Motion Planning in Dynamic Environments with Learned Scalable Policies

Serra Gomez, A.

DOI
10.4233/uuid:91cac381-d47f-4333-a7d0-b2336cf613a7

Publication date
2025

Document Version
Final published version

Citation (APA)
Serra Gomez, A. (2025). *Motion Planning in Dynamic Environments with Learned Scalable Policies*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:91cac381-d47f-4333-a7d0-b2336cf613a7

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Motion Planning in Dynamic Environments with Learned Scalable Policies

Motion Planning in Dynamic Environments with Learned Scalable Policies

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended publicly on
Monday the 10th of February 2025 at 10:00 o'clock.

by

Álvaro SERRA GOMEZ

Master of Science in Appled Mathematics and Computer Science,
École Polytechnique, France,
Master's degree in Automatic Control and Robotics,
Technical University of Catalonia, Spain,
born in Barcelona, Spain.

This dissertation has been approved by the promotors.

promotor: Dr. J. Alonso-Mora
copromotor: Dr. J. W. Böhmer

Composition of the doctoral committee:

Rector Magnificus,	Chairperson
Dr. J. Alonso-Mora	Delft University of Technology, promotor
Dr. J. W. Böhmer	Delft University of Technology, copromotor
Independent members:	
Prof. Dr. G. C. H. E de Croon	Delft University of Technology
Prof. Dr. S. Leutenegger	Technical University of Munich
Prof. Dr. A. Prorok	University of Cambridge
Prof. Dr. M. T. J. Spaan	Delft University of Technology
Dr. Ing. J. Kober	Delft University of Technology, *reserve member*

Keywords: Deep Reinforcement Learning, Attention-based Architectures, Multi-Robot Systems, Active Classification

Printed by: Gildeprint

Cover: Created by Alvaro Serra and image generation tools.

ISBN 978-94-6384-724-7

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

To become a good professional you must have a complete life. None of your accomplishments are important without your family and life outside of work.

Javier Serra-Aracil (Dad)

Contents

Summary

The application of multi-robot systems has gained popularity in recent years. Multi-robot systems show great potential in scaling up robotic applications in surveillance, monitoring, and exploration. Although single robots can already be used to automatize search and rescue, and surveillance tasks, their capability to solve their given task is still dependent on the region of interest size. For example, in expansive environments, a single robot's ability to cover or surveil the entire area effectively diminishes, resulting in information gaps. To this end, this thesis aims to improve multi-robot coordination and navigation in active perception tasks, e.g. exploration and surveillance. The multi-robot coordination problem aims to determine when and with whom multiple robots should exchange information to avoid collisions while navigating to their goal. The active perception problem involves guiding a sensing robot to positions rich in task-relevant information. The primary objective of this dissertation is to provide algorithmic contributions, specifically focusing on the obtention of robust policies that can adapt and scale to fleets and tasks of varying sizes.

To this end, this thesis leverages the combination of high-level policies, learned through Reinforcement Learning, with robust low-level optimal control policies. Firstly, the communication-based coordination problem for decentralized multi-robot systems is formally defined and modeled as a Markov Decision Process. This thesis then proposes a novel communication policy for decentralized multi-robot systems, improving coordination and collision avoidance. The proposed policy, learned via Reinforcement Learning, allows robots to selectively communicate, requesting trajectory plans from potential risks while assuming constant velocities for others. This utilizes an attention-based neural network architecture for scalability, integrated with Non-Linear Model Predictive Control for safe and robust motion planning. When tested with 12 robots, it reduced communications compared to alternatives, maintaining safety. Scalable and robust, it performed well with different team sizes and in the presence of observation noise. Real-world tests on quadrotors confirmed its practical applicability.

The Active Perception problem represents the second major challenge addressed in this dissertation. Specifically, this thesis addresses the challenge of using a drone to collect semantic information for classifying multiple moving targets, focusing on computing control inputs for optimal viewpoints. This task is complicated by the variable amount of targets to classify in the region of interest, and the use of a "black-box" classifier, like a deep learning neural network, which lacks clear analytical relationships between viewpoints and outputs. This thesis proposes an attention-based architecture, trained with Reinforcement Learning (RL), which determines the best viewpoints for the drone to gather evidence from multiple unclassified targets, considering their movement, orientation, and potential occlusions. A low-level MPC controller then guides the drone to these viewpoints. The approach outperforms various baselines and shows adaptability to new scenarios and scalability to

numerous targets with varying movement dynamics.

To conclude the thesis, the previous approach is extended to more realistic and larger environments where targets need to be localized, tracked, and then classified. This thesis introduces a novel decentralized hybrid multi-camera system designed for surveillance and monitoring applications. Traditional fixed camera networks suffer from blind spots and backlighting issues. A decentralized hybrid framework is proposed that integrates both static and mobile cameras to actively and dynamically enhance critical information gathering. All networked cameras collaborate to monitor and localize people in the environment by comparing their local information. The mobile camera is guided by a viewpoint control policy to maximize semantic information from observed targets. The framework was implemented in a photorealistic environment using Unreal Engine and enabled distributed communications through the Robot Operating System (ROS), bridging the gap between simulation and real-world applications. Results in large environments demonstrate the advantages of collaborative mobile cameras over static and individual setups both in target identification and tracking accuracy, respectively. In crowded scenarios, mobile cameras excel in avoiding occlusions and capturing desired viewpoints, improving the percentage of classified tracked targets compared to static setups. Qualitatively, mobile cameras provide superior target observation quality unmatched by the static framework.

In summary, this thesis makes significant contributions that are validated through extensive evaluations in simulated photo-realistic environments and with commercial drones, demonstrating the potential for practical applications. Despite the progress, the thesis acknowledges the remaining challenges in deploying multi-robot systems in real-world perception tasks, especially when the policy is learned, and suggests directions for future research.

Samenvatting

De toepassing van multi-robotsystemen heeft de laatste jaren aan populariteit gewonnen. Multi-robotsystemen tonen veel potentie in het opschalen van robottoepassingen op het gebied van bewaking, monitoring en verkenning. Hoewel enkele robots al kunnen worden gebruikt om zoek- en reddingsacties en bewakingstaken te automatiseren, is hun vermogen om hun gegeven taak op te lossen nog steeds afhankelijk van de grootte van het interessegebied. Bijvoorbeeld, in uitgestrekte omgevingen neemt het vermogen van een enkele robot om het hele gebied effectief te dekken of te bewaken af, wat resulteert in gemiste informatie. Om deze reden streeft dit proefschrift ernaar de coördinatie en navigatie van een multi-robot team te verbeteren in actieve perceptietaken, zoals verkenning en bewaking. Het multi-robot coördinatieprobleem beoogt te bepalen wanneer en met wie de robots informatie moeten uitwisselen om botsingen te vermijden terwijl ze naar hun doel navigeren. Het probleem van actieve perceptie houdt in dat een sensorrobot wordt geleid naar posities rijk aan taakrelevante informatie. Het primaire doel van dit proefschrift is om algoritmische bijdragen te leveren, specifiek gericht op het verkrijgen van robuuste strategieën die zich kunnen aanpassen en opschalen naar vloten en taken van verschillende groottes.

In dit kader maakt dit proefschrift gebruik van de combinatie van hoogwaardige strategieën, geleerd via Reinforcement Learning, met robuuste laagwaardige optimale strategieën. Ten eerste wordt het communicatie-gebaseerde coördinatieprobleem voor gedecentraliseerde multi-robotsystemen formeel gedefinieerd en gemodelleerd als een Markov-beslissingsproces. Vervolgens stelt dit proefschrift een nieuw communicatiestrategie voor voor gedecentraliseerde multi-robotsystemen, dat de coördinatie en het vermijden van botsingen verbetert. De voorgestelde strategie, geleerd via Reinforcement Learning, stelt robots in staat selectief te communiceren, waarbij ze trajectplannen aanvragen van potentiële risico's terwijl ze constante snelheden aannemen voor anderen. Dit maakt gebruik van een ättention-gebaseerd neurale netwerkarchitectuur voor schaalbaarheid, geïntegreerd met niet-lineaire model voorspellende controle voor veilige en robuuste bewegingsplanning. Getest met 12 robots, verminderde het de communicatie vergeleken met alternatieven, terwijl de veiligheid behouden bleef. Schaalbaar en robuust, presteerde het goed met verschillende teamgroottes en in de aanwezigheid van observatieruis. Praktijktests met quadrotors bevestigden de praktische toepasbaarheid ervan.

Het probleem van actieve perceptie vertegenwoordigt de tweede grote uitdaging die in dit proefschrift wordt aangepakt. Dit proefschrift richt zich specifiek op de uitdaging van het gebruik van een drone om semantische informatie te verzamelen voor het classificeren van meerdere bewegende doelen, met de focus op het berekenen van besturingssignalen voor optimale gezichtspunten. Deze taak wordt gecompliceerd door de variabele hoeveelheid doelen die in het interessegebied moeten worden geclassificeerd,

en het gebruik van een "black-box"classificator, zoals een diep neuraal netwerk, dat geen duidelijke analytische relaties heeft tussen gezichtspunten en uitvoer. Dit proefschrift stelt een ättention-gebaseerde architectuur voor, getraind met Reinforcement Learning (RL), die de beste gezichtspunten voor de drone bepaalt om bewijsmateriaal te verzamelen van meerdere niet-geclassificeerde doelen, waarbij rekening wordt gehouden met hun beweging, oriëntatie en mogelijke occlusies. Een laagwaardige MPC-besturingsalgoritme leidt vervolgens de drone naar deze gezichtspunten. De aanpak presteerde beter dan verschillende baselines en toonde aanpassingsvermogen aan nieuwe scenario's en schaalbaarheid naar talrijke doelen met verschillende bewegingsdynamiek.

Als laatste onderdeel in dit proefschrift wordt de voorgaande aanpak uitgebreid naar realistischere en grotere omgevingen waar doelen moeten worden gelokaliseerd, gevolgd en vervolgens geclassificeerd. Dit proefschrift introduceert een nieuw gedecentraliseerd hybride multi-camerasysteem ontworpen voor bewakings- en monitoringtoepassingen. Traditionele vaste cameranetwerken hebben last van dode hoeken en tegenlichtproblemen. Er wordt een gedecentraliseerd hybride raamwerk voorgesteld dat zowel statische als mobiele camera's integreert om kritische informatie actief en dynamisch te verbeteren. Alle camera's in het netwerk werken samen om mensen in de omgeving te monitoren en lokaliseren door hun lokale informatie te vergelijken. De mobiele camera wordt geleid door een gezichtspuntbesturingsstrategie om maximale semantische informatie uit waargenomen doelen te halen. Het raamwerk werd geïmplementeerd in een fotorealistische omgeving met Unreal Engine en maakte gedistribueerde communicatie mogelijk via het Robot Operating System (ROS), waarbij de kloof tussen simulatie en praktische toepassingen werd overbrugd. Resultaten in grote omgevingen tonen de voordelen van samenwerkende mobiele camera's boven statische en individuele opstellingen aan, zowel in doelidentificatie als in trackingnauwkeurigheid. In drukke scenario's excelleren mobiele camera's in het vermijden van occlusies en het vastleggen van gewenste gezichtspunten, waardoor het percentage geclassificeerde gevolgde doelen verbetert in vergelijking met statische opstellingen. Kwalitatief bieden mobiele camera's een superieure doelobservatiekwaliteit die ongeëvenaard is door het statische raamwerk.

Samenvattend maakt dit proefschriftsignificante bijdragen die zijn gevalideerd door uitgebreide evaluaties in gesimuleerde fotorealistische omgevingen en met commerciële drones, waardoor de potentie voor praktische toepassingen wordt aangetoond. Ondanks de vooruitgang erkent dit proefschrift de resterende uitdagingen bij het implementeren van multi-robotsystemen in realistische perceptietaken, vooral wanneer de strategie is aangeleerd, en suggereert richtingen voor toekomstig onderzoek.

Acknowledgments

At last! Four years and a pandemic later, this book is the result. Writing this thesis has not been an easy process but I have definitely enjoyed it both academically and personally despite its setbacks. This section is dedicated to thank all of the people who made this process easier, more enjoyable, and possible. Thank you all!

I would like to thank my supervisors, Javier Alonso-Mora and Wendelin Böhmer, for their time and feedback. Your guidance and patience have been invaluable in shaping my growth as a scientist and in the completion of this thesis.

I would like to express my gratitude to ONR Global for their financial support. The connections and discussions forged over the course of four years have been immensely valuable. A special thanks to Eduardo Montijano for the constructive feedback, inspiring conversations, and approachability. I always felt motivated after talking with you, even when the moments were tough. Your contributions have significantly influenced the development of this dissertation and the scientist I want to become. I am also grateful to Eduardo and the rest of the project members from Zaragoza: Ana Cris and Sara for creating a motivating work environment during my visit to the University of Zaragoza. I extend this thanks to the rest of the members from the group, who made me feel welcome from the first moment: Diego and Sara, it was great, and fun :), having you in Delft, Eduardo S. and Leon, for the great conversations and for indulging me by joining my afterwork bouldering session, Alberto, Clara, Carlos and David.

To Thomas and Aske, my new supervisors, for the opportunity, their mentorship, and their conviction in creating a safe and comfortable environment where all voices are heard and feel valued. I extend my thanks to the SELL group, Felix, Koen, Lindsay, and the RLG group for the nice conversations at lunch and the occasional office visits. Also, thanks to the people at LIACS, in particular Alberto, for insisting on talking about off-work topics, and Kash for his patience with me overusing our office whiteboard.

To my students, Shijie, Walter, Moji, Alessandro and Alex for their patience with my constant effort to improve my supervision skills.

To my coauthors for being one of the main reasons I can enjoy this frustrating but beautiful job. Working with you has been, and still is, a treat. To Daniel for the motivating and insightful talks about RL, the sporadic drinks in Barcelona/the Netherlands and sharing my first research project as a postdoc. Also to you, Dhruva, for joining the project and taking the time to share your feedback and experience despite your packed schedule, always maintaining a kind and optimistic attitude. To Gang Chen and Elia for allowing me to co-supervise a master's student and learn from them. To Max Lodel for allowing me to join his first research project and putting up with my ramblings on information gathering. Also to Bruno, and especially Hai, for joining my first publication and sharing your wisdom throughout experiments and summer schools. Last but definitely not least, to Manuel Turismo for his admirable work ethic and cheerfulness. Everyone should experience having a Manule as a co-author at least once.

If anything, I am deeply thankful for the social environment I got to enjoy in the AMR Lab and my group at the Cognitive Robotics Department. To Oscar, because there's always one coffee or tea left to share (even if we'll miss half of them), Andrés and Luzia for their kind attitude, Saray for her optimism, Andreu for sharing our passion for rice and giving great advice on how to build your own computer, and Dennis for being a great listener while building drones together. To Julian for his consistent baking skills, to Bas for our conversations over RL and the importance of a good simulator, to Tomas and Kyle for their friendship and motivating me to improve my English listening skills. To Tasos, thank you for your hospitality and our talks on Mediterranean culture, to Jelle, your jokes made the days in the office lighter, and to Linda, whose interdisciplinarity remains inspiring to this day, Mert, Padmaja, and Ashwin. I also want to thank Carlos, Laura, and Jens for their sympathy and sharing their experience over coffee.

To be fair, I came to Delft expecting this next section to be way shorter, yet here I am getting all nostalgic. To the friends I made during my Ph.D., you know how lucky I consider myself to be. Meeting you is part of this luck.

To my Ph.D. brothers, my Pennybois, Gio and Ro, who put up with me from day one until now. Thank you for all the dinners, BBQs, ice creams, walks, parties, and allowing me to show you Barcelona, but mostly thank you for all the support. Also thanks to Bea and Elena, who put up with our workaholic talks waaay too many times. I extend this thanks to Mnauel Kuramoto for allowing me to beat him at SET and for destroying me at chess, Lorenzoo for never failing to bring up a disrupting topic to spark a memorable group conversation, Italo for always lightening the mood, and Mariano for his patience putting up with all of us. To Max Lodel and his useful advice on improving framework setups, and Beatriz for having us over for lovely dinners and BBQs. Also thanks to Max Spahn, for being straightforward and genuine. Finally, thank you Carlos Celemin, for enduring the cold during our endless conversations and being a key pillar of support in Delft.

To my bouldering buddies who put up with my competitive nature. To Elia for being a grate friend you can always count on, and for sharing his culinary skills, also to Daniela, even if she still does not believe I got my pennyboard willingly. To Corrado and Patricija for reminding me to take life less seriously. I take more notes from you than you from me. To Lasse for being objectively an all-rounder. You can count on him to excel both academically and getting your bike back. I will always consider you our 3rd roommate. Also to Yuria, who is probably one of the coolest people I know. I definitely cannot leave out Tallo, for enduring our pets despite being confined to a dog's body. To Kinder Maxi King for getting me to run one last time during our pre-corona times, but mostly for being a wholesome person. Our walks, talks, and game sessions helped me a lot during tough times. Also thanks to Toni for enduring and even joining our rants about Ph.D. life.

Also thanks to my friends from my Bachelor and Master, who´ve had to suffer my WhatsApp sparsity. À Vincent pour me garder dans sa vie et montrer que ses capacités culinaires sont toujours top. Aussi à Fred, même si on se voit bien de temps en temps c'est toujours comme si on était toujours à Ferney-Voltaire et Genève. I extend my thanks to you too, Kyuhwa for your help during my Ph.D. application period, for being a great bouldering buddy and tricking me into your memorable surprise hikes. Gràcies Kittus per ensenyar-me el món de les magic i per ser el meu recomanador de videojocs oficial. También gracias a Martin por enseñarme chileno, y por ser y tratarme como familia alrededor de Europa.

Si me he considerado una persona con suerte ha sido por la gente con quien he tenido el placer de compartir mi vida hasta ahora. Sois la razón por la cual recuerdo el sol de Barcelona con cariño.

A la Mariana i Marimagda por tolerarme a mí y mis bromas por muy inapropiadas que fuesen, y por hacer como si el tiempo no hubiera pasado (incluso cuando el color de mi pelo insiste en lo contrario).

Gracias a mis amigos de la escuela, algunos de los cuales conozco desde los tres años, y a mis estrellas por conocerlos: Guillem, Laura y Marc por consentirme y visitarme a todas partes, a pesar de que eso supusiera convertir una pérdida de avión en una maraton de kung-fu panda, Edu por las largas conversaciones de camino a casa, Lluís por ser un amigo atento y un compañero de escalada excelente, Teresa i Júlia por tener siempre un cotilleo suculento que compartir. Gracias Ferran por tu ayuda durante la candidatura y por aguantar mis chapas sobre el Ph.D., a Adrián por su humildad y aguantar nuestras bromas, a Silvia por encontrar momentos para dar una vuelta a pesar de ser la persona más ocupada que conozco, y a Marc G. por ser un gran fichaje y mejor persona. Gracias a todos por hacerme sentir como si no hubiera dejado Barcelona durante estos cinco años y medio.

Gràcies a la Núria, Oriol i Pilar per rebre'm a casa seva amb els braços oberts. També gràcies a la Mireia, Pasqui, Marc, Lluís i Anna per acollir-me com un més de la família. Extenc els meus agraïments als nostres amics peluts Roc, Ona i Dobby per consentir els meus intents de fer-me amic seu.

A Meim, Cristina, y Peip, Javier, por enseñarnos, a Borja y a mi, a ser independientes, humildes y pensar críticamente, siendo vosotros ejemplo. Gracias por recibirnos con los brazos abiertos y hacernos sentir queridos y en casa sin pedir nada a cambio. Borja, mi hermano, mi tercer cerdito, gracias por creer en mi cuando yo no era capaz. Gracias por ser quien me introdujo al placer de viajar, la vida del expat y por ser un faro en mi vida. Te quiero tron. Y Brus, nuestro hermano peludo, gracias por permitirme disfrutar de tus días de cachorro. Perdona por irrumpir en tu casa sin aviso y dividir la atención de los papás. Por las graduaciones, los post-its en la maleta, las conversaciones hasta las 4am, las recogidas en el VIP del aeropuerto (y en Perpignan), los vuelos a las 7am y las despedidas Gómez, esta tesis tambien es vuestra.

A la meva parella, Natàlia, a qui vaig tenir la immensa sort de (re)conèixer fa sis anys. Gràcies per ser la meva companya de camí a través de pandemies, avions bi-setmanals, dues mudances travessant Delft i viatges per mig món. Gràcies per seguir-me a Delft tot i que això signifiqui haver d'escoltar les meves reflexions matutines sobre temes inconseqüents. Gràcies per la teva confiança, pel teu suport incondicional i per donar-me espai per recuperar la confiança i la seguretat que vaig perdre pel camí. Per obligar-me a descansar, aprendre junts a fer de la nostra casa una llar i per que ets un oasi, t'estimo Nat, aquesta tesi és per tu.

1

Introduction

1

1.1 Motivation

Multi-robot systems have the potential to make a significant societal impact by elevating the capabilities and efficiency of autonomous robot applications across diverse domains. While they have already demonstrated success in areas such as transportation [1–3] and logistics [4], their promise extends further into domains like surveillance and search and rescue [5], where their ability to cover extensive and intricate areas stands out as a notable advantage. While single robots can cover broader areas than human operators, they encounter notable scalability challenges in tasks requiring large area coverage [6]. In expansive environments, a single robot's ability to effectively and continuously cover the entire region of interest is greatly diminished, often leading to considerable gaps in information coverage. Additionally, environments rich in complexity necessitate robots equipped with sophisticated actuators and sensor capabilities for effective navigation and data processing [5]. However, incorporating these advanced features leads to increased costs, higher energy consumption, and diminished autonomy. Addressing these challenges typically requires additional investment to strike a balance between the robot's advanced capabilities and its energy efficiency [7]. Thus, the shift towards multi-robot systems marks a significant stride in transcending the limitations inherent in single-robot systems, paving the way for a more comprehensive realization of autonomous robotics across a wider spectrum of applications.

The strategic integration of multi-robot systems becomes particularly crucial in scenarios demanding swift action, such as search and rescue operations and surveillance tasks [5]. These applications often involve surveying extensive areas within a constrained time period. Multi-robot systems, in such cases, offer significant advantages due to their ability to cover larger areas, provide redundancy, and therefore enhance resiliency. The diverse sensors equipped in each robot, combined with their lower individual cost and increased flexibility, hold the promise of delivering fast response times, robust support in perilous environments, and the ability to parallelize complex tasks. This is especially evident in decentralized systems, where individual robots can amplify their problem-solving capacity and intervention efficacy while offering a more efficient computational approach.

Decentralized multi-robot systems enable the implementation of robust perception methods, effectively addressing potential faults and uncertainties in the robots' measurements [8]. Despite their capabilities, however, the current deployment of drones and multi-robot systems is primarily limited to structured, hand-crafted scenarios such as cinematography and entertainment [9], where the number of robots and the nature of tasks are known a priori. This limited scope represents only a fraction of their potential utility in more critical domains, highlighting a significant gap between current usage and the vast possibilities offered by multi-robot systems in enhancing operational efficiency and safety in crucial applications such as surveillance and triage/active perception in dynamic environments [10, 11]. Moreover, most industry-level applications, despite recent technological advances, still rely on offline planning and centralized solutions for multi-robot systems which are difficult to scale up, thereby underscoring the need for more adaptive, decentralized, and failure-resistant approaches in real-world applications [12].

The deployment of decentralized multi-robot system frameworks for motion planning

and active perception tasks poses significant challenges. First and foremost, these methods must be scalable, accommodating both a variable number of tasks and a variable number of robots within the environment. For instance, the system should be adaptable to different quantities of robots, ensuring the successful completion of all tasks in any case. Solving this already leads to ensuring the system's resilience to individual robot failures, the malfunction or breakdown of one or several robots no longer compromises the entire team's functionality [13]. Thirdly, it is crucial that the robots coordinate and interact efficiently. They should be able to cover the maximum possible area simultaneously, aiding each other in completing tasks, while avoiding interference or duplication of efforts.

This thesis tackles three core challenges in the domain of scalable robot navigation and active perception tasks: efficient communication for coordination, scalability across varying tasks and robots, and robustness. Firstly, efficient communication is key. While coordinating actions among robots does not necessarily require sharing future trajectory plans, doing so often leads to less conservative and more task-efficient behaviors. However, this approach has its trade-offs, notably increased resource consumption in terms of bandwidth and battery life. Secondly, the scalability challenge arises from the dynamic nature of perception and navigation tasks, such as reaching a set of goals or classifying dynamic targets. The number of tasks, e.g. the number of potential places to find victims in search and rescue operations, or the number of pedestrians in surveillance, can fluctuate over time and differ in various scenarios. Similarly, the number of robots available for active perception and navigation tasks is not fixed. Ideally, more robots would lead to faster task completion, but the actual number can vary based on robot availability or due to robot failures during execution. Often, this information is not known in advance. Lastly, robustness is addressed from two perspectives. The first pertains to the aforementioned changing scale of tasks and the size of the robot team, which may evolve due to robot failures or the addition of new members. The second perspective focuses on the transference from simulation to real applications (sim-to-real), emphasizing the importance of collision avoidance guarantees to manage risks effectively in robot navigation tasks. This thesis focuses on the motion planning, communication efficiency, and scalability of decision-making problems.

We envision a future where large multi-robot systems can reliably and robustly be used to monitor large areas populated by static and dynamic targets, either in hazardous environments, helping prioritize which survivors require aid, or farming, monitoring livestock and plant health. Therefore, in this dissertation the main research goal is to develop *algorithms that enable robust and scalable navigation of multi-robot systems in active perception tasks while coordinating using their resources efficiently.*

1.2 Approach

This thesis explores the application of learning-based methods to create scalable algorithms that enable efficient resource utilization during coordination and active classification within dynamic environments, characterized by a varying number of elements. Recognizing that learning methods inherently lack guarantees for dynamic feasibility and collision avoidance, this research adopts a modular approach. Within the proposed framework, learned policies are utilized to modulate the hyperparameters of a low-level controller. This strategy

1

ensures that the high-level adaptability of machine learning is effectively integrated with the reliability and precision of traditional control mechanisms. In this thesis, we follow this formulation to learn scalable and efficient communication policies in multi-robot systems, and viewpoint recommendation policies for active classification of multiple targets under full and partial environment observability.

From a high-level perspective, the algorithms developed in this thesis mainly rely on the following three pillars as the foundation to solve the problems of obtaining robust and scalable learning-based methods while maintaining collision avoidance guarantees.

1. **Reinforcement Learning:** The sequential decision problems (SDP) addressed in this thesis cannot be solved directly since the information required is difficult to model and can only be obtained by interacting with the environment. Fortunately, provided they are Markovian, these problems can generally be reformulated into a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$ describing the problem state $\mathcal{S}$ and action $\mathcal{A}$ spaces, dynamics or transition function $\mathcal{T}$, reward function R, and discount factor γ. Reinforcement learning (RL) involves an agent interacting with its environment, e.g. through trial and error, to learn behaviors that maximize cumulative reward. For each discrete time step t, given a state $s \in \mathcal{S}$, the agent undertakes an action $a \in \mathcal{A}$, following a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which results in receiving a reward r and transitioning to a new state s'. The return is defined as the discounted sum of rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i, s_{i+1})$ where γ is the discount factor that models the uncertainty of future rewards. The goal in RL is to find an optimal policy π_ϕ, parameterized by ϕ, that maximizes the expected return $J(\phi) = \mathbb{E}[\sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i, s_{i+1})]$. In this thesis, an on-policy RL algorithm named Proximal Policy Optimization (PPO) [14] is used to learn high-level policies that output high-level actions, e.g. communication requests or viewpoint recommendations, the long-term effect of which is difficult to model. PPO learns a policy that maximizes the cumulative rewards in a stable manner. In difference to other on-policy actor-critic methods, which have to periodically evaluate the policy every time it is updated by sampling new trajectories, PPO reuses samples where the policy has not changed much before resampling, making it more sample efficient.

2. **Model Predictive Control (MPC):** All the methods presented in this thesis end with a low-level Model Predictive Control (MPC) framework [15]. Assuming the availability of a model of the dynamical system, the MPC computes and predicts a sequence of control inputs and states that minimizes some costs for a given time horizon while satisfying a set of constraints. Following the receding horizon control paradigm, only the first control input of the sequence is applied, and the optimization is performed again with the state information updated. This process is iteratively carried out online until the goal system state is achieved. A primary advantage of MPC lies in its ability to foresee upcoming system events through forward prediction, allowing for proactive adjustments. Additionally, it can manage intricate costs and constraints, even in systems characterized by nonlinear dynamics. In this thesis, MPC is used as a low-level controller that tracks a given subgoal while satisfying robot dynamic feasibility and collision-avoidance constraints. Learning algorithms are used in combination to modulate the hyperparameters that shape the optimization

process, e.g. choice of subgoal being tracked, or modify the constraints restricting the solution. See Sections 2.3.4 and 2.3.3 for more details on the costs, and the formulation of the dynamic model employed throughout this thesis.

3. **Attention Mechanisms in Learning:** The interest in attention mechanisms started in the field of Natural Language Processing (NLP), arising as a tool for sequence-to-sequence modeling. Gaining significant traction through its successful applications in translation tasks [16], the field witnessed a paradigm shift with the introduction of the Transformer model [17]. Unlike RNNs and Graph Neural Networks (GNNs), attention mechanisms are permutation equivariant functions that are not constrained by the need for prior knowledge of the input sequence's structure and order. This allows them to adaptively focus on different segments of input data, enhancing their ability to encode large and variable-length sequences efficiently. Furthermore, attention mechanisms excel in learning latent embedding of the interactions among the elements of the sequence that is most suitable to solve complex tasks effectively. In the realm of Deep Learning's expansion into robotics, the application of attention models has become increasingly crucial, particularly in areas like task allocation and multi-robot motion planning [18]. These domains often involve the encoding of environments with dynamic and variable elements, such as multiple tasks or robots. In this thesis, attention models are employed as a key component for effectively encoding these environments, handling the complexities arising from multiple, dynamically evolving elements like tasks to be completed or coordination among team members.

1.3 Contribution and Outline

The goal of this thesis is to provide algorithmic innovations to enable robust and scalable multi-robot navigation in active perception tasks with efficient coordination. Hence, this thesis makes the following scientific contributions:

1. **A Scalable and Efficient Communication Policy** for multi-robot collision avoidance. A decentralized framework where robots learn to reason about the states of others and assess the need for communication. An attention-based neural architecture for the communication policy is presented which enables scalability. When a robot determines the need for communication with another, it requests its trajectory intentions and integrates these into its MPC constraints. If no communication is needed, constant velocity estimates are used instead. This method allowed a team of multiple robots to navigate safely making an efficient use of communication.

2. **A framework for Active Classification of Moving Targets with Learned Control Policies** that computes control inputs for drones to collect semantic information for the classification of multiple moving targets. An attention-based architecture, trained via Reinforcement Learning (RL), has been developed to determine the next optimal viewpoint for the drone. This method surpasses various baselines and shows strong generalization capabilities to different scenarios, including handling large numbers of targets with diverse movement dynamics.

3. **A Distributed Multi-Target Tracking and Active Perception framework with Mobile Camera Networks**. This is a hybrid camera system combining static and mobile cameras for improved surveillance and monitoring. This system enables collaborative observation and efficient visualization of dynamic targets within the environment, thereby overcoming the limitations of our previous work that assumed complete knowledge of all target poses. This framework maximizes the synergies of tracking and control modules, leading to a system that offers high-level scene understanding and is closer to real-world application. The effectiveness of collaborative mobile cameras over static or individual camera setups has been demonstrated through extensive analysis in a photo-realistic simulation environment.

The proposed algorithms of this thesis have been extensively evaluated and validated with commercial drones, see sections 2.3.4 and 2.6.1, and in photo-realistic simulated environments [19].

Figure 1.1 presents the overall thesis structure. Initially, Chapter 2 presents our scalable and efficient communication policy for decentralized multi-robot collision avoidance. Chapter 3 introduces the problem of Active Classification and our learning-based scalable solution for dynamic environments. Chapter 4 introduces our method for simultaneous tracking and classification of dynamic targets. Finally, Chapter 5 concludes this thesis and presents possible future research directions.

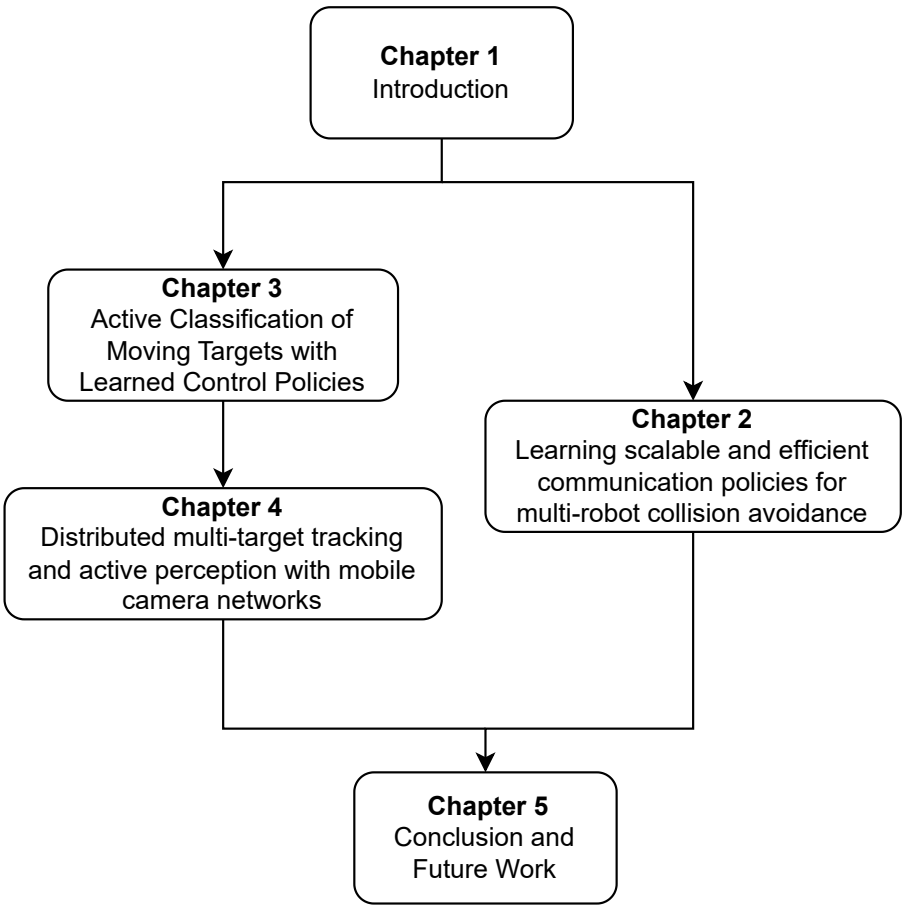


Figure 1.1: Thesis' structure: Chapter 1 introduces and motivates the research presented in this thesis. Chapter 2 presents the first contribution of this thesis: a scalable and efficient communication policy for multi-robot collision avoidance. Chapter 3 presents an active classification policy that is further explored in Chapter 4 in more realistic setting . Finally, Chapter 5 presents the conclusions and the proposed future research directions.

2

Scalable and Efficient Communication for Multi-Robot Collision Avoidance

The work of this chapter has appeared in:

Á. Serra-Gómez*, H. Zhu, B. Brito, J. J. Chung, J. Alonso-Mora, *With whom to communicate: Learning efficient communication for multi-robot collision avoidance*, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, pp. 11770-11776, 2020, doi: 10.1109/IROS45743.2020.9341762. [20]

Á. Serra-Gómez*, H. Zhu, B. Brito, W. Böhmer, J. Alonso-Mora, *Learning scalable and efficient communication policies for multi-robot collision avoidance*, in Autonomous Robots, vol. 8, no. 6, pp. 3717-3724, 2023, doi: 10.1007/s10514-023-10127-3 [21]

2.1 Introduction

Being able to account for the future trajectory of other robots is of utmost importance for safe navigation in environments shared with other robots. Centralized systems achieve this objective by using a central station to manage all of the robots' information and plans but they are difficult to scale up to large teams. Instead, decentralized systems can scale up by relying on each robot's on-board computation capabilities. Some decentralized solutions have each robot estimate other robots' behaviors or future trajectories through trained parametric functions, e.g. neural networks [22]. However, these solutions are generally computationally expensive and may have inaccuracies stemming from the lack of information on the other robot's goals and local observations. Instead, direct communication of each robot's trajectory intentions may allow to obtain accurate predictions with less computational effort. Common communication policies broadcast to all robots, employing distance-based heuristics to communicate trajectory plans. However, much of this information becomes redundant or unnecessary when robot motions do not present a threat to others, e.g. when they are far from each other, or may, at worst, harm the multi-robot system's performance [23]. Besides, designing a set of rules to communicate efficiently may be difficult as it would require to estimate a priori the future value of communicating with other robots, which also depends on their motion planner and dynamics. Since there is no clear intuition on how to hand-engineer an adequate trade-off between communication efficiency and safety, in this work, we focus on the following two issues: **a)** providing a solution to the problem of when and with whom to communicate that can scale up to large teams of robots, and **b)** how to couple this communication policy with existing motion planning methods.

We propose an efficient communication policy method combined with an optimal control motion planner for multi-robot collision avoidance that can handle large multi-robot systems with varying number of robots. The approach leverages the strengths of learning methods for decision-making and nonlinear receding horizon control, or Non-Linear Model Predictive Control (NMPC) for multi-robot motion planning. In particular, we use Multi-Agent Reinforcement Learning (MARL) to learn the robots' communication policies. For every robot and time instance, the policy selects a set of other robots [24] and requests their trajectory plans. Non-selected robots are assumed to follow their last communicated trajectory which is extended assuming constant velocity. This last communicated trajectory is exploited as long as they remain within a tolerance distance. Otherwise, robots that do not follow their last communicated trajectory are assumed to follow a constant velocity trajectory. Then, we formulate a nonlinear optimization problem to generate a safe trajectory. The planned trajectory takes into account the requested and estimated trajectories represented as constraints in the receding horizon framework.

The main contributions of this work are:

- A combined communication policy and trajectory planning method for micro-aerial vehicles (MAVs), that utilizes the strengths of non-linear model predictive control (NMPC) to plan safe trajectories, and multi-agent reinforcement learning (MARL) to learn an efficient communication policy.
- An on-line communication policy that uses MARL to learn (off-line) *when* and with *whom* it is useful to communicate, decreasing the amount of communication while

still achieving safe navigation and coordination among robots.

- We introduce a new neural architecture for the learned communication policy that scales to a large and varying number of robots while still providing safe navigation in a variety of situations.
- We demonstrate that the communication policy, which is trained in a simulator, works equally well in physical MAVs.

We evaluate our method with team of varying number of quadrotors in simulated scenarios requiring different levels of interaction for safe navigation and compare it with four other heuristic based methods for communication. We then test the robustness of our method under different levels of observation noise. Finally, we show that our method presents zero-shot generalization properties when tested in scenarios with more robots than during training while still maintaining safety online.

In an earlier conference version of this work [20], an early version of the framework to learn a communication policy and its combination with a local motion planner was introduced for a fixed number of robots. In this paper, we extend the approach with a new neural architecture and refine the training procedure of the communication policy to render the final navigation policy safer in interaction-rich situations, more robust to sampled training scenarios, and scalable to robot teams of varying amount of robots. We show that our learning method enables the emergence of more efficient and intuitive communication behaviours than before while maintaining a performance similar to that of broadcasting policies with regards to safe navigation.

2.2 Related Work

2.2.1 Communication in Collision Avoidance

We focus our work on online local motion planning for multi-robot systems (also referred as multi-robot collision avoidance), which has been actively studied over the past years. Traditional reactive controller-level approaches include the optimal reciprocal collision avoidance (ORCA) method [25], the artificial potential field (APF) based method [26], the buffered Voronoi cell (BVC) approach [27, 28], and control barrier functions (CBF) [29]. These methods are fully decentralized and each robot only needs to know other robots' current state, which can be measured by the robot via its onboard sensors. Hence, communication among robots is not necessary. However, these reactive methods are inefficient since they typically plan one time step ahead. This can result in overly conservative policies that are more vulnerable to deadlocks than predictive collision avoidance methods. These issues can be overcome by using a model predictive control (MPC) framework for collision-free trajectory generation that accounts for the plans of other robots [30].

For each robot to solve a local trajectory optimization problem in the MPC framework, it needs to know the future trajectories of other robots. One approach is to let each robot communicate its planned trajectory with every other robot in the team. Hence, robots can then update their own trajectories to be collision free with other robots' trajectory plans, as in these distributed MPC works [30, 31]. Another approach is to let each robot predict other robots' future motions based on its own observations. For instance, [32] employs a constant velocity model when predicting other robots' future trajectories. In

that case, communication among robots is not required. However, such a prediction can be inaccurate and may lead to unsafe trajectory planning, in particular when the robots are moving at a high speed [30]. In this paper, we aim to develop a communication policy, which determines "when" and "with whom" a robot should communicate with another robot in the system, to reduce the amount of communication while still keeping a high-level of safety.

2.2.2 Communication Scheduling

A lot of works tend to formulate the problem of efficient communication in a receding horizon fashion. Some methods formulate the problem as a decentralized version of a Markov Decision Process (Dec-MDP) [33] or Partially Observable MDP (Dec-POMDP) [34] and try to optimize a value function in which communications are penalized. Others, such as [35], choose to formulate a constrained optimization problem where communications must be directly minimized while still guaranteeing data flow throughout the network. These approaches assume access to an analytical model or require the design of a hand-engineered utility function to estimate the future effects of communication, which might not be available or intuitive to do, respectively. Recent work [36] manages to tackle this problem by triggering communication whenever uncertainty over another agent's actions exceeds a threshold. Ultimately, however, receding horizon methods are limited by their prediction horizon and the need for hand-engineered evaluation heuristics, which can unintentionally bias the resulting communication processes. In this work, we use reinforcement learning methods to learn the communication policy. Through learning from experience, this family of methods has the potential to discover more general policies without the need for fine-tuning hand-engineered heuristic functions..

2.2.3 Learning Methods for Coordination

One major challenge in Multi-Agent Reinforcement Learning (MARL) is the non-stationarity of multi-agent environments. This problem is caused by having multiple agents that learn and change their policy every learning iteration, which may result in the learning process being unstable. In order to mitigate this challenge, recent works on MARL [37–42] perform centralized training and decentralized execution. This paradigm has been applied in the field of non-communicating multi-robot collision avoidance tasks [43, 44] to learn an end-to-end navigation policy. Yet, these methods typically do not offer solid theoretical guarantees for collision avoidance. Instead we aim to learn the communication policy while leveraging already existing well-performing motion planners, e.g. [30].

Regarding tasks that require communication, several works have been published recently. Many of them focus on learning what content should be shared among agents, be it in the form of explicit messages [45], a composition of binary signals [46] and predefined symbols [47], policy hidden layers [48], or by directly sharing parameters among agents [49]. The most relevant to our work additionally focuses on learning, in a scalable way, policies that are able to appropriately choose *when* and with *whom* to communicate or cooperate in collision avoidance tasks. Prior work [50], address this task by assigning roles to every agent, making some of them in charge of organizing a common communication channel with their neighbours. However, regions where there are no agents with such a role are left without coordination capabilities. Instead, [51], [18] and [52] present end-to-end

MARL algorithms that design an attention module to assign and weight the importance of the messages received from other agents. While previous methods use dense attention mechanisms, [53] proposes an adaptive sparsity-inducing activation function to enable learning a sparse communication graph. Along these lines, [54] learn to choose whom to communicate with and evaluate the received messages to choose an action.

Similarly, the method we present in this paper can also be considered as an attention module targeting other agents. However, we set our communications to be unilateral to promote asymmetrical behaviour. Additionally, we decouple the problem of communication and motion planning, allowing the combination of our method with existing and well-tested solutions for motion planning in collision avoidance tasks.

2.3 Preliminaries

In this paper, we address the problem of deciding when and with whom to communicate during a multi-robot collision avoidance task. Though the proposed formulation is intended to be general, we are inspired by the results obtained in [30], which show how in a collision-avoidance scenario, methods that incorporate communication have a clear advantage over those that do not. We approach the information-sharing process as a MARL problem where the robots must learn to request information effectively. In this section, we set the context for our targeted communication process by formulating the problem of multi-robot collision avoidance. We provide an overview of the Non-Linear Model Predictive Control method used for motion control, as well as our MARL framework, introducing relevant notations for this work.

2.3.1 Multi-Robot Collision Avoidance

Consider a team of n robots moving in a shared workspace $\mathcal{W} \subseteq \mathbb{R}^3$, where each robot $i \in \mathcal{I} = \{1, 2, \dots, n\} \subset \mathbb{N}$ is modeled as an enclosing sphere with radius r. The dynamics of each robot $i \in \mathcal{I}$ are described by a discrete-time equation as follows,

$$\mathbf{x}_i^{t+1} = \mathbf{f}(\mathbf{x}_i^t, \mathbf{u}_i^t), \quad \mathbf{x}_i^0 = \mathbf{x}_i(0), \tag{2.1}$$

where $\mathbf{x}_i^t \in \mathcal{X} \subset \mathbb{R}^{n_x}$ denotes the state of the robot with dimension n_x, typically including its position $\mathbf{p}_i^t \in \mathbb{R}^3$ and velocity $\mathbf{v}_i^t \in \mathcal{R}^3$ (amongst others, see section 2.4.1), and $\mathbf{u}_i^t \in \mathcal{U} \subset \mathbb{R}^{n_u}$ the control inputs with dimension n_u. The function f is the model of the robot and is detailed in Section 2.3.4. The super-script $\cdot^t$ indicates the time step t. $\mathcal{X}$ and $\mathcal{U}$ are the admissible state space and control space respectively. $\mathbf{x}_i(0)$ is the initial state of robot i. Any pair of robots i and j from the group are mutually collision-free if $\left\|\mathbf{p}_i^t - \mathbf{p}_j^t\right\| \geq 2r, \forall i \neq j \in \mathcal{I}, \forall t = 0, 1, \dots$. Each robot has a given goal location $\mathbf{g}_i$, which generally comes from some high-level path planner or is specified by some user.

Robots in the team are allowed to communicate. Communication is assumed to be ideal, e.g. robots can communicate with each other perfectly and instantaneously. We also assume a point-to-point network topology. The implementation is viable provided that communication protocols with low energy consumption, such as a mesh network where links are established using Bluetooth LE, are utilized. Under this communication protocol, point-to-point communication topologies generally use less bandwidth than broadcasting topologies and scale better with the number of robots as they allow for redundant or

unnecessary communication to be avoided. Robots can associate other robots with the messages they send since, in practice, each sender could add its ID to the message or the receiving robot could infer it using the first position of the received trajectory.

The objective of multi-robot collision avoidance is to compute a local motion $\mathbf{u}_i^t$ for each robot in the group, that respects its dynamics constraints, makes progress towards its goal location $\mathbf{g}_i$ and is collision-free with other robots in the team for a time horizon $\tau = N\Delta t$, where Δt is the sampling time and N is the number of discrete steps.

2.3.2 Distributed Model Predictive Control

The key idea of using distributed model predictive control to solve the multi-robot collision avoidance problem is to formulate it as a receding horizon constrained optimization problem. For each robot $i \in \mathcal{I}$, the discrete-time constrained optimization formulation is

$$\begin{aligned} \min_{\mathbf{x}_i^{0:N}, \mathbf{u}_i^{0:N-1}} \quad & \sum_{k=0}^{N-1} J_i^t(\mathbf{x}_i^k, \mathbf{u}_i^k) + J_i^N(\mathbf{x}_i^N, \mathbf{g}_i) \\ \text{s.t.} \quad & \mathbf{x}_i^0 = \mathbf{x}_i(0), \\ & \mathbf{x}_i^{k+1} = \mathbf{f}(\mathbf{x}_i^k, \mathbf{u}_i^k), \\ & \left\| \mathbf{p}_i^k - \mathbf{p}_j^k \right\| \geq 2r, \\ & \mathbf{u}_i^{k-1} \in \mathcal{U}, \quad \mathbf{x}_i^k \in \mathcal{X}, \\ & \forall j \neq i \in \mathcal{I}; \forall k \in \{0, 1, \ldots, N\}. \end{aligned} \tag{2.2}$$

Where $J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k)$ and $J_i^N(\mathbf{x}_i^N, \mathbf{g}_i)$ are the stage and terminal costs, respectively [30] (defined in Section 2.3.3). At each time step, each robot in the team solves online the constrained optimization problem (2.2) and then executes the first step control inputs, in a receding-horizon fashion. In this paper, the generated future plans of robot i are also called robot i's (future) trajectory intentions. We reiterate that robot trajectory intentions are not equivalent but rather an approximation of their future trajectories.

2.3.3 Cost Functions

The components of the cost function $J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k), k = 0, 1, \ldots, N-1$ and $J_i^N(\mathbf{x}_i^N, \mathbf{g}_i)$ are defined in the following.

Goal navigation

We minimize the displacement between the trajectory's terminal position and the robot's goal location, and define a terminal cost

$$J_i^N(\mathbf{x}_i^N, \mathbf{g}_i) = w_i^N \frac{\left\| \mathbf{p}_i^N - \mathbf{g}_i \right\|}{\left\| \mathbf{p}_i^0 - \mathbf{g}_i \right\|}, \tag{2.3}$$

where $w_i^N \in \mathbb{R}^+$ is a tuning weight coefficient.

Control input cost

One of the stage cost terms is to minimize the control input,

$$J_{i,u}^k(\mathbf{u}_i^k) = w_{i,u} \left\| \mathbf{u}_i^k \right\|, \tag{2.4}$$

where $w_{i,u} \in \mathbb{R}^+$ is a tuning weight coefficient.

Collision cost

To improve safety, a stage collision potential field cost is introduced between the robot and each other robot,

$$J^k_{i,j,c}(\mathbf{x}^k_i) = \begin{cases} w_{i,c}(d^{\text{pot}} - d^k_{ij}), & \text{if } d^k_{ij} < d^{\text{pot}}, \\ 0, & \text{otherwise .} \end{cases} \tag{2.5}$$

where $w_{i.c} \in \mathbb{R}^+$ is a tuning weight coefficient, $d^k_{ij} = \left\| \mathbf{p}^k_i - \hat{\mathbf{p}}^k_j \right\|$ is the distance between robot i and each other robot j, and d^{pot} is the specified potential field distance, a scalar hyperparameter that establishes the limits of the potential field. The field grows linearly once a robot enters its limits as seen in equation 2.5. Then the collision potential cost is defined as

$$J^k_{i,c}(\mathbf{x}^k_i) = \sum_{j \in \mathcal{I}, j \neq i} J^k_{i,j,c}(\mathbf{x}^k_i). \tag{2.6}$$

Finally, the overall stage cost is

$$J^k_i(\mathbf{x}^k_i, \mathbf{u}^k_i) = J^k_{i,u}(\mathbf{u}^k_i) + J^k_{i,c}(\mathbf{x}^k_i). \tag{2.7}$$

2.3.4 Drone dynamic model

In this work, we use the same drone model and specifications for the Parrot Bebop2 SDK as in [30]. According to the Parrot Bebop2 SDK, the control inputs to the quadrotor are given by $\mathbf{u} = [\phi_c, \theta_c, v_{z_c}, \dot{\psi}_c] \in \mathbb{R}^4$, where ϕ_c and θ_c are the desired roll and pitch angles, v_{z_c} is the desired linear velocity in the z-axis and $\dot{\psi}_c$ is the yaw rate. To simulate the drone dynamics, we extend the state of each drone, as defined in Section 2.4.1, with information of its orientation (ϕ, θ, ψ). We use a first-order low-pass Euler approximation of the quadrotor dynamics [30], where the dynamics of the state velocity vector are:

$$\begin{cases} \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = R_Z(\psi) \begin{bmatrix} \tan\theta \\ -\tan\phi \end{bmatrix} g - k_D \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \\ \dot{v}_z = \frac{1}{\tau_{v_z}}(k_{v_z} v_{z_c} - v_z), \end{cases} \tag{2.8}$$

where $g = 9.81 m/s^2$ is the earth's gravity, $R_Z(\psi) \in SO(2)$ is the rotation matrix along the drone's local z-axis, k_D is the drag coefficient, k_{v_z} and τ_{v_z} are the gain and time constant of vertical velocity. The attitute dynamics of the quadrotor are:

$$\dot{\phi} = \frac{1}{\tau_\phi}(\phi_c - \phi), \qquad \dot{\theta} = \frac{1}{\tau_\theta}(\theta_c - \theta), \qquad \dot{\psi} = \dot{\psi}_c \tag{2.9}$$

where τ_ϕ, τ_θ are the time constants of roll and pitch angles respectively. In this work, due to the drone being able to move in any direction with any yaw angle, we fix the drone's yaw angle to zero. Consequently, $\dot{\psi}_c = 0$

2.3.5 Problem Formulation

For each robot to solve problem (2.2), it has to know the future trajectories of other robots in the team. Aside from the particular case of prioritized sequential motion planning schedules, obtaining the exact information on future trajectories beforehand is generally not feasible. Thus, other robots' future positions can be approximated either by estimating them [22] or by requiring other robots to communicate their trajectory intentions, computed during the previous time step [30].

At time t, let $\hat{\mathcal{T}}_{j|i}^{t} = \{\mathbf{p}_j^{t+1:t+N} \mid \text{predicted at time } t\}$ be the N-time horizon trajectory of robot $j \in \mathcal{I}, j \neq i$ that robot i assumes and uses in solving the problem (2.2), where the hat $\hat{}$ indicates that it is what robot i knows about the other agent's trajectory. Further denote by $\mathcal{T}_i^t = \{\mathbf{p}_i^{t:t+N} \mid \text{predicted at time } t\}$ the trajectory for robot i planned at time t. As mentioned, there are two ways for robot i to approximate the future trajectory of robot j, namely $\hat{\mathcal{T}}_{j|i}^{t}$:

- **Without communication:** robot i predicts another robot's future trajectory based on their current states, that is

$$\hat{\mathcal{T}}_{j|i}^{t} = \text{prediction}(\mathbf{x}_j^t), \quad \forall j \neq i \in \mathcal{I}. \tag{2.10}$$

  In [32], each robot was considered to follow constant velocity model for the prediction. However, this approach ignores the previously communicated information on future trajectory intentions from other robots, even when they could potentially hold more information on other robots' future positions than constant velocity estimates. Our prediction model uses the last communicated trajectory plans and expands it by assuming constant velocity. If robot j strays past a predetermined distance from its last communicated trajectory intentions, then robot j is estimated to follow a constant velocity model from its current position (section 2.4.4).

- **Communication request:** Robot i can request other robots j in the team to communicate their planned trajectories at each time step, that is:

$$\hat{\mathcal{T}}_{j|i}^{t}(0:N-1) = \mathcal{T}_j^{t-1}(1:N), \quad \forall j \neq i \in \mathcal{I}. \tag{2.11}$$

  where $\hat{\mathcal{T}}_{j|i}^{t}(a:b)$, with $a \leq b$, represents the subsequence of $\hat{\mathcal{T}}_{j|i}^{t}$ that goes from the a^{th} to the b^{th}-indexed element (inclusive). At time t, the last position of the communicated path of robot j: $\hat{\mathcal{T}}_{j|i}^{t}(N) = \hat{\mathbf{p}}_{j|i}^{N}$, cannot be communicated as it is beyond the N-time horizon at time step $t-1$. Therefore it is estimated by assuming constant velocity of robot j: $\hat{\mathbf{p}}_{j|i}^{N} = \hat{\mathbf{p}}_{j|i}^{N-1} + (\hat{\mathbf{p}}_{j|i}^{N-1} - \hat{\mathbf{p}}_{j|i}^{N-2})$.

Both of the two methods have their advantages and disadvantages. Fully communicating methods allow more accurate predictions and achieve safe collision avoidance as long as a feasible solution is found, but they require a large amount of communication among robots. If there is no communication, the robot may plan an unsafe trajectory if its prediction on other robots' trajectories deviates from their real ones or an overly conservative trajectory to avoid collisions.

Motivated by these facts, this paper aims to solve the problem of "with whom to communicate" for each robot in the team for collision avoidance. More precisely, at each time step, each robot i decides whether or not to request a trajectory intention from every other robot j. If robot i decides to request robot j, robot j should communicate its planned trajectory to robot i. If robot i decides not to request robot j, it predicts robot j's future trajectory based on its last communicated trajectory intention and the observed current state of robot j.

Denote by $\pi_i^t = \{c_{j|i}^t \mid \forall j \neq i\}$ the communication vector of robot i at time t, in which $c_{j|i}^t = 1$ indicates that robot i requires a communicated trajectory from robot j. Otherwise $c_{j|i}^t = 0$. Note that $c_{i|i}^t = 0$ since the robot does not need to communicate with itself. Let $\pi^t = \{\pi_1^t; \ldots; \pi_n^t\}$ be the communication matrix of the multi-robot system at time t in an episode of length T_e. We define the communication cost of the system to be

$$C(\pi^t) = \frac{1}{N_c(n)} \sum_i^n \sum_j^n c_{j|i}^t. \tag{2.12}$$

Where $N_c(n) = n(n-1)/T_e$ is a normalization factor depending on the number of agents and the length of the episode, that represents the maximum amount of communications that can happen within a system of n robots across T_e timesteps. The objective of this paper is to find a policy for each robot i,

$$\pi_i^t = \pi_i(\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_n^t) = \{c_{j|i}^t \mid \forall j \neq i\} \tag{2.13}$$

that minimizes $C(\pi^t)$ while ensuring that the robots are collision-free with each other in the system.

2.4 Method

An overview of the proposed method is given in Figure 2.1. It consists of two components: a learned communication policy, which we introduce as *WW2C*, that decides with whom to communicate, and a NMPC planner.

Every time step, based on its partial observation of the current joint state z_i^t, every robot targets a set of other robots π_i^t and requests their intended trajectory plans $\hat{\mathcal{T}}_{j|i}^t = \mathcal{T}_j^{t-1}$ according to a learnt parametric policy $\pi_{i,\theta_i}(z_i^t)$. Those robots not targeted are estimated to follow a previously communicated trajectory extended assuming constant velocity or, in case it is no longer useful, a constant velocity model $\hat{\mathcal{T}}_{j|i}^t = \text{prediction}(\mathbf{x}_j^t)$ as described in section 2.3.5.

A receding horizon optimization is then employed to plan the future intended trajectory $\mathcal{T}_i^t$ for robot i. To guarantee the safety of such a trajectory, the resulting path is constrained to not intersect with $\hat{\mathcal{T}}_{j|i}^t$ for any $j \neq i$. The first action input from the computed plan is applied and a new observation is gathered. Along this work we assume that robots plan and execute actions in a synchronized fashion. While this assumption is necessary for learning the communication policy, it can be alleviated during test time since robots employ previously received or estimated trajectories [22].

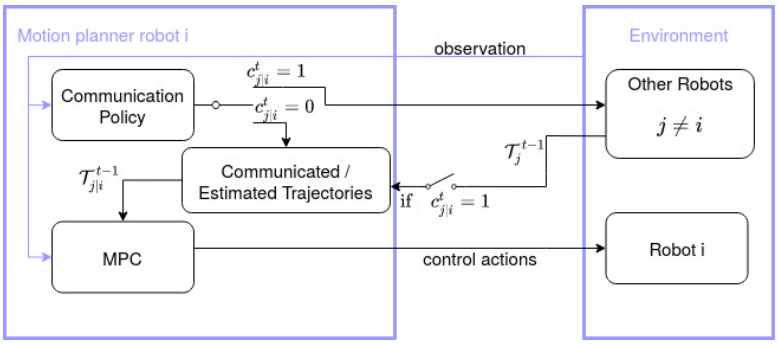


Figure 2.1: Schema of the proposed method for efficient communication. $\pi^t_i(z_i)$ is the communication policy dependent on the observation z_i. In practice, $\pi^t_i(z_i)$ is a boolean vector $c^t_{j|i}$ indicating if i requests j its trajectory intention, or estimates it. $\mathcal{T}^{t-1}_j$ is the trajectory intention of robot j at the previous time step. And $\hat{\mathcal{T}}^t_{j|i}$ is the combination of obtained and estimated trajectories of the other robots.

2.4.1 Reinforcement Learning Setup

We formulate a multi-robot reinforcement learning problem to compute an efficient communication policy. By considering the MPC based motion planner as part of the transition function, this problem can be transformed into a decentralized POMDP [55]. The decentralized POMDP is composed of six components, including state space, action space, observation space, reward function, transition model and observation model.

State space $\mathcal{X}$

For every robot i, $\boldsymbol{x}_i \in \mathcal{X}$ must account for the current physical state, its goal position, its sequence of intended future positions, computed during the previous time step by the motion planner, and its knowledge on other robots' future trajectory plans. Therefore, the state at time t can be defined as:

$$\boldsymbol{x}^t_i := [\boldsymbol{g}_i, \boldsymbol{p}^t_i, \boldsymbol{v}^t_i, \mathcal{T}^{t-1}_i, \hat{\mathcal{T}}^{t-1}_{-i|i}], \tag{2.14}$$

$$\boldsymbol{X}^t := \{\boldsymbol{x}^t_1, \boldsymbol{x}^t_2, ..., \boldsymbol{x}^t_n\}, \tag{2.15}$$

where $\boldsymbol{g}_i, \boldsymbol{p}^t_i, \boldsymbol{v}^t_i \in \mathbb{R}^3$ are the goal, position and velocities of robot i at time t, and $\hat{\mathcal{T}}^{t-1}_{-i|i} = \{\hat{\mathcal{T}}^{t-1}_{j|i} \mid \forall j \neq i\}$. Then, $\boldsymbol{X}^t$ is the joint state of the whole multi-robot system. Following a similar formulation as [44], robot i only has access to the information of its own state $\mathbf{x}^t_i$ and the terms from other robots j that can be estimated through its sensors, such as their positions and velocities.

Observation space $\mathcal{Z}$

We assume all robots are within sensor range (e.g. camera, lidar, ...) of each other and can always estimate the relative positions and velocities of all other robots. Each robot also knows the relative position of its own goal from a mission planner. For robot i, partial observations on the joint state at time t are:

$$\boldsymbol{z}^t_i = [\boldsymbol{v}^t_i, \boldsymbol{p}^t_{i,g}, \{\boldsymbol{d}^t_{j|i}\}_{j \in I \setminus i}, \{\boldsymbol{p}^t_{j|i}\}_{j \in I \setminus i}, \{\boldsymbol{v}^t_{j|i}\}_{j \in I \setminus i}], \tag{2.16}$$

where $d^t_{j|i}$, $\boldsymbol{p}^t_{j|i}$ and $\boldsymbol{v}^t_{j|i}$ are the relative distances, positions and velocities of the other robots with respect to the i^{th} robot, and $\boldsymbol{p}^t_{i,g}$ is the relative position of robot i's goal. The joint observation from all robots is denoted by $\boldsymbol{z}^t = \{\boldsymbol{z}^t_1, ..., \boldsymbol{z}^t_n\} \in \mathcal{Z}$

Action space $\mathcal{A} = \times_{i \in \mathcal{I}} \mathcal{A}_i$

As it has already been introduced in section 2.3.5, we denote by $\pi^t_i = \{c^t_{j|i} \mid \forall j \neq i\}$ the communication vector of robot i at time t. Note we have dropped the i^{th} element as the robot cannot communicate with itself. Therefore the action space for robot i is:

$$\mathcal{A}_i = \{0, 1\}^{n-1}$$

Note that the dimensionality of the action space depends on the number of agents. This matter will be further addressed later in 2.4.2.

Reward $R_i(\boldsymbol{x}^t, \pi^t)$

The reward function is chosen based on the multiple behaviors we want to achieve. It aims for the learned communication policy to communicate as little as possible while allowing each robot in the team to reach its goal and avoid collisions. The reward value $R(\boldsymbol{x}^t, \pi^t)$ is the immediate reward that every robot i gets at a state $\boldsymbol{x} \in \mathcal{X}$ after applying the communication vector π^t_i.

Global rewards are often used in multi-agent systems in order to capture coordinating coupled behaviors. However, this often leads to multi-agent credit assignment problems during training [56]. In this work we attempt to capture coupled behaviors by employing the same reward function conditioned by each robot's state individually. Since all agents share the same architecture and policy parameters, this allows to quickly learn to properly punish pairwise interactions such as collisions and communications as all samples from all robots can be used in the same way to compute the gradients and update the parameters of the communication policy. Also, even though reward signals are individual, optimising the same set of parameters for all agents at the same time allows to account for coordinating behaviours when there are more than two agents crossing paths. The reward function is composed of the following weighted combination of terms:

$$\begin{aligned} R_i(\boldsymbol{x}^t, \pi^t_i) = w_g R_{g,i}(\boldsymbol{x}^t) + w_{coll} R_{coll,i}(\boldsymbol{x}^t) \\ + w_c R_{c,i}(\pi^t) \end{aligned} \tag{2.17}$$

where w_g, w_{coll}, w_c are the weights for each term. Each reward term is defined:

$$R_{g,i}(\boldsymbol{x}^t) = \begin{cases} r_g & \left\| \boldsymbol{p}^t_{i,g} - \boldsymbol{p}^t_i \right\| \leq r_i \\ 0 & \text{otherwise} \end{cases}$$

with $r_g > 0$ is a tuned reward given at the end of the episode if robot i is within its goal, r_i is the radius of the smallest sphere containing the robot. This reward gives an incentive to learn communication patterns that stir the robot toward its own goal.

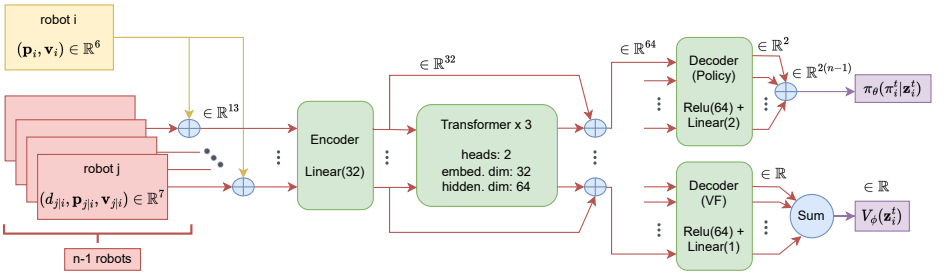


Figure 2.2: Proposed network policy architecture. Information from the ego and other robots is marked respectively in orange and red. Red arrows show the flow of information through the architecture of each individual robot j. Architecture layers are marked in green. Concatenation and Sum operations are marked in blue. Outputs are shown in purple.

$$R_{coll,i}(\boldsymbol{x}^t) = \begin{cases} -r_{coll} & \forall j \in \mathcal{I}, i \neq j, \\ & \left\| \boldsymbol{p}_i^t - \boldsymbol{p}_j^t \right\| \leq r_i + r_j \\ 0 & \text{otherwise} \end{cases}$$

where $r_{coll} > 0$ is a tuned penalty term for the collision between any two robots.

Finally the local penalization term for path plan requests is similar to the global version introduced before in section 2.3.5 and has the form:

$$R_{c,i}(\pi^t) = -C_i(\pi^t) = -\frac{1}{N_i(n)} \sum_{j \neq i}^{n} c_{j|i}^t.$$

Where $N_i(n)$ is a local normalization term depending on the number of agents.

Observation model $\mathcal{O}(z^{t+1}, \boldsymbol{x}^{t+1}, \pi^t)$
We assume that every robot i can directly observe the positions and velocities of other robots. Although not all information is observed, the observation vector is determined completely by the given state vector $\boldsymbol{x}^{t+1}$.

Transition model $T(\boldsymbol{x}^{t+1}, \pi^t, \boldsymbol{x}^t)$
The transition model can be decomposed into a communication step and a physical action step. The communication step is stochastic, only during training (see Section 2.4.2), and models the effects of communication π^t on the constrained optimization problem used to compute the control actions applied at time step t, $\boldsymbol{u}^t$. Then, the robot model f, introduced in Section 2.3.1, determines the joint state at the next time step $\mathbf{x}^{t+1}$. Note that, since we are sharing parameters, the communication matrix π^t depends directly on the shared policy. The robots employed in this paper are quadrotors, thus the state transition can be interpreted as the quadrotor model introduced in Section 2.3.4.

2.4.2 Network Architecture

Given the input (observation $\mathbf{z}_i^t$) and output (action π_i^t), we elaborate on the communication policy network mapping $\mathbf{z}_i^t$ to π_i^t. We want each robot to process all the information from

the environment and decide whether it needs to make a request to any of the other robots. While concatenation of other robots' observed information is possible at low scales, learnt policies are bound to quickly deteriorate in performance as the input vector dimensionality grows exponentially with the number of robots. We need an architecture that can provide a compact representation of the observed information of an arbitrary number of other robots, while still being able to leverage that information and choose whether each robot's trajectory is needed to compute a safe trajectory. Therefore, we need an architecture that can pool together all the information coming from all the different robots while still being able to output a different signal for each one of them.

There are several recent works in the field of motion planning that use information pooling mechanisms. Often, simulated laser scanner observations allow considering all information in the environment without having to explicitly define each element and its properties [57, 58]. However, individual information on each of the other agents and their interaction with the environment are lost. Other works address this issue by using Recurrent Neural Networks (RNN) [59] over the sequence of other robots observations [44, 60]. While these methods allow learning the additional coupled effects resulting from adding each robot into the environment representation, they are not permutation invariant and their performance depends on the heuristic ordering method chosen to feed the elements into the network. Methods using Graph Neural Networks (GNN) [61], represent elements in the environment as vertices in a graph and allow learning a permutation equivariant and compact representation of the set of observed information on each one of the vertices [45]. In [62], attention mechanisms [17] are formulated as a GNN for the particular case of fully connected graphs and are used to map a set of sensor measurements from an agent with multiple limbs to a set of actions to be applied by each one of them. We build our policy network on top of the attention-based architecture proposed in [62] and extend its use to homogeneous multi-agent environments.

Our communication policy architecture is depicted in Figure 3.2. We design a five-layer neural network as a nonlinear function approximation of the policy π_θ. For each robot i, we arrange the information on the other robots $\{\boldsymbol{d}^t_{j|i}\}_{j\in I\setminus i}, \{\boldsymbol{p}^t_{j|i}\}_{j\in I\setminus i}, \{\boldsymbol{v}^t_{j|i}\}_{j\in I\setminus i}$ into a sequence of vectors $\{(\boldsymbol{d}^t_{j|i}, \boldsymbol{p}^t_{j|i}, \boldsymbol{v}^t_{j|i})\}_{j\in I\setminus i}$ and append the observed information on robot i at the end of each element in the sequence. Our policy network consists of three parts: an encoder layer, a transformer block, and a decoder layer. Each one of these layers has the property of being permutation equivariant and enables processing sequences with an arbitrary number of vectors, even during testing. The encoder layer consists of a linear layer applied independently to each element of the sequence, mapping each element $\boldsymbol{z}^t_j$ to a latent representation of higher dimension $\tilde{\boldsymbol{z}}^t_j$. We employ a three-layered transformer [17], to allow each element $\tilde{\boldsymbol{z}}^t_j$ in the sequence to exchange information among themselves and encode the information present in the environment while still providing a different result for each element. However, information on each robot's relative position and velocities are still very important regardless of the additional information and coupled effects coming from other robots in the environment. To preserve this information, we concatenate each element j with its counterpart $\tilde{\boldsymbol{z}}^t_j$, which also enables the transformer block to focus on learning the coupled effects in communication arising from having multiple agents in the environment.

The network has two decoder heads applied independently to each one of the sequence

outputs: one computes the communication action for each robot j: π_i^t, while the other estimates the state-value function: $V^\pi(\boldsymbol{x}^t) = \mathbb{E}_{x\sim p_{\pi_\theta}, a\sim\pi_\theta}[\sum_{k=0}^{\infty}\gamma^k R_i(\boldsymbol{x}^k, a) \mid x^0 = x^t]$. Both of them start with a multi-layer perceptron with a hidden layer with a ReLU non-linearity and an output linear layer. The first head outputs a 2-dimensional vector per robot of communication scores that we project onto the probability 1-simplex with a softmax activation function resulting in the vector $[p_{j|i}, 1-p_{j|i}]$. To enhance exploration, during training the output action is sampled from the resulting Bernoulli distribution $\mathcal{B}(p_{j|i})$. While testing, we follow a deterministic policy where robot i requests j's trajectory intentions when $p_{j|i} > 0.5$.

The second head outputs a scalar representing the contribution $V_{j|i}$ of each robot j to the value function. Similar to the use of value decomposition networks in collaborative multi-agent tasks [40], we model agent's i value function as $V^\pi(\boldsymbol{z}_i^t) = \sum_{j\neq i, j=1}^{n} V_{j|i}$.

2.4.3 Multi-Scenario Multi-Stage Training

In order to learn a robust communication policy, we present a multi-stage training scheme in varied scenarios with a clear separation between training and test regime.

Training algorithm

Here we focus on learning a robust communication policy that, in combination with an MPC for motion planning, allows large multi-robot systems to coordinate and navigate at least as safely as when using broadcasting communication policies. To accomplish this, we use the extension to homogeneous multi-agent systems developed in [58] of the on-policy policy gradient algorithm: Proximal Policy Optimization (PPO) [14] under the assumption of parameter sharing across agents [49], although the general framework is agnostic to the specific RL training algorithm. For this matter, we take the *centralized learning, decentralized execution* paradigm, which is already popular in multi-agent reinforcement learning for decentralized systems [37, 44]. In particular, the individual policy shared by all agents is learned in a centralized way from the experiences gathered by all robots simultaneously during training. This has been shown to allow the policies of homogeneous agents to be trained more efficiently, and mitigate the non-stationarity in the environment dynamics that arises from having multiple agents learning at the same time. While testing, each robot has copy of the learned policy which is executed in a decentralized fashion.

Algorithm 1 describes the proposed training strategy which alternates between gathering experiences $(\mathbf{z}_i^t, \pi_i^t, R(\mathbf{x}_i^t, \pi_i^t), \mathbf{z}_i^{t+1})$ from all robots and performing PPO gradient updates. PPO is an on-policy method that addresses the high-variance and the difficult hyper-parameter tuning in policy gradient methods for continuous control problems. As suggested in [14], in this particular PPO implementation [63], we add to the surrogate objective an entropy bonus and a value function loss to ensure sufficient exploration and account for the shared parameters between the policy and the value function. We refer the reader to [14] and [58] for more information on the method's equations and details. The hyperparameters used for training are detailed in table 2.1.

As explained in [58], this multi-robot adaptation of the PPO algorithm can be parallelized and easily scaled to large-scale multi-robot systems since every robot counts as an independent worker gathering data. This reduces the sampling time cost and makes

Table 2.1: Hyperparameters for PPO training algorithm

Parameter	Value
Lambda λ	1.0
Gamma γ	0.99
Episodes each iteration n_e	40
Episode time steps/episode T_e	100
Number of epoch per iteration E_θ	30
SGD minibatch size $n_{minibatch}$	512
Clip param. ϵ	0.3
KL target KL_{target}	0.01
Learning rate lr_θ	5e-5
KL coeff. β	0.2
Value function loss coeff. c_1	1
Entropy loss coeff. c_2	0.001
Gradient Clipping	0.1

the algorithm suitable for training a large number of robots in various scenarios, profiting from frameworks specialized in distributed computation (i.e. Ray [64] and RLlib [63]) to accelerate our network's convergence.

Training scenarios

While proper exploration of the action and state spaces is crucial for the quality, robustness and generalization characteristics of the learned communication policy, it is difficult to achieve proper exploration of the state space since our policy only decides on *whom to communicate with*. Therefore, it is necessary to design interaction-rich training scenarios where the robots can sample meaningful experiences that will allow them to learn when it is necessary to cooperate and request other robots future trajectory intentions.

We have created a simulation environment [30] where a group of twelve drones navigate from an initial position to a goal position and may communicate their trajectory plans to perform collision avoidance. We have designed three different scenarios to train our communication policy, as depicted in the left column of Figure 2.3. Each scenario requires increasing levels of interaction and cooperation to perform collision avoidance, ranging from a simple scenario where almost no communication is needed (e.g., Figure 2.3a) to complex scenarios where the drones must communicate (e.g., Figure 2.3e) to successfully avoid each other. The employed scenarios are:

- **Random navigation (Figure 2.3a)**: Each robot must to move to a random goal position.
- **Random swapping (Figure 2.3)c**: The group of robots is arranged in pairs. Then each robot switches position with its counterpart.
- **Asymmetric swapping (Figure 2.3e)**: We split the $\mathbb{R}^2$ x-y plane into twelve quadrants and randomly initialize each robot in a different quadrant with random initial

position. Then, each robot swaps positions with a robot from the diametrically opposed quadrant.

The learned communication policy depends on the scenarios on which it is trained. For instance, if an agent is trained only on the first scenario it will learn a *no communication* behavior. In contrast, if only trained in the last one, it may learn to always communicate. Hence, we employ *curriculum learning* [67], training the agents first in a simple scenario, where communication is generally not needed, and subsequently introducing more difficult and complex scenarios where the agents must learn *when* and *with whom* to communicate. We design the learning process in three stages of 12500 episodes, through which we sample episodes from a scenario pool. During the first stage the scenario pool only has the first type of scenario, thus we sample random navigation scenarios with probability one. In the second step, we add the second type to the scenario pool, sampling the new scenario 75% of times. Finally, during the last stage we add the *asymmetric swapping* type to the pool, sampling this one 75% of times and 12.5% each of the others.

Training/Test regime differentiation

As explained in section 2.4.3, the learned communication policy cannot explore directly the state space since only the MPC is in charge of motion planning. Also, communication is not the only source of cooperation as the other robot's future trajectory can be approximated by constant velocity models. This results in our robots rarely exploring collisions or dangerous situations even when not communicating, thus creating sparsity and a lack of collision experiences to learn from. On top of this, while all communications are punished instantaneously when they happen, there is a delay between issuing a communication request (or not) and its associated reward for completing the episode successfully (or colliding).

We apply a crucial change between the training and testing regimes to promote exploration of the state space, specially collision events. During training, we turn off constant velocity predictions whenever robot i does not communicate with robot j to request its trajectory intentions. This makes robot i MPC planner virtually blind to robot j future intended positions. This will ultimately result in a learnt communication policy that keeps track and decides to request trajectory intentions from all robots that might put the safety of our future trajectory at risk, or need to be cooperated with. As seen in section 2.5.6, this modification results in an efficient and intuitive communication policy that is as safe as broadcasting policies. During testing, we turn on once again trajectory estimations whenever there is no communication both as a safety layer and to avoid generating oscillating trajectories due to switching on and off constraints related to other robots' future trajectories in the MPC.

The main idea of turning robot i blind to robot j whenever it does not request j's trajectory intentions, is to force collisions to happen whenever necessary communications are not effectuated. This way, it is easier to discriminate and learn the cause and effect relationship between a communication signal being (not) triggered between i and j and their subsequent collision (not) being avoided. The intuition behind this training approach is to learn *with whom to cooperate* rather than *with whom to communicate* as it results in a policy that puts the attention in those other robots in the team that need to be cooperated

with to achieve safe navigation. Although less information for the motion planner during training results in a suboptimal policy for testing, i.e. more communicative than necessary, we argue that the additional information can help in some situations and, at worst, will not make the resulting policy less safe. This is why the found solution is able to achieve similar performance to broadcasting policies in terms of safety. The robot motion planning loop during training and testing are detailed in Algorithm 2.

2.4.4 Predicting and Generating Safe Trajectory Intentions

Informed constant velocity estimations

At time step t, robot i can obtain $\hat{\mathcal{T}}_{j|i}$ either by communicating and requesting robot j's future trajectory intention ($\hat{\mathcal{T}}^t_{j|i} = \mathcal{T}_j^{t-1}$) or directly by computing an estimation of said robot's future trajectory ($\hat{\mathcal{T}}^t_{j|i} = \text{prediction}(\mathbf{x}^t_j)$). Algorithm 3 depicts the structure of the prediction function.

Whenever robot i requests robot j's trajectory intentions at time step t, it stores both the last communicated trajectory $\mathcal{T}_j^{t-1}$ and the time step of communication t. Every subsequent time step $t+k$, we make sure the last communicated trajectory intention is not obsolete, and that robot j is within a predefined tolerance region around its trajectory intention r_{tol}. If any of these conditions is not true, the last communicated trajectory is discarded and robot j's future positions are estimated by assuming it will follow constant velocity. If the communicated information is not discarded, we take the remaining $N-1-k$ steps from the tail of the communicated trajectory intention and expand them until obtaining a set of N future positions by assuming constant velocity.

Generating safe trajectory intentions

At time t, given robot i's requests for information π_i^t, we can determine $\hat{\mathcal{T}}^{t-1}_{j|i}$. Then, robot i's computed trajectory intentions $\mathcal{T}_i^t$ and control inputs $\mathbf{u}_i^t$ are computed by solving a constrained optimization problem. This optimization problem computes the optimal future values for $\{(\mathbf{x}_i^{t+l+1}, \mathbf{u}_i^{t+l}) \mid \forall l = 0, ..., N-1\}$, that minimize, over a N-time step horizon, a given cost function (defined in Section 2.3.3). The solution of the problem is constrained to follow the robot's dynamic model $\mathbf{f}$ and account for the estimates of other robots' trajectory intentions $\hat{\mathcal{T}}^{t-1}_{j|i}$ to avoid future collisions (equality and inequality constraints). The sequence of intended future positions in $\{\mathbf{x}_i^{t+l+1} \mid \forall l = 0, ..., N-1\}$ is used to construct $\mathcal{T}_i^t$, while only the first value of the sequence $\{\mathbf{u}_i^{t+l} \mid \forall l = 0, ..., N-1\}$, $\mathbf{u}_i^t$, is used.

The sequence $\{(\mathbf{x}_i^{t+l+1}, \mathbf{u}_i^{t+l}) \mid \forall l = 0, ..., N-1\}$ is computed at every time step t by formulating and solving the following constrained optimization problem:

$$
\begin{aligned}
\min_{\mathbf{x}_i^{0:N}, \mathbf{u}_i^{0:N-1}} \quad & \sum_{k=0}^{N-1} J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k) + J_i^N(\mathbf{x}_i^N, \mathbf{g}_i) \\
\text{s.t.} \quad & \mathbf{x}_i^0 = \mathbf{x}_i^t, \\
& \mathbf{x}_i^{k+1} = \mathbf{f}(\mathbf{x}_i^k, \mathbf{u}_i^k), \\
& \left\| \mathbf{p}_i^{k+1} - \hat{\mathbf{p}}^{k+1}_{j|i} \right\| \geq 2r, \quad \forall j \in \mathcal{I} \backslash \{i\} \\
& \mathbf{u}_i^k \in \mathcal{U}, \quad \mathbf{x}_i^{k+1} \in \mathcal{X},
\end{aligned}
\tag{2.18}
$$

Table 2.2: Hyperparameters for the WW2C framework

Parameter	Value
Episode length T_e	100
Trajectory prediction horizon N	20
Time step length Δt	0.05s
Tolerance length r_{tol}	0.1m

where $\hat{\mathbf{p}}_{j|i}^{k+1}$ are extracted from $\hat{\mathcal{T}}_{j|i}^{t-1}$. $J_i^k(\mathbf{x}_i^k, \mathbf{u}_i^k)$ and $J_i^N(\mathbf{x}_i^N, \mathbf{g}_i)$ are the stage and terminal cost functions to be minimized, which are defined in Section 2.3.3. Function $\mathbf{f}$ is the non-linear discrete function representing the dynamic model of the robot.

Algorithm 1 PPO for multiple agents with parameter sharing

Initialize policy network π_θ and value function V_ϕ. Set hyper-parameters as shown in Table 2.1. Note that θ and ϕ share the same set of parameters except for the decoder layer.
for iteration = 1,2,..., **do**
  for Robot i = 1,2,...,n **do**
    // Collect data in parallel. We define $r_i^t = R_i(\mathbf{x}_i^t, \pi_i^t)$
    for e = 1,2,...,n_e **do**
      Run comm. policy π_θ for episode e, collecting $\{\mathbf{z}_i^t, r_i^t, \pi_i^t\}$ where $t \leq T_e$ (**Algorithm 2**)
      Estimate and collect advantages using GAE [65]: $\hat{A}_i^t = \sum_{l=0}^{T_e - t} (\gamma\lambda)^l \delta_i^{t+l}$ where $\delta_i^t = r_i^t + \gamma V_\phi(\mathbf{z}_i^{t+1}) - V_\phi(\mathbf{z}_i^t)$
      Estimate and collect target values: $V_{target}^t(\mathbf{x}_i^t) = \sum_{t'>t}^{T_e} \gamma^{t'-t} r_i^{t'}$
    end for
  end for
  $\pi_{old} \longleftarrow \pi_\theta$
  //Update policy and value function
  for j = 1,...,E_π **do**
    for b = 1,...,$(n_e T_e n)//n_{minibatch}$ **do**
      Sample minibatch D_b from collected rollout data D
      //Surrogate objective. We define: $h_t^i(\theta) = \frac{\pi_\theta(\pi_i^t|\mathbf{z}_i^t)}{\pi_{old}(\pi_i^t|\mathbf{z}_i^t)}$
      $L^O(\theta) = \mathbb{E}_{(\mathbf{z}_i^t, r_i^t, \pi_i^t, V_{target}^t(\mathbf{x}_i^t), \hat{A}_i^t) \sim D_b} [min(h_t^i(\theta)\hat{A}_i^t, clip(h_t^i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i^t) + \beta KL[\pi_{old} \mid \pi_\theta]]$
      //Value function Loss
      $L^{VF}(\phi) = \mathbb{E}_{(\mathbf{z}_i^t, r_i^t, \pi_i^t, V_{target}^t(\mathbf{x}_i^t), \hat{A}_i^t) \sim D_b} [(V_{target}^t(\mathbf{x}_i^t) - V_\phi(\mathbf{z}_i^t))^2]$
      //Entropy objective. We define $S[\pi_\theta](\mathbf{z}_i^t)$ *as the entropy of the policy distribution* $\pi_\theta(\pi_i^t \mid \mathbf{z}_i^t)$
      $L^S(\theta) = \mathbb{E}_{(\mathbf{z}_i^t, r_i^t, \pi_i^t, V_{target}^t(\mathbf{x}_i^t), \hat{A}_i^t) \sim D_b} [S[\pi_\theta](\mathbf{z}_i^t)]$
      //Total PPO objective
      $L^{PPO}(\theta, \phi) = L^O(\theta) - c_1 L^{VF}(\phi) + c_2 L^S(\theta)$
      Update policy param. θ with lr_θ by Adam [66] with respect to $L^{PPO}(\theta, \phi)$
    end for
  end for
  //Adapt KL penalty coefficient
  if $KL[\pi_{old} \mid \pi_\theta] > 2KL_{target}$ **then**
    $\beta \longleftarrow 1.5\beta$
  else if $KL[\pi_{old} \mid \pi_\theta] < 0.5KL_{target}$ **then**
    $\beta \longleftarrow \beta/2$
  end if
end for

Algorithm 2 WW2C framework

Inputs: Number of robots n. Starting and goal positions and initial velocities: $\{\mathbf{p}_i^0, \mathbf{p}_{i,g}, \mathbf{v}_i^0\}, \forall i \in \mathcal{I}$. Training/Testing communication policy. n number of robots. Episode length: T_e. Maximum velocity: $\mathbf{v}_{max}$.
Initialize n robots in their initial positions and velocities. Copy an instance of the learned/still in training communication policy π_θ to all robots.
Initialize within each robot a set of last communicated trajectories $\{D_j\}_{j\neq i}$. Obtain the first observation of the environment $\{\mathbf{z}_i^0\}_{i\in\mathcal{I}}$
for time step t = 0,1,...,T_e **do**
  for robot i = 1,2,...,n **do**
    //Compute $\pi_i^t = \{c_{j|i}^t\}_{j\neq i}$
    if training **then**
      $\pi_i^t \sim \mathcal{B}(\pi_\theta(\mathbf{z}_i^t))$
    else
      $\pi_i^t = \mathbb{1}[\pi_\theta(\mathbf{z}_i^t) > 0.5]$
    end if
    //Compute $\hat{\mathcal{T}}_{j|i}^{t-1}$
    for robot $j \neq i, i = 1, ..., n$ **do**
      if $c_{j|i}^t == 1$ **then**
        //Traj. intention requested
        $\hat{\mathcal{T}}_{j|i}^{t-1} = \mathcal{T}_j^{t-1}$
        $D_j \longleftarrow \mathcal{T}_j^{t-1}$
      else if not training **then**
        //Traj. int. predicted (alg. 3)
        $\mathcal{T}_j^{t-k-1} \longleftarrow D_j, k > 0$
        $\hat{\mathcal{T}}_{j|i}^{t-1} = pred(\mathcal{T}_j^{t-k-1}, \mathbf{v}_j^t, \mathbf{p}_j^t)$
      else
        $\hat{\mathcal{T}}_{j|i}^{t-1} = \emptyset$
      end if
    end for
    $\{\mathcal{T}_i^t, \mathbf{u}_i^t\} \longleftarrow$ Solve eq.2.18 with $\mathbf{g}_i = \mathbf{p}_{i,g}$
  end for
  $\{\mathbf{z}^{t+1}\}_{i\in\mathcal{I}} \longleftarrow$ Step$(\{\mathbf{u}_i^t\}_{i\in\mathcal{I}})$
end for

Algorithm 3 Informed constant velocity estimator of robot j

Inputs: Last communicated trajectory at time step $t-k$: $\hat{\mathcal{T}}_{j|i}^{t-k-1}$, j's current velocity and position: $\mathbf{p}_j^t$ $\mathbf{v}_j^t$. Hyperparameters in table 2.2.
reject = 0
if $k \geq N-1$ **then**
    reject = 1
else if $\left\| \hat{\mathcal{T}}_{j|i}^{t-k-1}(k) - \mathbf{p}_j^t \right\| > r_{tol}$ **then**
    reject = 1
end if
if reject **then**
    $\hat{\mathcal{T}}_{j|i}^{t-1} = \{\hat{\mathbf{p}}_j^{t+l} \mid \hat{\mathbf{p}}_j^{t+l+1} = \hat{\mathbf{p}}_j^{t+l} + \Delta t \mathbf{v}_j^t, l = 0, ..., N-1; \hat{\mathbf{p}}_j^t = \mathbf{p}_j^t\}$
else
    //Expand the tail of the last communicated trajectory. Note that we consider: $\hat{\mathcal{T}}_{j|i}^{t-k-1}(0) = \mathbf{p}_j^{t-k-1}$
    $Tail = \{\hat{\mathbf{p}}_j^{t+l} \mid \hat{\mathbf{p}}_j^{t+l} = \hat{\mathcal{T}}_{j|i}^{t-k-1}(k+1+l), l = 0, ..., N-k-2\}$
    $step_{cte\_v} = \hat{\mathcal{T}}_{j|i}^{t-k-1}(N-1) - \hat{\mathcal{T}}_{j|i}^{t-k-1}(N-2)$ *//Cte. velocity step at the end of the tail*
    $\hat{\mathcal{T}}_{j|i}^{t-1} = Tail \cup \{\hat{\mathbf{p}}_j^{t+l} \mid \hat{\mathbf{p}}_j^{t+l+1} = \hat{\mathbf{p}}_j^{l} + step_{cte\_v}; l = N-k-1, ..., N-1; \hat{\mathbf{p}}_j^{t+N-k-2} = \hat{\mathcal{T}}_{j|i}^{t-k-1}(N-1)\}$
end if
return $\hat{\mathcal{T}}_{j|i}^{t-1}$

2.5 Simulation Experiments

In this section we first describe our implementation of the proposed method. Next, we provide a thorough evaluation of our learned communication policy by comparing it with our previous approach [20] and other communication baselines in several scenarios requiring increasing cooperation efforts to navigate safely.

2.5.1 Training Setup

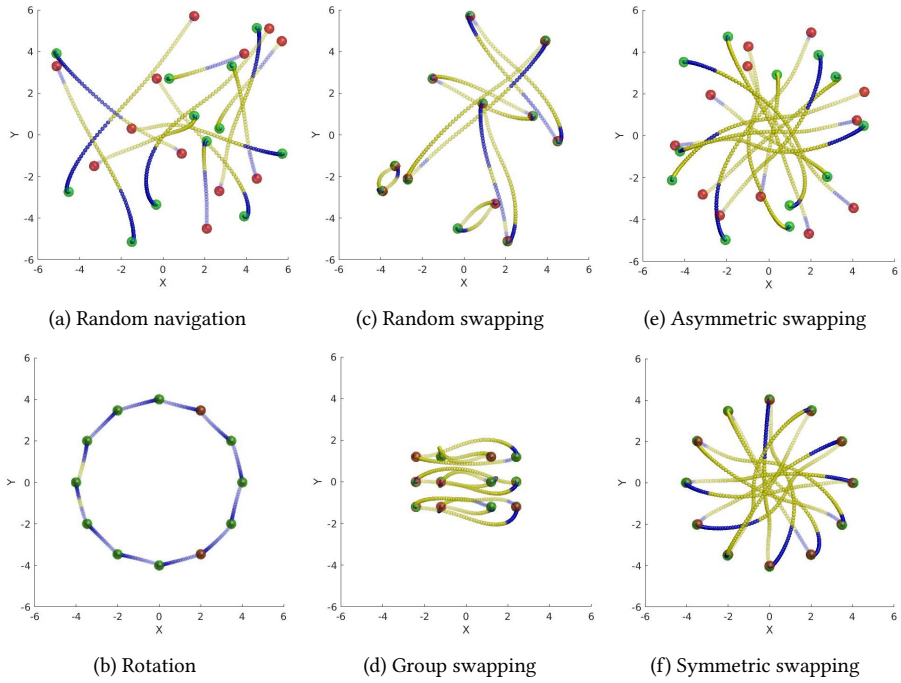
We train our communication policy for a team of twelve quadrotors moving in $\mathbb{R}^3$ and following Parrot Bebop 2 dynamics. We rely on the solver Forces Pro [68] to generate optimized NMPC code and its corresponding Python wrapper. As in [69], the time step used in the NMPC is 0.05s and the prediction horizon is $N = 20$ (1 second ahead). The constraints are formulated as soft-constraints to ensure the feasibility of the problem, and the solver iterations have been limited to 600 to have at least a control frequency of 20 Hz. Note that the framework is agnostic to the choice of solver as long as it allows a control frequency of 20 Hz. The learning algorithm and the training of our policy were implemented in Tensorflow, using the RLlib framework [63]. The Critic and Actor models follow the architecture shown in section 2.4.2 and were trained for 37200 episodes using an Intel i9-9900 CPU@3.10GHz computer. The hyperparameters used for training are explained in the Table 2.1. The simulation time step is set to 0.05s, which is the robot's control period. The quadrotors' dimensions are represented by a sphere of radius $r = 0.3m$ and their maximum speed is $v_{max} = 4.25m/s$. Computing both the communication policy and the MPC control inputs takes less than 0.01s per robot for each time step, which allows for a real-time implementation of the framework with a control and communication frequency of 20Hz. No noise is added into the simulation environment during the training process, in order to optimize the policy with low variance. Values for the reward weights were $w_g = 10$, $w_{coll} = 10$, $w_c = 10$. Tuned reward and penalty terms were $r_g = 1$, $r_{coll} = 1$ and $N_i(n) = 100(n-1)$. Goal reward is only received once during the episode. Episodes are finished after reaching 100 time steps or when all agents reach the goal.

2.5.2 Baselines

We introduce and compare our method with four other commonly used heuristic communication policies:

- *Full communication* (FC): At each time step each robot broadcasts its trajectory plans.
- *No communication* (NC): The robots never exchange their trajectory plans and a Constant Velocity model is used by each robot to infer the others trajectories.
- *A distance-based* communication policy (ϵ-DBCP): If the distance between two robots distance is smaller than a threshold ϵ (in meters) then the agents broadcast their trajectory information. $\epsilon \in \{4.25m, 8.5m\}$, which is once and twice the maximum distance within planning horizon, respectively.

Full communication and *no communication* policies give us a reference on what are the expected maximum and minimum performances in terms of safety and communication requests. On the one hand, since *full communication* policies allow each robot to request



(a) Random navigation (c) Random swapping (e) Asymmetric swapping

(b) Rotation (d) Group swapping (f) Symmetric swapping

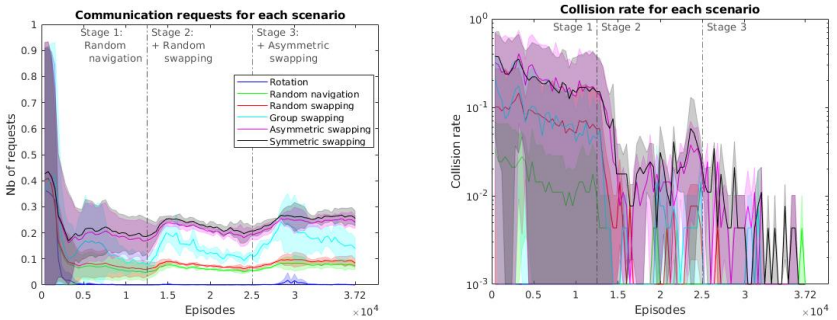
Figure 2.3: Simulation results for each scenario using our communication policy. The three figures on the left show the scenarios used for training while the three on the right are the ones used for testing. Solid lines represent the trajectories executed by the drone-swarm. Yellow represents the positions where the drones communicate their trajectory plans. Blue depicts the positions where the drones do not communicate. Green and Red represent the initial and goal position of each drone, respectively. Increasing opacity represents the episode progression.

trajectory intentions from all robots at every time step, we can consider it to be an over-conservative communication policy. Thus, if safe navigation is not achievable by applying *full communication* in a particular sampled episode, we can consider that it is difficult to find a better communication policy that can achieve collision avoidance for this particular configuration of robot initial positions and goals. On the other hand, *no communication* policies provide a reference on the expected minimum performance of the framework when only constant velocity estimations are used to predict other robots' future trajectories. Other baselines (ϵ-DBCP) give us a sense of our learned method's efficiency and safety in comparison with hand-crafted, reasonable and strong heuristics. The MPC motion planner is implemented with the same parameters for all baselines.

2.5.3 Testing Scenarios

To evaluate and compare our method with the baselines we design scenarios where we can evaluate how communication policies adapt to different levels of interaction. Therefore, aside from the scenarios used for training (see Sec. 2.4.3), we define three additional ones:

- **Rotation (Figure 2.3b)**: All drones are arranged in a circle and must rotate one



(a) Evolution throughout training of the learned policy's number of communication requests.

(b) Evolution throughout training of the learned policy's collision rate. Since the y-axis is in logarithmic scale, the 0 value is represented by 10^{-3}.

Figure 2.4: Policy evolution in each scenario throughout the different training stages of curriculum learning. We train three seeds and evaluate them every 400 training episodes for 100 episodes. We show the evolution of the mean and standard deviation of the number of communication requests and collision rate in each of the presented scenarios. We use the results obtained for the full communication policy, which represents the maximum communication possible, to normalize the number of communication requests of the learned policy between [0,1].

position either clockwise or counter-clockwise. This is a control scenario where no communication should be necessary. Therefore it allows us to evaluate the adaptability and communication efficiency of the policy.

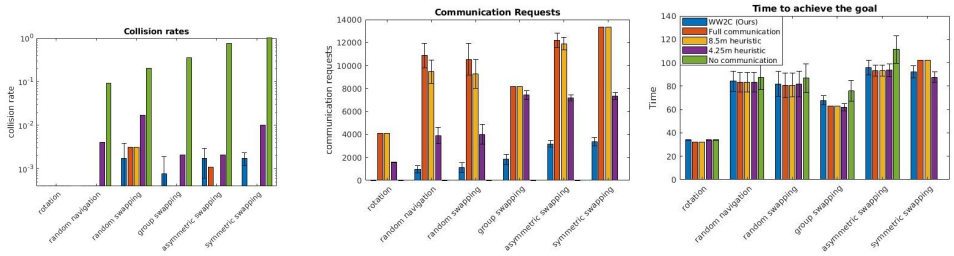
- **Group swapping (Figure 2.3d)**: We arrange the twelve drones in two groups of six symetrically opposed. Then, each drone must swap positions with its symmetrical counterpart.
- **Symmetric swapping (Figure 2.3f)**: All drones are arranged in a circle in symmetrically opposed initial positions and swap places with the opposite drone. As with the asymmetric swapping scenario, communication is required for all drones to ensure collision-free trajectories.

All scenarios used for evaluation are depicted in Figure 2.3.

2.5.4 Performance Evaluation

We evaluate the performance of the proposed learned collision-avoidance policy in terms of its safety and communication efficiency. We present multiple performance metrics and then compare our method with the indicated baselines. The metrics are:

- *Collision rate:* Proportion of episodes where there has been a collision between any of the robots in the team.
- *Number of communication requests along the episode:* Total number of communication requests throughout the episode. In deterministic scenarios, where we are certain of the low or high need for cooperation, this metric will allow to discern the adaptability of each model.



(a) Proportion of episodes with at least one collision in each scenario.

(b) Number of communication requests in each scenario.

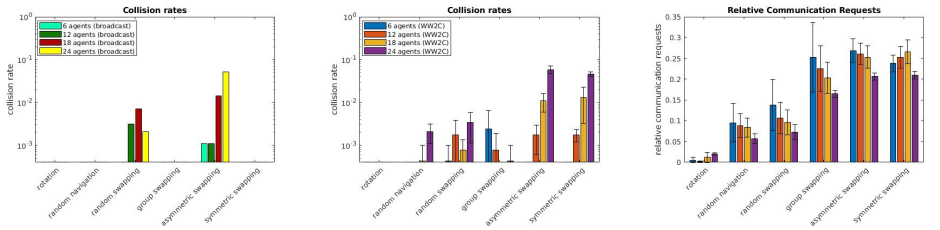
(c) Time steps needed to reach the goal in each scenario.

Figure 2.5: Performance evaluation for each scenario of the communication baselines and our learned policy. For our trained policy, we run three seeds and take their average performance and standard deviation. We run each method for 1000 episodes, gathering results for each metric. For collision rates, we show the proportion of episodes where we obtain at least one collision. For communication requests and number of time steps, we show the results for those episodes without collisions. In case none of the sampled episodes end without collisions, no bar is shown for that policy.

- *Time to achieve the goal:* Number of time steps needed to reach the goal. Failed episodes, where a collision has happened, are not accounted for when computing the mean and standard deviation.

Fig 2.4 shows the evolution of the learned policy throughout training in terms of average collision rate and number of communication requests for all scenarios. To account for the effect of different network initialization seeds into the final learned policy, we train three different initialization seeds and show the average and standard deviation of their performance in all our evaluations.
We normalize our results in communication requests between [0,1] using those obtained for the *full/no communication* policies for the same sets of sampled episodes for each scenario since this gives us a reference on the maximum/minimum values that we can score in both metrics. We can see that our method is able to learn an adaptable policy that makes close to no requests in simple settings such as *Rotation*, *Random navigation* and *Random swapping* scenarios while marginally affecting the number of collisions obtained in more difficult settings such as *Group swapping*, *Asymmetric* and *Symmetric swapping* scenarios. We can observe that our learned policy can adapt to the different amounts of communication that are required to achieve safe navigation. Note how at the end of training, the collision rate decreases drastically.

Figure 2.5 compares the learned communication policy against the proposed baselines using these metrics. The scenarios are ordered according to their levels of interaction. The results obtained for the *no communication* policy show correlation between the different complexity in scenarios and the need for communication. As shown in Figure 2.5a, the collision percentage of our method (WW2C) is the same or very similar as all other conservative baselines in all scenarios (0.4% difference at most). Even in the most complex scenarios, such as *asymmetric swapping*, we show that the difference in collisions is not significant in comparison with the safest communication policy: *full communication*. The clear advantage of our method is illustrated in Figure 2.5b, where WW2C shows better



(a) Collision rate over each scenario for *full communication* policy.

(b) Collision rate over each scenario for our method.

(c) Normalized number of communication requests over each scenario.

Figure 2.6: Results obtained when testing the *full communication* and our policy with 6, 12, 18 and 24 drones. Our method has been trained with 12 drones. Showed communication requests are have been scaled using the mean communication requests of the *full communication* policy under each scenario/number of agents.

results by communicating less than 50% and 30% in comparison with *4.25m heuristic* and the *full communication* in the worst cases. We show that the learned policy can adapt better to scenarios of different complexities since there is also a clear correlation between the amount of communication requested at each scenario and the expected need for cooperation shown in Figure 2.5a.

Due to our setup not allowing our drone to stop instantaneously, it will collide rather than run into deadlocks. Therefore, the success rate for any method is equivalent to one minus its collision rate. We show that the learned communication policy still manages to succeed in practically all scenarios with little effect in the time it needs to achieve the goal in comparison with the baselines. Our reward function accounts for achieving the goal only at the end of the episode. This is due to our main priority being to decrease the amount of communication while maintaining safety and avoiding deadlocks. We find that, although our reward function does not motivate achieving the goal as fast as possible, the sacrifice in terms of additional time steps is not significant.

2.5.5 Robustness and Zero-Shot Generalization Capabilities

Our approach allows to obtain a policy that is capable to generalize to an arbitrary number of robots in the environment, and to scenarios requiring different levels of interaction.Thus, we demonstrate the generalization of the learned communication policy with a series of experiments.

Lower/Larger scale multi-robot systems

We evaluate the performance of our method trained with 12 agents, on scenarios with a higher/lower number of agents. More specifically, in Figure 2.6, we show the obtained results from simulating 6,12,18 and 24 robots in each of the training and testing scenarios for 1000 episodes in comparison to the performance shown by the *full communication* policy under the same conditions.

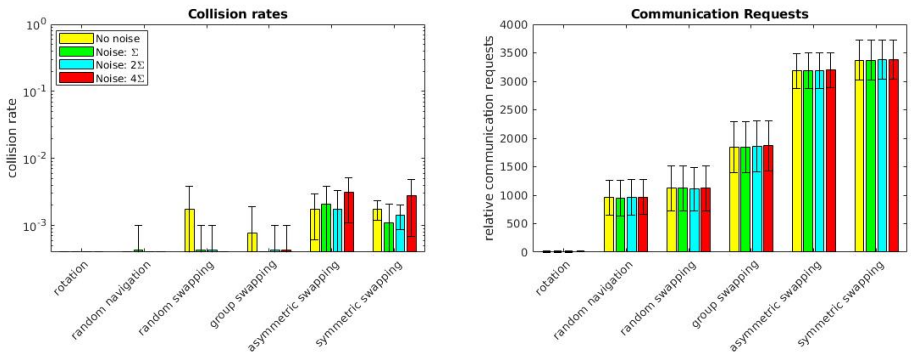
We show that our method is able to communicate at least 70% less in comparison with the *full communication* policy, while still being capable to adapt to scenarios requiring different levels of interaction. Note that the normalized communication requests for each scenario does not change significantly with the number of agents, even showing a

decreasing tendency when scaling up to 24 agents. This indicates that our communication policy generalizes well to environments with additional agents.

Regarding the obtained collision rates, our method generalizes well and shows better results when there are fewer robots in the environment. There is a degradation of performance when the number of agents in the environment is higher than seen during training. However, the degradation obtained for our method is low (i.e., less than 2% for 18 agents, and less than 10% for 24 agents) and is similar to the degradation seen when using *full communication* for the same numbers of agents.

Noisy positions and velocities

We also evaluate the robustness of our communication method under different levels of noisy inputs. We add a multiple of a gaussian noise to the other agents' relative positions and velocities in our observation vector $\mathbf{z}_i^t$ to simulate the effects of sensor measurement errors and localization uncertainties on our learned communication policy. The added measurement noise is zero mean with covariance: $\Sigma = \text{diag}(0.06\text{m}, 0.06\text{m}, 0.06\text{m})^2$. We simulate 1000 episodes for each scenario under three levels of noise: Σ, 2Σ, 4Σ. In figure 2.7, we show that our method is robust to these different levels of the added measurement noise since both performance and behaviour in terms of collision rates and communication requests suffer non-significant changes. In fact, note how the collision rates for each level of noise remains very low (lower than 1%) and similar to the other results.



(a) Collision rate over each scenario.

(b) Number of communication requests over each scenario.

Figure 2.7: Results obtained when testing our policy in presence of noise of the observed robot's positions and velocities. Standard deviations are taken over the results taken when evaluating 3 seeds of the trained policy over 1000 episodes.

2.5.6 Ablation Study

We analyse the key design choices we have introduced in this paper in comparison to [20]. Two main changes that we introduce are a model architecture that is able to function with an arbitrary number of robots, and a difference in conditions between training and testing regimes to obtain more robust and adaptable communication policies. Overall, these two changes allow us to learn policies that can decide better *when* and *with whom* to

communicate. We also changed the reinforcement learning algorithm from MADDPG [37] to PPO [14] with parameter sharing [49]. This was necessary as MADDPG requires one state-action value function per agent, which scales badly with the number of agents and tends to learn specialized agent roles that are situation-specific. PPO, on the other hand, only requires one state-value function for all agents and can learn the same communication policy for all agents using parameter sharing as well, avoiding over-specialization to specific scenarios.

We perform an ablation study to justify the modifications applied to the previous approach [20]. First, we will address the implementation of a targeted scalable attention-based architecture to encode the information of the dynamic environment. Second, we will empirically justify our decision of disabling informed constant velocity estimations whenever there's no communication during training.

Evaluation metrics and output models

We compare the different ablations across the same scenarios mentioned in section 2.5.4 showing the evolution of their training according to the following metrics (sorted in descending priority order):

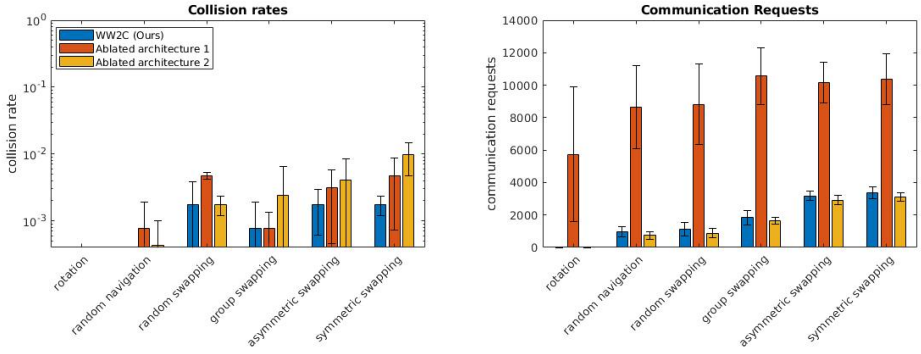
- *Collision rate:* Defined in section 2.5.4.
- *Number of communication requests:* Defined in section 2.5.4.

Similar to our method in section 2.4.2, each ablated communication policy outputs a normalized 2-dimensional vector $[p_{j|i}, 1-p_{j|i}]$ for each other robot j. This vector represents a Bernoulli distribution $\mathcal{B}(p_{j|i})$. The communication policy is stochastic if it samples such distribution to decide whether to request robot j's trajectory intentions ($c_{j|i} \sim \mathcal{B}(p_{j|i})$). The policy is deterministic if it decides to communicate by comparing the mean of the distribution to a predetermined threshold ($c^t_{j|i} = \mathbb{1}[p_{j|i} > 0.5]$). For fair comparison, we evaluate every ablation both as a stochastic and deterministic policy and show the one obtaining the best performance across all different evaluation scenarios.

Scalable attention-based architecture

One of the main limitations of our previous approach [20] is its difficulty to scale to multi-robot teams larger than four quadrotors, let alone react to an arbitrary number of robots in the environment. Similar to [44] and [62], in this work we address this challenge by incorporating three layers of transformer blocks into the core of the network architecture to encode the environment, which allows to provide a communication action for an arbitrary number of other robots. In Figure 2.8, we compare our approach with two ablated versions.

Ablated architecture 1 concatenates all the encoded vectors from other robots and substitutes the transformer block by three 64 neuron fully-connected layers with a ReLu activation. The decoder layer maps the resulting hidden layer to a vector of $2(n-1)$ communication scores. This ablated version cannot be used for a different number of agents than in training. The aim of this ablated version is to showcase the benefits of using each other drone individual information while using attention mechanisms to encode the state of the environment and compute each communication signal. In Figure 2.8, we show that the architecture used in this work is able to scale better to larger multi-robot systems



(a) Collision rate

(b) Number of communication requests

Figure 2.8: Performance evaluation over the ablated versions presenting different policy architectures. We add the performance of our own method for comparison. Standard deviations are taken over the results taken when evaluating 3 seeds of each trained policy over 1000 episodes.

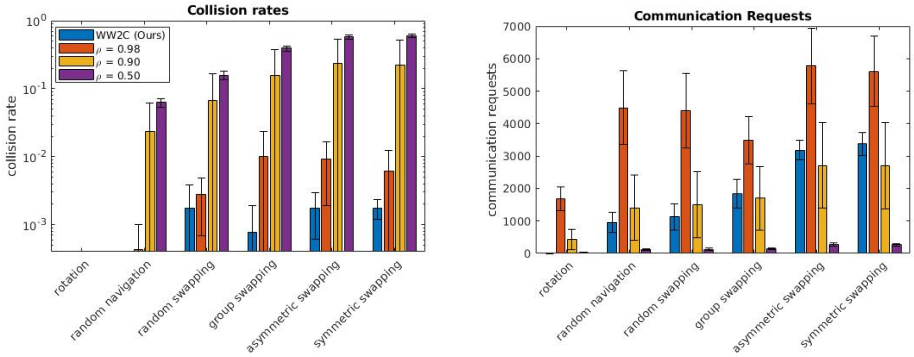
both in terms of collision rate and number of communication requests across episodes. These results also remark the importance of precise communication. A higher amount of communication requests do not necessarily translate to a safer communication policy.

Ablated architecture 2 also replaces the transformer block with three fully-connected layers of 64 neurons with ReLu activation functions. However, it processes each robot's information individually to decide whether to communicate or not. This network solves a simpler problem than the precedent ablated version since it learns to communicate with another other robot by only considering the information on its distance, relative position and velocity. Surprisingly, Figure 2.8 shows that this second ablation performs similar to our attention-based architecture. The intuition behind these results is that, at least in the tested scenarios, the individual relative information of every other robot contains most of the information that is relevant to decide whether to communicate with it or not. Although our method seems to result in slightly less collisions in all scenarios, we cannot draw a solid conclusion on this since the performance is not significantly different. In fact, while it is logical that there should be situations where the attention module would add a clear advantage over the pairwise communication ablation, it seems difficult to identify and reproduce these scenarios.

Training-test environment separation

As explained in section 2.4.3, distinction between test and training regimes was applied to increase the amount of collision experiences and increase the causality between lack of communication and collision events. The result of doing this is an efficient learned communication policy that is able to adapt to different scenarios with variating levels of interaction and still be practically as safe as *full communication* policies.

However, it could be unclear whether the same results could still be achieved by finding the right weighting trade-off between collision event and communication event penalizations. To verify this, we modify the reward function by adding a weighting variable ρ, as shown in Equation 2.19, and attempt to fine-tune it by training three models under



(a) Collision rate

(b) Number of communication requests

Figure 2.9: Performance evaluation over the ablated versions trained while enabling the prior information predictions. We add the performance of our own method for comparison. Standard deviations are taken over the results taken when evaluating 3 seeds of each trained policy over 1000 episodes.

different values for it: $\rho = \{0.98, 0.90, 0.50\}$. Informed constant velocity estimations are enabled during training. Note that $\rho = 0.5$ results in the original reward function proposed in section 2.4.1.

$$\begin{aligned} R_i(\boldsymbol{x}^t, \pi_i^t) = w_g R_{g,i}(\boldsymbol{x}^t) \\ + 2\rho w_{coll} R_{coll,i}(\boldsymbol{x}^t) + 2(1-\rho) w_c R_{c,i}(\pi_i^t) \end{aligned} \tag{2.19}$$

The values of ρ were chosen to showcase how difficult and counter-intuitive it is to properly balance communication and collision penalties when the training and test environment are the same. Rather than a wide range of parameters, Figure 2.9 shows which values of ρ are necessary to obtain a policy that matches ours in terms of collision rates ($\rho = 0.98$), communication requests ($\rho = 0.90$), and what happens when we balance both objectives equally ($\rho = 0.50$).

In Figure 2.9, we compare our method with different versions of the ablated model trained under the given different values for ρ. Increasing the value of the ρ hyperparameter results in learning more conservative communication policies that make more communication requests to navigate more safely. We can argue that fixing such value around $\rho = 0.90$ yields similar performance to our method as it is just slightly lower in terms of communication requests in the most complex training scenario: *Asymmetric swapping*. However, we show crucial differences in terms of fine-tuning difficulty, adaptability and reliability of the learned policy. In rotation scenarios, we should learn to decrease communications as no interaction is needed to perform safe navigation in this setting.

Note how the value of ρ has high impact on the converged policy for the ablated versions. In particular, their collision rate and overall communication amount throughout all scenarios variate greatly as shown in Figure 2.9. In contrast, our method allows us to decrease the number of communications while hardly compromising safety by just applying a simple modification.

Additionally, ablated versions fail to adapt to different scenarios. Policies trained with high values for ρ (≥ 0.90) tend to over-communicate, requesting other robots' trajectory intentions even when both of them are not moving. Instead, for lower values of ρ (≤ 0.90 and specially ≤ 0.50), learned policies tend to under-communicate as they rely too much on the predicted future trajectories which compromise their collision rate. In particular, a balanced value of $\rho = 0.5$ that equally punishes collisions and communication already results in close to 0 communication requests and high collision rates. We won't get any further interesting results from lower ρ values since that would mean a policy even closer to the *no communication* policy baseline. Enabling informed constant velocity estimations during training results in learned policies that leverage how much they can rely on informed constant velocity estimations. In practice, this means that we have a stochastic policy that controls the expected frequency of trajectory intention requests. While it is another valid strategy, it lacks adaptability and is less reliable and intuitive in complex scenarios. This explains why policies trained under high values for ρ tend to overcommunicate in all scenarios (Figure 2.9).

2.6 Real Experiments

In this section, we demonstrate that our communication policy learned through reinforcement learning in simulation can be deployed on physical quadrotors. In the following subsections, we first briefly introduce the hardware setup of our framework. Then, we present the multi-robot scenarios used for evaluation.

2.6.1 Hardware Setup

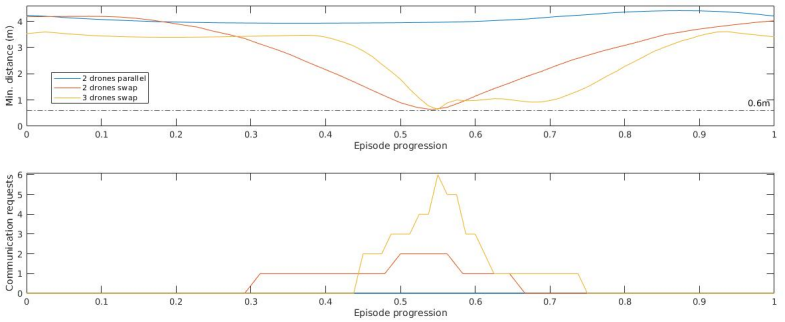
As in [30], our experimental platform is the Parrot Bebop 2 quadrotor. The radius of each quadrotor is set as 0.30m in the MPC. An external motion capture system (Optitrack) is used to measure the pose of each quadrotor, which provides an estimated pose for each quadrotor. We then use an UKF to estimate the state of quadrotors [30]. We use an Intel i7 CPU@2.6GHz computer for the communication policy and planner and use Robot Operating System (ROS) to send commands to the quadrotors. The communication policy and the NMPC configurations are explained in section 2.5.1.

2.6.2 Multi-Robot Scenarios

In this section, we design three scenarios to validate that the behaviors learnt in simulation during training (Figure 2.3) can be reproduced in real multi-drone teams. These experiments have been designed to showcase the adaptability of the communication policy to different amount of drones and to motion planning tasks requiring different amounts of interaction.

First, in Figure 2.10a we let two drones follow parallel trajectories (analogous to the rotation scenario in 2-drone settings) to verify that the learned communication policy does not communicate when it is not necessary. Additionally, we let them swap positions (Figure 2.10b) to demonstrate that two robots can reliably avoid each other using this framework and to verify the adaptability of our learned communication policy to different situations along the episode (i.e. they do not communicate unless needed to avoid collisions). Finally,

(a) Parallel trajectories (b) Swapping positions (2 drones) (c) Swapping positions (3 drones)



(d) Distance from each drone to its closest neighbour and the number of communication requests at each time step

Figure 2.10: Validation of our trained policy under different scenarios in real experiments.

we add a third robot to the environment and let them perform the swapping scenario (Figure 2.10c). Note that the communication policy is the same for all robots and does not need retraining when their number changes.

In Figure 2.10d, we plot the minimum distance among drones and the number of communications registered along these three scenarios. We show that in all three cases, our framework manages to avoid collisions with a minimal number of communication requests, and to adapt to a different number of robots without retraining or tuning the parameters from the NMPC.

To show the relationship between communication requests and distance holds even in real environments, we perform 9 swapping experiments with two drones (Figure 2.10b) while keeping record of the distance between them and the number of communications. Although there are overlaps among the distance distributions, the box plot in Figure 2.11 shows a clear relationship between the distance between drones and the number of communications. Note that the outliers in the 0-communication distribution and the overlaps between boxes could be due to the fact that the learned communication policy does not behave symmetrically in space. As seen in Figure 2.11, this means that the two drones do not necessarily communicate at the same time and does not behave equally before and

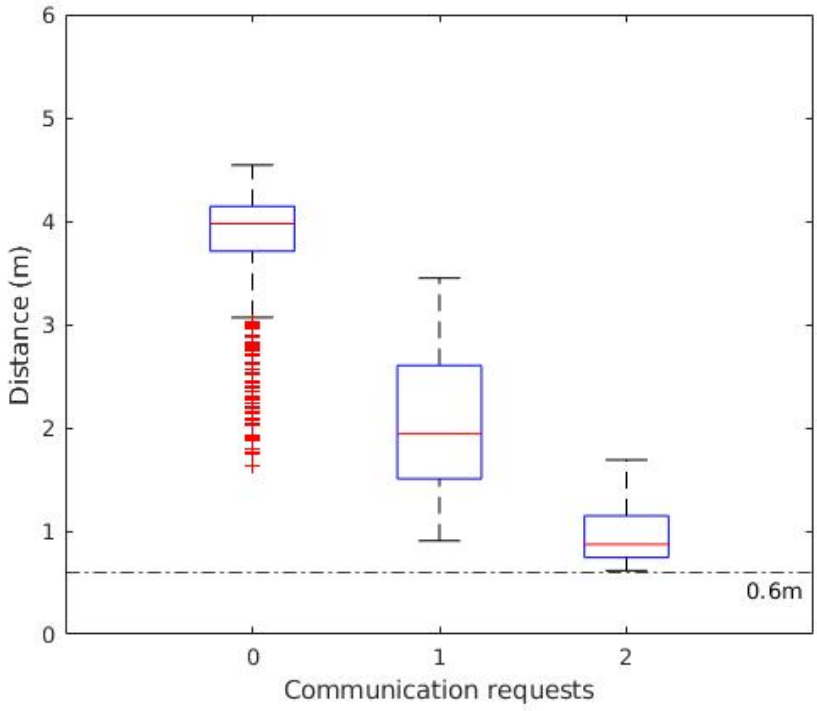


Figure 2.11: Box plot on the two-drone distance distribution for different levels of communication. The dotted line indicates the collision distance.

after the intersection (Figure 2.10d). Our results show therefore that the proposed learned communication strategy allows physical quadrotors to navigate tight situations with lower communication requests to avoid collisions.

2.7 Conclusions

In this paper, we have introduced an efficient communication policy integrating the strengths of MARL and NMPC in collision avoidance tasks. Simulation results show that our policy learns *when* and from *whom* to request planned trajectories to successfully avoid collisions. Experimental results show that the learned communication policy can be deployed on physical quadrotors. Further testing and the extension of our method to heterogeneous multi-robot systems is left for future work. Our method reduces the amount of communication requests significantly while achieving collision-free motions, practically achieving the same safety as more conservative communication baselines. The analysis and extension of our method under imperfect and delayed communication conditions are also left for future work. In comparison to [20], we use an architecture that enables us to scale our approach to higher and varying number of agents during and after training. Furthermore, we introduce a training method which allows to learn safe policies

without sacrificing adaptability. Future work will investigate how to prioritize episodes from scenarios which are rich in information to improve sample efficiency. Finally, our learned communication policy can only influence and coordinate the motion planning of each robots to a certain extent. It can only choose when additional information is needed to generate safe trajectories, but cannot modify this information nor modify the plans generated by the NMPC directly. Learning how to modify the information and/or plans generated by the NMPC to compensate for a lack in accuracy of our model is left for future work as well.

3

Active Classification of Moving Targets with Learned Control Policies

The work of this chapter has appeared in:

Á. Serra-Gómez*, E. Montijano, W. Böhmer, J. Alonso-Mora, *Active Classification of Moving Targets With Learned Control Policies*, in IEEE Robotics and Automation Letters (RA-L), vol. 8, no. 6, pp. 3717-3724, 2023, doi: 10.1109/LRA.2023.3271508.

3.1 Introduction

In surveillance and tracking applications an autonomous drone may be tasked with collecting relevant information from multiple targets, e.g., recognize people with blue eyes. Recent deep learning approaches show excellent results at detecting and categorizing single and multiple elements with images [70] or LiDAR [71]. However, these methods are generally not enough for active classification with a mobile drone, which also requires planning of the drone's movement and reasoning over the targets' future behavior.

The use of deep learning perception algorithms for information gathering comes with its own challenges. On the one hand, the "black-box" nature of these algorithms makes it difficult to determine the position that would yield the most informative data for classification. On the other hand, the drone also needs to reason about the targets' movement and orientation, as well as the possible occlusions among them, to plan a trajectory that will reveal the most information.

To overcome these issues, the main contribution of this paper is a complete solution to the problem of active classification of multiple moving targets. Differently to previous approaches, our framework can handle *dynamic* targets without requiring an explicit observation model, e.g., using a black-box classifier. Our solution leverages Deep Reinforcement Learning (DRL) to train a control policy that recommends informative viewpoints using the relative position of the targets and their current class probability estimations. We also propose a novel attention-based permutation-invariant architecture for the DRL policy that generalizes to more targets that move differently from those seen during training.

Moreover, to simplify the training, the policy is abstracted from the low-level dynamics of the drone, which are instead considered inside a low-level MPC controller at test time. Finally, the estimations are updated with an efficient information fusion method, conflation, suited to be used with black-box perception algorithms. A full overview of the method is shown in Fig. 3.1. Experiments under different conditions show that our approach outperforms a variety of baselines and is robust to scenarios unseen during training.

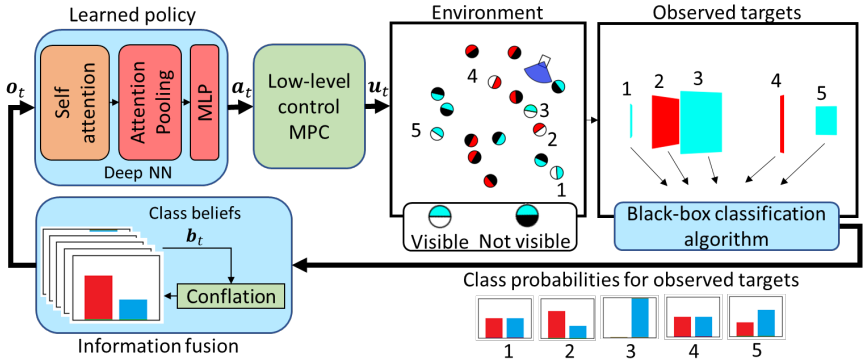


Figure 3.1: Overview of our method for active classification. The objective is to classify all targets into different classes, red and blue in the example. The method has three parts. First, observable targets, targets 1 to 5, represented with white back in the figure, are detected by the onboard sensor. We assume the existence of a classification algorithm, e.g., a CNN, able to provide probability estimates for each target (bars at the bottom right). Then, we use an efficient fusion mechanism, conflation, to obtain the beliefs, $\mathbf{b}_t$, over the classes for each target. Finally, this information, together with the observed probabilities and relative locations, $\mathbf{o}_t$, is fed to a control algorithm that computes the recommended viewpoint, $\mathbf{a}_t$, using a novel deep learning architecture trained using Reinforcement Learning. The recommended viewpoint is tracked with a low-level controller, which outputs the robot control input $\mathbf{u}_t$.

3.2 Related Works

Our contributions build upon recent work in multi-view active classification and learning for motion planning in dynamic environments.

3.2.1 Multi-view active classification

The problem of active classification is typically solved by pre-defining a set of viewpoints through which trajectories are planned to gather sufficient information to solve an active classification task. One-step greedy planners for active object classification [72] select object-dependent viewpoints based on class uncertainty and observation occlusions. Some non-myopic methods [73] account for the cost of movement and information gain between viewpoints to solve the problem of active classification. Others, e.g., [74], formulate the problem as a partially observable Markov Decision Process (POMDP) and plans a path over viewpoints while accounting for measurement costs, occlusions and possible misclassifications. Similarly, [75] computes plans using a variation of Monte Carlo tree search. However, these methods generally require access to a model to estimate viewpoint usefulness a priori.

Learning methods are non-myopic and allow to use black-box models. Recent works leverage the use of DRL for static multi-target pose estimation and active classification by learning a policy that moves the camera towards viewpoints that reduce observation uncertainty [76] or maximize information gain to enhance object classification [77]. Nevertheless, these methods assume that targets are static, which makes them unsuitable for active classification with moving targets. Although some works consider dynamic targets, they are often limited to a pre-defined closed environment [78] or, as its the case for aerial videography methods [79, 80], focus on target information visibility which requires prior

knowledge on where this information is visible from.

3.2.2 Learning for motion planning in dynamic environments

Learning approaches in motion planning tasks have the potential to encode and identify patterns in complex motion planning tasks. Recent advances in Deep Representation Learning show promise in learning latent representations of the state space, capturing the underlying structure and symmetries in dynamic environments [81]. Recent works in learning-based motion-planning policies rely on Convolutional Neural Networks (CNN) [82, 83] for visual-based navigation, or encode the state of multiple static/dynamic elements in the environment using Graph Convolutional Networks (GCN) [45, 84] or Long-Short Term Memory layers (LSTMs) [44, 60].

Yet these strategies require a priori knowledge on the priority order of the elements in the encoded sequence, or the structure of the encoded graph [62]. Instead, DeepSets [85] and self-attention based architectures [17, 86] are permutation invariant and enable encoding element sets without making further assumptions on their structure. Most related to our approach, [10] uses a combination of Self-attention [17] and DeepSets [85] to learn a policy that coordinates a multi-robot system to track a set of dynamic targets. However, DeepSets aggregate all targets' information by assigning equal weights to them. In contrast, we leverage the use of self-attention [17] and attention-based set-function approximators [87] to effectively encode the dynamic environment, learning the importance of each target in our future decision.

3.3 Background

3.3.1 Problem formulation

Consider a drone with state $\mathbf{x}_t$ at discrete time t, control inputs $\mathbf{u}_t$ and dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, obtained for a sampling time of τ_l seconds. The drone has to collect information from a set $\mathcal{J} = \{1, ..., M\}$ of M dynamic targets, where $\mathbf{y}_t^j$ is the state of target j at time step t, represented in the drone's reference frame and $\mathcal{Y}_t = \{\mathbf{y}_t^j\}_{j=1,...,M}$. The targets follow their own dynamics, $\mathbf{y}_{t+1}^j = g^j(\mathbf{y}_t^j)$, which are unknown to the drone. We assume that the drone flies above the targets, neglecting physical collisions with them. Nevertheless, the targets can still collide with each other and, more importantly, they can occlude the visibility of others to the drone, depending on where they are. Besides, we do not deal with how to measure and track the targets' relative information, assuming they are provided by some external perception algorithm, e.g., [88]. For the sake of simplicity, in the following we omit the t subscript, except when needed.

We denote by $C = \{1, ..., C\}$ the set of classes, such that each target in $\mathcal{J}$ belongs to one class in C (e.g., eye color). The objective of the drone is to classify all the targets. To do that, the drone has a belief vector for each target, $\mathbf{b}^j = \{b^{j1}, b^{j2}, ..., b^{jC}\}$, where $0 \leq b^{jc} \leq 1$ denotes the belief the drone has of target j belonging to class c at time t, and $\sum_{c=1}^{C} b^{jc} = 1$. Target j is tagged as classified whenever $\max_{c \in C} b^{jc}$ is above a pre-defined threshold $b_{\max}$. We use the Boolean variable l^j, to specify whether target j is classified or not at timestep t.

To compute the beliefs, every $\tau_h \gg \tau_l$ seconds, the drone is able to make an observation and use a black-box perception algorithm (e.g., a pre-trained CNN classifier) to compute a probability distribution over the class set for each target. We denote that distribution

as $\mathcal{P} = h(\mathcal{Y}) = \{\mathbf{p}^j\}_{j=1,...,M}$, where $\mathbf{p}^j = (p^{j1}, ..., p^{jC})$ is the probability vector for target j, and p^{jc} is the probability of target j belonging to class c. For unobserved targets, $\mathbf{p}^j$ is a uniform distribution. As it happens with the majority of real CNN classifiers, we assume the drone has no available model to map how the relative positions relate to the observed probability distributions. Beliefs are then computed by fusing these measurements, $\mathbf{b}_t^j = \zeta(\mathbf{p}_{1:t}^j)$, where ζ is the conflation operator [89], a function that models how the beliefs can be computed from the history of classification probabilities given by the black-box sensor (see Section 4.5.1).

Under these conditions, the problem considered in the paper is to actuate the drone in such a way that it is able to classify all targets as quickly as possible, i.e., make l_t^j true for all j in the minimum possible value of t. To address it, we formulate a sequential decision-making problem that the drone solves at every time step t. The objective of this problem is to find the actions over a time horizon T that minimize the accumulated entropy of all the targets' beliefs,

$$\begin{aligned} \min_{\mathbf{u}_{0:T}} & \sum_{t=1}^{T} \sum_{j \in \mathcal{J}} w_{\mathcal{H}} \mathcal{H}[\mathbf{b}_t^j] + w_u \|\mathbf{u}_t\| \\ \text{s.t.} \quad & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad \mathbf{y}_{t+1}^j = g^j(\mathbf{y}_t^j), \quad \forall t \\ & \mathcal{P}_t = h(\mathcal{Y}_t), \quad \mathbf{b}_t^j = \zeta(\mathbf{p}_{1:t}^j), \quad \forall t \propto \tau_h/\tau_l \\ & j \in \mathcal{J}, \quad 0 \le t \le T-1 \end{aligned} \tag{3.1}$$

where $\mathcal{H}[\mathbf{b}_t^j]$ denotes the entropy of belief $\mathbf{b}_t^j$, $w_{\mathcal{H}}$ and w_u are scaling weights, and $\propto$ is the proportional sign.

3.3.2 Target class observations and Belief Updates

An important aspect to consider is how to aggregate the different observations made by the drone about each target's class to produce the class beliefs. The first issue to consider is that standard Bayesian recursive estimation is not advisable because the measurement likelihood model for the update, $\mathbb{P}(\mathbf{p}_t^j | \mathbf{b}_{t-1})$, is not available from a black-box sensor. To train and learn an accurate pose-dependent model of the likelihood, a dense dataset must be built first. Then, to use it for optimal viewpoint search all targets and their occlusions must be considered. This process is costly and scales badly.

Instead, in this paper we propose to use a mathematical method called conflation [89]. Conflation is used to aggregate probability distributions obtained from measurements over the same phenomena under different circumstances. Notably, this technique has the property of minimizing the loss of Shannon information when flattening multiple independent probability distributions into a single one, that is, computing $\mathbf{b}_t^j$ given the measurements $\mathbf{p}_{0:t}^j$. In addition, it is a commutative and associative operator, enabling easy and efficient recursive computation and making it appealing for onboard computation. The conflation is defined by

$$\mathbf{b}_t^j = \zeta(\mathbf{p}_{1:t}^j) \equiv \zeta(\mathbf{b}_{t-1}^j, \mathbf{p}_t^j) = \frac{\mathbf{b}_{t-1}^j \odot \mathbf{p}_t^j}{(\mathbf{b}_{t-1}^j)^\top \mathbf{p}_t^j}, \tag{3.2}$$

where the Hadamard product $\odot$ in the numerator is taken component-wise, whereas the dot product is the normalization factor. Beliefs are initialized at $t = 0$ with a uniform prior over classes in C.

Lastly, although Eq. (3.2) considers all the measurements equally, it can easily be extended to a weighted version using the weights as powers of the probability distributions. This could be useful for example in cases where the black-box also outputs a confidence measurement over the class probabilities.

3

3.4 Methodology

The lack of information about $h(\mathcal{Y}_t)$ and $g^j(\mathbf{y}_t^j)$ hinders the direct solution of problem (3.1). Instead, in the paper we leverage Reinforcement Learning to train a control policy that implicitly learns these quantities (Viewpoint Control Policy). The policy π_ϕ, parametrized by ϕ, operates at the perception low-frequency, $\frac{1}{\tau_h}$, and computes informative viewpoints $\mathbf{a}_t$ for the drone. The viewpoints are then tracked with an MPC controller that generates the low-level control inputs, $\mathbf{u}_t$, at the necessary higher frequency, $\frac{1}{\tau_l}$. A positive side-effect of this decomposition in two temporal abstraction levels is the possibility to neglect the complex drone dynamics in the POMDP formulation used for the RL algorithm. An overview of the proposed framework is given in Figure 3.1. To simplify the notation, in this section t denotes time periods of τ_h, whereas the faster time steps of periods τ_l are denoted by k in Section 3.4.2.

3.4.1 Viewpoint Control Policy

POMDP Formulation

We formulate the high-level viewpoint recommendation problem as a POMDP, which is defined by the tuple $\langle S, A, \mathcal{T}, Z, \mathcal{O}, R\rangle$. The states $\mathbf{s}_t \in S$ contain the drone's position $\mathbf{q}_t$ and yaw orientation ψ_t, all targets' states $\mathbf{y}_t^j$ and ground truth class, and the current beliefs $\mathbf{b}_t^j$ and $l_t^j, 1 \leq j \leq M$. The action space, A, models the recommended viewpoints, defined by displacements over the drone's position and yaw, $\mathbf{a}_t = (\Delta\mathbf{q}_t, \Delta\psi_t)$. Recommended viewpoints are constrained to a neighborhood and orientation from the current pose, which depends on the maximum distance and angle the drone can traverse in τ_h seconds. The transition probability function, $\mathcal{T}$, simply assumes that the drone is able to reach the output viewpoint in time for the next measurement. The drone observes partial environment information $\mathbf{o}_t \in Z$, according to the observation function $\mathcal{O}$. This observation is defined by $\mathbf{o}_t = \{\mathbf{o}_t^j\}_{j=1,\dots,M}$, where $\mathbf{o}_t^j := (\mathbf{y}_t^j, \mathcal{H}[\mathbf{b}_t^j], \mathcal{H}[\mathbf{p}_t^j], l_t^j)$ is the information available from target j regarding its relative pose, velocity, normalized entropy of the belief and the current measurement, and whether it has already been classified. The use of the belief and measurement entropy, instead of the probability distribution vector, enables handling an arbitrary number of classes C and keep track of how much information can still be gained by observing a target. We also recall that for unobserved targets, $\mathbf{p}_t^j$ is a uniform distribution.

Finally, the reward function, R, is shaped based on the problem described in Eq. (3.1), and additional factors that the policy should take into account. First of all, there is a dense reward, $R_\mathcal{H}$, proportional to how much the entropy of each belief, $\mathbf{b}_t^j$, has decreased each time step, t, due to the new information gathered. Additionally, there is one sparse reward,

denoted by R_l, for individual target classifications, when any l_t^j changes from zero to one at time t, and another, $R_{\mathcal{J}}$, for completing the classification of all of the existing targets, when $\sum_j l_t^j = M$. On the other hand, to promote solving the task quickly and efficiently, the reward includes a constant penalty associated to the time required to classify the targets, R_t, and another one proportional to the distance to the recommended viewpoint, R_a, to favor small motions. The formal definition of all the reward terms is

$$\begin{aligned} R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) &= R_{\mathcal{H}}(\mathbf{s}_t, \mathbf{s}_{t+1}) + R_l(\mathbf{s}_t, \mathbf{s}_{t+1}) \\ &\quad + R_{\mathcal{J}}(\mathbf{s}_{t+1}) - R_t - R_a(\mathbf{a}_t), \end{aligned}$$

$$\begin{aligned} \text{where} \qquad & R_{\mathcal{H}}(\mathbf{s}_t, \mathbf{s}_{t+1}) = w_{\mathcal{H}} \sum_{j=1}^{M} \Big(\mathcal{H}(\mathbf{b}_t^j) - \mathcal{H}(\mathbf{b}_{t+1}^j) \Big), \\ & R_l(\mathbf{s}_t, \mathbf{s}_{t+1}) = w_l \sum_{j=1}^{M} (l_{t+1}^j - l_t^j), \\ & R_{\mathcal{J}}(\mathbf{s}_{t+1}) = \begin{cases} w_{\mathcal{J}} & \text{if } \sum_{j=1}^{M} l_{t+1}^j = M \\ 0 & \text{otherwise} \end{cases}, \\ & R_t = w_t, \\ & R_a(\mathbf{a}_t) = w_a(\|\Delta \mathbf{q}_t\| + |\Delta \psi_t|), \end{aligned} \tag{3.3}$$

with $w_{\mathcal{H}}$, w_l, $w_{\mathcal{J}}$, w_t and w_a weights that scale each term. Note that both $\mathbf{b}_t^j$ and l_t^j are stored within $\mathbf{s}_t$.

Architecture

The ability of any learned policy $\pi_\phi(\mathbf{a}_t|\mathbf{o}_t)$ to generalize beyond the exact situations seen during training, e.g., more targets or changing target behaviors, depends crucially on the chosen neural architecture, shown in Figure 3.2. We arrange the information available of each target, $\mathbf{o}_t^j$, into a set $\mathbf{o}_t$. The main challenge arises from the set being large and changing over time. Inspired by Relational Graph Convolutional Networks [90] and self-attention mechanisms [17] used on static knowledge graphs, we implement a self-attention block (SAB) to identify the relationship between the poses of all targets, i.e., at time t. Thus, the first layer is,

$$\begin{aligned} & \tilde{\mathbf{e}}_i^{1,h} = F(\mathbf{o}_t^i; \mathbf{W}_{q,h}^1) + \sum_{j \in \mathcal{J}} \lambda_{i,j}^h F(\mathbf{o}_t^j; \mathbf{W}_{v,h}^1), \\ & \mathbf{e}_i^1 = LN(Res^1(LN(concat(\{\tilde{\mathbf{e}}_i^{1,h}\}_{h=1\ldots H})))), \\ & \lambda_{i,j}^h = \text{softmax}\Big(\frac{1}{\sqrt{d_h}} F(\mathbf{o}_t^i; \mathbf{W}_{q,h}^1)^\top F(\mathbf{o}_t; \mathbf{W}_{k,h}^1) \Big)_j, \end{aligned} \tag{3.4}$$

where $i \in \mathcal{J}$, $Res^l(x) = x + \sigma(F(x; \mathbf{W}^l))$, with σ being a ReLU activation function and F a parametric affine transformation. $\mathbf{W}^1 \in \mathbb{R}^{d_{enc} \times (d_h+1)}$ and $\mathbf{W}_{w,h}^1 \in \mathbb{R}^{d_h H \times (d_{in}+1)}, w \in \{v, q, k\}$, are

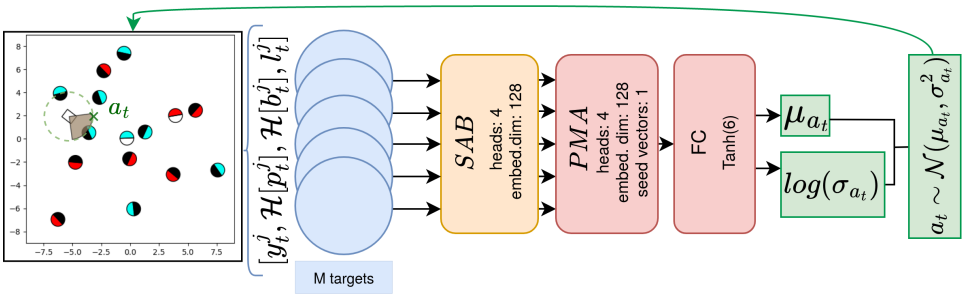


Figure 3.2: Policy neural network architecture. The sequence of target information is fed to the self-attention block (SAB) which encodes and identifies how each target pose affects the visibility of the others from the drone's perspective. This information is fed to the pooling multi-head attention layer (PMA) and mapped through a fully connected layer (FC) to obtain the parameters of a gaussian distribution. During training we sample the distribution to obtain the policy viewpoint recommendation. At test time we use the learned mean value instead.

learnable parameters. d_{in}, d_h, d_{enc} are the dimensionality of the input, each head h, and the first layer. Note that each head h encodes a different relation λ^h between targets. The purpose of this first layer is to encode information such as target visibility, perspective from which each target's information can be observed, occlusions, and possible simultaneous observations.

Next, we draw inspiration from Set Tranformers [87], used in static set-structured data, to aggregate the features of all latent target representations. We employ a pooling multi-head attention mechanism (PMA) where a learned seed vector per head $\mathbf{v}_s^h \in \mathbb{R}^{d_h}$ is employed to compute the attention weights for a single query,

$$\begin{aligned}
\tilde{\mathbf{e}}^{2,h} &= \mathbf{v}_s^h + \sum_{j \in \mathcal{J}} \lambda_j^h F(\mathbf{e}_j^1; \mathbf{W}_{v,h}^2), \\
\mathbf{e}^2 &= LN(Res^2(LN(concat(\{\tilde{\mathbf{e}}^{2,h}\}_{h=1\ldots H})))), \\
\lambda_j^h &= \text{softmax}\left(\left\{ \frac{1}{\sqrt{d_h}} \mathbf{v}_s^{h,\top} F(\mathbf{e}_j^1; \mathbf{W}_{k,h}^2) \right\}_{j \in \mathcal{J}} \right)_j .
\end{aligned} \tag{3.5}$$

This results in a latent vector $\mathbf{e}^2$ that is mapped, through a fully connected layer, to the parameters $\mu_{\mathbf{a}_t}$ and $\log(\sigma_{\mathbf{a}_t})$ of a diagonal Gaussian distribution $\mathcal{N}(\mu_{\mathbf{a}_t}, \sigma_{\mathbf{a}_t})$ over viewpoints. The learned policy π_ϕ samples recommended viewpoints $\mathbf{a}_t$ from this distribution. We use the Proximal Policy Optimization (PPO) algorithm to train the network [14, 63]. As learning algorithms like PPO also require an estimate of the state-value $V^{\pi_\phi}(\mathbf{s}_t) = \mathbb{E}[\sum_{t'=t}^{\infty} \gamma^{t'-t} R(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{t'} \sim \mathcal{T}, \mathbf{a}_{t'} \sim \pi_\phi]$, where γ is the discount factor, another linear layer predicts $V^{\pi_\phi}(\mathbf{s}_t) \approx \mathbf{v}_v^\top \mathbf{e}^2$. The latter is only used during training to guide the policy. We combine both the surrogate loss and KL-divergence term to stabilize training. We also use an entropy regularization term to encourage exploration [91]. We refer the reader to [14] for more information on the algorithm equations and details.

3.4.2 Low-level Controller

To account for the drone dynamics, the recommended viewpoint position $\mathbf{a}_t$ is tracked with an MPC low-level controller [92]. Let $\mathbf{x}_{at}$ denote the viewpoint state output by the policy in the world frame, obtained from the current drone state and setting to zero the information that is not considered in $\mathbf{a}_t$, e.g., roll and pitch. Every τ_l seconds, the controller solves the following receding horizon constrained optimization problem,

$$\begin{aligned} \min_{\mathbf{x}_{1:N}, \mathbf{u}_{0:N-1}} & \sum_{k=0}^{N-1} J^k(\mathbf{x}_k, \mathbf{u}_k) + J^N(\mathbf{x}_N, \mathbf{x}_{at}) \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{x}_t, \quad \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{u}_k \in \mathcal{U}, \quad 0 \leq k \leq N-1 \end{aligned} \tag{3.6}$$

where $\mathbf{u}_k$ is the low-level control input sent to the robot, that needs to be inside the possible values $\mathcal{U}$, $f(\mathbf{x}^k, \mathbf{u}^k)$ the internal dynamics and $J^k(\mathbf{x}_k, \mathbf{u}_k) = w_u \|\mathbf{u}_k\|$,

$$J^N(\mathbf{x}_N, \mathbf{x}_{at}) = w_g \frac{\|\mathbf{x}_N - \mathbf{x}_{at}\|}{\|\mathbf{x}^0 - \mathbf{x}_{at}\|}, \tag{3.7}$$

the stage and terminal costs, weighted by w_u and w_g respectively. For more details we refer the reader to [92].

3.5 Implementation Details

We train and test the proposed method both with simulated perception and with a real classifier in a photo-realistic simulator. The first environment has a simplified, computationally efficient observation model and is used for comparison between our method and the baselines introduced in Section 3.6.1. The second environment is used to test the proposed method under more realistic conditions.

3.5.1 Simulated Perception Environment

Observation Model

We consider a synthetic pinhole camera with focal length equal to 400 pixels that acquires images of 640×480 pixels. A target is modeled as visible if we can draw a line between its center and the drone's without any collision. For each target we consider only the half part of the cylinder that is facing forward and project it into the image frame, if it is visible from the camera perspective. This generates an image of a trapezoid with area and skew depending on the relative position and orientation of the target (Figure 3.1). We use these two parameters to determine the probability distribution of the observation over the class set, decreasing the probability of the true class exponentially with the skew and increasing it linearly with the area. We also penalize heavily trapezoids that do not fit in the image. Our method does not have any knowledge of neither the observation model nor the classification model, which makes them black-box to it without loss of generality.

Training conditions

Our high-level policy and the learned baselines are trained in closed environments of $50 \times 50\ m^2$ with simulation steps of $\tau_h = 0.25\ s$. As shown in Figure 3.1, targets are modeled

Table 3.1: Hyperparameters for PPO training algorithm

Parameter	Value
Time steps each update	16000
SGD minibatch size	256
Learning rate	3e-4
Entropy loss coeff. c_2	0.001
Gradient Clipping	0.1

as cylinders of 0.6 m radius, 1.8 m of height, with their class information only visible from the front. The targets follow constant velocity dynamics and belong to either class *red* or class *blue*. Targets' speed in every axis is sampled around 1 m/s and clipped at 1.5 m/s. Whenever targets collide among themselves or against walls, they rebound conserving kinetic energy and momentum.

The drone's maximum angular and linear speed in each axis is respectively $\dot{\psi}^{max} = 60^o/s$ and 2 m/s. This is to ensure that the drone can reach targets moving away from it. To reduce computation costs, only during training we assume that the drone follows first-order dynamics, and control its velocity to guide the drone to the recommended viewpoint.

Each episode, each method is given 100 seconds to classify all targets ($l_t^j = 1, \forall j \in \mathcal{J}$, $b_{\max} = 0.95$). Episodes are finished after reaching the timeout, or successfully classifying all targets. Values for the reward weights are $w_{\mathcal{J}} = 100$, $w_l = 5$, $w_{\mathcal{H}} = 1$, $w_t = 0.3$, $w_a = 0.01$.

3.5.2 Training Algorithm

The learning algorithm and the training of our policy are implemented using the RLlib framework [63]. Table 3.1 shows the hyperparameters tuned for training the PPO algorithm. We tuned these hyperparameters because they regulate the speed and quality of training and are more problem-specific. Other hyperparameters follow the default values provided by the framework. We refer the reader to [14, 93, 94] for a detailed analysis on the effect of these hyperparameters on the training algorithm and how to tune them. State space exploration is enhanced [95] to make the learned policy more robust at test time, by randomizing the drone's initial poses, and the targets' initial poses, velocities and beliefs over their classes. Some of the targets are randomly selected and enforced to remain static.

We follow a two-phase training schedule. First, we train the policy for the general case in scenarios, chosen randomly, containing between 1 and 12 targets. Then we fine tune the resulting policy by training it in scenarios containing only 1 to 6 targets. While counter-intuitive, this choice of schedule is motivated by how the task of active classification changes depending on the number of targets present in the environment. There are two motion planning tasks that need to be solved to perform active classification: simultaneous observations of multiple targets and, if that is not possible, then focus on single target high-quality viewpoints. We find that only learning both tasks simultaneously in environments with 1-12 targets results in a policy prioritizing simultaneous observations of multiple targets. Such policy scales up well but at the expense of poor choice of viewpoints when the number of targets is small. This is why a second training phase to fine-tune the policy to environments requiring single-target viewpoints is used.

The policy and value function (detailed in section 4.5.2) were trained for $9.6e7$ steps

in environments containing up to twelve targets, sampled uniformly (first phase). Then, they were further trained for $3.2e7$ steps in environments containing a random number between one to six targets (second phase). The training of the algorithm was done using an Intel i9-9900 CPU@3.10GHz computer.

Computing the policy actions takes less than $4ms$ per time step, which allows for a real-time implementation of the framework with a fast control frequency.

3.6 Simulated Results

We analyze our high-level policy's scalability and robustness to different target dynamics in comparison with other hard-coded and learned baselines.

3.6.1 Baselines

There are no existing approaches that address the problem of active classification of dynamic targets without relying on an available observation model. Therefore, the baselines used for comparison are:

- **Hand-crafted**: Hard-coded policy that guides the drone to a position 2 m in front of an unobserved target, pointing the camera at it.
- **Single-target (Ours)**: This is an ablation of our method. A learned single-target policy is given the information of an unobserved target and guides the drone to viewpoints allowing it to classify it as fast as possible. It is trained in scenarios with just one target.
- **LSTM encoder [44, 60]**: The self-attention and attention pooling layers are substituted by a linear layer followed by an LSTM. Every unclassified target is inversely sorted by proximity to the drone. The first layer encodes each target's information, and the sequence is fed into the LSTM.
- **DeepSets decoder [10]**: We replace the attention pooling layer by a mean pooling layer.

The latter two baselines are recently proposed scalable architectures employed in different, albeit similar [10], problems.

3.6.2 Test conditions

At test time, the whole pipeline described in section 3.4 is used, including the drone dynamics and controller described in section 3.4.2. We evaluate our method's robustness under target behavior both seen and unseen during training. Aside from constant velocity dynamics, we test our policy in environments where targets follow social forces pedestrian dynamics [96]. As in training, each method is given up to 100 seconds to classify all targets. Each policy is trained with 5 different seeds. The results of all seeds are averaged and shown with their standard deviations. For every test, we evaluate and average each method's results over 50 episodes. Initial conditions of all evaluation episodes are randomised but maintained equal across our method and all baselines.

Since the first two methods are designed to classify one target, the extension to multiple targets is done through their sequential classification. Unclassified targets are ordered by

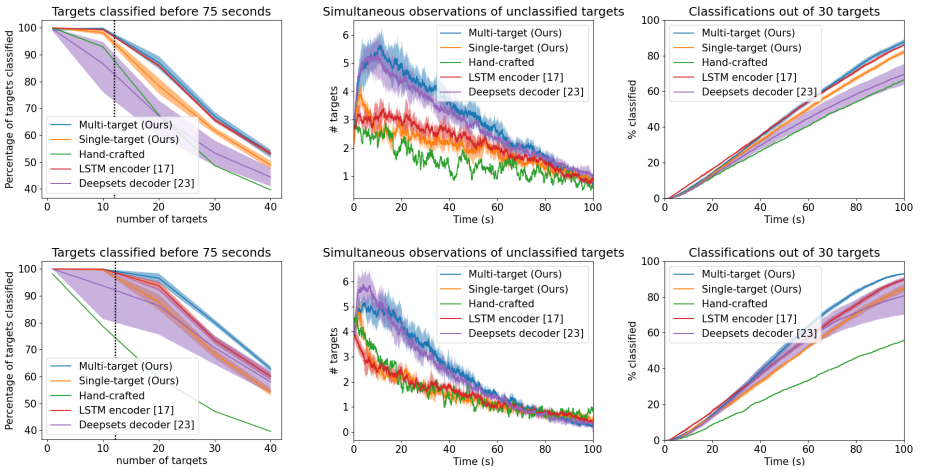


Figure 3.3: **Top row)** Experiments with targets following constant velocity dynamics, as in training. **Bottom row)** Experiments with targets following social forces dynamics not seen during training. **Left)** Comparison of the percentage of targets classified before timeout in environments with 1 to 40 targets. The black line denotes the maximum amount of targets seen during training. **Center)** Evolution of simultaneous observations along the episode in environments of 30 targets. **Right)** Classification speed in environments of 30 targets.

their distance to the drone. The drone classifies the first in the list before moving on to the next one. Observations of other targets are still accounted for in the information fusion and belief computation.

3.6.3 Scalability analysis

We evaluate each method in environments containing up to 40 targets to test their scalability to larger number of targets than seen during training. In Figure 3.3-**Left** , we report each method's percentage of classified targets at the 75 second mark in environments containing different numbers of targets. In general, as expected, there is a drop in the percentage of targets that can be classified when their quantity increases. This is due to the task theoretically requiring more steps and an increase of occlusions generated by other targets. The results show that our method is the one able to generalize best to target dynamics unseen during training and generally outperforms all other baselines in environments containing much larger amounts of targets than seen during training. While our method does not take any assumption on how target information should be weighted, [60] relies heavily on the priority order given to targets, as analysed in [44]. This is the reason why, in our experiments, [60] shows similar scalability results under target dynamics seen during training (Figure 3.3, **Top row**) but does not generalize as well as our method to unseen dynamics (Figure 3.3, **Bottom row**).

3.6.4 Policy behavior

We provide an empirical analysis on each method's behavior and the effect on its performance. In Figures 3.3-**Center** and 3.3-**Right**, we test our policy in simulated perception

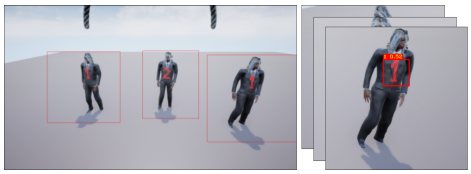


Figure 3.4: Pedestrian models, as seen from the drone's perspective in the realistic environment. Each pedestrian is tracked using AirSim's API. Its class is represented by a red number in the front, classified using YoloV3.

environments with 30 targets and report the number of unclassified targets simultaneously observed along the episode. During the first half of the episode, our method consistently is shown to observe and provide classification estimates of more targets than other baselines, which results in faster classification of targets. This shows that our method is able to learn the effect of each target on the observations of others and discover groups of simultaneously observable targets. The latter is difficult to achieve by single-target classification baselines or methods that assume distance-based relations among targets. Similarly, the lower performance albeit similarly high simultaneous observations of the *DeepSets decoder* baseline, especially at the end of the episode when there are less targets to classify, suggest that the attention-based pooling layer allows better aggregation of each target's latent information, effectively establishing a classification priority order. The sharp decline in simultaneous observations of unclassified targets is due to simultaneously observable targets becoming classified, which results in a sparser distribution of unclassified targets.

3.7 Photo-Realistic Results

To ascertain our framework's capabilities under realistic conditions, we test its performance in an environment generated with Unreal Engine using AirSim [97] to simulate drone control and perception. We simulate our targets using open-source 3D human meshes, produced in *MakeHuman* [98], which move to random goals while avoiding collisions in an environment of $50 \times 50\,m^2$. As shown in Figure 3.4, to remain close to the use-case shown in the earlier simpler environment, pedestrian classes are denoted by a red number in their front.

Photo-Realistic Observation Model

We obtain the cropped image of every pedestrian detected in the drone's field of view (FOV) using the AirSim API, avoiding the problem of data association. Each image is resized, and everything other than the painted number is filtered out.

An implementation of the YoloV3 [99] algorithm is used to detect and classify the digit in each processed image. We train the algorithm using a dataset of rotated, up-scaled and down-scaled MNIST images. Both YoloV3 implementation and the adapted dataset have been obtained from [100].

The output of the algorithm consists in a set of bounding boxes, each one associated to a detected digit and a normalized score stating how sure the algorithm is of it's detection. We rule out those boxes containing a number of pixels smaller than a threshold $B_a = 2000$ and take the box with the highest score. The normalized score of the detected digit is used as the target class probability estimate, distributing the remaining probability mass among

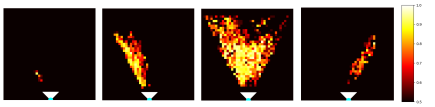


Figure 3.5: Probability heat-map of the employed black-box classifier for a target of class 1. For a robot placed at the bottom, facing upwards and a FOV of 90º, every plot shows the output probability of identifying the real class of the target at different relative positions and a fixed relative orientation from the robot.

the other classes. The output is a uniform distribution when no digits are detected, or the normalized score is below the uniform distribution.

Training conditions

We use the training setup explained in Section 3.5.1, using the realistic observation model. However, to avoid the computational cost of running Unreal/Airsim and YoloV3, for training we use AirSim to extract a dense library of pedestrian observations of all classes in different relative poses from the drone's FOV. For each class, each pose-dependent image is used to compute and save a probability distribution. During training, for each target in the drone's FOV, we use the probability distribution of the library image whose pose is the closest to the target's current relative pose. Being in a controlled environment, during training we monitor the output of the perception system to detect classification errors, and substitute them by a uniform probability. This enables our policy to prioritise good over bad classification viewpoints, adding an additional robustness mechanism to outlier classifications to the control policy. An example of the resulting normalized classification score is given in Figure 3.5.

Real Perception Environment

This time we test our method's performance in the photo-realistic environment. In Figures 3.6a and 3.6b, we only analyze the performance of the viewpoint recommendation policy without having AirSim's different drone dynamics and in-built controller affecting the results, which we use for comparison in Figure 3.6c. We show results for our method and how it compares in this new setting to the baselines presented in Section 3.6.1.

As in section 3.6, we evaluate each method 50 times in environments containing up to 40 targets for 75 and 100 seconds, and show the resulting mean and standard error. In all our experiments, the mean percentage of misclassifications was under 3% with no significant differences among different methods. Figure 3.6 shows that the results presented in Section 3.6 hold in the new photo-realistic environment, even when realistic drone dynamics and control are used. Note that the different performance in comparison to Figure 3.3 is due to the different observation model and the fact that we take measurements directly from the viewpoints recommended by the DRL policy in Figures 3.6-**Left** and 3.6-**Center**. This is very relevant from the Sim-to-Real perspective, considering that the policy has been trained in a different environment with a perception system approximated from the one used during testing.

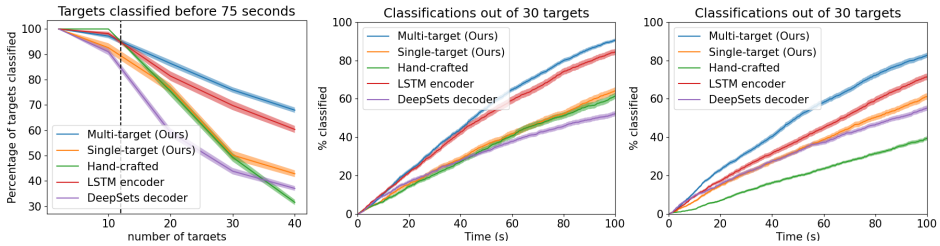


Figure 3.6: Mean and standard error over 50 experiments in the photo-realistic simulator. **Left)** Comparison of the percentage of targets classified before timeout of 75 seconds in environments with 1 to 40 targets, where measurements are directly taken from viewpoints that the DRL policy suggests. **Center)** Classification speed in the experiment of (Left) with 30 targets. **Right)** Classification speed in environments of 30 targets with realistic drone dynamics and AirSim's controller.

3.8 Conclusion

In this chapter, we have introduced a framework for active classification of multiple dynamic targets when the information is extracted using a "black-box" classifier. The proposed framework learns a policy that outputs viewpoints through Deep Reinforcement Learning using an attention-based, permutation invariant architecture. Then, a low-level MPC controller moves the drone to the viewpoints taking care of the complex dynamics at high-frequency. Sensor fusion of the black-box sensor is done through conflation. The results have shown that our policy outperforms multiple baselines, both in terms of generalization to target dynamics not seen during training and scalability to environments with more than double the amount of targets experienced during training. However, there is a limit to the number of targets one robot can classify under given time constraints. In the future, multiple drones could be employed for efficient dynamic active classification.

4

Distributed multi-target tracking and active perception with mobile camera networks

The work of this chapter has appeared in:

S. Casao*, Á. Serra-Gómez*, A. C. Murillo , W. Böhmer, J. Alonso-Mora, E. Montijano, *Distributed multi-target tracking and active perception with mobile camera networks*, in Computer Vision and Image Understanding, 2023, doi: 10.1016/j.cviu.2023.103876.

S. Casao worked on the distributed tracking framework and Á. Serra-Gómez worked on the active perception framework.

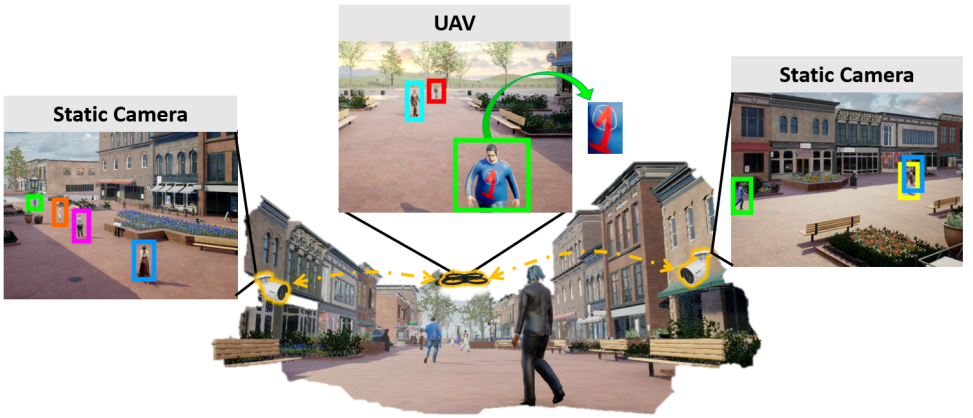


Figure 4.1: Overview of our multi-camera collaborative system. The system comprises a camera network that performs a distributed multi-target tracking process. The static cameras monitor the scene and the mobile cameras are guided by a control policy to capture close-up images of viewpoints likely to strengthen the classification of certain attributes.

4.1 Introduction

Multi-camera systems are common in applications such as surveillance or monitoring. The use of multiple cameras increases the coverage and the amount of information collected from large-scale scenes. Although the most frequent configuration in surveillance applications is a network of static cameras, including mobile cameras brings plenty of potential benefits. In addition to the improved coverage capabilities of such a hybrid system, mobile cameras can be guided to acquire more detailed information and particular viewpoints when needed. Enhancing collaborative behavior among them is then essential to achieve an efficient mutual scene understanding [101–103].

One of the main challenges of collaborative camera network systems is to attain robustness and efficiency. Hence, there has been a tendency to transition from centralized to distributed setups that can easily scale and are more robust against individual node failures ([104, 105]). Another common challenge in multi-camera systems is finding a suitable viewpoint that maximizes gaining new knowledge for a given recognition task. For instance, solving tasks such as person identification or clothing brand recognition requires a specific viewpoint, which should be free from occlusion or blind spots. Active perception enables the capability of moving a camera to the location of the most informative perspective. Developing and evaluating distributed solutions, where mobile cameras with autonomous decision-making are involved, is not a trivial task. To address all of these challenges we propose a novel active and distributed framework. Our system has static cameras to monitor the scene and mobile cameras to strengthen the visualization of certain attributes with high-resolution close-up target images, as summarized in Figure 4.1.

The mobile cameras, drones in our case, are guided by a control policy built upon previous work [106]. This policy continuously determines the cameras' next position and orientation to capture viewpoints that maximize the acquisition of relevant information for

certain people's attributes class. Differently from our prior work, here we consider multiple drones working together with a network of static cameras that provide information about the targets' position and orientation using real data, taking into account the challenges associated to the use of a real tracking system.

The distributed tracking process in charge of this task is based on [8]. Our contribution in this module is related to the implementation, making the transition to a real system easier thanks to the integration with ROS to handle communications. The assessment of the framework is performed with a photo-realistic simulator. In particular, we use the open-source Unreal Engine together with the AirSim simulator [97], which provide a photo-realistic environment to simulate drones and static camera data generation. Additionally, we employ specific tools for creating scenes involving multiple pedestrians from [19].[1]

To summarize, the main contributions of this work are:

- A novel hybrid multi-camera framework, composed of static and mobile nodes, that collaboratively tackles the problem of people monitoring. To do so, it combines distributed tracking and active perception of semantic knowledge from the scene.
- *Active Perception:* We extend prior work to consider multiple mobile cameras and real perception provided by the distributed tracking algorithm.
- *Distributed Tracking:* We incorporate distributed communications using ROS and perform the evaluation with a photo-realistic simulator, contributing to bridge the gap with real-world applications.

4.2 Related Work

4.2.1 Multi-camera Multi-target Tracking

Multi-camera centralized setups are commonly used in real-world applications to cover larger areas ([107, 108]) or acquire a greater amount of information ([109, 110]). These centralized approaches process the entire camera network information in one unique node, making it difficult to scale up. Thus, there is a trend toward distributed setups to increase the applicability of multi-camera systems [111]. While theoretical works have proposed solutions to problems such as event-trigger mechanisms for bandwidth requirements ([112]) or consensus algorithms to unify local estimations ([113, 114]), only a few works have addressed the distributed multi-target tracking with real data. For example, [115] combine the Information-weighted Consensus Filter (ICF) with the Joint Probabilistic Data Association Filter (JPDAF), which uses the previous target states, to fill the gap of relating measurements and trackers in the consensus algorithm. Based on the same ICF consensus method, [116] address the association of measurements and trackers through a global metric that merges appearance and geometry cues. To associate trackers across cameras, they employ the Euclidean distance between the 3D position of the targets. Different from [115] and [116], we tackle the problem of having mobile nodes in the camera network. Besides, we analyze in both data associations, trackers with measurements and cross-camera trackers, the geometric information together with the appearance representation.

[1] Simulated data and photo-realistic environment used available at `https://sites.google.com/unizar.es/poc-team/research/hlunderstanding/collaborativecameras`.

4.2.2 Collaborative systems for perception tasks

Multiple works have developed collaborative systems to address complex perception tasks. One of the most common problems tackled is active object tracking, where visual observations are transformed into a camera control signal to improve the tracking process, e.g., turning left or moving forward ([117]). The combination of a fixed camera, that globally monitors the scene, with a pan-tilt-zoom (PTZ) camera, used to increase the image quality of the target of interest, is proposed in [102] In [118], this setup is extended to a centralized PTZ camera network, where reinforcement learning techniques are employed to learn the new pose of the cameras for finding the target and tracking it as long as possible. In order to follow an object capable of moving in all directions, [119] develop a cooperative aerial robotic approach with two drones for achieving overlapping images and forming a pseudo-stereo vision system. The collaboration of hybrid systems has been studied for different tasks such as dynamic obstacle avoidance, where the information of the static cameras is leveraged by the mobile robot ([101]), or the localization, planning, and navigation of ground robots using a semantic map created by a high-altitude quadrotor ([103]). Furthermore, some works have focused on distributed collaborative perception tasks. [105] propose an approach for distributed learning where each robot only shares the weights of the network for privacy protection and [104] present a general-purpose graph neural network for fusing node information and obtaining accurate perception tasks. Closer to our work, [120] leverage the collaboration of fixed cameras, PTZ, and UAVs for crowd scene covering in a distributed manner. Different from [120], we do not assume as known the target positions, which entail addressing the challenges of a distributed tracking system.

4.2.3 Active Perception for Class Recognition

The active perception problem of recognizing certain classes is commonly addressed by defining a set of viewpoints in advance, which are then used to plan trajectories for gathering new information. One-step greedy planners select viewpoints specific to objects based on factors such as class uncertainty and observation occlusions [72]. Instead, non-myopic methods such as [73] consider both, movement costs and information gained between the object's viewpoints. Alternatively, some approaches formulate the problem as a partially observable Markov Decision Process (POMDP) and design paths over viewpoints by accounting for costs associated with measurements, occlusions, and potential misclassifications [74]. Likewise, [75] employs a modified version of Monte-Carlo tree search to generate plans. However, these techniques typically rely on a priori access to the black-box model for estimating the usefulness of viewpoints. More recent works use non-myopic learning methods like Deep Reinforcement Learning (DRL) for static multi-target pose estimation and active perception. They optimize camera movements to reduce observation uncertainty [76] or maximize information gain [77]. However, these approaches either assume static targets, are limited to closed environments [78], or require prior knowledge of where the information is visible from [79, 80]. Our work leverages an attention-based neural network architecture to encode dynamic targets and to provide viewpoint recommendations that are traced with a low-level controller. In addition, we enable the use of multiple drones and overcome the assumption of possessing prior knowledge about the positions and orientations of the targets by exploiting the collaboration with a multi-target tracking system.

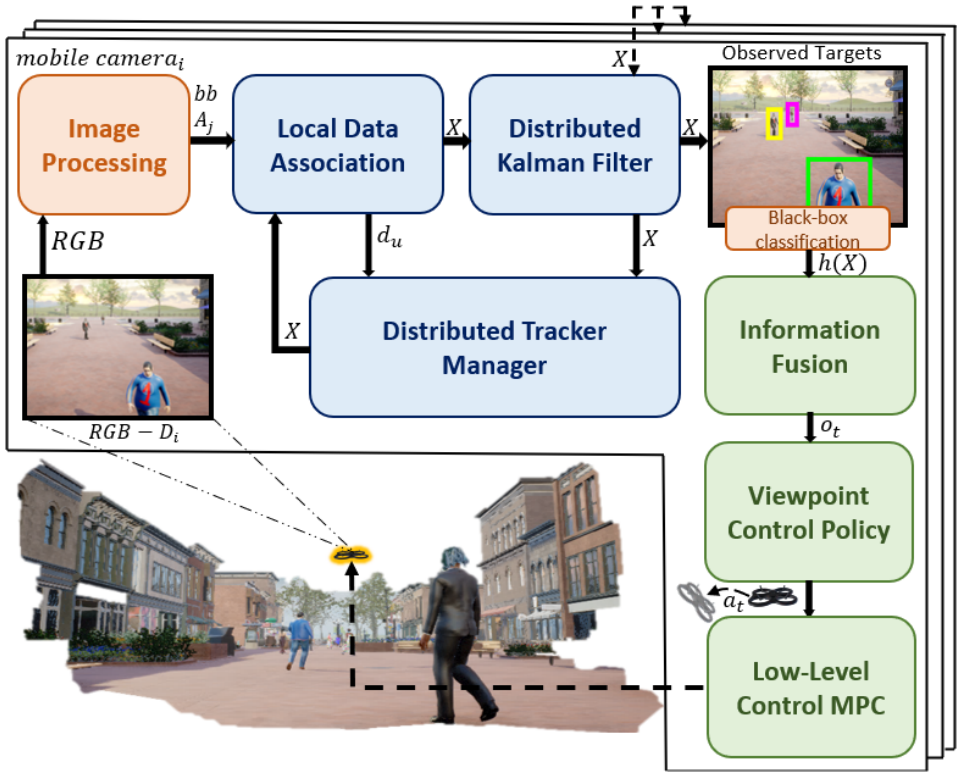


Figure 4.2: Method overview deployed in one mobile camera. The whole system is implemented in ROS, initializing each camera as a node and the image processing module as a service. First, the Local Data Association relates people detection (bb) with the corresponding trackers ($\mathcal{X}$). Then, the cameras exchange and fuse data with their neighboring cameras to obtain a collaborative distributed tracking system. The knowledge of the environment is provided to the control policy for obtaining a new recommendation of viewpoint (a_t) to improve the gathering people's information.

4.3 Preliminaries

4.3.1 Problem Formulation

This work addresses the distributed tracking and correct visualization of people's attributes in large-scale environments. We monitor an area populated by a set of I targets, $\{\mathcal{X}_i\}_{i=1}^{I}$, with a system of J cameras, $\{C_j\}_{j=1}^{J}$, where a subset of $Q < J$ cameras can translate and rotate, e.g. they are installed on drones. Each camera in the network captures an RGB image and a depth map to estimate the state of the targets locally by fusing its information with that received from its neighbors, $\mathcal{N}_j$. The state of target i in camera j is defined as $\mathbf{x}_i^j = (x_i^j, y_i^j, z_i^j, w_i^j, h_i^j, \dot{x}_i^j, \dot{y}_i^j)$ represented by a 3D cylinder with (x_i^j, y_i^j, z_i^j) the 3D coordinates of the center cylinder's base, w_i^j the width, h_i^j the height, and $(\dot{x}_i^j, \dot{y}_i^j)$ the velocity of the target in the x and y directions, respectively. The orientation of the target, φ_i^j, is estimated based on their velocities $\dot{x}_i^j$ and $\dot{y}_i^j$. The responsibility for correctly visualizing the attribute's class of the

targets lies in the moving cameras (drones). It is important to note that these attributes can only be observed from specific viewpoints, such as determining if the targets are wearing a backpack or glasses. The state of the drones $\mathbf{y}_q = (\mathbf{u}_q, \psi_q)$, assumed as known in this work, is represented as their position $\mathbf{u}_q$ and their heading ψ_q, being $q \in \{1, \dots, Q\}$. Each drone is controlled by a hierarchical policy, where a viewpoint control policy operating at $\frac{1}{\tau_h}$ Hz takes as input the knowledge of the scene and outputs a viewpoint recommendation $\mathbf{a}^q$. Next, the recommended viewpoint is traced with a low-level controller operating at $\frac{1}{\tau_l} \gg \frac{1}{\tau_h}$ Hz. The purpose of the policy is to position the targets' attributes within the field of view (FOV) of the drone. We assume that the drones are faster than the targets and fly at a constant height above them, avoiding collisions.

The goal of the presented work is to achieve an accurate estimation of the targets' position and visualize all people's attributes as quickly as possible.

4

4.3.2 Overview

Figure 4.2 presents an overview of the proposed method to address the problem described in the previous section. The complete framework has been implemented in ROS, with each camera defined as a node of the system and ensuring synchronization between them. Neural networks have been implemented in the image processing module as services to save memory.

First, each camera captures an RGB image and a depth map (D_i) to compute the re-projection between the image plane and the real-world coordinates. We incorporate depth information to simplify the re-projection but this could be replaced by a network calibration in a more realistic setup. Then, a general detector provides the people bounding boxes (bb) that are used as measures for the tracking system and that are associated with the current trackers through the Local Data Association module (LDA). Once the cameras in the networks exchange the targets' information ($\mathcal{X}$) with their neighbors, the Distributed Kalman Filter (DKF) implemented attempts to obtain consensus on the targets' state. Finally, the Distributed Tracker Manager (DTM) initializes new trackers and associates them locally with the trackers received from the neighboring cameras. The mobile cameras of the system obtain the output of a black-box CNN, with perception information about the visible targets ($h(\mathcal{X})$), and update their class beliefs with an efficient information fusion method. Based on the latter and the estimated state of the targets, the viewpoint control policy recommends a new camera pose (a_t) to maximize the information acquired in the next step. The new viewpoints are then tracked with a low-level controller.

4.4 Distributed Tracking

This section explains the different components of our approach to perform fully distributed multi-target tracking with hybrid collaborative cameras.

4.4.1 Distributed Kalman Filter

We define the target motion model as a discrete-linear dynamic system with constant velocity. Each camera executes a Kalman filter independently producing a local estimation of the target state, $\hat{\mathbf{x}}_i(k)$, and the associated error covariance matrix $\mathbf{P}_i(k)$. Note that local estimations may vary among different cameras. Therefore, the Distributed Kalman-

Consensus filter [113] is implemented to mitigate these differences and seek to reach a consensus in $\hat{\mathbf{x}}_i(k)$ for all cameras C_j.

The consensus algorithm assumes knowledge of the data association between the local measurement $\mathbf{z}_i(k)$ and the target prediction $\bar{\mathbf{x}}_i(k)$, which is obtained by applying the linear motion model to the previous target state estimation $\hat{\mathbf{x}}_i(k-1)$. The measurement $\mathbf{z}_i(k)$ is the 3D cylinder obtained as the projection of the bounding box given by the detector, and the velocity of the target computed with the last data association, i.e., $\mathbf{z}_i(k) = (x(k), y(k), z(k), w(k), h(k), \dot{x}(k), \dot{y}(k))$. This measurement is coupled in the filter with a zero mean Gaussian noise characterized with $\mathbf{R}_i(k)$ as its covariance matrix. Using mobile cameras requires online updates of the transformation matrix from the image plane to the three spatial global coordinates of the world. The cameras of the photo-realistic environment follow the pinhole model, which combined with the depth information, $d(k)$, enables the conversion of image plane coordinates $v_x(k)$ and $v_y(k)$, to the relative 3D world camera coordinates $x_r(k), y_r(k)$ and $z_r(k)$ by

$$x_r(k) = d(k), y_r(k) = \frac{d(k)}{f}(v_x(k) - c_x), z_r(k) = \frac{d(k)}{f}(v_y(k) - c_y) \tag{4.1}$$

where f is the focal length and, c_x and c_y are the image center coordinates in x and y, respectively. Then, the relative camera coordinates are transformed into the common global world system demand by the consensus-filter algorithm following

$$\begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ 1 \end{bmatrix} = \begin{bmatrix} R_j(k) & | & T_j(k) \\ 0 & & 1 \end{bmatrix} \begin{bmatrix} x_r(k) \\ y_r(k) \\ z_r(k) \\ 1 \end{bmatrix} \tag{4.2}$$

being $R_j(k)$ the rotation matrix and $T_j(k)$ the translation vector of the camera at instant k, assumed as known. In a real setup, this information could be computed by offline calibration of the cameras and using onboard sensors such as GPS or IMUs together with SLAM algorithms for the drones. Regarding the velocity, we take advantage of the online tracking to measure the time the target has taken to arrive at the current position at k since the last data association between $\bar{\mathbf{x}}_i$ and $\mathbf{z}_i$.

Once the camera j associates the local measurement $\mathbf{z}_i^j(k)$ with the local target prediction $\bar{\mathbf{x}}_i^j(k)$, the consensus algorithm transforms the measurement and its noise to the information form by

$$\mathbf{u}_i^j(k) = \mathbf{H}^T {\mathbf{R}_i}^{-1}(k)\mathbf{z}_i^j(k), \quad \mathbf{U}_i^j(k) = \mathbf{H}^T \left(\mathbf{R}_i^j(k)\right)^{-1} \mathbf{H}. \tag{4.3}$$

The obtained sensor data information, $\mathbf{u}_i^j(k)$, and its inverse-covariance matrix, $\mathbf{U}_i^j(k)$ are exchanged with the neighboring cameras in the network $\mathcal{N}_j$, together with $\bar{\mathbf{x}}_i(k)$. Due to the transformation into the information form, we are able to combine all the measurements received from other cameras with the acquired one by simply adding them,

$$\mathbf{y}_i^j(k) = \sum_{C \in \mathcal{N}_j} \mathbf{u}_i^C(k), \qquad \mathbf{S}_i^j(k) = \sum_{C \in \mathcal{N}_j} \mathbf{U}_i^C(k). \tag{4.4}$$

Finally, the estimated state is updated by correcting the prediction target state with the data computed in (4.4) and the predictions from the neighboring cameras following

$$\begin{aligned}\hat{\mathbf{x}}_i^j(k) = \bar{\mathbf{x}}_i^j(k) + \mathbf{M}_i^j(k)\left[\mathbf{y}_i^j(k) - \mathbf{S}_i^j(k)\bar{\mathbf{x}}_i^j(k)\right] \\ + \gamma \mathbf{M}_i^j(k) \sum_{C \in \mathcal{N}_j} (\bar{\mathbf{x}}_i^C(k) - \bar{\mathbf{x}}_i^j(k)),\end{aligned} \tag{4.5}$$

where $\mathbf{M}_i^j(k) = (\mathbf{P}_i^j(k)^{-1} + \mathbf{S}_i^j(k))^{-1}$ is the Kalman Gain in the information form and $\gamma = 1/\|\mathbf{M}_i^j(k) + 1\|$.

4.4.2 Local Data Association

For simplicity in the explanation, this subsection will focus on the data association in a single camera. Hence, the subscripts j used in the notation will refer to the different measurements locally observed and not the cameras in the network. The accurate update of the DKF relies on a correct association between the set of measurements, $\mathcal{Z} = \{\mathbf{z}_j\}$, and the set of targets prediction, $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_i\}$, during each estimation cycle. To tackle this issue, we assess two constraints based on geometry and appearance.

The similarity value in the geometry of both sets is obtained as

$$s_d(\mathbf{z}_j, \bar{\mathbf{x}}_i) = \begin{cases} \frac{1}{\alpha}\, d_M(\mathbf{z}_j, \bar{\mathbf{x}}_i) & \text{if } d_M(\mathbf{z}_j, \bar{\mathbf{x}}_i) < \tau_d \\ 1 & \text{otherwise,} \end{cases} \tag{4.6}$$

being α a configuration parameter, τ_d a threshold applied to ignore highly unlikely candidates and, $d_M(\mathbf{z}_j, \bar{\mathbf{x}}_i)$ the Mahalanobis distance between the x, y positions. The covariance matrix in the Mahalanobis distance is computed by adding the sub-matrices of $\mathbf{P}_i$ and $\mathbf{R}_j$ that encode the covariance position of the estimation $\bar{\mathbf{x}}_i$ and the measurement $\mathbf{z}_j$, respectively.

Then, those data whose distance is below τ_d are evaluated in appearance. To get representative appearance features for measuring similarity, we use the output of a person re-identification network ([121]) pre-trained in the MSMT17 Benchmark ([122]) as appearance descriptors. Inspired by this re-identification methodologies, each local tracker creates an online appearance model, $\mathcal{F}_i$, of the target i with budget size. This appearance model, also called gallery, is built based on a scoring system that estimates the usefulness and confidence of each appearance feature. Thus, every feature of the gallery, $\mathbf{f}_i^\ell \in \mathcal{F}_i$, has a score assigned $\varepsilon(k)$ whose value changes depending on two factors. First, the gallery component with the minimum distance to the final associated measurement appearance increases its score by one with

$$\varepsilon_i^\ell(k+1) = \begin{cases} \varepsilon_i^\ell(k) + 1 & \text{if } \ell = \arg\min_{\mathbf{f} \in \mathcal{F}_i} \delta(\mathbf{f}_j, \mathbf{f}), \\ \varepsilon_i^\ell(k) & \text{otherwise,} \end{cases} \tag{4.7}$$

where $\mathbf{f}_j$ is the appearance feature get from the $\mathbf{z}_j$ bounding box detection. Secondly, the closest component of the appearance model to the gallery centroid, $\bar{\mathbf{f}}_i$, increases by one its

value score while the farthest component decreases by one following

$$\varepsilon_i^{\ell}(k+1) = \begin{cases} \varepsilon_i^{\ell}(k)+1 & \text{if } j = \arg\min_{\mathbf{f}\in\mathcal{F}_i} \delta(\bar{\mathbf{f}}_i, \mathbf{f}), \\ \varepsilon_i^{\ell}(k)-1 & \text{if } j = \arg\max_{\mathbf{f}\in\mathcal{F}_i} \delta(\bar{\mathbf{f}}_i, \mathbf{f}), \\ \varepsilon_i^{\ell}(k) & \text{otherwise.} \end{cases} \tag{4.8}$$

The gallery is updated periodically every N iteration with a new feature. Once the budget size is reached, the component with the lowest score is dropped to make room for the newest one. Finally, the similarity between the appearance feature of the measurement, $\mathbf{f}_j$, and the tracker's gallery $\mathcal{F}_i$ used as a model of the appearance of the prediction state $\bar{\mathbf{x}}_i$ is provided by the minimum cosine distance

$$s_a(\mathbf{f}_j) = \min_{\mathbf{f}_i\in\mathcal{F}_i} \left(1 - \frac{\mathbf{f}_j^T\,\mathbf{f}_i}{\|\mathbf{f}_j\|\|\mathbf{f}_i\|}\right), \tag{4.9}$$

The final data association assignment between the measurements, $\mathcal{Z} = \{\mathbf{z}_j\}$, and the target predictions, $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_i\}$, is solved with the Hungarian algorithm [123] by defining the cost function as the product of both similarity scores, s_d and s_a.

4.4.3 Distributed Tracker Manager

In the practical implementation of distributed tracking systems, it is also essential to perform a correct association of trackers across the different cameras in the network. Our proposed approach to address this problem involves performing the same process as the one described in Section 4.4.2 for the local data association but with the set of measurements replaced by the set of other camera's predictions $\bar{\mathcal{X}}_j = \{\bar{\mathbf{x}}_j\}$, and using the Euclidean distance instead of the Mahalanobis distance. These modifications are based on the information exchanged in the communication message, which is subject to the data required in the DKF and does not include the covariance matrices $\mathbf{P}_j$. In case no local tracker is associated with those received from neighboring cameras, the current camera initializes a new tracker based on the tracker information received.

Since we limit sharing appearance exclusively to newly initialized trackers for saving bandwidth, the tracker consensus process across cameras occurs only when a new tracker is initialized in any of them. To ensure the robustness of mobile cameras in dynamic communication scenarios, where the cameras they exchange information with may change over time, we include the cross-camera trackers association in the communication message. This cross-camera trackers association consists of a look-up table where each tracker locally stores the unique identifier, i, assigned to the same target by the rest of the cameras in the network C_j. Consequently, once the message has traversed the entire network, the cameras achieve a global consensus on the association of trackers across all the cameras in the network.

4.5 Active Perception

In addition to collaborating in the distributed tracking of multiple targets, mobile cameras tackle the task of active perception to gain additional knowledge about the people presented

in the scene. They leverage shared information to efficiently position themselves for effectively visualizing each target's attribute class. In this work, mobile cameras are allowed to communicate between them in order to gather global knowledge of the visualization process's status.

4.5.1 Target class observations and Belief Updates

Every time step τ_h, the drone uses a black-box perception algorithm (e.g., a pre-trained CNN classifier) to compute the class probability distribution for each target visualized from the correct viewpoint. Let $\mathcal{P} = h(\mathcal{X}) = \{\mathbf{p}_i\}_{i=1}^{I}$ be the class probability distribution, where $\mathbf{p}_i$ represents the likelihood of target i belonging to each one of the G classes in the class set $\mathcal{G}$. To simplify the notation, in this complete Section 4.5, t will denote times periods of τ_h.

The probability distribution over time is modulated by belief vectors $\mathbf{b}_i^t$ for each target i. These vectors contain G belief values b_{ig}^t representing the aggregate likelihood of target i belonging to a class $g \in \mathcal{G}$ up to time t, i.e., combines the historical class probabilities distributions up to time t. The process of aggregating the drone's observations to derive class beliefs for each target is a crucial consideration. Standard Bayesian recursive estimation is not recommended in this case due to the unavailability of the measurement likelihood model, $\mathbb{P}(\mathbf{p}_i^t|\mathbf{b}^{t-1})$, from the black-box sensor. Building a precise pose-dependent likelihood model requires the construction of a dense dataset and considering all targets and occlusions for optimal viewpoint search. This process is expensive and does not scale well due to its computational demands.

Instead, we propose the use of the conflation operator $\zeta(\mathbf{p}_i^{1:t})$, a mathematical method introduced by [89]. Conflation enables the aggregation of probability distributions obtained from measurements of the same phenomena under different conditions. It possesses the remarkable property of minimizing the loss of Shannon information when combining multiple independent probability distributions into a single distribution, specifically when computing $\mathbf{b}_i^t$ based on the measurements $\mathbf{p}_i^{0:t}$. The conflation is defined by

$$\mathbf{b}_i^t = \zeta(\mathbf{p}_i^{1:t}) \equiv \zeta(\mathbf{b}_i^{t-1}, \mathbf{p}_i^t) = \frac{\mathbf{b}_i^{t-1} \odot \mathbf{p}_i^t}{(\mathbf{b}_i^{t-1})^\top \mathbf{p}_i^t}, \tag{4.10}$$

where the Hadamard product $\odot$ in the numerator is taken component-wise, whereas the dot product is the normalization factor. Conflation's commutative and associative properties enable efficient recursive computation, making it suitable for onboard and decentralized belief updates in the presence of multiple communicating drones. The beliefs are initialized at $t = 0$ with a uniform prior probability distribution over all possible target classes, formally $b_{ig}^0 = 1/G \quad \forall g \in \mathcal{G}$.

4.5.2 Viewpoint Control Policy

The lack of an observation model that maps target relative poses to a probability distribution, i.e., the h function that maps $\mathcal{P} = h(\mathcal{X})$, hinders the direct solution of the active perception for class recognition problem. Therefore, we leverage Reinforcement Learning to train a viewpoint control policy, π_ϕ, that learns to recommend viewpoints $\mathbf{a}_t$ that minimize the accumulated entropy of all targets' beliefs over a given time horizon. The policy is parameterized by ϕ and operates at the perception low-frequency, $\frac{1}{\tau_h}$.

Each drone uses a copy of the same learned viewpoint control policy that solves the viewpoint recommendation problem. The viewpoint recommendation problem is formulated as a Partial Observable Markov Decision Process (POMDP), denoted by $\langle S, A, \mathcal{T}, \Omega, \mathcal{O}, R\rangle$. The state S includes the state of the drones, the targets' pose, their beliefs, and their visualization status (visualized or not). Actions A represent recommended viewpoints within a constrained neighborhood and transitions $\mathcal{T}$ assume timely movement to the next viewpoint. The drone receives partial information Ω about the environment through the observation function $\mathcal{O}$. The observation of each target is defined by $\mathbf{o}^t_{q,i} = [\bar{\mathbf{o}}^p_{q,i}, \bar{\mathbf{o}}^c_{q,i}] \in \Omega$ where $\bar{\mathbf{o}}^p_{q,i}$ is the observation of each target physical attributes (poses and velocities). Each target's attribute information is represented by $\bar{\mathbf{o}}^c_{q,i}$ which includes the entropy of the local class estimates from the drone q and the entropy of the global class beliefs.We define the joint target observation vector as $\mathbf{o}^t_q = \{\mathbf{o}^t_{q,i}\}^I_{i=1} = [\bar{\mathbf{o}}^p_q, \bar{\mathbf{o}}^c_q]$.

The reward function in this work is based on the formulation of chapter 3. It provides rewards to the agent for successfully classifying each and all targets and reducing the entropy of target class beliefs. Additionally, it penalizes the agent for movement and for each time step in which the task remains incomplete. For more detailed information, we refer the interested reader to 3.

Architecture

The generalization ability of the learned policy $\pi_\phi(\mathbf{a}|\mathbf{o}_q)$ depends on the neural network architecture chosen. The main challenge lies in the size and dynamical changes over time of the set $\mathbf{o}^q_t = \{\mathbf{o}^t_{q,i}\}^I_{i=1}$.

Inspired by Relational Graph Convolutional Networks [90] and self-attention mechanisms [17] used in static knowledge graphs, we employ a self-attention block (SAB) to capture the relationships among all targets at time t. Note that the focus in this first layer is on spatial features such as poses and velocities $\bar{\mathbf{o}}^q_p$, since the purpose is to encode important information including target visibility, observation perspective, occlusions, and potential simultaneous observations. Therefore, the initial layer is,

$$\begin{aligned}
\tilde{\mathbf{e}}^{1,h}_{i,p} &= F(\bar{\mathbf{o}}^p_{q,i}; \mathbf{W}^1_{q,h}) + \sum_{j\in\mathcal{J}} \lambda^h_{i,j} F(\bar{\mathbf{o}}^p_{q,j}; \mathbf{W}^1_{v,h}), \\
\mathbf{e}^1_{i,p} &= LN(Res^1(LN(concat(\{\tilde{\mathbf{e}}^{1,h}_{i,p}\}_{h=1\dots H})))), \\
\lambda^h_{i,j} &= \text{softmax}\Big(\frac{1}{\sqrt{d_h}} F(\bar{\mathbf{o}}^p_{q,i}; \mathbf{W}^1_{q,h})^\top F(\bar{\mathbf{o}}^p_q; \mathbf{W}^1_{k,h})\Big)_j,
\end{aligned} \tag{4.11}$$

where $i \in \mathcal{J}$, $Res^l(x) = x + \sigma(F(x; \mathbf{W}^l))$, with σ being a ReLU activation function and F a parametric affine transformation. LN stands for Layer Normalization. $\mathbf{W}^1 \in \mathbb{R}^{d_{enc}\times(d_h H+1)}$ and $\mathbf{W}^1_{w,h} \in \mathbb{R}^{d_h\times(d_{in}+1)}, w \in \{v,q,k\}$, are learnable parameters. d_{in}, d_h, d_{enc} are the dimensionality of the input, each head h, and the first layer. Note that each head h encodes a different relation $\lambda^{\mathbb{h}}$ between targets. To incorporate the information acquired about each target's class, we concatenate it with the latent representation of each target from the previous layer. Then, we map it back to a latent space of dimension d_{enc} using a learned linear layer. The process can be expressed as $\mathbf{e}^1_i = F([\mathbf{e}^1_i, \bar{\mathbf{o}}^c_{q,i}]; \mathbf{W}_c)]$, where $\mathbf{e}^1_i$ represents the updated latent representation, $\bar{\mathbf{o}}^c_{q,i}$ is the class information of target i observed by drone q, and $\mathbf{W}_c$ is the learned weight matrix.

Next, we use a pooling multi-head attention mechanism (PMA) that incorporates a learned seed vector per head $\mathbf{v}_s^h \in \mathbb{R}^{d_h}$ to calculate the attention weights for a single query,

$$\begin{aligned}
\tilde{\mathbf{e}}^{2,h} &= \mathbf{v}_s^h + \sum_{j\in\mathcal{J}} \lambda_j^h F(\mathbf{e}_j^1; \mathbf{W}_{v,h}^2), \\
\mathbf{e}^2 &= LN(Res^2(LN(concat(\{\tilde{\mathbf{e}}^{2,h}\}_{h=1\ldots H})))), \\
\lambda_j^h &= \text{softmax}\Big(\Big\{\frac{1}{\sqrt{d_h}} \mathbf{v}_s^{h,\top} F(\mathbf{e}_j^1; \mathbf{W}_{k,h}^2)\Big\}_{j\in\mathcal{J}}\Big)_j .
\end{aligned} \tag{4.12}$$

The output latent vector $\mathbf{e}^2$ is further processed by a fully connected layer to obtain the parameters $\mu_{\mathbf{a}_t}$ and $\log(\sigma_{\mathbf{a}_t})$ of a diagonal Gaussian distribution $\mathcal{N}(\mu_{\mathbf{a}_t}, \sigma_{\mathbf{a}_t})$ over viewpoints. The learned policy π_ϕ then samples recommended viewpoints $\mathbf{a}_t$ from this distribution. We assume that the drone can reach the recommended viewpoint before the next time step.

For training the network, we employ the Proximal Policy Optimization (PPO) algorithm [14, 63]. PPO requires an estimate of the state-value $V^{\pi_\phi}(\mathbf{s}_t)$, which is approximated by a linear layer predicting $V^{\pi_\phi}(\mathbf{s}_t) \approx \mathbf{v}_v^\top \mathbf{e}^2$. This value estimation is used during training to guide the policy. The training process combines the surrogate loss and KL-divergence term to ensure stability. Additionally, an entropy regularization term is included to promote exploration [91]. For more detailed information and equations regarding the algorithm, we refer the reader to [14].

Low-Level Control MPC

During training, we assume the drone reaches the suggested viewpoint by the next time step. However, at test time we employ a low-level controller operating at a frequency $\frac{1}{\tau_l}$Hz >> $\frac{1}{\tau_h}$Hz, to guide it there while accounting for the drone dynamics. The controller solves the following receding-horizon constrained optimization problem:

$$\begin{aligned}
\min_{\mathbf{y}_{1:N}, \rho_{0:N-1}} \; & \sum_{k=0}^{N-1} w_\rho \left\|\rho_k\right\| + w_g \frac{\left\|\mathbf{y}_N - \mathbf{a}_t\right\|}{\left\|\mathbf{y}^0 - \mathbf{a}_t\right\|} \\
\text{s.t.} \quad & \mathbf{y}_0 = \mathbf{y}_t, \quad \mathbf{y}_{k+1} = f(\mathbf{y}_k, \rho_k) \\
& \rho_k \in \mathcal{P}, \quad 0 \le k \le N-1
\end{aligned} \tag{4.13}$$

where ρ_k is the low-level control input sent to the robot, that needs to be inside the possible values $\mathcal{P}$, $f(\mathbf{y}_k, \rho_k)$ the internal dynamics and w_u and w_g are the respective weights of the stage and terminal costs. For more details, we refer the reader to [92] and [106]. Although our full method accounts for the drone dynamics using this low-level controller, our formulation is flexible to other low-level controllers as long as they track the recommended viewpoint $\mathbf{a}_t$. This is why during simulation we employ both the in-built drone dynamic model and the controller from AirSim, see [97] for more information.

4.6 Experiments

4.6.1 Environment

We use two high-fidelity virtual environments in Unreal Engine to test the presented framework using our previous work [19], where we provide the essential tools for creating

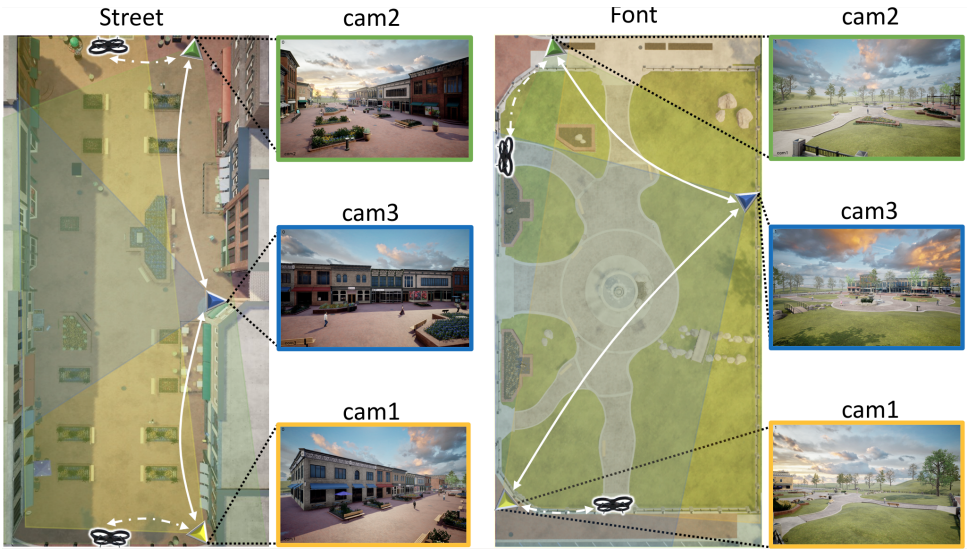


Figure 4.3: Experimental environments used to evaluate the proposed framework. On the left, we show the setup for the experiments performed in the commercial street, *Street*, and on the right the setup for the font area, *Font*. The starting points of the two drones used as mobile cameras are also shown and they always communicate with each other.

multi-pedestrian scenarios. Photo-realistic simulators offer several advantages, including obtaining automatically labeled data, easily varying testing conditions, and developing autonomous robotics approaches by filling the gap of using perception information. Previous works have shown that methods developed in such environments, which are increasingly prevalent, can generalize to real-world scenes with augmentation techniques ([124, 125]).

The designed scenes are presented in Fig. 4.3. The first scene is a commercial street (Street), while the second is a green open area (Font). Their respective dimensions are 97x27m and 97x50m. In both scenes, we place three static cameras with overlapping views for global area monitoring and define distant starting points for the drones. The size of the images captured by the camera network is set to $1440x900$ and the field of view to 90 degrees[2]. Regarding communications, to be as faithful as possible to a real-world scenario, we set the drones to share information with each other as well as with the closest camera to them at the time. Communication between static cameras is limited to their direct neighbor, as shown in Fig. 4.3. Finally, the number of pedestrians present on the scene varies between episodes, and their trajectories are randomized.

Regarding the task of correctly visualizing people's attributes, we devise a marketing study on clothing brands as a use case. Specifically, we create different pedestrians with a logo on the front of their T-shirts which can be visualized exclusively from the frontal view of the person.

[2]The rest of the camera parameters are those set by default in Unreal Engine and AirSim

4.6.2 Evaluation metrics

To comprehensively evaluate the proposed approach for distributed **multi-target tracking**, the common CLEAR MOT metrics ([126, 127]) are adopted for evaluation:

- Multiple Object Tracking Accuracy (**MOTA**): measures failures during the tracking taking into account the number of misses, false positives, and mismatches.
- Identity F1 Score (**IDF1**): evaluates the capability of the system for preserving the identities over time.
- Multiple Object Tracking Precision (**MOTP**): shows the ability of the tracker to estimate precise object positions through the error in estimated position.

The above evaluation is performed in the image plane where metrics require setting a threshold between the ground truth and the resulting trackers in order to consider a tracker valid. We evaluate the resulting bounding boxes in the image plane using a minimum intersection over union (IoU) of 0.3 as the threshold to validate the trackers. The final tracking results are those obtained as output of the Distributed Tracker Manager. The final result is the median of the cameras in the network.

Regarding the acquisition of the correct people's viewpoint obtained from the **active perception** approach, the evaluation is performed using a black-box clothing brands detector. Thus, we employ two metrics:

- Trackers Classified (**TC**): measures the percentage of trackers whose beliefs are higher than 95%.
- Precision (**P**): evaluates the percentage of trackers for which their beliefs exceed 95% and correctly identifies their brand (attribute).

To associate each tracker with a ground truth brand class, we perform a linear sum assignment problem between the trackers and ground truth bounding boxes. The ground truth bounding boxes obtained from the simulator contain the person's attribute class.

4.6.3 Sequence Evaluated

Several sequences are evaluated in each one of the environments with their corresponding ground truth being automatically obtained from the simulator. Specifically, we varied the number of pedestrians to assess the performance for 5, 10, and 15 pedestrians. Thus, the conducted experiments are named as sparse, medium, and busy for 5, 10, and 15 pedestrians respectively, resulting in the following sequences: *Street Sparse*, *Street Medium*, *Street Busy* for *Street* environment, and *Font Sparse*, *Font Medium*, *Font Busy* for *Font* environment. All of them have the same length of 500 frames.

4.6.4 Results and settings

In the following, we explain the baselines selected to compare the proposed method in the *Street* sequences and perform a detailed analysis of the obtained results. To conclude the experiments, we also present the performance of our approach in the metrics described above for both environments, *Street* and *Font*. We set the parameters defined in the method for all the experiments to $\tau_{dLDA} = 1$, $\alpha = 700$, $\tau_{aLDA} = 0.55$, $\tau_{dDTM} = 2$,$\tau_{aLDA} = 1$, $\tau_l = 0.05$, $\tau_h = 0.25$ and the size of the targets' gallery is set to 10.

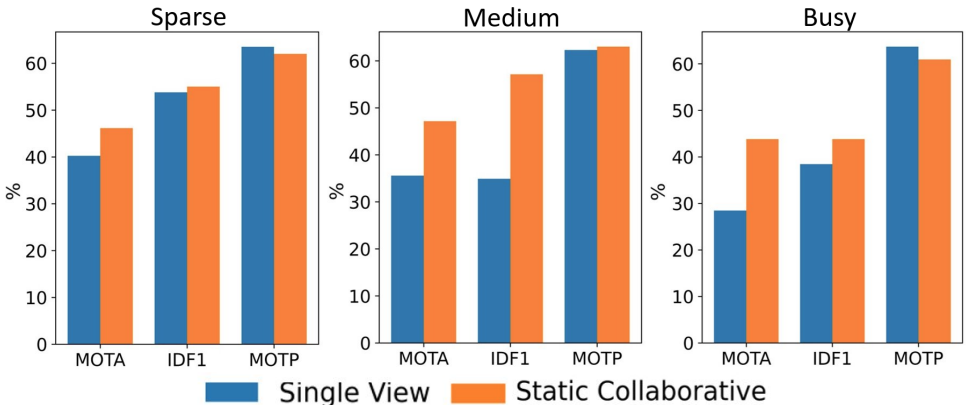


Figure 4.4: Comparison of the cameras responsible for distributed multi-target tracking collaborating with each other with a chain graph of communications (Static Collaborative) and a single view tracking with isolated cameras (Single View)

Collaborative behaviour analysis To demonstrate the benefits of collaborative behavior between nodes in a multi-target tracking network, we gather the three cameras from our system responsible for tracking and assessed their performance with and without communication. The first case (Static Collaborative) follows the initial setup where cameras communicate exclusively with their direct neighbor in a chain graph (Fig. 4.3). In the second setup, the different cameras perform individual tracking without any communication between nodes (Single View). The obtained results, shown in Fig. 4.4, demonstrate the benefits of sharing information once per iteration with minimum communications so that no node in the network is isolated. The Static collaborative setup achieves up to 21% and 15% of improvement in the IDF1 and MOTA metric, respectively, in comparison with the tracking in Single View. Therefore, we can conclude that in large scenarios, the use of collaborative cameras with overlapping perspectives enhances tracking performance in comparison to the use of independent cameras.

Mobile cameras analysis Furthermore, we evaluate the efficiency of our mobile cameras (MC) to correctly visualize the desired people's viewpoint against a baseline of static cameras (SC). The static setup is composed of five cameras, the three already existing in the system and two more located on the other side of the street for more visual coverage

Method	**Classification Process (%)**					
	Street Sparse		*Street Medium*		*Street Busy*	
	↑TC	↑P	↑TC	↑P	↑TC	↑P
SC	75	75	80	60	70.83	58.33
MC	71.5	64.3	76.2	66.7	81.5	74.1

Table 4.1: Percentage of trackers classified (TC) and percentage of trackers correctly identified (P) in the *Street* sequences. Results for the baseline static camera network (SC) and our hybrid system with two mobile cameras (MC).

of the scene. Communications among the five cameras are defined as a ring graph, i.e., each camera shares information with its two nearest neighbors. As a consequence of the distributed nature of the system, the static cameras collaborate to gain knowledge of the overall scene, and the evaluation of the correct viewpoint visualization is performed individually. The final results of the baseline are the median of all the cameras in the system.

The results obtained of the percentage of trackers classified (TC) and correctly identified their brands (P) with beliefs higher than 95% are presented in Table 4.1. In the sparse scenario, where the occlusions between targets are not frequent, the static camera setup gets better results than the mobile cameras. However, in more crowded scenarios, static cameras struggle to avoid occlusions for obtaining a view with high confidence from the pedestrian. In contrast, mobile cameras can be actively positioned to capture the desired viewpoint, achieving a coverage (TC) of 81.5%, against the 70.83% obtained from the static setup, in the most challenging scenario (*Street Busy*). In addition, the quality of the people data captured by each one of the systems is unmatched. Fig. 4.5 shows examples of the same pedestrian captured with the mobile cameras (blue box) and with the static cameras (red dashes box). Every two columns correspond to the same person and we can notice the great difference in quality. The images from mobile cameras revealed much more clear details than the static ones, whose images are of low quality and blurry. These result in better identification of the person's brand in most of the sequences evaluated (P).

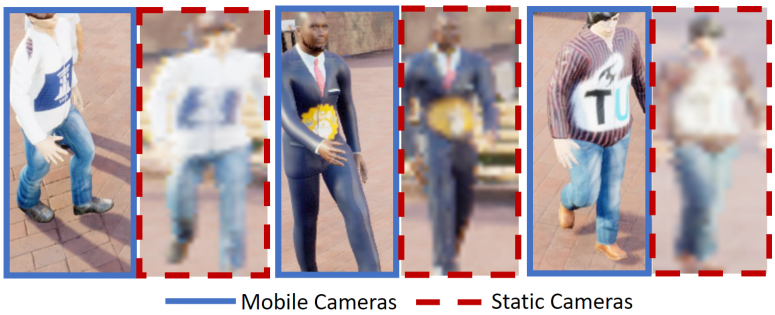


Figure 4.5: Examples of people images captured from the correct viewpoint: mobile cameras (blue box) and static cameras (red dashed box) in the *Street Busy* sequence. Every pair of columns displays images of the same person.

Final Evaluation As a summary, we present the performance of the proposed framework in both photo-realistic environments, *Street* and *Font*. The results obtained are shown in Table 4.2, from which we can conclude that the method is consistent under various conditions, including different numbers of people, size of the space, and type of environments. Specifically, the experiments focus on evaluating sparse, medium, and busy scenarios, with 5, 10, and 15 pedestrians, respectively. Moreover, the *Font* environment is larger than the *Street* environment with static cameras located further away from the path where people walk, making it more challenging for monitoring. Finally, we also perform a measurement of the mean time required by each of the modules comprising the proposed framework: detection 0.0198 s, local data association per tracker 0.038 s, distributed Kalman filter per

Sequence	Multi-target Tracking			Classification	
	↑MOTA%	↑IDF1%	↑MOTP%	↑TC%	↑P%
Street Sparse	54.18	48.34	61.43	71.5	64.3
Street Medium	43.62	42.57	60.45	76.2	66.7
Street Busy	38.83	42.52	60.1	81.5	74.1
Font Sparse	38.24	41	58.1	100	75
Font Medium	47.22	55.52	59	93.3	53.35
Font Busy	40.34	47.96	63.63	72.73	63.63

Table 4.2: Results of the evaluated metrics in the *Street* and the *Font* sequences where sparse, medium, and busy environments are analyzed.

Figure 4.6: Example of images captured by the hybrid system. First row *Street Busy* sequence and second row *Font Medium* sequence. Static cameras are mainly responsible for the global understanding of the scene while mobile cameras (drones) capture pedestrian images from the desired viewpoint.

tracker 0.002 s, distributed tracker manager 0.0015 s, class information fusion per tracker 0.00004 s, viewpoint control policy 0.005 s. The complete evaluation is conducted in one computer with an Intel® Core™ i7-9700 CPU @ 3.00GHz ×8 and a Nvidia GeForce GTX 1070. Both tracking modules, with mobile and static cameras, and classification modules, with mobile cameras, work in parallel. Provided that the poses of new targets are estimated and relayed to the mobile cameras within τ_h, our framework operates in real time. This isn't a strict constraint, as there's allowable latency; however, it's crucial that tracked targets remain within the recommended viewpoint FOV during any such delays.

In addition to the numerical results, Fig. 4.6 displays examples of images captured by the hybrid system at a specific time. The first row corresponds to images from the *Street* environment and the second row from the *Font* scenario. The overall understanding of the scene is mainly performed by the static cameras although the drones also assist in the distributed tracking, while the close-up person images are gathered from the mobile cameras. For example, in the first row, Drone2 correctly captures the viewpoint of the target with local identity 14, and in the second scenario, Drone1 accomplishes its goal with local identity 7. In the supplementary material, we include more examples of the complete framework working on both scenarios.

4.7 Conclusions

In this work, we have presented a collaborative hybrid system comprised of static and mobile cameras where all of them cooperate for pedestrian monitoring and high-resolution visualization of certain people's attributes. The proposed framework performs multi-camera distributed tracking providing a global understanding of the scene for which the static cameras are mainly responsible. We demonstrate that by allowing collaboration between cameras through sharing information once per cycle with the closest nodes, the multi-target tracking improves up to 21 points in the IDF1 metric and up to 15 points in MOTA. Global scene awareness and the current state of drones are used by the viewpoint control policy to provide a new position and orientation for mobile cameras whose goal is capturing a desired viewpoint of the people as quickly as possible. In comparison with a static multi-camera system, mobile cameras are able to capture the required viewpoint with higher precision in most of the scenes evaluated.

5

Conclusions and Future Work

5.1 Conclusions

This thesis has presented algorithms that enable robust and scalable coordination of multi-robot systems in navigation and active perception tasks. The first goal of this thesis was to enable the safe coordination of multiple drones through an efficient use of communication. In scenarios of point-to-point navigation, each robot is typically designated a single navigation task. However, in more intricate scenarios featuring multiple objectives, the tasks allocated to each robot can dynamically change, e.g. dynamic multi-target active classification. Consequently, the second goal of this thesis was to enhance motion-planning policies with properties that allow them to scale and efficiently adapt to time-varying numbers of tasks. When dealing with active perception tasks that involve dynamic targets, addressing their partial observability becomes essential. Therefore, the third goal of this thesis was to enhance our existing framework to better reflect real-world conditions, where targets need to be detected, tracked, and classified. Finally, multiple robots are necessary to realistically monitor large areas.

Chapter 2 introduced an efficient communication policy for decentralized multi-robot systems, enhancing coordination in collision avoidance tasks. This chapter addressed the challenge of efficiently enabling communication-based coordination. Previous models for multi-robot systems often relied on constant or periodic broadcasting of each robot's intentions, leading to unnecessary communication and inefficiency, especially when robots were not nearby. To overcome this, we developed a communication policy that determines when and with whom to communicate based on a communication policy trained through Reinforcement Learning. Each robot learned to selectively request trajectory plans from other robots posing potential risks while assuming constant velocities for non-selected robots. This learned policy used a novel attention-based neural architecture, allowing scalability across a varying number of robots while ensuring safe navigation. We coupled the communication policy with Non-Linear Model Predictive Control (NMPC) for motion planning to provide collision avoidance guarantees. The effectiveness of this method was

tested in simulation with 12 robots, showing 30% less communication than the closest competitor, minimally compromising safety. The method's scalability was evidenced by its successful application to lower/larger scale teams of robots than it was trained on, showing a degradation pattern comparable to that of the broadcasting policy in larger teams (e.g. less than 2% and 10% for 18 and 24 agents). The method also showed robustness to observation noise. This enabled us to perform real experiments, successfully demonstrating the transferability of our framework to physical quadrotors.

Chapter 3 presented a novel framework for the active classification of multiple dynamic targets using a drone and a "black-box" classifier. The primary challenge addressed was the computation of control inputs that guide the drone to informative viewpoints in dynamic environments. Previous methods were limited to static target classification or used predefined sets of target-centric viewpoints and trajectories, restricted by the need for previously established models to select viewpoints according to their estimated usefulness. Our approach learned a viewpoint control policy via Deep Reinforcement Learning (DRL), that recommended informative viewpoints which were then tracked by a low-level MPC. The class of the observed targets were estimated with the "black-box" classifier and aggregated with prior estimates through conflation. The proposed architecture used self-attention and set-function approximators for scalable encoding of dynamic environments, overcoming the requirements for prior knowledge in existing motion-planning strategies. The framework's effectiveness was tested against various baselines, including hand-crafted policies, ablations of our approach, and other existing learning-based motion-planning strategies. Simulation results with both simple and photo-realistic models showed the proposed method outperformed baselines in observing and classifying targets quickly. Additionally, its effectiveness in the photo-realistic environment indicates robustness in bridging the simulation-to-reality gap.

Chapter 4 presented a novel decentralized hybrid multi-camera system for surveillance and monitoring applications. The main limitations of traditional fixed camera networks are the presence of blind spots and back-lighting in the environment. Therefore, we proposed a decentralized hybrid framework that integrated both static and mobile cameras to enhance the system's ability to gather critical information actively and dynamically. All cameras in the network were coordinated to monitor people moving in the environment. They detected, distinguished, and coordinated to localize each person by comparing their local information. The mobile camera was guided through a viewpoint control policy (introduced in Chapter 3) towards viewpoints that maximized the observable semantic information from the observed targets. We implemented the framework in a photorealistic environment designed in Unreal Engine according to the guidelines provided in [19]. We enabled distributed communications among static and mobile cameras using the Robot Operating System (ROS), helping to bridge the gap between simulated and real-world applications. Results on the tracking performance of the framework in large environments demonstrated the advantages of using collaborative mobile cameras over purely static and individual camera setups, achieving an improvement of 21% and 15% respectively in target identification and tracking accuracy. The benefits of mobile cameras became particularly evident in crowded scenarios where static cameras often face challenges in

avoiding occlusions to obtain a high-confidence view of pedestrians. Mobile cameras were actively positioned to capture the desired viewpoint, resulting in over a 10% improvement in classified trackers compared to a static setup. Qualitatively, the mobile cameras provided superior target observation quality that was unmatched by the static framework.

Overall, these contributions address the key challenges in robust, scalable, and efficient coordination of multi-robot systems in navigation and active perception tasks. However, there are still several remaining challenges before we can realistically deploy robot swarms in real perception tasks, especially when the policy is learned. Toward that goal, possible directions of future work are described below.

5.2 Future work

This thesis contributed to the fields of motion planning and active perception, enabling scalable target triage and efficient coordination of multiple robots. Nevertheless, many open challenges remain to be addressed before reliable and robust multi-robot monitoring of large areas becomes a reality. Hereafter, we first assess the limitations and possible extensions of our work followed by recommendations on several future research directions for scalable and interpretable coordination, hierarchical control, and trading-off multiple objectives.

5.2.1 Limitations and extension

This thesis has shown advances in multi-robot motion planning coordination and scalable active perception. However, several limitations still offer possibilities for future research. Despite our last results showing two robots coordinating with static cameras to locate and classify dynamic targets, one notable gap is the lack of explicit coordination among mobile cameras to cover the region of interest, track identified targets efficiently, and actively search for remaining targets that have not been found yet [8, 11, 128]. Additionally, the current system's movement is limited to convex two-dimensional free space, which restricts its application to three-dimensional maps with obstacles, such as urban areas or inside buildings [129].

Our approach also assumes prior knowledge about the targets' dynamics, the classifier's behavior, and the viewpoints from which visual target information can be observed. This information was used to learn a viewpoint recommendation policy through model-free Reinforcement Learning. Being a family of generally data-hungry methods that require interacting with an environment, the use of accurate simulators was imperative to generate large amounts of data, learn accurate value functions, and obtain near-optimal policies. However, this information is often very specific to our problem and the assumptions taken, which affects the data, the flexibility of the learned policy, and therefore its transferability to real-life scenarios. On top of that, real-life environments are generally unstructured and uncertain, which makes simulating them for training, planning and prediction an open problem. This highlights the need for more general methods that can react and adapt seamlessly to sim-to-real changes during deployment instead of overfitting to the training environment.

Finally, a promising extension would involve integrating heterogeneous robots with diverse sensing and actuation capabilities [130]. Addressing these limitations could significantly refine the system's efficiency and effectiveness in real-world scenarios, broadening its range of applications.

5.2.2 Scalable and interpretable coordination policies

This work proposed to learn policies that can scale with the number of tasks and robots by using attention to adaptively encode the state, regardless of the number of tasks or other robots present. However, extending single-agent policy learning to multi-agent policy learning is far from trivial. Apart from the additional challenges coming from MARL, e.g. credit assignment and non-stationarity during training, the learned policies are limited to perform with the set of policies it has been trained with [11, 131]. Nevertheless, there is no guarantee their performance will not deteriorate when cooperating with other robots that follow policies that do not share the same weights or are trained with another initialization seed. With the rising energy demand, it is critical to obtain solutions that enable easy maintenance, replacement, and collaboration with other-party robot policies with minimal energy cost, e.g. without retraining the whole system. Explicit modeling of multi-agent interaction [132] would allow to obtain generalizable coordination among multiple single-agent policies. Future work may explore formulations of Dec-POMDPs and explicit multi-agent interaction models allowing predictable and controllable coordination of single-agent policies under more flexible conditions [133].

5.2.3 Modular Control

This thesis steers away from the trend of learning end-to-end general policies that replace the entire robotics pipeline (e.g. from perception to low-level actuators) in favor of the combination of learned and classic approaches for each sub-task. Similar to [60], it places special emphasis on obtaining learned high-level policies that enhance the performance of a low-level controller, e.g. by changing its constraints or the tracked goal. This allows our motion planner to solve complex tasks while respecting dynamic and collision constraints. Similar trends are followed by methods such as [134], where an MPC is used to redirect a learned policy to move according to dynamic and collision avoidance constraints. However, the learning algorithms in these methods do not account for the effects of the low-level controller and its imposed dynamic and task constraints on the learned policy, which tends to result in sample inefficiencies and artifacts. Thus, advancing learning methods incorporating such information is critical to enable learning policies for control. Two recent promising works further blend planning and gradient-based reinforcement learning by directly propagating gradients through a differentiable low-level controller [135], or by directly learning high-level policies that map states to hyperparameter settings of low-level controllers [136]. Nevertheless, there is still much work to be done in more general scenarios such as task and motion planning (TAMP) applications.

5.2.4 Hierarchical Control: Multi-objective encoding

Realistic robotics problems are generally multi-objective. Thus the methods proposed throughout this thesis solve multiple tasks at the same time, like classifying multiple

targets, or reaching a goal while avoiding obstacles. The former is a set of independent tasks that do not have any priority assigned to them. As seen in Chapter 3, this can be leveraged and encoded through attention. However, as shown in chapter 2, tasks in multi-objective scenarios are generally correlated, which requires coordination beyond task distribution. Furthermore, these objectives have generally different priority weights, which are difficult to trade off both at training and test time. Following the work from this thesis, future work on multi-objective cost encoding is necessary beyond the scalar representation cost and reward functions generally used in the fields of optimal control and reinforcement learning. How to represent, trade-off such objectives, and find Pareto front solutions have been widely studied in the field of Multi-Objective Decision Making, and its research is still ongoing [137]. However, previous efforts in RL only focus on how to modify the policy through training algorithmic modifications. Future work on careful formulation of the learned policy so that it represents the set of solutions on the Pareto front, from which to choose a policy according to the engineer's specifications without the need for retraining, could prove more intuitive to maintain and adapt to non-stationary multi-objective scenarios.

Bibliography

References

[1] Nils Boysen, Stefan Fedtke, and Stefan Schwerdfeger. Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, 43(1):1–58, 2021.

[2] Anne Goodchild and Jordan Toy. Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing co2 emissions in the delivery service industry. *Transportation Research Part D: Transport and Environment*, 61:58–67, 2018. Innovative Approaches to Improve the Environmental Performance of Supply Chains and Freight Transportation Systems.

[3] Dan Liu, Zhenghong Deng, Xinhua Mao, Yang Yang, and Evangelos I. Kaisar. Two-echelon vehicle-routing problem: Optimization of autonomous delivery vehicle-assisted e-grocery distribution. *IEEE Access*, 8:108705–108719, 2020.

[4] Giuseppe Fragapane, René de Koster, Fabio Sgarbossa, and Jan Ola Strandhagen. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*, 294(2):405–426, 2021.

[5] Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *IEEE Access*, 8:191617–191643, 2020.

[6] Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, and Yahya Zweiri. A survey on multi-robot coverage path planning for model reconstruction and mapping. *SN Applied Sciences*, 1(8):847, 2019.

[7] C. S. Chen, F. C. Lin, and C. J. Lin. The energy efficiency multi-robot system and disinfection service robot development in large-scale complex environment. *Sensors*, 23(12):5724, Jun 2023.

[8] Sara Casao, Abel Naya, Ana C Murillo, and Eduardo Montijano. Distributed multi-target tracking in camera networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1903–1909. IEEE, 2021.

[9] Drone Show Europe. Drone show europe. `https://droneshoweurope.com/`, 2023. Accessed on: November 2023.

[10] Christopher D. Hsu, Heejin Jeong, George J. Pappas, and Pratik Chaudhari. Scalable reinforcement learning policies for multi-agent control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4785–4791, 2021.

[11] Ekaterina Tolstaya, James Paulos, Vijay Kumar, and Alejandro Ribeiro. Multi-robot coverage and exploration using spatial graph neural networks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8944–8950, 2021.

[12] Amazon Science. How amazon robots navigate congestion. `https://www.amazon.science/latest-news/how-amazon-robots-navigate-congestion`, 2023. Accessed on: November 2023.

[13] Matthew Cavorsi, Orhan Eren Akgün, Michal Yemini, Andrea J. Goldsmith, and Stephanie Gil. Exploiting trust for resilient hypothesis testing with malicious robots. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7663–7669, 2023.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[15] Eduardo Camacho and Carlos Bordons. *Model Predictive Control*, volume 13. 01 2004.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[18] Qingbiao Li, Weizhe Lin, Zhe Liu, and Amanda Prorok. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540, 2021.

[19] Sara Casao, Andrés Otero, Álvaro Serra-Gómez, Ana C. Murillo, Javier Alonso-Mora, and Eduardo Montijano. A framework for fast prototyping of photo-realistic environments with multiple pedestrians. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9083–9089, 2023.

[20] Álvaro Serra-Gómez, Bruno Brito, Hai Zhu, Jen Jen Chung, and Javier Alonso-Mora. With whom to communicate: Learning efficient communication for multi-robot collision avoidance. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11770–11776. IEEE, 2020.

[21] Álvaro Serra-Gómez, Hai Zhu, Bruno Brito, Wendelin Böhmer, and Javier Alonso-Mora. Learning scalable and efficient communication policies for multi-robot collision avoidance. *Autonomous Robots*, 47(8):1275–1297, 2023.

[22] Hai Zhu, Francisco Martinez Claramunt, Bruno Brito, and Javier Alonso-Mora. Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 6(2):2256–2263, 2021.

[23] Mohamed S. Talamali, Arindam Saha, James A. R. Marshall, and Andreagiovanni Reina. When less is more: Robot swarms adapt better to changes with constrained communication. *Science Robotics*, 6, 2021.

[24] Thomas Wheeler, Ezhil Bharathi, and Stephanie Gil. Switching topology for resilient consensus using wi-fi signals. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2018–2024, 2019.

[25] Jur Van Den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Springer Tracts in Advanced Robotics*, volume 70, pages 3–19. 2011.

[26] Yan Yongjie and Zhang Yan. Collision avoidance planning in multi-robot based on improved artificial potential field and rules. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1026–1031. IEEE, 2009.

[27] Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.

[28] Hai Zhu and Javier Alonso-Mora. B-uavc: Buffered uncertainty-aware voronoi cells for probabilistic multi-robot collision avoidance. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 162–168. IEEE, 2019.

[29] Li Wang, Aaron D. Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

[30] Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.

[31] Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020.

[32] Mina Kamel, Javier Alonso-Mora, Roland Siegwart, and Juan Nieto. Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243. IEEE, 2017.

[33] Maayan Roth, Reid Simmons, and Manuela Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, page 786–793, New York, NY, USA, 2005. Association for Computing Machinery.

[34] Raphen Becker, Alan Carlin, Victor Lesser, and Shlomo Zilberstein. Analyzing myopic approaches for multi-agent communication. *Computational Intelligence*, 25:31–50, 2009.

[35] Abdallah Kassir, Robert Fitch, and Salah Sukkarieh. Communication-efficient motion coordination and data fusion in information gathering teams. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2016-November, pages 5258–5265. Institute of Electrical and Electronics Engineers Inc., nov 2016.

[36] Graeme Best, Michael Forrai, Ramgopal R. Mettu, and Robert Fitch. Planning-aware communication for decentralised multi-robot coordination. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1050–1057, 2018.

[37] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 6380–6391, 2017.

[38] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[39] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[40] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 2085–2087, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[41] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5887–5896. PMLR, 09–15 Jun 2019.

[42] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *ICML*, 2019.

[43] Michael Everett, Yu Fan Chen, and Jonathan P. How. Motion Planning among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3052–3059, 2018.

[44] Michael Everett, Yu Fan Chen, and Jonathan How. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access*, 9:10357–10377, 2021.

[45] Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. Graph neural networks for decentralized multi-robot path planning. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 11785–11792, 2020.

[46] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2145–2153, Red Hook, NY, USA, 2016. Curran Associates Inc.

[47] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 1495–1502, 2018.

[48] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *Advances in Neural Information Processing Systems*, (Nips):2252–2260, 2016.

[49] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 66–83, Cham, 2017. Springer International Publishing.

[50] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, 2018.

[51] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *36th International Conference on Machine Learning, ICML 2019*, 2019.

[52] Yuanzhao Zhai, Bo Ding, Xuan Liu, Hongda Jia, Yong Zhao, and Jie Luo. Decentralized multi-robot collision avoidance in complex scenarios with selective communication. *IEEE Robotics and Automation Letters*, 6(4):8379–8386, 2021.

[53] Chuangchuang Sun, M. Shen, and Jonathon P. How. Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11755–11762, 2020.

[54] Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[55] Daniel Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27, 12 2002.

[56] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4424–4429, 2016.

[57] Rose Wang, J. Chase Kew, Dennis Lee, Tsang-Wei Lee, Tingnan Zhang, Brian Ichter, Jie Tan, and Aleksandra Faust. Model-based reinforcement learning for decentralized multiagent rendezvous. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 711–725. PMLR, 16–18 Nov 2021.

[58] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, 39(7):856–892, 2020.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[60] Bruno Brito, Michael Everett, Jonathan How, and Javier Alonso-Mora. Where to go next: Learning a subgoal recommendation policy for navigation among pedestrians. *IEEE Robotics and Automation Letters*, 6(3):4616–4623, 2021.

[61] F. Gama, A. Marques, G. Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67:1034–1049, 2019.

[62] Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. In *International Conference on Learning Representations*, 2021.

[63] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3053–3062. PMLR, 10–15 Jul 2018.

[64] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 561–577, USA, 2018. USENIX Association.

[65] John Schulman, P. Moritz, S. Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2016.

[66] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[67] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[68] Alexander Domahidi and Juan Jerez. Forces professional. embotech gmbh (http://embotech. com/forces-pro), 2014.

[69] Hai Zhu, Jelle Juhl, Laura Ferranti, and Javier Alonso-Mora. Distributed multi-robot formation splitting and merging in dynamic environments. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9080–9086. IEEE, 2019.

[70] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[71] Inigo Alonso, Luis Riazuelo, Luis Montesano, and Ana Murillo. Domain adaptation in lidar semantic segmentation by aligning class distributions. In *Int. Conf. on Informatics in Control, Autom. and Robot.*, 2021.

[72] Timothy Patten, Michael Zillich, Robert C. Fitch, Markus Vincze, and Salah Sukkarieh. Viewpoint evaluation for online 3-d active object classification. *IEEE Robotics and Automation Letters*, 1(1):73–81, 2016.

[73] Marija Popović, Gregory Hitz, Juan Nieto, Inkyu Sa, Roland Siegwart, and Enric Galceran. Online informative path planning for active classification using uavs. In *IEEE Int. Conf. on Robotics and Automation*, pages 5753–5758, 2017.

[74] Nikolay Atanasov, Bharath Sankaran, Jerome Le Ny, George J. Pappas, and Kostas Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Trans. on Rob.*, 30(5):1078–1090, 2014.

[75] Timothy Patten, Wolfram Martens, and Robert Fitch. Monte carlo planning for active object classification. *Auton. Rob.*, 42(02):391–421, 2018.

[76] Juil Sock, Guillermo Garcia-Hernando, and Tae-Kyun Kim. Active 6d multi-object pose estimation in cluttered scenarios with deep reinforcement learning. In *IEEE/RSJ Int. Conf. on Intel. Rob. and Syst.*, pages 10564–10571, 2020.

[77] Qianli Xu et al. Towards efficient multiview object detection with adaptive action prediction. In *IEEE Int. Conf. on Robotics and Automation*, pages 13423–13429, 2021.

[78] David Kent and Sonia Chernova. Human-centric active perception for autonomous observation. In *IEEE Int. Conf. on Robotics and Automation*, pages 1785–1791, 2020.

[79] Alfonso Alcántara, Jesús Capitán, Rita Cunha, and Aníbal Ollero. Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles. *Robotics and Autonomous Systems*, 140:103778, 2021.

[80] Boseong Felipe Jeon, Dongsuk Shim, and H. Jin Kim. Detection-aware trajectory generation for a drone cinematographer. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1450–1457, 2020.

[81] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021.

[82] Adarsh Jagan Sathyamoorthy, Jing Liang, Utsav Patel, Tianrui Guan, Rohan Chandra, and Dinesh Manocha. Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors. In *IEEE Int. Conf. on Robotics and Automation*, pages 11345–11352, 2020.

[83] Yuxiang Cui, Haodong Zhang, Yue Wang, and Rong Xiong. Learning world transition model for socially aware robot navigation. In *IEEE Int. Conf. on Robotics and Automation*, pages 9262–9268, 2021.

[84] Yuying Chen, Congcong Liu, Bertram E. Shi, and Ming Liu. Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. *IEEE Rob. and Autom. Let.*, 5(2):2754–2761, 2020.

[85] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[86] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *IEEE Int. Conf. on Rob. and Autom.*, pages 6015–6022, 2019.

[87] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Int. Conf. on Mach. Learn.*, pages 3744–3753, 2019.

[88] Yoonchang Sung and Pratap Tokekar. Algorithm for searching and tracking an unknown and varying number of mobile targets using a limited fov sensor. In *IEEE Int. Conf. on Rob. and Autom.*, pages 6246–6252, 2017.

[89] Theodore Hill and Jack Miller. How to combine independent data sets for the same quantity. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 21(3):033102 (1–8), 2011.

[90] Michael Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *Extended Semantic Web Conference*, pages 593–607, 06 2018.

[91] Tuomas Haarnoja, Haoran Tang, P. Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[92] Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.

[93] Logan Engstrom et al. Implementation matters in deep rl: A case study on ppo and trpo. In *Int. Conf. on Learn. Repr.*, 2020.

[94] Marcin Andrychowicz et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *Int. Conf. on Learn. Repr.*, 2021.

[95] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 23–30, 2017.

[96] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51, 05 1998.

[97] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[98] Makehuman community., 2022. `http://www.makehumancommunity.org.`

[99] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[100] TensorFlow 2 YOLOv3 Mnist detection training tutorial. pylessons, 2020. `https://pylessons.com/YOLOv3-TF2-mnist.`

[101] Alhayat Ali Mekonnen, Frédéric Lerasle, and Ariane Herbulot. Cooperative passers-by tracking with a mobile robot and external cameras. *Computer Vision and Image Understanding*, 117(10):1229–1244, 2013.

[102] Xinzhao Li, Yuanqi Su, Yuehu Liu, Shaozhuo Zhai, and Ying Wu. Active target tracking: A simplified view aligning method for binocular camera model. *Computer Vision and Image Understanding*, 175:11–23, 2018.

[103] Ian D Miller, Fernando Cladera, Trey Smith, Camillo Jose Taylor, and Vijay Kumar. Stronger together: Air-ground robotic collaboration using semantics. *IEEE Robotics and Automation Letters*, 7(4):9643–9650, 2022.

[104] Yang Zhou, Jiuhong Xiao, Yue Zhou, and Giuseppe Loianno. Multi-robot collaborative perception with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):2289–2296, 2022.

[105] Javier Yu, Joseph A Vincent, and Mac Schwager. Dinno: Distributed neural network optimization for multi-robot collaborative learning. *IEEE Robotics and Automation Letters*, 7(2):1896–1903, 2022.

[106] Álvaro Serra-Gómez, Eduardo Montijano, Wendelin Böhmer, and Javier Alonso-Mora. Active classification of moving targets with learned control policies. *IEEE Robotics and Automation Letters*, 8(6):3717–3724, 2023.

[107] Yundong Guo, Zhenyu Liu, Hao Luo, Huijie Pu, and Jianrong Tan. Multi-person multi-camera tracking for live stream videos based on improved motion model and matching cascade. *Neurocomputing*, 492:561–571, 2022.

[108] Kha Gia Quach, Pha Nguyen, Huu Le, Thanh-Dat Truong, Chi Nhan Duong, Minh-Triet Tran, and Khoa Luu. Dyglip: A dynamic graph model with link prediction for accurate multi-camera multiple object tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13784–13793, 2021.

[109] Moonsub Byeon, Haanju Yoo, Kikyung Kim, Songhwai Oh, and Jin Young Choi. Unified optimization framework for localization and tracking of multiple targets with multiple cameras. *Computer Vision and Image Understanding*, 166:51–65, 2018.

[110] Ruiheng Zhang, Lingxiang Wu, Yukun Yang, Wanneng Wu, Yueqiang Chen, and Min Xu. Multi-camera multi-player tracking with deep player identification in sports video. *Pattern Recognition*, 102:107260, 2020.

[111] Alessio Xompero and Andrea Cavallaro. Cross-camera view-overlap recognition. In *European Conference on Computer Vision*, pages 253–269. Springer, 2022.

[112] Xiaohua Ge, Qing-Long Han, Xian-Ming Zhang, Lei Ding, and Fuwen Yang. Distributed event-triggered estimation over sensor networks: A survey. *IEEE transactions on cybernetics*, 50(3):1306–1320, 2019.

[113] Cristian Soto, Bi Song, and Amit K Roy-Chowdhury. Distributed multi-target tracking in a self-configuring camera network. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1486–1493. IEEE, 2009.

[114] Zhifei Li, Yan Liang, Linfeng Xu, and Shuli Ma. Distributed extended object tracking information filter over sensor networks. *International Journal of Robust and Nonlinear Control*, 33(2):1122–1149, 2023.

[115] Ahmed T Kamal, Jawadul H Bappy, Jay A Farrell, and Amit K Roy-Chowdhury. Distributed multi-target tracking and data association in vision networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1397–1410, 2015.

[116] Li He, Guoliang Liu, Guohui Tian, Jianhua Zhang, and Ze Ji. Efficient multi-view multi-target tracking using a distributed camera network. *IEEE Sensors Journal*, 2019.

[117] Melanie Schranz and Torsten Andre. Towards resource-aware hybrid camera systems. In *International Conference on Distributed Smart Cameras*, pages 1–7, 2018.

[118] Jing Li, Jing Xu, Fangwei Zhong, Xiangyu Kong, Yu Qiao, and Yizhou Wang. Pose-assisted multi-camera collaboration for active object tracking. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 759–766, 2020.

[119] Juan-Carlos Trujillo, Rodrigo Munguía, Eduardo Ruiz-Velázquez, and Bernardino Castillo-Toledo. A cooperative aerial robotic approach for tracking and estimating the 3d position of a moving object by using pseudo-stereo vision. *Journal of Intelligent & Robotic Systems*, 96:297–313, 2019.

[120] Niccoló Bisagno, Nicola Conci, and Bernhard Rinner. Dynamic camera network reconfiguration for crowd surveillance. In *International Conference on Distributed Smart Cameras*, pages 1–6, 2018.

[121] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Learning generalisable omni-scale representations for person re-identification. *Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[122] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 79–88, 2018.

[123] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[124] Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. Ad-vat+: An asymmetric dueling mechanism for learning and understanding visual active tracking. *IEEE Trans. on pattern analysis and machine intelligence*, 43(5):1467–1482, 2019.

[125] Wenhan Luo, Peng Sun, Fangwei Zhong, Wei Liu, Tong Zhang, and Yizhou Wang. End-to-end active object tracking and its real-world deployment via reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1317–1332, 2019.

[126] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: Clear mot metrics. *Journal on Image and Video Processing*, 2008.

[127] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35, 2016.

[128] Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J. Pappas. Anytime planning for decentralized multirobot active information gathering. *IEEE Robotics and Automation Letters*, 3(2):1025–1032, 2018.

[129] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. In *Robotics: Science and Systems (RSS)*, 2022.

[130] Amanda Prorok, M. Ani Hsieh, and Vijay Kumar. The impact of diversity on optimal control policies for heterogeneous robot swarms. *IEEE Transactions on Robotics*, 33(2):346–358, 2017.

[131] Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, page 1485–1494, 2023.

[132] Eduardo Sebastián, Thai Duong, Nikolay Atanasov, Eduardo Montijano, and Carlos Sagüés. Learning to Identify Graphs from Node Trajectories in Multi-Robot Networks. In *IEEE International Symposium on Multi-robot & Multi-agent Systems*, pages 1–7, 2023.

[133] Stefan Witwicki and Edmund Durfee. Towards a unifying characterization for quantifying weak coupling in dec-pomdps. In *10th International Conference on Autonomous Agents and Multiagent Systems 2011, AAMAS 2011*, volume 1, pages 29–36, 01 2011.

[134] Kim Wabersich and Melanie Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 07 2021.

[135] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[136] Yunlong Song and Davide Scaramuzza. Policy search for model predictive control with application to agile drone flight. *IEEE Transactions on Robotics*, 38(4):2114–2130, 2022.

[137] C. F. Hayes, R. Rădulescu, E. Bargiacchi, and et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(26), 2022.

Curriculum Vitæ

Álvaro Serra-Gómez was born in October 1993 in Barcelona, Spain. He received the M.Sc. (2019) degree both at the Polytechnic University of Catalonia in Automatic Control and Robotics and at the École Polytechnique de Paris in Data Science in the Computer Science and Applied Mathematics Departments.

In July 2019, he became a Ph.D. candidate at the Department of Cognitive Robotics, Delft University of Technology, Delft, the Netherlands. In his Ph.D. project, he worked on motion planning algorithms for the autonomous navigation of multi-agent systems and active perception under the supervision of Dr. Javier Alonso-Mora and Dr. Wendelin Böhmer. He was nominated for IEEE MRS Best Paper Award in 2023.

His research interests include learning-based hierarchical control, reinforcement learning, deep learning, and multi-robot systems.

List of Publications

Published

7. W. Jansma*, E. Trevisan, **Á. Serra-Gómez**, J. Alonso-Mora, *Interaction-Aware Sampling-Based MPC with Learned Local Goal Predictions*, in the International Symposium on Multi-Robot and Multi-Agent Systems (MRS), Boston, MA, USA, 2023. Best paper finalist.

6. S. Casao*, **Á. Serra-Gómez***, A. C. Murillo, W. Böhmer, J. Alonso-Mora, E. Montijano, *Distributed multi-target tracking and active perception with mobile camera networks, in Computer Vision and Image Understanding*, 2023, doi: 10.1016/j.cviu.2023.103876. 📄

5. **Á. Serra-Gómez***, H. Zhu, B. Brito, W. Böhmer, J. Alonso-Mora, *Learning scalable and efficient communication policies for multi-robot collision avoidance*, in Autonomous Robots, vol. 8, no. 6, pp. 3717-3724, 2023, doi: 10.1007/s10514-023-10127-3. 📄

4. **Á. Serra-Gómez***, E. Montijano, W. Böhmer, J. Alonso-Mora, *Active Classification of Moving Targets With Learned Control Policies*, in IEEE Robotics and Automation Letters (RA-L), vol. 8, no. 6, pp. 3717-3724, 2023, doi: 10.1109/LRA.2023.3271508. 📄

3. S. Casao*, A. Otero, **Á. Serra-Gómez**, A. C. Murillo, J. Alonso-Mora, E. Montijano, *A Framework for Fast Prototyping of Photo-realistic Environments with Multiple Pedestrians*, In IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, pp. 9083-9089, 2023, doi: 10.1109/ICRA46639.2022.9812190.

2. M. Lodel*, B. Brito, **Á. Serra-Gómez**, L. Ferranti, R. Babuska, J. Alonso-Mora, *Where to Look Next: Learning Viewpoint Recommendations for Informative Trajectory Planning, In Internationa*l Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, pp. 4466-4472, 2022, doi: 10.1109/ICRA46639.2022.9812190.

1. **Á. Serra-Gómez***, H. Zhu, B. Brito, J. J. Chung, J. Alonso-Mora, *With whom to communicate: Learning efficient communication for multi-robot collision avoidance*, In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, pp. 11770-11776, 2020, doi: 10.1109/IROS45743.2020.9341762. 📄

Under Review

2. M. Shi*, G. Chen, **Á. Serra-Gómez**, S. Yu, J. Alonso-Mora, *Evaluating Dynamic Environment Difficulty for Obstacle Avoidance Benchmarking*, to be submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems, 2024.

1. M. Boldrer*, **Á. Serra-Gómez**, L. Lyons, J. Alonso-Mora, L. Ferranti: *Rule-Based Lloyd Algorithm for Multi-Robot Motion Planning and Control with Safety and Convergence Guarantees*, submitted to IEEE Transactions on Robotics, 2023.

📄 Included in this thesis.