainly static objects and they are thus not challenging enough as a benchmark of state estimation robustness in dynamic environments. The Oxford Multimotion Dataset [145] is an indoor dataset that contains dynamic objects. Their dataset aims at providing a benchmark for motion estimation of moving objects as well as vehicle self-localization. However, the scenes do not contain environments typically encountered by robots such as drones. Furthermore, these datasets do not cover the most challenging dynamic scenes which can occur in real-world applications of vision-based localization.

To the best of our knowledge, the ADVIO dataset [146] contains the most challenging scenes in terms of dynamicity among the existing VIO datasets. However, their work does not contain quantitative information on the dynamic level. Moreover, unlike the VIODE dataset, dynamic objects cannot be isolated to estimate their effect on the robustness of VIO.

In conclusion, the existing VIO datasets are not suitable to systematically benchmark the robustness of VIO methods in the presence of dynamic objects. The VIODE dataset seeks to enhance the development of localization in dynamic scenes by providing a challenging, quantitative, and high-resolution benchmark. We achieve this goal with a systematic configuration of dynamic objects and an evaluation of dynamic levels in each scene.

VIO algorithms

Here, we provide an overview of existing VIO algorithms, including VINS-Mono [25] and ROVIO [147] which we benchmark on the VIODE dataset.

Common VIO algorithms can be classified as either filter-based or optimization-based. MSCKF [148] and ROVIO [147] are both filter-based algorithms that fuse measurements from cameras and IMUs using an extended Kalman filter. On the contrary, optimization-based approaches

utilize energy-function representations for estimating 6-DoF poses. OKVIS [149] utilizes non-linear optimization and corner detector. VINS-Mono [25] and VINS-Fusion [150] are optimization-based algorithms that contain a loop detection and closure module and, as a result, achieve state-of-the-art accuracy and robustness. Furthermore, the recently proposed ORB-SLAM3 [151] is an optimization-based visual-inertial SLAM algorithm that achieves high performance and versatility. In [152] the authors provide a comprehensive comparison among the existing open-sourced monocular VIO algorithms. Their survey highlights that VINS-Mono and ROVIO are the two best-performing methods among the existing monocular VIO algorithms in terms of accuracy, robustness, and computational efficiency. Thus, in this study, we compare these two algorithms on the VIODE dataset.

Vision-based localization in dynamic environments

Vision-based localization algorithms such as visual odometry (VO) and visual simultaneous localization and mapping (vSLAM) commonly make use of the random sample consensus (RANSAC) algorithm [153]. RANSAC is an iterative algorithm to estimate a model from multiple observations which include outliers. Thus, common vision-based methods often adopt this algorithm to enhance their performance in dynamic environments. One of the major drawbacks is that RANSAC is more likely to fail when the observation contains outliers that follow the same hypothesis. For example, RANSAC in visual localization is more vulnerable to one large rigid-body object than several small objects with different motions.

Some recent works report that the use of semantic segmentation or object detection can improve the robustness of algorithms in dynamic environments. Mask-SLAM [154] uses the ORB-SLAM architecture while masking out dynamic objects using a mask generated from semantic segmentation. DynaSLAM [155] is built on ORB-SLAM2 and combines a geometric approach with semantic segmentation to remove dynamic objects. The authors of Mask-SLAM and DynaSLAM evaluate proposal methods on their original dataset recorded in dynamic environments. Empty Cities [156] integrates dynamic object detection with a generative adversarial model to inpaint the dynamic objects and generate static scenes from images in dynamic environments.

There are several works that not only address robust localization in dynamic environments but also extend the capability to track surrounding objects [157, 158, 159, 160]. For example, the results of DynaSLAM II [157] show that motion estimation of objects in the environmnet can be beneficial for ego-motion estimation in dynamic environments.

Compared to VO/vSLAM, the performance of VIO algorithms in dynamic environments has not been investigated in depth. Several works, however, report that the utilization of semantic/instance segmentation improves the performance of VIO in dynamic environments [136, 137]. A limitation of these two studies is, however, that they reported performances only on limited data. For example, [136] uses only a short subsequence of the KITTI dataset. Our VIODE benchmark is well suited to systematically assess and compare these VIO algorithms.

A.3 Method

A.3.1 The VIODE dataset

We created the VIODE dataset with AirSim [138], a photorealistic simulator based on Unreal Engine 4, which features commonly used sensors such as RGB cameras, depth cameras, IMUs, barometers, and LiDARs. To simulate motions that are typically encountered in VIO applications, we used a quadrotor as the carrier vehicle as it is a platform where VIO algorithms are commonly employed.

The VIODE dataset is designed for benchmarking VIO algorithms in dynamic environments. This can be accredited to three unique features. Firstly, compared to other existing datasets, VIODE contains highly dynamic sequences. Secondly, we provide quantitative measures of the dynamic level of the sequences based on clearly defined metrics. Finally, and most importantly, VIODE allows us to isolate the effect of dynamic objects on the robustness of VIO. In general, the performance of VIO is influenced not only by dynamic objects but also by a variety of parameters such as the accuracy of the IMU, the type of movements (e.g. rotation or translation), the lighting condition, and the texture of the surrounding objects. Therefore, a way to fairly assess algorithm robustness in dynamic scenarios is to compare multiple sequences where the only difference is the dynamic level. To achieve this, we create four sequences of data on the same trajectory and strategically add dynamic objects.

Dataset content

The dataset is recorded in three *environments*, one indoor and two outdoor environments. In each environment, we generate four *sequences* which are recorded while executing the same trajectory. The only difference between these four recorded sequences is the number of dynamic vehicles. To set up the situation as realistic as possible, we also placed static vehicles in all the sequences. The three environments can be described as follows (Fig. A.2):

- `parking_lot` contains sequences from a parking lot indoor environment.
- `city_day` contains sequences from a modern city environment during day time. This is a commonly encountered outdoor environment that is surrounded by tall buildings and trees.
- `city_night` contains sequences from the same environment as `city_day`, however during night time. Trajectories of ego-vehicle and dynamic objects are different from those of the `city_day` environment.

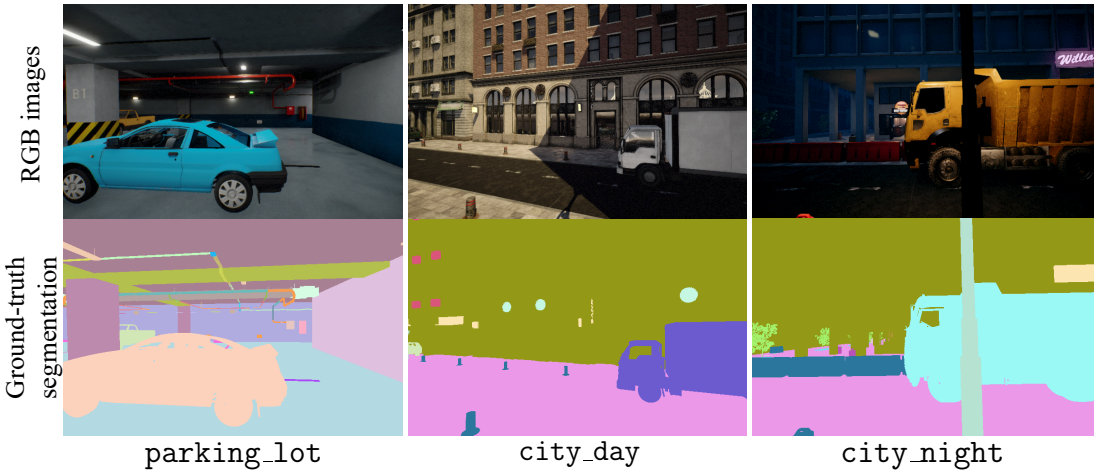


Figure A.2 – Example images and ground truth segmentations of the different environments. We show RGB images (top row) from each environment and ground-truth segmentations (bottom row) from each of the three environments. The images and their corresponding segmentations are time-synchronized and obtained directly from AirSim.

Dataset generation

The procedure for generating VIODE data involves two steps: 1) collect IMU and ground-truth 6-DoF poses, and 2) capture images and segmentation masks (Fig. A.1). These steps were executed on a desktop computer running Windows 10 with an Intel(R) Core i7-9700 CPU and a GeForce RTX 2700 SUPER GPU.

In the first step, we use the ROS wrapper provided by AirSim to generate synchronized IMU measurements and ground-truth odometry data at a rate of 200 Hz. While recording this sensory data, we control the drone with the PythonAPI in AirSim. We generate one trajectory for each environment. Each of the trajectories contains linear accelerations along the longitudinal, lateral, and vertical axes, as well as rotation movements around roll, pitch, and yaw. In order to obtain accurate timestamps, we slow down the simulation by a factor of 0.05 during this procedure.

In a second step, we add images to the above data. To do so, we undersample ground-truth poses from the odometry sequence to a rate of 20 Hz. In each of the undersampled poses, we capture synchronized RGB images and segmentation maps with stereo cameras. This procedure is done four times in each environment to generate four sequences with different dynamic levels: `none`, `low`, `mid`, and `high`.

Sensor setup and calibration

The drone in AirSim is equipped with a camera and an IMU. The camera captures RGB images and AirSim provides segmentation maps.

- **Stereo camera** uses two equal cameras and captures time-synchronized RGB images at a rate of 20 Hz with WVGA resolution (752 × 480 px). We set up the stereo camera with a baseline of 5 cm. Both of these cameras are global shutter cameras and have a 90° field of view (FOV).
- **Ground-truth segmentation** images are computed for both cameras at a rate of 20 Hz, synchronized with the corresponding RGB images (Fig. A.2). The provided segmentation is an instance segmentation. As such, the vehicles and all the other objects in the scene are labeled as different objects.
- **IMU** data is also acquired from AirSim at a rate of 200 Hz. This uses the same parameters as MPU-6000 from TDK InvenSense. The noise follows the model defined in [161]. Since the IMU data does not take into account the vibration of rotors, the noise magnitude is lower than that of IMU data recorded on real-world drone flights and similar to the one recorded with handheld carriers.
- **Ground-truth trajectory** is also included for all the sequences of the dataset. This consists of ground-truth 6-DoF poses provided by the simulator. This is recorded at a rate of 200 Hz.
- **Intrinsic and extrinsic parameters** are provided for the stereo camera and camera-IMU extrinsics.

Dynamic objects & dynamic level evaluation

We use cars provided in Unreal Engine 4 as dynamic objects since cars are commonly encountered dynamic objects in real-world applications. We use six types of vehicles with varying textures and sizes. These dynamic objects are controlled by Unreal Engine with constant speeds along the designated trajectories. As mentioned previously, we generated four sequences in each environment: `none`, `low`, `mid`, and `high`. The `none` sequences do not contain any dynamic objects. Starting from `none`, we progressively add the vehicles to generate three different dynamic sequences: `low`, `mid`, and `high`. Furthermore, we allocate not only dynamic vehicles but also some static ones to make the scenes more realistic.

We also introduce metrics to evaluate how dynamic the sequences are. Simply counting the number of dynamic vehicles in the scene is not a suitable metric of the dynamic level. For example, even if the number of dynamic objects is the same, the scene is more dynamic when the objects are close to the camera. Thus, a better metric is necessary to evaluate the dynamic level. Possible information sources for defining a better metric are semantic information, optical flow, ground-truth ego-motion, or rigid-body-motion of vehicles. Here, we assess the dynamic level by considering how much of the FOV is occupied by dynamic objects based on ground-truth instance segmentation. The pixel-based dynamic rate r_{pix} is defined as

$$r_{\mathrm{pix}} = N_{\mathrm{pix}}^{\mathrm{dyn}} \; N_{\mathrm{pix}}^{\mathrm{all}} \tag{A.1}$$

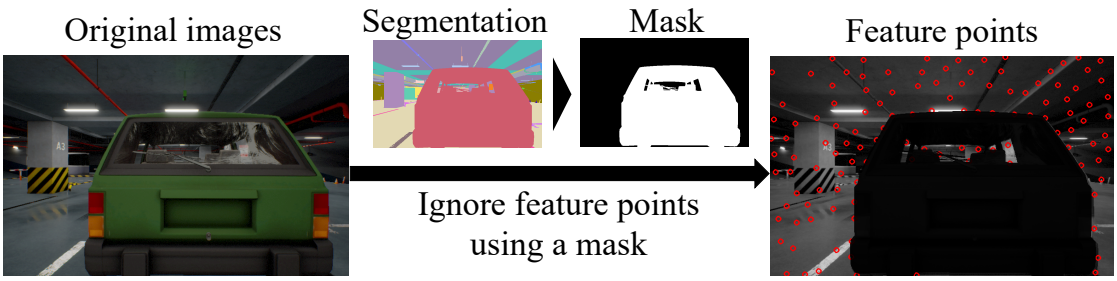


Figure A.3 – Overview of the processing steps of the VINS-Mask algorithm. We use semantic segmentation maps to create binary masks that represent dynamic (white) and static (black) image regions. For example, the semantic class *car* (pink) is likely dynamic and we therefore ignore its feature points in the image. The remaining feature points are then coupled with IMU information analogously to VINS-Mono to estimate the 6-DoF pose of the vehicle.

where $N_{\text{pix}}^{\text{dyn}}$ is the number of pixels occupied by dynamic vehicles and $N_{\text{pix}}^{\text{all}}$ is the total number of pixels in the image.

However, the pixel-based dynamic rate does not contain speed information of the surrounding objects. In order to obtain a metric for the magnitude of the motion of objects, we also define the optic-flow-based (OF-based) dynamic rate r_{of} as

$$r_{\text{of}} = \frac{1}{N} \sum_{(x,y)\in D} \sqrt{\|\nabla I(x,y) - \nabla I_{\text{none}}(x,y)\|^2} \tag{A.2}$$

where D is a set of pixel coordinates in the frame which belong to dynamic objects. We determine D from ground-truth instance segmentation. N is the total number of pixels, $\nabla I(x,y)$ is the optic flow at (x,y) in a frame, and $\nabla I_{\text{none}}(x,y)$ is the optic flow at (x,y) in a corresponding frame in `none` sequence from the same environment.

We further provide the two types of dynamic rate along the time axis (Fig. A.8).

A.3.2 The VINS-Mask algorithm

One of the goals of this study is to show that scene semantics have the potential to improve the performance of VIO in challenging dynamic scenes. To this end, we develop the method VINS-Mask, which is based on VINS-Mono [25]. VINS-Mask uses the same algorithm as VINS-Mono, except that it avoids using feature points on dynamic objects by leveraging semantic information. In the feature points extraction phase of VINS-Mask, we perform semantic/instance segmentation for each image. By exploiting the segmentation and preliminary knowledge, we mask out the region of the dynamic objects right before the feature extraction phase of VINS-Mono (Fig. A.3). By applying the mask, VINS-Mask can estimate the robots' pose based on reliable feature points (Fig. A.4). Although this technique is not a novel idea as stated in Sec. A.3.1, we use this method to assess the impact of semantic information on the robustness of VIO in dynamic scenarios.

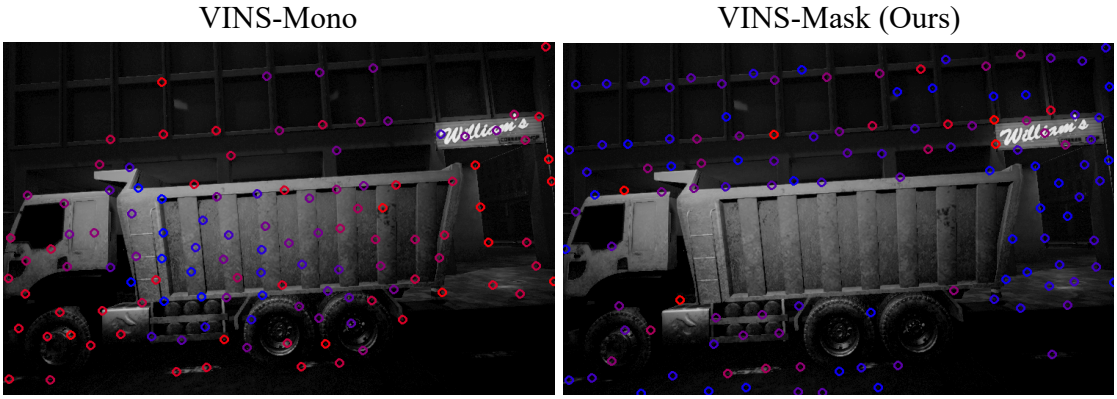


Figure A.4 – Comparison of feature points without (left) and with (right) masking of dynamic objects. We visualize both the extracted (blue circles) and tracked (red circles) feature points in VINS-Mono and VINS-Mask. The redder the feature point is colored, the longer the feature has been tracked and more likely to be used in VINS-Mono and VINS-Mask to estimate the 6-DoF pose. VINS-Mono relies on feature points located on dynamic objects while VINS-Mask excludes those regions entirely.

In this work, we use ground-truth segmentation labels provided in the VIODE dataset, instead of applying a semantic segmentation algorithm on camera images. The mask consists of the region which belongs to vehicles. We include both dynamic and static vehicles in the mask to make the setup more realistic, although it is possible to distinguish moving objects from static objects with ground-truth instance segmentation. This is because distinguishing moving objects from static objects in real-world applications is itself a challenging task.

A.4 Results

A.4.1 Performance metrics

To analyze the performance of VIO algorithms, we use the absolute trajectory error (ATE) [162] which is defined as

$$\mathrm{ATE}(\mathbf{P}_{1:T}, \mathbf{Q}_{1:T}) = \sqrt{\frac{1}{T}\sum_{k=1}^{T} \|\mathrm{trans}(\mathbf{F}_k)\|^2} \tag{A.3}$$

where $\mathbf{P}_1, ..., \mathbf{P}_T \in \mathrm{SE}(3)$ is the estimated trajectory, $\mathbf{Q}_1, ..., \mathbf{Q}_T \in \mathrm{SE}(3)$ the ground-truth trajectory, $\mathbf{F}_k = \mathbf{Q}_k^{-1}\mathbf{S}\mathbf{P}_k$ is the absolute trajectory error at time step k, $\mathrm{trans}(\mathbf{X})$ refers to the translational components of $\mathbf{X} \in \mathrm{SE}(3)$, and $\mathbf{S}$ the rigid-body transformation between $\mathbf{P}_{1:T}$ and $\mathbf{Q}_{1:T}$ calculated by optimizing a least-square problem.

Moreover, we use relative pose error (RPE) [162] for local accuracy of the sub-trajectory. RPE

	ROVIO	VINS-Mono	VINS-Mask (Ours)
parking_lot	5.27	14.7	**1.12**
city_day	17.1	16.2	**1.26**
city_night	5.19	1.91	**1.26**

Table A.1 – Performance degradation rate r_{deg} for different VIO algorithms and environments.

for the k-th time step is defined as

$$\text{RPE}(\mathbf{P}_{1:T}, \mathbf{Q}_{1:T}, k) = \|\text{trans}(\mathbf{E}_k)\| \tag{A.4}$$

where $\mathbf{E}_k = (\mathbf{Q}_k^{-1}\mathbf{Q}_{k+\Delta})^{-1}(\mathbf{P}_k^{-1}\mathbf{P}_{k+\Delta})$ is relative pose error at time step k. Here, Δ is a fixed time interval for which we calculate the local accuracy.

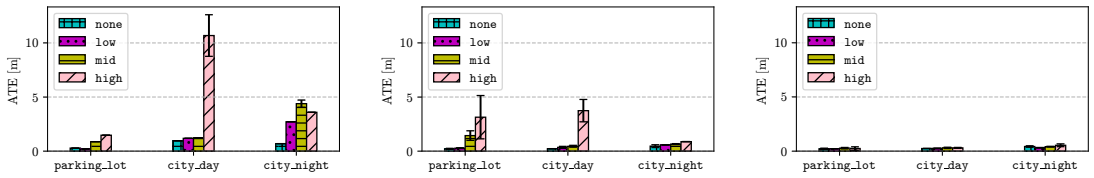
We additionally introduce the performance degradation rate r_{deg} for the VIODE dataset. This is defined as $r_{\text{deg}} = \text{ATE}_{\text{high}}/\text{ATE}_{\text{none}}$, where ATE_{high} and ATE_{none} are the ATE of the algorithm in the `high` and `none` sequences respectively. This metric illustrates the robustness of the VIO algorithm in dynamic sequences compared to that in static sequences. We calculate this value for all evaluated algorithms in each environment.

A.4.2 Evaluation of existing VIO algorithms

We apply ROVIO and VINS-Mono, two state-of-the-art VIO algorithms, on the VIODE dataset. Since the performance of these algorithms is not deterministic, each is evaluated ten times.

Our findings show that both ROVIO and VINS-Mono perform worse in the presence of dynamic objects. We observe that the ATE of both algorithms increases as the scene gets more dynamic (Fig. A.5). Furthermore, estimated trajectories illustrate the degradation of VINS-Mono in highly dynamic sequences. For example, in `parking_lot/high`, there is a drift around $(x, y) = (0, 8)$ while is not present in `parking_lot/none` (Fig. A.6). The degradation rate is mostly higher than 5.0 for ROVIO and VINS-Mono (Tab. A.1). Comparing the degradation rate of ROVIO and VINS-Mono among the three environments, `city_day` environment is the most challenging one among the three. The degradation rate of both ROVIO and VINS-Mono was the lowest in `city_night`, in which the average and maximum pixel-based dynamic rate were also the lowest (Fig. A.2).

We show that the wrong estimation often occurs when the dynamic objects are in field of view. One evidence is a correspondence between two types of dynamic rates (r_{pix} and r_{of}) and RPE estimated by VINS-Mono (Fig. A.8). For example, r_{pix}, r_{of}, and RPE mark the highest value between 40 and 50 s in `parking_lot/high`. These correspondences between RPE and two types of dynamic rate support the hypothesis that the performance degradation of current VIO algorithms is caused by dynamic objects.



(a) ROVIO (b) VINS-Mono (c) VINS-Mask (Ours)

Figure A.5 – Performance comparison of VIO algorithms in dynamic environments. We show the absolute trajectory error (ATE) of (a) ROVIO, (b) VINS-Mono, and (c) VINS-Mask for different dynamicity levels. The performance of these methods is not deterministic. Thus, we run the simulation ten times for every sequence. We observe an increase in errors as the dynamic level increases for ROVIO and VINS-Mono. The ATE of VINS-Mask is consistently lower than for the other two algorithms.

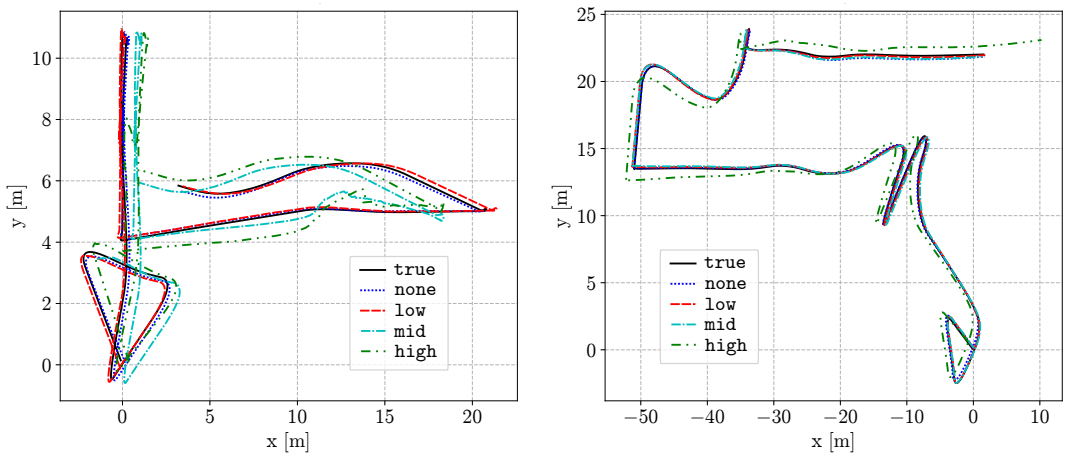
A.4.3 Evaluation of VINS-Mask

In addition to ROVIO and VINS-Mono, here we evaluate VINS-Mask on the VIODE dataset. VINS-Mask is also applied ten times for each sequence. We found that the performance of the VINS-Mask is almost independent of the dynamic level. The ATE of VINS-Mask is mostly lower than the results of ROVIO and VINS-Mono (Fig. A.5). The degradation rate of VINS-Mask is consistently lower than the other two algorithms (Tab. A.1). The estimated trajectories in `high` sequences for each algorithm also illustrates the improvement by VINS-Mask (Fig. A.7). While the existing algorithms exhibit some drifts in highly dynamic sequences, VINS-Mask successfully suppresses these drifts. It is also worth noting that VINS-Mask performs similarly with VINS-Mono in static sequences, in which VINS-Mask masks out static objects while VINS-Mono does not. The improvement by VINS-Mask suggests the usefulness of this type of approach as a technique to run VIO robustly in dynamic environments. Moreover, the results indicate that the performance degradation of the existing algorithms is due to the dynamic objects in the scene.

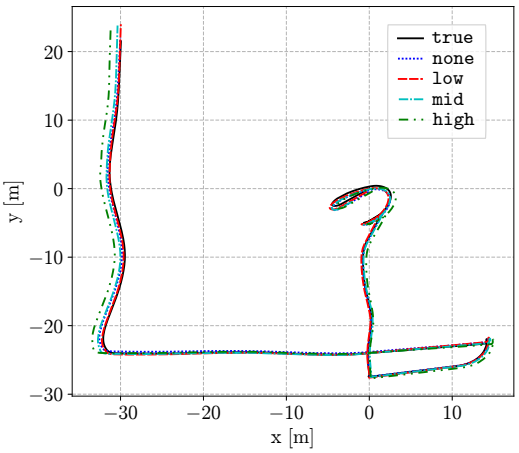
A.5 Conclusions

In this chapter, we proposed the VIODE dataset, a novel visual-inertial benchmark that contains variable and measurable dynamic events. Through the systematic introduction of dynamic objects, the users can isolate the effect of dynamic objects on the robustness of VIO. Using the VIODE dataset, we have shown that both VINS-Mono and ROVIO, the two state-of-the-art open-source VIO algorithms, perform worse as the scenes get more dynamic. We also demonstrated that the utilization of semantic information has the potential to overcome this degradation. By masking out the region of the objects, we could mitigate the impact of dynamic objects on the VIO algorithm.

As a future research direction, it is necessary to evaluate the VINS-Mask by using semantic



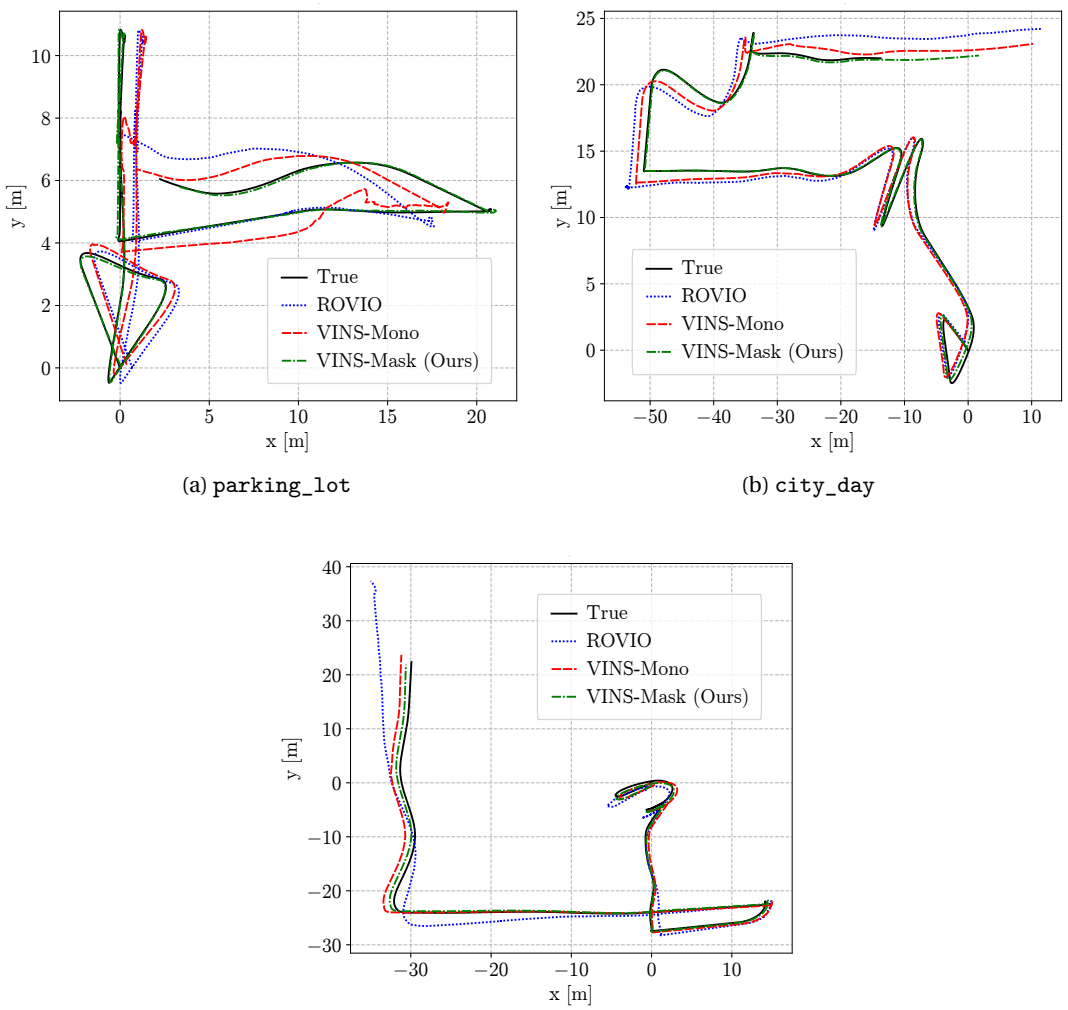
(a) `parking_lot` (b) `city_day`



(c) `city_night`

Figure A.6 – Estimated trajectories of VINS-Mono in each environment with different dynamicity levels: `none`, `low`, `mid`, and `high`.

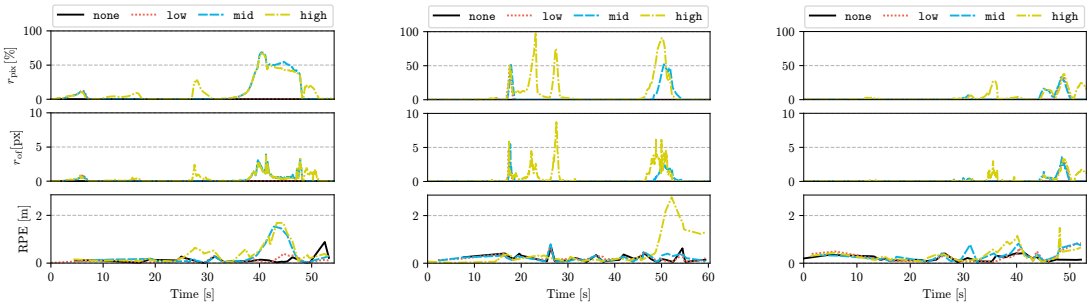
segmentation instead of ground-truth segmentation. One of the challenges would be a real-time onboard deployment, as latency can be critical for the performance of VIO. It is also important to investigate the better utilization of semantic segmentation for VIO. One of the directions would be to distinguish static objects from dynamic ones. Since the current VINS-Mask can mask out static objects such as parked vehicles, further research is still necessary for robust VIO in challenging dynamic scenes. As a future direction of the VIODE dataset, we are also interested in introducing other types of sensors in the dataset to enable the evaluation of localization algorithms with various sensor configurations. Another possible future work is, similarly to [163], to use real IMU and trajectory data recorded on a real drone platform instead of simulated ones.



(a) `parking_lot` (b) `city_day`

(c) `city_night`

Figure A.7 – Estimated trajectories of ROVIO, VINS-Mono, and VINS-Mask for sequences with `high` dynamicity level.



(a) `parking_lot` (b) `city_day` (c) `city_night`

Figure A.8 – Comparison of two types of dynamic rate and estimation performance of VINS-Mono over time axis for `none`, `low`, `mid`, and `high` sequences in each environment. The first row shows the pixel-based dynamic rate r_{pix} (Eq. A.1), the second row the OF-based dynamic rate r_{of} (Eq. A.2), and the last row shows the RPE (Eq. A.4).

B Open-source software

This thesis has lead to the development of several software packages. We briefly describe the purpose and content of these packages.

B.1 `vswarm`: vision-based flocking in simulation and reality

The `vswarm`[1] package — short for visual swarm — enables physics-based simulations and real-world experiments with quadcopters (Chapter 3, 4, and 5). It implements several computer vision algorithms such as generic object detection [95], marker-based detections [164], and background subtraction [94]. It also supports multi-target tracking with implementations of the linear, extended, and unscented versions of the GM-PHD filter [107]. It is mainly written in Python (and some C++) and is based on the ROS ecosystem [108]. The physics-based simulations rely on Gazebo [53], as well as sensors and models from RotorS [54] and PX4 [59]. The software stack can either be run in simulation (Fig. B.1) or on real hardware to synthesize vision-based swarms. It relies on PyTorch [85] for the neural network implementations and NumPy for numerical computations [165].

B.2 `vmodel`: agent-based simulations for swarms

The `vmodel`[2] package — short for visual model — enables agent-based swarm simulations and is used throughout the thesis (Chapter 3, 4, and 5). It implements point mass dynamics and several flocking algorithms such as Reynolds [64], Olfati-Saber [166, 71], Gregoire [167], Leonard [168], and Vásárhelyi [14] flocking. The package allows the evaluation of the flocking algorithms using several metrics (Sec. 5.2.1) such as order [14], union [67], and connectivity [16]. It supports common neighbor selection methods such as metric [64], topological [20], spatially-balanced [117], and Voronoi-based [121] strategies. It is written entirely in Python and supports live plotting for visualization purposes (Fig. B.2) based on Matplotlib [169]. It

[1] The `vswarm` package on GitHub: https://github.com/lis-epfl/vswarm.
[2] The `vmodel` package on GitHub: https://github.com/lis-epfl/vmodel.

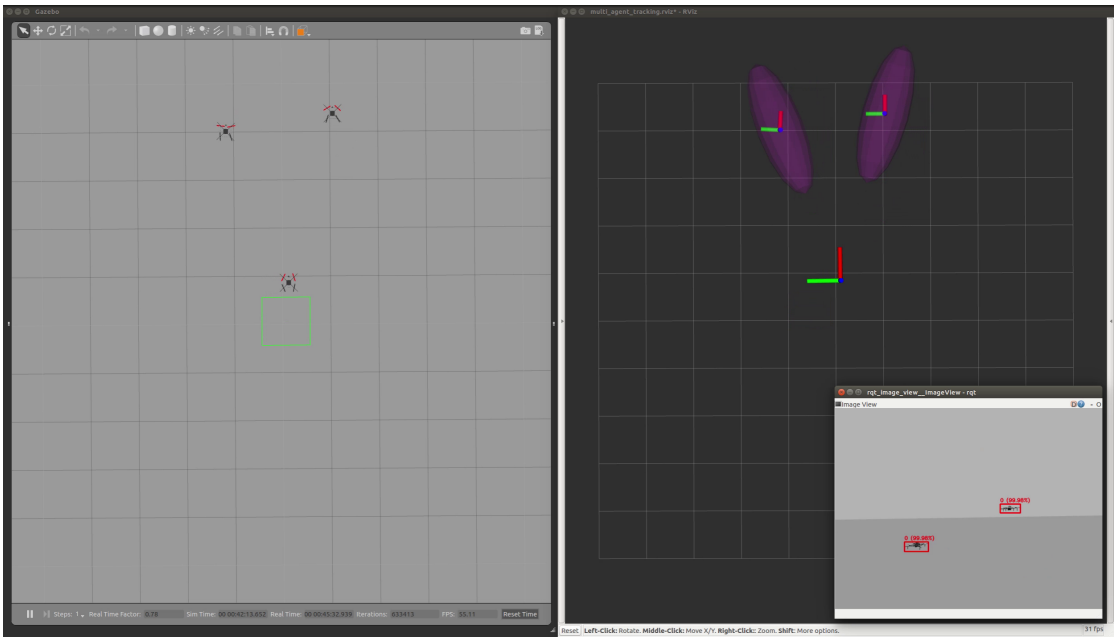


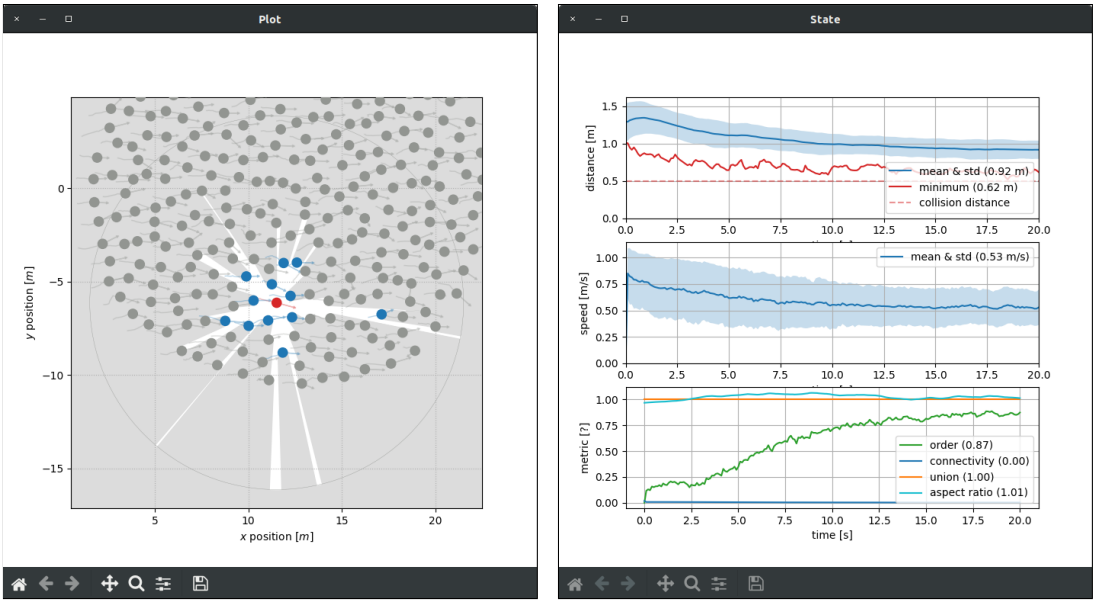
Figure B.1 – Screenshot of the `vswarm` simulation environment running the vision-based flocking algorithm with three agents. The screenshot contains a top view of the agents in the Gazebo simulator (left side), the same perspective from the RViz visualization tool (right side), and a live camera feed from the focal agent (bottom right window). We take the perspective of the focal agent (central drone in both halves) that estimates the positions and velocities of the two other drones. We show the position uncertainty of the other drones due to noise on range and bearing (pink ellipsoids, exaggerated for visualization purposes).

relies heavily on several libraries of the scientific Python ecosystem such as NumPy [165], SciPy [170], and IPython [171].

B.3 `openmv_cam`: ROS driver for the OpenMV Cam

The `openmv_cam`[3] package is a ROS driver for the OpenMV Cam[4], a low-powered and extensible machine vision camera. It implements a wrapper that acquires images over the USB interface and relays them as ROS (compressed) image messages.

[3] The `openmv_cam` package on GitHub: https://github.com/fabianschilling/openmv_cam.
[4] OpenMV: https://openmv.io/.



(a) Live view

(b) States and metrics

Figure B.2 – Screenshot of the `vmodel` simulator running a simulation with three hundred vision-based agents. The (a) live view screen shows the agents (their size, positions, and velocities) and highlights the perspective of a single focal agent. The (b) state screen shows statistics on inter-agent distances, speed, and several metrics.

C Publications

Articles published in peer-reviewed journals:

- F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," in *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4523–4530, Oct. 2019 [60].
- F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, Apr. 2021 [61].
- K. Minoda, F. Schilling, V. Wüest, D. Floreano, and T. Yairi, "VIODE: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1343–1350, Apr. 2021 [63].
- V. Ramachandran, F. Schilling, A. Wu, and D. Floreano, "Smart textiles that teach: Fabric-based haptic device improves the rate of motor learning," in *Advanced Intelligent Systems*, vol. X, no. X, pp. 2100043, Jul. 2021 [172].

Articles submitted to peer-reviewed journals:

- F. Schilling, E. Soria, and D. Floreano, "On the scalability of vision-based drone swarms in the presence of occlusions," in *IEEE Access*, vol. 1, no. 1, pp. 1–13, Aug. 2021 [62].

Bibliography

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[2] J.-C. Zufferey, Hauert, Sabine, Stirling, Timothy, Leven, Severin, Roberts, James, and Floreano, Dario, "Aerial Collective Systems," in *Handbook of Collective Robotics*, S. Kernbach, Ed. Pan Stanford, 2011, pp. 609–660.

[3] S. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics (T-RO)*, vol. 34, no. 4, pp. 837–855, 2018.

[4] G. Loianno and V. Kumar, "Cooperative Transportation Using Small Quadrotors Using Monocular Vision and Inertial Sensing," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 680–687, 2018.

[5] M. Varga, "Fixed-wing drones for communication networks," phdthesis, École polytechnique fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2016.

[6] S. Hauert, "Evolutionary Synthesis of Communication-Based Aerial Swarms," phdthesis, École polytechnique fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2010.

[7] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. H. E. de Croon, "A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints," *Frontiers in Robotics and AI*, vol. 7, p. 18, 2020.

[8] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[9] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3299–3304.

[10] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual Inertial Odometry Swarm: An Autonomous Swarm of Vision-Based Quadrotors," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1801–1807, 2018.

[11] E. Soria, F. Schiano, and D. Floreano, "Predictive control of aerial swarms in cluttered environments," *Nat. Mach. Intell.*, pp. 1–10, 2021.

[12] C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. Somorjai, T. Nepusz, and T. Vicsek, "Flocking algorithm for autonomous flying robots," *Bioinspiration & Biomimetics*, vol. 9, no. 2, p. 025012, 2014.

[13] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2014, pp. 3866–3873.

[14] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, p. eaat3536, 2018.

[15] K. N. McGuire, C. D. Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, p. eaaw9710, 2019.

[16] F. Schiano and P. R. Giordano, "Bearing rigidity maintenance for formations of quadrotor UAVs," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1467–1474.

[17] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, and D. Floreano, "Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2011, pp. 5015–5020.

[18] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-Efficient Decentralized Visual SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2466–2473.

[19] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic, "Empirical investigation of starling flocks: A benchmark study in collective animal behaviour," *Animal Behaviour*, vol. 76, no. 1, pp. 201–215, 2008.

[20] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic, "Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study," *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 105, no. 4, pp. 1232–1237, 2008.

[21] G. Dell'Ariccia, G. Dell'Omo, D. P. Wolfer, and H.-P. Lipp, "Flock flying improves pigeons' homing: GPS track analysis of individual flyers versus small groups," *Animal Behaviour*, vol. 76, no. 4, pp. 1165–1172, 2008.

[22] A. Strandburg-Peshkin, C. R. Twomey, N. W. F. Bode, A. B. Kao, Y. Katz, C. C. Ioannou, S. B. Rosenthal, C. J. Torney, H. S. Wu, S. A. Levin, and I. D. Couzin, "Visual sensory networks

and effective information transfer in animal groups," *Current Biology*, vol. 23, no. 17, pp. R709–R711, 2013.

[23] S. B. Rosenthal, C. R. Twomey, A. T. Hartnett, H. S. Wu, and I. D. Couzin, "Revealing the hidden networks of interaction in mobile animal groups allows prediction of complex behavioral contagion," *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 112, no. 15, pp. 4690–4695, 2015.

[24] J. D. Davidson, M. M. G. Sosna, C. R. Twomey, V. H. Sridhar, S. P. Leblanc, and I. D. Couzin, "Collective detection based on visual information in animal groups," *Journal of the Royal Society Interface*, vol. 18, no. 180, p. 20210142, 2021.

[25] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics (T-RO)*, vol. 34, no. 4, pp. 1004–1020, 2018.

[26] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting Flying Objects Using a Single Moving Camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 5, pp. 879–892, 2017.

[27] A. Rozantsev, "Vision-based detection of aircrafts and UAVs," phdthesis, École polytechnique fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2017.

[28] D. J. G. Pearce, A. M. Miller, G. Rowlands, and M. S. Turner, "Role of projection in the control of bird flocks," *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 111, no. 29, pp. 10 422–10 426, 2014.

[29] M. Basiri, F. Schill, D. Floreano, and P. U. Lima, "Audio-based localization for swarms of micro air vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4729–4734.

[30] M. Basiri, F. Schill, P. Lima, and D. Floreano, "Onboard Relative Bearing Estimation for Teams of Drones Using Sound," *IEEE Robotics and Automation Letters (RA-L)*, vol. 1, no. 2, pp. 820–827, 2016.

[31] M. Coppola, K. N. McGuire, K. Y. W. Scheper, and G. C. H. E. de Croon, "Onboard communication-based relative localization for collision avoidance in Micro Air Vehicle teams," *Autonomous Robots*, vol. 42, no. 8, pp. 1787–1805, 2018.

[32] S. van der Helm, M. Coppola, K. N. McGuire, and G. C. H. E. de Croon, "Onboard range-based relative localization for micro air vehicles in indoor leader–follower flight," *Autonomous Robots*, 2019.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[34] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 779–788.

[36] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: A review of recent research," *Adv. Robot.*, vol. 31, no. 16, pp. 821–835, 2017.

[37] J. M. Graving, D. Chae, H. Naik, L. Li, B. Koger, B. R. Costelloe, and I. D. Couzin, "DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning," vol. 8, p. e47994, 2019.

[38] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A Survey of Deep Learning-Based Object Detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019.

[39] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep Learning for Generic Object Detection: A Survey," *International Journal of Computer Vision (IJCV)*, 2019.

[40] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, no. 01, pp. 1–1, 2021.

[41] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2999–3007.

[42] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 4, pp. 834–848, 2018.

[43] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research (IJRR)*, vol. 37, no. 4-5, pp. 421–436, 2018.

[44] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[45] A. Giusti, J. Guzzi, D. C. Cireşan, F. L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters (RA-L)*, vol. 1, no. 2, pp. 661–667, 2016.

[46] F. J. H. Heras, F. Romero-Ferrero, R. C. Hinz, and G. G. de Polavieja, "Deep attention networks reveal the rules of collective motion in zebrafish," *PLOS Computational Biology*, vol. 15, no. 9, p. e1007354, 2019.

[47] B. T. Fine and D. A. Shell, "Unifying microscopic flocking motion models for virtual, robotic, and biological flock members," *Autonomous Robots*, vol. 35, no. 2, pp. 195–219, 2013.

[48] E. Soria, F. Schiano, and D. Floreano, "SwarmLab: A Matlab Drone Swarm Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8005–8011.

[49] M. Coppola, "Automatic Design of Verifiable Robot Swarms," phdthesis, Delft University of Technology, Delft, Netherlands, 2021.

[50] H. Hildenbrandt, C. Carere, and C. Hemelrijk, "Self-organized aerial displays of thousands of starlings: A model," *Behavioral Ecology*, vol. 21, no. 6, pp. 1349–1359, 2010.

[51] N. W. F. Bode, D. W. Franks, and A. J. Wood, "Limited interactions in flocks: Relating model simulations to empirical data," *J. R. Soc. Interface*, vol. 8, no. 55, pp. 301–304, 2011.

[52] R. Bastien and P. Romanczuk, "A model of collective behavior based purely on vision," *Science Advances*, vol. 6, no. 6, p. eaay0792, 2020.

[53] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE/RSJ, 2004, pp. 2149–2154 vol.3.

[54] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS: A Modular Gazebo MAV Simulator Framework," in *Robot Operating System (ROS)*, ser. Studies in Computational Intelligence. Springer, Cham, 2016, pp. 595–625.

[55] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *IEEE Transactions on Robotics (T-RO)*, 2020.

[56] N. Dousse, "Aerial Human-Comfortable Collision-free Navigation in Dense Environments," phdthesis, École polytechnique fédérale de Lausanne (EPFL), Lausanne, 2017.

[57] J. Lecoeur, "Insect-Inspired Visual Perception for Flight Control and Collision Avoidance," phdthesis, École polytechnique fédérale de Lausanne (EPFL), 2019.

[58] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[59] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6235–6240.

[60] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning Vision-based Flight in Drone Swarms by Imitation," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 4523–4530, 2019.

[61] F. Schilling, F. Schiano, and D. Floreano, "Vision-Based Drone Flocking in Outdoor Environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2954–2961, 2021.

[62] F. Schilling, E. Soria, and D. Floreano, "On the Scalability of Vision-based Drone Swarms in the Presence of Occlusions," *IEEE Access*, vol. X, no. X, p. (submitted), 2021.

[63] K. Minoda, F. Schilling, V. Wüest, D. Floreano, and T. Yairi, "VIODE: A Simulated Dataset to Address the Challenges of Visual-Inertial Odometry in Dynamic Environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 1343–1350, 2021.

[64] C. W. Reynolds, "Flocks, Herds and Schools: A Distributed Behavioral Model," in *ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, vol. 14. ACM, 1987, pp. 25–34.

[65] H. Kunz and C. K. Hemelrijk, "Simulations of the social organization of large schools of fish whose perception is obstructed," *Applied Animal Behavior Science*, vol. 138, no. 3, pp. 142–151, 2012.

[66] J. Holland and S. K. Semwal, "Flocking Boids with Geometric Vision, Perception and Recognition," in *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2009.

[67] E. Soria, F. Schiano, and D. Floreano, "The Influence of Limited Visual Sensing on the Reynolds Flocking Algorithm," in *IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 138–145.

[68] G. R. Martin, "Visual fields and their functions in birds," *Journal of Ornothology*, vol. 148, no. S2, pp. 547–562, 2007.

[69] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Transactions on Robotics and Automation (T-RA)*, vol. 18, no. 5, pp. 813–825, 2002.

[70] W. Gao, K. Wang, W. Ding, F. Gao, T. Qin, and S. Shen, "Autonomous aerial robot using dual-fisheye cameras," *J. Field Robot.*, vol. n/a, no. n/a, 2020.

[71] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[72] U. Mehmood, N. Paoletti, D. Phan, R. Grosu, S. Lin, S. D. Stoller, A. Tiwari, J. Yang, and S. A. Smolka, "Declarative vs Rule-based Control for Flocking Dynamics," in *ACM/SIGAPP Symposium on Applied Computing (SAC)*, ser. SAC '18. ACM, 2018, pp. 816–823.

[73] N. Dousse, G. Heitz, F. Schill, and D. Floreano, "Human-Comfortable Collision-Free Navigation for Personal Aerial Vehicles," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 1, pp. 358–365, 2017.

[74] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning Monocular Reactive UAV Control in Cluttered Natural Environments," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 1765–1772.

[75] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 1088–1095, 2018.

[76] F. Sadeghi and S. Levine, "CAD2RL: Real Single-Image Flight without a Single Real Image," in *Robotics: Science and Systems (RSS)*, vol. 13, 2017.

[77] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4241–4247.

[78] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar, "System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization," *Autonomous Robots*, vol. 41, no. 4, pp. 919–944, 2017.

[79] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, and M. Mejail, "A Practical Multirobot Localization System," *Journal of Intelligent and Robotic Systems*, vol. 76, no. 3-4, pp. 539–562, 2014.

[80] V. Walter, N. Staub, A. Franchi, and M. Saska, "UVDAR System for Visual Relative Localization with application to Leader-Follower Formations of Multirotor UAVs," *IEEE Robotics and Automation Letters (RA-L)*, pp. 1–1, 2019.

[81] M. Saska, J. Vakula, and L. Přeućil, "Swarms of micro aerial vehicles stabilized under a visual relative localization," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3570–3575.

[82] S. Ross, G. Gordon, and D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 14. JMLR.org, 2011, pp. 627–635.

[83] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[84] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, 2014.

[85] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, and Z. Lin, "Automatic differentiation in PyTorch," in *Conference on Neural Information Processing Systems (NIPS)*, 2017, p. 4.

[86] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[87] Z. Wang and J. Yang, "Diabetic Retinopathy Detection via Deep Convolutional Networks for Discriminative Localization and Visual Explanation," in *Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2018, p. 8.

[88] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme, "Downwash-aware trajectory planning for large quadrotor teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 250–257.

[89] S. Roelofsen, D. Gillet, and A. Martinoli, "Reciprocal Collision Avoidance For Quadrotors Using On-board Visual Detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2015, pp. 4810–4817.

[90] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit, D. Gillet, P. Fua, and A. Martinoli, "Vision-based Unmanned Aerial Vehicle Detection and Tracking for Sense and Avoid Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2016, pp. 1556–1561.

[91] M. Vrba and M. Saska, "Marker-Less Micro Aerial Vehicle Detection and Localization Using Convolutional Neural Networks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 2459–2466, 2020.

[92] R. Opromolla, G. Fasano, and D. Accardo, "A Vision-Based Approach to UAV Detection and Tracking in Cooperative Applications," *Sensors (Basel)*, vol. 18, no. 10, 2018.

[93] P. M. Wyder, Y.-S. Chen, A. J. Lasrado, R. J. Pelles, R. Kwiatkowski, E. O. A. Comas, R. Kennedy, A. Mangla, Z. Huang, X. Hu, Z. Xiong, T. Aharoni, T.-C. Chuang, and H. Lipson, "Autonomous drone hunter operating by deep learning and all-onboard computations in GPS-denied environments," *PLOS ONE*, vol. 14, no. 11, p. e0225092, 2019.

[94] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[95] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv*, 2018.

[96] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 658–666.

[97] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.

[98] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. (2020) YOLOv4: Optimal Speed and Accuracy of Object Detection.

[99] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *IEEE European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science. Springer, Cham, 2014, pp. 740–755.

[100] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," *ArXiv*, 2016.

[101] M. Pfister, "Real-time Monocular Detection of Drones," Lausanne, Switzerland, 2020.

[102] A. Rozantsev, V. Lepetit, and P. Fua, "Flying objects detection from a single moving camera," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4128–4136.

[103] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable Convolutional Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 764–773.

[104] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 1280–1286.

[105] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 28, no. 8, pp. 1335–1340, 2006.

[106] V. Usenko, N. Demmel, and D. Cremers, "The Double Sphere Camera Model," in *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 552–560.

[107] B. Vo and W. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.

[108] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS - an open-source Robot Operating System," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, p. 6.

[109] Y. Tang, Y. Hu, J. Cui, F. Liao, M. Lao, F. Lin, and R. Teo, "Vision-aided Multi-UAV Autonomous Flocking in GPS-denied Environment," *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2018.

[110] P. Petráček, V. Walter, T. Báča, and M. Saska, "Bio-inspired compact swarms of unmanned aerial vehicles without communication and external localization," *Bioinspiration & Biomimetics*, vol. 16, no. 2, p. 026009, 2020.

[111] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 2, pp. 307–321, 2015.

[112] J. Hu, A. E. Turgut, T. Krajník, B. Lennox, and F. Arvin, "Occlusion-Based Coordination Protocol Design for Autonomous Robotic Shepherding Tasks," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2020.

[113] T. Vicsek and A. Zafeiris, "Collective motion," *Physics Reports*, vol. 517, no. 3-4, pp. 71–140, 2012.

[114] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective Memory and Spatial Sorting in Animal Groups," *Journal of Theoretical Biology*, vol. 218, no. 1, pp. 1–11, 2002.

[115] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[116] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 97–120, 2008.

[117] M. Camperi, A. Cavagna, I. Giardina, G. Parisi, and E. Silvestri, "Spatially balanced topological interaction grants optimal cohesion in flocking models," *Interface Focus*, vol. 2, no. 6, pp. 715–725, 2012.

[118] M. Lindhe, P. Ogren, and K. Johansson, "Flocking with Obstacle Avoidance: A New Distributed Coordination Algorithm Based on Voronoi Partitions," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 1785–1790.

[119] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stable flocking of mobile agents part II: Dynamic topology," in *IEEE International Conference on Decision and Control (CDC)*, vol. 2, 2003, pp. 2016–2021 Vol.2.

[120] Y. Shang and R. Bouffanais, "Influence of the number of topologically interacting neighbors on swarm dynamics," *Scientific Reports*, vol. 4, no. 1, p. 4184, 2014.

[121] A. Okabe, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed., ser. Wiley Series in Probability and Statistics. Wiley, 2000.

[122] D. Dias, R. Ventura, P. Lima, and A. Martinoli, "Onboard vision-based 3D relative localization system for multiple quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1181–1187.

[123] M. Vrba, D. Heřt, and M. Saska, "Onboard Marker-Less Detection and Localization of Non-Cooperating Drones for Their Safe Interception by an Autonomous Aerial System," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3402–3409, 2019.

[124] P. Petracek, V. Kratky, M. Petrlik, T. Baca, R. Kratochvil, and M. Saska, "Large-Scale Exploration of Cave Environments by Unmanned Aerial Vehicles," *IEEE Robotics and Automation Letters (RA-L)*, pp. 1–1, 2021.

[125] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A monocular pose estimation system based on infrared LEDs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 907–913.

[126] M. Varga, M. Basiri, G. Heitz, and D. Floreano, "Distributed formation control of fixed wing micro aerial vehicles for area coverage," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 669–674.

[127] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 604–611, 2020.

[128] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier, "Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments," *IEEE Robotics and Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[129] P. Schmuck and M. Chli, "Multi-UAV collaborative monocular SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3863–3870.

[130] B. T. Fine and D. A. Shell, "Flocking: Don't need no stinkin' robot recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 5001–5006.

[131] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for Deep Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[132] B. Zhou, P. Krähenbühl, and V. Koltun, "Does computer vision matter for action?" *Science Robotics*, vol. 4, no. 30, p. eaaw6661, 2019.

[133] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[134] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[135] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1736–1741.

[136] X. Mu, B. He, X. Zhang, T. Yan, X. Chen, and R. Dong, "Visual Navigation Features Selection Algorithm Based on Instance Segmentation in Dynamic Environment," *IEEE Access*, vol. 8, pp. 465–473, 2020.

[137] X. Bai, B. Zhang, W. Wen, L.-T. Hsu, and H. Li, "Perception-aided Visual-Inertial Integrated Positioning in Dynamic Urban Areas," in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2020, pp. 1563–1571.

[138] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Conference on Field and Service Robotics (FSR)*, ser. Springer Proceedings in Advanced Robotics, M. Hutter and R. Siegwart, Eds. Springer International Publishing, 2018, pp. 621–635.

[139] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 10, pp. 1157–1163, 2016.

[140] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1680–1687.

[141] A. L. Majdik, C. Till, and D. Scaramuzza, "The Zurich urban micro aerial vehicle dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 3, pp. 269–273, 2017.

[142] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013.

[143] A. R. Gaspar, A. Nunes, A. M. Pinto, and A. Matos, "Urban@CRAS dataset: Benchmarking of visual odometry and SLAM techniques," *Robotics and Autonomous Systems*, vol. 109, pp. 59–67, 2018.

[144] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research (IJRR)*, vol. 38, no. 6, pp. 642–657, 2019.

[145] K. M. Judd and J. D. Gammell, "The Oxford Multimotion Dataset: Multiple SE(3) Motions With Ground Truth," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 800–807, 2019.

[146] S. Cortés, A. Solin, E. Rahtu, and J. Kannala, "ADVIO: An Authentic Dataset for Visual-Inertial Odometry," in *IEEE European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Springer International Publishing, 2018, pp. 425–440.

[147] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust Visual Inertial Odometry Using a Direct EKF-Based Approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2015, pp. 298–304.

[148] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2007, pp. 3565–3572.

[149] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research (IJRR)*, vol. 34, no. 3, pp. 314–334, 2015.

[150] T. Qin, J. Pan, S. Cao, and S. Shen. (2019) A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors.

[151] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós. (2020) ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM.

[152] J. Delmerico and D. Scaramuzza, "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, p. 8.

[153] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[154] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa, "Mask-SLAM: Robust Feature-Based Monocular SLAM by Masking Using Semantic Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 258–266.

[155] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 4076–4083, 2018.

[156] B. Bescos, C. Cadena, and J. Neira, "Empty Cities: A Dynamic-Object-Invariant Space for Visual SLAM," *IEEE Transactions on Robotics (T-RO)*, vol. 37, no. 2, pp. 433–451, 2021.

[157] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-Coupled Multi-Object Tracking and SLAM," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 3, pp. 5191–5198, 2021.

[158] K. Qiu, T. Qin, W. Gao, and S. Shen, "Tracking 3-D Motion of Dynamic Objects Using Monocular Visual-Inertial Sensing," *IEEE Transactions on Robotics (T-RO)*, vol. 35, no. 4, pp. 799–816, 2019.

[159] R. Sabzevari and D. Scaramuzza, "Multi-body Motion Estimation from Monocular Vehicle-Mounted Cameras," *IEEE Transactions on Robotics (T-RO)*, vol. 32, no. 3, pp. 638–651, 2016.

[160] K. M. Judd, J. D. Gammell, and P. Newman, "Multimotion Visual Odometry (MVO): Simultaneous Estimation of Camera and Third-Party Motions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3949–3956.

[161] O. J. Woodman, "An introduction to inertial navigation," Cambridge, UK, p. 37, 2007.

[162] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[163] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman, "The Blackbird UAV dataset," *The International Journal of Robotics Research (IJRR)*, vol. 39, no. 10-11, pp. 1346–1364, 2020.

[164] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3400–3407.

[165] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[166] R. Olfati-Saber, "A Unified Analytical Look at Reynolds Flocking Rules," Fort Belvoir, VA, 2003.

[167] G. Grégoire, H. Chaté, and Y. Tu, "Moving and staying together without a leader," *Physica D: Nonlinear Phenomena*, vol. 181, no. 3, pp. 157–170, 2003.

[168] N. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *IEEE International Conference on Decision and Control (CDC)*, vol. 3, 2001, pp. 2968–2973 vol.3.

[169] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

[170] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey,

I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[171] F. Perez and B. E. Granger, "IPython: A System for Interactive Scientific Computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 21–29, 2007.

[172] V. Ramachandran, F. Schilling, A. R. Wu, and D. Floreano, "Smart Textiles that Teach: Fabric-Based Haptic Device Improves the Rate of Motor Learning," *Advanced Intelligent Systems*, vol. X, no. X, p. 2100043, 2021.

Fabian Schilling

Ph.D. Student in Robotics, Control, and Intelligent Systems · M.Sc. in Computer Science

Born: 14 Jun '91 in Munich, DE · Address: Avenue de Sévelin 13B, 1004 Lausanne, CH · Nationality: German

+41 77 968 90 70 | fabian@schilli.ng | fabianschilling.org | fabianschilling | fabianschilling

Experience

Laboratory of Intelligent Systems (LIS) at EPFL
Lausanne, CH

Doctoral Assistant — Sep '17 - present

- Conducted research on vision-based control for aerial robot swarms using simulation and real-world experiments.
- Implemented imitation learning, object detection, and multi-target tracking methods in ROS, PyTorch/NumPy, and PX4.
- Developed course materials and held lab exercises in Master-level courses on aerial robots and motion capture.
- Supervised student projects in vision-based localization, real-time object detection, omnidirectional imaging, etc.
- Presented at several conferences & workshops, held poster presentations, and contributed to grant proposals.

Robotics, Perception, and Learning Laboratory (RPL) at KTH
Stockholm, SE

Research Engineer — Jul '16 - Mar '17

- Conducted research on terrain classification methods for autonomous ground robot navigation.
- Implemented real-time semantic segmentation system and transfer learning method using TensorFlow & ROS.
- Prepared live demo for review meeting, integrated with mapping framework and motion planner on new robot.
- Wrote deliverable reports, attended consortium meetings, co-authored article about terrain classification system.

Royal Institute of Technology (KTH)
Stockholm, SE

Teaching Assistant — Jul '15 - Oct '16

- Conducted oral exams and help sessions about AI concepts, e.g. heuristic search and hidden Markov models.
- Developed and administered course portal for quizzes, assignment submissions, and presentation slot booking.

Greenely AB
Stockholm, SE

Software Developer — Mar '15 - Apr '15

- Developed iOS app for energy consumption visualization and energy efficiency recommendations using Swift.
- Designed user on-boarding and multi-stage sign-up process; populated diagrams and graphs using RESTful APIs.

Contigua GmbH
Munich, DE

Software Developer — Nov '13 - Mar '14

- Developed iOS app for mobile payments, virtual receipts, and point-of-sale rewards using Objective-C.
- Implemented credit card detection and optical character recognition with OpenCV and Tesseract OCR.

Hoffmann & Schilling Marketing GbR
Neuried, DE

Co-founder and Web Developer — Sep '10 - Dec '13

- Designed and developed websites, corporate identities, business cards, and brochures for several small businesses.

Neumann-Deutschmann-Schilling GbR
Munich, DE

Co-founder and Project Manager — Jan '12 - Jul '12

- Led communications project with focus on public relations and social media for the TU Munich math faculty.

Education

Swiss Federal Institute of Technology in Lausanne (EPFL)
Lausanne, CH

Ph.D. in Robotics, Control, and Intelligent Systems — Sep '17 - present

- Focus on machine learning, state estimation, and modeling for vision-based control of multi-robot systems.
- Ph.D. thesis (preliminary title): "Vision-based flocking in aerial robot swarms."

Swiss Federal Institute of Technology in Zürich (ETH)
Zürich, CH

Pre-doc Summer School on Learning Systems — Jul '17

- Lectures and workshops on deep learning for NLP, optimization methods, and visual localization.

Royal Institute of Technology (KTH) — Stockholm, SE

M.Sc. in Computer Science — Aug '14 - Apr '17

- Specialization in autonomous systems: machine learning, computer vision, robotics, and artificial intelligence.
- M.Sc. thesis: "The effect of batch normalization on deep convolutional neural networks".

University of California, Santa Barbara (UCSB) — Santa Barbara, CA, USA

Exchange studies in Computer Science — Sep '12 - Jun '13

- Courses in computer vision, augmented reality, network computing, database systems, and statistics.

Ludwig Maximilian University of Munich (LMU) — Munich, DE

B.Sc. in Computer Science — Oct '10 - Jul '14

- Focus on mobile and distributed systems. B.Sc. thesis: "Tracking smartphones using 802.11 management frames".

Volunteering

Academy Consult e.V. (Junior Enterprise) — Munich, DE

Junior Consultant — May '11 - Mar '13

- Organized and conducted seminars on presentation skills, customer acquisition, and project management.
- Participated in pro bono consultant project about strategic faculty communications at TU Munich (TUM).

Scholarships

PROSA LMU	Exchange scholarship for studies at UCSB funded by DAAD and the Bavarian state.	Aug '12 - Jun '13
Hans Rudolf	Financial support for studies at LMU donated by the Hans-Rudolf foundation.	Oct '12 - Oct '13

Grants

NVIDIA hardware grant	Donation of Tesla K40c GPU for Master thesis about convolutional neural networks.	Mar '16

Publications

F. Schilling, E. Soria, and D. Floreano, "On the scalability of vision-based drone swarms in the presence of occlusions," in *IEEE Access*, (**submitted**) Aug '21.

V. Ramachandran, **F. Schilling**, A. Wu, and D. Floreano, "Smart textiles that teach: Fabric-based haptic device improves the rate of motor learning," in *Advanced Intelligent Systems*, p. 2100043, Jul '21.

F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," in *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2954–2961, Apr '21.

K. Minoda, **F. Schilling**, V. Wüest, D. Floreano, and T. Yairi, "VIODE: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments," in *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 1343–1350, Apr '21.

F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," in *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 4523–4530, Oct '19.

F. Schilling, X. Chen, J. Folkesson, and P. Jensfelt, "Geometric and visual terrain classification for autonomous mobile navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2678–2684, Sep '17.

F. Schilling, "The effect of batch normalization on deep convolutional neural networks", Master thesis, *KTH Royal Institute of Technology, Stockholm*, Aug '16

Skills & activities

Natural languages	German (native) · English (fluent) · French (advanced) · Spanish (basics) · Dutch (basics)
Programming languages	Python · C++ · Swift · Java · Lua · MATLAB · Objective-C · C · Javascript · Latex
Frameworks & libraries	ROS · PyTorch · PX4 · OpenCV · TensorFlow/Keras · PCL · NumPy · Matplotlib · sklearn
Development tools	VSCode · Vim · Git · Zsh · Tmux · Jupyter · Gazebo · Bash · Docker · Xcode · Eclipse
Operating systems	GNU/Linux (Debian-based distributions such as Ubuntu) · macOS · iOS · Unix (FreeBSD)
Creative tools	Adobe Illustrator and Photoshop · Apple Final Cut Pro · Blender · draw.io · Omnigraffle
Sports	Surfing · skiing/snowboarding · paragliding · paddleboarding · volleyball · cycling · hiking
Hobbies	Traveling · music & podcasts · self-hosting · cooking · camping · reading · programming