# Vision-based Learning for Drones in Collaborative and Adversarial Environments

**Jiaping Xiao**

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

**2024**

# Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

05/03/2024

Date

Jiaping Xiao

# Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

05/03/2024

Date

Asst. Prof. Mir Feroskhan

# Authorship Attribution Statement

This thesis contains material from 5 papers published in the following peer-reviewed journals and 1 preprint in arXiv in which I am listed as an author.

Chapter 2 is published as J. Xiao, R. Zhang, Y. Zhang, and M. Feroskhan, "Vision-based Learning for Drones: A Survey," in arXiv preprint (2023), arXiv:2312.05019. The contributions of the co-authors are as follows:

- I initialized and frameworked this survey and wrote the manuscript.
- I, R. Zhang and Y. Zhang collaborated in analyzing and writing the applications and challenges section.
- Prof. M. Feroskhan supervised the survey and contributed to discussing and revising the manuscript.

Chapter 3 is published as J. Xiao, P. Pisutsin and M. Feroskhan, "Collaborative Target Search With a Visual Drone Swarm: An Adaptive Curriculum Embedded Multistage Reinforcement Learning Approach," in IEEE Transactions on Neural Networks and Learning Systems (2023), doi: 10.1109/TNNLS.2023.3331370.

The contributions of the co-authors are as follows:

- I designed and implemented the methodology in simulation and wrote the manuscript.
- I and P. Pisutsin collaborated in conducting the physical experiments.
- Prof. M. Feroskhan helped initializing this project and contributed to revising the manuscript.

Chapter 4 is published as J. Xiao, P. Pisutsin and M. Feroskhan, "Toward Collaborative Multitarget Search and Navigation with Attention-Enhanced Local Observation" in Advanced Intelligent Systems, vol 6, issue 6, June 2024, 2300761, doi: 10.1002/aisy.202300761.

The contributions of the co-authors are as follows:

- I formulated the idea, developed and implemented the methodology in simulation and wrote the manuscript.
- I and P. Pisutsin collaborated in conducting the physical experiments.
- Prof. M. Feroskhan guided the project, helped improving the methodology, and contributed to revising the manuscript.

Chapter 5 is published as J. Xiao, J. H. Chee and M. Feroskhan, "Real-Time Multi-Drone Detection and Tracking for Pursuit-Evasion With Parameter Search," in IEEE Transactions on Intelligent Vehicles(2024), doi: 10.1109/TIV.2024.3360433. The contributions of the co-authors are as follows:

- I designed and implemented the methodology in simulation, conducted the physical experiments and wrote the manuscript.
- I and J. H. Chee collaborated in collecting the dataset and conducting the simulation experiments.
- Prof. M. Feroskhan supervised the project and contributed to refining the manuscript.

Chapter 6 is published as J. Xiao and M. Feroskhan, "Learning Multi-Pursuit Evasion for Safe Targeted Navigation of Drones," in IEEE Transactions on Artificial Intelligence(2024), doi: 10.1109/TAI.2024.3366871.

- I designed and implemented the methodology in simulation, conducted the physical experiments and analysis and wrote the manuscript.
- Prof. M. Feroskhan provided the oversight for the project and contributed to the manuscript's critical revision.

Chapter 7 is published as J. Xiao and M. Feroskhan, "Cyber Attack Detection and Isolation for a Quadrotor UAV With Modified Sliding Innovation Sequences," in IEEE Transactions on Vehicular Technology, vol. 71, no. 7, pp. 7202-7214, July 2022, doi: 10.1109/TVT.2022.3170725.

- I designed and implemented the methodology in simulation, conducted the analysis and wrote the manuscript.
- Prof. M. Feroskhan supervised the project, helped to formulate the experiments, and contributed to revising the manuscript in-depth.

05/03/2024

Date

Jiaping Xiao

# Acknowledgements

From a peaceful village nestled in the mountains and rivers to a bustling metropolis brimming with skyscrapers, what a waving and memorable study journey. I would like to express my deepest gratitude towards my supervisor, Professor Mir Feroskhan, for his endless guidance and support in determining the topic and plan of my research. Professor Mir has provided me with the precious opportunity to pursue my research dream, shaping this thesis with his profound knowledge, rigorous research attitude, and patience. He has always been considerate of my research needs, offering support unconditionally and influencing me on how to become a good scholar. Furthermore, he has cared for my career and family like a close friend.

I am also grateful for the assistance provided by my Thesis Advisory Committee (TAC), Professor Chen Lyu and Professor Bihan Wen. Their invaluable guidance and advice supported me throughout this long research journey. I enjoyed our research discussions and talks immensely. Special thanks to Professor Sameer Alam for his kindness and support during my Ph.D. qualification examination and teaching assistance.

My gratitude extends to all my colleagues and peers in the NTU Intelligent Cybernetics Group (ICG), who have made this journey as enjoyable and comfortable as home. I would especially like to thank Liangming Chen, Karanjot Singh, Jun Kiat Tan, Celescia, Clarence Wei Rui Teo, Yu Wang, Qianlei Jia, Lu Bai, Nuan Wen, Xiaoqiang Lu, Yuhang Zhang, and Rangya Zhang, for their companionship and support. My collaborators, Phumrapee Pisutsin, Yuming Zheng, Jian Hui Chee, Yi Xuan Marcus Tan, Ho Chi Ng, and Thio Teng Kiat, deserve special mention for their collaborative efforts, meaningful discussions, and supportive work over these four years. I am also grateful to our school staff, especially Ms. Isadira Mohamed Zaid, Mr. Ng Tian Seng, Mr. Seet Thian Beng, Mr. Foo Jong Hin, Mr. Poon Weng Wai, and Ms. Halimatun Bte Ma'arof, for their endless support during my

stays in MAE. My sincere thanks go to all my friends at Nanyang Technological University, especially Xinlaing Zhou, Chenyu Liu, Changhong Linghu, Yong Wang, Zhihua Deng, and Zipeng Huang, for their companionship and enjoyable hiking in Singapore. I also thank all the leaders and friends in the NTU graduate student association (GSA) where I served for three years.

Lastly, I extend my deepest appreciation to my parents for their selfless support and help. My everlasting love goes to my wife, Yuwan Li, for all the joys and challenges we've shared. Their self-sacrifices will always be deeply engraved in my heart. I send my final greetings to my baby Muzhe Xiao, wishing him a happy and lovely life. Thank you all.

*"Out of clutter, find simplicity. From discord, find harmony. In the middle of difficulty, lies opportunity."*

—Einstein, Albert

To my dear family and friends

# Abstract

Drones as advanced cyber-physical systems (CPSs) are undergoing a transformative shift with the advent of vision-based learning, a field that is rapidly gaining prominence due to its profound impact on drone autonomy and functionality. From agile forest navigation to becoming world-level racing champions, the rapid development of vision-based learning in drones represents a significant leap forward in the realm of intelligent systems, with applications extending across various sectors. Despite significant advancements in vision-based learning for single drone systems, the potential for multi-drone systems has been underexplored, considering multi-drone systems show advantages in performing large-scale and complex tasks with improved efficiency and resilience. Similar to natural biological swarms, which always need to consider behaviours in collaborative and adversarial environments, multi-drone systems are required to perform collaborations and maintain safety and security against physical and cyber attacks from adversarial environments. Several critical challenges emerge in developing effective, safe, and secure vision-based learning strategies for multi-drone systems. Hence, the core motivation of this thesis is to explore the potential of vision-based learning for multi-drone systems in real-world operations, tackling the challenges of effective collaboration, rapid response to adversarial agents, and cybersecurity.

This thesis addresses these challenges by proposing novel vision-based learning strategies that leverage the collective capabilities of visual drone swarms for enhanced task performance and resilience against both physical and cyber attacks. This thesis contributes to the development of an adaptive curriculum embedded multistage learning (ACEMSL) approach for collaborative target search, a posthumous mix-credit assignment with attention (POMA) framework for collaborative multi-target search, a real-time multi-drone detection and tracking system (Drone-YOLOSORT) for adversarial environments, and an asynchronous multistage deep reinforcement learning (AMS-DRL) approach for the multi-pursuit evasion mission. Specifically, to address the sparse reward issue in 3D space exploration, we

embed the sample-efficient adaptive curriculum into multistage learning to gradually guide agents in achieving effective collaboration. To address the posthumous credit assignment in environments with a variable number of agents, we integrate adaptive curriculum learning (ACL), mixed credit assignment (MCA), and attention-enhanced observation (AEO), allowing agents to balance individual efforts and group contributions in changing environments. To ensure the safety of drones flying in adversarial environments with preset constraints, we first collect the multi-class drone dataset "ICG-Drone" and fine-tune the detection model with a genetic algorithm, and then compile the detection and tracking models into a ROS package with edge acceleration. The AMS-DRL approach evolves the navigation policy towards Nash equilibrium by learning against adversaries in a bipartite way. Lastly, a general state estimation-based method with modified sliding innovation sequences (MSIS) for cyber attack detection and isolation is introduced, underscoring the importance of security in drone operations.

This thesis also focuses on improving the generalization and the Sim2Real capabilities of trained policies on unseen environments. Through extensive simulations and real-world experiments, this thesis demonstrates the superior performance and effectiveness of these proposed approaches, highlighting their potential to revolutionize the deployment of vision-based drones in a variety of applications.

By pushing the boundaries of what is possible with vision-based learning for multi-drone systems, this thesis lays the groundwork for future innovations in the field of not only autonomous drones but also a wide range of intelligent systems, such as self-driving cars and autonomous underwater vehicles, ensuring that they can operate more effectively and securely in increasingly complex and dynamic environments.

# Contents

**Acknowledgements** ix

**Abstract** xiii

**List of Figures** xxi

**List of Tables** xxvii

**Symbols and Acronyms** xxix

**1 Introduction** 1
- 1.1 Background . . . 1
- 1.2 Motivation and objectives . . . 4
- 1.3 Major contributions . . . 6
- 1.4 Thesis outline . . . 8

**2 Related works** 11
- 2.1 Background . . . 11
  - 2.1.1 Quadrotor drone dynamics . . . 11
  - 2.1.2 Visual perception . . . 12
    - 2.1.2.1 Lidar . . . 12
    - 2.1.2.2 Camera . . . 15
  - 2.1.3 Machine learning . . . 15
  - 2.1.4 Reinforcement learning . . . 16
    - 2.1.4.1 Single-agent RL . . . 17
    - 2.1.4.2 Multiagent RL . . . 18
    - 2.1.4.3 Policy gradient theorem . . . 19
    - 2.1.4.4 Proximal policy optimization . . . 20
- 2.2 Object detection with visual perception . . . 20
  - 2.2.1 Multistage algorithms . . . 21
  - 2.2.2 One-stage algorithms . . . 23
  - 2.2.3 Vision transformer . . . 25
  - 2.2.4 Summary . . . 27
- 2.3 Vision-based control . . . 28

2.3.1 Indirect methods . . . 29
2.3.2 End-to-end methods . . . 31
2.3.3 Semi-direct methods . . . 36
2.4 Applications and challenges . . . 40
2.4.1 Single drone application . . . 40
2.4.1.1 Obstacle avoidance . . . 40
2.4.1.2 Surveillance . . . 40
2.4.1.3 Search and rescue . . . 42
2.4.1.4 Environmental monitoring . . . 42
2.4.1.5 Industrial inspection . . . 42
2.4.1.6 Autonomous racing . . . 43
2.4.2 Multi-drone application . . . 44
2.4.2.1 Coordinated surveying . . . 45
2.4.2.2 Cooperative tracking . . . 45
2.4.2.3 Synchronized monitoring . . . 46
2.4.2.4 Disaster response . . . 46
2.4.3 Heterogeneous systems application . . . 47
2.4.3.1 Localization and path planning . . . 47
2.4.3.2 Precise landing . . . 48
2.4.3.3 Inspection and detection . . . 49
2.5 Conclusion . . . 49

**I Vision-based learning in collaborative environments 51**

**3 Learning target search in collaborative environments 53**
3.1 Introduction . . . 54
3.2 Related work . . . 58
3.3 Methodology . . . 59
3.3.1 Problem formulation . . . 59
3.3.1.1 Environment setup . . . 60
3.3.1.2 Observation space . . . 62
3.3.1.3 Action space . . . 62
3.3.1.4 Reward function . . . 63
3.3.2 Network architecture . . . 65
3.3.3 Adaptive curriculum embedded multistage learning . . . 66
3.3.3.1 Adaptive embedded curriculum (AEC) . . . 66
3.3.3.2 Stage 1 . . . 68
3.3.3.3 Stage 2 . . . 69
3.3.3.4 Training algorithm . . . 69
3.3.4 Domain randomization . . . 70
3.4 Experiments and results . . . 71
3.4.1 Simulation experiments . . . 72
3.4.1.1 Settings . . . 72

3.4.1.2 Baselines and ablations . . . 72
3.4.1.3 Training . . . 73
3.4.1.4 Validation . . . 76
3.4.1.5 Testing . . . 79
3.4.2 Physical experiments . . . 79
3.4.3 Analysis and discussion . . . 83
3.4.3.1 Stability analysis . . . 83
3.4.3.2 Generalization capability analysis . . . 84
3.4.3.3 Failure case analysis . . . 84
3.5 Conclusion . . . 85

**4 Learning multitarget search in collaborative environments 87**
4.1 Introduction . . . 88
4.2 Related work . . . 90
4.2.1 Multiagent reinforcement learning . . . 90
4.2.2 Credit assignment . . . 91
4.3 Preliminaries . . . 92
4.3.1 Multiagent reinforcement learning . . . 92
4.3.2 Attention mechanism . . . 94
4.4 Methodology . . . 95
4.4.1 Problem formulation . . . 95
4.4.1.1 Observation and action space . . . 96
4.4.1.2 State transition function . . . 96
4.4.2 Posthumous mix-credit assignment with attention . . . 97
4.4.2.1 Adaptive curriculum learning . . . 97
4.4.2.2 Mixed credit assignment . . . 98
4.4.2.3 Attention enhanced observation . . . 100
4.4.2.4 Network architecture . . . 101
4.4.2.5 CTDE with behavior manger . . . 101
4.5 Simulation experiments . . . 102
4.5.1 Settings . . . 102
4.5.2 Environment setup . . . 104
4.5.3 Hyperparameters . . . 105
4.5.3.1 Hyperparameters for RL . . . 105
4.5.3.2 Hyperparameters for POMA . . . 106
4.5.4 Domain randomization . . . 106
4.5.5 Main results . . . 108
4.5.5.1 Baseline comparison . . . 108
4.5.5.2 Ablation study . . . 109
4.5.5.3 Qualitative analysis . . . 109
4.5.5.4 Quantitative analysis . . . 110
4.6 Physical experiments . . . 111
4.6.1 Settings . . . 111

4.6.2 Physical implementation . . . 112
4.6.3 Main results and discussion . . . 113
4.7 Conclusion . . . 113

**II Vision-based learning in adversarial environments 115**

**5 Learning drone detection and tracking in adversarial environments 117**
5.1 Introduction . . . 118
5.2 Related work . . . 120
5.3 ICG-Drone dataset . . . 123
5.3.1 Data collection . . . 123
5.3.2 Data annotation . . . 125
5.3.3 Data splitting . . . 125
5.3.4 Data analysis . . . 127
5.4 Methodology . . . 127
5.4.1 Evaluation metrics . . . 128
5.4.2 Hyperparameter tuning with GA search . . . 129
5.4.3 Multi-drone tracking . . . 130
5.4.4 ROS package . . . 131
5.5 Experiments and results . . . 132
5.5.1 Implementation settings . . . 132
5.5.2 Benchmark with baseline models . . . 133
5.5.3 Performance with parameter search . . . 137
5.5.4 Model deployment with edge accelerator . . . 140
5.6 Conclusion . . . 141

**6 Learning multi-pursuit evasion in adversarial environments 143**
6.1 Introduction . . . 144
6.2 Related work . . . 146
6.2.1 Obstacle avoidance . . . 146
6.2.2 Pursuit-evasion game . . . 148
6.2.3 Reinforcement learning with self-play . . . 149
6.3 Multi-pursuit evasion with targeted navigation problem . . . 150
6.3.1 Problem formulation . . . 150
6.3.1.1 MDP in adversarial games . . . 151
6.3.2 RL description of MPETN . . . 152
6.3.2.1 Environment setup . . . 153
6.3.2.2 State space . . . 154
6.3.2.3 Observation space . . . 154
6.3.2.4 Action space . . . 154
6.3.2.5 State transition function . . . 155
6.3.2.6 Reward function . . . 155
6.4 Asynchronous multistage deep reinforcement learning approach . . . 156

6.4.1 AMS-DRL algorithm . . . 156
6.4.1.1 Cold-start learning stage . . . 157
6.4.1.2 Asynchronous learning Stage . . . 157
6.4.2 Network architecture . . . 160
6.4.3 Training scheme . . . 160
6.5 Experiments and results . . . 160
6.5.1 Simulation experiments . . . 161
6.5.1.1 Settings . . . 161
6.5.1.2 Baselines . . . 162
6.5.1.3 Ablations . . . 165
6.5.1.4 Training results . . . 165
6.5.1.5 Testing results . . . 166
6.5.2 Physical experiments . . . 168
6.5.2.1 Settings . . . 169
6.5.2.2 Sim2Real performance evaluation . . . 170
6.5.2.3 Results and discussion . . . 171
6.6 Conclusion . . . 172

**7 Cyber attack detection and isolation in adversarial environments 173**
7.1 Introduction . . . 174
7.2 Related work . . . 176
7.3 Preliminaries and problem formulation . . . 177
7.3.1 Quadrotor dynamics . . . 177
7.3.2 Extended kalman filter . . . 180
7.3.3 Quadrotor state estimation based on the EKF . . . 181
7.3.4 Control system . . . 183
7.3.5 Attack model . . . 184
7.3.5.1 Random attack . . . 185
7.3.5.2 False data injection (FDI) attack . . . 186
7.3.5.3 Denial of service (DoS) attack . . . 186
7.4 Attack detection and isolation with modified sliding innovation sequences . . . 187
7.4.1 Chi-square detector . . . 187
7.4.2 Modified sliding innovation sequences detector . . . 189
7.4.2.1 Calibration phase . . . 189
7.4.2.2 Detection phase . . . 190
7.4.3 Sample covariance-based cyber attack isolation . . . 193
7.5 Simulation results . . . 193
7.5.1 Absence of cyber attacks . . . 197
7.5.2 Cyber attacks on IMU sensor . . . 200
7.5.2.1 Random attack . . . 200
7.5.2.2 FDI attack . . . 200
7.5.2.3 DoS attack . . . 201

7.5.3 Cyber attacks on thrust actuator . . . 201
7.5.4 Cyber attacks on position sensors . . . 201
7.5.5 Isolation of cyber attacks . . . 202
7.5.6 Summary . . . 202
7.6 Conclusion . . . 203

**8 Conclusions and future work** **205**
8.1 Conclusions . . . 205
8.2 Future work . . . 208

**List of Author's Awards, Patents, and Publications** **211**

**Bibliography** **213**

# List of Figures


1.1 Applications of vision-based drones. (a) Parcel delivery [1]; (b) Power grid inspection [2]; (c) Precision agriculture [3]; (d) Intelligent warehouse management [4]. . . . . 2
1.2 General framework of vision-based drones . . . . . 3

2.1 Lidar on drones for visual perception. (a) A typical surrounding lidar; (b) Generated point cloud with lidar. . . . . 13
2.2 Visual perception with cameras for drones. (a) Visual Odometry for drones' positioning; (b) Object detection for drones' obstacle avoidance with event camera [5]. . . . . 14
2.3 The reinforcement learning process. . . . . 16
2.4 (a) R-CNN neural network architecture [6]; (b) Fast R-CNN neural network architecture [7]. . . . . 22
2.5 Faster RCNN neural network architecture [8]. . . . . 23
2.6 YOLO network architecture [9]. . . . . 24
2.7 SSD network architecture [10]. . . . . 24
2.8 Swin Transformer V2 framework [11]. . . . . 25
2.9 Air-to-air object detection of micro-UAVs with a monocular camera [12]. . . . . 26
2.10 Vision-based control methods for drone applications. Based on the ways of visual perception and control, the methods can be divided into indirect methods, semi-direct methods, and end-to-end methods. 28
2.11 A vision-based drone is traversing through a cluttered indoor environment with generated point cloud maps and online path planning [13]. . . . . 31
2.12 Depth maps generated from the viewpoint of a drone [14]. . . . . 31
2.13 Artificial potential field (APF) methods in drones' obstacle avoidance [15] (a) Generated artificial potential field; (b) Flight trajectory based on the APF. . . . . 32
2.14 Quadrotor drone flying in a wild environment with end-to-end reinforcement learning [16]. (a) Training process in the simulation platform; (b) Real flight test in the wild snow environment. . . . . 32
2.15 The neural network architectures of end-to-end reinforcement learning methods. (a) Target driven navigation with ResNet50 encoder[17]; (b) Target driven visual navigation for a robot with shallow CNN encoder [18]. . . . . 34

2.16 Training process used in [16] to fly an agile drone through forest. . . 35
2.17 A drone avoids the obstacles in the forest with semi-direct methods [19]. . . . . 37
2.18 A drone is tracking and chasing a target drone with object detection and deep reinforcement learning [20]. . . . . 37
2.19 Via deep learning, a 3D space generated from the monocular image for obstacle avoidance [21]. . . . . 39
2.20 Applications of single drone. (a) Surveillance [22]; (b) Search and Rescue [23]; (c) Environmental Monitoring [24]; (d) Autonomous Racing [25]. . . . . 41
2.21 Applications of multi-drone. (a) Coordinated Surveying [26]; (b) Cooperative Tracking [27]; (c) Synchronized Monitoring [28]; (d) Disaster Response [29]. . . . . 44
2.22 Applications of heterogeneous systems. (a) UAV-UGV path planning [30]; (b) UAV-UGV precise landing [31]; (c) UAV-UGV inventory inspection [32]; (d) UAV-UGV object detection [33]. . . . . 48

3.1 (a) A honeybee swarm is searching for nectar collaboratively [`https://encr.pw/ZK1oB`]. (b) A visual drone swarm is searching for a target parcel box controlled by reinforcement learning. . . . . 56
3.2 A bird's-eye view of the base environment, which consists of visual drone agents, obstacles, and a target. The furniture objects are the obstacles, such as sofas, the bed, and tables; The cubic textured with AprilTag is the target that drone agents need to find and reach. Each object is bounded by a collider for collision detection. Target can spawn in a closed sphere set $\boldsymbol{B}$ which is easily seen, or a discrete set $\boldsymbol{C}$ consists of several hidden locations marked with numbers $1-9$. 61
3.3 Illustration of the neural network architecture for target search, which consists of the image encoder, the policy network, and the memory network. . . . . 64
3.4 Illustration of the training and experiment framework. . . . . 71
3.5 Learning curves of the PPO-AEC (ours), PPO-AEC-SW (ours), the baseline TD-A3C and the ablation PPO-w/o-AEC, SAC-AEC in the single-agent training. Mean and standard deviation (shaded) over 3 independent runs conducted every 12 million training steps (episodes). Note that due to the high computational cost of SAC-AEC, we only consider 3 million training episodes for it. The moving SR is illustrated for PPO-AEC-SW in (d). All these values are the means over $3 \times 6 = 18$ training environments. From (c) and (d), for PPO-AEC, when the cumulative SR crosses the $sr_h = 0.85$ (4.5 million steps), the TDL increases with $\delta_\epsilon = 0.1$ and converges to 0.48 (9.4 million steps) with the SR of 80%. (a) Cumulative reward; (b) Episode length; (c) Task difficulty level; (d) Cumulative success rate. . . . . 75

3.6 Stage 1 training curves with single-agent over 3 million steps (episodes). Note that the same initial policy is used for both ACEMSL and CM3 in the Stage 2 training. (a) Cumulative reward; (b) Episode length; (c) Task difficulty level; (d) Cumulative success rate. . . . . 76
3.7 Learning curves of the ACEMSL (ours), the baseline CM3 and the ablation "Direct" in the multiagent training. Mean and standard deviation (shaded) over 3 independent runs conducted every 12 million training steps (episodes). (a) Cumulative reward; (b) Episode length; (c) Task difficulty level; (d) Cumulative success rate. . . . . 77
3.8 The snapshots of the visual drone swarm performing the CTS. The images perceived by the cameras of Drone 1 and Drone 2 are labeled as 1 and 2, respectively at the top right side of each snapshot. . . . 78
3.9 The trajectories of the drone swarm after successfully performing the CTS in four tests with different initial conditions (Test 1-2 from Env 1 and Test 3-4 from Env 2). The stability of the search policy is demonstrated with different initial positions and forward directions for the drones and the target. . . . . 81
3.10 Image blur and loss of the video stream cause the failure of tasks. (a) Obstacle blur; (b) Target blur. . . . . 82

4.1 A drone swarm is inspecting parcels in a warehouse where efficient collaborations are required [4]. . . . . 88
4.2 Overview of our POMA framework. A centric behavior manager controls the TDL of the environment with an ACL feedback module. The MCA module integrates the individual critic into the group critic to address the individual-group conflicts. The AEO encoder with a self-attention mechanism improves the generation capability of policy with variable neighbors. . . . . 95
4.3 The MCA architecture combines the group critic and individual critic, which optimizes the team contribution while encouraging individual exploration. . . . . 98
4.4 The simulation environment for the CMTSN with 3 drones, 3 targets (3D3T), and 3 obstacles (3O) of different sizes. The task is successful only if all targets are found and approached. The left 3 views are drones' visual perception. . . . . 104
4.5 Learning curves of IPPO, MA-POCA, POMA (ours) and different ablations. . . . . 105
4.6 Snapshots of simulation with 3D3T. The time sequence is from (a) to (d). It shows significant collaborative behaviors (Agent 1 moves back to provide space for Agent 2 to navigate to the final target as observed in (c)). . . . . 107
4.7 The 6D9T6O mission in a $10 \times 10 \times 3m$ room. . . . . 110
4.8 The physical implementation framework. . . . . 111

4.9 Snapshots of physical experiments in different missions. The trained policy model shows strong generalization capability even only trained in a 3 drones vs. 3 targets scenario. (a) 1 drone vs. 1 target; (b) 1 drone vs. 2 targets; (c) 2 drones vs. 2 targets. . . . . 111
4.10 (a) Physical experiments on 3 drones vs. 3 targets (3D3T) mission. The initial positions are labeled with directions. (b) Trajectories of 3 Tello drones from the motion capture system. Their trajectories show significant collaborative behaviors as Tello 3 attempts to navigate to Target 1 considering Tello 2 already found Target 3 and Target 2. Meanwhile, Tello 1 tries to find other hidden targets. . . . 112

5.1 The multi-drone detection and tracking outdoors with edge accelerator. All micro drones are tracked even with low-resolution video streams. . . . . 120
5.2 Types of drones in the dataset for detection and tracking. (a) DJI F450 evader drone; (b) Tello pursuer drone; (c) QAV250 evader drone; (d) View from the Tello in the pursuit-evasion scene. . . . . 123
5.3 Distribution of the dataset in different sources with number count. . 124
5.4 Annotated image using LabelImg of 4 Tello drones. . . . . 125
5.5 Annotated image using LabelImg of DJI F450 and Tello drone. . . . 126
5.6 Barplot of the dataset after split. . . . . 126
5.7 Label frequency and label variation of XYWH. . . . . 127
5.8 Genetic algorithm process for parameter search. . . . . 128
5.9 Proposed ROS package pipeline for Drone-YOLOSORT. . . . . 131
5.10 Predictions of "ICG-Drone" dataset with the baseline YOLOv5s model. The baseline model is insufficient to detect all small Tello drones. . . . . 133
5.11 YOLOV5 comparisons of mean of mAP50 and mAP50-95. . . . . 134
5.12 YOLOV8 comparisons of mean of mAP50 and mAP50-95. . . . . 134
5.13 The evolution process of hyperparameter with GA. . . . . 137
5.14 Training and validation loss curves after hyperparameter tuning. . . 137
5.15 Parameter search comparisons with mean of mAP50 and mAP50-95. 138
5.16 Procedure for conversion from PyTorch model to engine file. . . . . 139
5.17 Indoor results for Tello drones on Jetson Xavier NX. The Drone-YOLOSORT can track all drones with distinct IDs. . . . . 139
5.18 Outdoor results for detecting both F450 and Tello drones. . . . . 139

6.1 Evading multiple pursuers with learned policy. The runner (evader) is labeled as white, and the two chasers (pursuers) are labeled as blue. The target is a box with AprilTag. . . . . 145

6.2 The simulation environment for the 3D multi-pursuit evasion with targeted navigation scenario, which consists of two blue chaser drones (pursuers) and one white runner drone (evader) flying towards the target (red box). All simulated objects are bound with box colliders ($L \times W \times H$) for collision detection. The chasers aim to crash down the runner while the runner is required to safely reach the target. . 153
6.3 Illustration of the AMS-DRL training sequence for the shared chaser policy and the runner policy. . . . . 159
6.4 Illustration of the neural network architecture for chaser agents and the runner agent. . . . . 159
6.5 Average reward of AMS-DRL during the training. The training phases are denoted by $S_0$-$S_1$. (a) Runner policy; (b) Chaser policy. 163
6.6 Total steps of AMS-DRL during the training. The training phases are denoted by $S_0$-$S_1$. (a) Runner policy; (b) Chaser policy. . . . . 164
6.7 Runner success rate with different relative maximum speed. . . . . 166
6.8 Runner success rate with different numbers of chaser drones. . . . . 166
6.9 The spacial geometry distribution of agents in the physical tests. . . 168
6.10 The testing result distribution with different spacial geometry configurations. (a) The cases scatter; (b) the success rate heatmap. The runner is fixed at the origin $[0.0, 0.0]$ while the chasers are mirrored along the y-axis. . . . . 168
6.11 The snapshots of multi-pursuit evasion with targeted navigation in physical experiments. The white runner drone eventually reached the target box without being captured by intelligent blue chasers. The beginning of the arrow is the current position, while the end of arrow is the position in next frame. . . . . 169

7.1 The quadrotor model, including the definition of the body frame, rotor thrust vector and gravitation $g_E$. . . . . 178
7.2 Illustration of the hybrid system for a quadrotor under cyber attacks. 184
7.3 3D trajectories (the desired trajectory, the trajectory from position sensors and the real trajectory) of the quadrotor in the absence of cyber attacks. . . . . 195
7.4 Detection results of the quadrotor in the absence of cyber attacks. **(a)** the chi-square detector and the SUM detector; **(b)** the MSIS detectors without calibration phase; **(c)** the MSIS detectors with calibration phase. . . . . 196
7.5 Detection of cyber attacks on the IMU sensor. In each sub-figure, **top**: the sensor readings (e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **middle**: the chi-square detector and the SUM detector; **bottom**: the MSIS detectors with Euclidean norm (MSIS-L2) and infinity norm (MSIS-LI). . . . . 197

7.6 Detection of cyber attacks on the thrust actuator. In each sub-figure (a), (b) and (c), **top**: the sensor readings a(e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **middle**: the chi-square detector and the SUM detector; **bottom**: the MSIS detectors with Euclidean norm (MSIS-L2) and infinity norm (MSIS-LI). . . . . 198
7.7 Detection of cyber attacks on position sensors. In each sub-figure (a), (b) and (c), from top to bottom, **1**: the sensor readings (e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **2**: the chi-square detector and the SUM detector; **3**: The MSIS detectors; **4**: The 3D trajectory of the quadrotor. In (a), the quadrotor becomes unstable and crashes to the ground under the random attack. . . . . 199
7.8 Isolation of three types of cyber attacks on thrust actuator for a dynamic quadrotor. **(a)** the sensor readings (e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **(b)** the chi-square detector and the SUM detector; **(c)** the MSIS detectors with window length $n = 100$; **(d)** the MSIS detectors with window length $n = 10$. . . . . 203

# List of Tables


2.1 Summary of vision-based control methods in drone technology . . . 38

3.1 Domain randomization . . . 70
3.2 Parameters used for the AEC . . . 72
3.3 Hyperparameters used for training . . . 73
3.4 Absolute training runtime of different algorithms (seconds) . . . 77
3.5 Performance (SR: % / TTR: Avg (steps)) with various task difficulty levels (TDLs) in simulation . . . 78
3.6 Performance of ACEMSL (SR: % / TTR: Avg. (steps)) with various room dimensions in simulation . . . 79
3.7 Computing performance (Avg. ± Std.) of devices . . . 80
3.8 Target search performance in physical experiments . . . 82

4.1 Reward function used for various methods. . . . 100
4.2 Hyperparameters for all RL algorithms . . . 106
4.3 Parameters for the POMA and environment . . . 107
4.4 Domain randomization parameters . . . 108
4.5 Comparison with baselines and ablation models over different testing missions with 3 obstacles (ASR ↑ /ATS ↓). . . . 108
4.6 Performance comparison in large-scale missions (ASR ↑ /ATS ↓). . . . 109

5.1 Details of video acquisition . . . 122
5.2 Comparison of existing UAV datasets with ours . . . 122
5.3 Settings of Colab . . . 132
5.4 Settings of workstation . . . 132
5.5 Edge computer setting for Drone-YOLOSORT . . . 134
5.6 Default hyperparameters of YOLOv5s . . . 135
5.7 Performance metrics for various models on ICG-Drone-2c . . . 135
5.8 Performance metrics for various models on ICG-Drone-3c . . . 136
5.9 Model training details in workstation . . . 136
5.10 Performance of YOLOV5s with hyperparameters of default, random search, and GA search (ICG-Drone-2c/ICG-Drone-3c) . . . 138
5.11 Performance of YOLOV5s with hyperparameters of GA search in various public datasets (Default/GA search) . . . 138
5.12 Inference time comparison between PyTorch model and TensorRT engine . . . 140

6.1 Basic specifications of drones in simulation . . . . . 161
6.2 Parameters of AMS-PPO . . . . . 162
6.3 Navigation performance comparison (success rate of the runner $sr^r$ %) against various chaser policies (bold means the best runner policy while red means the best chaser policy) . . . . . 165
6.4 Comparison of inference runtime in various devices (ms). . . . . 165
6.5 Navigation performance of runner in real-time flight . . . . . 171

7.1 Data-driven cyber attacks . . . . . 185
7.2 Simulation parameters . . . . . 194
7.3 Performance comparison with $n = 10$, $k = 800$ . . . . . 199
7.4 Performance of the MSIS-LI detector with different $n$ . . . . . 200

# Symbols and Acronyms

## Symbols

| | |
|---|---|
| $\mathcal{R}^n$ | the $n$-dimensional Euclidean space |
| $\|\cdot\|$ | the 2-norm of a vector or matrix in Euclidean space |
| $\odot$ | the Hadamard (component-wise) product |
| $\otimes$ | the Kronecker product |
| $\langle\cdot,\cdot\rangle$ | the inner product of two vectors |
| $\mathcal{M}$ | Marcov decision process |
| $[\mathcal{N}]$ | agents with number of n |
| $\mathcal{S}$ | state space |
| $\{\mathcal{O}^i\}_{i\in\mathcal{N}}$ | observation space |
| $\{\mathcal{A}^i\}_{i\in\mathcal{N}}$ | action space |
| $\mathcal{P}$ | state transformation |
| $\mathcal{R}$ | reward functions |
| $\rho_0$ | intial distribution |
| $\nabla f$ | the gradient vector |
| $\boldsymbol{x}$ | a vector or matrix |
| $x$ | a scalar variable |
| $\dot{\boldsymbol{x}}$ | the first-order derivative of $\boldsymbol{x}$ |
| $\hat{\boldsymbol{x}}$ | the state estimate |
| $x_{i,k}$ | the $i$-th component of a vector $x$ at time $k$ |
| $\bar{x}$ | the vector with the average of all components of $x$ as each element |
| $\mathbf{1}$ | all-ones column vector with proper dimension |
| $\mathcal{N}_i$ | the index set of the neighbors of agent $i$ |
| $p(\boldsymbol{x})\sim\aleph(\boldsymbol{\mu},\boldsymbol{\sigma}^2)$ | a Gaussian distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ |
| $\{O_E,x_E,y_E,z_E\}$ | the Earth inertial frame |

| | |
|---|---|
| $\{O_B, x_B, y_B, z_B\}$ | the body frame |
| $\boldsymbol{p}_E = [x, y, z]^T$ | the position of quadrotor |
| $\boldsymbol{v}_E = [v_x, v_y, v_z]^T$ | the translational velocity of quadrotor |
| $\boldsymbol{q}_{EB} = [q_0, q_1, q_2, q_3]^T$ | a unit quaternion of quadrotor |
| $\boldsymbol{\omega}_B = [\omega_x, \omega_y, \omega_z]^T$ | the body angular velocity of quadrotor |
| $\boldsymbol{R}_B^E$ | the transform matrix from $O_B$ to $O_E$ |
| $\boldsymbol{f} = [0, 0, f]^T$ | the mass-normalized thrust vector |
| $\boldsymbol{g} = [0, 0, -g_z]^T$ | the gravity acceleration on Earth |
| $\boldsymbol{\Omega}(\boldsymbol{\omega}_B)$ | the skew-symmetric matrix |
| $f_{min}$ | the minimum normalized thrust |
| $f_{max}$ | The maximum normalized thrust |
| $\omega_{max}$ | the maximum angular velocity |
| $\boldsymbol{x} \in \mathbb{R}^n$ | the state vector |
| $\boldsymbol{u} \in \mathbb{R}^p$ | the control commands |
| $\boldsymbol{y} \in \mathbb{R}^m$ | the measurement vector |
| $\boldsymbol{w}$ | the process noise |
| $\boldsymbol{v}$ | the measurement noise |
| $\boldsymbol{f}(\cdot)$, $\boldsymbol{h}(\cdot)$ | the state function and measurement function |
| $k$ | the time steps |
| $T_s$ | the sampling interval |
| $t_k = kT_s$ | the discrete time at time step $k$ |
| $\hat{\boldsymbol{x}}_{k\|k-1}$ | the priori state estimate |
| $\boldsymbol{P}$ | the estimate covariance matrix |
| $\boldsymbol{K}$ | the Kalman gain |
| $a_x$, $a_y$, $a_z$ | the measured acceleration |
| $\psi_m$ | the azimuth angle |
| $\boldsymbol{a}_k \in \mathbb{R}^p$ | the attack vector |
| $\boldsymbol{y}'_k$, $\boldsymbol{x}'_k$ | the measurements and states under attacks |
| $\boldsymbol{\tau}$ | the attack selection matrix |
| $\boldsymbol{\Delta x}_k$ | the state estimation error at time $t_k$ |
| $\boldsymbol{z}_k$ | the innovation at time $t_k$ |
| $\boldsymbol{\beta}_k$ | the covariance matrix of $\boldsymbol{z}_k$ |
| $\boldsymbol{Q}$ | the process noise covariance matrix |
| $\boldsymbol{R}$ | the measurement noise covariance matrix |
| $1 - \alpha$ | the significance level |

| | |
|---|---|
| $\lambda$ | the relaxation factor in the MSIS detector |
| $n$ | the length of the sliding window |
| $\bar{\epsilon}$ | the calibration iteration error threshold |
| $\bar{N}$ | the maximum calibration iteration number |
| $k_p, k_i, k_d$ | the PID gains |
| $t_{total}$ | the total simulation time |
| $\{t_a\}$ | the time period of cyber attacks |

# Acronyms

| | |
|---|---|
| AC | Actor Critic |
| ACEMSL | Adaptive Curriculum Embedded Multistage Learning |
| ACL | Adaptive Curriculum Learning |
| AEC | Adaptive Embedded Curriculum |
| AEO | Attention Enhanced Observation |
| AL | Asynchronous Learning |
| AMS-DRL | Asynchronous Multistage Deep Reinforcement Learning |
| APF | Artificial Potential Fields |
| ASR | Average Success Rate |
| CMTSN | Collaborative Multi-target Search and Navigation |
| CNN | Convolutional Neural Network |
| CPS | Cyber-Physical System |
| CSL | Cold-Start Learning |
| CTS | Collaborative Target Search |
| CTDE | Centralized Training with Decentralized Execution |
| CTE | Centralized Training and Execution |
| DRL | Deep Reinforcement Learning |
| DTE | Decentralized Training and Execution |
| DoS | Denial-of-Service |
| EDA | Exploratory Data Analysis |
| EKF | Extended Kalman Filter |
| FDI | False Data Injection |
| FOV | Field of View |
| FSP | Fictitious Self-Play |

| | |
|---|---|
| GA | Genetic Algorithm |
| IoU | Intersection Over Union |
| LSTM | Long Short Term Memory |
| MADRL | Multiagent Deep Reinforcement Learning |
| MAFMDP | Multiagent Finite Markov Decision Process |
| MARL | Multiagent Reinforcement Learning |
| MAS | Multiagent Systems |
| MCA | Mixed Credit Assignment |
| MDP | Markov Decision Process |
| mAP | Mean Average Precision |
| MSIS | Modified Sliding Innovation Sequences |
| MPETN | Multi-pursuit Evasion with Targeted Navigation |
| MOT | Multi-Object Tracking |
| NE | Nash Equilibrium |
| OOI | Object of Interest |
| PEG | Pursuit-Evasion Game |
| PETN | Pursuit-Evasion with Targeted Navigation |
| POMA | Posthumous Mix-credit Assignment with Attention |
| POMDP | Partially Observable Markov Decision Process |
| RL | Reinforcement Learning |
| ROS | Robotic Operating System |
| SR | Success Rate |
| TDL | Task Difficulty Level |
| TTR | Time to Reach |
| UAV | Unmanned Aerial Vehicles |
| VIO | Visual-Inertial Odometry |

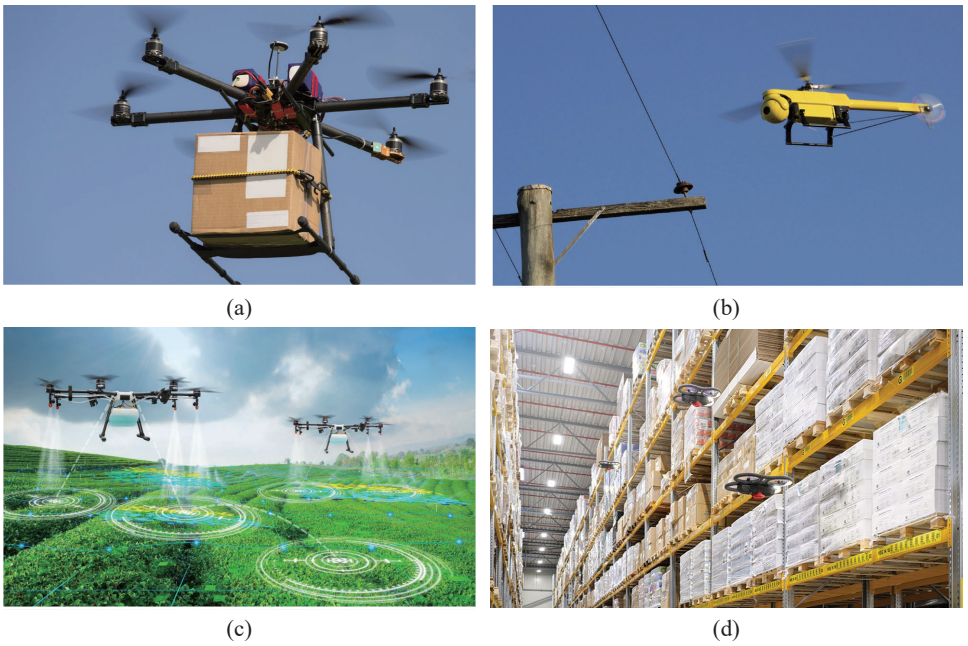
# Chapter 1

# Introduction

## 1.1 Background

Drones are intelligent cyber-physical systems (CPS) [34, 35] that rely on functional sensing, system communication, real-time computation, and flight control to achieve perception and autonomous flight. Due to their high autonomy and maneuverability, drones [36] have been widely used in various missions, such as parcel delivery [37, 38], industrial inspection [39], precision agriculture [40], and search-and-rescue [41] (some applications are shown in Fig. 1.1). As such, drones in smart cities are an essential feature of the service industry of tomorrow. With the advent of novel materials (adhesive and flexible materials), miniaturization of components (sensors, onboard computers), and advanced localization systems (SLAM, UWB, GPS, etc.), drones are evolving into more complex and intelligent systems. This evolution is directed towards minimizing drone size for cost-effective and spatially unrestrictive missions, innovating in design for superior maneuverability and flight performance, and enhancing autonomy through advanced visual perception and decision-making algorithms.

Vision-based learning, combining advanced sensing with learning capabilities, has emerged as a pivotal technology for drones, pushing them towards artificial general intelligence (AGI) in the three-dimensional (3D) physical world, especially when integrated with rapid-growing large language models (LLMs) [42] and embodied intelligence [43]. This integration promises enhanced autonomy and intelligence, enabling drones to navigate and perform tasks in complex environments. A typical



(a) (b) (c) (d)

FIGURE 1.1: Applications of vision-based drones. (a) Parcel delivery [1]; (b) Power grid inspection [2]; (c) Precision agriculture [3]; (d) Intelligent warehouse management [4].

vision-based drone consists of three parts (see Fig. 1.2): (1) **Visual perception**: sensing the environment around the drone via monocular cameras or stereo cameras; (2) **Image processing**: extracting features from an observed image sequence and output specific patterns or information, such as navigation information, depth information, object information; (3) **Flight controller**: generating high-level and low-level commands for drones to perform assigned missions. Image processing and flight controllers are generally conducted on the onboard computer, while visual perception relies on the performance of visual sensors. Vision-based drones have been widely used in traditional missions such as environmental exploration [44], navigation [45], and obstacle avoidance [5, 46]. With efficient image processing and simple path planners, they can avoid dynamic obstacles effectively. Currently, vision-based learning drones, which utilize visual sensors and efficient learning algorithms, have achieved remarkable advanced performance in a series of standardized visual perception and decision-making tasks, such as agile flight control [16], navigation [47] and obstacle avoidance [5]. Various cases have showcased the power of learning algorithms in improving the agility and perception capabilities of vision-based drones. For instance, using only depth cameras, inertial measurement units (IMU) and a lightweight onboard computer, the vision-based learning drone in [16]

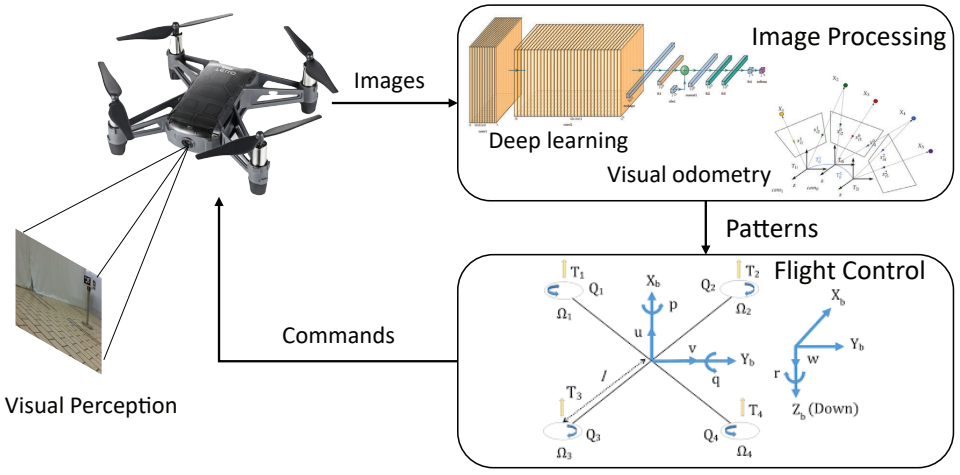



FIGURE 1.2: General framework of vision-based drones

succeeded in performing high-speed flight in unseen and unstructured wild environments. The controller of the drone in [16] was trained from a high-fidelity simulation and transferred to a physical platform. Following that, the Swift system was developed in [25] to achieve world champion-level autonomous drone racing with a tracking camera and a shallow neural network. Such kinds of vision-based learning drones are leading the future of drones due to their perception and learning capabilities in complex environments. Similarly, with event cameras and a trained narrow neural network-EVDodgeNet, the vision-based learning drone in [48] was able to dodge multiple dynamic obstacles (balls) during flight. Afterwards, to improve the perception onboard in the real world, an uncertainty estimation module was trained with the Ajna network [49], which significantly increased the generalization capability of learning-based control for drones. The power of deep learning in handling uncertain information frees traditional approaches from complex computation with the necessary accurate modeling.

While vision-based learning has significantly advanced the capabilities of single-drone systems, its potential for multi-drone systems remains largely underexplored. These systems are pivotal for applications such as coordinated surveying [26, 50–52], cooperative tracking [27, 53, 54], synchronized monitoring [28, 55, 56], and disaster response [23, 29, 57, 58]. They introduce complexities beyond the scope of most current path planning methods for visual drone swarms [59, 60], which have primarily focused on simple formation maintenance and maneuvers, often overlooking the essence of efficient inter-drone collaboration. These complexities include the necessity for effective inter-drone communication, seamless coordination, and

the real-time integration of measurements from multiple sources—critical elements for collaborative environments where drones must work together to achieve collective objectives. Conversely, the operational environment for drones is fraught with adversarial challenges. The presence of physical and cyber threats [61] adds layers of complexity that can significantly jeopardize the safety and success of drone operations, potentially leading to substantial economic losses and endangering lives. This challenge becomes even more daunting when considering physical attacks from intelligent adversaries such as birds or enemy drones [62], demanding a higher level of situational awareness and rapid response far beyond basic obstacle avoidance capabilities. For instance, bird attacks due to territorial behavior during nesting seasons are a well-documented threat to drones, with such a phenomenon curtailing Google's plans for last-mile drone delivery services [63]. In addition, these tasks are essentially required to be accomplished in the real-time physical world, which brings other computation challenges to be addressed, such as algorithm complexity and communication latency requirements.

This thesis endeavors to address these challenges by exploring the integration of vision-based learning in drones for optimized performance in both collaborative and adversarial environments. Particularly, this thesis aims to address the sample efficiency issue in sparse reward space, team credit assignment in collaboration, open-ended evolution in competition, as well as real-world flights with learning policies. The goal is to develop methodologies and frameworks that enhance the cooperative abilities of multi-drone systems while simultaneously bolstering their defenses against the diverse threats inherent in adversarial environments, including physical and cyber attacks.

## 1.2 Motivation and objectives

The motivation behind this thesis stems from the aforementioned challenges and contributes to advancing state-of-the-art vision-based learning for multi-drone systems, particularly within collaborative and adversarial environments. In this thesis, real-time operation and physical flight are essentially focused on, introducing computational challenges such as managing algorithm complexity and overcoming communication latency. The incorporation of onboard computers for perception, communication, and control not only enhances drone capabilities but also exposes

them to potential cyber threats affecting communication links, sensor networks, and actuator channels. This situation highlights the urgent need for a comprehensive approach to cyber attack detection and isolation, specifically tailored for real-time multi-drone applications. The research problems aimed to be solved in this thesis are summarized as follows.

**Research Question 1: How to efficiently learn effective collaborations for a visual drone swarm?** Visual drone swarms have shown significant advantages in performing complex and large-scale missions compared to single visual drone systems. These advantages include a wider sensing range and parallel execution capabilities. However, existing strategies [59, 60, 64, 65] for drone swarms were rarely considered effective collaborations, where each drone was only optimized to maintain a desired formation or generate optimal trajectories. To achieve better collaboration, each drone is required to be rewarded based on its contribution to the team. This requirement can be well met by adopting multiagent reinforcement learning, which generates optimal policies by maximizing group return. However, there are several challenges to learning effective collaborations for a visual drone swarm, namely, sample efficiency in sparse reward spaces, optimal credit assignment among agents, and variable observations from dynamic environments.

**Research Question 2: How to optimally learn rapid response for visual drones with multiple adversaries?** Visual drones are fundamentally required to be equipped with the capability to detect, maneuver, and evade intelligent adversaries such as birds and enemy drones to improve flight reliability and safety in complex missions. Most traditional obstacle avoidance methods considered static obstacle avoidance [5, 45, 66, 67] or simple dynamic obstacles like throwing balls [5, 48], where the position of the obstacles would not change or the paths of the moving obstacles could be easily predicted in flight.

**Research Question 3: How to generally handle cyber attacks for drones during rapid flight?** To maintain secured drone operations, it is essential to detect and isolate cyber attacks within a short period of time before losing control. Learning-based detection and model-based detection are the two main types of cyber attack detection methods that can be used for drones. Existing learning-based detectors [68, 69] require time-consuming training processes and massive real-life flight datasets, which are physically difficult to obtain for drone swarms. Furthermore, these methods are usually platform-centric and hard to generalize

for multiple platforms. On the other hand, most of the current model-based cyber attack detectors [70, 71] can only cater to a specific type of attack. These methods are based on linear systems under equilibrium conditions and can hardly be used for a drone in six degrees of freedom (6DOF), and hence are not applicable for drones in agile flight.

To address the research questions outlined, the objectives of this thesis are as follows:

- Enhance the sample efficiency and collective performance of vision-based learning for collaborative multi-drone systems;
- Equip drones with real-time onboard visual perception capabilities to detect and track adversaries in physical environments;
- Develop a learning-based control approach enabling drones to swiftly outmaneuver multiple adversaries;
- Implement a system on drones for the real-time detection and isolation of various cyber attacks, including random data attacks, fault data injection (FDI) attacks, and denial of service (DoS) attacks, particularly during rapid flight;
- Validate the proposed solutions through comprehensive simulations and practical experiments, demonstrating their effectiveness in real-world scenarios.

## 1.3 Major contributions

This thesis contributes to providing several state-of-the-art solutions to improving mission efficiency for drones in collaborative environments and enhancing the safety and security of drones in adversarial environments. Our main contributions are summarized as follows:

**Contribution 1:** *A sample-efficient adaptive curriculum embedded multistage reinforcement learning approach (ACEMSL) for collaborative target search.* Compared to the standard curriculum learning methods [72–74] where the learning curriculum involves pre-designed parameters that change across the training process,

our approach divides the training process into two adaptive curriculum embedded stages. This can reduce the amount of data required to achieve the best performance due to the self-adjustment capability of the curriculum. In each stage, the probability of hiding the target from a drone swarm increases when the desired success rate is achieved. As such, the drone agents are gradually guided to explore the environment with sparse rewards and try to search for the potential target. To the best of our knowledge, our work is the first to learn collaborative target search for a visual drone swarm in the real world. More details are elaborated in Chapter 3.

**Contribution 2:** *A posthumous mix-credit assignment with attention reinforcement learning framework (POMA) for collaborative multitarget search.* Our POMA first combines adaptive curriculum learning and mixed individual-group posthumous credit assignment based on posthumous credit assignment in MA-POCA [75], and then introduces the attention mechanism into observations to improve the scalability of the framework when variable local observations are involved in dynamic environments. We further validated that our framework outperforms baselines through extensive simulation and real-world flight experiments. More details are elaborated in Chapter 4.

**Contribution 3:** *A real-time multi-drone detection and tracking system (Drone-YOLOSORT) for pursuit-evasion with parameter search.* To detect and track multiple adversaries, we develop a whole life cycle multi-drone detection and tracking approach for pursuit-evasion operations, incorporating parameter search and edge acceleration techniques. We first collect and develop a new "ICG-Drone" dataset from various environments, including indoors, outdoors, and internet sources, filling a gap in existing datasets [76, 77] tailored for multi-drone detection and tracking. We further benchmark multi-drone detection performance with various models, considering mAP, GPU occupancy, and inference speed. The developed Drone-YOLOSORT system can be seamlessly integrated into the robot operation system with a real-time inference frequency of 24.3 FPS. More details are elaborated in Chapter 5.

**Contribution 4:** *A asynchronous multistage deep reinforcement learning approach (AMS-DRL) for multi-pursuit evasion.* Motivated by nature-inspired pursuit-evasion phenomena where pursuers and chasers keep learning from each other with steady policy evolution,the AMS-DRL is designed to evolve adversarial agents from

multistage DRL, where the chaser drones and the runner drone are asynchronously trained in a bipartite graph way during different stages. Hence, the AMS-DRL is an unsupervised learning algorithm toward open-ended evolution [78], which approximates the capability limits of agents consistently until the adversarial agents reach a Nash equilibrium (NE) [79]. We further demonstrate that the resulting policies can provide drones with rapid evasive maneuvers for self-preservation when facing multiple adversaries on real-world flights. More details are elaborated in Chapter 6.

**Contribution 5:** *A general state estimation-based method with modified sliding innovation sequences (MSIS) for cyber attack detection and isolation.* A novel and effective state estimation-based detection and isolation method applicable to any nonlinear aerial platform is proposed. Unlike previous work in [70, 80, 81], which have been limited to implementations on simplified linear models involving translational motions only, to the best of the authors' knowledge, this is the first time that a full-state nonlinear hybrid dynamics model of a dynamic quadrotor, including translational and rotational states, is established in the detection and isolation problem with different cyber attacks on sensors and actuators. Not limited to hovering or low-speed translational motion scenarios, the detection strategy is verified by means of a complex periodic trajectory tracking flight of a quadrotor involving high-speed nonlinear dynamics. More details are elaborated in Chapter 7.

## 1.4 Thesis outline

Chapter 1 introduces the background of drones and provides an overview of vision-based drones, as well as the current state of vision-based learning for drones. The motivation and major contributions of this thesis are highlighted.

Chapter 2 summarizes the prerequisite knowledge of visual-based learning, including drone dynamics, visual perception, machine learning, reinforcement learning, and object detection and tracking. Then state-of-the-art methods in vision-based control for drones are analyzed and categorized. Meanwhile, applications of vision-based learning for drones are discussed, from single-agent systems to multiagent systems and heterogeneous systems. Several challenges are analyzed, which highlights the necessity of work in this thesis.

Chapter 3 proposes a sample-efficient adaptive curriculum embedded multistage learning (ACEMSL) approach for collaborative target search with a visual drone swarm. The drone swarm is trained to find and approach a designated target, which showcases the advantages of DRL-controlled visual drone swarms in collaborative environments. The Sim2Real transfer and actual flight tests verified the effectiveness of vision-based learning control for drones.

Chapter 4 presents a salable posthumous mix-credit assignment with attention (POMA) multiagent reinforcement learning framework for collaborative multitarget search. The visual drone swarm is trained to collaboratively explore the cluttered environment and find multiple targets. The posthumous mix-credit assignment and attention mechanism are introduced to address individual-group credit assignment and variable observation issues when multiple targets are involved. The superior performance of POMA is demonstrated in missions involving a variety of drones and targets with simulations and real-world flights.

Chapter 5 develops a whole life cycle real-time multi-drone detection and tracking system (Drone-YOLOSORT) for visual drones operating in adversarial environments. A novel multiclass drone dataset "ICG-Drone" is collected, and a fine-tuned drone detector is compiled into the ROS package, providing seamless integration into navigation methods in adversarial environments.

Chapter 6 offers an asynchronous multistage deep reinforcement learning approach (AMS-DRL) for multi-pursuit evasion, which is an unsupervised learning framework towards open-ended evolution for learning rapid response in adversarial environments. The effectiveness of trained policies is analyzed in various scenarios of different space placement, relative speed limitation and number of adversaries.

Chapter 7 develops a cyber attack detection and isolation method for a dynamic drone using modified sliding innovation sequences (MSIS), where three data-driven cyber attacks are studied, including random attack, data injection attack, and DoS attack. By evaluating the sample covariance of the normalized innovation sequence, the MSIS detector has the capability to isolate cyber attacks. Simulation results of a quadrotor drone in a periodic, complex trajectory flight are provided to verify the effectiveness of the MSIS detection and isolation method.

Chapter 8 summarizes the main findings of this thesis and provides open questions for future research in the domain of vision-based learning for drones with higher autonomy and operation safety.

# Chapter 2

# Related works

This chapter reviews the current state of vision-based learning for drones, highlighting its classifications and applications. The aim is to explore the visual perception and learning intricacies in collaborative and adversarial environments, setting the understanding for addressing these challenges within this thesis.

## 2.1 Background

### 2.1.1 Quadrotor drone dynamics

The translational and rotational dynamics of a 6 degree-of-freedom (6DOF) quadrotor of mass $m$ can be described as follows, regardless of wind disturbance and aerodynamic drag:

$$\dot{\boldsymbol{p}}_W = \boldsymbol{v}_W \tag{2.1a}$$

$$\dot{\boldsymbol{v}}_W = \boldsymbol{R}_B^W(\boldsymbol{q}_{WB})\boldsymbol{f} + \boldsymbol{g} \tag{2.1b}$$

$$\dot{\boldsymbol{q}}_{WB} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_B)\boldsymbol{q}_{WB} \tag{2.1c}$$

where, $\boldsymbol{p}_W = [x, y, z]^T$ and $\boldsymbol{v}_W = [v_x, v_y, v_z]^T$ are the position and translational velocity vectors of the quadrotor in the world inertial frame $O_W$, where the $z$ axis of the world frame is aligned with the direction of the gravity $\boldsymbol{g}$. $\boldsymbol{q}_{WB} = [q_0, q_1, q_2, q_3]^T$ denotes a unit quaternion which is used to represent the quadrotor's attitude, while $\boldsymbol{\omega}_B = [\omega_x, \omega_y, \omega_z]^T$ denotes the body angular velocity (roll, pitch, and yaw,

respectively) in the body frame $O_B$. $\boldsymbol{R}_B^W$ defines the transform matrix from the body frame $O_B$ to the world inertial frame $O_W$, and it is a function of $\boldsymbol{q}_{WB}$. $\boldsymbol{f} = [0, 0, f]^T$ is the mass-normalized thrust vector with $f = \sum_{i=1}^{4} f_i^B/m$ (where $f_i^B$ are the trust from four propellers) and $\boldsymbol{g} = [0, 0, -g_z]^T$ with $g_z = 9.81m/s^2$ being the gravitational acceleration on Earth. We denote $\boldsymbol{\Omega}(\boldsymbol{\omega}_B)$ as the skew-symmetric matrix given by

$$\boldsymbol{\Omega}(\boldsymbol{\omega}_B) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{2.2}$$

$\boldsymbol{x} = [\boldsymbol{p}_W^T, \boldsymbol{v}_W^T, \boldsymbol{q}_{WB}^T]^T$ and $\boldsymbol{u} = [f, \omega_x, \omega_y, \omega_z]^T$ are selected as the states and the control inputs of a quadrotor, respectively. The inputs are constrained within limited ranges, i.e.,

$$f_{min} \leq f \leq f_{max} \quad \omega_{min} \leq ||\boldsymbol{\omega}_B|| \leq \omega_{max}. \tag{2.3}$$

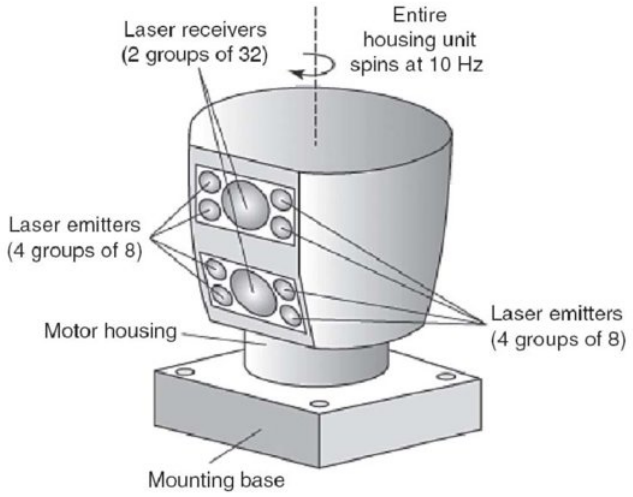
The body's momentum is not examined because the deployed high-bandwidth controller can precisely track the angular velocity commands, allowing angular dynamics to be ignored. Generally, the full 6DOF dynamics model is used to simulate the motion of the drone while a simplified 3DOF dynamic model is considered in the policy generation to align with the high-level velocity control.

### 2.1.2 Visual perception

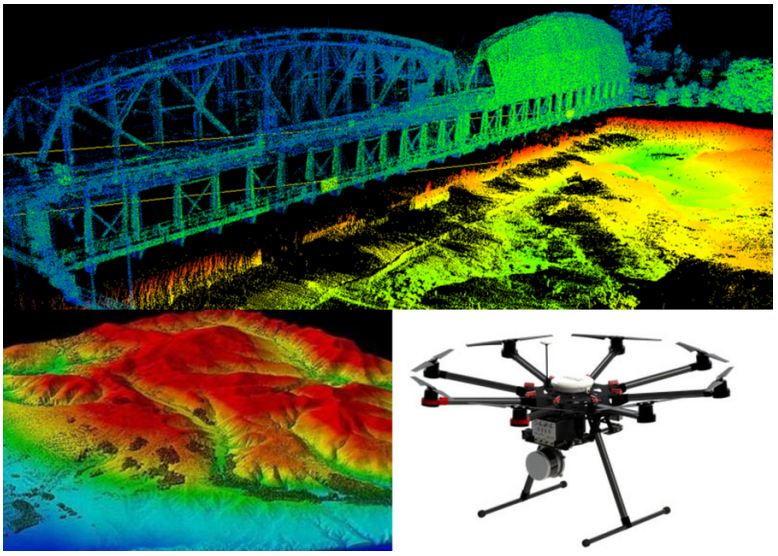
Visual perception for drones is the ability of drones to perceive their surroundings and their own states through the extraction of necessary features for specific tasks with visual sensors. Light detection and ranging (lidar) and cameras are commonly used sensors to perceive the surrounding environment for drones.

#### 2.1.2.1 Lidar

Lidar is a kind of active range sensor that relies on the calculated time of flight (TOF) between the transmitted and received beams (laser) to estimate the distance between the robot and the reflected surface of objects [82]. Based on the



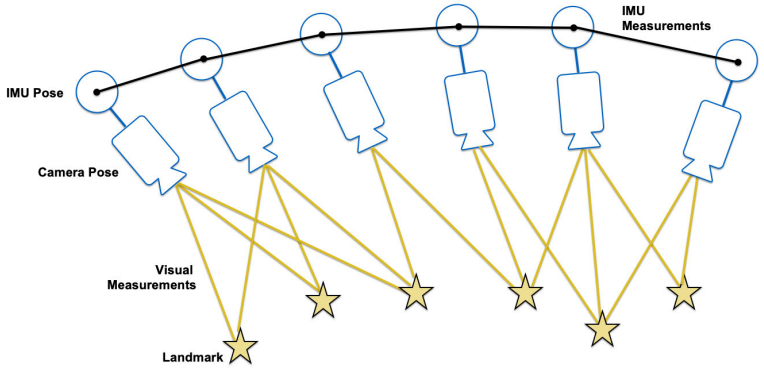

(a)



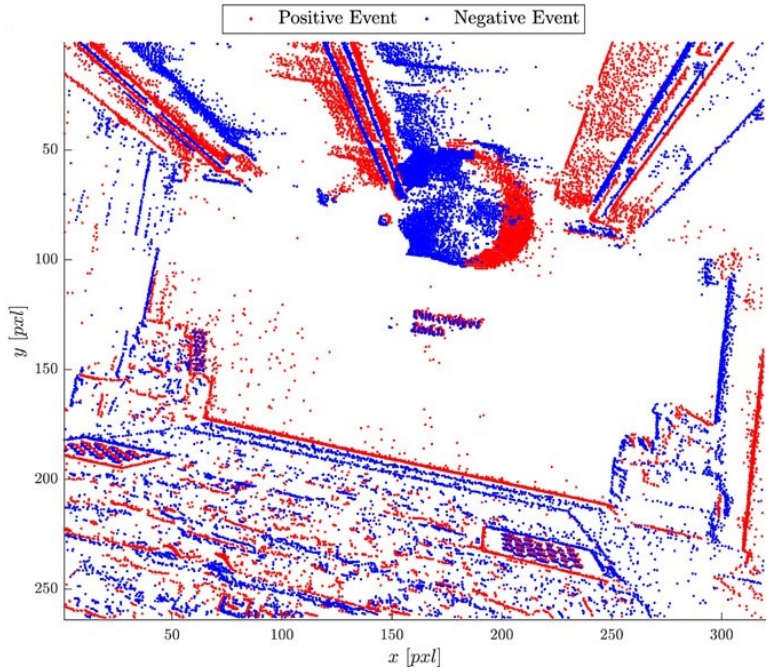
(b)

FIGURE 2.1: Lidar on drones for visual perception. (a) A typical surrounding lidar; (b) Generated point cloud with lidar.

scanning mechanism, lidar can be divided into solid-state lidar, which has a fixed field of view without moving parts, and surrounding lidar, which spins to provide a 360-degree horizontal view. Surrounding lidar is also referred to as "laser scanning" or "3D scanning", which creates a 3D representation of the explored environment using eye-safe laser beams. A typical lidar (see Fig. 2.1) consists of laser emitters, laser receivers, and a spinning motor. The vertical field of view (FOV) of a lidar is determined by the number of vertical arrays of lasers. For instance, a vertical array of 16 lasers scanning 30 degrees gives a vertical resolution of 2 degrees in a




(a)




(b)

FIGURE 2.2: Visual perception with cameras for drones. (a) Visual Odometry for drones' positioning; (b) Object detection for drones' obstacle avoidance with event camera [5].

typical configuration. Lidar has recently been used on drones for mapping [83], power grid inspection [84], pose estimation [85] and object detection [86]. Lidar provides sufficient and accurate depth information for drones to navigate in cluttered environments. However, it is bulky and power-hungry and does not fit within the payload restrictions of agile autonomous drones. Meanwhile, using raycast representation in a simulation environment makes it hard to match the inputs of a real lidar device, which brings many challenges for the Sim2Real transfer when a learning approach is considered.

#### 2.1.2.2 Camera

Compared to lidar, cameras provide a cheap and lightweight way for drones to perceive the environment. Cameras are external passive sensors used to monitor the drone's geometric and dynamic relationship to its task, environment, or the objects that it is handling. Cameras are commonly used as perception sensors for drones to sense environment information, such as objects' positions and a point cloud map of the environment (see Fig. 2.2). In contrast to the motion capture system, which can only broadcast global geometric and dynamic pose information within a limited space from an offboard synchronized system, cameras enable a drone to fly without space constraints. Cameras can provide positioning for drone navigation in GPS-denied environments via visual odometry (VO) [87–89] and visual simultaneous localization and mapping systems (V-SLAM) [90–92]. Meanwhile, object detection and depth estimation can be performed with cameras to obtain the relative positions and sizes of obstacles. However, to avoid dynamic obstacles, even physical attacks like bird chasing, the agile vision-based drone poses fundamental challenges to visual perception. Motion blur, sparse texture environments, and unbalanced lighting conditions can cause the loss of feature detection in the VIO and object detection. Lidar and event cameras [93] can partially address these challenges. However, lidar and event cameras are either too bulky or too expensive for agile drone applications. Considering the agility requirement of physical attack avoidance, lightweight dual-fisheye cameras are used for visual perception. With dual fisheye cameras, the drone can achieve better navigation capability [94] and omnidirectional visual perception [95]. Some sensor fusion and state estimation techniques are required to alleviate the accuracy loss brought by the motion blur.

### 2.1.3 Machine learning

Recently, machine learning (ML), especially deep learning (DL), has attracted much attention from various fields and has been widely applied to robotics for environmental exploration [96, 97], navigation in unknown environments [17, 18, 98], obstacle avoidance, and intelligent control [99]. In the domain of drones, learning-based methods have also achieved promising success, particularly for deep reinforcement learning (DRL) [16, 25, 73, 100–102]. In [73], a curriculum learning augmented end-to-end reinforcement learning was proposed for a UAV to fly through

a narrow gap in the real world. A vision-based end-to-end learning method was successfully developed in [16] to fly agile quadrotors through complex wild and human-made environments with only onboard sensing and computation capabilities, such as depth information. A visual drone swarm was developed in [103] to perform collaborative target search with adaptive curriculum embedded multistage learning. These works verified the marvelous power of learning-based methods on drone applications, which pushes the agility and cooperation of drones to a level that classical approaches can hardly reach. Different from the classical approaches relying on separate mapping, localization, and planning, learning-based methods map the observations, such as the visual information or localization of obstacles, to commands directly without further planning. This greatly helps drones handle uncertain information in operations. However, learning-based methods require massive experiences and training datasets to obtain good generalization capability, which poses another challenge in deployment over unknown environments.

### 2.1.4 Reinforcement learning

RL [104] is a technique for mapping state (observation) space into the action space in order to maximize a long-term return with given rewards. The learner is not explicitly told what action to carry out, but must figure out which action will yield the highest reward during the exploring process. A typical RL model (see Fig. 2.3) features agents, the environment, reward functions, action, and state space. The policy model achieves convergent status via constant interactions between the agents and the environment, where the reward function guides the training process.

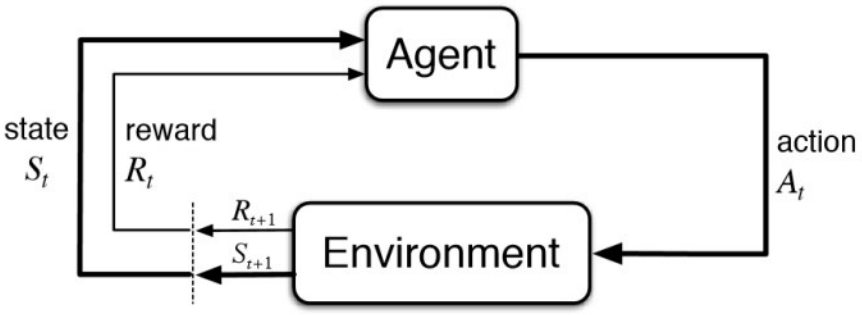



FIGURE 2.3: The reinforcement learning process.

#### 2.1.4.1 Single-agent RL

The RL problem can be modeled with a Markov Decision Process (MDP) [104], which is defined by a tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ denotes a set of possible states of agents. At time step $t$ and current state $s_t \in \mathcal{S}$, the agent obtains an observation $o_t \in \mathcal{O} \subseteq \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ guided by a stochastic policy $\pi(a_t \mid o_t) : \mathcal{O} \times \mathcal{A} \mapsto [0, 1]$. $\mathcal{P}(s_{t+1} \mid s_t, a_t) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition probability of the environment moving from the current state $s_t$ to the next state $s_{t+1}$ after taking an action $a_t$. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ refers to the reward function evaluating the bounded instantaneous reward $r_t := \mathcal{R}(s_t, a_t) \in [r_{\min}, r_{\max}]$ for agent taking action $a_t$ under state $s_t$. The agent starts from the state $s_0$ sampled from an initial state distribution $s_0 \sim d_0 : \mathcal{S} \mapsto [0, 1]$. Given a policy $\pi$ and a state $s_t$, the state value function of a finite MDP $V : \mathcal{S} \mapsto \mathcal{R}$ is defined as:

$$V_\pi(s_t) := \mathbb{E}_\pi \left[ \sum_{k=0}^{k=T-t} \gamma^k r_{t+k} \mid s_t \right] \tag{2.4}$$

where $\gamma \in [0, 1)$ is the discount factor and $T$ is the final step of an episode. Similarly, considering the value of taking an action $a_t$, the state-action value function (Q-function) $Q : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{R}$ can be calculated as a Bellman function:

$$\begin{aligned} Q_\pi(s_t, a_t) &:= \mathbb{E}_\pi \left[ \sum_{k=0}^{k=T-t} \gamma^k r_{t+k} \mid s_t, a_t \right] && \text{(2.5a)} \\ &= \mathbb{E}_{s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)} \left[ r_t + \gamma V_\pi(s_{t+1}) \right] && \text{(2.5b)} \\ &= \mathbb{E}_{s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)} \left[ r_t + \gamma \, \mathbb{E}_{a_{t+1} \sim \pi} \left[ Q_\pi(s_{t+1}, a_{t+1}) \right] \right] && \text{(2.5c)} \end{aligned}$$

Currently, three main types of RL methods are adopted, namely value-based method, policy-based method [105], and actor-critic (AC) framework method [106]. Different from the value-based methods which calculate the value function to guide the policy, the policy-based method aims to directly find an optimal policy $\pi_\theta^* : \mathcal{O} \mapsto \mathcal{A}$ to maximize the cumulative reward along a state-action trajectory $\tau := \{s_0, a_0, s_1, a_1, ..., s_T, a_T\}$ by adjusting the weights $\theta$ of the parameterized policy $\pi_\theta$ over finite time steps (a training episode). The objective function can be formulated as

$$\mathcal{J}(\pi_\theta) = \mathbb{E}_{s_0 \sim d_0, \tau \sim \pi_\theta} \left[ \sum_{t=0}^{t=T} \gamma^t r_t \mid s_0 \right] \tag{2.6}$$

The optimal policy is therefore obtained by maximizing the state value function $V_\pi(s_0)$ starting from $s_0$, i.e.,

$$\pi_\theta^* = \underset{\theta}{\operatorname{argmax}}\, V_{\pi_\theta}(s_0) \tag{2.7}$$

The optimal policies also share the same optimal Q-function, i.e., $\pi_\theta^* = \operatorname{argmax}_\theta Q_{\pi_\theta}(s_0, a_0)$. The optimal Q-function satisfies the Bellman optimality equation:

$$Q_{\pi^*}(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{P}(\cdot\,|s_t,a_t)}\left[r_t + \gamma \max_{a_{t+1}} Q_\pi(s_{t+1}, a_{t+1})\right] \tag{2.8}$$

The AC framework combines the value-based method and the policy-based method with two neural networks. The policy network $\pi_\theta(a_t \mid o_t)$ is the actor network and the Q-function $Q_{\pi_\theta}(s_t, a_t)$ is the critic network. The optimal policy $\pi_\theta^*$ can be obtained from policy iteration along the gradient of the critic network.

#### 2.1.4.2 Multiagent RL

Similarly, the MARL problem can be formulated as a Multi-Agent Finite Markov Decision Process (MAFMDP) [104]. Similar to the single-agent RL, the MAFMDP is defined with a joint tuple $\langle \mathcal{S}, \{\mathcal{O}^n\}, \{\mathcal{A}^n\}, \mathcal{G}, \mathcal{P}, \{\mathcal{R}^n\}, N, \gamma\rangle$ with $N$ agents denoted by $n \in [N]$. Since there is no knowledge of the target in our work, goal signals $\mathcal{G}$ can be ignored in the following description. Each agent $n$ obtains a local observation $o_t^n := o^n(s_t) \in \mathcal{O}^n$ from the global state $s_t \in \mathcal{S}$, and takes an action $a_t^n \in \mathcal{A}^n$. Let $\mathcal{A} := \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$, the transition probability $\mathcal{P}(s_{t+1} \mid s_t, \boldsymbol{a}_t) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ can be formulated with joint action $\boldsymbol{a}_t := \{a_t^1, ..., a_t^N\}$. For each agent $n$, the joint action can also be described as $\boldsymbol{a}_t := \{a_t^n, a_t^{-n}\}$ where $a_t^{-n}$ denotes all other agents'(except $n$) actions. Each agent will receive an instantaneous reward $\mathcal{R}_t^n := \mathcal{R}(s_t, a_t^n)$ at each time step. Note that the instantaneous rewards include the team reward based on multiagent credit assignment [108]. A joint stochastic policy is denoted as $\boldsymbol{\pi}(\boldsymbol{a}_t \mid s_t) := \prod_{n=1}^N \pi^n(a_t^n \mid o_t^n)$ for all agents transiting from the current state $s_t$ to the next state $s_{t+1}$ with the probability $\mathcal{P}(s_{t+1} \mid s_t, \boldsymbol{a}_t)$ where $\pi^n(a_t^n \mid o_t^n) : \mathcal{O}^n \times \mathcal{A}^n \mapsto [0, 1]$ is the decentralized stochastic policy for the agent $n$. The objective of MARL is to find a joint policy to maximize cumulative reward

along the action trajectory $\tau^n$. The objective function of MARL is:

$$\mathcal{J}(\boldsymbol{\pi_\theta}) := \mathbb{E}_{\tau^n \sim \boldsymbol{\pi}} \left[ \sum_{t=0}^{t=T} \gamma^t \sum_{n=1}^{n=N} \mathcal{R}_t^n \mid s_0 \right] \tag{2.9}$$

The joint optimal policy is obtained by:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \mathcal{J}(\boldsymbol{\pi_\theta}) \tag{2.10}$$

The optimal policies for each agent are achieved along the policy gradient with a centralized critic $Q_{\boldsymbol{\pi}}(s_0, \boldsymbol{a}_0)$ (for fully cooperative teams, where $\mathcal{R}^1 = \cdots = \mathcal{R}^N$) [109] or their critics $Q_{\boldsymbol{\pi}}^n(s_0, \boldsymbol{a}_0) := \mathbb{E}_{\boldsymbol{\pi}} \left[ \sum_{t=0}^{t=T} \gamma^t \mathcal{R}_t^n \mid s_0, \boldsymbol{a}_0 \right]$ with a shared policy conditional on joint action [110].

#### 2.1.4.3 Policy gradient theorem

Within policy-based methods, the state value function $V^{\pi_\theta}(s_t)$ is always approximated by state-action value function $Q^{\pi_\theta}(s_t, a_t)$. In the actor-critic (AC) theory [106], the policy network $\pi_\theta(a_t \mid o_t)$ is the actor network and the state value function approximator $Q^{\pi_\theta}(s_t, a_t)$ is the critic network. The optimal policy $\pi_\theta^*$ can be obtained via policy iteration along the gradient of the critic network (performance objective $\mathcal{L}(\pi_\theta)$) according to the policy gradient theorem [107]. The policy update with learning rate $\alpha_t$ is:

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{s_t \sim \mathcal{P}, a_t \sim \pi_\theta} \left[ \nabla_\theta \log \pi_\theta(a_t \mid o_t) Q_{\pi_\theta}(s_t, a_t) \right] \tag{2.11}$$

$$\theta_{t+1} = \theta_t + \eta_t \nabla_{\theta_t} \mathcal{J}(\pi_{\theta_t}) \tag{2.12}$$

To reduce the variance of the gradient during the training, an advantage function [104] $A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t)$, which measures the performance of an action, is commonly adopted in the stochastic gradient ascent algorithm (2.12). However, an estimator of the advantage function $\hat{A}^{\pi_\theta}(s_t, a_t)$ is considered in most existing algorithms since the accurate value of the advantage function is hard to obtain in real time. Hence, the variance-reduced gradient can be described as follows:

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{s_t \sim \mathcal{P}, a_t \sim \pi_\theta} \left[ \nabla_\theta \log \pi_\theta(a_t \mid o_t) \hat{A}^{\pi_\theta}(s_t, a_t) \right] \tag{2.13}$$

#### 2.1.4.4 Proximal policy optimization

The proximal policy optimization (PPO) algorithm [111] is an efficient and robust actor-critic algorithm that uses a "clipped" surrogate objective function and on-policy learning. In PPO, the clipped surrogate objective function is formulated with *advantage estimates* $\hat{A}_t^{\pi_\theta} := \hat{A}^{\pi_\theta}(s_t, a_t)$ over fixed $T$ timesteps:

$$\mathcal{J}^{clip}(\pi_\theta) = \mathbb{E}_t \left[ \min(\mu_t(\theta)\hat{A}_t^{\pi_\theta}, clip(\mu_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi_\theta})) \right] \tag{2.14}$$

where $\mu_t(\theta)$ denotes the policy probability ratio $\mu_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{old}}(a_t|o_t)}$ and $\epsilon$ is a hyperparameter controlling the clip range. With the policy probability ratio and the clip scheme, the PPO forms a low bound on the performance of the policy $\pi$ and keeps the policy updated without far deviation from the old policy. In addition, the advantage estimates $\hat{A}_t^{\pi_\theta}$ in PPO enable efficient reuse of training data.

## 2.2 Object detection with visual perception

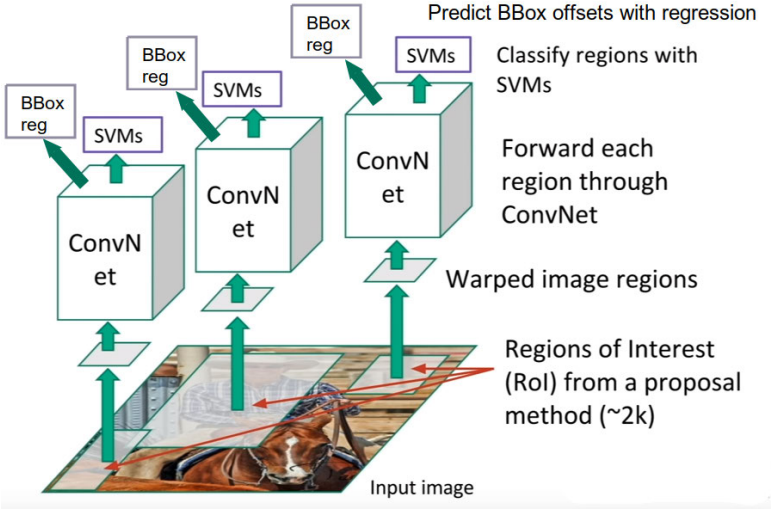
Object detection is a pivotal module in vision-based learning drones when handling complex missions such as inspection, avoidance, and search and rescue. Object detection is to find out all the objects of interest in the image and determine their position and size [112]. Object detection is one of the core problems in the field of computer vision (CV). Nowadays, the applications of object detection include face detection, pedestrian detection, vehicle detection, and terrain detection in remote sensing images. Object detection has always been one of the most challenging problems in the field of CV due to the different appearances, shapes, and poses of various objects, as well as the interference of factors such as illumination and occlusion during imaging. At present, the object detection algorithm can be roughly divided into two categories: **multistage (two-stage) algorithm**, whose idea is to first generate candidate regions and then perform classification, and **one-stage algorithm**, the idea of which is to directly apply the algorithm to the input image and output the categories and corresponding positions. Beyond that, to retrieve 3D positions, depth estimation has been a popular research subbranch related to object detection, whether using monocular [113] or stereo depth estimation [114]. For a very long time, the core neural network module (backbone) of object detection has been the convolutional neural network (CNN) [115]. CNN

is a classic neural network in image processing that originates from the study of the human optic nerve system. The main idea is to convolve the image with the convolution kernel to obtain a series of reorganization features, and these reorganization features represent the important information of the image. As such, CNN not only has the ability to recognize the image but also effectively decreases the requirement for computing resources. Recently, vision transformers (ViTs) [116], originally proposed for image classification tasks, have been extended to the realm of object detection [117]. These models demonstrate superior performance by utilizing the self-attention mechanism, which processes visual information non-locally [118]. However, a major limitation of ViTs is their high computational demand. This presents difficulties in achieving real-time inference, particularly on platforms with limited resources like drones.

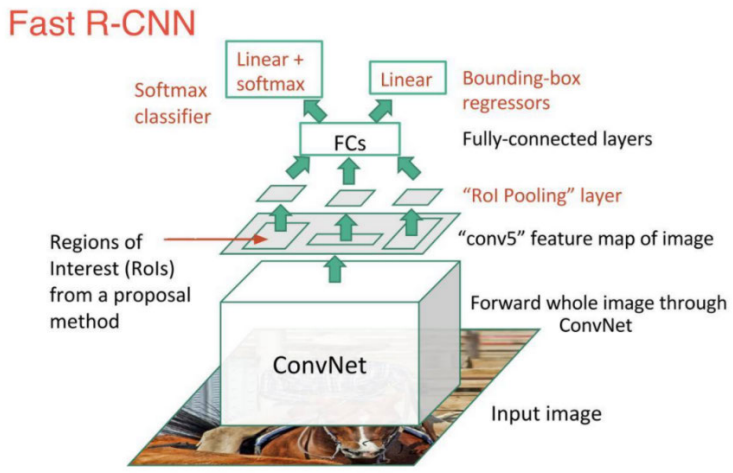
### 2.2.1 Multistage algorithms

Classic multistage algorithms include RCNN (Region-based Convolutional Neural Network) [6], Fast R-CNN [7], Faster R-CNN [8]. Multistage algorithms can basically meet the accuracy requirements in real-life scenarios, but the model is more complex and cannot be really applied to scenarios with high-efficiency requirements. In the R-CNN structure [6], it is necessary to first give some regional proposals (RP), then use the convolutional layer for feature extraction, and then classify the regions according to these features (see Fig. 2.4 (a)). That is, the object detection problem is transformed into an image classification problem. The R-CNN model is very intuitive, but the disadvantage is that it is too slow, and the output is obtained via training multiple Support Vector Machines (SVMs). To solve the problem of slow training speed, the Fast R-CNN model is proposed (see Fig. 2.4 (b)). This model has two improvements to R-CNN: (1) first use the convolutional layer to perform feature selection on the image so that only one convolutional layer can be used to obtain RP; (2) convert training multiple SVMs to use only one fully-connected layer and a softmax layer. These techniques greatly improve the computation speed but still fail to address the efficiency issue of the Selective Search Algorithm (SSA) for RP.

Faster R-CNN is an improvement on the basis of Fast R-CNN (see Fig. 2.5). In order to solve the problem of SSA, the SSA that generates RP in Fast R-CNN is




(a)




(b)

FIGURE 2.4: (a) R-CNN neural network architecture [6]; (b) Fast R-CNN neural network architecture [7].

replaced by a Region Proposal Network (RPN) and uses a model that integrates RP generation, feature extraction, object classification and object box regression. RPN is a fully convolutional network that simultaneously predicts object boundaries at each location. RPN is trained end-to-end to generate high-quality region proposals, which are then detected by Fast R-CNN. At the same time, RPN and Fast R-CNN share convolutional features. Meanwhile, in the feature extraction stage, Faster R-CNN uses a convolutional neural network. The model achieves 73.2% and 70.4% mean Average Precision (mAP) per category on the PASCAL VOC 2007 and 2012 datasets, respectively. Faster R-CNN has been greatly improved in speed than Fast R-CNN, and the accuracy has reached the state-of-the-art (SOTA), and it also fully

developed an end-to-end object detection framework. However, Faster R-CNN still cannot achieve real-time object detection. Besides, after obtaining RP, it requires heavy computation for each RP classification.

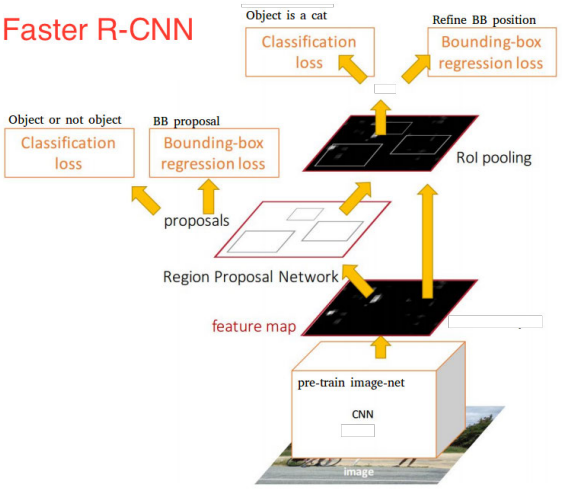



FIGURE 2.5: Faster RCNN neural network architecture [8].

### 2.2.2 One-stage algorithms

One-stage algorithms such as the Single Shot Multibox Detector (SSD) model [10] and the YOLO series models [9] are generally slightly less accurate than the two-stage algorithms, but have simpler architectures, which can facilitate end-to-end training and are more suitable for real-time object detection. The basic process of YOLO (see Fig. 2.6) is divided into three phases, namely, zooming the image, passing the image through a full convolutional neural network, and using maximum value suppression (NMS). The main advantages of the YOLO model are that it is fast, with few background errors via global processing, and it has good generalization performance. Meanwhile, YOLO can formulate the detection task as a unified, end-to-end regression problem, and simultaneously obtain the location and classification by processing the image only once. But there are also some problems with the YOLO, such as rough mesh, which will limit YOLO's performance over small objects. However, the subsequent YOLOv3, YOLOv5, YOLOX [119] and YOLOv8 improved the network on the basis of the original YOLO and achieved better detection results.

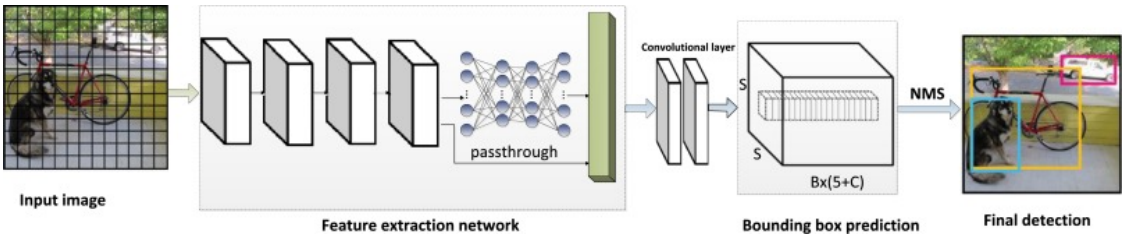



FIGURE 2.6: YOLO network architecture [9].

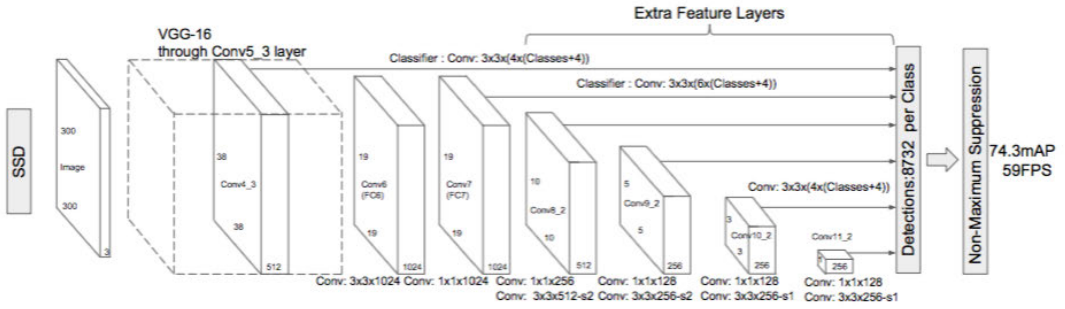



FIGURE 2.7: SSD network architecture [10].

SSD is another classic one-stage object detection algorithm. The flowchart of SSD is (1) first to extract features from the image through a CNN, (2) generate feature maps, (3) extract feature maps of multiple layers, and then (4) generate default boxes at each point of the feature map. Finally, (5) all the generated default boxes are collected and filtered using NMS. The neural network architecture of SSD is shown in Fig. 2.7. SSD uses more convolutional layers than YOLO, which means that SSD has more feature maps. At the same time, SSD uses different convolution segments based on the VGG model to output feature maps to the regressor, which tries to improve the detection accuracy over small objects.

The aforementioned multistage algorithms and one-stage algorithms have their own advantages and disadvantages. Multistage algorithms achieve high detection accuracy, but bring more computing overhead and repeated detection. The one-stage model generally consists of a basic network (Backbone Network) and a Detection Head. The former is used as a feature extractor to give representations of different sizes and abstraction levels of images; the latter one learns classification and location associations based on these representations and a supervised dataset. The two tasks of category prediction and position regression, which are responsible for detecting the head, are often carried out in parallel, formulating a multi-task loss function for joint training. There is only one class prediction and position regression, and most of the weights are shared. Hence, one-stage algorithms are more

time-efficient at the cost of accuracy. In any case, with the continuous development of deep learning in the field of computer vision, object detection algorithms are also constantly learning from and improving each other.

### 2.2.3 Vision transformer

ViTs have emerged as the most active research field in object detection tasks recently, with models like Swin-Transformer [11, 117], ViTdet [120], and DINO [121] leading the forefront. Unlike conventional CNNs, ViTs leverage self-attention mechanisms to process image patches as sequences, offering a more flexible representation of spatial hierarchies. The core mechanism of these models involves dividing an image into a sequence of patches and applying Transformer encoders [122] to capture complex dependencies between them. This process enables ViTs to efficiently learn global context, which is pivotal in understanding comprehensive scene layouts and object relations. For instance, the Swin-Transformer [117] introduces a hierarchical structure with shifted windows, enhancing the model's ability to capture both local and global features. In the following, the Swin-Transformer was scaled to Swin-Transformer V2 [11] with the capability of training high-resolution images (see Fig. 2.8).

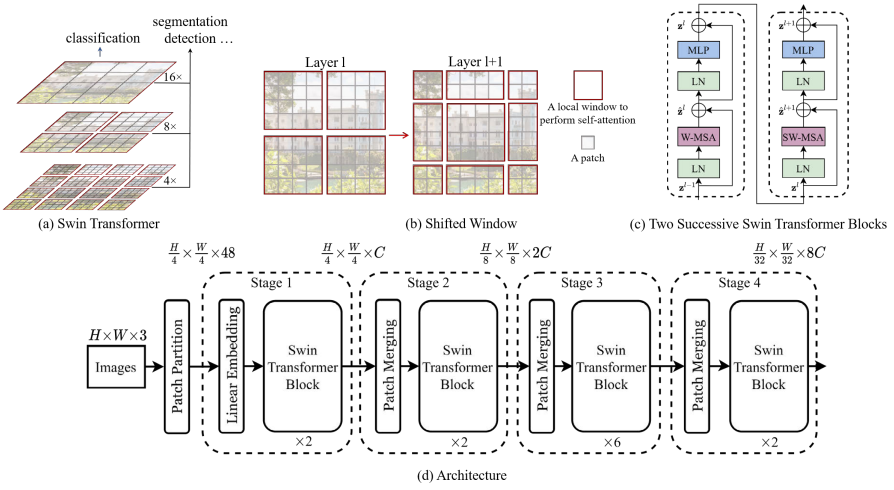



FIGURE 2.8: Swin Transformer V2 framework [11].

The primary advantages of ViTs in object detection are their scalability to large datasets and superior performance in capturing long-range dependencies. This makes them particularly effective in scenarios where contextual understanding is

FIGURE 2.9: Air-to-air object detection of micro-UAVs with a monocular camera [12].

crucial. Additionally, ViTs demonstrate strong transfer learning capabilities, performing well across various domains with minimal fine-tuning. However, challenges with ViTs include their computational intensity due to self-attention mechanisms, particularly when processing high-resolution images. This can limit their deployment in real-time applications where computational resources are constrained. Additionally, ViTs often require large-scale datasets for pre-training to achieve optimal performance, which can be a limitation in data-scarce environments. Despite these challenges, ongoing advancements in ViT architectures, such as the development of efficient attention mechanisms [123] and hybrid CNN-Transformer models [124], continue to enhance their applicability and performance in diverse object detection tasks.

### 2.2.4 Summary

When applying object detection algorithms to drone applications, it is necessary to find the best balance between computation speed and accuracy. Besides, massive drone datasets are required for training and testing. Zheng Ye *et al.* [12] collected an air-to-air drone dataset "Det-Fly" (see Fig. 2.9) and evaluated air-to-air object detection of a micro-UAV with eight different object detection algorithms, namely RetinaNet [125], SSD, Faster R-CNN, YOLOv3 [126], FPN [127], Cascade R-CNN[128] and Grid R-CNN [129]. The evaluation results in [12] showed that the overall performance of Cascade R-CNN and Grid R-CNN is superior compared to the others. However, the YOLOv3 provides the fastest inference speed among others. Wei Xun *et al.* [130] conducted another investigation into drone detection, employing the YOLOv3 architecture and deploying the model on the NVIDIA Jetson TX2 platform. They collected a dataset comprising 1435 images featuring various UAVs, including drones, hexacopters, and quadcopters. Utilizing custom-trained weights, the YOLOv3 model demonstrated proficiency in drone detection within images. However, the deployment of this trained model faced constraints due to the limited computation capacity of the Jetson TX2, which posed challenges for effective real-time application.

In agile flight, computation speed is more important than accuracy since real-time object detection is required to avoid obstacles swiftly. Therefore, a simple gate detector and a filter algorithm are adopted as the basis of the visual perception of Swift [25]. Considering the agility of the drone, a stabilization module is required to obtain more robust and accurate object detection and tracking results in real-time flight. Moreover, in the drone datasets covered in existing works, each image only includes a single UAV. To classify and detect different classes of drones in multi-drone systems, a new dataset of multiple types of drones has to be built from scratch. Furthermore, the dataset can be adapted to capture adversary drones in omnidirectional visual perception to enhance avoidance capability.

## 2.3 Vision-based control

Vision-based control for robotics has been widely studied in recent years, whether for ground robots or aerial robotics such as drones. For drones flying in a GPS-denied environment, visual inertial odometry (VIO) [87–89] and visual simultaneous localization and mapping systems (SLAM) [90–92] have been preferred choices for navigation. Meanwhile, in a clustered environment, research on obstacle avoidance [16, 19, 45, 67] based on visual perception has attracted much attention in the past few years. Obstacle avoidance has been a main task for vision-based control as well as for the current learning algorithms of drones.

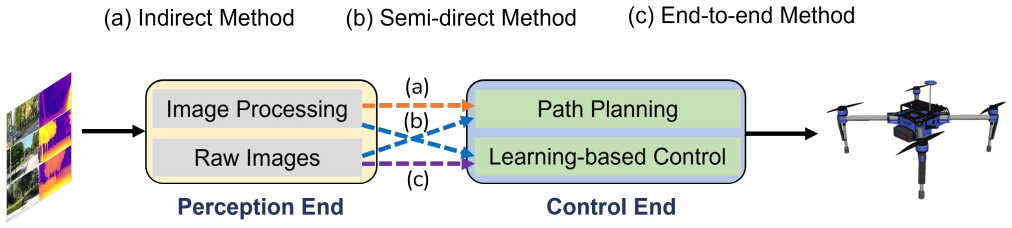



FIGURE 2.10: Vision-based control methods for drone applications. Based on the ways of visual perception and control, the methods can be divided into indirect methods, semi-direct methods, and end-to-end methods.

From the perspective of how drones obtain visual perception (perception end) and how drones generate control commands from visual perception (control end), existing vision-based control methods can be categorized into indirect methods, semi-direct methods, and end-to-end methods. The indirect methods divide the task into perception, image processing, and path planning, while the end-to-end methods generate the actions based on learning from raw images directly. There are two ways to design the semi-direct method architectures, one is to generate the required intermediate information from image processing (such as the relative positions) and train the control policy with deep reinforcement learning, another way is to extract the required states (such as depth image) from images directly and avoid the obstacles with path planning; By understanding and leveraging these different methods, we can enhance the capabilities and efficiency of vision-based drones. The relationship between these three categories is illustrated in Fig. 2.10. In the following, we will discuss and evaluate these methods in three different categories, respectively.

### 2.3.1 Indirect methods

Indirect methods [5, 13, 45, 131–138] refer to extracting features from images or videos to generate visual odometry, depth maps, and 3D point cloud maps for drones to perform path planning based on traditional optimization algorithms (see Fig. 2.11). Obstacle states, such as 3D shape, position, and velocity, are detected and mapped before a maneuver is taken. Once online maps are built or obstacles are located, the drone can generate a feasible path or take actions to avoid obstacles.

SOTA indirect methods generally divide the mission into several subtasks, namely **perception**, **mapping** and **planning**. On the perception side, depth images are always required to generate corresponding distance and position information for navigation. A depth image is a grey-level or color image that can represent the distance between the surfaces of objects from the viewpoint of the agent. Fig. 2.12 shows color images and corresponding depth images from a drone's viewpoint. The illuminance is proportional to the distance from the camera. A lighter color denotes a nearer surface, and darker areas mean further surfaces. A depth map provides the necessary distance information for drones to make decisions to avoid static and dynamic obstacles. Currently, off-the-shelf RGB-D cameras, such as the Intel RealSense depth camera D415, the ZED 2 stereo camera, and the Structure Core depth camera, are widely used for drone applications. Therefore, traditional obstacle avoidance methods can treat depth information as a direct input. However, for omnidirectional perception in wide-view scenarios, efficient onboard monocular depth estimation is always required, which is a challenge to address with existing methods.

On the mapping side, a point cloud map [139] or Octmap [140], representing a set of data points in a 3D space is commonly generated. Each point has its own Cartesian coordinates and can be used to represent a 3D shape or an object. A 3D point cloud map is not from the view of a drone but constructs a global 3D map that provides global environmental information for a drone to fly. The point-cloud map can be generated from a lidar scanner or many overlapped images combined with depth information. An illustration of an original scene and a point cloud map are shown in Fig. 2.11, where the drone can travel around without colliding with static obstacles.

Planning is a basic requirement for a vision-based drone to avoid obstacles. Within the indirect methods, planning can be further divided into two categories: one is offline methods based on high-resolution maps and pre-known position information, such as Dijkstra's algorithm [141], A-star [142], RRT-connect[143] and sequential convex optimization [144]; the other is online methods based on real-time visual perception and decision-making. Online methods can be further categorized into online path planning [13, 45, 136] and artificial potential field (APF) methods [5, 15].

Most vision-based drones rely on online methods. Compared to offline methods, which require an accurate pre-built global map, online methods provide advanced maneuvering capabilities for drones, especially in a dynamic environment. Currently, due to the advantages of optimization and prediction capabilities, online path planning methods have become the preferred choice for drone obstacle avoidance. For instance, in the SOTA work [13], Zhou Boyu *et al.* introduced a robust and efficient motion planning system called Fast-Planner for a vision-based drone to perform high-speed flight in an unknown cluttered environment. The key contributions of this work are a robust and efficient planning scheme incorporating path searching, B-spline optimization, and time adjustment to generate feasible and safe trajectories for vision-based drones' obstacle avoidance. Using only onboard vision-based perception and computing, this work demonstrated agile drone navigation in unexplored indoor and outdoor environments. However, this approach can only achieve maximum speeds of $3m/s$ and requires $7.3ms$ for computation in each step. To improve the flight performance and save computation time, Zhou Xin *et al.* [45] provided a Euclidean Signed Distance Field (ESDF)-free gradient-based planning framework solution, EGO-Planner, for drone autonomous navigation in unknown obstacle-rich situations. Compared to the Fast-Planner, the EGO-Planner achieved faster speeds and saved a lot of computation time. However, these online path planning methods require bulky visual sensors, such as RGBD cameras or lidar, and a powerful onboard computer for the complex numerical calculation to obtain a local or global optimal trajectory.

In contrast to online path planning, the artificial potential field methods require less computation resources and can well cope with dynamic obstacle avoidance using limited sensor information. The APF algorithm (see Fig. 2.13) is one of the algorithms in the robot path planning approach that uses attractive force to

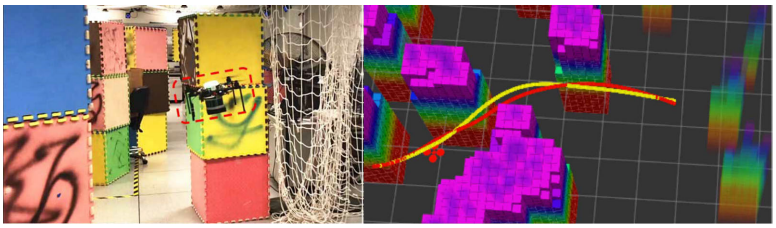


FIGURE 2.11: A vision-based drone is traversing through a cluttered indoor environment with generated point cloud maps and online path planning [13].

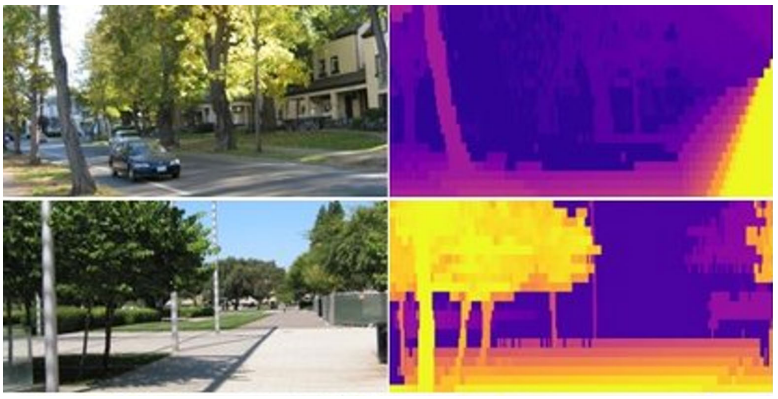
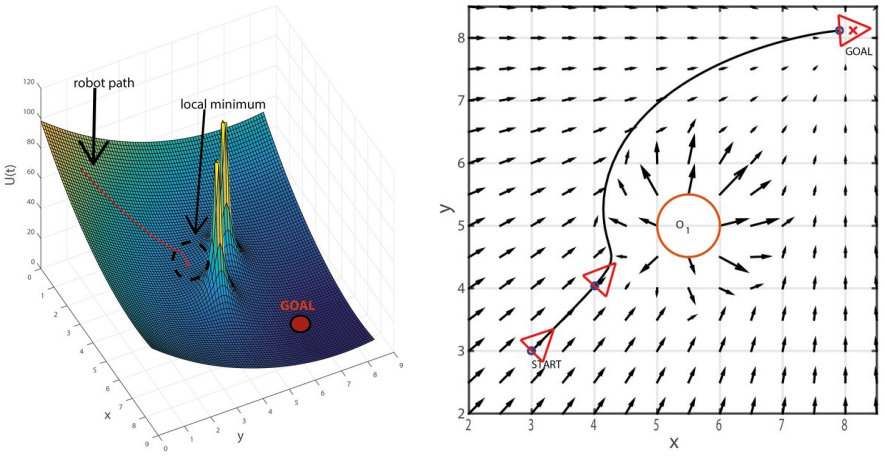


FIGURE 2.12: Depth maps generated from the viewpoint of a drone [14].

achieve the objective position and repulsive force to avoid obstacles in an unknown environment [145]. Falanga *et al.* [5] developed an efficient and fast control strategy based on the artificial potential field method to avoid fast approaching dynamic obstacles. The obstacles in [5] are represented as repulsive fields that decay over time, and the repulsive forces are generated from the first-order derivation of the repulsive fields at each time step. However, the repulsive forces computed only reach substantial values when the obstacle is very close, which may lead to unstable and aggressive behavior. Besides, artificial potential field methods are heuristic methods that cannot guarantee global optimization and robustness for drones. Hence, it is not ideal to adopt potential field methods to navigate through cluttered environments.

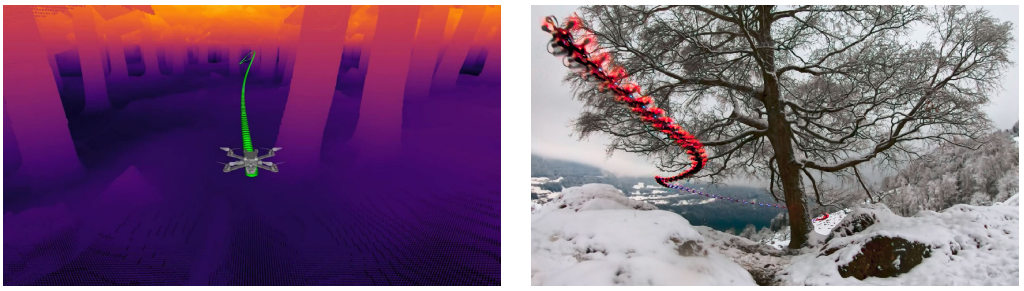
### 2.3.2 End-to-end methods

In contrast to the indirect methods, which divide the whole mission into multiple sub-tasks, such as perception, mapping, and planning, end-to-end methods [16, 17, 146–148] combine computer vision and reinforcement learning (RL) to map the visual observations to actions directly.




(a) (b)

FIGURE 2.13: Artificial potential field (APF) methods in drones' obstacle avoidance [15] (a) Generated artificial potential field; (b) Flight trajectory based on the APF.



(a) (b)

FIGURE 2.14: Quadrotor drone flying in a wild environment with end-to-end reinforcement learning [16]. (a) Training process in the simulation platform; (b) Real flight test in the wild snow environment.

With end-to-end methods, the visual perception of drones is encoded by a deep neural network into an observation vector of the policy network. The mapping process is usually trained offline with abundant data, which requires high-performance computers and simulation platforms. The training data set is collected from expert flight demonstrations for imitation learning or from simulations for online training. To better generalize the performance of a trained neural network model, scenario randomization (domain randomization) is essential during the training process. Malik Aqeel Anwar *et al.* [147] presented an end-to-end reinforcement learning approach called NAVREN-RL to navigate a quadrotor drone in an indoor environment with expert data and knowledge-based data aggregation. The reward function in [147] was formulated from a ground truth depth image and a generated

depth image. Loquercio *et al.* [16] developed an end-to-end approach that can autonomously guide a quadrotor drone through complex wild and human-made environments at high speeds with purely onboard visual perception (depth image) and computation (see Fig. 2.14). The neural network policy was trained in a high-fidelity simulation environment with massive expert knowledge data. While end-to-end methods provide us with a straightforward way to generate obstacle avoidance policies for drones, they require massive training data with domain randomization (usually counted in the millions) to obtain acceptable generalization capabilities. Meanwhile, without expert knowledge data, it is challenging for the neural network policy to update its weights when the reward space is sparse. To implement end-to-end methods, we commonly consider the following aspects: neural network architecture, training process, and Sim2Real transfer.

**Neural Network Architecture:** The neural network architecture is the core component in the end-to-end methods, which determines the computation efficiency and intelligence level of the policy. In the end-to-end method, the input of the neural network architecture is the image raw data (RGB/RGBD), and the output is the action vector an agent needs to take. The images are encoded into a vector and then concatenated with other normalized observations to form an input vector of a policy network. For the image encoder, there are many pre-trained neural network architectures that can be considered, such as ResNet [149], VGG [150], and nature Convolutional Neural Network (CNN) [151]. Zhu *et al.* [17] developed a target-driven visual navigation approach for robots with end-to-end deep reinforcement learning, where a pre-trained ResNet-50 is used to encode the image into a feature vector (see Fig. 2.15 (a)). In [18], a more data-efficient image encoder with 5 layers was designed for target-driven visual navigation for robots with end-to-end imitation learning (see Fig. 2.15 (b)). Before designing the neural network architecture, we first need to determine the observation space and action space of the task. The training efficiency and space complexity are the two aspects we need to consider in the designing process.

**Training Process:** The training process is the most time-consuming part of the end-to-end learning methods. It requires the use of gradient information from the loss function to update the weights of the neural network and improve the policy model. There are two main training algorithms for reinforcement learning, namely the on-policy algorithm and the off-policy algorithm. On-policy algorithms

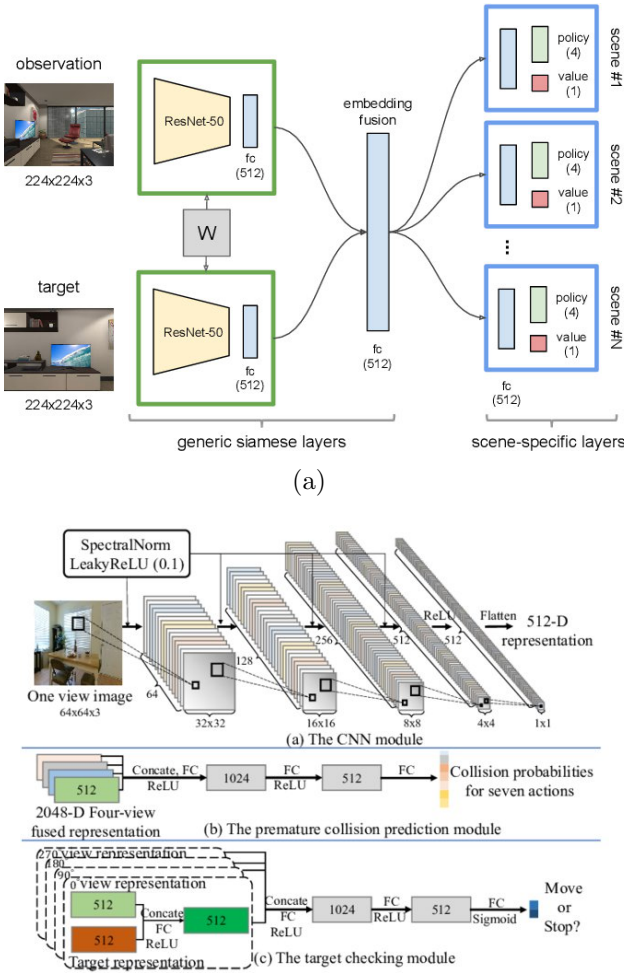



FIGURE 2.15: The neural network architectures of end-to-end reinforcement learning methods. (a) Target driven navigation with ResNet50 encoder[17]; (b) Target driven visual navigation for a robot with shallow CNN encoder [18].

attempt to evaluate or improve the policy that is used to make decisions, i.e., the behavior policy for generating actions is the same as the target policy for learning. When the agent is exploring the environment, on-policy algorithms are more stable than off-policy algorithms when the agent is exploring the environment. SARSA (State-Action-Reward-State-Action experiences to update the Q-values) [104] is an on-policy reinforcement learning algorithm that estimates the value function of the policy being carried out. PPO [111] is another efficient on-policy algorithm widely used in reinforcement learning. By compensating for the fact that more probable actions are going to be taken more often, PPO addresses the issue of high variance and low convergence with policy gradient methods. To avoid large weights' vibration, a "clipped" policy gradient update formulation is designed.

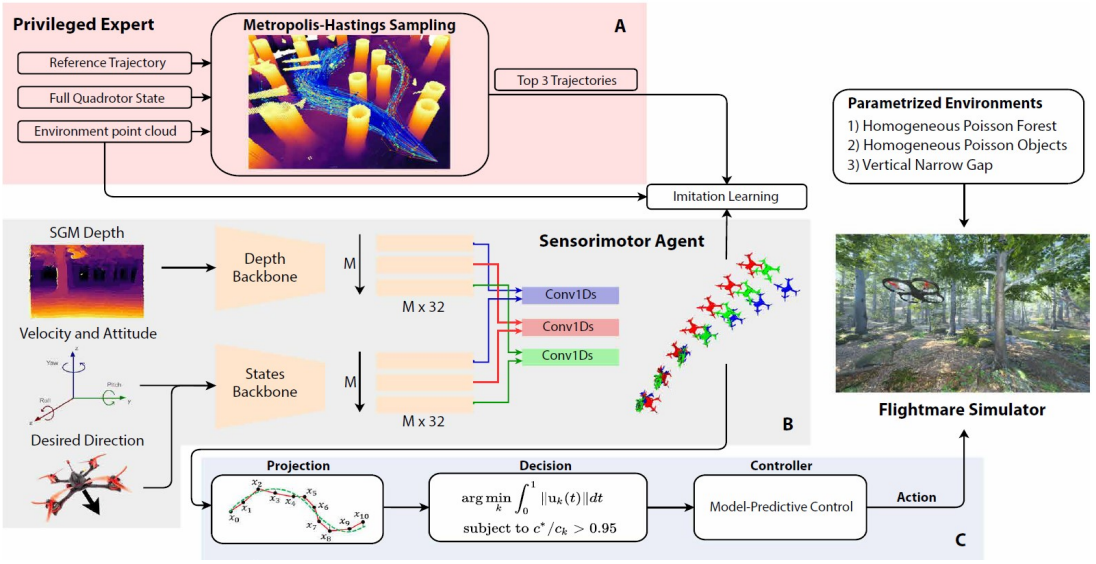



FIGURE 2.16: Training process used in [16] to fly an agile drone through forest.

In contrast, off-policy algorithms evaluate or improve a policy different from that used to generate the action currently. The off-policy algorithms such as Q-learning [152, 153] try to learn a greedy policy in every step. Off-policy algorithms perform better at movement predictions, especially in an unknown environment. However, off-policy algorithms can be very unstable due to the unstatic environments the offline data seldom covers. Imitation learning [154] is another form of off-policy algorithm that tries to mimic demonstrated behavior in a given task. Through training from demonstrations, imitation learning can save remarkable exploration costs. [16] adopted imitation learning to speed up the training process. The whole training process is illustrated in Fig. 2.16. With this privileged expert knowledge, the policy can be trained to find a time-efficient trajectory to avoid obstacles.

For multiagent systems, multiagent reinforcement learning (MARL) is widely studied to encourage agent collaboration. In MARL, the credit assignment is a crucial issue that determines the contribution of each agent to the group's success or failure. COMA[109] is a baseline method that uses a centralized critic to estimate the action-value function, and for each agent, it computes a counterfactual advantage function to represent the value difference, which can determine the contribution of each agent. However, COMA still does not fully address the model complexity issue. QMIX [155] is well developed to address the scalability issue by decomposing the global action-value function into individual agent's value functions. However,

QMIX assumes the environment is fully observable and may not be able to handle scenarios with continuous action space. Hence, attention mechanism-enabled MARL is a promising direction to address the variable observations. Besides, to balance individual and team reward, a MARL with mixed credit assignment algorithm, POCA-Mix, was proposed in [156] to achieve collaborative multitarget search with a visual drone swarm.

**Sim2Real Transfer:** For complex tasks, the policy neural networks are usually trained on simulation platforms such as AirSim [157], Unity [158] or Gazebo [159]. The differences between the simulation and the real environment are non-negligible. Sim2Real is the way to deploy the neural network model trained in a simulation environment on a real physical agent. For deployment, it is essential to validate the generalization capability of trained neural network models. A wide variety of Sim2Real techniques [160–164] have been developed to improve the generalization and transfer capabilities of models. Domain randomization [163] is one of these techniques to improve the generalization capability of the trained neural network to unseen environments. Domain randomization is a method of trying to discover a representation that can be used in a variety of scenes or domains. Existing domain randomization techniques [16, 73] for drones' obstacle avoidance include position randomization, depth image noise randomization, texture randomization, size randomization, etc. Therefore, we need to apply domain randomization in our training process to enhance the generalization capability of the trained model in deployment.

### 2.3.3 Semi-direct methods

Compared to the end-to-end methods, which generate actions from image raw data directly, semi-direct methods [19, 20, 66, 95] introduce an intermediate phase for drones to take actions from visual perception, aiming to improve the generalization and transfer capabilities of the methods over unseen environments. There are two ways to design the semi-direct method architectures: one is to generate the required information from image processing (such as the relative positions of obstacles from object detection and tracking or the point cloud map) and train the control policy with deep reinforcement learning; another is to obtain the required states (such as depth image) directly from the raw image data with deep learning and avoid the

FIGURE 2.17: A drone avoids the obstacles in the forest with semi-direct methods [19].

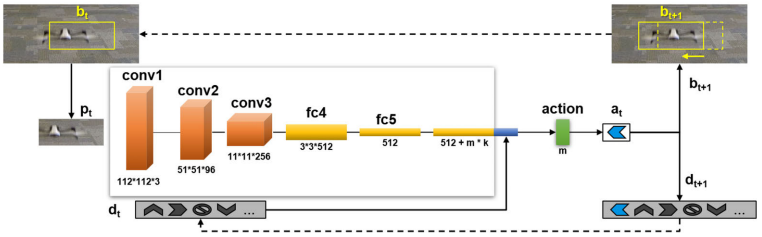



FIGURE 2.18: A drone is tracking and chasing a target drone with object detection and deep reinforcement learning [20].

obstacles using numerical or heuristic methods. These two methods can be denoted as indirect (front end)-direct (back end) methods and direct (front end)-indirect (back end) methods.

**Indirect-direct methods** [19, 20] firstly obtain intermediate features such as relative position or velocity of the obstacles from image processing and then use this intermediate feature information as observations to train the policy neural network via deep reinforcement learning. Indirect-direct methods generally rely on designing suitable intermediate features. In [19], the features related to depth cues such as Radon features (30 dimensional), structure tensor statistics (15 dimensional), Laws' masks (8 dimensional), and optical flow (5 dimensional) were extracted and concatenated into a single feature vector as visual observation. Together with the other nine additional features, the control policy was trained with imitation learning to navigate the drone through a dense forest environment (see Fig. 2.17).

TABLE 2.1: Summary of vision-based control methods in drone technology

| Method Type | Perception End | Control End | Key Studies | Description and Applicability |
|---|---|---|---|---|
| Indirect | Depth maps, 3D point cloud maps | Traditional optimization algorithms | [131], [132], [133], [136] | Focus on generating visual odometry, depth maps, and 3D point cloud maps for path planning. Suitable for safety-critical tasks with accurate models. |
| | Depth maps, 3D point cloud maps | Online path planning, APF, etc. | [5], [137] | Utilize depth information for real-time visual perception and decision-making in dynamic environments. Suitable for rapid response tasks with certain information. |
| End-to-End | Visual observations encoded by DNNs | RL for action mapping | [103], [146], [147] | Combine deep learning for visual perception with RL for direct action response. Suitable for complex tasks with uncertain information. |
| | Encoded visual observations | Imitation learning and online training | [165], [16] | Use deep learning for visual encoding and train the control system with expert demonstrations and online training. Suitable for sample-efficient tasks with expert demonstrations. |
| Semi-Direct | Intermediate features from image processing | DRL for action | [19], [20], [25], [166] | Extract intermediate features like relative positions or velocities and use DRL for action decisions. Suitable for complex tasks with high generalization requirement. |
| | Raw image data for depth images or obstacle tracking | Numerical / Heuristic methods for obstacle avoidance | [66], [95] | Utilize direct image data to obtain necessary states for obstacle avoidance using non-learning-based methods. Suitable for tasks where robust stereo information is not available. |

Moulay *et al.* [20] proposed a semi-direct vision-based learning control policy for UAV pursuit-evasion. Firstly, a deep object detector (YOLOv2) and a search area proposal (SAP) were used to predict the relative position of the target UAV in the next frame for target tracking. Afterward, deep reinforcement learning (see Fig. 2.18) was adopted to predict the actions the follower UAV needs to perform to track the target UAV. Indirect-direct methods are able to improve the generalization

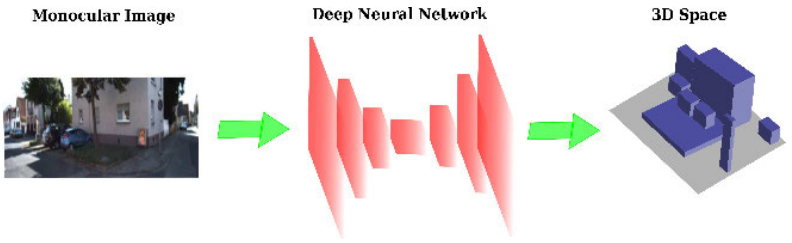



FIGURE 2.19: Via deep learning, a 3D space generated from the monocular image for obstacle avoidance [21].

capability of policy neural networks, but at the cost of heavy computation and time overhead.

**Direct-indirect methods** try to obtain depth images [167, 168] and track obstacles [21, 169] from training (see Fig. 2.19) and use non-learning-based methods, such as path planning or APF to avoid obstacles. Direct-indirect methods can be applied to a microlight drone with only monocular vision, but they require a lot of training data to obtain depth images or 3D poses of the obstacles. Michele *et al.* [21] developed an object detection system to detect obstacles at a very long range and at a very high speed, without certain assumptions on the type of motion. With a deep neural network trained on real and synthetic image data, fast, robust and consistent depth information can be used for drones' obstacle avoidance. Direct-indirect methods address the ego drift problem of monocular depth estimation using Structure from Motion (SfM) and provide a direct way to get depth information from the image. However, the massive train dataset and limited generalization capability are the main challenges for their further applications.

In summary, the field of vision-based control for drones encompasses a variety of methods, each with its own unique approach to perception and control. Indirect methods rely on traditional optimization algorithms and depth or 3D point cloud maps for navigation and obstacle avoidance. End-to-end methods leverage deep neural networks for visual perception and utilize reinforcement learning for direct action mapping. Semi-direct methods balance between computational efficiency and generalization by using intermediate features from image processing and a combination of DRL and heuristic methods for action generation. A comprehensive overview of these methods, along with key studies in each category, is summarized in Table 2.1, which provides a detailed comparison of their perception and control strategies.

## 2.4 Applications and challenges

### 2.4.1 Single drone application

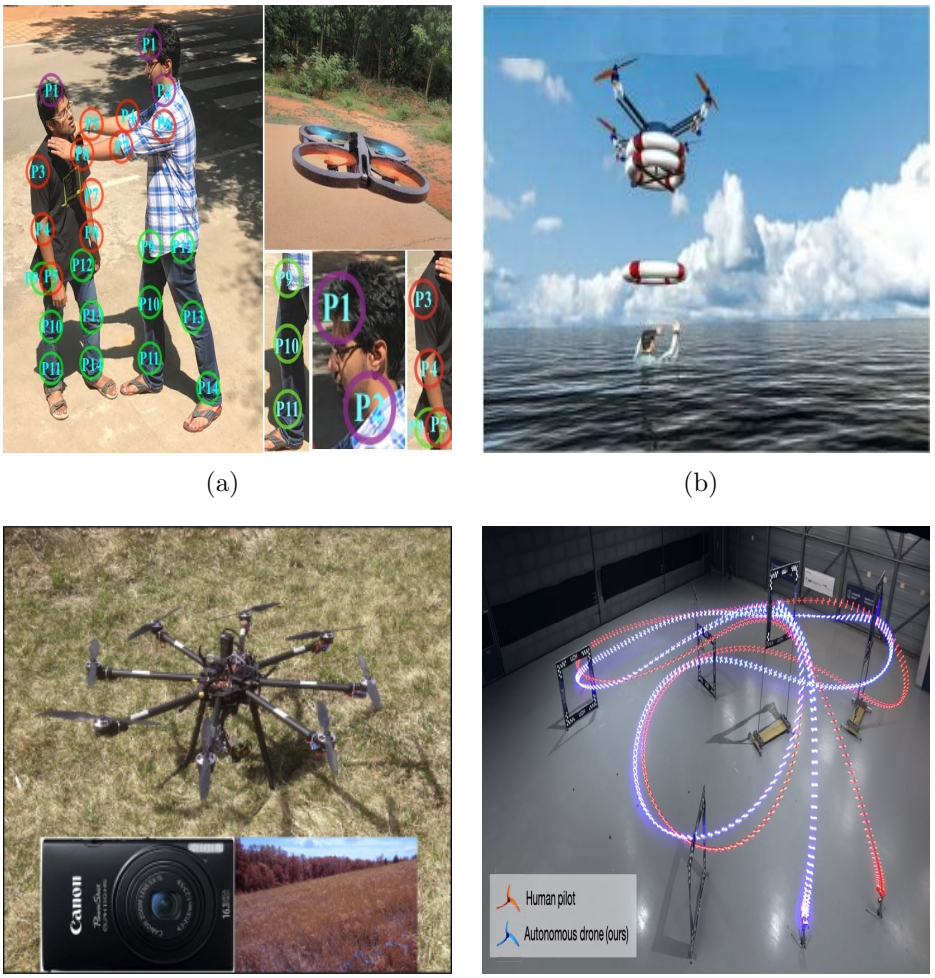
The versatility of single drones is increasingly recognized in a variety of challenging environments. These unmanned vehicles, with their inherent advantages, are effectively employed in critical areas such as hazardous environment detection and search and rescue operations. Single drone applications in vision-based learning primarily involve tasks like obstacle avoidance [16, 48], surveillance [22, 170–172], search-and-rescue operations [173–175], environmental monitoring [24, 176–178], industrial inspection [179–181] and autonomous racing [182–184] (several examples are illustrated in Fig. 2.20). Each field, while benefiting from the unique capabilities of drones, also presents its own challenges and areas for development.

#### 2.4.1.1 Obstacle avoidance

The development of obstacle avoidance capabilities in drones, especially for vision-based control systems, poses significant challenges. Recent studies have primarily focused on static or simple dynamic environments, where obstacle paths are predictable [16, 46, 185]. However, complex scenarios involving unpredictable physical attacks from birds or intelligent adversaries remain largely unaddressed. For instance, [5, 46, 48, 186] have explored basic dynamic obstacle avoidance but do not account for adversarial environments. To effectively handle such threats, drones require advanced features like omnidirectional visual perception and agile maneuvering capabilities. Current research, however, is limited in addressing these needs, underscoring the necessity for further development in drone technology to enhance evasion strategies against smart, unpredictable adversaries.

#### 2.4.1.2 Surveillance

While drones play a pivotal role in surveillance tasks, their deployment is not without challenges. Key obstacles include managing high data processing loads and addressing the limitations of onboard computational resources. In addressing these challenges, the study by Singh *et al.* [22] presented a real-time drone surveillance system used to identify violent individuals in public areas. The proposed




(a) (b)

(c) (d)

FIGURE 2.20: Applications of single drone. (a) Surveillance [22]; (b) Search and Rescue [23]; (c) Environmental Monitoring [24]; (d) Autonomous Racing [25].

study was facilitated by cloud processing of drone images to address the challenge of slow and memory-intensive computations while still maintaining onboard short-term navigation capabilities. Additionally, in the study [172], a drone-based crowd surveillance system was tested to achieve the goal of saving scarce energy of the drone battery. This approach involved offloading video data processing from the drones by employing the Mobile Edge Computing (MEC) method. Nevertheless, while off-board processing diminishes computational demands and energy consumption, it inevitably heightens the need for data transmission. Addressing the challenge of achieving real-time surveillance in environments with limited signal connectivity is an additional critical issue that requires resolution.

#### 2.4.1.3 Search and rescue

In the field of search and rescue operations, a number of challenges exist that hinder the development of drone technology. A primary challenge faced by drones is extracting maximum useful information from limited data sources. This is crucial for improving the efficiency and success rate of these missions. Goodrich *et al.* [173] address this by developing a contour search algorithm designed to optimize video data analysis, enhancing the capability to identify key elements swiftly. However, incorporating temporal information into this algorithm introduces additional computational demands. These increased requirements present new challenges, such as the need for more powerful processing capabilities and potentially greater energy consumption.

#### 2.4.1.4 Environmental monitoring

A major challenge in the application of drones for environmental monitoring lies in efficiently collecting high-resolution data while navigating the constraints of battery life, flight duration, and diverse weather conditions. Addressing this, Senthilnath *et al.* [176] showcased the use of fixed-wing and VTOL (Vertical Take-Off and Landing) drones in vegetation analysis, focusing on the challenge of detailed mapping through spectral-spatial classification methods. In another study, Lu *et al.* [24] demonstrated the utility of drones for species classification in grasslands, contributing to the development of methodologies for drone-acquired imagery processing, which is crucial for environmental assessment and management. While these studies represent significant steps in drone applications for environmental monitoring, several challenges persist. Future research may need to address the problems of improving the drones' resilience to diverse environmental conditions, and extend their operational range and duration to comprehensively cover extensive and varied landscapes.

#### 2.4.1.5 Industrial inspection

In industrial inspection, drones face key challenges like safely navigating complex environments and conducting precise measurements in the presence of various disturbances. Kim *et al.* [179] addressed the challenge of autonomous navigation by

using drones for proximity measurement among construction entities, enhancing safety in the construction industry. Additionally, Khuc *et al.* [180] focused on precise structural health inspection with drones, especially in high or inaccessible locations. Despite these advancements in autonomous navigation and measurement accuracy, maintaining data accuracy and reliability in industrial settings with interference from machinery, electromagnetic fields, and physical obstacles continues to be a significant challenge, necessitating further research and development in this domain.

#### 2.4.1.6 Autonomous racing

In autonomous drone racing, the central challenge is reducing delays exist in visual information processing and decision making and enhancing the adaptability of perception networks. In [182], a novel sensor fusion method was proposed to enable high-speed autonomous racing for mini-drones. This work also addressed issues with occasional large outliers and vision delays commonly encountered in fast drone racing. Another work [183] introduced an innovative approach to drone control, where a deep neural network (DNN) was used to fuse trajectories from multiple controllers. In the latest work [25], the vision-based drone outperformed world champions in the racing task, purely relying on onboard perception and a trained neural network. The primary challenge in autonomous drone racing, as identified in these studies, lies in the need for improved adaptability of perception networks to various environments and textures, which is crucial for the high-speed demands of the sport.

Overall, the primary challenges in single drone applications include limited battery life, which restricts operational duration, and the need for effective obstacle avoidance in varied environments. Additionally, limitations in data processing capabilities affect real-time decision-making and adaptability. Advanced technological solutions are essential to overcome these challenges, ensuring that single drones can operate efficiently and reliably in diverse scenarios and paving the way for future innovations.

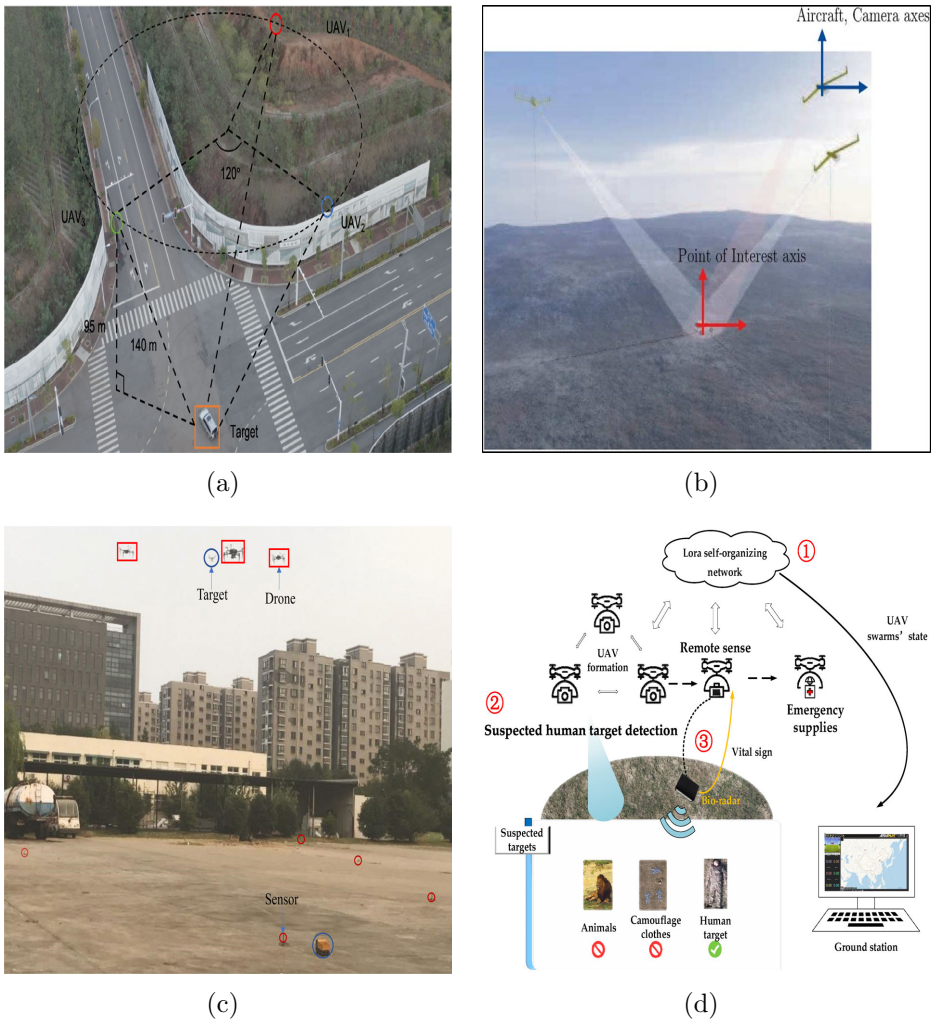



FIGURE 2.21: Applications of multi-drone. (a) Coordinated Surveying [26]; (b) Cooperative Tracking [27]; (c) Synchronized Monitoring [28]; (d) Disaster Response [29].

### 2.4.2 Multi-drone application

While single drones offer convenience, their limited monitoring range has prompted interest in multi-drone collaboration. This approach seeks to overcome range limitations by leveraging the collective capabilities of multiple drones for broader, more efficient operations. Multi-drone applications, encompassing activities such as coordinated surveying [26, 50–52], cooperative tracking [27, 53, 54], synchronized monitoring [28, 55, 56], and disaster response [23, 29, 57, 58] (see several examples in Fig. 2.21), bring the added complexity of inter-drone communication, coordination and real-time data integration. These applications leverage the combined

capabilities of multiple drones to achieve greater efficiency and coverage than single drone operations.

#### 2.4.2.1 Coordinated surveying

In coordinated surveying, several challenges are prominent: merging diverse data from individual drones, and addressing computational demands in cooperative process. These challenges were tackled by some works. In [50], a monocular visual odometry algorithm was used to enable autonomous onboard control with cooperative localization and mapping. This work addressed the challenges of coordinating and merging different maps constructed by each drone platform and what's more, the computational bottlenecks typically associated with 3D RGB-D cooperative SLAM. Micro-air vehicles also play an outstanding role in the field of coordinated surveying. Similarly, in [52], a sensor fusion scheme was proposed to improve the accuracy of localization in MAV fleets. Ensuring that each MAV effectively contributes to the overall perception of the environment poses a challenge addressed in this work. Both methods depend on effective collaboration and communication among multiple agents. However, maintaining stable communication between drones remains a critical and unresolved issue in multi-drone operations.

#### 2.4.2.2 Cooperative tracking

In the field of multi-drone object tracking, navigating complex operational environments and overcoming limited communication bandwidth are significant challenges. These topics have also garnered significant research interest. In the study [53], researchers developed a system for cooperative surveillance and tracking in urban settings, specifically tackling the issue of collision avoidance among drones. Additionally, another work by Farmani *et al.* [54] explores a decentralized tracking system for drones, focusing on overcoming limited communication bandwidth and intermittent connectivity challenges. However, a persistent difficulty in this field is navigating complex external environments. Effective path planning and avoiding obstacles during multi-drone operations remain crucial challenges that require ongoing attention and innovation.

#### 2.4.2.3 Synchronized monitoring

In synchronized monitoring missions with multiple drones, the focus is on how to effectively allocate tasks among drones and improve overall mission efficiency, as well as overcoming computational limitations. Gu *et al.* in [28] developed a small target detection and tracking model using data fusion, establishing a cooperative network for synchronized monitoring. This study also addresses task distribution challenges and efficiency optimization. Moreover, [55] explores implementing deep neural network (DNN) models on computationally limited drones, focusing on reducing classification latency in collaborative drone systems. However, issues like collision avoidance in complex environments and signal interference in multi-drone systems are not comprehensively addressed in these studies.

#### 2.4.2.4 Disaster response

In the domain of multi-drone systems for disaster response, challenges such as autonomous navigation, communication limitations, and real-time decision-making are paramount. To address these problems, Tang *et al.* [57] introduced a tracking-learning-detection framework with an advanced flocking strategy for exploration and search missions. However, the study does not examine the crucial aspect of task distribution and optimization for enhancing mission efficiency in multi-drone disaster response. This shortcoming points to an essential area for further research, as effective task management is key to utilizing the full capabilities of multi-drone systems, particularly in the dynamic and urgent context of disaster scenarios.

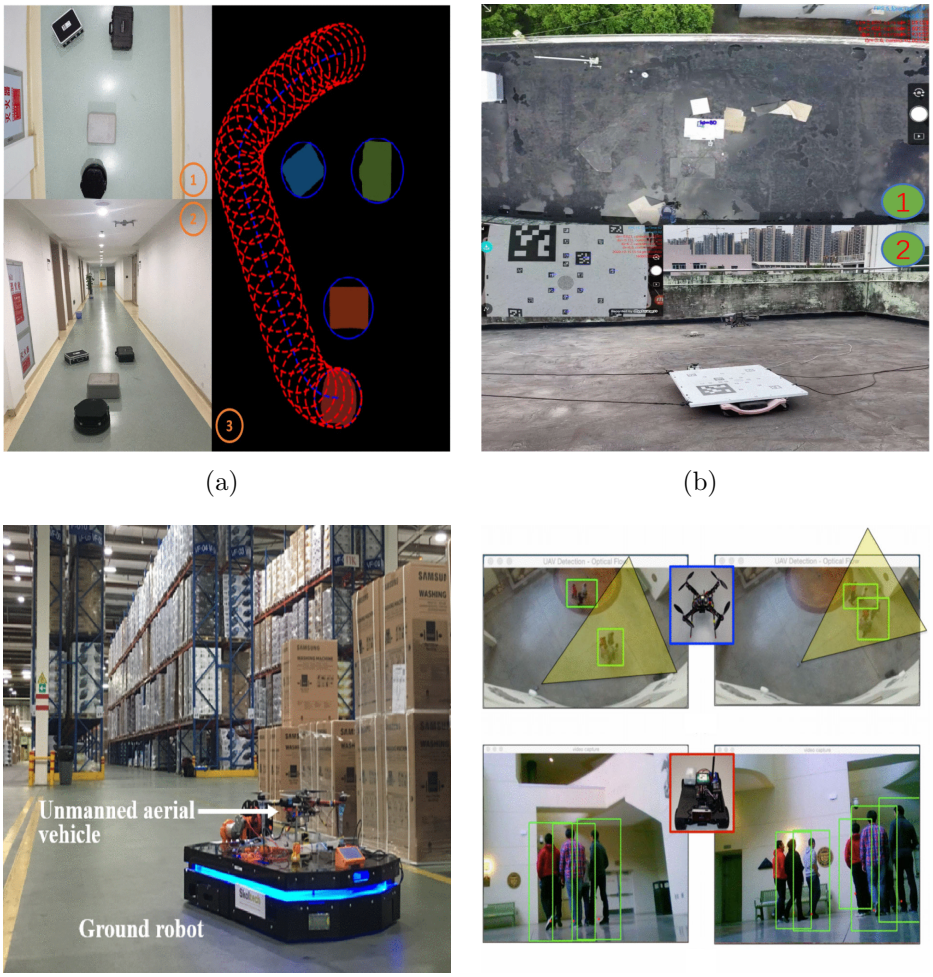
The primary challenges in multi-drone applications include maintaining stable and reliable communication links, collision avoidance between drones, and effective distribution of tasks to optimize the overall mission efficiency. Additionally, issues like signal interference and managing the flight paths of multiple drones simultaneously are significant hurdles. Addressing these challenges is vital for unlocking the full potential of multi-drone systems, paving the way for advancements in various application domains.

### 2.4.3 Heterogeneous systems application

As the complexity and uncertainty of task scenarios escalate, it becomes increasingly challenging for a single robot or even a homogeneous multi-robot system to efficiently adapt to diverse environments. Consequently, in recent years, heterogeneous multi-robot systems (HMRS) have emerged as a focal point of research within the community [187]. In the domain of UAV applications, HMRS mainly refers to communication-enabled networks that integrate various UAVs with other intelligent robotic platforms, such as unmanned ground vehicles (UGVs) and unmanned surface vehicles (USVs). This integration facilitates a diverse range of applications, leveraging the unique capabilities of each system to enhance overall operational efficiency. These systems execute a range of tasks, either individually or in a collaborative manner. The inherently heterogeneous scheduling approach of HMRS significantly enhances feasibility and adaptability, thereby effectively tackling a series of demanding tasks across different environments. Consequently, applications of HMRS are rapidly evolving; examples include but are not limited to localization and path planning[30, 188–190], precise landing [31, 191, 192], and comprehensive inspection and detection [32, 33, 193] (see examples in Fig. 2.22).

#### 2.4.3.1 Localization and path planning

The UAV-UGV cooperation system can address the challenges of GPS denial and limited sensing range in UGVs. In this system, UAVs provide essential auxiliary information for UGV localization and path planning, relying solely on cost-effective cameras and wireless communication channels. Niu [30] introduced a framework wherein a single UAV served multiple UGVs, achieving optimal path planning based on aerial imagery. This approach surpassed traditional heuristic path planning algorithms in performance. Furthermore, Liu *et al.* [188] presented a joint UAV-UGV architecture designed to overcome frequent target occlusion issues encountered by single-ground platforms. This architecture enabled accurate and dynamic target localization, leveraging visual inputs from UAVs.




(a) (b)

(c) (d)

FIGURE 2.22: Applications of heterogeneous systems. (a) UAV-UGV path planning [30]; (b) UAV-UGV precise landing [31]; (c) UAV-UGV inventory inspection [32]; (d) UAV-UGV object detection [33].

#### 2.4.3.2 Precise landing

Given the potential necessity for battery recharging and emergency maintenance of UAVs during extended missions, the UAV-UGV heterogeneous system can facilitate UAV landings. This design reduces reliance on manual intervention while enhancing the UAVs' capacity for prolonged, uninterrupted operation. In the study [31], a vision-based heterogeneous system was proposed to address the challenge of UAVs' temporary landings during long-range inspections. This system accomplished precise target geolocation and safe landings in the absence of GPS data by detecting QR codes mounted on UGVs. Additionally, Xu *et al.* [191] illustrated the application of UAV heterogeneous systems for landing on USVs. A similar

approach was explored for UAVs' target localization and landing, leveraging QR code recognition on USVs. Collectively, these studies underscored the feasibility and adaptability of the heterogeneous landing system across different platforms.

#### 2.4.3.3 Inspection and detection

Heterogeneous UAV systems present an effective solution to overcome the limitations of background clutter and incoherent target interference often encountered in single-ground vision detection platforms. By leveraging the expansive field of view and swift scanning capabilities of UAVs, in conjunction with the endurance and high accuracy of UGVs, such heterogeneous systems can achieve time-efficient and accurate target inspection and detection in specific applications. For instance, Kalinov *et al.* [32] introduced a heterogeneous inventory management system, pairing a ground robot with a UAV. In this system, the ground robot determined motion trajectories by deploying the SLAM algorithm, while the UAV, with its high maneuverability, was tasked with scanning barcodes. Furthermore, Pretto [193] developed a heterogeneous farming system to enhance agricultural automation. This innovative system utilized the aerial perspective of the UAV to assist in farmland segmentation and the classification of crops from weeds, significantly contributing to the advancement of automated farming practices.

To sum up, most of the applications above primarily focus on single UAV to single UGV or single UAV to multiple UGV configurations, with few scenarios designed for multiple UAVs interacting with multiple UGVs. It is evident that there remains significant research potential in the realm of vision-based, multiagent-to-multiagent heterogeneous systems. Key areas such as communication and data integration within heterogeneous systems, coordination and control in dynamic and unpredictable environments, and individual agents' autonomy and decision-making capabilities warrant further exploration.

## 2.5 Conclusion

This chapter has thoroughly explored the rapidly growing field of vision-based learning for drones, particularly emphasizing their evolving role in multi-drone

systems and complex environments such as collaborative missions and adversarial settings. The investigation revealed that drones are increasingly becoming sophisticated, autonomous systems capable of intricate tasks, largely driven by advancements in AI, machine learning, and sensor technology. Crucially, the integration of visual perception with machine learning algorithms, including deep reinforcement learning, opens up new avenues for drones to operate with greater efficiency and intelligence. These capabilities are particularly pertinent in the context of object detection and decision-making processes, which are vital for complex drone operations. This chapter categorized vision-based control methods into indirect, semi-direct, and end-to-end methods, offering a nuanced understanding of how drones perceive and interact with their environment. Applications of vision-based learning drones, spanning from single-agent to multiagent and heterogeneous systems, demonstrate their versatility and potential in various sectors, including agriculture, industrial inspection, and emergency response. However, this expansion also brings forth challenges such as data processing limitations, real-time decision-making, and ensuring robustness in diverse operational scenarios, which motivate the work of this thesis.

# Part I

# Vision-based learning in collaborative environments

# Chapter 3

# Learning target search in collaborative environments

While there has been significant work on individual drones using DRL, the application of DRL to a swarm of visual drones remains largely unexplored. This part aims to investigate how DRL can enhance the collective behavior and decision-making capabilities of a drone swarm, enabling more efficient collaboration. In this research, collaborative target search problem is studied as this problem involves drones working together to search for and navigate to targets without prior information. The limitations of single-agent systems, such as limited perception and reduced efficiency, are addressed through collaborative efforts.

This chapter explores learning collaborative target search for a visual drone swarm. Equipping drones with target search capabilities is highly desirable for applications in disaster rescue and smart warehouse delivery systems. Multiple intelligent drones that can collaborate with each other and maneuver among obstacles show more effectiveness in accomplishing tasks in a shorter amount of time. However, carrying out collaborative target search (CTS) without prior target information is extremely challenging, especially with a visual drone swarm. In this chapter, we propose a novel data-efficient deep reinforcement learning approach called Adaptive Curriculum Embedded Multistage Learning (ACEMSL) to address these challenges, mainly 3D sparse reward space exploration with limited visual perception and collaborative behavior requirements. Specifically, we decompose the CTS task into several subtasks including individual obstacle avoidance,

target search, and inter-agent collaboration, and progressively train the agents with multistage learning. Meanwhile, an adaptive embedded curriculum is designed, where the task difficulty level can be adaptively adjusted based on the success rate achieved in training. ACEMSL allows data-efficient training and individual-team reward allocation for the visual drone swarm. Furthermore, we deploy the trained model over a real-world visual drone swarm and perform CTS operations without fine-tuning. Extensive simulations and real-world flight tests validate the effectiveness and generalizability of ACEMSL. The project is available at `https://github.com/NTU-ICG/CTS-visual-drone-swarm.git`.

## 3.1 Introduction

As microelectronic technology, sensors, and onboard computing capabilities have advanced significantly in recent years, autonomous microdrones have been widely used in various missions [194]. The use of autonomous agents in target search is becoming increasingly necessary in smart warehouse systems and disaster management [195], such as parcel pickup and delivery, and building collapse rescue. These scenarios demand efficient environment exploration, accurate situation awareness, and effective navigation in cluttered environments. However, the limited sensing range of a single drone can hinder the success of these tasks. Using a drone swarm, on the other hand, allows more comprehensive observation information and the ability to execute tasks in parallel [196], making it an ideal solution for efficient target search. A drone swarm with formation maneuverability [197] has been investigated to perform primary inspection tasks in desired geometric shapes but lacks significant collaborations. However, effective collaboration among agents is crucial for a drone swarm to successfully carry out the aforementioned tasks. It allows drones to assist each other in increasing the success rate of tasks and reducing the time and effort required to maneuver in cluttered environments without collisions.

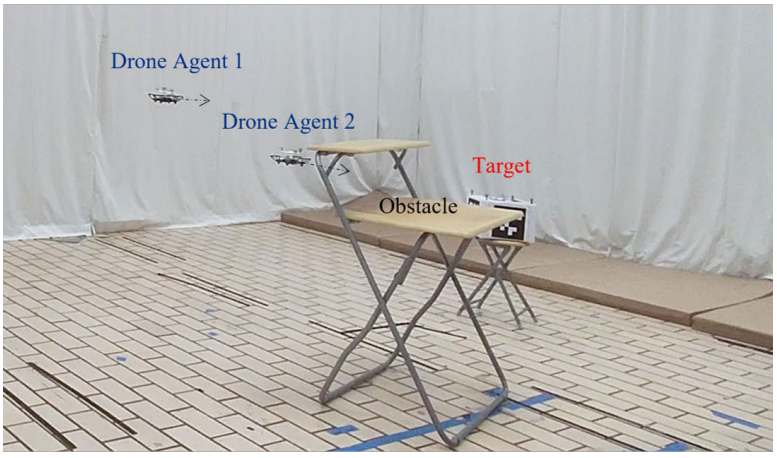
Collaborative Target Search (CTS) behavior is commonly observed among intelligent biological swarms [198], such as honeybees searching for nectar (see Fig. 3.1), ants finding food and birds looking for a living space. CTS tries to locate and reach the target using multiple agents while efficiently exploring the environment. Achieving effective collaboration in a drone swarm is rather challenging, as the individual agent must be able to make intelligent decisions without the guidance

of a leader or knowledge of the global environment map. To accomplish CTS, real-time perception and coordinated decision-making and planning are fundamentally required [199].

The conventional idea to accomplish the CTS task is to formulate an optimization problem and break it down into local optimization sub-problems. There are two traditional ways to solve this local optimization problem, namely numerical analysis methods [96, 200–202] and heuristic methods [203–209]. Under the category of numerical analysis methods, Haumann et al. [96] achieved multi-robot exploration and path planning via optimization partitioning of the objective function for each agent. Soria et al. [200] adopted Nonlinear Model Predictive Control (NMPC) to optimize the navigation performance of an aerial robotic swarm towards a known targeted area in a cluttered indoor environment. The numerical analysis method necessitates knowledge of precise agents' dynamics and complex computation to obtain feasible solutions that satisfy constraints and boundary conditions. Therefore, it is extremely challenging to extend these methods to a large-scale swarm. Under the category of heuristic methods, based on the genetic algorithm, Hayat et al. [205] formulated a multi-objective optimization algorithm to allocate tasks and find a target in a bounded area for a team of UAVs. In [206], evolutionary algorithms (EAs) were developed to tackle criminal search problems with human-UAV collaboration. A search strategy based on particle swarm optimization (PSO) was proposed in [207] for a nano-drone swarm to locate a gas source in unknown and cluttered environments. These heuristic methods, on the other hand, can save modeling and computation resources but do not guarantee to obtain the optimal solution for CTS tasks, resulting in feasible-only or local search behaviors even with perfect information.

Recently, deep reinforcement learning (DRL) has been applied to robots for collaborative exploration [210–212] in unknown environments. Luo et al. [210] formulated the environment exploration problem in a graph structure and applied the graph convolutional network (GCN) to achieve space allocation. In [211], a hierarchical DRL integrated control architecture with dynamic Voronoi partitions was proposed to accomplish multiple mobile robots' cooperative exploration while relative target position observation was required. To address the problem of an exponential increase in the joint action space of multiple agents, Liu et al. [212] proposed Feudal Latent-space Exploration (FLE) to improve the performance of multiagent

(a)




(b)

FIGURE 3.1: (a) A honeybee swarm is searching for nectar collaboratively [`https://encr.pw/ZK1oB`]. (b) A visual drone swarm is searching for a target parcel box controlled by reinforcement learning.

coordination. However, differentiating from the full environment exploration problem, the CTS problem aims to infer the potential locations of an unknown target instead of maximizing the exploration area covered. Even though full environment exploration guarantees a feasible solution for target search, it would be more efficient for agents to directly focus on the potential space containing the target, i.e., probabilistic search methods [201]. Furthermore, real-time visual perception is challenging to integrate into coordinate exploration as it requires complex computation, which is rarely considered in existing DRL-based environment exploration methods. While the target-driven visual navigation [17, 213] with DRL utilizes purely visual perception and a policy neural network to guide a robot through an indoor environment where the target is fixed within a spatial layout, our work

focuses on the highly challenging multiagent CTS mission over a 3D sparse reward space with only limited visual perception and egocentric observations and the target is randomly placed without prior knowledge.

To the best of our knowledge, this work is the first visual drone swarm adopted for CTS in a 3D cluttered environment to search for and approach a parcel box without prior target information. The Success Rate (SR) and the Time To Reach (TTR) are the main metrics used to evaluate the performance of CTS. There are several challenges to accomplish this CTS task for drones: 1)firstly, drone agents need to be motivated to efficiently explore the 3D environment with sparse rewards; 2) secondly, the drone is required to differentiate the target, obstacles, and other agents with only forward color cameras during flight; 3) lastly, the drone swarm should display collaborative behaviors for target search, i.e., without colliding with obstacles or other agents. When it comes to real-world applications with limited sensing capabilities, these challenges become more pronounced and existing methods fail to address them effectively.

To this end, we propose a novel and data-efficient DRL-based method named Adaptive Curriculum Embedded Multistage Learning (ACEMSL) to address the aforementioned challenges. Compared to the standard curriculum learning methods [72–74] where the learning curriculum involves pre-designed parameters that change across the training process, our approach divides the training process into two adaptive curriculum embedded stages. This can reduce the amount of data required to achieve the best performance due to the self-adjustment capability of the curriculum. Meanwhile, CM3 (Cooperative Multi-goal, Multi-stage, Multiagent) [110] has validated multistage learning's data efficiency and outstanding performance for cooperative multi-goal multiagent tasks like cooperative navigation. Similar to CM3, in our approach, a single greedy agent is trained prior to multiple agents' collaborative learning. However, CM3 considers its two stages as a complete curriculum, which is hard to extend on a large scale and requires knowledge of goal assignment, while our ACEMSL embeds an adaptive curriculum into each stage without knowing the target's location. In each stage, the probability of hiding the target from a drone swarm increases when the desired SR is achieved. As such, the drone agents are gradually guided to explore the environment with sparse rewards and try to search for the potential target. To improve learning efficiency, the shared neural network of each agent is trained in a small space before being

transferred to an unknown larger space. To enhance the generalization capability of the model, domain randomization, which includes the initial state of drones, parameters of the target, and the intensity of light, is adopted during the training process. To validate our algorithm and learning framework, physical experiments with real-time visual perception are conducted in various scenarios. Our **main contributions** are summarized as follows:

- Without any prior knowledge of the target and the environment layout, we propose a novel multistage reinforcement learning approach named ACEMSL to efficiently train a scalable decentralized visual drone swarm to collaboratively find the target with limited observation.
- Differentiating from the pre-designed curriculum in the existing DRL-based environment exploration methods, we propose a novel adaptive embedded curriculum (AEC) algorithm to guide the visual drone to explore and navigate 3D cluttered environments with sparse rewards. The convergence of AEC is further proved.
- To the best of our knowledge, our work is the first to successfully transfer a trained policy from simulation to a real-life visual drone swarm and demonstrate their CTS performance with real-time visual perception in physical experiments.

## 3.2 Related work

In the domain of drones [16, 73, 101, 214–216] and multiagent system (MAS) [74, 110, 217–220], DRL has achieved promising success. The feasibility and transfer capabilities of DRL make it possible to address the collaborative exploration and target search problem in large-scale cluttered environments effectively. Existing DRL-based methods in drone scenarios are mostly applied to single drone tasks [16, 73, 214, 216] or scenarios with discrete action spaces and grid observation environments [74, 110]. In [73], an augmented curriculum learning approach was proposed to guide a drone through a narrow gap, but the pre-designed rules used to vary the curriculum limited the potential benefits of DRL. Wu et al. [101] proposed a Snake algorithm with DRL to search for a target using an autonomous UAV and

a global grid map, but failed to extend it to practical scenarios where only local visual observations are available.

In [110], CM3 was proposed for multiagent reinforcement learning (MARL) in a 2D space to solve a collaborative navigation problem where the positions of landmarks and goals were known to each agent. Besides, function augmentation was adopted in CM3 to bridge the agent's own state in Stage 1 and the egocentric observations in Stage 2. However, this mechanism hinders the scalability of the trained model, as a new model must be trained if the number of agents changes. [219] proposed a two-level reinforcement learning-based control method for a UAV swarm to collaboratively perform surveillance in an unknown 3D urban area. However, this approach requires that each UAV has information on the target's position and requires the environment to be constructed in grids with blocks of regular shape to enable the agents to differentiate the obstacles and the target easily. This limits the application of this method in real-world scenarios.

However, exploring a 3D unstructured environment and searching for a hidden target with a visual drone swarm is more challenging since the 3D reward space is sparse and more constraints must be taken into consideration, such as continuous actions, limited observations and the scalability of the policy model. Our work addressed this challenging CTS problem for a visual drone swarm with only local visual perception, individual state, and limited egocentric observations. We improve scalability in our policy model by introducing a relative position measurement with the shortest distance as egocentric observation in multistage learning.

## 3.3 Methodology

### 3.3.1 Problem formulation

In this section, the CTS problem is formulated into a MARL problem. The agent $n$ in this chapter refers to the visual drone. Our goal is to learn a scalable decentralized optimal shared policy $\pi_\theta^* : \boldsymbol{\pi}_{\boldsymbol{\theta}}^* = \prod_{n=1}^{N} \pi_\theta^{n,*}$ for a drone swarm (a fully cooperative team) with $N$ agents to quickly find and navigate toward a hidden target denoted by $g$ and avoid collisions (can be formulated by rewards $\mathcal{R}$). Only raw

color images, ego states, and egocentric measurements are available in the observation space $\mathcal{O}^n$. There is no prior knowledge of the target $g$, and the only connection between each drone is the range sensor, from which the drone can obtain the relative positions of other agents. During the inference process, each agent continues to take actions based on the decentralized optimal policy until the termination condition is triggered.

#### 3.3.1.1 Environment setup

To train and evaluate agents, a high-fidelity simulation environment is required in which we can add customized objects, such as obstacles, agents, and sensors, to represent the global states $\mathcal{S}$. We build our simulation environment using Unity rendering engine[1] which allows us to generalize our model into different scenarios. The simulation scenarios become increasingly complex over different training stages. However, the simulation's base environment, which contains the agent drones and objects, is largely unchanged. Fig. 3.2 shows the base environment, which consists of drone agents, obstacles, and a box target. More complex environments can be generated by increasing the size of the room, adding more obstacles, and hiding the target at corners where the drone agents can hardly find.

The target can be spawned randomly within a closed sphere set $\boldsymbol{B}(ct, rd)$, in which the drone can easily detect at first sight. $ct$ and $rd$ are the center point and the radius of the sphere, respectively. Otherwise, it will be randomly placed at a predefined hidden corner drawn from a discrete set $\boldsymbol{C}(n_{loc})$. $n_{loc}$ is the size of $\boldsymbol{C}$, i.e., the number of hidden locations chosen. For instance, we can choose these locations with $n_{loc} = 5$, namely, under the desk, behind the couch, below the cupboard, below the lamp, and above the cupboard. The probability of spawning a target at different locations is controlled by the spawning probability threshold $\epsilon \in [0, 1]$. A random value $r_{seed}$ within the interval $[0, 1)$ is generated when each episode begins. If $r_{seed}$ is greater than $\epsilon$, the target will be spawned in $\boldsymbol{B}$, otherwise, it will be hidden at a place of $\boldsymbol{C}$. The probability density function (pdf) of spawning a target can be described as:

$$Pr(target) = \begin{cases} U(\boldsymbol{B}), & r_{seed} > \epsilon \\ U(\boldsymbol{C}), & r_{seed} \leq \epsilon; \end{cases} \tag{3.1}$$

[1] https://unity.com/

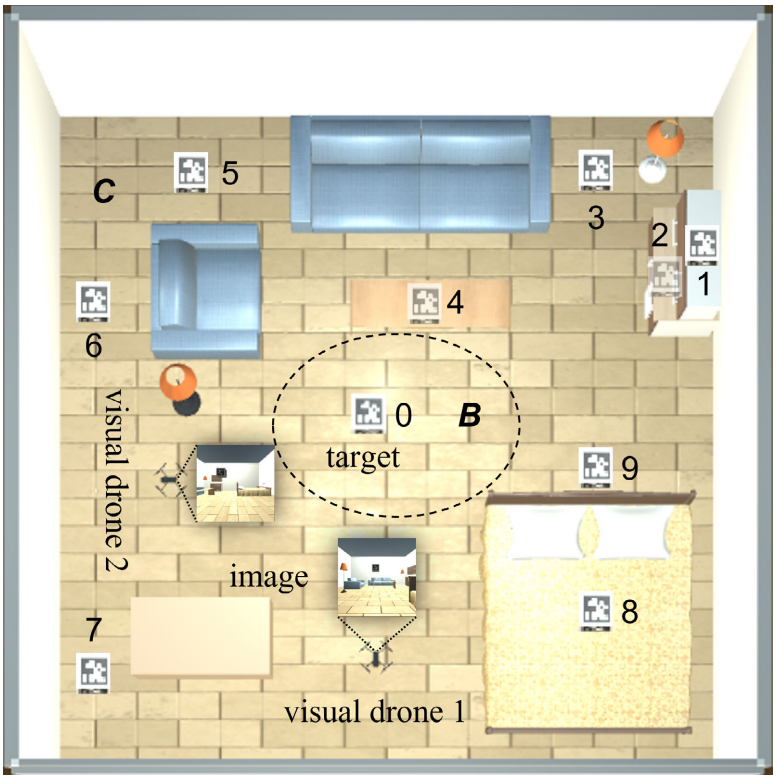



FIGURE 3.2: A bird's-eye view of the base environment, which consists of visual drone agents, obstacles, and a target. The furniture objects are the obstacles, such as sofas, the bed, and tables; The cubic textured with AprilTag is the target that drone agents need to find and reach. Each object is bounded by a collider for collision detection. Target can spawn in a closed sphere set $\boldsymbol{B}$ which is easily seen, or a discrete set $\boldsymbol{C}$ consists of several hidden locations marked with numbers $1-9$.

where $U(\cdot)$ is a uniform distribution among a set. The spawning probability threshold $\epsilon$ will be adaptively adjusted in a curriculum learning way in each stage to make the task more challenging for drone agents. The validity of spawning positions will be checked, where if the target's shape overlaps with other objects, the position is regarded as invalid, and the target will be spawned again.

The components of each obstacle are outlined by Unity's mesh collider, which allows for the detection of collisions with another collider. The drone agent is modeled as a visual quadrotor equipped with a monocular RGB camera in front of the drone. The raw image of fixed pixel size obtained from the forward camera can be used for recognizing the target and navigating towards the target. If the drone agent collides with the obstacles, other agents, or the target, a penalty (negative reward value) will be imposed.

#### 3.3.1.2 Observation space

The agent must decide which action to take based on the current observations $o_t^n$ to maximize the rewards. The drone agent receives observation information from its onboard camera and state sensors, such as IMU and positioning device. The image observation, the egostate observation, and the egocentric observation are denoted as $o_{I,t}^n$, $o_{S,t}^n$ and $o_{E,t}^n$, respectively. The RGB raw image from the camera is down-sampled to a fixed resolution $3 \times 224 \times 224$ image $I_t \in \mathbb{R}^{3\times224\times224}$ at every time step. The agent $n$ knows its own translational and rotational states, i.e., local position $\boldsymbol{p}_W^n$ and rotational quaternion $\boldsymbol{q}_{WB}^n$, and the relative positions of other team agents $\boldsymbol{p}_W^{nj}$, where $j \in [N]^{-n}$. However, only the relative position with the shortest distance $\boldsymbol{p}_W^{nj^*}$, i.e., $\|\boldsymbol{p}_W^{nj^*}\| = min(\|\boldsymbol{p}_W^{nj}\|)$ for $j \in [N]^{-n}$, is considered as the input observation of the policy model. Besides, a normalized forward direction vector $\bar{\boldsymbol{d}}_W^n \in \mathbb{R}^3$ is included in the observation space to ensure that the drone agent senses the direction. To ensure smooth maneuvering for the drone, the last action $a_{t-1}^n \in \mathbb{R}^4$ is also stored as an observation. For each drone, only the current RGB image $I_t^n$ is used and encoded into a vector $o_{EV,t}^n \in \mathbb{R}^{512}$ with a Convolutional Neural Network (CNN). This CNN allows the drone to detect different objects in their environment, such as obstacles, team agents, and the target, and sense the depth information. The concatenated observation vector for the drone $n$ at time step $t$ is $o_t^n := \left[o_{EV,t}^n, \boldsymbol{p}_{W,t}^n, \boldsymbol{q}_{WB,t}^n, \bar{\boldsymbol{d}}_{W,t}^n, a_{t-1}^n, \boldsymbol{p}_{W,t}^{nj^*}\right] \in \mathbb{R}^{529}$.

#### 3.3.1.3 Action space

In this work, we only focus on the high-level controller for drones. The low-level controller inside the drone tries to keep track of the high-level commands via inputs $\boldsymbol{u}$. Since the quadrotor is an under-actuated system, the selected high-level commands (actions $a_t^n$) are the four velocity commands generally used in the Robot Operation System (ROS), i.e., the desired velocity of the quadrotor in body frame $O_B$, namely $\hat{v}_x^B$, $\hat{v}_y^B$, $\hat{v}_z^B$ and $\hat{\omega}_z$. The action space $\mathcal{A}^n$ consists of these four continuous actions that drive the drone to fly towards the target while exploring the constrained 3D space.

#### 3.3.1.4 Reward function

The reward function consists of two parts: the existential penalty $\mathcal{R}_P = -t/T_{max}$ of the team, where $T_{max}$ is the allowable time steps in an episode, and the terminal reward $\mathcal{R}_T$. The existential penalty accelerates the exploring progress and the terminal reward serves to guide the drone to fly towards the target in the correct forward direction and to avoid crashing simultaneously.

The terminal reward is a set function described as:

$$\mathcal{R}_T = \begin{cases} +5, & if\ agent\ reaches\ the\ target \\ r_C, & if\ agent\ crashes \end{cases} \tag{3.2}$$

Where $r_C$ is the penalty when the drone collides with the obstacles or other agents.

$$r_C = -\alpha \frac{\|\boldsymbol{d}_T\|}{\|\boldsymbol{d}_{init}\|} - \frac{\beta}{\pi} \arccos \frac{\langle \boldsymbol{d}_T, \bar{\boldsymbol{d}}_W \rangle}{\|\boldsymbol{d}_T\| \cdot \|\bar{\boldsymbol{d}}_W\|} - 3 \tag{3.3}$$

where $\boldsymbol{d}_T$ is the vector from the collision position to the target position and $\boldsymbol{d}_{init}$ is the vector from the initial position to the target position; $\|\cdot\|$ denotes the norm of the vector and $\langle \cdot \rangle$ is the inner product of two vectors; $\alpha \in (0, 1]$ and $\beta \in (0, 1]$ are the weights used to adjust the penalty from the distance to fly and the penalty caused by the forward direction. Note that the terminal reward $r_C$ is accounted for each agent, and the episode terminates only when all agents crash. The positive reward of $+5$ is a team reward and distributed with an explicit **equal credit assignment**, i.e., each agent receives $+5$ if one agent reaches the target. It is important to note that $\mathcal{R}_P \geq -1$ and $min(\mathcal{R}_T) < \mathcal{R}_P < max(\mathcal{R}_T)$ to prevent the agents from taking suicidal actions without motivation to complete the task. Additionally, we enforce the condition $-r_C \leq max(\mathcal{R}_T)$ to encourage the agents to explore the environment instead of staying in a safe space. Reward scale is also an essential factor affecting the performance of DRL. According to the results from [221], a reward scale $\hat{\sigma} = 10$ is preferred since no layer normalization is used for our network. Hence, $\mathcal{R}_T \in [-10, 10]$ is considered in this work. We further verify a larger value such as "+10" does not change the performance significantly.

**Theorem 3.1** (Existence of Optimal Policy)**.** *With multiagent equal credit assignment, the optimal shared policy $\pi^*(a_t^n \mid o_t^n)$ for CTS exists and can be achieved along the improvement of local critic $Q_{\boldsymbol{\pi}}^n(s_0, a_0^n) := \mathbb{E}_{\boldsymbol{\pi}} \left[ \sum_{t=0}^{t=T} \gamma^t \mathcal{R}_t^n \mid s_0, a_0^n \right]$ conditional on the joint action $\boldsymbol{a}_t$ and the local observations $\{o_t^n\}$.*

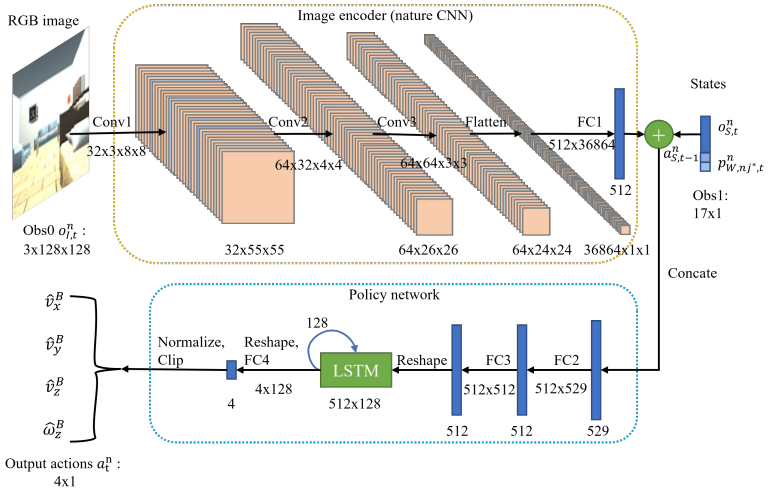



FIGURE 3.3: Illustration of the neural network architecture for target search, which consists of the image encoder, the policy network, and the memory network.

*Proof.* With equal credit assignment, the CTS becomes a fully cooperative game, i.e., the reward function of each agent is the same $\mathcal{R}^1 = \cdots = \mathcal{R}^N$. For each agent $n$, accumulate the joint probability over $a^{-n}$ and we have the Q-function as follows,

$$\begin{aligned} Q^n_{\boldsymbol{\pi}}(s_0, a^n_0) &= \sum_{a^{-n}} \pi(a^{-n} \mid s) Q^n_{\boldsymbol{\pi}}(s_0, a^n_0, a^{-n}_0) \\ &= \sum_{a^{-n}} \pi(a^{-n} \mid s) Q^n_{\boldsymbol{\pi}}(s_0, \boldsymbol{a}_0) \end{aligned} \tag{3.4}$$

The objective function (2.9) can then be described with

$$\begin{aligned} \mathcal{J}(\boldsymbol{\pi}_{\boldsymbol{\theta}}) &= \sum_{n=1}^{n=N} \mathbb{E}_{\boldsymbol{\pi}} \left[ \sum_{t=0}^{t=T} \gamma^t \mathcal{R}^n_t \mid s_0 \right] \\ &= \sum_{n=1}^{n=N} V^n_{\boldsymbol{\pi}}(s_0) \\ &= \sum_{n=1}^{n=N} \mathbb{E}_{\boldsymbol{a} \sim \boldsymbol{\pi}(\boldsymbol{a}|s)} Q^n_{\boldsymbol{\pi}}(s_0, \boldsymbol{a}_0) \\ &= \sum_{n=1}^{n=N} \mathbb{E}_{a^n \sim \pi(a^n|o^n)} \sum_{a^{-n}} \pi(a^{-n} \mid s) Q^n_{\boldsymbol{\pi}}(s_0, \boldsymbol{a}_0) \\ &= \sum_{n=1}^{n=N} \mathbb{E}_{a^n \sim \pi(a^n|o^n)} Q^n_{\boldsymbol{\pi}}(s_0, a^n_0) \end{aligned} \tag{3.5}$$

According to **Proposition 1** in [110], we have a Bellman function $Q_{\boldsymbol{\pi}}^n(s_t, a_t^n) = \mathbb{E}_{\boldsymbol{\pi}}\left[\mathcal{R}_t^n + \gamma Q_{\boldsymbol{\pi}}^n(s_{t+1}, a_{t+1}^n)\right]$. Hence, the objective function can be further expanded as

$$\begin{aligned}\mathcal{J}(\boldsymbol{\pi_\theta}) &= \sum_{n=1}^{n=N} \mathbb{E}_{a^n\sim\pi(a^n|o^n)}\left[\mathbb{E}_{\boldsymbol{\pi}}\left[\mathcal{R}_0^n + \gamma Q_{\boldsymbol{\pi}}^n(s_1, a_1^n)\right]\right] \\ &= \sum_{n=1}^{n=N} \mathbb{E}_{\boldsymbol{\pi}}\mathcal{R}_0^n + \sum_{n=1}^{n=N} \mathbb{E}_{a^n\sim\pi(a^n|o^n)}\left[\mathbb{E}_{\boldsymbol{\pi}}\left[\gamma Q_{\boldsymbol{\pi}}^n(s_1, a_1^n)\right]\right] \\ &= N\,\mathbb{E}_{\boldsymbol{\pi}}\mathcal{R}_0^n + \sum_{n=1}^{n=N} \mathbb{E}_{a^n\sim\pi(a^n|o^n)}\left[\mathbb{E}_{\boldsymbol{\pi}}\left[\gamma Q_{\boldsymbol{\pi}}^n(s_1, a_1^n)\right]\right]\end{aligned} \tag{3.6}$$

Hence, at each episode $t$, we select the greedy policy $\pi'(a^n \mid o^n) := \text{argmax}_a\, Q_{\boldsymbol{\pi}}^n(s_t, a_t^n)$ for agent $n$ and the shared policy $\pi'$ is the optimal policy satisfying (2.10). □

### 3.3.2 Network architecture

As shown in Fig. 3.3, the neural network used in this chapter consists of three parts: the image encoder, the policy network, and the memory network. The memory network is embedded within the policy network for processing temporal sequence observations. We have conducted performance analysis on 3 visual encoders: a simple 2-layer CNN, a nature CNN [151], and a ResNet [222]. After considering efficiency and search performance, a shallow neural network-nature CNN is used in our model. The image encoder-nature CNN consists of 3 convolutional layers, 1 flatten layer, and 1 fully connected (FC) layer. A Leaky Rectified Linear Unit (Leaky ReLU) is adopted as the activation function for all convolutional layers and the encoder's FC layer. The image encoder converts the RGB image of fixed resolution into a 512-dimensional flattened vector. The policy neural network is designed with 2 FC (512-nodes with a Sigmoid activation function) layers which map the concatenated observation vector $o_t^n \in \mathbb{R}^{529}$ into a continuous action vector $a_t^n \in \mathbb{R}^4$. The memory network-Long Short Term Memory (LSTM) is a recurrent neural network used to feed previous outputs back and generate smooth trajectories for drones.

### 3.3.3 Adaptive curriculum embedded multistage learning

#### 3.3.3.1 Adaptive embedded curriculum (AEC)

In the standard curriculum learning methods [72–74], the curriculum is manually designed with a static, discrete parameter sequence in which the task difficulty level (TDL) is progressively increased with a pre-designed mode. Even though the pre-designed curriculum parameter can be post-adjusted according to the training results, obtaining a satisfactory curriculum can be inconvenient and inefficient. Hence, it is hard for agents to achieve their best performance. In contrast to standard curriculum learning methods, our proposed AEC aims to explore agents' proficiency capability, where the TDL is adaptively adjusted at the beginning of each episode with a changing rate $\delta_\epsilon > 0$ based on the SR of finding the target achieved by drone agents. If the SR rises to a high boundary $sr_h$, the TDL increases. Correspondingly, if the SR drops to a low boundary $sr_l$, the TDL decreases. The principle behind is that the higher TDL of environment will try to decrease the SR of the team while the drone team is learning to obtain a higher SR. As such, the trained policy of the drone team and the TDL of environment will converge to an equilibrium with this AEC algorithm. This equilibrium provides the optimal policy for the drone swarm to search for the target.

There are several environment parameters that can be considered to derive the TDL, such as the size of the space, the number of obstacles, the size of the target, and the spawning probability threshold ($\epsilon$) in (3.1). In our scenario, the TDL is formulated by the spawning probability threshold since our main goal is to find the hidden target as fast as possible. Hence, in the following discussion, the TDL is denoted by $\epsilon$ and clipped within $[\epsilon_{min}, \epsilon_{max}]$. Other environment parameters can be randomly selected in the domain randomization to improve the generalization of the trained policy. In this chapter, we provide two ways to calculate the success rate. One is the cumulative success rate $sr = (success\ episodes)/(all\ episodes)$; another is the moving success rate $sr = (success\ episodes\ in\ a\ sliding\ window)/wl$, where $wl$ is the sliding window length. The AEC with a cumulative success rate is described in Algorithm 1 and the AEC with a moving success rate is presented in Algorithm 2.

**Theorem 3.2** (Convergence Analysis)**.** *If (i) the policy gradient algorithm used for policy improvement converges to the optimal policy in any fixed TDL* $\epsilon \in [\epsilon_{min}, \epsilon_{max}]$

**Algorithm 1:** Adaptive Embedded Curriculum (AEC)

**Input:** An initial TDL $\epsilon_0$, changing rate $\delta_\epsilon$, $sr_l$, $sr_h$
**Output:** The spawning position $\mathbf{p}_{goal}$ of target
On the start of training:
Initialize $\boldsymbol{p}_{W,0}$, $\boldsymbol{v}_{W,0}$, $\boldsymbol{q}_{WB,0}$, $SR = 0.0$, $successCount = 0$, $episodeCount = 0$ ;
Construct the critic Q-function $Q^n_{\boldsymbol{\pi_\theta}}(s_0, a^n_0)$;
**for** *episode* $= 0$ *: maximum episodes* **do**
  On each episode begin:
  **if** $episodeCount \neq 0$ **then**
    $SR \leftarrow successCount/episodeCount$;
  $episodeCount \leftarrow episodeCount + 1$;
  **if** $SR > sr_h$ **then**
    $\epsilon \leftarrow max(\epsilon + \delta_\epsilon, \epsilon_{max})$;
  **if** $SR < sr_l$ **then**
    $\epsilon \leftarrow min(\epsilon - \delta_\epsilon, \epsilon_{min})$;
  Spawn the target according to (3.1);
  **for** *step* $t = 0 : T$ **do**
    Conduct policy improvement, for $t = 0$ to $T$;
    **if** *one drone reaches the target* **then**
      $successCount \leftarrow successCount + 1$;
      end the episode;
    **if** *all drones crash* **then**
      end the episode;
      back to 4;

*and (ii) the success rate sr is reciprocal correlated with $\epsilon$, then AEC converges to the optimal policy, and the TDL converges to $\epsilon^*$ for any $\epsilon_0 > 0$ and $sr_h \geq sr_l$ with finite training steps $T'$.*

*Proof.* WLOG, let assume $sr = k/\epsilon + c$, with $k > 0, c \geq 0$. We have $\epsilon \leftarrow max(\epsilon + \delta_\epsilon, \epsilon_{max})$ when $sr > sr_h$, i.e., $k/\epsilon + c > sr_h$. That is, when $\epsilon < k/(sr_h - c)$, $\epsilon \leftarrow max(\epsilon + \delta_\epsilon, \epsilon_{max})$. Therefore, $\forall \epsilon_0 < k/(sr_h - c), \exists T'$ s.t. $\epsilon^* = (\epsilon_0 + T'\delta_\epsilon) \geq k/(sr_h - c)$. Similarly, when $sr < sr_l$, i.e. $\epsilon > k/(sr_l - c)$, $\epsilon \leftarrow min(\epsilon - \delta_\epsilon, \epsilon_{min})$. Hence, $\forall \epsilon_0 > k/(sr_l - c), \exists T'$ s.t. $\epsilon^* = (\epsilon_0 - T'\delta_\epsilon) \leq k/(sr_l - c)$. Combine together, $\forall \epsilon_0 > 0, \exists T'$ s.t. $\epsilon$ converges to $\epsilon^* \in [k/(sr_h - c), k/(sr_l - c)] \cap [\epsilon_{min}, \epsilon_{max}]$.

Since the policy converges to the optimal policy $\pi^*(a^n \mid o^n, \epsilon)$ given any $\epsilon$, according to the policy improvement theorem, the AEC converges to the optimal policy $\pi^*(a^n \mid o^n, \epsilon^*)$. This completes the proof. □

*Remark* 3.1. The converge episodes required depend on the changing rate $\delta_\epsilon$ and the sliding window length $wl$. With a higher value of $\delta_\epsilon$ and a smaller $wl$, the convergence progress is more likely to fluctuate.

**Algorithm 2:** AEC with Sliding Window (AEC-SW)

**Input:** An initial TDL $\epsilon_0$, changing rate $\delta_\epsilon$, $sr_l$, $sr_h$
**Output:** The spawning position $\mathbf{p}_{goal}$ of target
On the start of training:
Initialize $\boldsymbol{p}_{W,0}$, $\boldsymbol{v}_{W,0}$, $\boldsymbol{q}_{WB,0}$, $movingSR = 0.0$, $slidingWindow = \mathbf{0}_{[wl]}$, $episodeCount = 0$ ;
Construct the critic Q-function $Q^n_{\boldsymbol{\varphi}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}}}(s_0, a^n_0)$;
**for** *episode* $= 0$ *: maximum episodes* **do**
  On each episode begin:
  **if** $episodeCount \neq 0$ **then**
    $movingSR \leftarrow mean(slidingWindow)$;
  $index \leftarrow episodeCount \mod wl$;
  $slidingWindow[index] \leftarrow 0$;
  $episodeCount \leftarrow episodeCount + 1$;
  **if** $movingSR > sr_h$ **then**
    $\epsilon \leftarrow max(\epsilon + \delta_\epsilon, \epsilon_{max})$;
  **if** $movingSR < sr_l$ **then**
    $\epsilon \leftarrow min(\epsilon - \delta_\epsilon, \epsilon_{min})$;
  Spawn the target according to (3.1);
  **for** *step* $t = 0 : T$ **do**
    Conduct policy improvement;
    **if** *one drone reaches the target* **then**
      $slidingWindow[index] \leftarrow 1$;
      end the episode;
    **if** *all drones crash* **then**
      end the episode;
      back to 4;

#### 3.3.3.2 Stage 1

At Stage 1, the objective is to train a single drone to identify, track and fly towards a randomly spawned target according to Algorithm 1 or 2. During Stage 1, the drone explores the environment with the designed reward function. The MARL problem is reduced into a single agent ($N = 1$) problem, and the group reward becomes the agent $n$'s reward. The optimal policy achieved by agent $n$ is the greedy policy

[110]. The drone will learn to find and approach the stationary target without crashing into obstacles.

#### 3.3.3.3 Stage 2

Once the drone has successfully learned the target-approaching behavior, another drone is added to the scene in Stage 2 for generating collaborative behavior. The trained model in Stage 1 is used to initialize the model in Stage 2. These two drones are further trained to search for the same target without colliding with each other. Once the agents crash (collide with the obstacles or other agents), their status becomes inactive until the next episode. The episode ends when all agents crash or one agent reaches the target.

#### 3.3.3.4 Training algorithm

The Proximal Policy Optimization (PPO) algorithm [111] is selected for our policy training according to Theorem 3.2 even though the state-of-the-art (SOTA) off-policy algorithm Soft Actor-Critic (SAC) [223] performs more data-efficient than PPO over general RL tasks especially in static environments, such as the Humanoid benchmark task. SAC adopts an entropy maximization term into the actor-critic framework to trade off the exploit-exploration and to improve the data efficiency via experience replay. However, from the perspective of an individual agent, our CTS scenarios have non-stationary states such as the positions of teammates and the target are always changing. This can cause significant divergence issues in the experience replay mechanism utilized by SAC, as indicated in [224] and the overview page of Unity ML-Agents Toolkit [158]. Thus, PPO is preferred for our application since it is more effective and stable for problems with dynamic environments by using a "clipped" surrogate objective and on-policy learning. We further compared the performance of SAC and PPO in our Stage 1 training in Section 3.4 to validate the advantages of PPO in our application.

In our training process, only the states of the agents and the raw image are collected as observations, without the need for information on relative positions between the agents and the target, which is usually the case for the traditional numerical methods[206]. Only at the end of each episode are the distance values counted in

TABLE 3.1: Domain randomization

| **Environment Randomization** | |
|---|---|
| Light intensity $I_v$ | $U(0.2, 1.0)$ |
| Scale of target $\lambda_{target}$ | $U(0.2m, 0.3m)$ |
| Yaw angle of target $\psi_{target}$ | $U(0°, 360°)$ |
| **Agent Randomization** | |
| Position noise | $U(\boldsymbol{B}(ci, 0.2m))$ |
| Noise in yaw angle of agent | $U(-30°, 30°)$ |

the reward function. Meanwhile, the existential penalty is designed to encourage the agent to reach the target as fast as possible.

### 3.3.4 Domain randomization

Domain randomization [163] is an effective technique to improve the generalization capability of the model during training and hence increase the success rate of Sim2Real [163] transfer to unseen scenarios. At the beginning of each episode, environment parameters and agents' states are randomly sampled from uniform distributions. In our work, the domain randomization used can be divided into (1) environment randomization that involves randomization of parameters, such as light intensity in Unity scenes $I_v$, the scale $\lambda_{target}$, and the yaw angle $\psi_{target}$ of the target; (2) agent randomization that involves randomization of parameters, such as the initial states of agents. The environment randomization extends the trained model to accomplish the tasks in unseen scenarios while the agent randomization enables the agents to start with perturbations in their initial configurations.

As listed in Table 3.1, the environment parameters are randomly sampled from a uniform distribution $U(min, max)$, while the agent randomization is generated by adding a uniformly distributed random noise $U(\cdot)$. Note that the agent position noise is sampled from a uniformly distributed closed sphere set $\boldsymbol{B}(ci, ri)$, i.e., $U(\boldsymbol{B}(ci, ri))$, where $ci$ is the initial position of agents and $ri$ is the radius of sphere set. $U(min, max)$ and $U(\boldsymbol{B}(ci, ri))$ are generated via the $Random.Range(min, max)$ function and $Random.insideUnitSphere * ri$ operator in Unity.

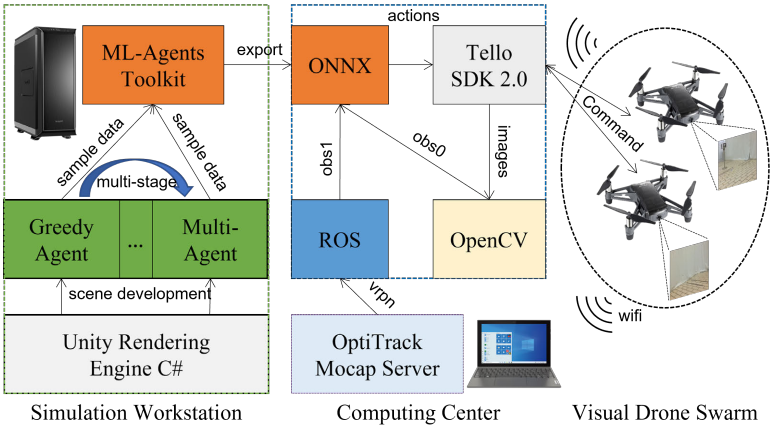



FIGURE 3.4: Illustration of the training and experiment framework.

## 3.4 Experiments and results

In this section, we present the simulation experiments, including the training process and inference tests, and the physical experiments covering the model Sim2Real transfer and the real-time flight tests. The architecture of simulation and physical experiment platform is illustrated in Fig. 3.4, which consists of a simulation workstation (with AMD Ryzen 9 5900X 12 cores CPU, Nvidia RTX 3090 Gaming OC 24GB GPU, and 32GB RAM), a laptop computing center, the OptiTrack motion capture server streaming the position data of selected rigid bodies and a Tello Edu visual drone swarm.

The simulation workstation is used to develop simulation environments for training and testing and export the trained neural network model for deployment. With the trained ONNX[2] model, the computing center acquires real-time observations from the motion capture server and Tello Edu drones, and generates corresponding actions which drive the Tello Edu visual drones to search for the target via Tello SDK 2.0[3]. The computing center is connected to the OptiTrack motion capture system to subscribe to the position data of drones via Robots Operation System (ROS) nodes. The image observation is acquired from the onboard camera of Tello Edu drones using necessary image processing. Note that a Tello Edu drone can only be set as a wireless access point (AP) to stream the video. Hence, multiple USB WIFI adapters are used in the computing center for connecting to each drone separately using WIFI.

[2]https://onnx.ai/
[3]https://www.ryzerobotics.com/tello-edu/downloads

TABLE 3.2: Parameters used for the AEC

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Environment size | $5 \times 5 \times 3(m^3)$ | Initial TDL $\epsilon_0$ | 0.1 |
| Minimum TDL $\epsilon_{min}$ | 0.1 | Maximum TDL $\epsilon_{max}$ | 0.9 |
| TDL changing rate $\delta_\epsilon$ | 0.1 | High SR boundary $sr_h$ | 0.85 |
| Low SR boundary $sr_l$ | 0.40 | Maximum step $T_{max}$ | 5000 |
| Reward weight $\alpha$ | 1.0 | Reward weight $\beta$ | 0.1 |
| Maximum speed $v_{max}$ | 1.0 m/s | Window length $wl$ | 1000 |

## 3.4.1 Simulation experiments

### 3.4.1.1 Settings

Our simulation platform is developed based on the Unity rendering engine and ML-Agents Toolkit [158], a flexible simulation and training platform for multiagent reinforcement learning. Scenes of target search with a single greedy agent and CTS with a drone swarm are developed successively. The dimensions of the base simulation environment in Fig. 3.2 is $5m \times 5m \times 3m$ (width $\times$ length $\times$ height). In the AEC, the TDL starts from $\epsilon_0 = 0.1$ and changes with $\delta_\epsilon = 0.1$. The TDL is clipped within $[0.1, 0.9]$, not its domain $[0.0, 1.0]$ during the training process since we aim to avoid the situation where the policy converges to only searching the set $B$ at the beginning. The SR boundaries are set to $sr_h = 0.85$ and $sr_l = 0.40$, which are the expected SR and the allowable lowest SR for our task, respectively. The allowed maximum time step is $T_{max} = 5000$ in one episode. The reward weights are set to $\alpha = 1$ and $\beta = 0.1$ as we focus more on the distance to reach during training. The initial linear and angular velocity of drones are reset to zero at the start of each episode. The parameters of AEC used in our simulation are concluded in Table 3.2. These parameters are tried and adjusted in a reasonable range to reach the near-best performance.

### 3.4.1.2 Baselines and ablations

The performance of our approach is compared against SOTA methods: (1) TD-A3C [17], which only uses visual observations; (2) CM3 [110], which is trained via two-stage learning but without AEC (with a fixed TDL $\epsilon = 0.3$). We ensure that TD-A3C and CM3 are modified so that both can be applied for the target search tasks

TABLE 3.3: Hyperparameters used for training

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| Batch size | 2048 | Buffer size | 10240 |
| Learning rate $\eta$ | 0.0003 | Discount factor $\gamma$ | 0.99 |
| Number of epochs | 3 | Learning rate schedule | linear |
| Checkpoints | 10 | Time horizon | 128 |
| **PPO** | | **SAC** | |
| Entropy bonus beta | 0.01 | Interpolation factor $\tau$ | 0.005 |
| Clip threshold $\epsilon^{ppo}$ | 0.2 | Update steps | 10.0 |
| Regularization factor $\lambda^{ppo}$ | 0.95 | Initial entropy coefficient | 0.5 |
| Optimizer | Adam | Replay size | 1000 |
| Maximum episodes | 12 million | Maximum episodes | 3 million |

in the training. Since TD-A3C is proposed to address a single-agent navigation problem, we evaluate its performance with our PPO with AEC (PPO-AEC) and PPO without AEC (PPO-w/o-AEC) in the single-agent training. PPO-AEC and PPO-w/o-AEC are compared to examine the contribution of AEC to our approach. Meanwhile, PPO-AEC-SW is compared with PPO-AEC to investigate the effect of the sliding window scheme. To validate the advantages of PPO over SAC, PPO-AEC is compared with SAC-AEC (single agent trained with SAC algorithm) regarding the training runtime and task performance. Note that the SAC-AEC is computationally expensive; thus we only consider 3 million training episodes.

To investigate the data efficiency of the proposed multistage scheme, fully trained Stage 2 without pre-trained policy ("Direct" method) is compared with CM3 and ACEMSL in the multiagent training. Since we only compare the performance of ACEMSL and CM3 in Stage 2, Stage 1 of ACEMSL and CM3 are the same, when trained for 3 million episodes with the AEC. Note that for fair evaluation, we also trained TD-A3C and CM3 with the PPO algorithm since we consider the method frameworks, not the training algorithm itself. The hyperparameters of PPO for all training remain the same.

#### 3.4.1.3 Training

The training hyperparameters are determined and fine-tuned based on the suggestions and results from [221, 225] and Unity ML-Agents Toolkit [158], such as batch size, learning rate, and epochs. These hyperparameters are listed in Table 3.3. The

versions of used training tools are ml-agents-toolkit: 0.27.0, ml-agents-envs: 0.27.0, communicator API: 1.5.0, and PyTorch: 1.8.2+cu11.1. To accelerate the training process, 6 copies of the environment are placed in one scene, and 3 concurrent Unity instances (in total 18 parallel environments) are invoked at the start of the training.

The mean and the standard deviation values over 3 runs of the single-agent training are illustrated in Fig. 3.5. As shown in Fig. 3.5, our PPO-AEC achieves the best performance even at a higher TDL ($\epsilon = 0.48$) compared to PPO-w/o-AEC and TD-A3C based on the convergent cumulative SR (80%) and the episode length (350). PPO-AEC achieves higher cumulative rewards (3.5) compared to PPO-w/o-AEC with the same TDL ($\epsilon$ =0.3), while also achieving the same rewards with a higher TDL. Our PPO-AEC-SW converges faster to a higher TDL ($\epsilon = 0.8$) with the help of a sliding window but induces a larger oscillation during training. It verifies the data efficiency and effectiveness of the proposed AEC. TD-A3C and SAC-AEC cannot accomplish the search task as the SR is below 10% even with the lowest TDL ($\epsilon = 0.1$). SAC-AEC obtains the worst performance as the policy diverges without improvement. Agent trained with TD-A3C can hardly find the target, illustrating the drawbacks of targeted-driven navigation with purely visual perception for target search.

For the multiagent training, Stage 2 training is initialized from the existing model of Stage 1 trained over 3 million episodes with PPO-AEC. Stage 1 training curves are shown in Fig. 3.6. The mean values and the standard deviation over 3 runs of the multiagent training are illustrated in Fig. 3.7. With the help of Stage 1, ACEMSL and CM3 converge more than **1 million episodes** (total 3.1 million episodes: 3 million in Stage 1 and 0.1 million in Stage 2) faster than the "Direct" method (total 4.1 million episodes) as indicated in Fig. 3.7 (a), which demonstrates the significant data efficiency of the multistage learning. Compared to CM3, ACEMSL benefits significantly from the AEC algorithm, as ACEMSL achieves a higher cumulative reward and SR, even with higher TDLs (see Fig. 3.7 (c)). The AEC can fully liberate the advantages of multistage learning for CTS tasks with a visual drone swarm.

The proposed ACEMSL effectively addresses the challenges of 3D sparse continuous space exploration and the requirement for collaborative behavior in multiagent

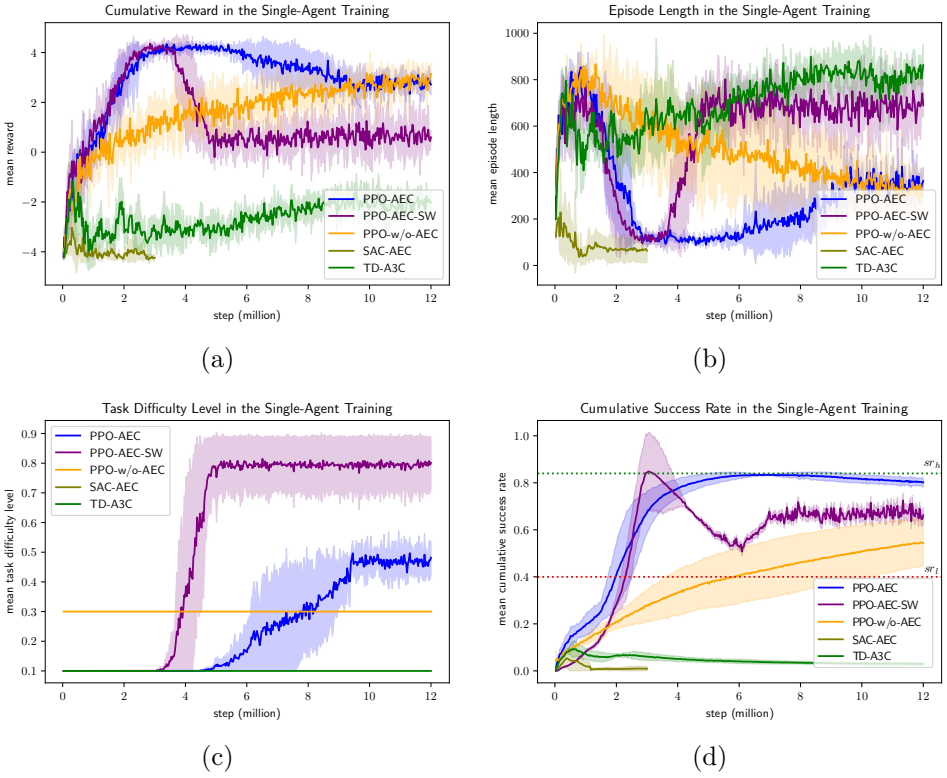



FIGURE 3.5: Learning curves of the PPO-AEC (ours), PPO-AEC-SW (ours), the baseline TD-A3C and the ablation PPO-w/o-AEC, SAC-AEC in the single-agent training. Mean and standard deviation (shaded) over 3 independent runs conducted every 12 million training steps (episodes). Note that due to the high computational cost of SAC-AEC, we only consider 3 million training episodes for it. The moving SR is illustrated for PPO-AEC-SW in (d). All these values are the means over $3 \times 6 = 18$ training environments. From (c) and (d), for PPO-AEC, when the cumulative SR crosses the $sr_h = 0.85$ (4.5 million steps), the TDL increases with $\delta_\epsilon = 0.1$ and converges to 0.48 (9.4 million steps) with the SR of 80%. (a) Cumulative reward; (b) Episode length; (c) Task difficulty level; (d) Cumulative success rate.

systems, as demonstrated in both single-agent and multiagent training. By combining the advantages of adaptive curriculum learning and multistage learning, our method significantly improves data efficiency in the training process. Table 3.4 lists the absolute training runtime for each of the algorithms. Note that Stage 1 training time is included in the CM3 and ACEMSL results. Our PPO-AEC algorithm for single-agent training and ACEMSL approach for multiagent training cost average $(5807 \pm 80)$ $s/(million\ episode)$ and $(5732 \pm 87)$ $s/(million\ episode)$, respectively. From the obtained results, our ACEMSL framework outperforms CM3 in terms of the average training runtime metric. Compared to the on-policy PPO algorithm, the off-policy SAC algorithm takes an average $(305261 \pm 23614)$ $s/(million\ episode)$ during training due to the memory replay mechanism, which is not suitable for our

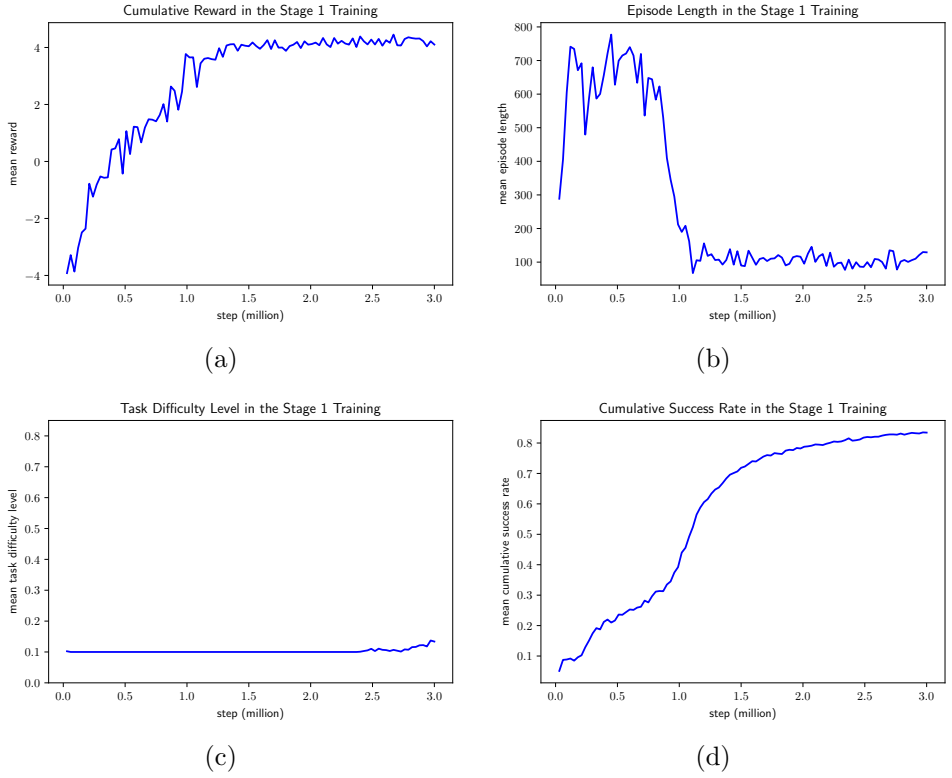



FIGURE 3.6: Stage 1 training curves with single-agent over 3 million steps (episodes). Note that the same initial policy is used for both ACEMSL and CM3 in the Stage 2 training. (a) Cumulative reward; (b) Episode length; (c) Task difficulty level; (d) Cumulative success rate.

applications. The converge training runtime of ACEMSL can be easily obtained from $(5732 \pm 87)\ s/(million\ episode) \times 4.1\ million\ episodes = (23501.2 \pm 356.7)\ s \approx 6.53\ hours$. From the analysis, we can conclude that ACEMSL can be applied to a large-scale drone swarm without much computational time cost.

#### 3.4.1.4 Validation

To further validate the better performance of the proposed ACEMSL approach, we conducted inference experiments with trained policies in scenarios with different TDLs, namely $\epsilon = 0.3, \epsilon = 0.5, \epsilon = 0.7, \epsilon = 0.9$, at the predetermined initial position. The room dimension was the same as the training environment, i.e., $5 \times 5 \times 3\ (m^3)$. The SR and the average TTR were calculated over 500 episodes for each approach. The results (see Table 3.5) show that our proposed approaches achieve the best performance over all test TDLs in the single-agent and two-agent scenarios, respectively, in terms of SR. The use of the AEC-SW algorithm leads

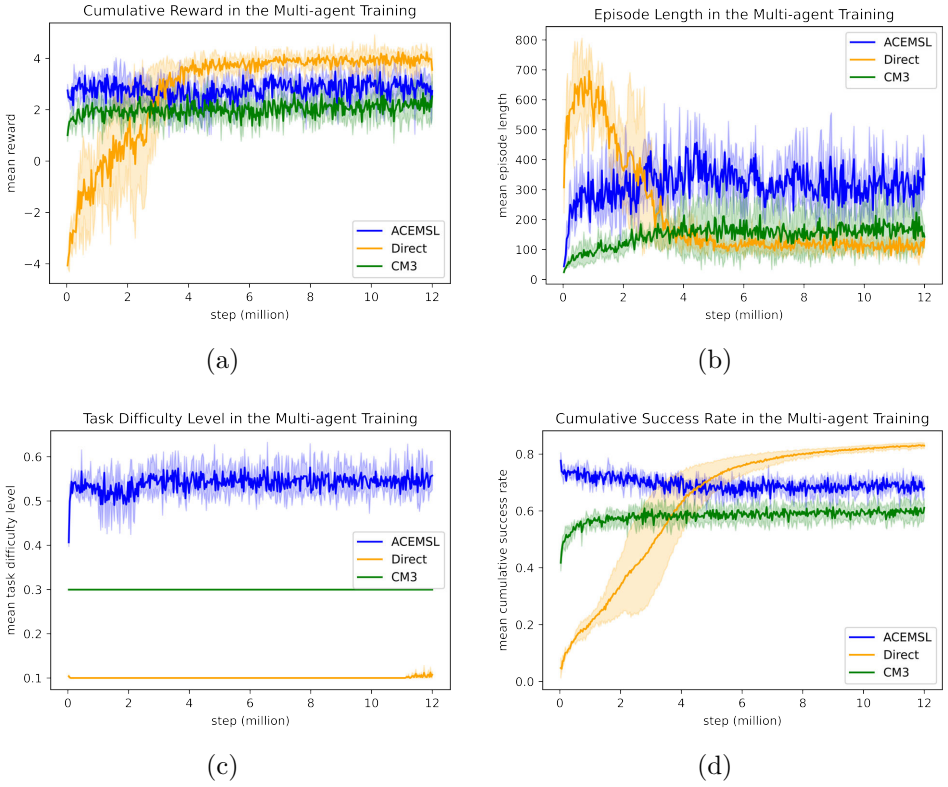


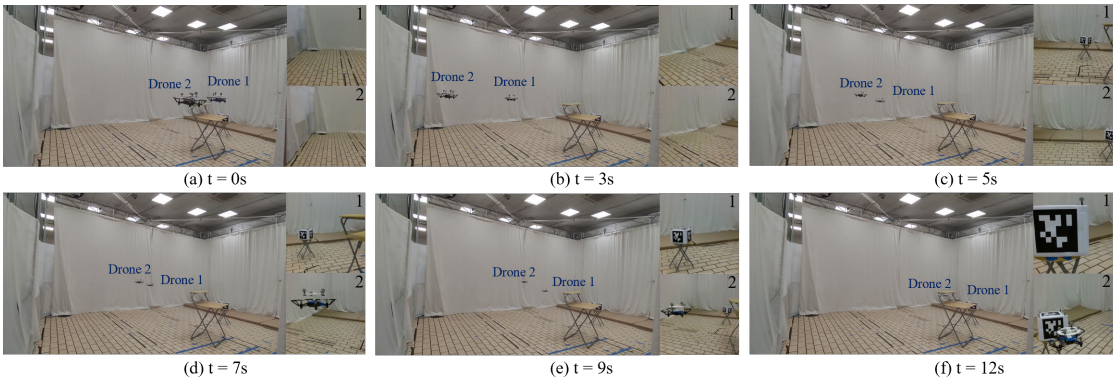

FIGURE 3.7: Learning curves of the ACEMSL (ours), the baseline CM3 and the ablation "Direct" in the multiagent training. Mean and standard deviation (shaded) over 3 independent runs conducted every 12 million training steps (episodes). (a) Cumulative reward; (b) Episode length; (c) Task difficulty level; (d) Cumulative success rate.

TABLE 3.4: Absolute training runtime of different algorithms (seconds)

| Algorithms | Episodes (million) | Absolute Runtime (s) | Average Runtime (s/million episode) |
|---|---|---|---|
| TD-A3C[17] | 12 | 61023 ± 520 | **5085 ± 43** |
| PPO-w/o-AEC [111] | 12 | 66573 ± 1727 | 5548 ± 144 |
| SAC-AEC | 3 | 915783 ± 70841 | 305261 ± 23614 |
| **PPO-AEC (ours)** | 12 | 69689 ± 958 | 5807 ± 80 |
| **PPO-AEC-SW (ours)** | 12 | 67186 ± 361 | 5599 ± 30 |
| CM3[110] | 15 | 100724 ± 2580 | 6715 ± 172 |
| Direct | 12 | 71893 ± 1320 | 5991 ± 110 |
| **ACEMSL (ours)** | 15 | 85985 ± 1298 | **5732 ± 87** |

TABLE 3.5: Performance (SR: % / TTR: Avg (steps)) with various task difficulty levels (TDLs) in simulation

| Algorithms / TDL ($\epsilon$) | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|
| TD-A3C[17] | 5.4 / 2052 | 3.8 / 1434 | 0.6 / 1800 | 0.0 / N.A. |
| PPO-w/o-AEC[111] | 57.1 / 129 | 43.9 / 142 | 39.3 / 468 | 25.1 / 829 |
| SAC-AEC | 0.4 / 255 | 0.4 / 77 | 0.4 / 154 | 0.0 / N.A. |
| **PPO-AEC (ours)** | 70.5 / 180 | 52.9 / 240 | **48.7** / 295 | 31.7 / 322 |
| **PPO-AEC-SW (ours)** | **74.7** / 158 | **63.7** / 192 | 47.1 / 208 | **32.7** / 312 |
| CM3[110] | 74.7 / 154 | 64.3 / 216 | 50.1 / 232 | 37.3 / 384 |
| Direct | 69.1 / 244 | 52.1 / 333 | 35.3 / 357 | 22.6 / 522 |
| **ACEMSL (ours)** | **79.3 / 107** | **66.5 / 150** | **51.5 / 182** | **38.5 / 297** |




(a) t = 0s (b) t = 3s (c) t = 5s

(d) t = 7s (e) t = 9s (f) t = 12s

FIGURE 3.8: The snapshots of the visual drone swarm performing the CTS. The images perceived by the cameras of Drone 1 and Drone 2 are labeled as 1 and 2, respectively at the top right side of each snapshot.

to improved performance compared to the AEC. Compared to PPO-AEC-SW, ACEMSL demonstrates that collaborative search significantly improves over single-agent search in terms of SR and TTR. While TD-A3C allows agents to avoid obstacles with a longer search time, it fails to find the target. SAC-AEC, on the other hand, fails to learn even an obstacle avoidance policy and only randomly touches the target with the least TTR as it diverges during training, as discussed in Section 3.3.3.4. The pre-trained policy in Stage 1 also leads to better performance in Stage 2, as seen with the CM3 method outperforming the "Direct" method.

TABLE 3.6: Performance of ACEMSL (SR: % / TTR: Avg. (steps)) with various room dimensions in simulation

| Room Dimensions ($m$) | $5 \times 5 \times 3$ | $8 \times 8 \times 3$ | $10 \times 10 \times 3$ |
|---|---|---|---|
| 1 drone | 75.4% / 162 | 52.8% / 301 | 16.0% / 325 |
| 2 drones | **79.3%** / 107 | 61.1% / 236 | 35.9% / 256 |
| 3 drones | 63.8% / **104** | **62.0% / 178** | **41.9 % / 254** |

#### 3.4.1.5 Testing

We tested our models from single-agent and multiagent training on various room dimensions with 500 episodes evaluated per experiment, where the TDL is set as $\epsilon = 0.3$. We also demonstrated the scalability of the trained model with ACEMSL, where multiple drones are spawned in the start area. The SR and the mean TTR with different number of drones on various room dimensions are listed in Table 3.6. As the room dimensions increase, the SR drops, and the required time steps to reach the target increase. On the other hand, deploying more drones helps increase the SR and reduce the required time steps to reach the target, which illustrates the advantages of using a visual drone swarm for target search. However, the scale of the swarm is constrained by the size of the room (the SR drops from 75.4% to 63.8% in Table 3.6 when 3 drones are deployed in the room of $5m \times 5m \times 3m$) since there is a higher chance of the drones crashing into one another if too many drones are flying within a small room.

### 3.4.2 Physical experiments

To evaluate the trained models' generalization and Sim2Real capabilities, we transfer our models to an unseen large indoor environment without fine-tuning, and their performance on different devices is compared. The test environment is an enclosed room with the size of $8m \times 7m \times 4m$. Each side, excluding the floor and ceiling of the environment, is covered with white curtains. The target is a parcel box with the size of $0.2m \times 0.3m \times 0.2m$, placed randomly in/behind obstacles within two environments in several test positions such as edges and corners to control the TDL near $\epsilon = 0.5$ (Env 1, fewer obstacles) or $\epsilon = 1.0$ (Env 2, all choosing from the set $U(\boldsymbol{C})$ with more obstacles; for detailed locations, see the project website). Each

TABLE 3.7: Computing performance (Avg. ± Std.) of devices

| Devices | Raspberry Pi | Xavier NX | Laptop |
|---|---|---|---|
| Program Component | | | |
| Image Processing (ms) | 85.1 ± 3.8 | 35.9 ± 0.6 | 16.1 ± 0.6 |
| ONNX Model (ms) | 48.2 ± 2.8 | 10.5 ± 0.4 | 4.47 ± 0.49 |
| Control Commands (ms) | 3.08 ± 0.75 | 1.62 ± 0.15 | 0.35 ± 0.03 |
| All Process | | | |
| (ms) | 128 ± 7 | 48.1 ± 0.9 | **19.8 ± 0.6** |
| (fps) | 8.15 ± 0.33 | 20.9 ± 0.4 | **51.2 ± 1.3** |

side of the box is attached with a printed AprilTag (Tag36h11) [226]. An OptiTrack motion capture system with eight cameras is used to broadcast the positions of the drone(s) and the target. The position of the target is used to detect whether the drones can achieve their goal. In our experiment, the laptop computing center and Tello Edu drones are used to deploy our trained models and conduct flight tests. In addition, for deployment on the onboard computers, three of the following computing platforms are evaluated on their performance in terms of the execution speed of the trained model using only their CPUs. Note that the trained model for only a single drone deployment is considered for this evaluation.

- Lenovo Ideapad Slim 5 Pro laptop running Ubuntu OS with AMD Ryzen 7 5800H CPU and 16GB DDR4 RAM.
- NVIDIA Jetson Xavier NX running Ubuntu OS with a 6-core NVIDIA Carmel 64-bit CPU and 8GB RAM.
- Raspberry Pi 4 Model B with 64-bit ARM Cortex-A72 (ARMv8) and 8GB RAM.

All of these devices use Python3 and ONNXRuntime[4] for inference with the trained model in ONNX format. PyTorch and Torchvision are used for data processing on all devices. Table 3.7 illustrates the execution speed of the trained model in units of frames per second (fps).

Real-time processing capability is crucial for model deployment on computing devices. The evaluation criterion for device performance is set such that for any

[4]https://onnxruntime.ai/

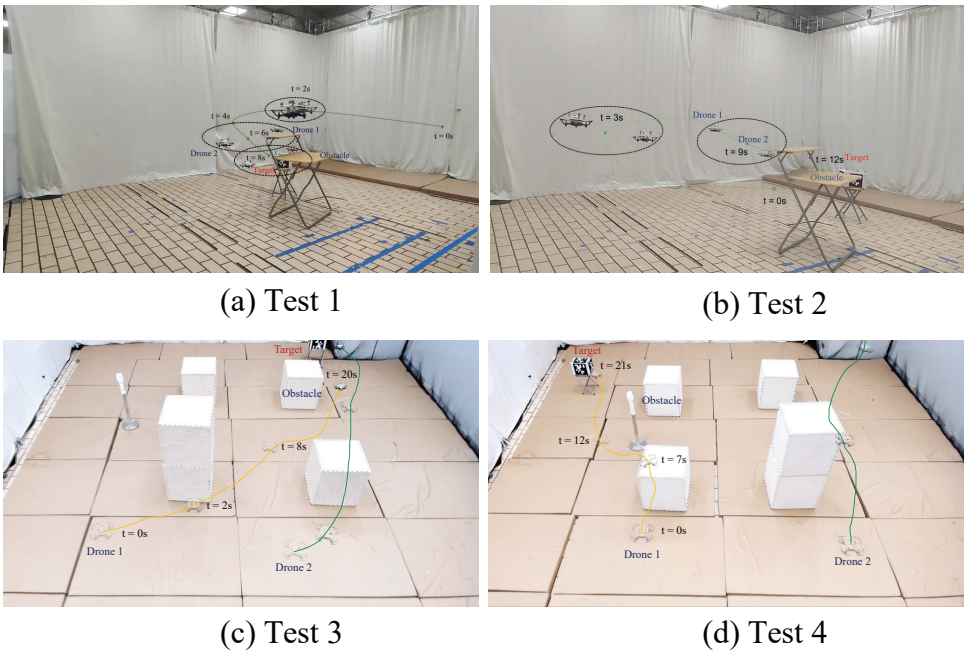



FIGURE 3.9: The trajectories of the drone swarm after successfully performing the CTS in four tests with different initial conditions (Test 1-2 from Env 1 and Test 3-4 from Env 2). The stability of the search policy is demonstrated with different initial positions and forward directions for the drones and the target.

device with real-time processing capability for neural network models, it should be able to execute every part of the trained models, including the data pre-processing step, with a speed of at least 20 frames per second (fps). According to the results in Table 3.7, only the NVIDIA Jetson Xavier NX and Lenovo Ideapad Slim 5 laptop meet the evaluation criterion; therefore, these devices have real-time processing capabilities. However, the Raspberry Pi execution speed is too low to be considered real-time processing. As the experiment evaluated only the performance using CPU, the performance of NVIDIA Jetson Xavier NX and Lenovo Ideapad Slim 5 laptop can be improved further using GPUs, parallel computing with CUDA, and deep learning accelerators, such as TensorRT[5].

Each drone is configured as a wireless AP connected to the computing center during real-time flight tests. As every Tello Edu drone has a similar, immutable IP address (192.168.10.1) on its network, there is an issue of IP address conflict when all drones are connected to the same computer. To resolve, network routing is required to route drones' network to different virtual IP addresses perceived by the computer via docker services[6].

[5]https://developer.nvidia.com/tensorrt
[6]https://www.docker.com/

(a) (b)

FIGURE 3.10: Image blur and loss of the video stream cause the failure of tasks. (a) Obstacle blur; (b) Target blur.

TABLE 3.8: Target search performance in physical experiments

| TDL | Env 1, $\epsilon = 0.5$ | | Env 2, $\epsilon = 1.0$ | |
|---|---|---|---|---|
| Agents | Single Drone | Two Drones | Single Drone | Two Drones |
| SR | 48% | **64%** | 28% | **44%** |
| TTR | | | | |
| (second) | $12.9 \pm 0.8$ | $\mathbf{12.2 \pm 0.8}$ | $45.4 \pm 14.9$ | $\mathbf{39.1 \pm 9.2}$ |
| (step) | $143 \pm 10$ | $\mathbf{142 \pm 13}$ | $656 \pm 236$ | $\mathbf{567 \pm 124}$ |

The drones stream the video data captured by their cameras to the laptop computing center via WIFI. PyAV library is imported to open and decode drones' video stream into image frames with a size of $3 \times 720 \times 960$ recognized by Torchvision. Considering the mismatch between the aspect ratios of the raw images and the model training input, the image frames are firstly cropped to $3 \times 720 \times 720$ around the image center. Thenceforth, Torchvision reshapes and converts each cropped image into a NumPy array of $3 \times 224 \times 224$, consistent with the size of visual observation in the trained model. The model generates translational and angular velocity commands in a loop block based on the obtained visual data and the drone's position. These velocity commands drive connected drones to search for and reach the target collaboratively in real-time (see Fig. 3.8).

In this work, we conducted 25 tests each for single-drone and collaborative two-drone target search to evaluate the performance of trained models in an unseen physical environment. This evaluation focuses on SR and TTR. The experiments

for single-drone and collaborative two-drone target search use different models from the single-agent and multiagent training, respectively. During the tests, the drones were randomly placed in the start area with varying forward directions and positions. A test is considered a success if a drone reaches and lands within 0.5 $m$ of the target; otherwise it is considered a failure. Fig. 3.9 shows the trajectories of several successful test cases.

Table 3.8 compares the performance of the target search using a single drone and two collaborative drones in different physical experiments with $\epsilon = 0.5$ and $\epsilon = 1.0$. The results show that the CTS has higher performance in terms of TTR and SR (64%/44% for two drones, compared to 48%/28% for a single drone). This indicates that collaboration among drones helps to find the target faster and more accurately. Furthermore, the SR in physical experiments is close to that in simulations though in unseen environments, as shown in Table 3.6, demonstrating the generalization and Sim2Real capabilities of trained models. More details can be found in `https://github.com/NTU-ICG/CTS-visual-drone-swarm.git`.

### 3.4.3 Analysis and discussion

#### 3.4.3.1 Stability analysis

Although drones with the trained optimal policy receive feedback from visual perception, the explicit relative position between the target and the agents cannot be obtained in our scenarios. Since the cascade controller is adopted and we only focus on the high-level search decision, the system can be formulated as

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}) \cdot \boldsymbol{u} \tag{3.7}$$

where $\boldsymbol{x} = [x, y, z, \psi]^T$ with $\psi$ as yaw angle, $\boldsymbol{g}(\boldsymbol{x}) = [\cos\psi, \sin\psi, 0, 0; \sin\psi, \cos\psi, 0, 0; 0, 0, 1, 0; 0, 0, 0, 1]$, and the output command of trained policy $\boldsymbol{u} = [\hat{v}_x^B, \hat{v}_y^B, \hat{v}_z^B, \hat{\omega}_z]^T$. Assuming that the image encoder can extract the relative position $\boldsymbol{x_d}$ once the target is recognized, we choose a Lyapunov function $\mathcal{L} = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e}$ with $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_d$. The time derivative $\dot{\mathcal{L}} = \boldsymbol{e}^T\boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} \leq 0$ if $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{e}$. Hence, if the output of the trained policy has negative feedback of the state error $\boldsymbol{e}$ with the positive constant gain $\boldsymbol{K}$, the closed-loop system is stable. An in-depth analysis can be conducted with the help of the interpretability of the image encoder in our future

work. During the inference process, the drones are able to continue the search task when their lost visual perception is recovered, as observed in our experiments. The stability capability of the search policy is also demonstrated in Fig. 3.8, where drones start searching with initial state perturbations.

#### 3.4.3.2 Generalization capability analysis

The proposed ACEMSL, which utilizes various domain randomization techniques, including environment randomization and agent randomization, is capable of transferring trained policies from simulation to real drones performing CTS in a totally different physical environment. The generalization capability can be further improved by using more domain randomization techniques, such as wall texture and image noise filtering. Moreover, the use of relative position observation with the shortest distance allows for easy deployment of the trained policy in a large-scale drone swarm without additional training, unlike the function augmentation in CM3 [110].

#### 3.4.3.3 Failure case analysis

In this work, we aim to achieve the best performance for CTS in physical environments using a visual drone swarm. The lack of global map memory, the position of the target, and the limited visual perception make our CTS task significantly more challenging compared to existing environment exploration problems [210, 211], particularly in real-world environments. Despite these challenges, our approach is still able to achieve the best SR (64%) in a $8m \times 7m \times 4m$ room. This is particularly impressive considering that the SOTA target-driven visual navigation approach [213] only achieves an SR of 43% along a $5 \sim 10m$ path with a fixed room layout.

Some of the failure cases can be attributed to the following factors: (1) video blur, as shown in Fig. 3.10, which provides incorrect information to the trained model, causing the drones to fail to detect the target during flight; (2) imbalanced training dataset for target recognition in unseen corners and seen regions; (3) insufficient memory of the policy model without reserved map buffer. To address these issues, we can introduce the concept of adaptive loss function [227] and anomaly detection [228], which improves the robustness of the model to video with losses and noise.

Additionally, similar to [229], we can utilize the spatio-temporal signal property of image sequences to retrieve target information from blurry images during runtime. Furthermore, we can apply the Recurrent Convolutional Attention (RCA) network [230] to the imbalanced training dataset to improve environment understanding and memory during training and inference, which can help the drone infer the position and presence of the target in hidden corners. These techniques can improve the SR of our proposed approach for CTS tasks.

## 3.5 Conclusion

This chapter proposes a data-efficient RL-based approach called ACEMSL to address the challenges of collaborative target search (CTS) for a visual drone swarm in a cluttered 3D space of sparse reward. Our approach improves upon existing curriculum learning by introducing a novel AEC algorithm and integrating it into task-specific multistage learning. Without prior knowledge of the target, the drone swarm performs CTS using only local visual perception and egocentric observations. Simulation results show that ACEMSL outperforms SOTA direct RL methods and the multistage learning CM3 in policy optimality, training runtime, as well as search efficiency. Moreover, we also demonstrate the generalization and Sim2Real capabilities of our trained models with real-world flight tests in an unseen physical environment. The experiment results verify the effectiveness of a visual drone swarm performing CTS based on the metrics of SR and TTR. To the best of our knowledge, this work is the first to achieve and demonstrate CTS with a visual drone swarm in the real world. The failure cases of our DRL framework motivate future works such as safe DRL with higher bandwidth to evaluate and improve the CTS performance under communication loss, image blur, and even cyber attacks on visual perception.

# Chapter 4

# Learning multitarget search in collaborative environments

This chapter extends the collaborative target search problem in Chapter 3 to learning collaborative multitarget search and navigation (CMTSN) for a visual drone swarm. CMTSN is highly demanded in more complex missions such as rescue and warehouse management in large-scale scenarios. Traditional centralized and decentralized approaches fall short in terms of scalability and adaptability to real-world complexities such as unknown targets and large-scale missions. This chapter addresses this challenging CMTSN problem in three-dimensional spaces, specifically for agents with local visual observation operating in obstacle-rich environments. To overcome these challenges, we present the POsthumous Mix-credit assignment with Attention (POMA) framework. POMA integrates adaptive curriculum learning and mixed individual-group credit assignment to efficiently balance individual and group contributions in a sparse reward environment. It also leverages an attention mechanism to manage variable local observations, enhancing the framework's scalability. Through extensive simulations, we demonstrate that POMA outperforms a variety of baseline methods. Furthermore, we deployed the trained model over a physical drone swarm, demonstrating the effectiveness and generalization of our approach in real-world flight.

## 4.1 Introduction

Multitarget search and navigation with intelligent systems like autonomous drones [231] are highly demanded in critical applications [232], such as disaster rescue in earthquakes, surveillance and reconnaissance in hazardous areas, and parcel management in cluttered warehouses (see Fig. 4.1). The utilization of multiagent systems (MAS) has been considered a more efficient way with more information exchange to conduct such missions while enhancing the resilience and robustness of the search operations [233]. However, ensuring effective collaboration among these agents is a challenging task, especially in obstacles-rich environments with multiple targets. This involves not only information exchange but also synchronization of actions and strategies among all participating agents. Moreover, coordinated interaction is fundamentally required where agents work together towards a common goal, sharing information and making decisions that benefit the overall mission.

FIGURE 4.1: A drone swarm is inspecting parcels in a warehouse where efficient collaborations are required [4].

Traditional approaches mainly address multitarget search problems via search algorithms or sub-task optimization in centralized or decentralized manners. Centralized approaches, such as task assignment from the central node, rely on a central controller to allocate tasks to individual nodes and execute their actions based on global information. Gábor *et al.* [234] proposed a centralized optimization algorithm that considers the spatial constraints of the environment and the capabilities of drones to control a swarm of drones to achieve coordinated flocking in constrained environments. While these methods are easy to deploy, they are

vulnerable to single-node failures and lack scalability. Once the central node is attacked or disturbed, the entire system will cease to function. Meanwhile, the complexity of communication and computation increases rapidly with the number of nodes in the system, which hinders the extension to large-scale scenarios. Decentralized approaches, such as distributed predictive control [235] and decentralized trajectory planner [59], have been proposed to overcome the limitations of centralized approaches. These methods rely on local observation and communication among agents to achieve the desired performance, but suffer from state consensus problems [236]. Besides, these approaches often struggle to adjust to uncertainties and complexities when operating in real-world environments since strong pre-knowledge is assumed. Furthermore, they do not always fully leverage the potential for collaboration when multiple agents and targets are involved as all agents are computed to take the optimal strategy individually.

Multi-agent reinforcement learning (MARL) [64, 218, 225, 237, 238] has recently emerged as a promising tool for generating effective collaboration among agents in sophisticated tasks such as path planning and navigation in cluttered environments. These methods leverage reinforcement learning (RL) algorithms to find the optimal policy for agents with intentionally designed reward functions that aim to encourage collaboration among agents and benefit the entire team. However, these methods either require global observations from the environment [64] or rely on bird-view grid states and stable communication [218, 238], which are infeasible to achieve fully decentralized planning in an uncertain real-world environment such as the multitarget search. In this chapter, we aim to solve the collaborative multitarget search and navigation (CMTSN) problem in 3D space with local visual perception. There are several challenges to achieving fully decentralized decision-making for this task: **(1) balancing** exploration and exploitation in a 3D sparse reward space; **(2) addressing** efficient credit assignment problems within a team, and **(3) improving** the scalability of the trained model. An ACEMSL framework [103] is proposed to address the first challenge, yet it was only for the single target search. MA-POCA [75] solved the posthumous credit assignment by considering the group contribution among multiple agents and introducing the attention mechanism to the centralized critic network. However, MA-POCA ignores the individual's motivation and does not consider the variable observation during execution, which is critical for local perception since the detected signal may be lost.

In this work, we propose a POsthumous Mix-credit assignment with Attention (POMA) framework to address the remaining challenges, namely efficient credit assignment and variable observation. POMA firstly combines adaptive curriculum learning and mixed individual-group credit assignment based on MA-POCA, and then introduces the attention mechanism into observations to improve the scalability of the framework when variable local observations are involved. We validate our framework through extensive simulation and real-world flight experiments.

In our study, we address the complex CMTSN task by formulating it as a partially observable MARL problem. We introduce the POMA framework, adept at tackling critical issues of credit assignment and scalability in a 3D sparse environment, and demonstrate its substantial performance improvement over well-established methods such as MA-POCA and IPPO [239]. Further, we pioneer the development of a drone swarm with local visual perception capabilities for CMTSN in indoor, obstacle-rich environments, utilizing an attention mechanism to enhance collaboration and handle variable observations. This research marks the first successful deployment of a DRL model on a physical drone swarm for CMTSN tasks, proving its effectiveness in real-time flight scenarios and underscoring its practical applicability, thereby setting a new precedent in robot swarm intelligence.

## 4.2 Related work

### 4.2.1 Multiagent reinforcement learning

Recently, MARL [240–242] has been widely applied in path planning, navigation, and target search for intelligent systems. From the perspective of the training framework, MARL can be categorized mainly into three structures, namely, centralized training and execution (CTE), decentralized training and execution (DTE), and centralized training with decentralized execution (CTDE). The CTE [243] contains a central controller that can collect all the state observations and actions from the agents and share them with the group. The controller not only trains the policy but also executes the policy with perfect information. However, CTE is not scalable due to the curse of dimensionality and is limited in environments where perfect information is not accessible. DTE provides a scalable way to search for the optimal policy for each agent via local observations, without communicating with

other agents. PPO [111] is a common algorithm to generate individual policies but can be well extended to MARL within the DTE framework [225]. However, the agents are simply existing together without significant collaboration. Hence, CTDE is the preferred framework in most research works, such as MADDPG [244] with centralized critics, and COMA [109] with centralized joint policy search and QMIX [155] with value function factorization. CTDE allows involved agents to collaborate during training while maintaining scalability during execution, as it combines the advantages of CTE and DTE.

However, to effectively apply the CTDE framework to the CMTSN tasks, we need to address the sparse reward space and credit assignment issues during the training, especially for agents with only local visual perceptions. PRIMAL[218] and PRIMAL2 [74] managed to combine imitation learning and reinforcement learning to train fully decentralized policies for multiagent path finding (MAPF) in sparse reward world and highly structured warehouses, respectively. To better utilize the information from the neighbors, Li *et al.*[238] developed a GNNHIM framework based on the graph neural network to extract the information from multi-hop neighbors. However, all these methods assume 2D grid environments and known target information, which is not practical for CMTSN with visual drones in our case.

### 4.2.2 Credit assignment

Credit assignment is a crucial issue in the cooperative MARL problems, which determines the contribution of each agent to the group's success or failure. In our CMTSN problem where agents work together to find all targets, it is essential to identify which action by which agent led to a higher search success rate. Various works have been done to address the credit assignment issue in MARL. To estimate the contribution of each agent, Devlin S. *et al.*[245] calculated the difference between the global reward and the estimated reward if ignoring this agent. But this method suffers the computational cost since the combinations of each agent's absence are simulated at every step. COMA[109] used a centralized critic to estimate the action-value function, and for each agent, it computes a counterfactual advantage function to represent the value difference which can determine the contribution of each agent. However, COMA still does not fully address the model

complexity issue. QMIX [155] is well developed to address the scalability issue by decomposing the global action-value function into individual agent's value functions. However, QMIX assumes the environment is fully observable and may not be able to handle scenarios with continuous action space. Another issue regarding credit assignment is the variable number of agents, which induces the posthumous credit assignment problem. For instance, in the CMTSN task, drones may crash into an obstacle or other drones during the flight but the remaining drones are required to continue the search task. MA-POCA [75] provides valuable insights for addressing the posthumous credit assignment via introducing a self-attention mechanism across value function estimation and counterfactual baseline [122] but it lacks consideration of the individual's motivation. This would bring down the performance of MA-POCA, especially for environments with sparse reward space and multiple targets like CMTSN.

To address these challenges, in this chapter, we combine the advantages of MA-POCA and PPO to balance group contribution and individual motivation through well-designed group reward and individual reward functions. We further introduce the attention mechanism to encode the local observations, which significantly improves the inner collaboration and the scalability of the policy in various scenarios.

## 4.3 Preliminaries

### 4.3.1 Multiagent reinforcement learning

The CMTSN problem can be characterized as a decentralized Partially Observable Markov Decision Process (POMDP) defined by $\mathcal{M} : \langle \mathcal{N}, \mathcal{S}, \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \mathcal{R}, \rho_0 \rangle$, where $\mathcal{N} = \{1, ..., N\}$ represents the set of finite $N \geq 1$ agents. The state of the environment at time $t$ is denoted by $s_t \in \mathcal{S}$. The local observation and action of agent $i$ related to $s_t$ are defined by $o_t^i \in \mathcal{O}^i$, $a_t^i \in \mathcal{A}^i$, respectively. Let us define joint action space $\mathcal{A} := \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ and joint observation space $\mathcal{O} := \mathcal{O}^1 \times \cdots \times \mathcal{O}^N$, then $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ represents the transition function from state $s_t \in \mathcal{S}$ to the subsequent state $s_{t+1} \in \mathcal{S}$ given the joint action $\boldsymbol{a}_t \in \mathcal{A}$ with probability $\mathcal{P}(s_{t+1} \mid s_t, \boldsymbol{a}_t)$. The reward function shared by all agents is given by $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ with $r(s_t, \boldsymbol{a}_t)$ representing the received reward for all agent. To be simple, $r_t$ is

used to denote $r(s_t, \boldsymbol{a}_t)$. $\rho_0$ is the distribution of initial states. $\gamma \in [0, 1)$ is the discount factor for calculating the cumulative rewards.

Let $\boldsymbol{\pi}(\boldsymbol{a}_t \mid \boldsymbol{o}_t) := \prod_{i \in \mathcal{N}} \pi^i(a_t^i \mid o_t^i)$ is the joint policy with $\pi^i(a_t^i \mid o_t^i)$ representing the policy of each independent agent $i \in \mathcal{N}$. $T$ is the maximum allowable steps in an episode. In the CTDE framework, the centralized state-value function for state $s_t$ is described as

$$V^{\boldsymbol{\pi}}(s_t) := \mathbb{E}_{\boldsymbol{\pi}} \left[ \sum_{k=0}^{k=T-t} \gamma^k r_{t+k} \mid s_t \right] \tag{4.1}$$

The centralized state-action value function is

$$Q^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) := r_t + \mathbb{E}_{\boldsymbol{\pi}} \left[ \sum_{k=1}^{k=T-t} \gamma^k r_{t+k} \mid s_t \right] \tag{4.2}$$

The objective of MADRL is to find an optimal joint policy $\boldsymbol{\pi}^*$ maximizing the state-value of the team starting from the initial state $s_0$. To evaluate the contribution of each agent, the counterfactual baseline [109] $b_i(s, \boldsymbol{a})$ is computed by marginalizing the action of agent $i$ from the team,

$$b_i(s, \boldsymbol{a}) = \mathbb{E}_{a' \sim \pi^i(\cdot \mid o^i)} \left[ Q^{\boldsymbol{\pi}}(s, (\boldsymbol{a}^{-i}, a')) \right] \tag{4.3}$$

Here, $a'$ is the action of agent $i$ and $\boldsymbol{a}^{-i}$ is the joint action without $a'$. With the baseline, the advantage of agent $i$ can be defined by

$$\hat{A}_i = Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) - b_i(s, \boldsymbol{a}) \tag{4.4}$$

The optimal policy for each agent $i$ is obtained by iteratively updating the policy with the gradient

$$\Delta_{\theta^i} \mathcal{J}(\theta^i) = \mathbb{E}_{s \sim P, a^i \sim \pi^i} \left[ \Delta_{\theta^i} \log \pi^i(a^i \mid o^i) \hat{A}_i \right] \tag{4.5}$$

The advantage function considers the contributions of individual agents to the shared team reward by measuring the value difference with or without the agent's involvement.

To address the posthumous credit assignment issue, MA-POMA [75] utilized observation entity encoders and self-attention modules to handle the varying number

of agents at every time step. First, to obtain the $Q$ value, the attention masked centralized state value function parameterized with $\phi$ in MA-POMA is:

$$V_{\phi}^{\boldsymbol{\pi}}(RSA(g_i(o_t^i)_{i\in[k_t]})) \tag{4.6}$$

where $[k_t] : 1 \leq k_t \leq N$ is the number of active agents at time step $t$; $g_i : \mathcal{O}^i \mapsto E$ is the observation entity encoder for agent $i$ and $RSA$ is the self-attention module used. Similarly, the counterfactual baseline parameterized with $\psi$ for agent $i$ to calculate $b_i(s, \boldsymbol{a})$ is:

$$Q_{\psi}^{\boldsymbol{\pi}}(RSA(g_i(o_t^i), f_{-i}(\mathbf{o}_t^{-i}, \mathbf{a}_t^{-i})_{-i\in[k_t]})) \tag{4.7}$$

here, $f_{-i} : \mathcal{O}^{-i} \times \mathcal{A}^{-i} \mapsto E$ is the encoding network for observation-action pairs from agents except for $i$.

### 4.3.2 Attention mechanism

The attention mechanism [122] has been widely applied in various deep-learning architectures. In DRL, attention has been used to create a weighted sum of observations, where the weights reflect the relevance of each observation to the task [246]. Let's consider a partial observable task, where we have an uncertain observation sequence $\boldsymbol{o}_t^i := \{s_t^j \mid j \in \mathcal{N}^{-i}\}$ for agent $i$ and we want to encode the observation into a fixed-length context vector $c_t^i$ with attention weights $w_{ij}$:

$$c_t^i = \sum_{j\in\mathcal{N}^{-i}} w_{ij} W_v s_t^j \tag{4.8}$$

The attention weights can be calculated by:

$$w_{ij} = softmax(score((W_q s_t^i)^T, W_k s_t^j)) \tag{4.9}$$

where $score(\cdot)$ is the scoring function to calculate the relevance of the neighbor states $s_t^j$ to the ego state $s_t^i$. The piece-wise dot product is widely adopted as the score function. $W_v$, $W_q$, and $W_k$ are the parameters to be trained to encode the states.

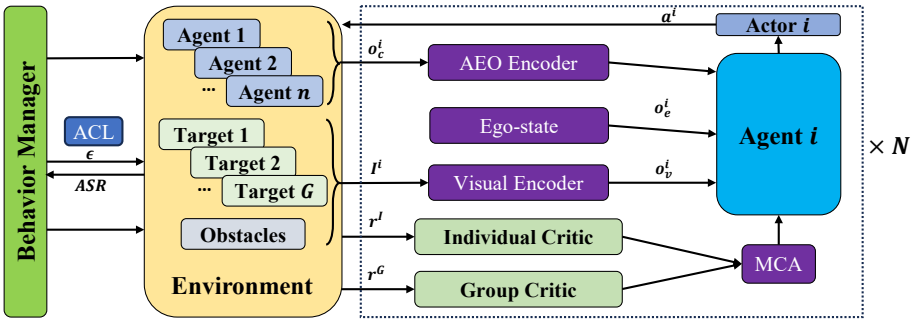



FIGURE 4.2: Overview of our POMA framework. A centric behavior manager controls the TDL of the environment with an ACL feedback module. The MCA module integrates the individual critic into the group critic to address the individual-group conflicts. The AEO encoder with a self-attention mechanism improves the generation capability of policy with variable neighbors.

## 4.4 Methodology

### 4.4.1 Problem formulation

CMTSN with autonomous agents is highly demanded and has been investigated in various applications such as disaster search and rescue and warehouse management in complicated environments. Unlike the two-dimensional (2D) discrete grid scenario considered in other works [74, 238], in such real-life environments, QR code distribution and prior goal information are infeasible in most cases.

In this chapter, we consider a bounded obstacle-rich three-dimensional (3D) continuous space $D \in \mathbb{R}^3$ with the size of $L \times W \times H$, where $L$, $W$ and $H$ is the length, width, and height of the environment space respectively. Assume that we have $N$ agents $\mathcal{N} = \{1, \ldots, N\}$ that can move in 3D with continuous action, and $M$ blocks $\mathcal{B} = \{B_1 \ldots B_M\}$ with size of $l^j \times w^j \times h^j, B_j \in \mathcal{B}$. Each agent starts from a preset initial position $p^i_{ini}$ and attitude $q^i_{ini}$ with random position perturbation $p^i_w$ and rotation perturbation $q^i_w$ for $i \in \mathcal{N}$. The CMTSN task is defined as that agents need to find and approach all $G$ targets $\mathcal{T} = \{T_1, \ldots, T_G\}$ with the shortest time. All targets will be marked with found $F$ or not-found $NF$. The status of each target is denoted as $ST^k \in \{F, NF\}_{k \in \mathcal{T}}$. Note that, during this task, the agent can always continue finding other targets unless it crashes or all targets are found.

#### 4.4.1.1 Observation and action space

In the partial observable RL task, each agent draws actions from its individual policy on the condition of the current local observation, i.e., $a_t^i \sim \pi^i(o_t^i \mid \theta_t^i)$ with $\theta_t^i$ the trainable parameter. In this chapter, for each agent $i$, the local observation includes the visual observation $o_{v,t}^i$, i.e., RGB image $I_t^i \in \mathbb{R}^{3\times 224\times 224}$, the ego-state observation $o_{e,t}^i$, i.e., position $\boldsymbol{p}_t^i \in \mathbb{R}^3$, rotational quaternion $\boldsymbol{q}_t^i \in \mathbb{R}^4$, normalized forward direction $\bar{\boldsymbol{n}}_W^i$ and previous action $a_{t-1}^i \in \mathbb{R}^4$, and the ego-centric observation $o_{c,t}^i$, i.e., the relative position towards neighbours $\boldsymbol{p}_t^{ij} = \boldsymbol{p}_t^j - \boldsymbol{p}_t^i$ for $j \in \mathcal{N}^{-i}$. Since the number of agents is not fixed in our task, a maximum number of observable neighbors $N_{max}$ is preset, and hence $\boldsymbol{p}_t^{ij} \in \mathbb{R}^{N_{max}\times 3}$. The attention mechanism is adopted to calculate the scores of observable neighbors, which is discussed in the following proposed POMA framework. The visual observation $I_t^i$ is encoded into a vector $o_{v,t}^i \in \mathbb{R}^{256}$ with a simple convolutional neural network (CNN).

To align with the real operation of drones, the action space of agents consists of four continuous velocity commands, namely the translational velocity $\hat{v}_x^B$, $\hat{v}_y^B$, $\hat{v}_z^B$ and the yaw speed $\hat{\omega}_z$ in the body-fixed frame. The action space of agents is constrained with $||\hat{\boldsymbol{v}}^B|| \leq v_{max}$ and $\hat{\omega}_z \leq \omega_{max}$.

#### 4.4.1.2 State transition function

Assuming the low-level PID actuator controller can track the actions generated with acceptable delay, we consider the high-level policy for the drone. The 4-DOF dynamics with the aforementioned velocity commands can be described as:

$$\dot{\boldsymbol{s}} = \boldsymbol{g}(\boldsymbol{s}) \cdot \boldsymbol{a} + \boldsymbol{\omega} \tag{4.10}$$

where $\boldsymbol{s} = [x, y, z, \psi]^T$ with $x, y, z$ the world position coordinates and $\psi$ the heading angle. $\boldsymbol{g}(\boldsymbol{s})$ is the transition matrix from body-fixed frame $\Omega^B$ to world frame $\Omega^W$ and $\boldsymbol{a} := [\hat{v}_x^B, \hat{v}_y^B, \hat{v}_z^B, \hat{\omega}_z]^T$ is velocity commands (generated actions) sent to each drone. $\boldsymbol{\omega}$ represents the process noise and uncertainties, which satisfies the

Gaussian distribution. Assuming no significant tilting perturbation, $\boldsymbol{g}(\boldsymbol{s})$ is:

$$\boldsymbol{g}(\boldsymbol{s}) = \begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

Note that the transition function (4.10) applies to each agent and determines the next state $s_{t+1}$ in the discrete space given the current state $s_t$ and the joint action $\{\boldsymbol{a}_t^i\} \in \mathcal{A}$ with stochastic noise $\boldsymbol{\omega}$. Hence the policy is not deterministic in our task.

### 4.4.2 Posthumous mix-credit assignment with attention

In this section, we describe our proposed POMA framework (see Fig. 4.2) extended from MA-POCA (keep the same posthumous credit assignment mechanism), which leverages Adaptive Curriculum Learning (ACL), Mixed Credit Assignment (MCA), and Attention-Enhanced Observation (AEO) to improve the performance and generalization capability of MA-POCA over CMTSN tasks.

#### 4.4.2.1 Adaptive curriculum learning

Similar to automatic curriculum learning [247], the motivation of our ACL module is to gradually guide the agent learning to search for targets from near locations (simple tasks) to far locations (difficult tasks) considering the sparse rewards in our scenario. Our ACL adjusts the task difficulty level (TDL) with on-policy training but only controls the initial state $\rho_0$. Hence it can be regarded as one specific instance of automatic curriculum learning.

During each episode beginning, the targets are randomly spawned in three space sets based on the search distance ($L$ direction), namely, the near set $\mathcal{C}_{\mathcal{N}}$, the middle set $\mathcal{C}_{\mathcal{M}}$, and the far set $\mathcal{C}_{\mathcal{F}}$, with probabilities of $Pr(\mathcal{C}_{\mathcal{N}})$, $Pr(\mathcal{C}_{\mathcal{M}})$ and $Pr(\mathcal{C}_{\mathcal{F}})$ respectively. Note that $Pr(\mathcal{C}_{\mathcal{N}}) + Pr(\mathcal{C}_{\mathcal{M}}) + Pr(\mathcal{C}_{\mathcal{F}}) = 1.0$ and $Pr(i)_{i \in \mathcal{C}_{\mathcal{N}}, \mathcal{C}_{\mathcal{M}}, \mathcal{C}_{\mathcal{F}}} \geq 0$. Since the agents are expected to find the far targets more difficult, the task difficulty level (TDL) is denoted by $\epsilon = 1 - Pr(\mathcal{C}_{\mathcal{N}}) \in (0, 1)$ which represents the difficulty

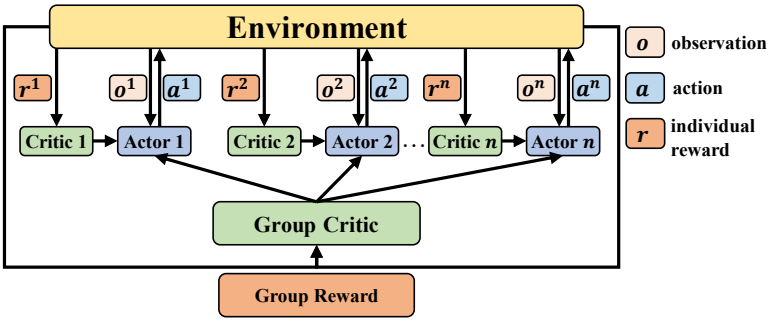



FIGURE 4.3: The MCA architecture combines the group critic and individual critic, which optimizes the team contribution while encouraging individual exploration.

to find all targets. The main performance metric for our task is the average success rate (ASR) of the team over a moving time window $T_w$,

$$ASR(t) = \frac{1}{T_w} \sum_{i=max\{t-T_w,1\}}^{i=t} \mathbf{1}(Out(i) == success) \tag{4.12}$$

Here, $\mathbf{1}(1) = 1$, otherwise $\mathbf{1}(0) = 0$ and $Out(i)$ denotes if task is successful. Our ACL adjusts TDL to change against the variation direction of the success rate achieved by the team with clipped control law:

$$\epsilon_{t+1} = clip(\epsilon_t + \eta(ASR - ASR_d), 0.5 - \xi, 0.5 + \xi) \tag{4.13}$$

where $\eta$ is the rate that controls the update speed and $\xi$ is the coefficient that controls the update range of $\epsilon$. $\epsilon_0$ and $ASR_d$ are the initial TDL value and the desired average success rate in one task. The basic idea is that when the ASR is low at the beginning, the TDL drops to the lower boundary $0.5 - \xi$ and when the ASR is greater than the desired ASR, the TDL will increase to give more challenges to the team but bring down the ASR. The feedback from the ASR aims to adjust the TDL in real-time and $\epsilon$ converges to $\epsilon^*$ with any ASR.

#### 4.4.2.2 Mixed credit assignment

Different from existing baselines IPPO [239] using only individual critics to evaluate individual agent's behaviors or MA-POCA [75] only considering a group critic to evaluate each individual agent's behavior towards the group objectives, the proposed MCA module combines the individual critics and a group critic to optimize

the policy with individual and group rewards, as illustrated in Fig. 4.3. Within the MCA module, the individual critic aims to encourage the agent to search for targets as many as possible with individual effort while the objective of the group critic is to collaborate with other agents to find all targets efficiently. Compared to MA-POCA, an additional individual critic is added in POMA to provide more information and criticism for agents to make better informed decisions. The individual critic is defined by

$$Q_i(s_t, a_t^i) = r_t^I(s_t, a_t^i) + \gamma V_i(s_{t+1}) \tag{4.14}$$

where $Q_i$ and $V_i$ are the state-action value function and state value function kept for the agent $i$. $r_t^I(s_t, a_t^i)$ is the individual reward. The generalized advantage estimation (GAE) for this individual critic is:

$$\hat{A}_i'(\lambda) = \sum_{j=0}^{T} (\gamma\lambda)^j \delta_{t+j} \tag{4.15}$$

where $\delta_{t+j} = Q_i(s_{t+j}, a_{t+j}^i) - V_i(s_{t+j})$ denotes the Temporal Difference (TD) error. $\gamma$ is the discount factor and $\lambda$ is the hyperparameter.

Since each actor now has two separate critics (individual critic and group critic) providing guidance, the dilemma of pursuing individual or group rewards is posed to the actor. Compared to IRAT in [248] trying to clip the difference between the individual policy and group policy for each agent $i$, which brings heavy trainable parameters, especially for large-scale multiagent systems, our MCA module aims to shape the advantage function from the individual rewards and group rewards straightforward. We combine the advantage function from individual reward and group reward for agent $i$ as:

$$\hat{A}_i^c = \lambda^c \hat{A}_i + (1 - \lambda^c)\hat{A}_i'(\lambda) \tag{4.16}$$

Here $\hat{A}_i$ is the advantage function calculated from (4.2), (4.4) and (4.7) and $\lambda^c$ is the hyperparameter to adjust the group critic and individual critic. Note that the reward in (4.2) is the group reward $r_t^G(s_t, \boldsymbol{a}_t)$. The updated gradient to iterate the policy for agent $i$ is:

$$\Delta_{\theta^i} \mathcal{J}(\theta^i) = \mathbb{E}_{s\sim P, a^i \sim \pi^i} \left[ \Delta_{\theta^i} \log \pi^i(a^i \mid o^i) \hat{A}_i^c \right] \tag{4.17}$$

TABLE 4.1: Reward function used for various methods.

| **Rewards** | **PPO** (Individual) | **MA-POCA** (Group) | **POMA** (Mixed) |
|---|---|---|---|
| Crash into wall | -1.0 | -1.0 | -1.0 (I) |
| Crash into agent | -3.0 | -3.0 | -3.0 (I) |
| All drones crash | -3.0 | -3.0 | -3.0 (G) |
| Found target | +5.0 | +5.0 | +5.0 (I) |
| Found all targets | +10.0 | +10.0 | +10.0 (G) |
| Existential reward | $r_e$ | $r_e$ | $r_e$ (I) |
| Collision reward | $r_c$ | $r_c$ | $r_c$ (I) |

To better utilize the MCA module, the reward functions need to be well-designed to balance individual effort and team contribution in different tasks. In this chapter, individual rewards $r^I$ are given only when the agent finds a target or when it crashes. This encourages competitive behaviors within the team and ensures that the entire group is not punished due to a particular agent's mistake. By adding group rewards for finding all targets and all drones crashing, the group rewards provide better feedback to the group on collaborative search behavior while avoiding higher crashes. Beyond these task rewards, the existential reward $r_e = -1.0/T$ and the collision reward $r_c$ are added for all methods. Here,

$$r_c^i = -\alpha \frac{\|\boldsymbol{p}_T^{ig}\|}{\|\boldsymbol{p}_{ini}^{ig}\|} - \frac{\beta}{\pi} \arccos \frac{\langle \boldsymbol{p}_T^{ig}, \bar{\boldsymbol{n}}_W^i \rangle}{\|\boldsymbol{p}_T^{ig}\| \cdot \|\bar{\boldsymbol{n}}_W^i\|} \tag{4.18}$$

where we define $\boldsymbol{p}_T^{ig}$ as the displacement from the collision position to the target position for agent $i$, and $\boldsymbol{p}_{ini}^{ig}$ as the initial displacement value. The norm of a vector is denoted by $\|\cdot\|$, and the inner product of two vectors is denoted by $\langle \cdot \rangle$. We introduce two weights, $\alpha \in (0, 1]$ and $\beta \in (0, 1]$, which are utilized to adjust the penalty associated with the navigation distance and the penalty arising from the forward direction. The individual rewards and group rewards are listed in Table 4.1 where G means group reward, while I means individual reward.

#### 4.4.2.3 Attention enhanced observation

To address the generalization issue of policy with variable ego-centric observations, the relative position towards the agent with the minimum distance is set as the ego-centric observation in [103]. However, such a design ignores the information from

other agents. To better utilize all observable neighbors, the attention mechanism is adopted. The ego-centric observations are first padded with zeros to $I^i$ with a fixed size of $N_{max} \times 3$. The padding observation is then embedded into vectors query $Q$, key $K$, and value $V$ with linear normal layers $LN(I^i \mid \Theta^i)$ of embedding size $d_k/m$. The output of the attention enhanced observation is:

$$C_t^i = softmax\left(\frac{Q^T \cdot K}{\sqrt{d_k/m}}\right) \cdot V \tag{4.19}$$

where $m$ is the number of heads and $d_k$ is the dimension of padding observation. The output is further processed with Residual modules and batch normalization as the same with Transformer architecture [122].

#### 4.4.2.4 Network architecture

The actor network architecture comprises a visual encoder and a policy network. The visual encoder is implemented as a simple CNN with two convolutional layers $(16 \times 8 \times 8 \mid 32 \times 4 \times 4)$. The activation function used in the visual encoder is Leaky ReLU. Its purpose is to encode the raw image input into a compact 256-dimensional vector representation. Additionally, the visual encoder is followed by the concatenation of ego-state observation $o_e$ ($C_1 \in \mathbb{R}^{270}$) and then cross-encoded with ego-centric observation ($o_c \in \mathbb{R}^{N_{max} \times 3}$) with a multi-head attention network, resulting in a vector $C_2 \in \mathbb{R}^{398}$ with concatenation from $C_1$. On the other hand, the policy network consists of two fully connected (FC) layers, each consisting of 256 nodes. The final layer of the policy network comprises 4 nodes, responsible for generating four continuous action values to control the agent. Both the individual critic and group critic networks are the same with two FC layers of 256 nodes and a final value layer.

#### 4.4.2.5 CTDE with behavior manger

The proposed framework is trained with CTDE, where the critic network collects all agents' trajectories, and each agent executes the same actor network. A centric behavior manager is designed to manage the group behavior of all OOIs (targets, agents, and obstacles) such as the group registration and the number of OOIs while

the agent's behavior like dynamics is controlled by its own behavior script. The group rewards are also defined in the behavior manager and triggered in the agent behavior script. The detailed training process for POMA is listed in **Algorithm 3**.

The behavior manager controls the initialization and updates of the environment with the TDL rule (4.13). Each agent is updated with individual Monobehaviour to obtain the observations and transit to the next time step with dynamics and actions from the current policy. The policy is updated for each episode with the policy gradient (4.17).

**Theorem 4.1.** *With the clipped control law (4.13), the TDL* $\epsilon$ *converges to* $\epsilon^* \in [0.5-\xi, 0.5+\xi]$ *with any given desired ASR* $ASR_d$ *and initial TDL* $\epsilon_0$.

*Proof.* Assume that any $\epsilon_0 \in (0,1)$ is given such that $ASR < ASR_d$ over finite episodes $[E]$, then $\epsilon_t \to 0.5-\epsilon$ within time steps $E \times T$ such that $ASR$ converges to $ASR'$ since $ASR - ASR_d < 0$. If $ASR' \leq ASR_d$ then $\epsilon_t \to \epsilon^* = 0.5-\xi$. If $ASR' > ASR_d$, there is a $\epsilon^*$, $\epsilon_t \to \epsilon^* > 0.5-\xi$ such that $ASR'$ converges to $ASR'' \leq ASR_d$ since $ASR(t) \propto \frac{1}{\epsilon_t}$. Otherwise, $\epsilon_t \to \epsilon^* = 0.5+\xi$ and $ASR'' > ASR_d$. If $\epsilon_0$ is given so that $ASR > ASR_d$ over finite episodes $[E]$, let $ASR' = ASR$ be denoted, similarly, it follows the conditions discussed and $\epsilon_t \to \epsilon^* > 0.5-\xi$ or $\epsilon_t \to \epsilon^* = 0.5+\xi$. If $\epsilon_0$ is given such that $ASR \equiv ASR_d$, then $\epsilon_0 \to \epsilon^* = \epsilon_0$. This completes the proof. □

Theorem 4.1 provides theoretical support to ensure the convergence of training with ACL if the basic policy improvement is convergent.

## 4.5 Simulation experiments

### 4.5.1 Settings

Our simulation environments are developed with the Unity game engine. The proposed POMA framework is coded based on the Unity ML-Agents Toolkit Release 20 [249]. In the training process, two visual drones with a Field of View (FOV) of 82.5 deg are randomly spawned to search for two targets ($G = 2$) in a room of size

**Algorithm 3:** The training process of POMA

**Input:** An initial TDL $\epsilon_0$, changing rate $\eta$, desired ASR $ASR_d$ and range factor $\xi$.

**Output:** The decentralized policy $\pi^*$.

At the start of training:
Register all agents to the group;
Initialize agents' position $\boldsymbol{p}_{W,0}$, velocity $\boldsymbol{v}_{W,0}$ and rotation $\boldsymbol{q}_{WB,0}$ with randomization;
Initialize all obstacles and targets according to $\epsilon_0$ (make sure all OOIs are not overlapped);
Set $ASR = 0.0$, $ATS = 0$, $slidingWindow = \mathbf{0}_{[T_w]}$, $episodeCount = 0$ ;
Construct the critic network and actor network.
**for** *episode* $= 0$ *: maximum episodes* **do**
    On each episode begin:
    **if** $episodeCount \neq 0$ **then**
        $ASR \leftarrow mean(slidingWindow)$;
    $index \leftarrow episodeCount \mod T_w$;
    $slidingWindow[index] \leftarrow 0$;
    $episodeCount \leftarrow episodeCount + 1$;
    Update $\epsilon$ according to (4.13);
    Reset and register all agents back to the group;
    Randomize the intensity of light; Spawn the obstacles and targets according to updated TDL $\epsilon$ (make sure all OOIs are not overlapped);
    **for** *step* $t = 0 : T$ **do**
        Add group reward $r_e$;
        **for** *Each agent* $i \in \mathcal{N}$ **do**
            Collect and encode observations $o^i$; Perform actions $a^i$ with current policy $\pi^i$; Add individual reward $r^I$;
        **if** *Find one target* **then**
            Set the target to inactive;
        **if** *Crash with OOIs* **then**
            Set the agent $i$ to inactive;
        **if** *all targets found* **then**
            $slidingWindow[index] \leftarrow 1$; Add group reward +10; End the episode;
        **if** *all drones crash* **then**
            Add group reward $-3$; End the episode;
            back to 7;
    Conduct policy improvement with (4.17);

$5m \times 5m \times 2m$ with 2 obstacles. The initial TDL is set to $\epsilon_0 = 0.1$. All OOIs are randomly spawned and pre-checked to avoid any collisions at the beginning of each episode. The ASR is calculated in 500 time steps ($T_w = 500$) after each episode

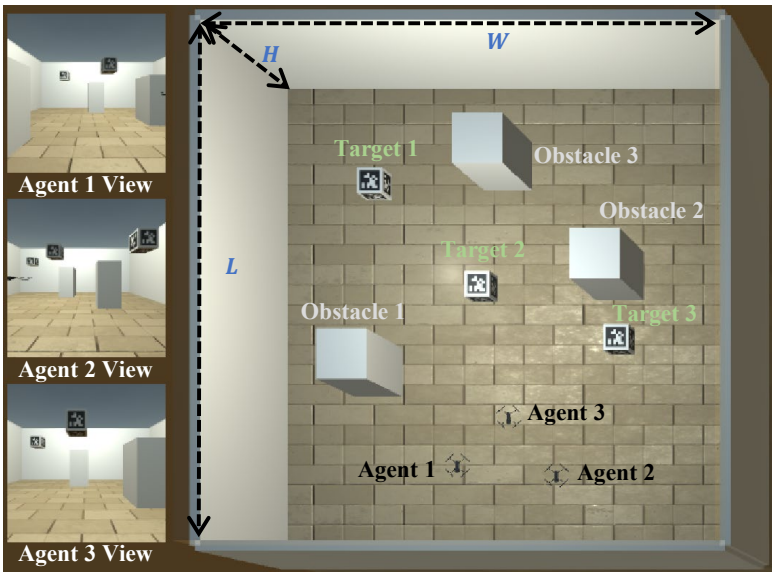



FIGURE 4.4: The simulation environment for the CMTSN with 3 drones, 3 targets (3D3T), and 3 obstacles (3O) of different sizes. The task is successful only if all targets are found and approached. The left 3 views are drones' visual perception.

and the TDL is adaptively adjusted according to the rule (4.13) with $\xi = 0.4$, $\eta = 0.001$, $ASR_d = 0.7$. For each episode, a maximum of 5000 time steps are set and we train the policy for 10 million episodes. We set $N_{max} = 5$ since the room size limits the number of agents in the team. The training is accelerated with 2 parallel environment instances with each of the 6 scenarios inside. The training hyperparameters are listed in the supplementary material.

We use the IPPO [239] and MA-POCA [75] as the baselines and test their performance with different missions such as 1 drone vs. 2 targets (1D2T), 3D3T and etc. in the 3-obstacle room and set the TDL $\epsilon = 0.2$. The ablation study is further conducted to examine the effectiveness of designed modules.

### 4.5.2 Environment setup

In this chapter, the policy is first trained in simulated environments and then transferred to the physical world. The training scenario is developed with a Unity simulation engine as illustrated in Fig. 4.4, where the drones are trained to search for several target boxes covered with AprilTag and meanwhile to avoid obstacles (blocks and walls). To better the generalization capability of the policy, the size

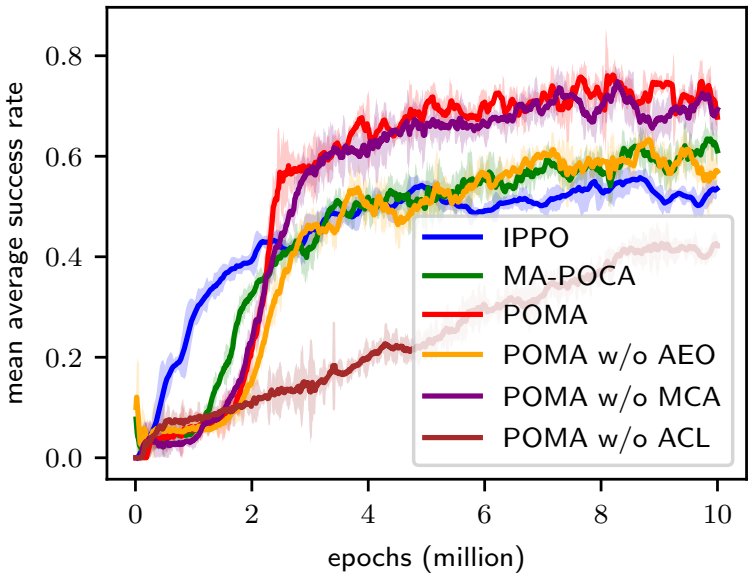



FIGURE 4.5: Learning curves of IPPO, MA-POCA, POMA (ours) and different ablations.

of blocks is randomized at the beginning of each training episode. Each block is randomly placed in space $D$ and cannot overlap with other blocks.

All objects of interest (OOIs) are meshed with colliders and labeled with agent, target, and obstacle tags. The agent disappears and the number of agents $N$ decreases by one if the agent collides with other agents or obstacles. Similarly, the target disappears if the target is approached by any agent, and the number of targets $G$ decreases by one. The training episode terminates if $N = 0$ or $G = 0$ or the maximum allowable steps $T$ is reached.

## 4.5.3 Hyperparameters

### 4.5.3.1 Hyperparameters for RL

Since all of our implementations are extended from the PPO and MA-POCA, they share the same policy update mechanism and hyperparameters for RL. The critics and actor networks share the same network size. Table 4.2 lists the hyperparameters used for all training.

TABLE 4.2: Hyperparameters for all RL algorithms

| Parameter | Value |
|---|---|
| Image size | $3 \times 224 \times 224$ |
| Batch size | 2048 |
| Buffer size | 20480 |
| Learning rate $\eta$ | 0.0003 |
| Discount factor $\gamma$ | 0.99 |
| Number of epochs | 3 |
| Optimizer | Adam |
| Checkpoints | 10 |
| Time horizon | 128 |
| Entropy bonus $\beta$ | 0.01 |
| Clip threshold $\epsilon^{ppo}$ | 0.2 |
| Regularization factor $\lambda^{ppo}$ | 0.95 |
| Maximum episodes | 10 million |
| Hidden units | 256 |
| FC layer | 2 |

#### 4.5.3.2 Hyperparameters for POMA

The parameters used in this work are optimized with trials and tests in several modules. For instance, the changing rate $\eta$ is chosen to reduce the training vibration. Table 4.3 summarizes other hyperparameters specifically for POMA in the training environment.

All training and testing are carried out on a workstation with the configuration of Ubuntu 20.04, CPU AMD ZEN 3 RYZEN 9 5900X, and GPU RTX 3090 Ti. Unity version 2021.3.11, ml-agents 0.27.0, Pytorch 1.8.2, and Python 3.8 are used to develop the simulation and algorithm.

### 4.5.4 Domain randomization

To better the generalization of policies during the training process, several domain randomization techniques are adopted. Table 4.4 lists the parameters for different types of domain randomization. Note that $U(x_1, x_2)$ means the uniform distribution over $[x_1, x_2]$. $B(c, r)$ denote the sphere uniform distribution with center position $c$ and radius of $r$. $x_l$, $x_r$, $z_l$, and $z_u$ are the left side, right side, low side, and upper side of the room, respectively. The position of targets depends on the

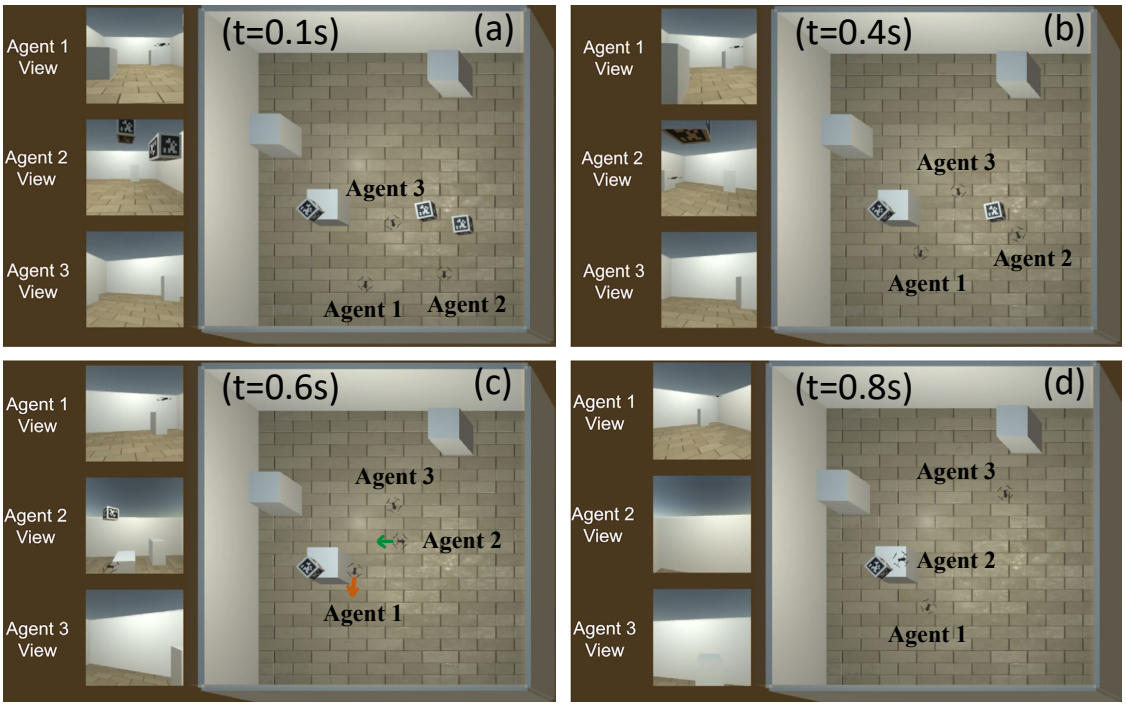



FIGURE 4.6: Snapshots of simulation with 3D3T. The time sequence is from (a) to (d). It shows significant collaborative behaviors (Agent 1 moves back to provide space for Agent 2 to navigate to the final target as observed in (c)).

TABLE 4.3: Parameters for the POMA and environment

| **Parameter** | **Value** |
|---|---|
| Environment size | $5 \times 5 \times 2m$ |
| FOV of camera | 82.5 deg |
| Training IOOs | 2D2T, 2 obstacles |
| Testing IOOs | 3 obstacles |
| Maximum speed $v_{max}$ | 1.0 m/s |
| Maximum angular speed $\omega_{max}$ | 1.0 rad/s |
| Initial TDL $\epsilon_0$ | 0.1 |
| TDL clip factor $\xi$ | 0.4 |
| Changing rate $\eta$ | 0.001 |
| Desired ASR | 0.7 |
| Maximum step $T_{max}$ | 5000 |
| Group critic weight $\lambda^c$ | 0.5 |
| Reward weight $\alpha$ | 1 |
| Reward weight $\beta$ | 0.1 |
| Maximum observation entities $N_{max}$ | 5 |
| Attention entity embedding size | 256 |
| Attention entity embedding layers | 1 |
| Number of Attention heads | 4 |

TABLE 4.4: Domain randomization parameters

| **Types** | **Range** |
|---|---|
| Light intensity $I_v$ | $U(0.2, 1.0)$ |
| Size of obstacles | $[0.5, 0.5 + U(0, 0.5), 0.5]$ |
| Position of obstacles | $[U(x_l, x_r), U(0, 0.5), 0]$ |
| Scale of target $\lambda_{target}$ | $U(0.2m, 0.3m)$ |
| Yaw angle of target $\psi_{target}$ | $U(0°, 360°)$ |
| Position of target | $[U(x_l, x_r), U(1, 1.5) + \epsilon * L, U(z_l, z_u)]$ |
| Position noise of agents | $U(\boldsymbol{B}(c^i, 0.2))$ |
| Noise in yaw angle of agents | $U(-30°, 30°)$ |

TABLE 4.5: Comparison with baselines and ablation models over different testing missions with 3 obstacles (ASR ↑ /ATS ↓).

| Model | 1D1T | 2D1T | 3D1T | 1D2T | 2D2T | 3D2T | 1D3T | 2D3T | 3D3T |
|---|---|---|---|---|---|---|---|---|---|
| IPPO | **51%**/1510 | 75%/827 | 64%/906 | **25%**/2014 | 48%/1295 | 47%/1272 | **15%**/1876 | 34%/1431 | 31%/1521 |
| MA-POCA | 48%/1165 | 68%/923 | 69%/659 | 24%/1236 | 41%/1257 | 48%/1246 | 14%/1139 | 22%/1389 | 33%/1469 |
| POMA (ours) | 43%/958 | **80%/658** | **81%/539** | 17%/1325 | **67%/1066** | **71%/948** | 10%/1435 | **41%/864** | **43%/903** |
| No AEO | **57%/691** | 76%/725 | 76%/951 | **30%/1200** | 43%/1388 | 50%/1541 | **17%/1375** | 25%/1529 | 35%/1575 |
| No MCA | 31%/1358 | 79%/852 | 74%/681 | 17%/1682 | 62%/1254 | 60%/1053 | 8%/1613 | 47%/1528 | 39%/1122 |
| No ACL | 1%/1024 | 65%/1587 | 41%/1613 | 1%/1021 | 45%/2264 | 25%/2158 | 0%/N.A. | 36%/2650 | 15%/2347 |

TDL $\epsilon$ when $\epsilon$ is larger, the mean distance between the drones and targets is further in the $L$ direction. The light intensity randomization provides generalization to visual perception.

### 4.5.5 Main results

Table 4.5 lists the performance of our trained model in comparison with baseline methods and ablation methods over different testing missions with 3 obstacles. Their performance is evaluated with metrics including ASR and average time steps (ATS) over 500 episodes. More results from training and testing are provided in supplementary materials.

#### 4.5.5.1 Baseline comparison

In all multi-drone missions, POMA consistently exhibits the highest ASR and lowest ATS, outperforming MA-POCA and IPPO. This can be greatly attributed to its superiority in addressing the sparse rewards and in capturing the effects of variable neighbors. IPPO is better in single-agent missions but with the highest ATS

TABLE 4.6: Performance comparison in large-scale missions (ASR ↑ /ATS ↓).

| Model | 6D3T6O | 6D6T6O | 6D9T6O |
|---|---|---|---|
| IPPO | 7.0%/2750 | 2.8%/3153 | 0.4%/3407 |
| MA-POCA | 17.4%/2879 | 6.2%/3137 | 1.2%/3311 |
| POMA(ours) | **23.6%/1304** | **8.6%/1536** | **2.4%/1702** |

since only individual contributions are considered. However, IPPO doesn't encourage collaboration as their success rate decreases in 3-drone missions compared to 2-drone missions. This is because of the limited room space when more drones are involved. On the contrary, MA-POCA and POMA can handle collaborations in constrained spaces since their success rate increases when more drones are involved.

#### 4.5.5.2 Ablation study

From the ablation study, we can observe that the AEO module improves the POMA framework in multiagent missions as the performance of POMA decreases a lot without the AEO module. The attention mechanism considers neighbors with scores and hence improves collaborations. But the AEO module downgrades the performance in single-agent missions since the model achieves the best in single-agent missions without the AEO module. It is the side-effect of introducing an extra Transformer network. Without the MCA module, the POMA shows worse performance in single-agent missions, which verifies the effectiveness of the designed MCA module. In No ACL training, we fix $\epsilon = 0.2$. From the results, the ACL module significantly addresses the sparse reward issue and improves training efficiency as illustrated in the supplementary material.

#### 4.5.5.3 Qualitative analysis

The training curves regarding the ASR are illustrated in Fig. 4.5. From the curves, the IPPO shows higher learning efficiency but POMA obtains the highest success rate even under a higher TDL. Without the ACL module in the sparse reward space, it is hard to find the optimal policy as observed in the POMA w/o ACL curve. Compared to the baseline MA-POCA, all designed modules have contributed to the improvement of POMA.

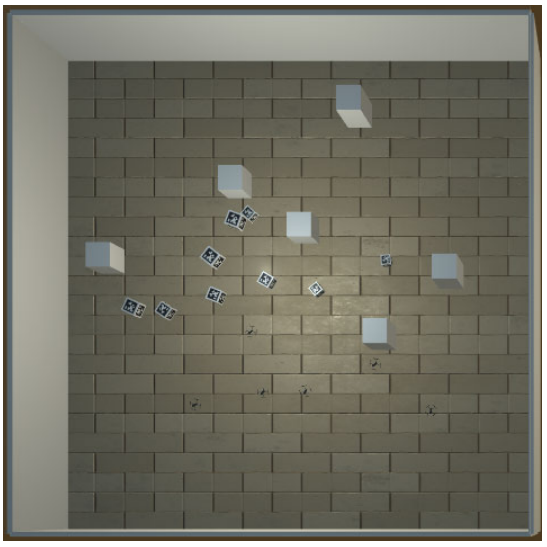


FIGURE 4.7: The 6D9T6O mission in a $10 \times 10 \times 3m$ room.

We further test our model and baseline models over extended large-scale missions, including 6 drones-3 targets -6 obstacles (6D3T6O), 6D6T6O, and 6D9T6O in a $10 \times 10 \times 3m$ room (see an example of 6D9T6O in Fig. 4.7). The performance is listed in Table 4.6. The search performance of POMA decreases when more targets are involved and a larger search space with more obstacles is considered. However, POMA still achieves the best ASR in all large-scale missions. It is noticed that, for the 6D9T6O mission, it is quite challenging to find all targets with limited visual perception even with POMA.

#### 4.5.5.4 Quantitative analysis

The snapshots from the simulation are illustrated in Fig. 4.6. From (a) to (b), Agent 2 sees the two targets and navigates toward them, while Agent 3 is searching for other targets in front, and Agent 1 stays behind. From (c) to (d), Agent 2 finds the last target when Agent 1 is moving forward to searching. To avoid collisions, Agent 1 moves back to provide space for Agent 2 to navigate to the final target. These maneuvers show significant collaborations and the advantages of using a visual drone swarm to search targets when prior information is not available.

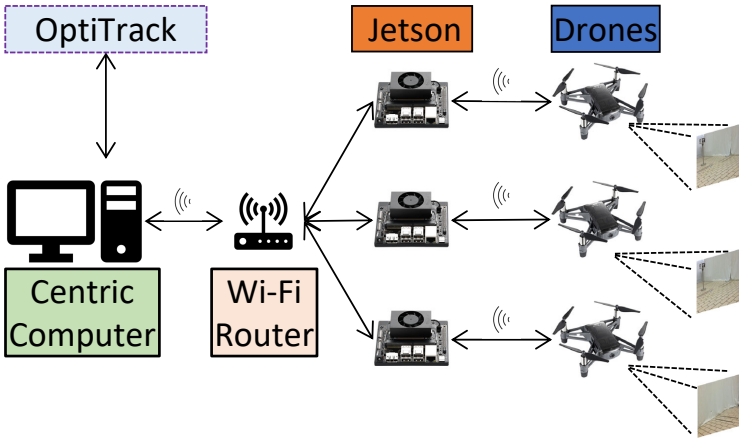
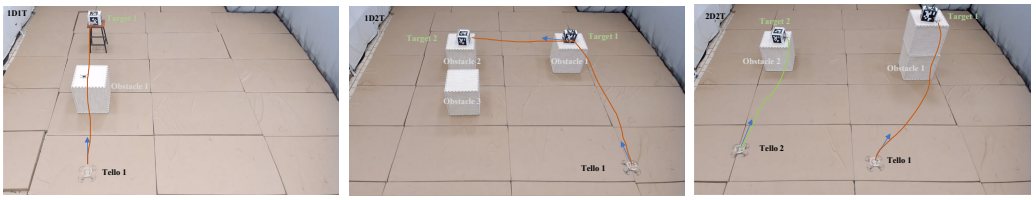



FIGURE 4.8: The physical implementation framework.



(a) (b) (c)

FIGURE 4.9: Snapshots of physical experiments in different missions. The trained policy model shows strong generalization capability even only trained in a 3 drones vs. 3 targets scenario. (a) 1 drone vs. 1 target; (b) 1 drone vs. 2 targets; (c) 2 drones vs. 2 targets.

## 4.6 Physical experiments

### 4.6.1 Settings

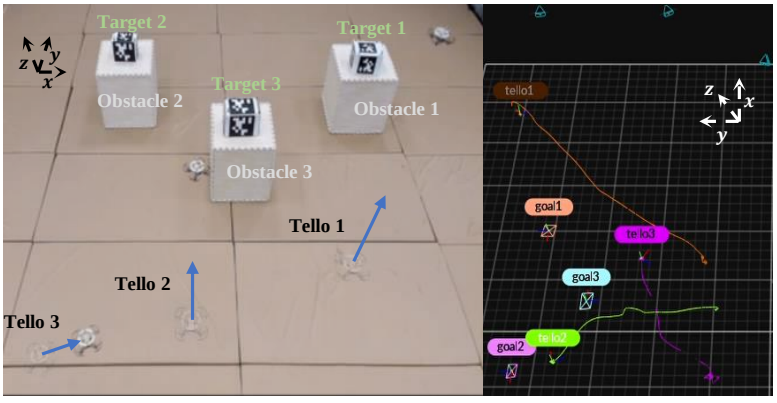
We validate the Sim2Real and generalization capabilities of the trained policy through extensive physical experiments in an $8m \times 8m \times 5m$ flight cage. The policy model is exported in ONNX format and loaded with ONNX runtime package in Python. The Tello Edu drones with onboard cameras are controlled from decentralized edge computers via WiFi communication. To label if the target is found, the status of targets is broadcast from a centric computer. The positions of OOIs are captured with a motion capture system and streamed from the centric computer. Implementation details are provided in the supplementary materials. We tested our model in 1D1T, 1D2T, 2D2T, and 3D3T missions.

### 4.6.2 Physical implementation

The policy model is trained in the simulation (2D2T mission) and deployed over several edge computers. The framework of the control is illustrated in Fig. 4.8.

Each edge computer (Jetson Orin NX) is connected to the WiFi hotpot from the Tello Edu drone (AP mode with different names) and meanwhile wire-connected to the TP-Link router with different fixed IP addresses (192.168.0.121-123). The centric computer is wire-connected to the OptiTrack system and streams the pose information via WiFi connected to the TP-Link router (192.168.0.100).

The Robotic Operating System (ROS) Melodic is deployed on all computers, with the centric computer as the master node. The edge computer subscribes to the pose topic from the centric computer and the video stream from the connected drone. The obtained information is further processed and passed into the loaded policy model with the ONNX runtime package on the edge computer. The output actions drive the Tello Edu drones to search and navigate to targets with the Tello Edu SDK 2.0. To label the status of targets, a *target-status* topic (vector) is created and maintained at the centric computer side and updated by all drones. If one target is found (distance to the drone is within 5cm), the status of the target is labeled with 1 and its position is set to infinity. When all targets are found, the drones stop flying. This can avoid repetitive searching.




(a) (b)

FIGURE 4.10: (a) Physical experiments on 3 drones vs. 3 targets (3D3T) mission. The initial positions are labeled with directions. (b) Trajectories of 3 Tello drones from the motion capture system. Their trajectories show significant collaborative behaviors as Tello 3 attempts to navigate to Target 1 considering Tello 2 already found Target 3 and Target 2. Meanwhile, Tello 1 tries to find other hidden targets.

### 4.6.3 Main results and discussion

Fig. 4.10 illustrates the performance of our trained model in a typical 3D3T mission. Three drones fly to search for targets while avoiding obstacles with real-time video stream (see Fig. 4.10 (a)). Their trajectories show significant collaborative behaviors as Tello 3 attempts to navigate to Target 1 when Tello 2 already found Target 3 and Target 2 (see Fig. 4.10 (b)). Meanwhile, Tello 1 tries to find other hidden targets. These collaborations improve search and navigation efficiency, especially in time-critical missions such as search and rescue. Figures 4.9 illustrate trajectories of drones in different missions (1D1T, 1D2T, and 2D2T), demonstrating the effectiveness of Sim2Real with the trained model in different missions. Compared to traditional approaches, the DRL-based policy can be seamlessly applied to various missions without fine-tuning. The trained policy has showcased outstanding adaptiveness and scalability in different scenarios when a variable number of targets and agents are involved.

Note that in the simulation, the target disappears when touched by drones, while the target is still observable in physical experiments when it is labeled "found". This definitely affects the search performance of drones (the drone may hover some time in front of the target) even though we designed the mechanism to avoid repetitive searching (set the found target's position to infinity). To address this issue, we can add some rules in the loop to guide the drone. In addition, the unstable position stream from the motion capture system also affects the search behavior as we can observe that some drones may hesitate to move at the corners of the room. However, the search policy still shows the resilience to finish the task and can be extended to various missions. More physical experiments are provided in the supplementary video.

## 4.7 Conclusion

This chapter presents the POMA framework, an advanced solution that leverages MADRL for the complex problem of CMTSN. Our proposed method incorporates novel adaptive curriculum learning and mixed individual-group credit assignment mechanisms that balance individual and group contribution in sparse reward environments. Simultaneously, the attention mechanism in POMA refines variable local

observations, leading to significant enhancements in collaborations and scalability. Experimental results conducted on drone simulations demonstrate that our model attains superior performance over baseline methods. We further deploy the trained model over a visual drone swarm and conduct physical tests in different missions. Real-world flight experiments in various missions demonstrate the effectiveness and generalization capability of our approach.

The framework's ability to balance individual and group contributions in sparse reward environments, coupled with its demonstrated effectiveness in drone swarm management for real-world applications, underscores its potential to enhance efficiency, decision-making, and adaptability across different diverse and critical fields when collaborative behaviors are required. In future work, the ease of communication loss and human guidance in the loop will be investigated to improve the performance of our policy in more complex scenarios.

# Part II

# Vision-based learning in adversarial environments

# Chapter 5

# Learning drone detection and tracking in adversarial environments

In this part, the vision-based learning in adversarial environments is studied to enhance the safety and security of drones operations. In the adversarial environment involved multi-drone systems, the agents are often in opposition to each other, competing for resources or trying to avoid capture. The multi-pursuit evasion problem is explored to avoid physical attacks in the adversarial environments. This involves drones avoiding being chased by multiple intelligent pursuers using visual perception. The main objectives have been changed from traditional navigating through static or predicted obstacles to employ high-agility maneuvers to evade the pursuers. Real-world scenarios include avoiding dynamic threats in airspace or during missions where adversarial agents are present. To tackle this problem, a semi-direct learning method, combining object detection and tracking with reinforcement learning control, is adopted.

This chapter investigates real-time multi-drone detection and tracking for visual drones in adversarial environments. Real-time multi-object detection and tracking are primarily required for intelligent multi-drone systems. This chapter presents a whole life cycle multi-drone detection and tracking approach for collaborative

drone pursuit-evasion operations, incorporating parameter search and edge acceleration techniques. Specifically, to address the single-class drone detection limitation of existing drone datasets, we first collect a new dataset "ICG-Drone" from various environments and then establish a performance benchmark with different models, such as YOLOv5, YOLOv8, and Swin Transformer. Based on the outstanding performance regarding accuracy, inference speed, etc., the selected YOLOv5s is further fine-tuned with a genetic algorithm, which achieves 14.8% / 3.6% improvement over 2 drone classes and 3 drone classes, respectively, in terms of mean average precision (mAP). Moreover, we develop an edge-accelerated detector and tracking system Drone-YOLOSORT focusing on "evader" and "pursuer" drones using TensorRT and deliver a ROS package for modular integration, which can be easily applied in a multi-drone system for recognizing friends and non-friends. Our system is able to reach about 24.3 FPS during inferencing, fulfilling the criteria of real-time drone detection at 20 FPS. The project is available at `https://github.com/NTU-ICG/multidrone-detection-tracking.git`.

## 5.1 Introduction

The application of drones has been growing rapidly in different fields [250], especially for multi-drone systems that use visual perception to perform autonomous swarm operations such as collaborative target search [103, 251] and multi-pursuit evasion [252, 253] where the evader aims to reach a destination while multiple pursuers seek to track and intercept the evader. The drone swarm operations have shown significant advantages in system resilience and efficiency. However, as low-altitude airspace becomes increasingly congested with aerial vehicles of various and often unidentified origins, the imperative for real-time detection and tracking becomes critical. Current advances in object detection and multi-object tracking (MOT) techniques [254–256], principally driven by deep convolutional neural networks (CNNs) and Transformer[257], provide a foundational framework to tackle this challenge. While both object detection and MOT have seen separate advancements, their intersection becomes critically important in dynamic, resource-constrained applications like drone navigation [258]. Specifically, drones engaged in pursuit evasion require real-time, accurate, and computationally efficient solutions, making the integration of optimized object detection and MOT algorithms an open

research challenge. The computational overhead of running intricate neural networks in flight can lead to inefficiencies, making it crucial to find an optimized trade-off between accuracy and computational resource consumption.

To address these challenges, this chapter presents a whole life cycle multi-drone detection and tracking approach for pursuit-evasion operations, incorporating parameter search and edge acceleration techniques. Specifically, we first collect a new drone dataset "ICG-Drone" from indoor and outdoor environments and then establish a model benchmark with different object detection models. Instead of using classical hyperparameter tuning methods such as random or grid search [259, 260], the model is further improved with a genetic algorithm (GA). We further develop an edge-accelerated detection and tracking system focusing on multiple "evader" and "pursuer" drones using TensorRT and deliver a ROS package Drone-YOLOSORT for convenient modular integration. Our model is implemented on the Jetson Xavier NX platform, aiming for edge acceleration and real-time inference on camera feeds from the pursuer drones (see Fig. 5.1). We introduce three evaluation metrics that consider inference speed and GPU occupancy with mean average precision (mAP), effectively serving as the criterion for model selection. Our system is highly compatible with existing navigation algorithms, ensuring dynamic adaptability. This synergy markedly bolsters the effectiveness of drone swarm strategies in both pursuit and evasion scenarios. The main contributions of this chapter are summarized as follows.

1) We collect and develop a new "ICG-Drone" dataset from various environments including indoors, outdoors, and internet sources, filling a gap in existing datasets tailored for multi-drone detection and tracking. We further benchmark multi-drone detection performance with various models considering mAP, GPU occupancy, and inference speed.

2) We employ a parameter search method using genetic algorithm (GA) coupled with edge computing techniques that existing drone detection methods seldom consider. This innovative combination optimizes multi-drone detection and tracking, adeptly addressing the computational constraints intrinsic to drone platforms.

3) We integrate state-of-the-art object detection with optimized multiple object tracking (MOT) algorithms into a modular ROS package. This seamless consolidation offers an invaluable resource specifically for drone swarms in pursuit-evasion

FIGURE 5.1: The multi-drone detection and tracking outdoors with edge accelerator. All micro drones are tracked even with low-resolution video streams.

operations.

The remainder of this chapter is organized as follows. Section 5.2 summarizes related works to this chapter; Section 5.3 elaborates on the "ICG-Drone" dataset in detail; Section 5.4 describes our multi-drone detection and tracking approach; Section 5.5 provides the experiment details and results of our benchmarks and Drone-YOLOSORT system. Section 5.6 concludes this chapter and future works.

## 5.2 Related work

Object detection remains a pivotal research area within the domains of deep learning and computer vision. Fundamentally, it serves the dual purpose of localizing and classifying object instances within an image. Existing methods fall primarily into two categories: one-stage and two-stage methods. One-stage methods such as YOLO [261] and SSD [10] prioritize inference speed, performing object classification and localization in a single forward pass of the neural network. This makes them particularly suitable for real-time applications. On the contrary, two-stage methods such as Faster R-CNN [7] and EfficientDet [262] allocate separate stages for proposing candidate object regions and classifying them, often leading to higher detection accuracy, but at the cost of computational efficiency. Recently, Vision Transformers (ViT) [116], originally designed for image classification, have been adapted for object detection [117]. While they showcase competitive results by leveraging the self-attention mechanism to process visual data in a non-local

manner, a significant drawback of ViTs is their computational intensity, which poses challenges for real-time inference, especially on resource-limited platforms. Performance evaluations across these methods are commonly executed using the mean average precision (mAP) metric, benchmarked against standardized datasets like COCO [263] and Pascal VOC [264]. In contrast to object detection, which is concerned with the identification and classification of objects in individual frames, MOT [255] focuses on the temporal aspect of object instances across sequential frames. The core objective is to maintain identity consistency for each tracked object throughout a video sequence. This involves a cascade of tasks that range from pixel-level motion detection and region-of-interest (ROI) localization to segmenting different moving objects within the same frame.

Recently, drone detection and tracking have attracted much attention in the field of anti-UAV techniques to ensure the safety of air traffic. "Anti-UAV" was introduced as a new dataset and benchmark for unmanned aerial vehicle (UAV) tracking, which contains 318 RGB-T video pairs, each containing an RGB video and a thermal video, and over 580k manually annotated bounding boxes [76]. Various DJI and Parrot UAVs were recorded in day and night conditions at a frame rate of 25 FPS. An extended dataset "Real-World" and semi-automated labelling were developed in [265] for drone detection applicable to industrial cameras, but they were restricted to a binary classification task. Y. Zheng et al. [266] proposed "Det-Fly", another dataset consisting of 13,271 images of the DJI Mavic drone, each frame having a resolution of 3840 $\times$ 2160, sampled at 5 FPS. The images are manually annotated by professionals and cover four types of environmental backgrounds: sky, field, urban, and mountain. Det-Fly's main purpose was to serve as a benchmark dataset to evaluate various object detection methods. Another study was done by Daniel et al. [130] on drone detection using YOLOv3 with transfer learning on the NVIDIA Jetson TX2. The author prepared 1,435 drone images including hexacopters, quadcopters, and UAVs for training. The YOLOv3 with custom-trained weights was able to detect drones on the image, but had limited success in deploying the trained model onto the Jetson TX2 due to the low computing power of the device. A small drone dataset "SUAV-DATA" covering all-weather and multi-scenario was constructed in [267] and a Transformer-enhanced YOLOv5 model was proposed to detect drones without considering the tracking performance. An anti-UAV dataset called "DUT Anti-UAV" that contains both

TABLE 5.1: Details of video acquisition

| Time (24H) | Weather | Device | Resolution (FPS) |
|---|---|---|---|
| 28 Oct 2021, 11-12 | Indoors | Galaxy S8+ | 3840 × 2160, 60 |
| | | GoPro Hero 3 | 1920 × 1080, 30 |
| 30 Dec 2021, 10-11 | Cloudy | Galaxy S8+ | 3840 × 2160, 60 |
| | | DJI Tello Camera | 1280 × 720, 30 |
| 11 Feb 2022, 9-10 | Cloudy | GoPro Hero 3 | 1920 × 1080, 30 |
| | | DJI Tello Camera | 1280 × 720, 30 |

TABLE 5.2: Comparison of existing UAV datasets with ours

| Dataset Name | Number | Conditions | Frame Rate | Classes | Tasks |
|---|---|---|---|---|---|
| Anti-UAV [76] | 318 RGB-T videos | Outdoors, Day and Night | 25 FPS | 1 | Detection; tracking |
| Real-World [265] | 55539 images | Outdoors, Multi-scenario | 640 × 480 | 1 | Detection |
| Det-Fly [266] | 13,271 images | Outdoors, Sky, Field, Urban, Mountain | 5 FPS/3840 × 2160 | 1 | Detection |
| Daniel et al. [130] | 1,435 images | Outdoors, Sunny and cloudy | Not specified | 1 | Detection |
| SUAV-DATA [267] | 10,000 images | Outdoors, All-weather, Multi-scenario | Not specified | 1 | Detection |
| DUT Anti-UAV [77] | 10,000 images<br>20 videos | Outdoors, Multi-scenario | 1080 × 1920 | 1 | Detection; tracking |
| ICG-Drone (ours) | 1,823 images<br>5 videos | **Outdoors and indoors**, Multi-senario | 1920 × 1080 | **3** | Detection; tracking |

detection and tracking subsets was developed in [77] and different detectors and trackers are benchmarked in this dataset.

However, the datasets covered in the previous work only cover a single drone or one class of drone. One of the requirements of collaborative pursuit-evasion operations is to classify and detect different classes of drones at the same time. The similarities between different types of drones pose additional challenges to detecting and tracking multiple targets. Meanwhile, flight operations in indoor environments suffer from light reflection and exposure. Hence, the existing datasets cannot be used in our work, and a new dataset of multiple types of drones has to be built up from scratch. Our newly acquired dataset captures both indoor and outdoor environments to enhance the generalization capability. There is also little research done on the ROS package development of multi-drone recognition using object detection and tracking systems on edge deployment. This work seeks to address these issues by developing a whole life cycle multi-drone detection and tracking system.

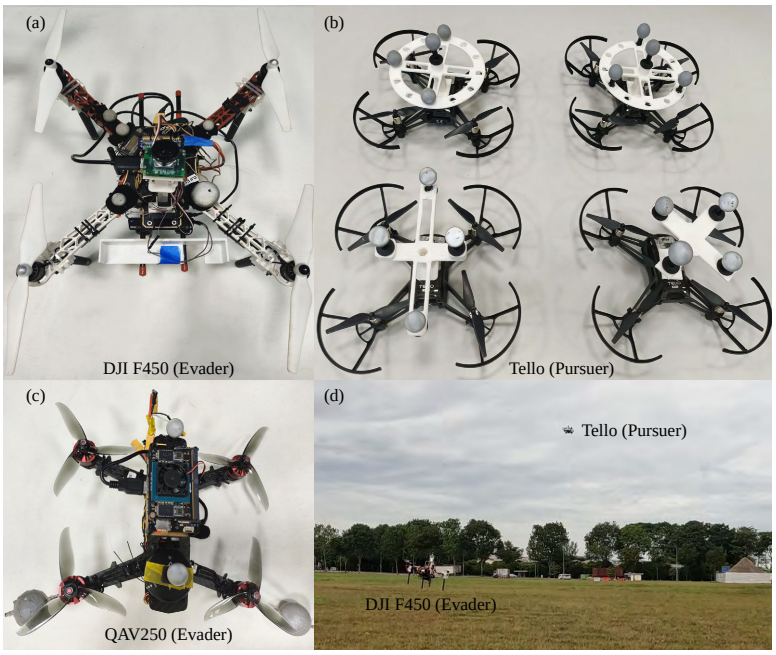



FIGURE 5.2: Types of drones in the dataset for detection and tracking. (a) DJI F450 evader drone; (b) Tello pursuer drone; (c) QAV250 evader drone; (d) View from the Tello in the pursuit-evasion scene.

## 5.3 ICG-Drone dataset

### 5.3.1 Data collection

To the best of our knowledge, existing open-source datasets are notably deficient in accommodating the unique challenges presented by indoor and outdoor drone detection and tracking. Furthermore, no publicly available datasets currently offer categorization among different types of drones, a crucial feature for enabling intelligent collaborative pursuit-evasion operations based on drone types. To address this gap, we undertake a comprehensive video acquisition process, followed by manual annotation to construct a targeted training dataset. Detailed parameters of the video acquisition process for both indoor and outdoor settings are listed in Table 5.1.

Three different cameras were used during the video acquisition process: a Galaxy S8+ model phone camera (12MP, 1/2.55" sensor, f/1.7 aperture with OIS, FOV of 70°), a GoPro Hero 3 (5MP, f/2.8 aperture, FOV of 170°), and a DJI Tello drone camera (5MP, FOV of 82.6°). RGB video acquisition was done in both NTU's UAV laboratory (indoor) and at an open field opposite JTC MedTech Hub, Boon

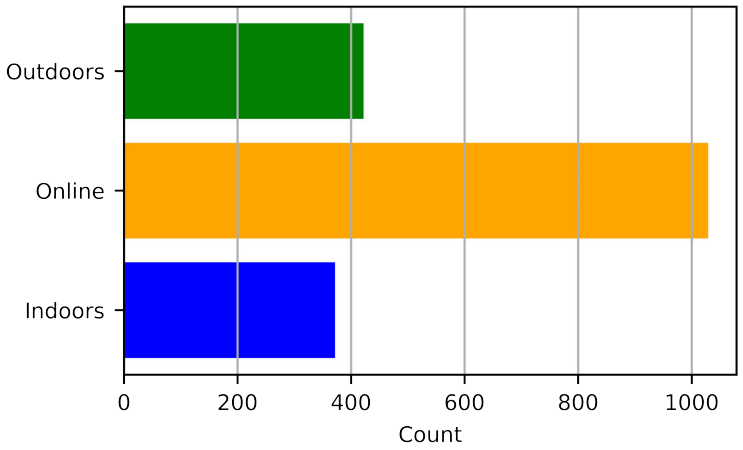



FIGURE 5.3: Distribution of the dataset in different sources with number count.

Lay, Singapore (1°19'26.3"N, 103°42'38.6"E). The footage includes three specific types of drones for added granularity: the DJI Tello drone (pursuer), the DJI F450 drone (evader), and the QAV250 (evader), as illustrated in Fig. 5.2. A typical pursuit-evasion scenario is also demonstrated in Fig. 5.2(d). The Tello is generally smaller and used for pursuit tasks, while the latter two are the targeted evaders in different scenarios. To accomplish the collaborative pursuit task, Tello drones are required to differentiate their friends from the target evader.

To augment our original video acquisition, we also integrated several videos sourced from the Internet that feature DJI Tello (Tello), DJI F450 (F450) and QAV250 (QAV) drones, thereby enriching our dataset's diversity and robustness. These videos were converted into individual image frames using OpenCV, with a frame sampled every 100th frame to ensure adequate coverage without redundancy. Post-extraction, we filtered the dataset to remove frames that did not feature either type of drone; however, a subset of images without any drone presence was intentionally retained to improve the model's ability to minimize false positives. Considering there are only two types of drones in the pursuit-evasion scenario, we provide two datasets for training and testing, namely "ICG-Drone-2c" which only includes Tello and F450, and "ICG-Drone-3c" which includes all types of drones. The final curated dataset comprises 1,575 image frames for "ICG-Drone-2c" and 1,823 images for "ICG-Drone-3c", all resized to a resolution of 1920 × 1080 pixels for consistency. The distribution of the full dataset "ICG-Drone-3c", segmented by data sources, is illustrated in Fig. 5.3 where the numbers (count) of images are counted. We also provide five self-collected videos for tracking tasks, including indoor and outdoor

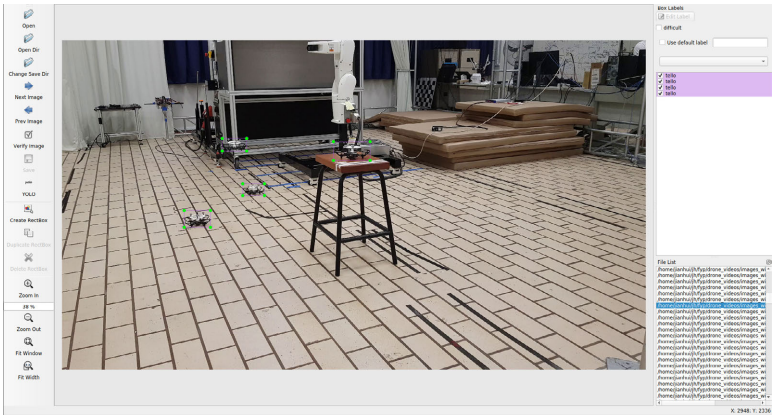


FIGURE 5.4: Annotated image using LabelImg of 4 Tello drones.

scenarios. Our "ICG-Drone" dataset is compared with existing UAV datasets in Table 5.2, and it is the only one providing multi-classes and indoors and outdoors scenarios, which is imperative for the multi-pursuit evasion mission.

### 5.3.2 Data annotation

Data annotation was performed using LabelImg [268]. The label file contains the class (Tello, F450 and QAV) and box coordinates in normalized XYWH format (x, y, width and height) for YOLO series model training. A definition of YOLO format is listed in [268]. LabelImg is a graphical image annotation tool that allows for efficient and precise image labeling and visualization. This can reduce human error in the long run when labeling ground-truth data. LabelImg also supports annotations to be exported to YOLO Darknet TXT format. Figs. 5.4 and 5.5 show the frames annotated in different scenarios using LabelImg. To conduct the training and validation with Swin Transformer [117] and TMDet [269] on our datasets, the YOLO format is further converted to COCO format [270] where the annotations are combined into train and validation JSON files with different object IDs. In this work, both the YOLO format and the COCO format are provided in the dataset for subsequent training and validation.

### 5.3.3 Data splitting

As discussed in the previous section, the full dataset comprises 1,823 images featuring Tello, F450 and QAV drone labels. Conventional practice suggests an 8:2

FIGURE 5.5: Annotated image using LabelImg of DJI F450 and Tello drone.

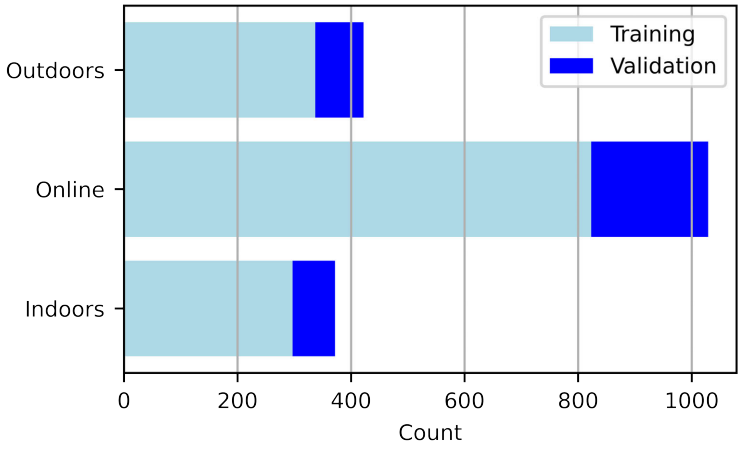



FIGURE 5.6: Barplot of the dataset after split.

ratio for dividing the dataset into training and validation sets, where the training set serves as the primary data for model training, and the validation set is used to assess the model's predictive accuracy. Such data splitting is a standard procedure to prevent model overfitting and ensure robust performance in real-world applications. However, our dataset presents an uneven distribution of image frames from various sources, as indicated in Fig. 5.3. To address this, we employed an 8:2 split from each individual source, ensuring a more balanced representation across the dataset. This strategy is visualized in Fig. 5.6, and it results in 1,457 images allocated for the training set and 366 images set aside for the validation set.

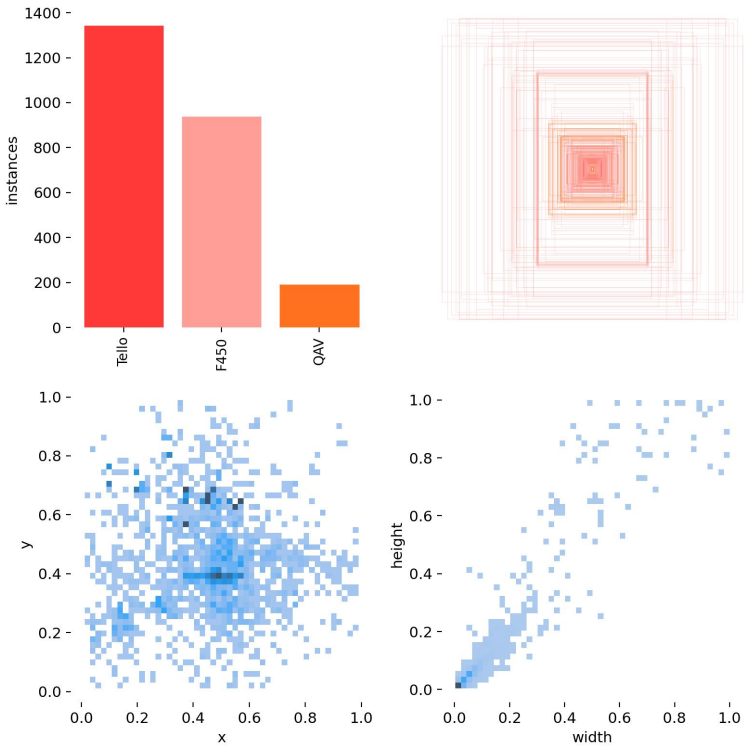



FIGURE 5.7: Label frequency and label variation of XYWH.

### 5.3.4 Data analysis

An exploratory data analysis (EDA) is conducted to gain insights into the statistical characteristics of the dataset. As shown in Fig. 5.7, the dataset maintains a balanced distribution of instances between Tello and F450 drones but fewer QAV drones. Furthermore, the XY coordinates of the bounding boxes are widely distributed, albeit with a noticeable cluster near the center of the images. When it comes to label size, our analysis reveals significant clustering at smaller values. More than half of the bounding boxes occupy less than 5% of the total image frame, classifying them as small objects, a category notoriously challenging for accurate detection. More data analysis can be found on our project website.

## 5.4 Methodology

Despite the growing research on drone detection [76, 266], notably scarce are studies specifically focusing on multi-class drone detection in dynamic scenarios such

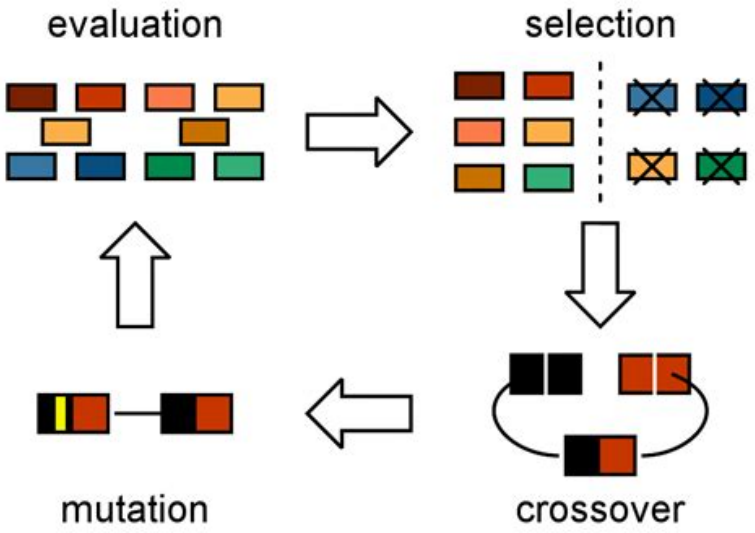



FIGURE 5.8: Genetic algorithm process for parameter search.

as pursuit-evasion operations. Within the limited works that tackle drone detection, the training models generally employ empiric parameter settings, lacking the rigor of hyperparameter optimization. This omission is particularly striking, given that fine-tuning through hyperparameter search could significantly enhance model performance. To this end, our work pioneers a multi-class drone detection model tailored for pursuit-evasion operations, emphasizing the critical role of hyperparameter search in achieving optimal performance.

### 5.4.1 Evaluation metrics

The classic metrics are used to evaluate the model performance: precision, recall and mean Average Precision (mAP). Precision is used for evaluating the accuracy of the model, while recall is used to evaluate the model's ability to detect every positive sample. For object detection, the Intersection Over Union (IoU) score is used to return the intersection between the bounding box of the predictions and the ground truth. A bounding box with an IoU score greater than the minimum IoU threshold (usually 0.5) is counted as a positive prediction, and vice versa. Precision and recall are defined as:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{5.1}$$

where TP, FP, and FN are true positive, false positive, and false negative cases, respectively. In the object detection domain, average precision (AP) and mAP are better metrics to evaluate the performance of models. The AP score represents the area under the precision-recall curve and can be calculated with a weighted sum of the precision score at each minimum IoU threshold and the corresponding recall score. The mAP is the mean of the AP scores for each class.

$$AP = \sum_{k=0}^{n-1} (recall[k] - recall[k+1]) \cdot precision[k] \tag{5.2}$$

$$mAP = \frac{1}{C} \sum_{i=1}^{C} AP_i \tag{5.3}$$

where $k$ is the recall interval sequence and $C$ is the number of classes. In this work, both mAP50 and mAP50-95 are used for evaluation, which follows the COCO evaluation metric for object detection [263]. The mAP50 represents the AP score with a minimum IoU of 0.5, and the mAP50-95 represents the mean of the AP scores with the IoU threshold from 0.50 to 0.95 with steps of 0.05.

Beyond the classical metrics, inference speed and GPU memory usage are considered in this work as the model is to be deployed on the onboard processor to perform real-time inference for drones.

### 5.4.2 Hyperparameter tuning with GA search

Hyperparameters are critical in determining the performance of designed neural network models and are often specified manually by the developer. Hyperparameter tuning is necessary to yield the best performance for the model when too many hyperparameters are involved. Classical hyperparameter tuning methods like random or grid search [259] can be intractable quickly for several reasons. The high dimensionality of the hyperparameter search space can lead to a large amount of computing resources required. The computational cost to evaluate the fitness at each point can be expensive, adding to the computing workload. Besides, the correlations among the hyperparameter dimensions are not known, which means that a brute force strategy has to be conducted to yield the best set of parameters. Existing drone detection methods seldom consider large-scale hyperparameter fine-tuning. In this case, using a genetic algorithm (GA) can be relatively cost-efficient

to obtain the optimal parameters. However, the initial guess for GA can heavily affect the final results and performance. Hence, in this chapter, the best set of hyperparameters obtained using random search is initialized as the initial guess for the hyperparameter evolution. The number of epochs is set to 10 epochs every generation with a batch size of 64. The steps of the hyperparameter evolution are illustrated in Fig. 5.8.

For each generation, all children will undergo an evaluation based on their fitness function, and the five best-performing children are selected in the selection phase. For YOLO models, there is no crossover, and YOLO models strictly follow an elitist genetic algorithm process where a random offspring from the selection phase is selected as the single parent for the next generation. Then, a mutation operation with an 80% probability and a 0.04 variance was used to create new offspring for the next generation. The offspring with the highest fitness will be saved every generation, and the cycle repeats for the new generation. The purpose of a fitness function in the genetic algorithm is to evaluate how close a given solution is to the optimum solution. A well-designed fitness function should be easily understandable, computationally efficient, and capable of accurately quantifying the suitability of a solution to the problem at hand. Ideally, higher scores should correlate with better candidates, while lower scores should indicate less desirable options. To identify the optimal parameter set for our work, we employ a fitness function that incorporates weighted performance metrics, as detailed in Section 5.4.1. Specifically, the fitness function is formulated as:

$$\begin{aligned} Fitness = & w_1 \cdot mAP50 + w_2 \cdot mAP50-95 \\ & + w_3 \cdot Precision + w_4 \cdot Recall \end{aligned} \tag{5.4}$$

where $\{w_i \mid i \in [1:4]\}$ are the metric weights. In the fine-tuning of selected YOLO models, we allocate the weights as follows: $w_1 = 10\%$ for mAP50 and $w_2 = 90\%$ for mAP50-95. We omit weights for precision ($w_3$) and recall ($w_4$) because they are not considered primary metrics for the object detection task in this work.

### 5.4.3 Multi-drone tracking

Deep SORT is chosen for the multi-drone tracking pipeline in this work as it is one of the best approaches for multiple object tracking due to its speed and robustness.

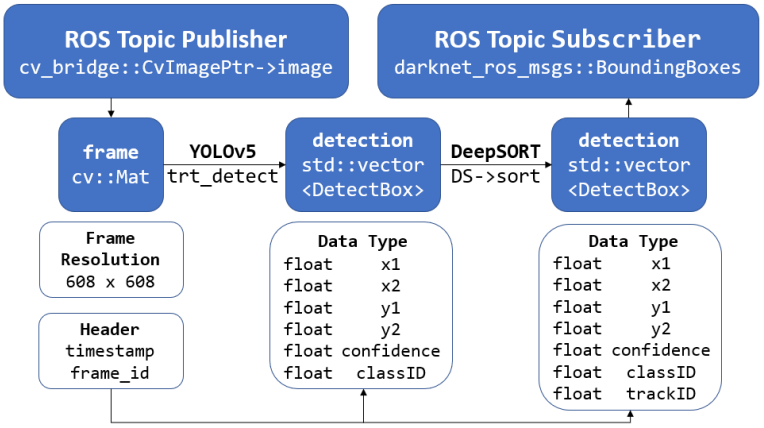



FIGURE 5.9: Proposed ROS package pipeline for Drone-YOLOSORT.

Deep SORT integrates a deep appearance-based metric derived from convolutional neural networks, the Hungarian algorithm [271], and Kalman filters, resulting in higher reliability for problems such as the change in viewpoints of non-stationary cameras and occlusion. Deep SORT has excelled on the MOT16 challenge, bagging the best score in each of the metrics. Deep SORT has also managed to achieve 25–50 FPS with the usage of modern GPUs [272], making it a potential candidate for tracking the drone trajectory in real-time. Pre-trained weights for Deep SORT are available from the Deep SORT GitHub repository[1] and are trained on the MARS dataset, which is a large-scale video-based person reidentification dataset [273]. In this work, we integrate the fine-tuned selected YOLO model into the Deep SORT to achieve satisfactory performance for multi-drone detection and tracking at the same time.

### 5.4.4 ROS package

To enable real-time multi-drone detection and tracking in a ROS environment, we implement a ROS package called Drone-YOLOSORT that integrates a selected hyperparameter-tuned YOLO model with the Deep SORT tracking algorithm. This package is designed to process real-time video streams from onboard cameras, as illustrated in Fig. 5.9. In this architecture, Drone-YOLOSORT subscribes to a designated ROS topic that feeds in video streams. Each incoming image frame is analyzed by the YOLO model for initial object detection. The detection results are subsequently refined by Deep SORT, which assigns unique identification tags

[1] https://github.com/nwojke/deep_sort

TABLE 5.3: Settings of Colab

| OS | CPU | GPU | RAM | OpenCV |
|---|---|---|---|---|
| Ubuntu 22.04 LTS x86_64 | Intel Xeon @ 2.299GHz | Tesla T4 16GB | 16GB | 4.8.0 |
| **Cuda** | **Python** | **Pytorch** | **Torchvision** | **MMDetection** |
| 11.8 | 3.10.12 | 2.1.0+cu118 | 0.16.0 | 3.2.0 |

TABLE 5.4: Settings of workstation

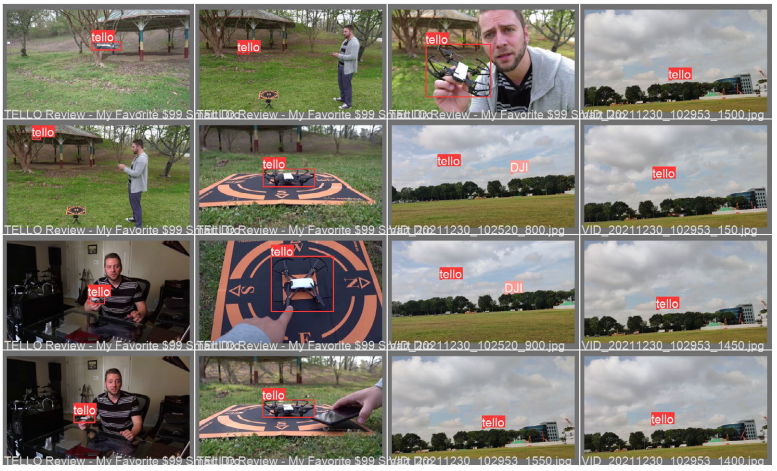
| OS | CPU | GPU | RAM | OpenCV |
|---|---|---|---|---|
| Ubuntu 20.04 LTS x86_64 | AMD RYZEN 5900X @ 3.7GHz | RTX 3090Ti 24GB | 64GB | 4.8.1 |
| **Cuda** | **Python** | **Pytorch** | **Torchvision** | **MMDetection** |
| 11.7 | 3.9.18 | 2.0.1+cu117 | 0.15.2 | 3.2.0 |

and bounding boxes to each detected drone. This adds a layer of stability and robustness to the identification process, allowing for differentiation even among drones of the same class. These processed results are then published as an output ROS topic, serving as observations for a pursuer or evader agent to make the control decision.

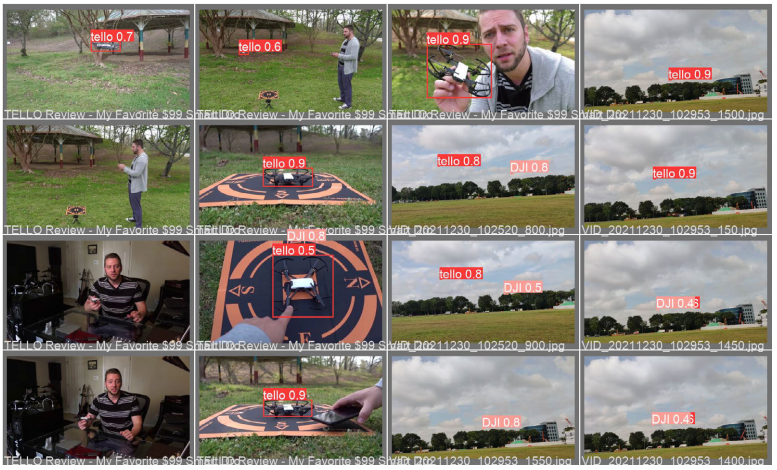
## 5.5 Experiments and results

### 5.5.1 Implementation settings

To objectively compare the performance of baseline object detection models, all training is conducted in Google Colab with Python Jupyter and the local workstation with GPU RTX 3090Ti. To deploy the Swin-Transformer [117] and RTMDet [269], the MMDetection project [274] is forked and adjusted to conduct the customized training with our ICG-Drone dataset. The hardware and software used for model training are listed in Table 5.3 and Table 5.4. Training images are resized to a resolution of 640 × 640 pixels, consistent with recommended best practices, and the batch size is set to 64 to maximize the batch size that the hardware can allocate. Note that for Swin-Transformer and RTMDet, we used the default batch size of 16. Default hyperparameters (see Table 5.6 for YOLOv5) are used for



(a) Ground truth with annotation



(b) Detection results

FIGURE 5.10: Predictions of "ICG-Drone" dataset with the baseline YOLOv5s model. The baseline model is insufficient to detect all small Tello drones.

these baseline models for benchmarking. Other configurations can be found in our GitHub project. To facilitate robust experiment tracking and model management, Weights and Biases (wandb), a widely used machine learning operation (MLOps) platform, is employed. The inference is conducted in Jetson NX (the configuration is listed in Table 5.5) to examine the real-time capability of models.

### 5.5.2 Benchmark with baseline models

In this work, we use the YOLOv5 series [275], YOLOv8 series [276], Swin-Transformer [117], and RTMDet [269] as our baseline models, due to their advanced and robust performance in object detection tasks. The models are pretrained on the COCO

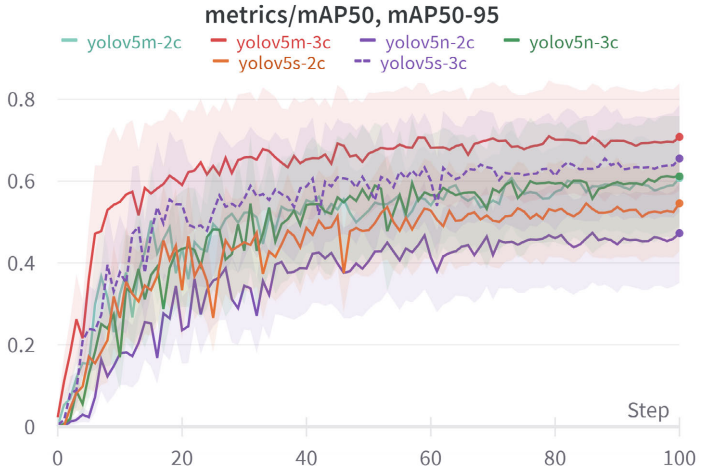



FIGURE 5.11: YOLOV5 comparisons of mean of mAP50 and mAP50-95.

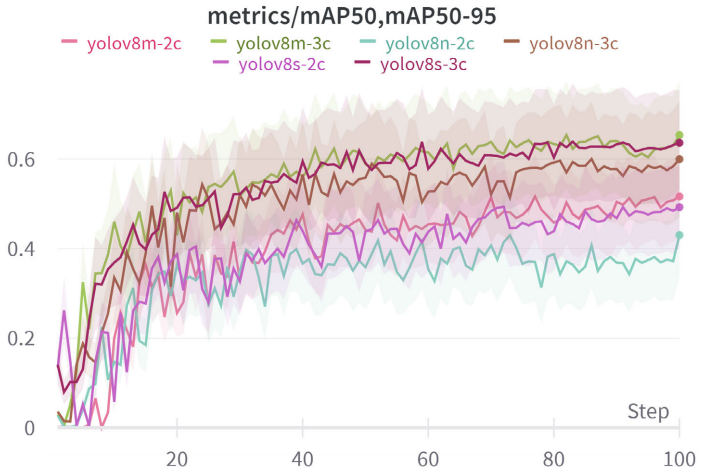



FIGURE 5.12: YOLOV8 comparisons of mean of mAP50 and mAP50-95.

TABLE 5.5: Edge computer setting for Drone-YOLOSORT

| **Requirements** | **Version** | **Requirements** | **Version** |
|---|---|---|---|
| Jetson Xavier NX | L4T 32.5.1 | JetPack SDK | 4.5.1 |
| Python3 | 3.7.9 | TensorRT | 7.1.3.0 |
| PyTorch | 1.8.0 | torchvision | 0.9.0 |
| torch2trt | 0.3.0 | onnx | 1.4.1 |
| opencv-python3 | 4.5.3.56 | protobuf | 3.17.3 |
| scipy | 1.5.4 | ROS | Noetic |

2017 dataset [263] and then our training dataset is used for training the models for 100 epochs. Table 5.9 shows details and training performance of these models, varied by model size. The default YOLOv5 parameters are listed in Table 5.6. We first

TABLE 5.6: Default hyperparameters of YOLOv5s

| **Hyperparameter** | **Value** | **Remarks** |
|---|---|---|
| lr0 | 0.01 | initial learning rate (SGD=1E-2, Adam=1E-3) |
| lrf | 0.1 | final OneCycleLR learning rate (lr0 * lrf) |
| momentum | 0.937 | SGD momentum/Adam beta1 |
| weight_decay | 0.0005 | optimizer weight decay 5e-4 |
| warmup_epochs | 3.0 | warmup epochs (fractions ok) |
| warmup_momentum | 0.8 | warmup initial momentum |
| warmup_bias_lr | 0.1 | warmup initial bias lr |
| box | 0.05 | box loss gain |
| cls | 0.5 | cls loss gain |
| cls_pw | 1.0 | cls BCELoss positive_weight |
| obj | 1.0 | obj loss gain (scale with pixels) |
| obj_pw | 1.0 | obj BCELoss positive_weight |
| iou_t | 0.2 | IoU training threshold |
| anchor_t | 4.0 | anchor-multiple threshold |
| fl_gamma | 0.0 | focal loss gamma (efficientDet default gamma=1.5) |
| hsv_h | 0.015 | image HSV-Hue augmentation (fraction) |
| hsv_s | 0.7 | image HSV-Saturation augmentation (fraction) |
| hsv_v | 0.4 | image HSV-Value augmentation (fraction) |
| degrees | 0.0 | image rotation (+/- deg) |
| translate | 0.1 | image translation (+/- fraction) |
| scale | 0.5 | image scale (+/- gain) |
| shear | 0.0 | image shear (+/- deg) |
| perspective | 0.0 | image perspective (+/- fraction), range 0-0.001 |
| flipud | 0.0 | image flip up-down (probability) |
| fliplr | 0.5 | image flip left-right (probability) |
| mosaic | 1.0 | image mosaic (probability) |
| copy_paste | 0.0 | image mixup (probability) |
| mixup | 0.0 | segment copy-paste (probability) |

TABLE 5.7: Performance metrics for various models on ICG-Drone-2c

| Model | T4 | | | | | 3090Ti | | | | | Average | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | precision | recall | mAP50 | mAP50-95 | speed(ms)(↓) | precision | recall | mAP50 | mAP50-95 | speed(ms)(↓) | precision(↑) | recall(↑) | mAP50(↑) | mAP50-95(↑) |
| yolov5n | 0.748 | 0.535 | 0.609 | 0.364 | 3.4 | 0.756 | 0.545 | 0.594 | 0.351 | **1.0** | 0.752 | 0.540 | 0.602 | 0.358 |
| yolov5s | 0.915 | 0.547 | 0.657 | 0.410 | 4.2 | 0.781 | 0.569 | 0.663 | 0.429 | 1.5 | **0.848** | 0.558 | 0.660 | 0.419 |
| yolov5m | 0.816 | 0.666 | 0.728 | 0.481 | 7.3 | 0.810 | 0.663 | 0.724 | 0.491 | 2.4 | 0.813 | **0.664** | **0.726** | **0.486** |
| yolov8n | 0.742 | 0.464 | 0.561 | 0.357 | **3.3** | 0.606 | 0.479 | 0.523 | 0.338 | 1.7 | 0.674 | 0.472 | 0.542 | 0.347 |
| yolov8s | 0.699 | 0.525 | 0.596 | 0.406 | 7.8 | 0.791 | 0.451 | 0.583 | 0.402 | 1.9 | 0.745 | 0.488 | 0.589 | 0.404 |
| yolov8m | 0.736 | 0.589 | 0.630 | 0.425 | 5.9 | 0.771 | 0.532 | 0.609 | 0.424 | 3.1 | 0.754 | 0.561 | 0.620 | 0.424 |
| SwinT-t | 0.346 | 0.449 | 0.580 | 0.294 | 16.2 | 0.366 | 0.471 | 0.571 | 0.301 | 8.2 | 0.356 | 0.460 | 0.576 | 0.298 |
| RTMDet-t | 0.518 | 0.501 | 0.554 | 0.342 | 22.1 | 0.483 | 0.516 | 0.497 | 0.315 | 10.4 | 0.501 | 0.509 | 0.526 | 0.329 |

benchmark the performance of baseline models from the YOLO series. The mean of mAP50 and mAP50-95 of the YOLO models during the training process is illustrated in Figs. 5.11 and 5.12. The overall metrics for the baseline models over the "ICG-Drone-2c" and "ICG-Drone-3c" datasets are summarized in Table 5.7 and Table 5.8 respectively. Surprisingly, the YOLOv5 series outperforms other baseline

TABLE 5.8: Performance metrics for various models on ICG-Drone-3c

| Model | T4 | | | | | 3090Ti | | | | | Average | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | precision | recall | mAP50 | mAP50-95 | speed(ms)(↓) | precision | recall | mAP50 | mAP50-95 | speed(ms)(↓) | precision(↑) | recall(↑) | mAP50(↑) | mAP50-95(↑) |
| yolov5n | 0.837 | 0.627 | 0.738 | 0.469 | **3.1** | 0.781 | 0.721 | 0.759 | 0.464 | **1.1** | 0.809 | 0.674 | 0.749 | 0.467 |
| yolov5s | 0.907 | 0.710 | 0.789 | 0.521 | 4.5 | 0.823 | 0.725 | 0.786 | 0.525 | 1.5 | 0.865 | 0.718 | 0.788 | 0.523 |
| yolov5m | 0.887 | 0.755 | 0.831 | 0.584 | 7.1 | 0.899 | 0.767 | 0.838 | 0.577 | 1.9 | **0.893** | **0.761** | **0.835** | **0.581** |
| yolov8n | 0.819 | 0.646 | 0.727 | 0.486 | 3.6 | 0.824 | 0.662 | 0.728 | 0.471 | 1.3 | 0.822 | 0.654 | 0.728 | 0.479 |
| yolov8s | 0.793 | 0.693 | 0.771 | 0.532 | 5.5 | 0.908 | 0.637 | 0.764 | 0.522 | 2.0 | 0.851 | 0.665 | 0.768 | 0.527 |
| yolov8m | 0.932 | 0.681 | 0.781 | 0.536 | 7.7 | 0.863 | 0.713 | 0.773 | 0.534 | 3.4 | 0.898 | 0.697 | 0.777 | 0.535 |
| SwinT-t | 0.461 | 0.553 | 0.697 | 0.395 | 16.2 | 0.461 | 0.553 | 0.692 | 0.395 | 7.7 | 0.461 | 0.553 | 0.695 | 0.395 |
| RTMDet-t | 0.550 | 0.587 | 0.613 | 0.401 | 24.3 | 0.546 | 0.585 | 0.608 | 0.392 | 9.8 | 0.548 | 0.586 | 0.611 | 0.397 |

TABLE 5.9: Model training details in workstation

| **Model** | **Parameter Size (↓) (M)** | **Weight Sizes (↓) (MB)** | **Training Time (↓) (mins)** | **GPU Memory Allocated (↓) (%)** |
|---|---|---|---|---|
| YOLOv5n | 1.9 | 3.8 | 9.88 | 39.85 |
| YOLOv5s | 7.2 | 14.4 | 11.03 | 63.22 |
| YOLOv5m | 21.2 | 42.1 | 21.82 | 95.82 |
| YOLOv8n | 3.2 | 6.5 | 9.50 | 46.50 |
| YOLOv8s | 11.2 | 22.6 | 12.22 | 67.31 |
| YOLOv8m | 25.9 | 52.1 | 27.67 | 88.09 |
| SwinT-t | 48 | 219.6 | 33.27 | 82.67 |
| RTMDet-t | 4.8 | 23.1 | 20.23 | 82.16 |

models in both the "ICG-Drone-2c" and "ICG-Drone-3c" datasets. This can be due to the similarities between different types of drones and the fewer classes compared to the COCO2017 dataset. Hence, the improvement of YOLOv8 is not significantly reflected in our dataset. In addition, by adding one more type of drone (QAV), the detection performance of all models increased a bit. This can be attributed to the extra features brought by QAV data. Meanwhile, the Swin-Transformer-tiny (SwinT-t) and RTMDet-tiny (RTMDet-t) show much worse detection performance and much slower inference speed compared to the YOLO series in our dataset.

Although the YOLOv5m model exhibits superior accuracy, it suffers from lower inference speed, clocking in at around 246 ms on the Tesla K80 GPU (previous Colab settings), or approximately 4 FPS, while the YOLOv5s can reach 63 ms. Given that these tests are conducted on a Tesla K80 GPU, performance is expected to decline on the edge processor Jetson Xavier NX. Therefore, YOLOv5s is selected as the optimal model, balancing both accuracy and inference speed. While YOLOv5s demonstrates some capability in drone detection, as illustrated in Fig. 5.10, it falls short in detecting drones in challenging conditions such as small size or complex backgrounds. This shortcoming underscores the need for further hyperparameter tuning.

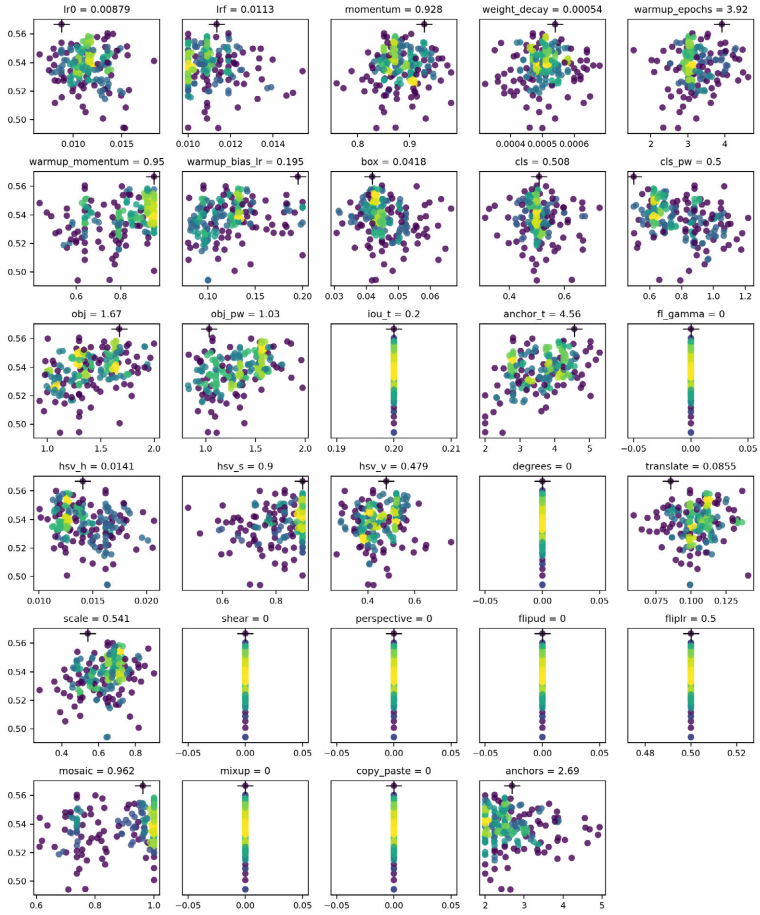



FIGURE 5.13: The evolution process of hyperparameter with GA.

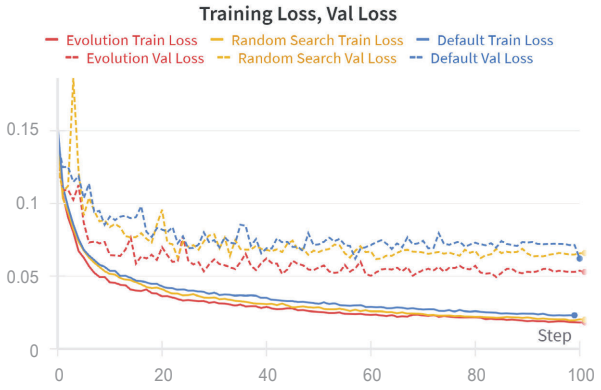



FIGURE 5.14: Training and validation loss curves after hyperparameter tuning.

### 5.5.3 Performance with parameter search

The selected YOLOv5s model is further re-trained for 100 epochs with three hyperparameter sets of default, random search, and GA search with a batch size of 64. A total of 200 generations were run on the workstation. The GA was able to

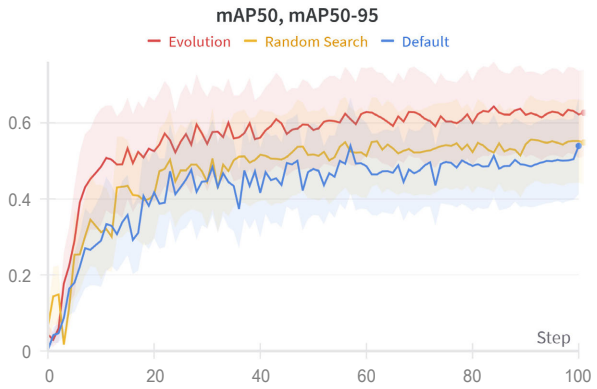



FIGURE 5.15: Parameter search comparisons with mean of mAP50 and mAP50-95.

TABLE 5.10: Performance of YOLOV5s with hyperparameters of default, random search, and GA search (ICG-Drone-2c/ICG-Drone-3c)

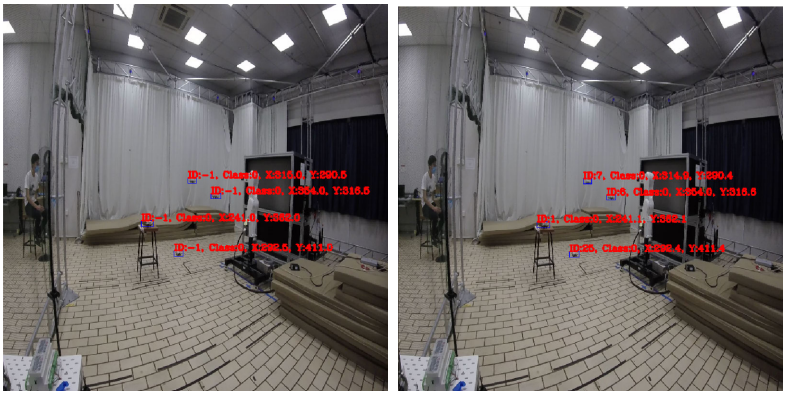
| **Hyperparameter Search** | **Precison(↑)** | **Recall(↑)** | **mAP50(↑)** | **mAP50-95(↑)** |
|---|---|---|---|---|
| Default | 0.781 / 0.823 | 0.569 / 0.725 | 0.663 / 0.786 | 0.429 / 0.525 |
| Random Search | 0.833 / 0.810 | 0.582 / 0.708 | 0.662 / 0.758 | 0.447 / 0.435 |
| GA Search | **0.884 / 0.865** | **0.700 / 0.749** | **0.761 / 0.814** | **0.525 / 0.539** |
| **Percentage increase (%)** | 13.2 / 5.1 | 23.0 / 3.3 | 14.8 / 3.6 | 22.4 / 2.7 |

TABLE 5.11: Performance of YOLOV5s with hyperparameters of GA search in various public datasets (Default/GA search)

| **Dataset** | **Precison(↑)** | **Recall(↑)** | **mAP50(↑)** | **mAP50-95(↑)** |
|---|---|---|---|---|
| Real-World[265] | 0.872 / **0.912** | 0.788 / **0.806** | 0.857 / **0.871** | 0.428 / **0.438** |
| Det-Fly [266] | 0.961 / **0.986** | 0.925 / **0.981** | 0.940 / **0.986** | 0.603 / **0.693** |
| DUT Anti-UAV [77] | 0.970 / **0.976** | 0.897 / **0.902** | 0.929 / **0.936** | 0.609 / **0.611** |

find the optimal set of hyperparameters at the $75^{th}$ generation. The evolution plot is illustrated in Fig. 5.13, with one subplot per hyperparameter. The y-axis of each subplot depicts the fitness score while the x-axis denotes the hyperparameter values. Yellow indicated a greater concentration within the clusters. The value of each respective hyperparameter with the highest fitness score is reported at the top of each subplot. These values combine into the set of hyperparameters to use for model deployment. The box loss, object loss and classification loss are summed up and plotted during training and validation in Fig. 5.14. The mean of mAP50 and mAP50-95 are also plotted in Fig. 5.15. These results indicate that the model with the evolved hyperparameters (Evolution) has the lowest train and validation loss, outperforming the other two models with random search and default parameters.

FIGURE 5.16: Procedure for conversion from PyTorch model to engine file.



(a) YOLOv5 result (b) Drone-YOLOSORT result

FIGURE 5.17: Indoor results for Tello drones on Jetson Xavier NX. The Drone-YOLOSORT can track all drones with distinct IDs.

(a) YOLOv5 result (b) Drone-YOLOSORT result

FIGURE 5.18: Outdoor results for detecting both F450 and Tello drones.

Table 5.10 summarises the final results for the three methods. The percentage change for all four metrics is calculated for the evolved model against the default model. In conclusion, the evolved model is the best performer out of the set in both the "ICG-Drone-2c" and "ICG-Drone-3c" datasets, with its performance metrics scoring the best among the models. Hence, the evolved model is chosen for further deployment.

To further validate the robustness of the GA search, we conducted training on the existing public drone datasets with default parameters and the GA search,

respectively. Note that selected drone datasets, namely "Real-World", "Det-Fly" and "DUT Anti-UAV", are annotated in Pascal VOC format. To conduct the training, all annotations are transformed to the YOLO format. Table 5.11 lists the performance of YOLOv5s training on various public datasets with parameters of default and GA search. With GA search, the performance improved significantly on all three datasets, which validates the robustness of proposed method in drone detection.

### 5.5.4 Model deployment with edge accelerator

We integrated the trained YOLOv5s model on the "ICG-Drone-3c" dataset and Deep SORT into one ROS package, and the compiled Drone-YOLOSORT ROS package is further deployed on the Jetson Xavier NX. The Jetson Xavier NX contains a 384-core NVIDIA Volta™ GPU with 48 Tensor Cores, 6 Carmel ARM CPUs, and two NVIDIA Deep Learning Accelerators (NVDLA) engines, which allows the Jetson Xavier NX to run multiple modern neural networks in parallel. TensorRT is utilized to accelerate model inference by analyzing the given neural network architecture and reconstructing the computational graph. To make use of TensorRT to optimize speed, the selected YOLOv5s model is required to be converted into a TensorRT engine file. Fig. 5.16 describes the procedure for conversion from a PyTorch model to an engine file. In essence, the conversion comprises two phases. Firstly, the PyTorch model is converted to a weight file using TensorRTx[2]. Then, TensorRTx rebuilds the entire neural network using TensorRT network definition APIs and generates the engine file. The engine file can then be used to run inference.

TABLE 5.12: Inference time comparison between PyTorch model and TensorRT engine

| **Model** | **Detection (ms)** | **Detection (FPS)** | **Detection with Tracking (ms)** | **Detection with Tracking (FPS)** |
|---|---|---|---|---|
| PyTorch | 120 | 8.3 | 246 | 4.0 |
| TensorRT | 20 | 50.0 | 41 | 24.3 |
| **Speedup ratio** | | ×6 | | ×6 |

[2]https://github.com/wang-xinyu/tensorrtx

The inference results in Jetson Xavier NX for both YOLOv5s and Drone-YOLOSORT are illustrated in Fig. 5.17 and Fig. 5.18. YOLOv5s model returns the bounding boxes for each instance of drone that is detected without ID numbers. This is seen in Fig. 5.17(a) where the class type, center X, and Y of the bounding boxes are returned. With Deep SORT, Drone-YOLOSORT then takes each instance and assigns it a unique ID number. This unique ID number can be further used by the pursuit policy to get the bearings [197] of each pursuer friend and the targeted evader and further drive pursuers to enclose the evader. With our ROS package, the designed pursuit policies can be seamlessly adopted in real-world applications.

Table 5.12 compares the inference speed between the original PyTorch model and the TensorRT engine. The speedup ratio is computed to be 6, implying that conversion from PyTorch to TensorRT accelerates the performance of inferencing by about six times. Drone-YOLOSORT was able to reach about 24.3 FPS during inferencing, fulfilling the criteria for real-time drone detection at 20 FPS, which is suitable for real-world operations.

## 5.6 Conclusion

In this chapter, we propose a whole life cycle multi-drone detection and tracking system designed for drone pursuit-evasion operations. We created a new drone dataset "ICG-Drone", which covers a range of indoor and outdoor scenarios. We benchmarked the performance of several object detection models such as YOLOv5, YOLOv8, and Swin Transformer, and then fine-tuned the most promising model YOLOv5s using a GA search strategy. Our edge-accelerated Drone-YOLOSORT system meets real-time performance criteria with about 24.3 FPS, accelerating inference speed by 6x through TensorRT during inferencing, and the developed ROS package ensures seamless integration into existing drone systems for complex multi-pursuit evasion tasks. For future work, expanding the "ICG-Drone" dataset with more drone types and exploring alternative machine learning architectures could further improve model generalizability and inference speed. This lays the groundwork for more advanced real-time, multi-drone tracking technologies in agile flights.

# Chapter 6

# Learning multi-pursuit evasion in adversarial environments

With the obtained detection and tracking information from Chapter 5, this chapter studies the control policy to learn multi-pursuit evasion for drones. Safe navigation of drones in the presence of adversarial physical attacks from multiple pursuers is a challenging task. This chapter proposes a novel approach, asynchronous multistage deep reinforcement learning (AMS-DRL), to train adversarial neural networks that can learn from the actions of multiple evolved pursuers and adapt quickly to their behavior, enabling the drone to avoid attacks and reach its target. Specifically, AMS-DRL evolves adversarial agents in a pursuit-evasion game where the pursuers and the evader are asynchronously trained in a bipartite graph way during multiple stages. Our approach guarantees convergence by ensuring Nash equilibrium among agents from the game-theory analysis. We evaluate our method in extensive simulations and show that it outperforms baselines with higher navigation success rates. We also analyze how parameters such as the relative maximum speed affect navigation performance. Furthermore, we have conducted physical experiments and validated the effectiveness of the trained policies in real-time flights. A success rate heatmap is introduced to elucidate how spatial geometry influences navigation outcomes. Project website: `https://github.com/NTU-ICG/AMS-DRL-for-Pursuit-Evasion`.

## 6.1 Introduction

Nowadays, autonomous obstacle avoidance has become an indispensable component of drones' safe navigation. Massive obstacle avoidance methods have been proposed for drones' safe operation in complex environments [13, 16, 73, 186, 214, 277]. However, most of them considered static cluttered obstacles, such as trees [45] or buildings [16], whose positions would not change throughout the flight period. Even though some works proposed feasible approaches to avoid simple or ad-hoc dynamic obstacles (such as a moving gate [186], balls being thrown [277, 278], and moving pedestrians [279]), the paths of the obstacles can be easily detected and predicted while in flight. As such, these works only deal with unintelligent and unperceptive obstacles. The drone only needs to perform the required avoidance maneuvers at the predicted time. Furthermore, the self-learning capabilities of intelligent pursuers like birds can push drones to constantly approach their limits of computation and agility, which are intractable to solve with conventional obstacle avoidance approaches. To effectively handle aforementioned threats, drones require higher agility and powerful learning capabilities. Recently, the pursuit-evasion problem for drones has been investigated in the field of intelligent robotics [280–284] which provides valuable insights to improve the maneuverability of drones. However, these works only focused on designing pursuit strategies without considering the safe navigation requirement from the evaders' side. Besides, they assumed that all dynamics of pursuers and evaders were pre-obtained, lacking consideration of evolving capabilities for both sides. Once the agents' behavior changes, the problem needs to be solved from scratch. To achieve PETN, a control policy must be designed to have sufficient agility and long-term evolution ability [285] when performing navigation tasks, which existing traditional methods cannot provide.

DRL has recently attracted much attention for drone applications, such as obstacle avoidance [73], disturbance rejection [216], autonomous navigation [156, 286], task coordination [103, 287], drone racing [25], etc. DRL has shown outstanding performance over traditional numerical methods such as polynomial-based trajectory generation[288] regarding the agility of drones[214]. However, these DRL methods cannot be extended to solve the pursuit-evasion problem, as their policies are trained without the evolution of environments and adversarial agents. To address the PETN problem, especially multi-pursuit evasion with targeted navigation problem (MPETN) (see Fig. 6.1) more efficiently, this chapter proposes

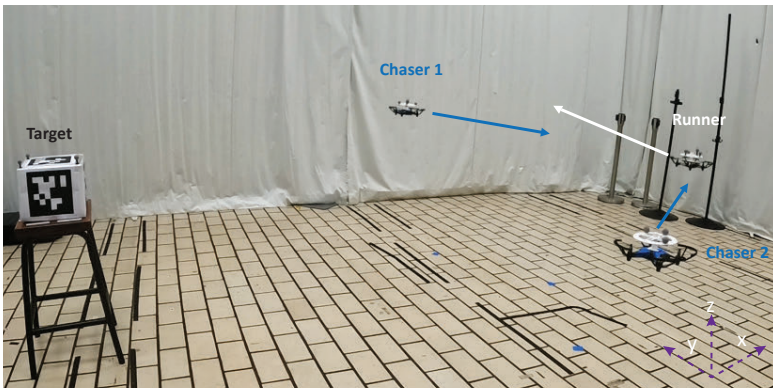



FIGURE 6.1: Evading multiple pursuers with learned policy. The runner (evader) is labeled as white, and the two chasers (pursuers) are labeled as blue. The target is a box with AprilTag.

a novel **A**synchronous **M**ulti-**S**tage **D**eep **R**einforcement **L**earning (**AMS-DRL**) approach that can provide drones with rapid response and adaptive capabilities to evade pursuers.

Motivated by nature-inspired pursuit-evasion phenomena where pursuers and chasers keep learning from each other with steady policy evolution, the proposed AMS-DRL is designed to evolve adversarial agents from multistage deep reinforcement learning, where the chaser drones (pursuers) and the runner drone (evader) are asynchronously trained in a bipartite graph way during different stages. With perfect information on position observation, the control policy for the runner drone is initially trained to reach the target box (the destination of the runner drone) in the cold-start learning stage ($S_0$). During the following stages, the chaser drones and the runner drone are trained against each other with one team fixed with a pre-trained policy from the previous stage; e.g., during Stage 1 ($S_1$), the chaser drones are trained to attack the runner drone while the runner drone is driven with the control policy from $S_0$. AMS-DRL can be seen as an unsupervised learning algorithm toward open-ended evolution [78], which approximates the capability limits of agents consistently only if the adversarial agents reach a Nash equilibrium (NE) [79]. We demonstrate that the resulting policies can provide drones with evasive maneuvers for self-preservation, opening the possibility of using DRL for solving multi-pursuit evasion problems. Our main contributions are summarized as follows:

(1) We formulate the MPETN problem into an adversarial mixed game and present a DRL-based approach involving multiple drones. A novel learning-based approach

named AMS-DRL is proposed to provide drones with sufficient maneuverability and long-term evolution ability to evade multiple pursuers during navigation.

(2) The proposed AMS-DRL is proven to converge to optimal policies based on game theory. Through extensive simulations, we demonstrate the superiority of our navigation policy over baseline methods such as PPO and the artificial potential fields (APF) method against collaborative pursuers.

(3) Sim2Real and physical experiments under different spatial geometry configurations are conducted to verify the effectiveness of the trained navigation policy in real flights.

The rest of this chapter is organized as follows. Section II summarizes existing methods related to our work. Section III provides several preliminary works, including the quadrotor drone dynamics and RL framework. Section IV presents the problem formulation and the RL description of MPETN. Section V describes the AMS-DRL approach in detail. Experiments and results are presented and discussed in Section VI, and we conclude our work in the final section.

## 6.2 Related work

### 6.2.1 Obstacle avoidance

Obstacle avoidance aims to achieve designed objectives (e.g., reaching the destination) while subject to non-intersection or non-collision constraints. Obstacle avoidance is a basic requirement for intelligent drones to fly through cluttered environments. From the perspective of how drones generate control commands, existing obstacle avoidance can be categorized into indirect methods and end-to-end methods. Indirect methods [13, 45, 136, 137, 277] generally divide the obstacle avoidance problem into several subtasks, namely **perception**, **localizing** and **planning**. The states of obstacles, such as shape, position, and velocity, are detected and localized before a maneuver is performed. Once obstacles are located, the drone can generate a feasible path or take action to avoid the obstacles. Within the indirect methods, planning can be further divided into two categories: one is offline methods based on high-resolution maps and pre-known position information, such as Dijkstra's algorithm [141], A$^\star$ [142] and sequential convex optimization [144]; the other is

online methods based on real-time decision-making. Online methods can be further categorized into online path planning [13, 45, 136] and artificial potential field (APF) methods [277, 289]. Currently, due to the advantages of optimization and prediction capabilities, online path planning methods have become the preferred choice for drone obstacle avoidance. For instance, Zhou Boyu *et al.*[13] introduced Fast-Planner for a drone to perform high-speed flight in an unknown cluttered environment. The key contributions are a robust and efficient planning scheme incorporating path searching, B-spline optimization, and time adjustment to generate feasible and safe trajectories for drones' obstacle avoidance. However, these online path planning methods necessitate onboard complex optimization computation to obtain a local or global optimal trajectory. In contrast to online path planning, the APF methods require fewer computation resources and can well cope with dynamic obstacle avoidance using limited sensor information. Falanga *et al.* [277] developed an efficient and fast control strategy based on APF to avoid fast-approaching dynamic obstacles. However, the repulsive forces computed only reach substantial values when the obstacle is very close, which may lead to unstable and jagged behavior. Besides, APF methods are heuristic methods that cannot guarantee global optimization and robustness for drones. Hence, it is not ideal to adopt APF methods for drones to evade pursuers.

Compared to the indirect methods, end-to-end methods [16, 73, 146, 147] exploit reinforcement learning (RL) to map perception to actions directly. Reinforcement learning [290] is a technique for mapping the state (observation) space to the action space in order to maximize a long-term return with given rewards. A typical RL model features agents, the environment, reward functions, action, and state space. The policy model achieves convergent status via constant interactions between the agents and the environment, where the reward function guides the training process. Malik *et al.* [147] presented an end-to-end reinforcement learning approach called NAVREN-RL to navigate a quadrotor drone in an indoor environment with expert data and knowledge-based data aggregation. The reward function in [147] was formulated from a ground truth depth image and a generated depth image. Loquercio *et al.* [16] developed an end-to-end approach that can autonomously guide a quadrotor drone through complex wild and human-made environments at high speeds with purely onboard visual perception and computation. The neural network policy was trained in a high-fidelity simulation environment with massive expert knowledge data. While the aforementioned end-to-end methods provide a

straightforward way to generate obstacle avoidance policies for drones, they require massive training data to obtain acceptable generalization capabilities. Meanwhile, without the evolutionary environment, it is challenging for these designed neural network policies to update their weights to evade intelligent pursuers.

### 6.2.2 Pursuit-evasion game

Pursuit-evasion games (PEGs) have been widely studied in the fields of game theory and control theory. PEGs involve capturing mobile agents (evaders) with one or more pursuers. Back to the 1950s, Isaacs R. provided a comprehensive introduction to the mathematical theory of pursuit-evasion games[291], which is considered a classic reference in this field. Fisac Jaime F. *et al.* [292] investigated the problem of the pursuit-evasion-defense differential game with dynamic obstacles in constrained environments. Xu Fang *et al.* [280] studied the multi-pursuer single-evader PEG with an evader moving faster than the pursuers in a constraint-free environment, and proposed a distributed pursuit algorithm to enable pursuers to form an encirclement and approach the faster evader, which was applied and verified in mobile ground robots. These works provide traditional game/control-theoretic approaches for solving PEG problems in dynamic, constrained, or obstacle-free environments. Most of the existing traditional approaches require simplified dynamics or environment assumptions and are challenging to extend to general pursuit-evasion scenarios.

Recently, RL has been adopted to ease strict assumptions for pursuit-evasion problems [281–284, 293, 294]. A distributed cooperative pursuit strategy is proposed in [282] to address the multiple pursuers-one superior evader problem with RL and efficient communication topology networks. A curriculum learning approach in [283] is developed to train a decentralized multiagent pursuit policy to capture mobile targets. In [284], a target prediction network (TP Net) is integrated into the multiagent deep deterministic policy gradient (MADDPG) to address the target PEG problem of drones in the obstacle-filled environment, where high-quality 3D simulation scenarios are developed. The attention-enhanced neural network was introduced in [293] to address predator-prey games. However, most of these RL-based approaches assume fixed or simple evader behaviors and rarely consider the NE of the PEGs. Moreover, they focus only on the pursuit side and ignore

the safety of evaders. In this chapter, we formulate the safe targeted navigation problem for the evader drone and address it by finding the NE with AMS-DRL.

### 6.2.3 Reinforcement learning with self-play

With self-play [295–297], RL has achieved remarkable success in adversarial environments (or competitive games), such as DeepMind's AlphaZero [297], AlphaStar [298] and OpenAI Five [217]. Self-play uses the agent's current or past "selves" as opponents to train agents to automatically generate curricula, which can be a powerful algorithm for learning behaviors within high-dimensional continuous environments. Silver *et al.* [297] demonstrated the superhuman performance of AlphaZero, a general RL algorithm with self-play, in many challenging perfect-information games (such as chess, shogi, and Go) without domain knowledge. To address large-scale imperfect-information games with RL, Heinrich *et al.* [296] introduced Neural Fictitious Self-Play (NFSP) via a combination of Fictitious Self-Play (FSP) [295] and neural networks, and demonstrated it can approach NE without prior domain knowledge over real-world scale games. Following that, Vinyals *et al.* [298] provided the first Grandmaster level player-AlphaStar in the real-time strategy game StarCraft II with complex, imperfect information and long-time horizons. AlphaStar features a Prioritized Fictitious Self-Play (PFSP) algorithm, which adopts a match-making mechanism to provide a better learning signal instead of uniformly playing against all previous "selves" to improve the training efficiency. In 2019, OpenAI Five [217] first defeated the world champions in the Dota 2 game with proposed self-play experience buffers and surgery techniques, which achieved large-scale distributed learning and long-time training. These methods paved the way for training agents in complicated adversarial environments and provided them with superhuman performance against opponents. Furthermore, OpenAI found that multiagent self-play can induce distinctive emergent strategies toward open-ended growth in complexity [299], which helped solve evolution-required decision-making problems. However, all the aforementioned methods only process zero-sum two-team games where the rewards of two adversarial teams can be flipped, i.e., the sum of rewards for two teams remains zero during training. In fact, it is determined by the inherent property of self-play that exploits "selves" as opponents during training, which means two adversarial teams' observations and rewards must be symmetric.

Nevertheless, pursuit-evasion for drones is not a zero-sum game since the runner drone is required to reach the target while the chaser drones aim to pursue the runner drone. The success of reaching the target will not bring a loss to the chasing team. Meanwhile, the observation spaces for the runner agent and chaser agent are not symmetric since the chaser drones cannot perceive the location of the target. In our work, we extend the capabilities of DRL to address the pursuit-evasion problem for drones via AMS-DRL. AMS-DRL provides a cold-start learning stage for the runner drone to reach the target and asynchronous training for two adversarial teams at multiple stages/phases. In each phase, agents can compete with their opponents driven by the policy from the prior stage/phase. The limitation of self-play is addressed via the asynchronous training scheme toward an approximate NE.

## 6.3 Multi-pursuit evasion with targeted navigation problem

### 6.3.1 Problem formulation

In this section, we introduce the idea of the MPETN problem for drones and formulate it into an adversarial mixed game described with MDPs, which lies the foundation of our control algorithm AMS-DRL. Note that our multi-pursuit evasion problem in this chapter considers one runner drone against a chaser drone swarm with the same speed. The formulated problem and the proposed algorithm can be extended to a large-scale scenario where massive chaser drones of a finite number are involved. We assume that the runner only knows the relative positions of the chasers and the target, while all chasers can obtain the relative positions of the runner and other chasers. The positions of all agents and the target are randomly placed with a uniform distribution in a constrained sample space. The system dynamics are unknown to each agent, i.e., the states cannot be predicted directly by solving system differential functions.

#### 6.3.1.1 MDP in adversarial games

The pursuit evasion problem in this chapter involves two adversarial teams, namely a runner team (single-agent) $\mathcal{M}^r : \langle \mathcal{S}, \mathcal{O}^r, \mathcal{A}^r, \mathcal{P}, \mathcal{R}^r, \gamma \rangle$ and a chaser team (multia-gent) $\mathcal{M}^c : \langle \mathcal{N}, \mathcal{S}, \{\mathcal{O}_i^c\}_{i\in\mathcal{N}}, \{\mathcal{A}_i^c\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}_i^c\}_{i\in\mathcal{N}}, \gamma \rangle$, where $\mathcal{S} := \mathcal{S}^c \cup \mathcal{S}^r$ defines the state space covered by all chaser agents $\mathcal{S}^c$ and the runner agent $\mathcal{S}^r$, $\mathcal{N} = \{1, ..., N\}$ denotes the set of finite $N \geq 1$ chaser agents, $\mathcal{O}_i^c$, $\mathcal{A}_i^c$, and $\mathcal{R}_i^c$ are the observation space, action space, and reward function of the chaser agent $i$, respectively. Let $\mathcal{A} := \mathcal{A}_1^c \times \cdots \times \mathcal{A}_N^c \times \mathcal{A}^r$, then $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ defines the transition probability from state $s_t \in \mathcal{S}$ to next state $s_{t+1} \in \mathcal{S}$ with joint action $a_t \in \mathcal{A}$. The goal of the runner agent is to reach the target $g$ and avoid the attack from the chaser team while the chaser team is trying to crash down the runner agent, i.e., to minimize the cumulative reward of the runner agent. Hence, the pursuit-evasion problem for drones can be formulated as an adversarial mixed game with two adversarial teams $\{\mathcal{M}^r, \mathcal{M}^c\}$. The objective of the pursuit-evasion problem can be described as:

$$\pi_\theta^{r*} = \underset{a_t^r \sim \pi_\theta^r}{\operatorname{argmax}} \mathcal{L}(\pi_\theta^r \mid \pi_{\theta'}^{c*}) \tag{6.1a}$$

$$s.t. \quad \pi_{\theta'}^{c*} = \underset{\{a_{i,t}^c\} \sim \pi_{\theta'}^c}{\operatorname{argmin}} \mathcal{L}(\pi_\theta^r) \tag{6.1b}$$

$$\dot{\boldsymbol{x}} = g(\boldsymbol{x}, \boldsymbol{u}) \tag{6.1c}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max} \tag{6.1d}$$

where $\pi^c := \prod_{i\in\mathcal{N}} \pi_i^c(a_i^c \mid s)$ is the joint chaser policy and $\pi^r$ is the runner policy. Subscripts $\theta$ and $\theta'$ mean the policies $\pi^r$ and $\pi^c$ are parameterized with different neural networks. $g(\,\cdot\,)$ is the dynamics constraint, and $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$ are the input bounds. In the game theory [300], the optimal solution of a game is always achieved at a NE. The NE of the adversarial mixed game $\{\mathcal{M}^r, \mathcal{M}^c\}$ is defined as follows.

**Definition 6.1** (Nash Equilibrium of Adversarial Mixed Game)**.** A Nash Equilibrium of the adversarial mixed game described with MDPs $\mathcal{M}^r : \langle \mathcal{S}, \mathcal{O}^r, \mathcal{A}^r, \mathcal{P}, \mathcal{R}^r, \gamma \rangle$ and $\mathcal{M}^c : \langle \mathcal{N}, \mathcal{S}, \{\mathcal{O}_i^c\}_{i\in\mathcal{N}}, \{\mathcal{A}_i^c\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}_i^c\}_{i\in\mathcal{N}}, \gamma \rangle$ is a joint policy $\pi^* := \{\pi_1^{c*}, \ldots, \pi_N^{c*}, \pi^{r*}\}$, such that for any $s \in \mathcal{S}$ and $i \in \mathcal{N} \cup \{r\}$,

$$V_{\pi_i^*, \pi_{-i}^*}^i(s) \geq V_{\pi_i, \pi_{-i}^*}^i(s), \quad for\ any\ \pi_i \tag{6.2}$$

where $\{r\}$ denotes the runner agent, and $-i$ means all other agents except for agent $i$.

**Theorem 6.1.** *For the adversarial mixed game $\{\mathcal{M}^r, \mathcal{M}^c\}$, there always exists a unique NE, and this unique NE is the optimal policy $\pi_\theta^{r*}$ for the runner agent, i.e., the solution of the pursuit-evasion problem (6.1).*

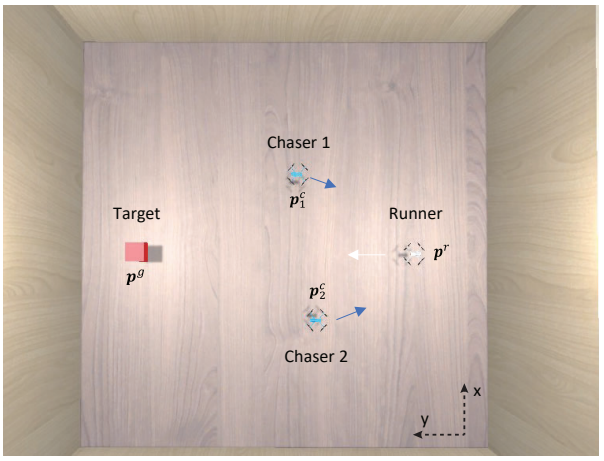
*Proof.* Firstly, we prove the existence and uniqueness of NE of the adversarial mixed game $\{\mathcal{M}^r, \mathcal{M}^c\}$. Since the chaser team $\mathcal{M}^c$ consists of homogeneous agents with a common reward function and same policies, i.e., $\pi^c_{i\in\mathcal{N}} = \pi^c_1$, we can regard the chaser team as one player $c$. Therefore, the adversarial mixed game $\{\mathcal{M}^r, \mathcal{M}^c\}$ shrinks into a two-player mixed game with player $\{c, r\}$. Due to the control constraints (2.3), the action space of the agents is finite. According to **Theorem 1.1** in [300], there always exists an NE for finite strategic games. Suppose there are two NE strategies $\pi^\alpha$ and $\pi^\beta$, based on (6.2), we have $V^i_{\pi_i^\alpha,\pi^*_{-i}}(s) \geq V^i_{\pi_i^\beta,\pi^*_{-i}}(s)$ and $V^i_{\pi_i^\beta,\pi^*_{-i}}(s) \geq V^i_{\pi_i^\alpha,\pi^*_{-i}}(s) \Rightarrow V^i_{\pi_i^\alpha,\pi^*_{-i}}(s) \equiv V^i_{\pi_i^\beta,\pi^*_{-i}}(s)$ at any time. Hence, $\pi^\alpha \equiv \pi^\beta$ holds, and it is the unique NE of the adversarial mixed game.

From **Definition 1**, let $i \in \mathcal{N}$, we have $V^i_{\pi^{c*},\pi^r}(s) \geq V^i_{\pi^c,\pi^r}(s) = -(\mathcal{L}(\pi^r_\theta) - r_T)$ for any $\pi^c$, where $r_T$ is the positive constant terminal reward of the runner agent. It follows that $-V^i_{\pi^{c*},\pi^r}(s) \leq (\mathcal{L}(\pi^r_\theta) - r_T) < \mathcal{L}(\pi^r_\theta)$. Therefore, $\pi^{c*}$ is the desired $\pi^{c*}_{\theta'}$ in (6.1b). Similarly, let $i \in \{r\}$, we have $V^i_{\pi^{r*},\pi^{c*}}(s) \geq V^i_{\pi^r,\pi^{c*}}(s) = \mathcal{L}(\pi^r_\theta \mid \pi^{c*}_{\theta'})$ for any $\pi^r$ under $\pi^{c*}$, i.e., $\pi^{r*}_\theta = \pi^{r*}$. This completes the proof. □

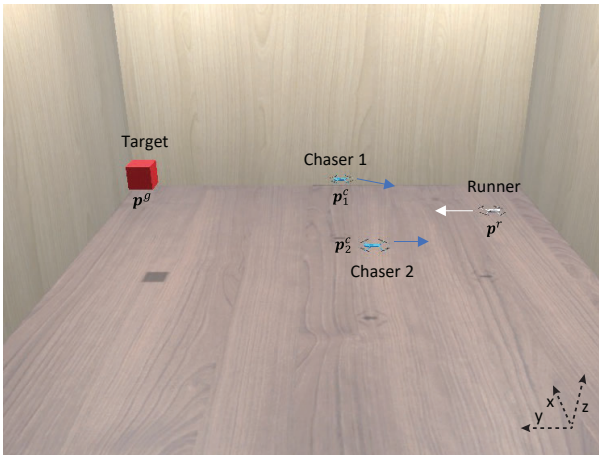
*Remark* 6.1. Note that the terminal reward of the runner agent $r_T$ is assumed to be a positive constant when the runner agent reaches the target. Considering an indoor environment, there is a probability of the runner agent colliding with the wall, which gives a negative reward $r'_T$. In this case, we assign an opposite reward $-r'_T$ to each chaser agent since the runner agent is trained to fly away from the wall during the initial stage and the collision can be attributed to the pursuit of the chaser agents. Therefore, Theorem 6.1 holds even though the terminal state of the runner drone is the wall collision.

### 6.3.2 RL description of MPETN

In this section, we describe the MPETN problem from the RL formulation. The environment setup, state space, observation space, action space, state transition function, and reward function are described in detail.




(a) Top view




(b) Side view

FIGURE 6.2: The simulation environment for the 3D multi-pursuit evasion with targeted navigation scenario, which consists of two blue chaser drones (pursuers) and one white runner drone (evader) flying towards the target (red box). All simulated objects are bound with box colliders ($L \times W \times H$) for collision detection. The chasers aim to crash down the runner while the runner is required to safely reach the target.

#### 6.3.2.1 Environment setup

To train and evaluate the policies of chaser and runner agents, a high-fidelity simulation environment is created using the Unity game engine. Our simulation environment is an obstacle-free, constrained indoor room, which is illustrated in Fig. 6.2. During training, all three drone agents and the red target box will spawn in random locations every episode to prevent any agent from learning undesirable behaviors such as flying toward a fixed spawn point.

The components of each object are outlined by Unity's box collider, and all objects are tagged as target, chaser, and runner, respectively. The wall, floor, and ceiling are tagged as wall colliders. These tags allow for the detection of collisions with each other, with different rewards.

#### 6.3.2.2 State space

Since the drone is an under-actuated system, in a three-dimensional (3D) space, the state of a drone $i \in \mathcal{N} \cup \{r\}$ can be represented as $s_i := [x_i, y_i, z_i, \psi_i]^T \in \mathbb{R}^4$, where $x_i, y_i, z_i$ are the elements of position $\boldsymbol{p}_i$, and $\psi$ is the heading angle. We denote the positions of target, runner, and chasers as $\boldsymbol{p}^g$, $\boldsymbol{p}^r$, and $\boldsymbol{p}^c$, respectively. The global state can be represented as $s := \{\boldsymbol{p}^g, \{s_i\}_{i\in\mathcal{N}\cup\{r\}}\}$.

#### 6.3.2.3 Observation space

In a partially observed scenario, the agents draw actions with a certain probability from observations. To improve the generalization of the trained model, relative positions are considered in the observation space. The observation for the runner agent $o^r$ consists of the relative position towards the target $\boldsymbol{p}^{rg} = \boldsymbol{p}^g - \boldsymbol{p}^r$ and relative positions towards chaser agents $\{\boldsymbol{p}_j^{rc}\}_{j\in\mathcal{N}}$, where $\boldsymbol{p}_j^{rc} = \boldsymbol{p}_j^c - \boldsymbol{p}^r$. The concatenated observation is $o^r = [\boldsymbol{p}^{rg}, \{\boldsymbol{p}_j^{rc}\}_{j\in\mathcal{N}}] \in \mathbb{R}^{(N+1)\times 3}$. For chaser agents $\{c_i\}_{i\in\mathcal{N}}$, they share the positions with each other and can track the position of the runner all the way. Hence, the observation of chaser agent $c_i$ is $o_i^c = \left[\{\boldsymbol{p}_{ij}^{cc}\}_{j\in\mathcal{N}\setminus\{i\}}, \boldsymbol{p}_i^{cr}\right] \in \mathbb{R}^{N\times 3}$, where $\boldsymbol{p}_{ij}^{cc}$ is the relative position from the chaser $c_i$ to the chaser $c_j$, i.e., $\boldsymbol{p}_{ij}^{cc} = \boldsymbol{p}_j^c - \boldsymbol{p}_i^c$ and $\boldsymbol{p}_i^{cr}$ is the relative position from the position of chaser $c_i$ to the position of runner $r$, i.e., $\boldsymbol{p}_i^{cr} = \boldsymbol{p}^r - \boldsymbol{p}_i^c$.

#### 6.3.2.4 Action space

In this work, the action spaces for the runner drone $a^r$ and the chaser drone $a^c$ are the same. The selected actions $a^c = [\hat{v}_x^{B,c}, \hat{v}_y^{B,c}, \hat{v}_z^{B,c}, \hat{\omega}_z^c] \in \mathbb{R}^4$ and $a^r = [\hat{v}_x^{B,r}, \hat{v}_y^{B,r}, \hat{v}_z^{B,r}, \hat{\omega}_z^r] \in \mathbb{R}^4$ are four desired velocity commands generally used in the Robot Operation System (ROS), where $\hat{v}_x^B \in (-v_{max}, v_{max})$, $\hat{v}_y^B \in (-v_{max}, v_{max})$,

$\hat{v}_z^B \in (-v_{max}, v_{max})$ and $\hat{\omega}_z \in (-\omega_{max}, \omega_{max})$ are the desired velocities of the quadrotor in the body frame $O_B$. The action space $\mathcal{A}$ consists of these four continuous actions that drive drones to maneuver in the obstacle-free 3D space.

#### 6.3.2.5 State transition function

Since we consider the high-level controller based on the selected velocity commands, the quadrotor dynamics (1a)-(1c) can be simplified as:

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}) \cdot \boldsymbol{u} + \boldsymbol{\omega} \tag{6.3}$$

where $\boldsymbol{x} = [x, y, z, \psi]^T$, $g(\boldsymbol{x})$ is the transition matrix, and $\boldsymbol{u} = [\hat{v}_x^B, \hat{v}_y^B, \hat{v}_z^B, \hat{\omega}_z]^T$. $\boldsymbol{\omega}$ represents the process noise and uncertainties, which satisfies the Gaussian distribution $p(\boldsymbol{w}) \sim \aleph(0, \boldsymbol{Q})$ with $\boldsymbol{Q} = diag\{\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\psi^2\}$. $\boldsymbol{g}(\boldsymbol{x})$ is given by:

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.4}$$

The simplified system (6.3) is a stochastic continuous system with process noise. Note that even with this simplification, the drone is driven with a full 6-DOF model throughout both training and evaluation phases in simulation, assuming a high-bandwidth controller capable of precisely following these velocity commands.

#### 6.3.2.6 Reward function

The reward functions determine the desired behavior of the agents during the training process. The reward function consists of two parts: one is the existential penalty $\mathcal{R}_P = -w_1/T_{max}$ of the team, where $T_{max}$ is the allowable number of time steps in an episode and $w_1$ is the adjustable weight; the other is the task reward $\mathcal{R}_T$. The total reward of an agent is $\mathcal{R} = \mathcal{R}_P + \mathcal{R}_T$. The existential penalty serves to accelerate the exploration progress, while the task reward serves to guide the drones in completing their tasks.

For runner agents, there are three objectives: reaching the desired target while evading the attacks from the runner and avoiding colliding with the wall. Hence, the task reward for the runner agent is specified as:

$$\mathcal{R}_T^r = \begin{cases} C_1(1 - \|\boldsymbol{p}_T^{rg}\|/\|\boldsymbol{p}_0^{rg}\|), & if\ step\ ends\ at\ T \\ -C_2, & if\ agent\ crashes \end{cases} \tag{6.5}$$

where $C_1$ and $C2$ are positive constants, i.e., $C_1, C2 \in \mathbb{R}^+$. $\boldsymbol{p}_T^{rg}$ and $\boldsymbol{p}_0^{rg}$ are the terminal relative position and the initial relative position. Note that the runner agent crashes with the chasers or wall, receiving the same penalty. The first item is to encourage the runner drone to reach the target.

For chasers, their objectives are to crash down the runner while avoiding colliding with other teammates and the wall. The task reward for the chaser agent $c_i$ is:

$$\mathcal{R}_T^{c_i} = \begin{cases} C_1, & if\ team\ captures\ the\ runner \\ -C_2, & if\ agent\ collides\ with\ the\ wall \\ -C_2\mathcal{H}(d_\epsilon - d_{min}^i), if\ agents\ are\ too\ close \end{cases} \tag{6.6}$$

where $\mathcal{H}(\cdot)$ is a Heaviside step function (unit step function), $d_{min}^i$ is the minimum distance of the chaser $c_i$ with other chasers, i.e., $d_{min}^i = min(\{\|\boldsymbol{p}_{ij}^{cc}\|_2\}_{j\in\mathcal{N}\setminus\{i\}})$ and $d_\epsilon$ is the preset risk threshold.

## 6.4 Asynchronous multistage deep reinforcement learning approach

### 6.4.1 AMS-DRL algorithm

According to Theorem 6.1, to solve the MPETN problem (6.1) becomes to find the NE of the adversarial mixed game $\{\mathcal{M}^r, \mathcal{M}^c\}$. Here, we propose the AMS-DRL approach to approximate the NE of the game. AMS-DRL features a cold-start learning stage and an asynchronous learning stage. The AMS-DRL approach is described in Algorithm 4 in detail.

#### 6.4.1.1 Cold-start learning stage

At the Cold-Start Learning (CSL) stage $S_0$, the objective is to train the runner drone to navigate toward a target. The runner drone will learn to approach the stationary target without colliding with walls. Note that the position of the target is randomly generated during the training process. This stage necessitates an initial navigation policy for the runner drone.

#### 6.4.1.2 Asynchronous learning Stage

Once the runner drone has successfully learned the targeted navigation behavior, multiple chaser drones are added to the scene in the Asynchronous Learning (AL) stage to generate multi-pursuit evasion with targeted navigation behavior. The runner drone is placed in inference mode with an initial policy from the CSL stage while the chaser drones start to train. During inference mode, the runner drone moves according to its previously learned behavior and does not improve upon it. The chaser drones are trained asynchronously until they learn to seek out the runner drone. With their basic fundamental behaviors learned, the chaser and runner drones are trained against each other so that they both learn new strategies while also combating the other side's strategies.

The training sequence of AMS-DRL is illustrated in Fig. 6.3. There are several training phases in the AL stage, denoted by $S_1$ to $S_k$. During each phase, one team is put in training mode while the other is set in inference mode. The operation $mode(\cdot)$ returns the corresponding parameters once the mode of policy is set as training, e.g., $mode(\theta, \theta') = \theta'$ if $\pi^c_{\theta'}$ is under training mode. Once the training policy converges, training for one team halts while that for the other team begins. Let $sr^r$ and $sr^c$ denote the success rate (SR) of the runner team and the chaser team over a window size of $ws$, respectively. The whole training stage ends for both teams when the difference in their success rates converges on a preset threshold $\|sr^r - sr^c\| \leq \eta$, which means an approximate NE. Note that it is a successful task when the runner reaches the target safely, while success for the chasers is to capture the runner or push the runner to walls.

**Theorem 6.2** (Converge Analysis)**.** *Assuming that policies $\pi^r_{i,\theta}$ and $\pi^c_{i,\theta'}$ converge in every stage/phase $S_i$ with policy improvement. Given any threshold $\eta \in (0, 1)$,*

**Algorithm 4:** AMS-DRL

**Start the CSL stage** $S_0$

**Input:** Initial runner policy $\pi^r_{0,\theta_0}$ for the CSL stage.

At the start of training:

Initialize training parameters, such as discount factor $\gamma$, learning rate $\alpha_0$, batch size $bs$;

**for** *episode* $= 0 : maximum\ episodes$ **do**

  Randomly spawn $\boldsymbol{p}^g$, $\boldsymbol{p}^r$ and $\psi^r$ within the constrained space.

  **for** *step* $t = 0 : T$ **do**

    Execute the policy $\pi^r_{0,\theta_t}$

    Collect data $\{s_t, a_t, r_t\}$

    Calculate the objective function $\mathcal{L}(\pi^r_{\theta_t})$

    Update $\theta_t$ via policy gradient as (2.12) and (2.13)

**Output:** Trained runner policy $\pi^r_{0,\theta}$

**Start the AL stage**

**Input:** Pre-trained runner policy $\pi^r_{0,\theta}$, shared initial chaser policy $\pi^c_{1,\theta'_0}$, threshold $\eta$

**for** *Phase* $S_i = S_1 : S_k$ **do**

  **if** *i is odd* **then**

    Set $\pi^c_{i,\theta'}$ as training mode while set $\pi^r_{i,\theta}$ as inference mode, i.e., $\pi^r_{i,\theta} = \pi^r_{i-1,\theta}$

  **else**

    Inverse the mode for $\pi^c_{i,\theta'}$ and $\pi^r_{i,\theta}$

  **for** *episode* $= 0 : converged\ episodes$ **do**

    Randomly spawn the initial global state within the constrained space $s_0 := \{\boldsymbol{p}^g, \{s_i\}_{i\in\mathcal{N}\cup\{r\}}\}$

    **for** *step* $t = 0 : T$ **do**

      Execute the runner policy $\pi^r_{i,\theta_t}$ and the shared chaser policy $\pi^c_{i,\theta'_t}$

      Collect data $\{s_t, a_t, r_t\}$ from trained agents

      Calculate the objective function $\mathcal{L}(mode(\pi^r_{\theta_t}, \pi^c_{\theta'_t}))$

      Update $mode(\theta_t, \theta'_t)$ via policy gradient as (2.12) and (2.13)

    Calculate the success rates $sr^r$ and $sr^c$ over the test $ws$ episodes

    **if** $\|sr^r - sr^c\| \leq \eta$ **then**

      Break

**Output:** The optimal policies, $\pi^{r*}_{\theta}$, $\pi^{c*}_{\theta'}$

*with finite phases $k$, the output of AMS-DRL converges to the optimal policies $\pi^{r*}_{\theta}$, $\pi^{c*}_{\theta'}$ with $\|sr^r - sr^c\| \leq \eta$.*

*Proof.* Let denote the SR of the runner policy $\pi^r_{i,\theta}$ from stage/phase $S_i$ as $sr^r_i \in [0, 1]$, for $i \in \{0, \ldots, k\}$ and the SR of the chaser policy $\pi^c_{i,\theta'}$ from $S_i$ as $sr^c_i \in [0, 1]$, for $i \in \{1, \ldots, k\}$. We have $sr^c_i + sr^r_i = 1$ for $i \in \{1, \ldots, k\}$. In $S_0$, since there is no

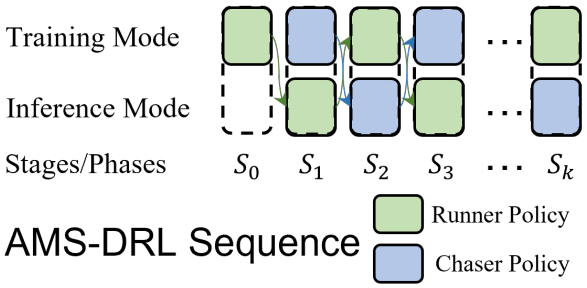



FIGURE 6.3: Illustration of the AMS-DRL training sequence for the shared chaser policy and the runner policy.

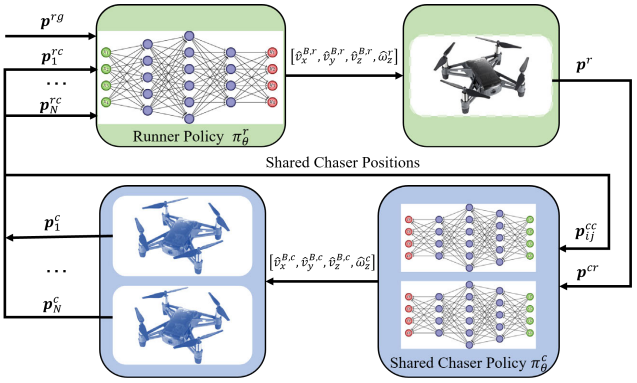



FIGURE 6.4: Illustration of the neural network architecture for chaser agents and the runner agent.

chaser around, $\pi^r_{0,\theta}$ achieves the highest SR $sr^r_0 = sr^r_{max}$. In phase $S_1$, the objective of the chaser policy $\pi^c_{1,\theta'}$ is to bring down the SR of $\pi^r_{1,\theta} = \pi^r_{0,\theta}$. Hence, $sr^r_1 < sr^r_0$ and $sr^r_1 = sr^r_{min}$ since $sr^c_1 = sr^c_{max} = 1 - sr^r_1$ due to the Policy Improvement Theorem. For phase $S_2$, we can easily obtain $sr^r_0 > sr^r_2 > sr^r_1$ since we train the runner policy to achieve better navigation performance. For the following phases $S_i, i = 3, \ldots, k$, similarly we have $1 \geq sr^r_0 > sr^r_2 > \cdots > sr^r_k > \cdots > sr^r_3 > sr^r_1 \geq 0$.

Now, given any threshold $\eta$, we want to find a finite $k$, such that $\|sr^r - sr^c\| \leq \eta$, i.e., $(1-\eta)/2 \leq sr^r_k \leq (1+\eta)/2$. This forms an interval $[0.5 - \eta/2, 0.5 + \eta/2]$ for $sr^r_k$. Since we have a finite sequence of $sr^r_k$ such that $1 \geq sr^r_0 > sr^r_2 > \cdots > sr^r_k > \cdots > sr^r_3 > sr^r_1 \geq 0$, it's guaranteed that there is at least one $k$ such that $sr^r_k$ lies in this interval and policies converge to $\pi^{r*}_\theta$, $\pi^{c*}_{\theta'}$. This completes the proof. □

### 6.4.2 Network architecture

As shown in Fig. 6.4, there are two to-be-trained policy neural networks, namely the runner policy $\pi_{\theta}^{r*}$ and the shared chaser policy $\pi_{\theta'}^{c*}$. In the actor-critic (AC) framework, the policy network is the actor network, which maps observations to actions directly, i.e., $a^i = \mathcal{F}(o^i \mid \theta), i \in \{r, c\}$. The architectures of the two policy neural networks are similar, with three fully-connected (FC) hidden layers activated by the Sigmoid function. Each layer consists of 512 nodes, 512 nodes and 4 nodes, respectively. The input dimension of the actor network is the same as the dimension of the observation, i.e., $3 * (N + 1)$ for the runner policy and $3 * N$ for the chaser policy. As illustrated in Fig. 6.4, the runner policy and the shared chaser policy are connected with the state $s_i$ of the agent. The relative positions are obtained from the egocentric observations. Note that, in our scenario, the objectives of these two policy networks are opposite, as described in (6.1).

### 6.4.3 Training scheme

In our training process, we utilize the PPO algorithm for the policy update as outlined in (2.14). PPO is a robust and effective choice for problems with continuous action spaces, as it employs a "clipped" surrogate objective and on-policy learning. Hence, the algorithm used in this chapter can also be denoted as AMS-PPO. Note that, in each training stage or phase, convergence is monitored using a specific condition: the training is considered converged if there is no significant improvement in the average reward over the last $l_{sw}$ episodes (1000 episodes in our study) compared to the best average reward achieved so far. Upon convergence, the current stage ends, and the training progresses to the next stage. To prevent premature termination due to initial fluctuations, this convergence criterion is applied only after the first 1 million episodes.

## 6.5 Experiments and results

This section presents the simulation and physical experiments conducted to evaluate the effectiveness of our approach. The simulation experiments cover the training process and inference tests, while the physical experiments focus on the model's

TABLE 6.1: Basic specifications of drones in simulation

| **Parameter** | **Value** |
|---|---|
| Number of chasers $N$ | 2 |
| Number of runners | 1 |
| Mass (g) $m$ | 100 |
| Box collider dimension (cm) $L \times W \times H$ | $30 \times 30 \times 5$ |
| Maximum speed (m/s) $v_{max}$ | 1.0 |
| Maximum angular velocity (rad/s) $\omega_{max}$ | 20.0 |

ability to transfer from simulation to real-world scenarios and performance in real-time flight tests. Results are analyzed and compared to the traditional Artificial Potential Field (APF) method and other DRL methods. In addition, we investigate the effect of relative maximum speed on the success rate of the navigation task $sr^r$. Furthermore, extensive physical experiments provide a success rate heatmap to examine the impact of the spatial geometry distribution of the runner and chasers on navigation performance.

### 6.5.1 Simulation experiments

#### 6.5.1.1 Settings

The simulation is run on a workstation. The simulation workstation is used to develop simulation environments for training and testing and export the trained neural network models for deployment. The configuration of the simulation workstation is: AMD Ryzen 9 5900X 12-core CPU, Nvidia RTX 3090 Gaming OC 24GB GPU, and 32GB RAM. Our simulation environment was designed using the Unity rendering engine. The proposed algorithm and training scheme were developed with CSharp and Python based on the ML-Agents Toolkit [158], which is a flexible training platform for MARL using the Pytorch library.

The size of the simulation environment in Fig. 6.2 is $5m \times 5m \times 3m$ (width $\times$ length $\times$ height). One drone is colored white as the runner, while the other two are colored blue as chasers. The target is the red box of dimension $20 \times 20 \times 20cm$. The basic specifications of drones are listed in Table 6.1. Note that, in the basic environment, the runner and chasers are the same. The control velocities of drones are limited with $v_{max} = 1m/s$. The goal of the MPETN problem is to avoid physical

TABLE 6.2: Parameters of AMS-PPO

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Convergence threshold $\eta$ | 10% | Maximum steps $T_{max}$ | 1000 |
| Maximum phases $k_{max}$ | 10 | Test episodes $ws$ | 500 |
| Task reward $C_1$ | 1000 | Task reward $C_2$ | 1000 |
| Existential penalty $w_1$ | 10000 | Risk threshold $d_\epsilon$ | 0.5 |
| Batch size | 1024 | Buffer size | 10240 |
| Initial learning rate $\alpha_0$ | 0.0003 | beta | 0.01 |
| Clip epsilon $\epsilon$ | 0.2 | lambda $\lambda$ | 0.95 |
| Number of epochs | 3 | Learning rate schedule | linear |
| Checkpoints | 10 | Maximum episodes | 6000000 |

attacks from the chasers meanwhile to reach the target. The physical attacks mean colliding with the runner of the collider dimension of $30 \times 30 \times 5cm$. The initial linear and angular velocity of drones is reset to zero at the start of each episode.

The hyperparameters for the AMS-DRL and the PPO training (AMS-PPO) are listed in Table 6.2. Note that we set the maximum phases $k_{max}$ as 10 to avoid inefficient long-time training. The training tools versions are: ml-agents-toolkit: 0.27.0; ml-agents-envs: 0.27.0; communicator API: 1.5.0; PyTorch: 1.8.2+cu11.1. To accelerate the training process, 6 copies of the environment are placed in one scene, and 3 concurrent Unity instances are invoked at the start of training.

#### 6.5.1.2 Baselines

To evaluate the performance of the proposed approach, we designed several navigation and pursuit strategies (runner/chaser policies) as baselines, including the traditional PID pursuit policy, the APF navigation policy [277], learned navigation policy with PPO [214] , and the state-of-the-art (SOTA) learning-based pursuit policy CPC-TP in [284].

(1) Random Pursuit Policy (Random): The chasers move randomly within the enclosed room with a maximum speed of $1m/s$. The initial positions and instant actions of two chasers are randomly generated.

(2) PID Pursuit Policy (PID): The chasers are independently driven by a fast proportional gain controller, aiming to reach the runner's real-time position. The PID controller for the chaser $i$ is formulated as: $a_i^c = [k_p \boldsymbol{p}_i^{cr}, 0]$ with $k_p = 2.0$.

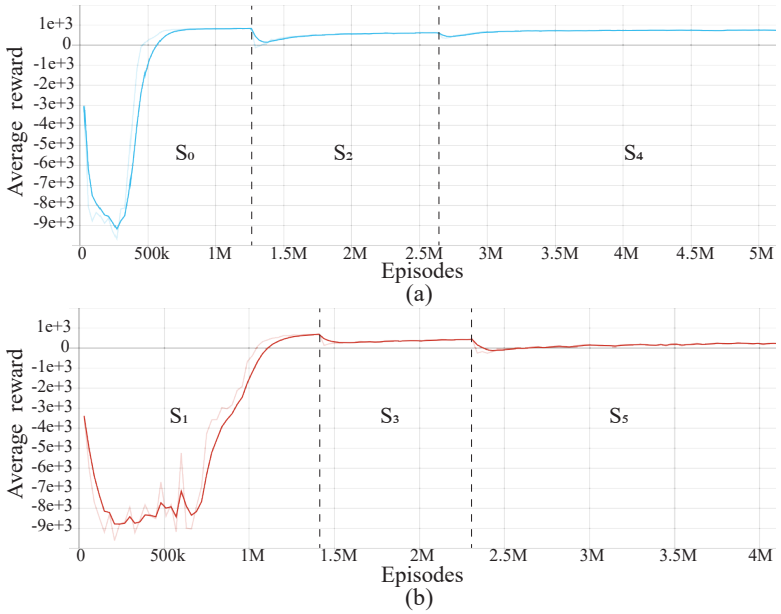



FIGURE 6.5: Average reward of AMS-DRL during the training. The training phases are denoted by $S_0$-$S_1$. (a) Runner policy; (b) Chaser policy.

(3) Learned Pursuit Policy (CPC-TP): A pursuit policy trained with CPC-TP in [284], which combines 10 timestep history observations and an LSTM hidden layer for the policy network and the value network. During the training, the runner moves straight toward a randomly generated target with a maximum velocity $1m/s$. The reward function for the chaser agents is the same as (6.6).

(4) Manual Navigation Policy (Manual): A manually controlled navigation policy using the keyboard based on the eyes' observation. This is to simulate the manual control mode of drones. Before we evaluate this policy, 50 trials have been practiced to master the keyboard operations. The SR is calculated over 100 episodes.

(5) Learned Navigation Policy (PPO): A one-stage leaning-based navigation policy trained with PPO against the PID pursuit policy. During training, the runner learned to avoid chasers driven by the PID pursuit policy and try to reach the randomly generated target. The training parameters are the same as in Table 6.2.

(6) Learned Navigation Policy (CPC-TP): A navigation policy trained with CPC-TP in [284], against the PID pursuit policy. The policy network and value network are the same as the CPC-TP pursuit policy in (3).

(7) APF Navigation Policy (APF): A traditional heuristic navigation policy with an attractive potential field and a repulsive potential field. In our work, the field

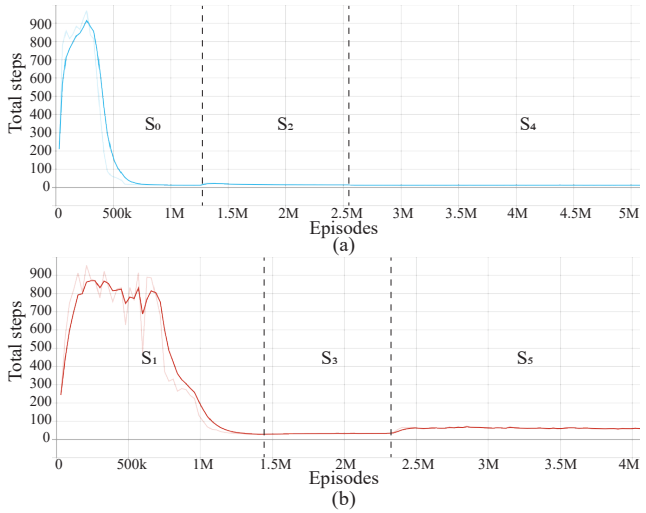



FIGURE 6.6: Total steps of AMS-DRL during the training. The training phases are denoted by $S_0$-$S_1$. (a) Runner policy; (b) Chaser policy.

functions in [301] are adopted. The attractive potential field is described as:

$$U_{att}(\boldsymbol{p}^r) = \begin{cases} 0.5\xi d^2(\boldsymbol{p}^r, \boldsymbol{p}^g), & d(\boldsymbol{p}^r, \boldsymbol{p}^g) \leq d_g^*; \\ d_g^* \xi d(\boldsymbol{p}^r, \boldsymbol{p}^g) - 0.5\xi (d_g^*)^2, & d(\boldsymbol{p}^r, \boldsymbol{p}^g) > d_g^*. \end{cases} \tag{6.7}$$

where $\xi$ is the attractive force gain, $d(\boldsymbol{p}^r, \boldsymbol{p}^g)$ is the Euclid distance between $\boldsymbol{p}^r$ and $\boldsymbol{p}^g$. $d_g^*$ is the adjustable distance. The repulsive potential field is:

$$U_{rep}(\boldsymbol{p}^r) = \begin{cases} 0.5\zeta (1/D(\boldsymbol{p}^r) - 1/D^*)^2, & D(\boldsymbol{p}^r) \leq D^*; \\ 0, & D(\boldsymbol{p}^r) > D^*. \end{cases} \tag{6.8}$$

where $\zeta$ is the repulsive force gain and $D(\boldsymbol{p}^r)$ is the nearest distance between the runner and the chasers. $D^*$ is the adjustable safe distance. The desired combined force generated from the potential fields is:

$$F(\boldsymbol{p}^r) = -\nabla U(\boldsymbol{p}^r) \tag{6.9}$$

where $U(\boldsymbol{p}^r) = U_{att}(\boldsymbol{p}^r) + U_{rep}(\boldsymbol{p}^r)$. $\nabla(\cdot)$ is the gradient vector w.r.t. the position of runner $\boldsymbol{p}^r$. Then we can obtain:

$$\nabla U_{att}(\boldsymbol{p}^r) = \begin{cases} \xi(\boldsymbol{p}^r - \boldsymbol{p}^g), & d(\boldsymbol{p}^r, \boldsymbol{p}^g) \leq d_g^*; \\ d_g^* \xi (\boldsymbol{p}^r - \boldsymbol{p}^g)/d(\boldsymbol{p}^r, \boldsymbol{p}^g), & d(\boldsymbol{p}^r, \boldsymbol{p}^g) > d_g^*. \end{cases} \tag{6.10}$$

TABLE 6.3: Navigation performance comparison (success rate of the runner $sr^r$ %) against various chaser policies (bold means the best runner policy while red means the best chaser policy)

| Chaser Policy (Pursuit Policy) | Runner Policy (Navigation Policy) | | | | | |
|---|---|---|---|---|---|---|
| | Manual | APF | Direct | PPO | CPC-TP | AMS-DRL |
| Random | 77.6 | **99.4** | 21.6 | 94.4 | 80.0 | 95.1 |
| PID | 10.2 | **67.1** | 0.0 | 30.3 | 29.5 | 44.1 |
| CPC-TP | 2.4 | **77.9** | 0.0 | 11.2 | 9.4 | 44.8 |
| AMS-DRL | 0.0 | 31.8 | 0.0 | 3.2 | 4.6 | **37.1** |

TABLE 6.4: Comparison of inference runtime in various devices (ms).

| Method/Device | Desktop | Xavier NX | RaspberryPi 4B |
|---|---|---|---|
| CPC-TP[284] | 0.152±0.038 | 0.889±0.118 | 4.748±0.892 |
| AMS-DRL(Chaser) | 0.084±0.011 | 0.580±0.135 | 2.296±0.866 |
| AMS-DRL(Runner) | 0.085±0.006 | 0.536±0.172 | 1.592±0.562 |

$$\nabla U_{rep}(\boldsymbol{p}^r) = \begin{cases} 0, & D(\boldsymbol{p}^r) > D^* \\ \zeta(1/D^* - 1/D(\boldsymbol{p}^r))\frac{1}{D^2(\boldsymbol{p}^r)}\nabla D(\boldsymbol{p}^r), & others. \end{cases} \tag{6.11}$$

Here, we set $\xi = 2.0$, $\zeta = 2.0$, $d_g^* = 0.5m$ and $D^* = 0.5m$.

#### 6.5.1.3 Ablations

To investigate the effect of asynchronous multistage learning, we compared the AMS-DRL with the "Direct" learning method. The "Direct" learning method trains two different policies, i.e., runner policy and chaser policy, at the same time with the same configurations of AMS-DRL.

#### 6.5.1.4 Training results

During the training, all objects are randomly generated within the constrained 3D simulation space. The mean rewards and total steps over three epochs of the AMS-DRL training are illustrated in Fig. 6.5 and Fig. 6.6. We can find that the policies converge at phase $S_5$ with a threshold $\eta = 10\%$. Both runner and chaser policies fluctuate in the early training phases, i.e., $S_0$ and $S_1$ for runner and chaser policies, respectively. In each phase, the trained policy improves until the

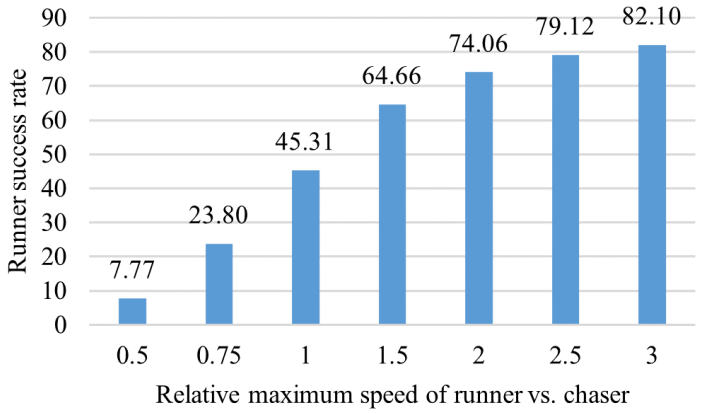



FIGURE 6.7: Runner success rate with different relative maximum speed.

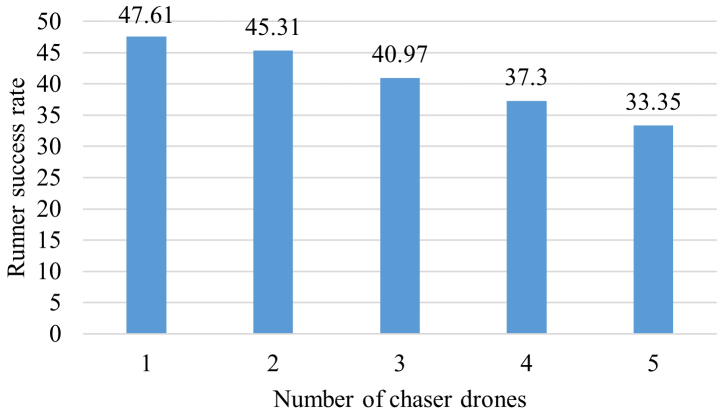



FIGURE 6.8: Runner success rate with different numbers of chaser drones.

aforementioned convergence condition is met. The success rates $sr^r$ and $sr^c$ of $S_5$ are 45.31% and 54.69% respectively. From Fig. 6.5, we can observe that at each phase there is a reward drop compared to the last inference policy at the beginning of training. This indicates that their opposites learned to become better from the previous phase.

#### 6.5.1.5 Testing results

In this work, the navigation success rate $sr^r$ (reach the target without collisions) is the evaluation metric for the proposed approach. We tested different combinations of pursuit policies and navigation policies over 500 episodes. The AMS-DRL policies are exported from the last phase $S_5$. The Direct policy is trained from the ablation "Direct" method. Note that for fair evaluation, during testing, we fixed the target at position $[1.0, 3.5, 0.5]m$ and the runner at position $[1.0, 0.0, 0.2]m$ with

random noise inside a sphere of radius $0.2m$. This setting will decrease the navigation performance compared to the training process. The success rates $sr^r$ are listed in Table 6.3. The inference runtime of different policies is tested on various devices, namely the desktop with Intel i7-10700, Jetson Xavier NX, and Raspberry Pi 4B. From the test results listed in Table 6.4, the trained policies can be executed onboard in real time.

Compared to other learning-based baselines such as Direct, PPO and CPC-TP, our proposed learning-based approach AMS-DRL has achieved the best performance against different chaser policies. The direct method cannot learn an effective navigation policy since the runner is unable to focus on reaching the target while avoiding the chasers at the same time. APF obtains the highest success rate against random, PID and CPC-TP pursuit policies while falling behind against the AMS-DRL pursuit policy. A well-designed APF has been validated with excellent dynamic obstacle avoidance performance in [277] when the environment is not cluttered, but our learning-based approach is better at evading collaborative learning-controlled pursuers and with fewer navigation steps (average 182 steps compared to 237 steps for AFP). Meanwhile, our learning-based approach is more promising when a full nonlinear quadrotor model and individual thrust control are adopted, while APF can only handle the point-mass model. From column-wise, the chaser policy trained with AMS-DRL outperforms all other chaser policies, as the success rate decreases when the AMS-DRL chaser policy is adopted. This provides insights into designing pursuit strategies.

In the training scenario, the same maximum speed of chasers and runner is assumed, which limits the runner's success rate. To showcase how the relative maximum speed affects $sr^r$, we conducted 1000 simulation tests, each with various relative maximum speeds of runner-chaser from 0.5 to 3. Note that relative maximum speed $\tilde{v}_{max}$ is defined as the ratio of the maximum speed of the runner to that of the chasers, i.e., $\tilde{v}_{max} = v^r_{max}/v^c_{max}$, thereby providing a comparative measure of their speed capabilities. In our experiments, we fixed the maximum speed of the chasers at $1m/s$ while changing the maximum speed of the runner drone with different relative maximum speeds. The results in Fig. 6.7 show that a faster runner is able to outmaneuver multiple collaborative chasers, especially in cases where the chasers are located between the runner and the target box. To test our approach in more adversarial scenarios, we trained the policies with different numbers of chaser

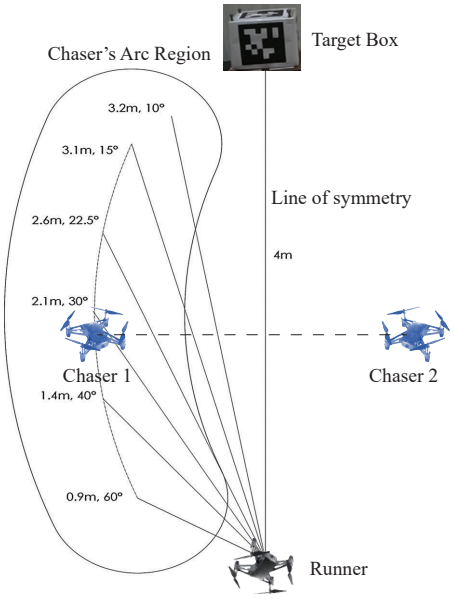



FIGURE 6.9: The spacial geometry distribution of agents in the physical tests.

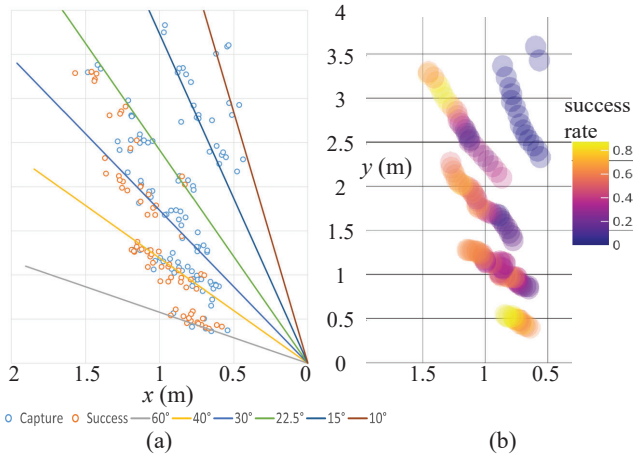
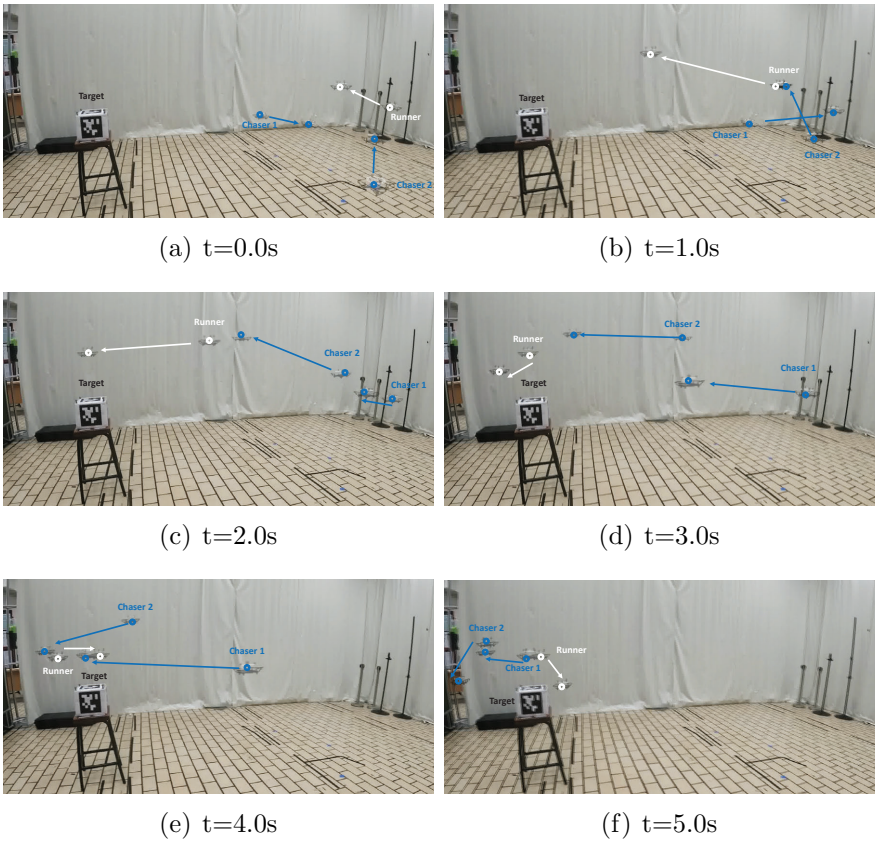



FIGURE 6.10: The testing result distribution with different spacial geometry configurations. (a) The cases scatter; (b) the success rate heatmap. The runner is fixed at the origin $[0.0, 0.0]$ while the chasers are mirrored along the y-axis.

drones. The results in Fig. 6.8 show that more chaser drones could bring down the success rate slightly but the runner policy still performs better than other methods in Table 6.3.

### 6.5.2 Physical experiments

In this section, we present the physical experiments conducted and evaluate the navigation performance of the trained runner policy in real-time flight. To evaluate




(a) t=0.0s (b) t=1.0s

(c) t=2.0s (d) t=3.0s

(e) t=4.0s (f) t=5.0s

FIGURE 6.11: The snapshots of multi-pursuit evasion with targeted navigation in physical experiments. The white runner drone eventually reached the target box without being captured by intelligent blue chasers. The beginning of the arrow is the current position, while the end of arrow is the position in next frame.

the models' generalization and Sim2Real capability, we transfer our trained models with the proposed AMS-DRL approach to an unseen large indoor environment to control three Tello Edu drones [1] without fine-tuning. Note that both the chaser policy and the runner policy are exported from the final phase of AMS-DRL. More details can be found at `https://github.com/NTU-ICG/AMS-DRL-for-Pursuit-Evasion`.

#### 6.5.2.1 Settings

The physical test environment is an indoor flight enclosure with a size of $8 \times 7 \times 4m$. The target is a parcel box with a size of $20 \times 30 \times 20cm$. The runner and two chasers are the same Tello Edu drones of size $20 \times 19 \times 5cm$, tagged with white and blue colors, respectively. The detailed specifications of Tello Edu refer to `https://www.`

[1] https://www.ryzerobotics.com/tello-edu/downloads

`ryzerobotics.com/tello-edu/specs`. The real-time position of each drone and the target box was streamed by the OptiTrack motion capture system with tracker balls attached to each object of interest (OOI). Fig. 6.1 shows one instance of our experiment, where the runner, chasers, and box are OOIs in our experiments.

All drones are driven by ROS Melodic operating on a center computer (Ubuntu 18.04 OS) with a frequency of $240Hz$. The center computer subscribes to the positions of all OOIs and computes the desired command velocities with loaded trained policy models (exported as ONNX files). Each drone is set as AP mode and connected to the existing WiFi network with different IP addresses (192.168.120-192.168.122) during real-time flight tests. The command velocities are sent through the Tello ROS package[2] via a connected WiFi network. During testing, Python3 and ONNXRuntime[3] are used for inference with trained models in ONNX format.

To minimize damage and prevent actual collisions between the chaser and runner drones, a minimum distance of $35cm$ must be maintained between the drones; failure to do so will indicate a corresponding collision outcome. Similarly, a $20cm$ threshold was implemented to detect the arrival of the runner drone at the target box. To fairly evaluate the navigation performance of the runner drone in different situations, the target is fixed in a test position $[-2.0, 0.8, 1.0]m$. The runner drone is placed at a position $[0.45, 1.3, 1.0]m$ with random noise, while the chaser drones' locations are mirrored along the line of symmetry (LOS) formed between the target box and the runner, as illustrated in Fig. 6.9.

#### 6.5.2.2 Sim2Real performance evaluation

In this work, to investigate the navigation performance of the runner policy in various situations, we conducted 189 tests with different spacial geometry configurations. As illustrated in Fig. 6.9, the chaser's takeoff region (approximated by an arc region) is segregated along 4 lines at 60°, 40°, 30°, and 15° with respect to the LOS, starting at the runner position. Along each line, several starting positions were denoted with equal spacing between them, and an average of 5-8 flights were conducted at each starting position. Additional lines at 22.5° and 10° were included towards the end of this data collection process for a more comprehensive

[2]https://github.com/xjp99v5/tello-driver-ros
[3]https://onnxruntime.ai/

TABLE 6.5: Navigation performance of runner in real-time flight

| Angle | 60° | 40° | 30° | 22.5° | 15° | 10° |
|---|---|---|---|---|---|---|
| Success | 17 | 27 | 17 | 13 | 0 | 0 |
| Capture | 6 | 31 | 24 | 24 | 20 | 10 |
| Total | 23 | 58 | 41 | 37 | 20 | 10 |
| Success rate | 73.9% | 46.6% | 41.5% | 35.1% | 0% | 0% |

evaluation. All flight test results are collected and plotted in Fig. 6.10 (a), where the success and capture cases are shown on one side of the corresponding chaser starting region. Here, a success case means the runner drone managed to evade the chasers and reach the target box while a capture case indicates the chaser drones managed to intercept and collide with the runner drone. A successful case is demonstrated in Fig. 6.11, which shows the maneuvers performed by the runner and the chasers. From Fig. 6.11, the runner drone finally arrived at the target box and meanwhile succeeded in evading the collision from chaser drones. Note that both the drone's body momentum and the air downwash influence its motion, as illustrated in Figs. 6.11 (b) and (f).

#### 6.5.2.3 Results and discussion

To further analyze the navigation performance of the runner policy, the success rate heatmap is generated over the averaged coordinates of the first 8 closest data points along each angle edge in terms of absolute distance to the chaser drone. This process was repeated iteratively for the next 8 data points until all points along the angle edge were included. The runner drone's success rate is around 50% along the 40° line, and the success rate decreases when the angle decreases. The success rates with different angles are listed in Table 6.5. For angles beyond 90°, the runner drone has a nearly 100% success rate as the chaser drones are behind the runner drone when all of them have the same maximum speed. Note that this simplified physical setup assumes that both chaser drones will spawn mirrored along the LOS, while in a truly random case, the probability is higher since one chaser drone can spawn closer to the runner drone, lowering the runner drone's success rate.

Among these 189 physical tests, 115 tests were labeled as capture (60.8%) while another 74 tests were labeled as success (39.2%), which benchmarked against the simulation results as listed in Table 6.3 (37.1%). The results verified the generalization capabilities of trained policies in a physical application. The differences could be chalked up to the low-level control limitations and the experiment setup. The lack of thrust makes drones unable to maintain the desired flight path under the influence of another drone's downwash as seen in Fig. 6.11(f). An improved navigation policy is expected to be learned with a fully nonlinear model and individual thrust control. The scalability of the proposed approach could be enhanced with the help of an attention mechanism.

## 6.6 Conclusion

In this chapter, we present a novel approach, AMS-DRL, for ensuring the safe navigation of drones under physical attacks from multiple pursuers. The AMS-DRL method trains a deep adversarial neural network to learn from the actions of the pursuers and respond effectively to avoid collisions and reach the target destination. Through extensive simulations and physical experiments, our results show that this approach outperforms existing learning-based baselines in terms of navigation success rate. Additionally, the success rate heatmap provides valuable insights into the impact of spatial geometry on drone navigation performance.

This research highlights the benefits of DRL in solving complex and challenging problems in drone navigation. In addition, the performance of the learning-based chaser policy provides benchmarks for research on anti-drone techniques with DRL. Further work will leverage individual thrust control and visual sensors to improve the agility and applicability of the runner drone in more adversarial environments.

# Chapter 7

# Cyber attack detection and isolation in adversarial environments

Beyond the physical attacks in the adversarial environments as studied in Chapters 5 and 6, this chapter focuses on improving drone security with cyber attack detection and isolation in adversarial environments. Common vulnerabilities in typical intelligent cyber-physical systems such as drones can be easily exploited by cyber attackers to cause serious accidents and harm. For successful drone operations, security against cyber attacks is imperative. In this chapter, we propose a modified sliding innovation sequences (MSIS) detector, based on the extended Kalman filter optimal state estimation, for a dynamic quadrotor system to detect cyber attacks inflicted on both its actuators and sensors in real time. These cyber attacks include random attacks, false data injection (FDI) attacks and denial-of-service (DoS) attacks. The MSIS detector computes the operator norm of the normalized innovation (residual) sequence within a sliding time window and triggers the alarm if the value is above the preset threshold. For a quadrotor undergoing rapid turns in a complex trajectory, the detector observes a reduced false alarm rate as compared to other state estimation-based detectors. To address the initial estimation error problem, we implement an iteration procedure to initiate and calibrate the detector. By evaluating the sample covariance of the normalized innovation sequence, the MSIS detector has the capability to isolate cyber attacks. Finally, simulation

results of a quadrotor in a periodic, complex trajectory flight are provided to verify the effectiveness of the MSIS detection and isolation method.

## 7.1 Introduction

Drones are intelligent cyber-physical systems (CPSs)[34, 35, 302, 303] that rely on functional sensors, real-time information communication, and flight control systems to achieve perception and autonomous flight. As drones are widely deployed in applications, such as parcel delivery[37], surveillance, and rescue[304], they require reliable security solutions against cyber attacks on their sensors and actuators[305–309], on top of solutions against physical protection [38, 310]. The vulnerabilities which commonly exist in CPSs can easily place drones into an adversarial environment with hidden cyber attacks and hence cause serious accidents [311, 312]. Drone communication networks such as WiFi, GPS, Bluetooth, infrared (IR) and ZigBee, can be easily compromised by cyber attacks [313–318]. For instance, "DJI" drones were reported to have experienced security breaches [314]. In another instance, a Johns Hopkins computer security team demonstrated the ease of hacking an off-the-shelf hobby drone with basic denial of service (DoS) and buffer-overflow attacks during its ARDiscovery connection process[315]. More remarkably, Iran allegedly brought down a US RQ-170 Sentinel UAV via spoofing the UAV's control system in 2011 [319]. Meanwhile, in 2012, a military UAV was reportedly hacked when the operator used the controller to download a malware-infected video game [320]. Essentially, the use of on-board computers on drones for real time perception, communication and control capabilities make them vulnerable to possible cyber attacks in communication links, sensor networks and even actuator channels, resulting in degraded flight control performance.

In this chapter, a novel and effective state estimation-based detection and isolation method applicable to any nonlinear aerial platforms, called the Modified Sliding Innovation Sequences (MSIS) detector, is proposed. By considering the quadrotor's full states (translational and rotational), the MSIS detector is able to detect cyber attacks on sensors and actuators for quadrotors with a complex trajectory. It features a specified operator norm to the normalized innovation sequence in a sliding window and an inner iteration procedure in the calibration phase to lower the false alarm rate during high-speed rotational motion. By focusing on three main

data-driven cyber attacks, namely random attacks, false data injection attacks and DoS attacks, we also present a cyber attack isolation technique based on statistical property (mean, covariance) identification of the signals. The **main contributions** of this chapter are summarized as follows:

(1) Unlike previous work in [70, 80, 81], which have been limited to implementations on simplified linear models involving translational motions only, to the best of authors' knowledge, this is the first time that a full-state nonlinear hybrid dynamics model of a dynamic quadrotor, including translational and rotational states, is established in the detection and isolation problem with different cyber attacks on sensors and actuators.

(2) A novel and effective real-time cyber attack detection method using MSIS is proposed for a dynamic quadrotor. It employs an iteration procedure within the state estimation framework to calibrate the detector.

(3) Attack isolation is achieved for the three above-mentioned data-driven cyber attacks according to their statistical properties.

(4) Not limited to the hovering or low-speed translational motion scenarios, the detection strategy is verified by means of a complex periodic trajectory tracking flight of a quadrotor involving high-speed nonlinear dynamics.

The rest of the chapter is composed of the following sections. Section 7.2 summarizes related cyber attack detection methods for drones. The problem description is presented in Section 7.3, where a nonlinear 6DOF hybrid quadrotor model is established, the extended Kalman filter (EKF) to estimate the states of the quadrotor is described, and three types of cyber attacks are modeled. Section 7.4 presents the real-time cyber attack detection and isolation mechanism with MSIS in detail. Simulation results of the cyber attack detection and isolation for a dynamic quadrotor in a periodic trajectory tracking flight scenario are illustrated in Section 7.5. Discussions and conclusions are given in the final section.

*Notations:* Let bold $\boldsymbol{x}$ denote a vector or matrix and $x$ denotes a scalar variable. $\dot{\boldsymbol{x}}$ is the first order derivative of $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ is the state estimate. $\odot$ represents the inner product of two matrices. $\mathbb{R}^n$ denotes the sets of real number row vectors with $n$ dimensions. $\boldsymbol{x}^T$ stands for the matrix transpose of $\boldsymbol{x}$. $p(\boldsymbol{x}) \sim \aleph(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ means vector $\boldsymbol{x}$ follows a Gaussian distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$

while $p(x) \sim \aleph(\mu, \sigma^2)$ for a scalar. $:=$ means that the item on the left hand side is being defined to be what is on the right hand side. More detailed contextual notations can be found in Symbols and Acronyms.

## 7.2 Related work

There are two main types of cyber attack detection methods that can be used for drones, namely learning-based detection [68, 69, 321–326] and state estimation-based detection [70, 71, 80, 81, 327, 328]. In learning-based detection methods, learning processes such as neural networks[69, 321–323], clustering[324], reinforcement learning[325, 326], and federated learning[68] are generally combined with filters or state estimation to provide a feasible feature extraction for nonlinear hybrid systems. In [322], the authors utilized the kernel principal component analysis and trained the support vector regression model to achieve anomaly detection for an aircraft system, while the adaptive neural network is used in [324] to detect FDI attacks on a UAV. Moreover, a communication-efficient anomaly detection framework with attention-embedded federated learning was proposed in [68] for industrial IoT. However, learning-based detection methods have several challenges for their application in real time as they require massive training data sets and the training process can be very time-consuming given a complex, nonlinear time-varying hybrid system, such as a quadrotor drone.

State estimation-based detection methods have been widely discussed, such as the generally used chi-square detector[71, 328]. The core of state estimation-based detection methods is to construct an effective state estimation for the targeted systems using designed filters, e.g, Kalman filters, and subsequently verify if the innovation (residual)[70, 71] or similarity[80, 327, 328] between the state estimation and sensor readings exceeds the preset threshold. A Euclidean distance detector was introduced in [71] to detect an FDI attack in power grid applications. However, the Euclidean distance detector would not be feasible for drone applications as it requires the reference of the state values of the drone undergoing the same maneuver without cyber attacks. In [327], a detection method based on two ellipsoid sets, namely the prediction set and the estimation set, was developed to detect replay attacks and FDI attacks on a vehicle tracking system, but these ellipsoid sets cannot be formulated for a drone system. Although [70, 80, 81] developed

detection methods on drones, these methods are based on linear systems under equilibrium conditions and can hardly be used for a dynamic drone with translational and rotational motions in six degrees of freedom (6DOF). For instance, in [81], a weighted matrix-based threat-detection level function with a bank of estimators was used to detect attacks on a linear aircraft system model. The SUM detector in [70] was developed for FDI attack detection on a flight vehicle by utilizing a summation of historical innovations. Nevertheless, on a rapidly maneuvering drone, the alarm will be triggered even in the absence of cyber attacks because large detection values are generated due to the accumulation of estimation errors over the whole historical innovation sequences. Therefore, it is challenging for existing state estimation-based detection methods to detect cyber attacks on drones maneuvering in a complex trajectory in real time. Despite the significant research in CPS technology for cyber attack detection and isolation[329–331], there is no comprehensive theory or specific framework for cyber attack detection and isolation for drones due to large model uncertainties and high nonlinearities that exist in their flight dynamic systems. As such, the motivation of this chapter lies in addressing the urgent need for a comprehensive analysis and solution for cyber attack detection and isolation for drones in real time applications.

## 7.3 Preliminaries and problem formulation

### 7.3.1 Quadrotor dynamics

To obtain a rigid body with a 6DOF dynamics model for a quadrotor, we first introduce two coordinate systems, the Earth inertial frame $\{O_E, x_E, y_E, z_E\}$ and the body frame $\{O_B, x_B, y_B, z_B\}$. The Earth inertial frame is originated at a fixed point on the ground. The body frame is originated at the center of gravity (CG) of the quadrotor with $x_B$ pointing forward, $y_B$ pointing leftward, and $z_B$ aligning with the thrust vector. The coordinate systems in a quadrotor are shown in Fig. 7.1, where the thrust vector parameters, gravitation $\boldsymbol{g}$, and the rotor index definition, are also presented. Regardless of the wind disturbance, the translational and rotational

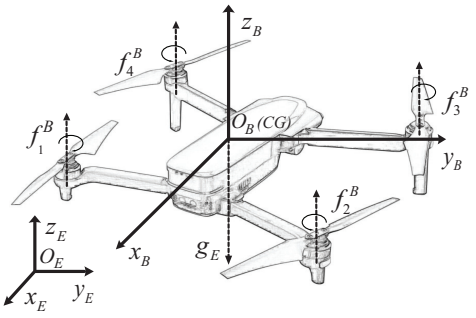



FIGURE 7.1: The quadrotor model, including the definition of the body frame, rotor thrust vector and gravitation $g_E$.

dynamics of the quadrotor can be modeled as follows [332]:

$$\dot{\boldsymbol{p}}_E = \boldsymbol{v}_E \tag{7.1a}$$

$$\dot{\boldsymbol{v}}_E = \boldsymbol{R}_B^E(\boldsymbol{q}_{EB}) \odot \boldsymbol{f} + \boldsymbol{g} \tag{7.1b}$$

$$\dot{\boldsymbol{q}}_{EB} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_B)\boldsymbol{q}_{EB} \tag{7.1c}$$

where, $\boldsymbol{p}_E = [x, y, z]^T$ and $\boldsymbol{v}_E = [v_x, v_y, v_z]^T$ are the position and translational velocity of a quadrotor in the Earth inertial frame $O_E$. $\boldsymbol{q}_{EB} = [q_0, q_1, q_2, q_3]^T$ is a unit quaternion which is used to represent the orientation of the quadroter and $\boldsymbol{\omega}_B = [\omega_x, \omega_y, \omega_z]^T$ is the body angular velocity (roll, pitch, and yaw respectively) in the body frame $O_B$. $\boldsymbol{R}_B^E$ denotes the transform matrix from the body frame $O_B$ to the Earth inertial frame $O_E$, and it is a function of $\boldsymbol{q}_{EB}$ which can be expressed as follows:

$$\boldsymbol{R}_B^E(\boldsymbol{q}_{EB}) = \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2q_1^2-2q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2q_1^2-2q_2^2 \end{bmatrix} \tag{7.2}$$

$\boldsymbol{f} = [0, 0, f]^T$ is the mass-normalized thrust vector, i.e., $f = \sum_{i=1}^{4} f_i^B/mass$ and $\boldsymbol{g} = [0, 0, -g_z]^T$ with $g_z = 9.81m/s^2$ being the gravity acceleration on Earth. We

denote the $\boldsymbol{\Omega}(\boldsymbol{\omega}_B)$ as the skew-symmetric matrix and is given by

$$\boldsymbol{\Omega}(\boldsymbol{\omega}_B) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{7.3}$$

External forces such as aerodynamic drag and momentum of the body will not be discussed here since high-bandwidth controllers are used to perfectly track the angular velocity commands, i.e., angular dynamics can be neglected. We select vector $\boldsymbol{x} = [\boldsymbol{p}_E^T, \boldsymbol{v}_E^T, \boldsymbol{q}_{EB}^T]^T$ and $\boldsymbol{u} = [f, \omega_x, \omega_y, \omega_z]^T$ as the states and control commands of a quadrotor respectively. The achievable thrust $f$ and angular velocity commands are limited within a feasible set, i.e., $0 \leq f_{min} \leq f \leq f_{max}$ and $\|\boldsymbol{\omega}_B\| \leq \omega_{max}$, where $\|\cdot\|$ denotes the Euclidean norm.

Therefore, the quadrotor dynamics modeled in (1a)-(1c) can be formulated as a nonlinear time-varying hybrid system which combines continous differential dynamics of the quadrotor and discrete signals from noises and cyber attacks:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{w} \\ \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{v} \end{cases} \tag{7.4}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the states, $\boldsymbol{y} \in \mathbb{R}^m$ is the measurement and $\boldsymbol{u} \in \mathbb{R}^p$ is the control input. $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are the state function and the measurement function respectively. The process noise $\boldsymbol{w}$ and the measurement noise $\boldsymbol{v}$ are independent random variables that satisfy the Gaussian distribution with zero mean value. The process noise contains un-modeled dynamics, such as gravity $\boldsymbol{g}$, and disturbance inputs, i.e., uncertainties. The Gaussian distributions of process noise $p(\boldsymbol{w})$ and measurement noise $p(\boldsymbol{v})$ satisfy the following probability density functions:

$$\begin{cases} p(\boldsymbol{w}) \sim \boldsymbol{\aleph}(\boldsymbol{0}, \boldsymbol{Q}) \quad with \quad \boldsymbol{Q} = diag\{\sigma_{w1}^2, \sigma_{w2}^2, \ldots\}, \\ p(\boldsymbol{v}) \sim \boldsymbol{\aleph}(\boldsymbol{0}, \boldsymbol{R}) \quad with \quad \boldsymbol{R} = diag\{\sigma_{v1}^2, \sigma_{v2}^2, \ldots\} \end{cases} \tag{7.5}$$

where $\sigma^2$ is the variance of the noise distribution.

### 7.3.2 Extended kalman filter

Our state estimation-based detector is derived from the extended Kalman filter (EKF) [333]. We firstly discretizes the system dynamics (7.4) using the Euler integration method. The discretized system are formulated as follows:

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)T_s + \boldsymbol{w}_k T_s \\ \boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k \end{cases} \tag{7.6}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are the state and control input vectors at the discrete time $t_k = kT_s$, and $T_s$ is the sampling interval. Then the linearized system matrices of (7.6) at the current operating time point $t_k$, $\boldsymbol{A}_{lin,k}$, $\boldsymbol{B}_{lin,k}$, $\boldsymbol{H}_{lin,k}$ and $\boldsymbol{D}_{lin,k}$ are obtained approximately based on the state estimate $\hat{\boldsymbol{x}}_k$, the priori state estimate $\hat{\boldsymbol{x}}_{k|k-1}$, and the input $\boldsymbol{u}_k$. While ignoring high order items, they can be expressed as follows:

$$\boldsymbol{A}_{lin,k} = \left( \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)}{\partial \boldsymbol{x}_k} \right)^T |_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_k} T_s + \boldsymbol{I} \tag{7.7a}$$

$$\boldsymbol{B}_{lin,k} = \left( \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)}{\partial \boldsymbol{u}_k} \right)^T |_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_k} T_s \tag{7.7b}$$

$$\boldsymbol{H}_{lin,k} = \left( \frac{\partial \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)}{\partial \boldsymbol{x}_k} \right)^T \Big|_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_{k|k-1}} \tag{7.7c}$$

$$\boldsymbol{D}_{lin,k} = \left( \frac{\partial \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)}{\partial \boldsymbol{u}_k} \right)^T \Big|_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_{k|k-1}} \tag{7.7d}$$

The EKF algorithm below is used to compute the state estimate $\hat{\boldsymbol{x}}_k$:

- **Step 1**

  Initial value selection: select the initial state estimate $\hat{\boldsymbol{x}}_0$ and the initial estimate covariance matrix $\boldsymbol{P}_0$.

- **Step 2**

  Prior predictions: estimate state values and the state covariance matrix as (8)-(9).

$$\hat{\boldsymbol{x}}_{k|k-1} = \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1})T_s \tag{7.8}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{A}_{lin,k-1}\boldsymbol{P}_{k-1}\boldsymbol{A}_{lin,k-1}^T + Ts\boldsymbol{Q}T_s \tag{7.9}$$

where $\hat{\boldsymbol{x}}_{k|k-1}$ and $\boldsymbol{P}_{k|k-1}$ are priori state estimate and priori estimate covariance matrix respectively.

- **Step 3**

  Posterior correction: firstly, calculate the Kalman gain:

$$\boldsymbol{V}_k = \boldsymbol{H}_{lin,k}\boldsymbol{P}_{k|k-1}\boldsymbol{H}_{lin,k}^T + \boldsymbol{R} \tag{7.10}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_{lin,k}^T\boldsymbol{V}_k^{-1} \tag{7.11}$$

  secondly, update the estimate covariance matrix and the state estimate based on the measurement:

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_{lin,k})\boldsymbol{P}_{k|k-1} \tag{7.12}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k)) \tag{7.13}$$

  where $\hat{\boldsymbol{x}}_k$ and $\boldsymbol{P}_k$ are the posteriori state estimate and the posteriori estimate covariance matrix.

When the system is uniformly completely controllable and observable, $\boldsymbol{K}_k$ and $\boldsymbol{P}_k$ will converge to constants.

### 7.3.3 Quadrotor state estimation based on the EKF

Due to the possibility of linear decoupling translational dynamics and rotational dynamics, we can denote $\boldsymbol{x}_t = [\boldsymbol{p}_E^T, \boldsymbol{v}_E^T]^T$ and $\boldsymbol{x}_q = \boldsymbol{q}_{EB}$ as translational states and rotational states. The measurement is represented as $\boldsymbol{y} = [\boldsymbol{y}_t^T, \boldsymbol{y}_q^T]^T$. The translational measurement $\boldsymbol{y}_t = \boldsymbol{p}_E$ is obtained from the GPS or visual-inertial odometry (VIO) [89]. The rotational measurement is $\boldsymbol{y}_q = [a_x, a_y, a_z, \psi_m]^T$, where $a_x$, $a_y$ and $a_z$ denote the measured acceleration values from the Inertial-Measurement-Units (IMU), and $\psi_m$ is the azimuth angle obtained from the magnetometer.

The discrete rotational states equation with process noise is expressed as:

$$\boldsymbol{x}_{q,k+1} = \boldsymbol{x}_{q,k} + \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_B)_k\boldsymbol{x}_{q,k}T_s + \boldsymbol{w}_qT_s \tag{7.14}$$

where, $\omega_x$, $\omega_y$, and $\omega_z$ can be read from the IMU and $\boldsymbol{w}_q$ is the corresponding process noise. Subsequently, we can obtain linearized state matrices of (7.14) as the follows:

$$\boldsymbol{A}_{q,lin,k} = \begin{bmatrix} 1 & -\frac{\omega_{x,k}T_s}{2} & -\frac{\omega_{y,k}T_s}{2} & -\frac{\omega_{z,k}T_s}{2} \\ \frac{\omega_{x,k}T_s}{2} & 1 & \frac{\omega_{z,k}T_s}{2} & -\frac{\omega_{y,k}T_s}{2} \\ \frac{\omega_{y,k}T_s}{2} & -\frac{\omega_{z,k}T_s}{2} & 1 & \frac{\omega_{x,k}T_s}{2} \\ \frac{\omega_{z,k}T_s}{2} & \frac{\omega_{y,k}T_s}{2} & -\frac{\omega_{x,k}T_s}{2} & 1 \end{bmatrix}_{\hat{\boldsymbol{x}}_{q,k}} \tag{7.15}$$

$$\boldsymbol{B}_{q,lin,k} = \frac{1}{2} \begin{bmatrix} 0 & -q_1 & -q_2 & -q_3 \\ 0 & q_0 & -q_3 & q_2 \\ 0 & q_3 & q_0 & -q_1 \\ 0 & -q_2 & q_1 & q_0 \end{bmatrix}_{\hat{\boldsymbol{x}}_{q,k}} T_s \tag{7.16}$$

The measurement equation is established as:

$$\boldsymbol{y}_q = \begin{bmatrix} -2g_z(q_1q_3 - q_0q_2) \\ -2g_z(q_2q_3 + q_0q_1) \\ -g_z(1 - 2q_1^2 - 2q_2^2) + f \\ \arctan\left(\frac{2(q_1q_2+q_0q_3)}{1-2q_2^2-2q_3^2}\right) \end{bmatrix} + \boldsymbol{v} \tag{7.17}$$

The linearized measurement matrix can be represented as:

$$\boldsymbol{H}_{q,lin,k} = \begin{bmatrix} 2g_zq_2 & -2g_zq_3 & 2g_zq_0 & -2g_zq_1 \\ -2g_zq_1 & -2g_zq_0 & -2g_zq_3 & -2gq_2 \\ 0 & 4g_zq_1 & 4g_zq_2 & 0 \\ \frac{a_1}{a_0} & \frac{a_2}{a_0} & \frac{a_3}{a_0} & \frac{a_4}{a_0} \end{bmatrix}_{\hat{\boldsymbol{x}}_{q,k|k-1}} \tag{7.18}$$

where $a_1 = 2q_3c_1$, $a_2 = 2q_2c_1$, $a_3 = 2q_1c_1 + 8q_2c_2$, $a_4 = 2q_0c_1 + 8q_3c_2$, $a_0 = c_1^2 + 4c_2^2$ and $c_1 = 1 - 2q_2^2 - 2q_3^2$, $c_2 = q_1q_2 + q_0q_3$.

$$\boldsymbol{D}_{q,lin,k} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.19}$$

For translational states, we can get the linearized form from 1(a) and 1(b). Therefore, we can obtain the linearized form of (7.6) for all states.

$$\boldsymbol{A}_{lin,k} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{I}_{3\times3}T_s & \boldsymbol{0}_{3\times4} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{A}_{v,lin,k} \\ \boldsymbol{0}_{4\times3} & \boldsymbol{0}_{4\times3} & \boldsymbol{A}_{q,lin,k} \end{bmatrix} \tag{7.20}$$

where, subscript $3 \times 3$ denotes the dimension of matrices.

$$\boldsymbol{A}_{v,lin,k} = \begin{bmatrix} 2q_2 f & 2q_3 f & 2q_0 f & 2q_1 f \\ -2q_1 f & -2q_0 f & 2q_3 f & 2q_2 f \\ 0 & -4q_1 f & -4q_2 f & 0 \end{bmatrix}_{\hat{\boldsymbol{x}}_{q,k},\boldsymbol{u}_k} T_s \tag{7.21}$$

$$\boldsymbol{B}_{lin,k} = \begin{bmatrix} \boldsymbol{0}_{3\times4} \\ \boldsymbol{B}_{v,lin,k} \\ \boldsymbol{B}_{q,lin,k} \end{bmatrix} \tag{7.22}$$

where,

$$\boldsymbol{B}_{v,lin,k} = \begin{bmatrix} 2(q_1q_3 + q_0q_2) & 0 & 0 & 0 \\ 2(q_2q_3 - q_0q_1) & 0 & 0 & 0 \\ 1 - 2q_1^2 - 2q_2^2 & 0 & 0 & 0 \end{bmatrix}_{\hat{\boldsymbol{x}}_{q,k}} T_s \tag{7.23}$$

$$\boldsymbol{H}_{lin,k} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times4} \\ \boldsymbol{0}_{4\times3} & \boldsymbol{0}_{4\times3} & \boldsymbol{H}_{q,lin,k} \end{bmatrix} \tag{7.24}$$

$$\boldsymbol{D}_{lin,k} = [\boldsymbol{0}_{4\times3}, \boldsymbol{D}_{q,lin,k}^T]^T \tag{7.25}$$

The strictly aligned quaternion values and initial position and velocity are used as initial values of the priori estimates.

### 7.3.4 Control system

Fig. 7.2 illustrates the hybrid system for a quadrotor under cyber attacks. $Z^{-1}$ represents the command in the last time step required to calculate the state estimates. The controller receives the ground signals and sensor readings to generate

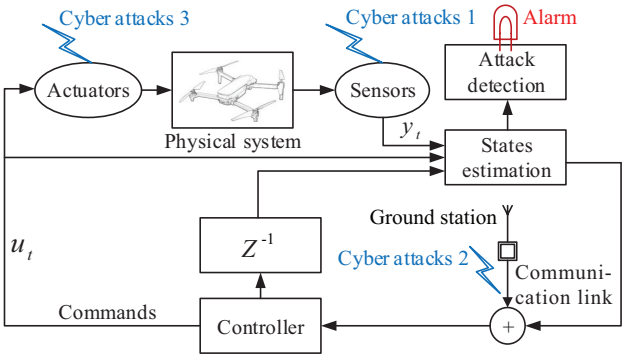



FIGURE 7.2: Illustration of the hybrid system for a quadrotor under cyber attacks.

trajectory onboard and outputs commands needed for trajectory tracking to the corresponding actuators that alter the states of system. A PID control law [334] is employed in this chapter, which can be expressed in the Laplace form as:

$$\bar{\boldsymbol{u}}_k(s) = (\boldsymbol{k}_p + \boldsymbol{k}_i/s + \boldsymbol{k}_d s)(\boldsymbol{y}_k - \boldsymbol{r}_k) \tag{7.26}$$

where $\bar{\boldsymbol{u}}$ is the desired command; $\boldsymbol{k}_p$, $\boldsymbol{k}_i$, and $\boldsymbol{k}_d$ are the proportional, integral and derivative gains respectively and $\boldsymbol{r}_k$ is the reference states at time $t_k$. Here, we only provide the desired trajectory for the quadrotor to track, i.e., $\boldsymbol{p}_E^*$, the desired position in the Earth frame. The PID parameters are fine tuned offline according to the dynamic response of flight and the tracking errors (bounded within $0.1m$) of the controlled system. Since $\omega_z$ does not affect the translational motion of the quadrotor, we can set $\omega_z = 0$ in (7.3). As shown in Fig. 7.2, the attacks to the quadrotor generally occur in the sensors and comunication links at the same time.

### 7.3.5 Attack model

Since our detector is derived from the EKF, the main objective of this chapter is to investigate the cyber attack detection and isolation problem for a discretized attack model described by:

$$\begin{cases} \boldsymbol{x}'_{k+1} = \boldsymbol{x}'_k + \boldsymbol{f}(\boldsymbol{x}'_k, \boldsymbol{u}_{a,k})T_s + \boldsymbol{w}_k T_s \\ \boldsymbol{y}'_k = \boldsymbol{h}(\boldsymbol{x}'_k, \boldsymbol{u}_{a,k}) + \boldsymbol{v}_k + \boldsymbol{y}_{a,k} \end{cases} \tag{7.27}$$

TABLE 7.1: Data-driven cyber attacks

| **Type** | **Definition** |
|---|---|
| Delayed attack | The data reaches the system's node with delayed time. |
| Cross site request forgery attack | The data collection has malicious data elements. |
| DoS attack | Meaningless data is being sent continuously to cause cyber traffic jam without the system's information. |
| FDI attack | The data introduces undetected errors into the states of the system fully or partially. |
| Replay attack | The data is recorded and used repeatedly. |
| Random attack | The data is randomly sampled to disturb the target system without the system's information. |

where $\boldsymbol{u}_{a,k} \in \mathbb{R}^p$ is the input vector when the actuators are being attacked. $\boldsymbol{u}_{a,k}$ can be expressed as:

$$\boldsymbol{u}_{a,k} = \boldsymbol{u}_k + \boldsymbol{a}_k \tag{7.28}$$

where $\boldsymbol{a}_k \in \mathbb{R}^p$ is the attack vector at time $t_k = kT_s$ and $\boldsymbol{a}_k = \boldsymbol{0}$ if no attacks is inflicted on the actuators. $\boldsymbol{y}_{a,k} \in \mathbb{R}^m$ is the attack signal on the sensors generated by the attacker at the moment $t = t_k$; $\boldsymbol{y}'_k$ and $\boldsymbol{x}'_k$ indicate the measurements and states of the system under cyber attacks. Note that if no attack occurs, the system (7.27) is equivalent to the plant (7.6).

Several data-driven cyber attacks that can be inflicted on drones are classified and described in Table 7.1. In this chapter, three main types of data-driven cyber attacks are studied: Random attack, False Data Injection (FDI) attack and Denial of Service (DoS) attack. The malicious data can be introduced via sensors, communication links or actuator channels. Generally, cyber attacks on sensors (Cyberattacks 1) or communication links (Cyberattacks 2) affect a feedback control system in the same way, as they both alter the feedback error signal.

#### 7.3.5.1 Random attack

The attacker attempts to disturb the sensors or actuators by sending a packet of random data without the knowledge of the targeted systems, trying to bypass the detector and attack the targeted systems. In this scenario, $\boldsymbol{a}_k$ in (7.28) and $\boldsymbol{y}_{a,k}$ in (7.27) are the random data generated by the attacker.

#### 7.3.5.2 False data injection (FDI) attack

The attacker has already identified the parameters of the system model, including the state function of the system, the parameters of the measurement function, and the Kalman gain $\boldsymbol{K}$. The attacker can control specific parts of the system through actuators and sensor readings to deceive the specific functions of the control system, such as GPS spoofing[335], error track guidance and attitude changes. As a result, specifically designed FDI attacks are more challenging to detect than random attacks and DoS attacks. The measurement equation in (7.27) under the FDI attack is established as:

$$\boldsymbol{y}'_k = \boldsymbol{h}(\boldsymbol{x}'_k, \boldsymbol{u}_{a,k}) + \boldsymbol{v}_k + \boldsymbol{\tau}\boldsymbol{y}_{a,k} \tag{7.29}$$

where $\boldsymbol{\tau}$=$diag(\gamma_1, ..., \gamma_m)$ represents the attack selection matrix in the sensor. If the sensor is attacked $\gamma_i = 1$, otherwise $\gamma_i = 0$. $\boldsymbol{y}_{a,k} \in \mathbb{R}^m$ is the malicious data from the attacker and $\boldsymbol{u}_{a,k} \neq \boldsymbol{0}$ if actuators are under attack. The attack sequences in [336] that can fool the chi-square detector is selected as:

$$\boldsymbol{y}_{a,k} - \boldsymbol{y}_{a,k-s} = -\rho\lambda_e^{k-s+1}\boldsymbol{y}^* \tag{7.30}$$

where, $s$ is a positive constant integer and $\rho$ is a coefficient that ensure the innovation is bounded below the threshold. $|\lambda_e| \geq 1$ is the unstable eigenvalue of $\boldsymbol{A}_{lin,k}$, and $\boldsymbol{y}^* = \boldsymbol{H}_{lin,k}\boldsymbol{v}_e$, where $\boldsymbol{v}_e$ is the eigenvector corresponding to $\lambda_e$.

#### 7.3.5.3 Denial of service (DoS) attack

The DoS attack is a kind of jamming approach, where the attacker blocks data transmission between sensors by sending a steady stream of data to override the communication link. As such, the control system cannot receive any signals from its sensors or external controller. Therefore, no control commands can be passed on to the actuators. DoS attack can occur in the sensor network, communication links and controller-actuators links. During a DoS attack on the quadrotor's sensors and actuators, $\boldsymbol{y}'_k = \boldsymbol{0}$ and $\boldsymbol{u}_{a,k} = \boldsymbol{0}$.

## 7.4 Attack detection and isolation with modified sliding innovation sequences

The chi-square detector is a widely used detection method based on the state estimation that operates by comparing the actual sensor readings with estimate values via a one-step iteration estimation, i.e., the state estimate is iterated once in a time step. However, the accuracy of the chi-square detector depends on the system model and there exist high frequency fluctuations in the chi-square detector when applied on a dynamic quadrotor. In this chapter, we extend the chi-square detector to formulate a more accurate detector, called the MSIS detector that can be applied on a dynamic quadrotor.

### 7.4.1 Chi-square detector

In the chi-square detector, a normalized random variable (indicator) that satisfies the chi-square probability distribution is constructed. The detection result is obtained by comparing the indicator with a threshold preset according to the standard chi-square table.

Let $\boldsymbol{\Delta x}_k := \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k$ denotes the state estimation error, combining (7.6)-(7.13), the error function can be described as:

$$\begin{aligned}\boldsymbol{\Delta x}_{k+1} = &(\boldsymbol{A}_{lin,k} - \boldsymbol{K}_{k+1}\boldsymbol{H}_{lin,k}\boldsymbol{A}_{lin,k})\boldsymbol{\Delta x}_k \\ &+ (\boldsymbol{I} - \boldsymbol{K}_{k+1}\boldsymbol{H}_{lin,k})\boldsymbol{w}_k T_s - \boldsymbol{K}_{k+1}\boldsymbol{v}_k\end{aligned} \tag{7.31}$$

**Theorem 7.1.** *The innovation $\boldsymbol{z}_k$ at time $t_k$ is defined as:*

$$\boldsymbol{z}_k := \boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k) \tag{7.32}$$

*and $p(\boldsymbol{z}_k) \sim \aleph(\boldsymbol{0}, \boldsymbol{\beta}_k)$ if the system operates normally, where covariance matrix $\boldsymbol{\beta}_k = \boldsymbol{H}_{lin,k}\boldsymbol{P}_{k|k-1}\boldsymbol{H}_{lin,k}^T + \boldsymbol{R}$.*

*Proof.* According to (7.6), (7.31), (7.32), and 1st Taylor expansion,

$$\begin{aligned}
\boldsymbol{z}_k &= \boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k) \\
&= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k) \\
&= \boldsymbol{h}(\boldsymbol{x}_{k-1} + \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})T_s + \boldsymbol{w}_{k-1}T_s, \boldsymbol{u}_k) + \boldsymbol{v}_k \\
&\quad - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1})T_s, \boldsymbol{u}_k) \\
&\approx \boldsymbol{H}_{lin,k-1}\boldsymbol{A}_{lin,k-1}\boldsymbol{\Delta x}_{k-1} + \boldsymbol{H}_{lin,k-1}\boldsymbol{w}_{k-1}T_s + \boldsymbol{v}_k \\
&= \boldsymbol{H}_{lin,k-1}(\boldsymbol{A}_{lin,k-1}\boldsymbol{\Delta x}_{k-1} + \boldsymbol{w}_{k-1}T_s) + \boldsymbol{v}_k
\end{aligned} \tag{7.33}$$

As mentioned, the estimate covariance of $\boldsymbol{\Delta x}_{k-1}$ is $\boldsymbol{P}_{k-1}$. The covariance of the process noise and the measurement noise are $\boldsymbol{Q}$ and $\boldsymbol{R}$ respectively. Thus, according to (7.9), the covariance matrix of $\boldsymbol{z}_k$ is $\boldsymbol{\beta}_k$. □

According to the chi-square detector, we can construct a scalar statistical indicator described as:

$$\vartheta_k = \boldsymbol{z}_k{}^T \boldsymbol{\beta}_k{}^{-1} \boldsymbol{z}_k \tag{7.34}$$

**Corollary 7.2.** *Since the innovation $\boldsymbol{z}_k$ satisfies the Guassian distribution $\aleph(\boldsymbol{0}, \boldsymbol{\beta}_k)$, then the scalar indicator $\vartheta_k$ satisfies the chi-square distribution given both the process and measurement noises satisfy the zero-mean Gaussian distribution.*

**Corollary 7.3.** *Since the indicator $\vartheta_k$ satisfies the chi-square distribution, the decision boundary for cyber attack detection is based on the hypotheses $H_0$ and $H_1$ as follows:*

$$\begin{aligned}
H_0: &\quad \vartheta_k \leq \chi^2_{\alpha,N\times m}, \quad \textit{no cyber attack} \\
H_1: &\quad \vartheta_k > \chi^2_{\alpha,N\times m}, \quad \textit{cyber attack occurs}
\end{aligned} \tag{7.35}$$

*i.e., if the indicator $\vartheta_k$ is greater than $\chi^2_{\alpha,N\times m}$, the system is considered to be under attack with false alarm probability $\alpha$. $\chi^2_{\alpha,N\times m}$ is determined from the standard chi-square table according to the significance level $\alpha$ and the degree of freedom $N \times m$, where $N$ is the sample number and $m$ is the dimension of the measurement vector.*

### 7.4.2 Modified sliding innovation sequences detector

A false alarm easily occurs when using the chi-square detector because high nonlinearities and complex maneuverings of dynamic quadrotors will produce large innovation values frequently. Also, the FDI attacks can be designed to generate the limit value $\lim_{t\to\infty} \|\hat{\boldsymbol{x}} - \boldsymbol{x}\| = \infty$ while $\lim_{t\to\infty} \|\boldsymbol{y}' - \boldsymbol{y}\| \leq 1$ [71], which means the chi-square detector may fail to detect these specifically designed FDI attacks. Therefore, to acquire enhanced performance based on the Kalman filter-based detection framework, the MSIS detector is proposed to detect the cyber attacks for both sensors and actuators. The MSIS detector comprises two phases: the calibration phase and the detection phase.

#### 7.4.2.1 Calibration phase

Due to the high nonlinearities that exist in the quadrotor dynamics when the quadrotor undergoes complex maneuvers (such as rapid turns), its EKF-based state estimation would produce large estimation errors at the beginning of the maneuver. The initial state estimation can be formulated as an optimization problem and the optimal state estimate within a sliding window is obtained by:

$$\hat{\boldsymbol{x}}_i^* = \underset{\hat{\boldsymbol{x}}_i(\boldsymbol{y}_i,\boldsymbol{u}_i)}{\operatorname{argmin}} \mathcal{J} := \sum_{i=\max(1,k-n+1)}^{k} (\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i) \tag{7.36}$$

The time window is defined as:

$$\kappa_k := [\max(1, k-n+1), ..., k-1, k] \tag{7.37}$$

where $n$ is the length of the sliding window. To obtain $\hat{\boldsymbol{x}}_i^*$, a calibration phase is introduced to allow the detector to obtain an accurate initial state estimate and an initial estimate covariance matrix. Here, an inner iteration-based optimization method is implemented which converts equations (7.11) and (7.13) in Step 3 to:

$$\boldsymbol{K}_k^i = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_{lin,k}^{i\,T}(\boldsymbol{H}_{lin,k}^i\boldsymbol{P}_{k|k-1}\boldsymbol{H}_{lin,k}^{i\,k} + \boldsymbol{R})^{-1} \tag{7.38}$$

$$\hat{\boldsymbol{x}}_k^{i+1} = \hat{\boldsymbol{x}}_k^i + \boldsymbol{K}_k^i(\boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^i, \boldsymbol{u}_k)) \tag{7.39}$$

where, $i \in \{0, 2, ..., \bar{N}-1\}$ is the iteration step, and the state estimate $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^N$ in the time step $t_k$. The iteration procedure stops when $\hat{\boldsymbol{x}}_k^{i+1}$ and $\hat{\boldsymbol{x}}_k^i$ differ by less than a preset threshold $\bar{\epsilon}$ or when $i = \bar{N}-1$ is satisfied. In the inner iteration procedure, $\boldsymbol{P}_{k|k-1}$, $\boldsymbol{u}_k$ and $\boldsymbol{y}_k$ remain unchanged while $\boldsymbol{H}_{lin,k}^i$ is changed with respect to $\hat{\boldsymbol{x}}_k^i$.

#### 7.4.2.2 Detection phase

In the detection phase, the state estimate and the estimate covariance matrix obtained from the calibration phase are used as an initial state estimate and an estimate covariance matrix in the EKF (step 1 in the section 7.3.2). An indicator is constructed with a modified innovation sequence within a sliding window. Compared to the chi-square detector which only uses an innovation value at each time, the MSIS detector utilizes a continuously updated innovation sequence within a sliding window of a fixed length.

When the system is under attack, $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ in (7.32) are replaced by $\boldsymbol{y}'_k$ and $\boldsymbol{z}'_k$. Under DoS attack, the effect on system sensors is equivalent to a long time packet loss. As such, the state estimate is equal to the prior state estimate, i.e., $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1}$. The estimate covariance matrix can then be expressed as:

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_{k+1|k} = \boldsymbol{A}_{lin,k}\boldsymbol{P}_k\boldsymbol{A}_{lin,k}^T + T_s\boldsymbol{Q}T_s \tag{7.40}$$

Since there are no sensor readings, the innovation also does not exist. In this case, we can set the sensor readings to the default value 0. As for the other two cyber attacks, random attacks and FDI attacks, the innovation $\boldsymbol{z}'_k$ is expressed as:

$$\boldsymbol{z}'_k = \boldsymbol{h}(\boldsymbol{x}'_k, \boldsymbol{u}_{a,k}) + \boldsymbol{v}_k + \tau\boldsymbol{y}_{a,k} - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k) \tag{7.41}$$

Without loss of generality, the innovation $\boldsymbol{z}'_k$ in the attack model can be expressed as:

$$\boldsymbol{z}'_k = \begin{cases} \boldsymbol{h}(\boldsymbol{x'}_k, \boldsymbol{u}_{a,k}) + \boldsymbol{v}_k + \tau\boldsymbol{y}_{a,k} - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, u_k), \\ \qquad\qquad\qquad\qquad if\ not\ DoS\ attack \\ -\boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k), \qquad\qquad if\ DoS\ attack \end{cases} \tag{7.42}$$

Notice that in the attack model, if no attacks occur, system (7.27) is equivalent to the normal system (7.6). Since Theorem 7.1 holds, $\boldsymbol{z}'_k = \boldsymbol{z}_k$ with the same

distribution, i.e., $p(\boldsymbol{z}'_k) \sim \aleph(\mathbf{0}, \boldsymbol{\beta}_k)$. Therefore, the normalized innovation of $\boldsymbol{z}'_k$ is:

$$\boldsymbol{e}_k = \begin{cases} {\boldsymbol{\beta}_k}^{-\frac{1}{2}} \boldsymbol{z}'_k, & if\ not\ DoS\ attack \\ -{\boldsymbol{\beta}_k}^{-\frac{1}{2}} \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k), & if\ DoS\ attack \end{cases} \tag{7.43}$$

In this chapter, a general operator norm is introduced into the normalized innovation sequence over a sliding window, and there come the modified sliding innovation sequences.

**Definition 7.1** (Modified Sliding Innovation Sequences)**.** The modified innovation sequence are denoted as $\boldsymbol{\zeta}_k$, and modified sliding innovation sequences are defined as a set $S(\boldsymbol{\zeta}_k) := \{\boldsymbol{\zeta}_k \mid k \in [1, \infty]\}$. $\boldsymbol{\zeta}_k$ is expressed as:

$$\boldsymbol{\zeta}_k := \{\zeta_{max(1,k-n+1)}, \zeta_{\max(1,k-n+1)+1}, ..., \zeta_k\} \tag{7.44}$$

where $\{\zeta_i = \rho(\boldsymbol{e}_i) \mid i \in [\max(1, k-n+1), k]\}$ and $\rho(\cdot)$ is a operator norm to $\boldsymbol{e}_i$.

The detection statistical indicator $\vartheta'_k$ at time $t_k$ in the MSIS detector is then expressed as:

$$\vartheta'_k = \frac{1+\lambda}{n} \sum \|\zeta_i\|, \quad \zeta_i \in \boldsymbol{\zeta_k} \tag{7.45}$$

where $\lambda$ represents the relaxation factor used to adjust the detection threshold.

**Theorem 7.4.** *The decision boundary for the MSIS detector with a confidence level* $1-\alpha$ *can be generated by:*

$$\begin{aligned} H_0 : &\quad \vartheta'_k \leq \epsilon, \quad no\ cyber\ attack \\ H_1 : &\quad \vartheta'_k > \epsilon, \quad cyber\ attack\ occurs \end{aligned} \tag{7.46}$$

*where, the detection threshold* $\epsilon$ *is the absolute value of z-score with the selected confidence level, i.e.,* $\epsilon = \|\mathbf{Z}_{\alpha/2}\|$.

*Proof.* Since $\boldsymbol{e}_i$ in (7.43) is a normalized innovation following a normal distribution, i.e., $p(\boldsymbol{e}_i) \sim \aleph(\mathbf{0}, \boldsymbol{I}_{m\times m})$, with operator norm $\rho(\cdot)$, the scalar $p(\zeta_i) \sim \aleph(0, 1)$ holds. Then the indicator without relaxation factor ($\lambda = 0$), which is the mean value of $\|\zeta_i\|$, $p(\vartheta'_k) \sim \aleph(0, 1)$ holds. Therefore, $\|\mathbf{Z}_{\alpha/2}\|$ can be used as the decision boundary with confidence level $1-\alpha$. □

Note that the relaxation factor $\lambda$ is chosen for real applications, which will not affect the validity of Theorem 7.4.

The MSIS detector is described in Algorithm 5. The time and space complexity of the MSIS detector are both $O(n)$, since we only need to store and summate the modified innovation sequence within the sliding window of fixed length $n$. Compared to the SUM detector whose complexity is $O(k)$, i.e., linear with the time steps $k$, the MSIS detector is much more time-efficient and space-efficient since the time steps $k$ will increase sharply during the real time flight.

**Algorithm 5:** Modified sliding innovation sequences estimation detection algorithm

**Initialization:** $\boldsymbol{P_0}, \hat{\boldsymbol{x}}_0, \boldsymbol{Q}, \boldsymbol{R}, T_s$;
**Input:** $\boldsymbol{y'_k}, \boldsymbol{u}_k, \boldsymbol{u}_{k-1}$;
**Result:** cyber attacks detection results
$t_k = 0$;
**while** $t_k \leq$ *End Time* **do**
  state estimation based on the EKF as (7.8), (7.9), (7.10) and (7.11);
  $i = 0$;
  **if** *calibration phase is not finished* **then**
    **while** $\|\hat{\boldsymbol{x}}_k^{i+1} - \hat{\boldsymbol{x}}_k^i\| > \bar{\epsilon}$ *and* $i < \bar{N}$ **do**
      obtain an optimal state estimate and an estimate covariance matrix iteratively as (7.38), (7.39);
      $i \leftarrow i + 1$;
    **end**
  **else**
    ;
  **end**
  obtain the state estimate $\hat{\boldsymbol{x}}_k$ and the covariance matrix $\boldsymbol{P}_k$ as (7.13) and (7.12);
  calculate the statistical indicator $\vartheta'_k$ as equation (7.45);
  $t_k \leftarrow t_k + T_s$;
  **if** $\vartheta'_k > \epsilon$ **then**
    output alarm triggered;
    continue;
  **else**
    continue;
  **end**
**end**

### 7.4.3 Sample covariance-based cyber attack isolation

Each cyber attack is a type of signal that alters the system's responses by causing drifts in the state, abnormal measurements, sudden shifts in the measurement, and an increase of background noise. Specifically, cyber attacks will affect the stochastic characteristics of systems differently, and displace the zero mean and unit covariance of the normalized innovation with the stochastic characteristics of cyber attacks. Therefore, we can recognize the type of attacks by evaluating the covariance of normalized innovation sequence. The covariance of normalized innovation sequence is denoted as $\boldsymbol{P}_e$ where $\boldsymbol{P}_e = E\{\boldsymbol{e}_k^T \boldsymbol{e}_j\} - E\{\boldsymbol{e}_k\}E\{\boldsymbol{e}_j\}$. The trace of the sample covariance matrix $\boldsymbol{S}_k$ can be used to estimate $\boldsymbol{P}_e$:

$$\boldsymbol{S}_k = \frac{1}{n-1} \sum_{i=\max(1,k-n+1)}^{k} (\boldsymbol{e}_i - \bar{\boldsymbol{e}}_k)(\boldsymbol{e}_i - \bar{\boldsymbol{e}}_k)^T \tag{7.47}$$

where $\bar{\boldsymbol{e}}_k$ is the sample mean of the normalized innovation sequence at time $t_k$ and

$$\bar{\boldsymbol{e}}_k = \frac{1}{n} \sum_{i=\max(1,k-n+1)}^{k} \boldsymbol{e}_i \tag{7.48}$$

We can compare sample covariance matrices of different cyber attacks and isolate these cyber attacks according to the relative magnitudes of their sample covariance matrices.

Also, we can isolate the locations where the cyber attacks occur by decoupling the normalized innovation. With the decoupled normalized innovation used in (7.45), the cyber attacks occur on position sensors can be detected via $\boldsymbol{e}_k = [e_{k,1}, e_{k,2}, e_{k,3}]^T$ calculated with the corresponding position senor readings while cyber attacks on the IMU sensor can be detected via $\boldsymbol{e}_k = [e_{k,4}, e_{k,5}, e_{k,6}, e_{k,7}]^T$ from the selected IMU sensor readings, where $e_{k,j}, j \in \{1, 2, ..., 7\}$ is the $jth$ component of the normalized innovation $\boldsymbol{e}_k$ in (7.43).

## 7.5 Simulation results

To illustrate the cyber attack detection and isolation results, the MSIS detector method is compared to two other methods, namely the chi-square detector which is

TABLE 7.2: Simulation parameters

| **Parameter** | **Value** |
|---|---|
| The mass of quadrotor $mass$ | $0.25kg$ |
| Minimum normalized thrust $f_{min}$ | $5m/s^2$ |
| Maximum normalized thrust $f_{max}$ | $25m/s^2$ |
| Maximum angular velocity $\omega_{max}$ | $20rad/s$ |
| Initial state estimate $\hat{\boldsymbol{x}}_0$ | $[0, 0, 2m, 0, 0, 0, 1, 0, 0, 0]^T$ |
| Initial estimate covariance matrix $\boldsymbol{P}_0$ | $1.0 \times 10^{-5}\boldsymbol{I}_{10\times10}$ |
| Process noise covariance matrix $\boldsymbol{Q}$ | $1.0 \times 10^{-1}\boldsymbol{I}_{10\times10}$ |
| Measurement noise covariance matrix $\boldsymbol{R}$ | $1.0 \times 10^{-3}\boldsymbol{I}_{7\times7}$ |
| Simulation frequency | $100HZ$ |
| Process noise power $NP_p$ | $1.0 \times 10^{-3}\boldsymbol{I}_{10}$ |
| Measurement noise power $NP_m$ | $1.0 \times 10^{-5}\boldsymbol{I}_{7}$ |
| Significance level $1-\alpha$ | 0.9973 |
| Alarm threshold in the $\chi^2$ detector $\chi^2_{\alpha,7}$ | 18.475 |
| Relaxation factor in the MSIS detector $\lambda$ | $-0.09$ |
| Sliding window length $n$ | 10 |
| Calibration iteration error threshold $\bar{\epsilon}$ | 0.01 |
| Maximum calibration iteration number $\bar{N}$ | 20 |
| PID gains for position $z$, $[k_{p,z}, k_{i,z}, k_{d,z}]$ | $[3.05, 28.4, 10.2]$ |
| PID gains for position $y$, $[k_{p,y}, k_{i,y}, k_{d,y}]$ | $[2.6, 0.0, 10.5]$ |
| PID gains for position $x$, $[k_{p,x}, k_{i,x}, k_{d,x}]$ | $[2.6, 0.0, 10.5]$ |
| Total simulation time $t_{total}$ | $16s$ |
| Time period of cyber attacks $\{t_a\}$ | $1.0s \sim 6.0s$ |

widely used and the SUM detector developed in [70]. Moreover, two operator norms are used for the MSIS detector, which are the infinity norm ($l_\infty$), i.e., $\zeta_i = \max|\boldsymbol{e}_i|$ in (7.45) and the Euclidean norm ($l_2$). This comparative analysis serves to show the effectiveness of the infinity norm over the commonly used Euclidean norm for the MSIS detector. Detection for random attacks, FDI attacks and DoS attacks on the IMU sensor, position sensors and actuators are tested. For the following discussion, $\chi^2$ denotes the chi-square detector and SUM represents the SUM detector. MSIS-L2 means the modified sliding innovation sequences detector with Euclidean operator norm while MSIS-LI denotes the modified sliding innovation sequences detector with infinity operator norm.

In this work, simulation tests are conducted over the SIMULINK platform running on a laptop with CPU AMD R5-4600H and 16GB DDR4 RAM. The desired trajectory is generated by Python 3.6, which provides the desired global positions with time series. The quadrotor drone dynamics model, the attack model and the attack

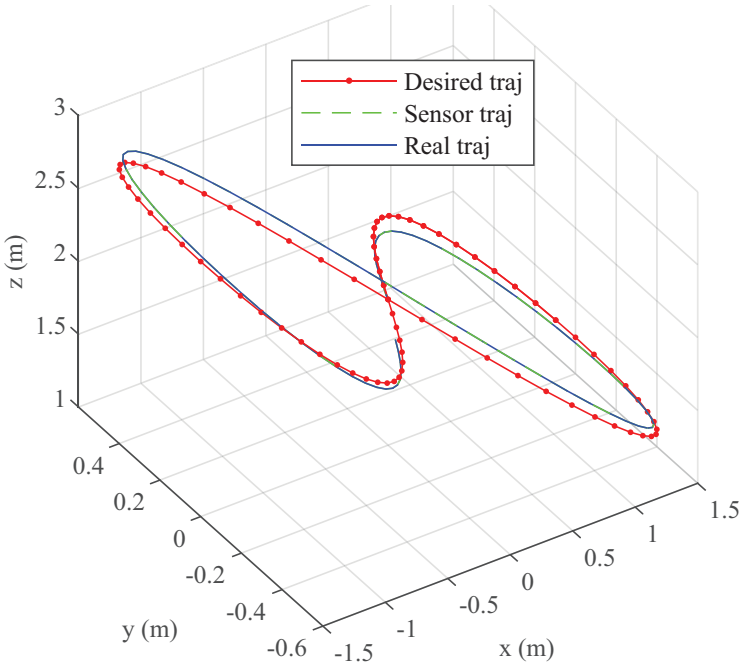



FIGURE 7.3: 3D trajectories (the desired trajectory, the trajectory from position sensors and the real trajectory) of the quadrotor in the absence of cyber attacks.

detector are simulated via S-functions in the MATLAB. The mass of the quadrotor, $mass$, is set to $0.25kg$. The simulation frequency in SIMULINK is 100Hz, and Band-Limited White Noise blocks are used to generate normally distributed random process noise with noise power $NP_p = 1.0 \times 10^{-3}\boldsymbol{I}_{10}$ and measurement noise with noise power $NP_m = 1.0 \times 10^{-5}\boldsymbol{I}_7$ since $NoisePower = T_s \times Covariance$. The PID parameters for each control channel are fine-tuned according to the dynamic response and the steady-state error of the controlled quadrotor, which are similar to the parameters used in [334]. The false alarm probability $\alpha$, i.e., significance level, satisfies $1 - \alpha = 0.9973$. For the calibration phase, the iteration error threshold is $\bar{\epsilon} = 0.01$ and the maximum iteration number is $\bar{N} = 20$. The length of the sliding window is $n = 10$. The total simulation time is $t_{total} = 16s$ and cyber attacks occur at $t_a = 1.0s \sim 6.0s$. The detailed simulation parameters can be found in Table 7.2.

Since the translational states and the rotational states can be decoupled, we let the IMU sensor, position sensors and actuators be attacked separately. For actuators, we only consider cyber attacks on the total thrust command $f$. The random attack on the IMU sensor, position sensors, and the thrust actuator are $\boldsymbol{y}_{a,k} = [\boldsymbol{0}_3, 0.5rand(4)]^T$, $\boldsymbol{y}_{a,k} = [0.5rand(3), \boldsymbol{0}_4]^T$, and $\boldsymbol{y}_{a,k} = 0.5rand(1)$ respectively, where $rand(n)$ is an uniformly random number of $n$ dimension generated by a MATLAB function. According to the eigenvalues of $A_{lin,k}$, the position states $\boldsymbol{p_E}$

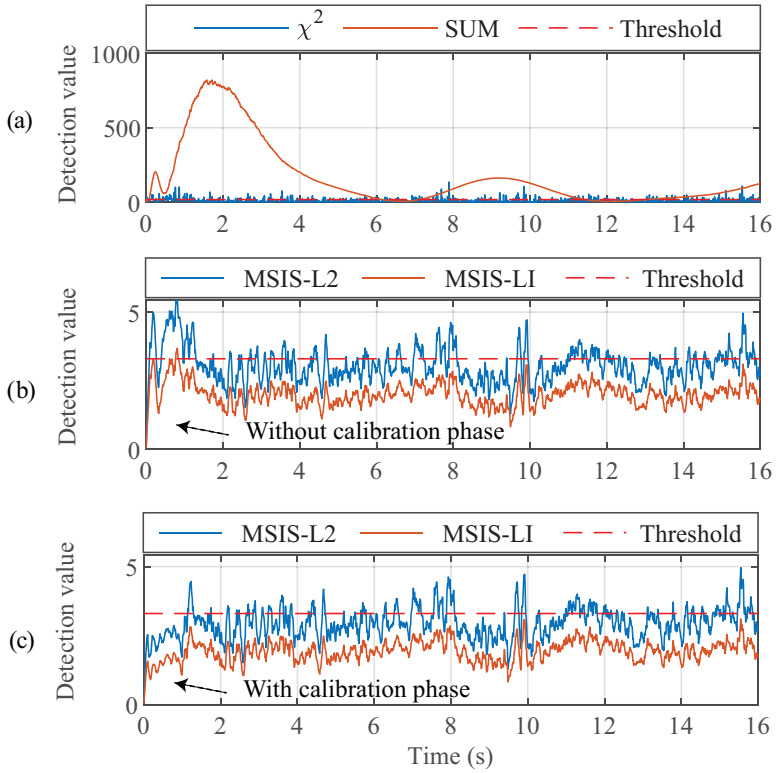
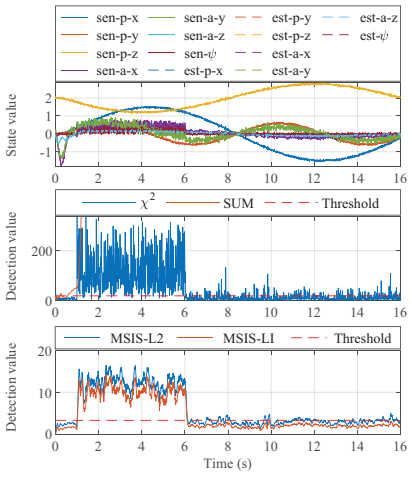


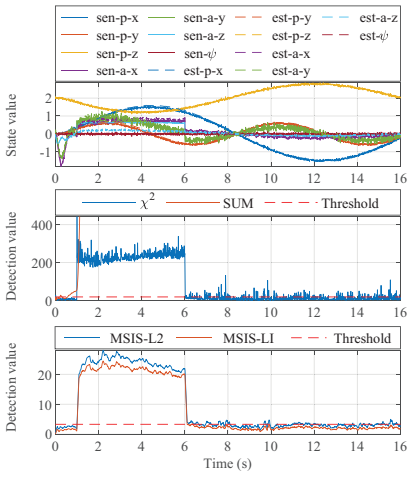

FIGURE 7.4: Detection results of the quadrotor in the absence of cyber attacks. **(a)** the chi-square detector and the SUM detector; **(b)** the MSIS detectors without calibration phase; **(c)** the MSIS detectors with calibration phase.

is perfectly attackable. However, since the parameters of the quadrotor are hard to obtain in real-time applications, we assume $\rho = 0$ in (7.30). Therefore, for FDI attack, we just choose a constant false data vector. The false data vector on IMU is $\tau \boldsymbol{y}_{a,k} = [0.5; 0.5; 0.5; 0.0]$, on position sensors, is $\tau \boldsymbol{y}_{a,k} = [0; 0; -0.367]$, and on thrust actuator, is $\tau \boldsymbol{y}_{a,k} = 2$. The attack frequency is the same as the simulation frequency. The calibration phase is initiated for the first $1s$. The period of the calibration time does not affect the accuracy of the MSIS detector.

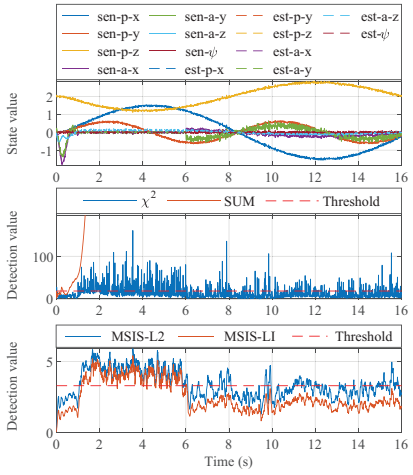
In the trajectory tracking motion scenario, the quadrotor is controlled to track a Figure-8 trajectory periodically. The period time of the desired trajectory is $T_p = 16s$. The trajectory of the quadrotor in the absence of cyber attacks are shown in Fig. 7.3. The detection results for quadrotor in the absence of cyber attacks are illustrated in Fig. 7.4. The detection results of cyber attacks on the IMU sensor, cyber attacks on the thrust actuator, and cyber attacks on position sensors are shown in Fig. 7.5, Fig. 7.6 and Fig. 7.7 respectively.




(a) Random attack




(b) FDI attack

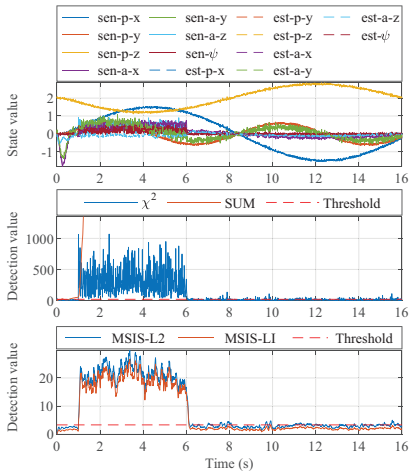



(c) DoS attack

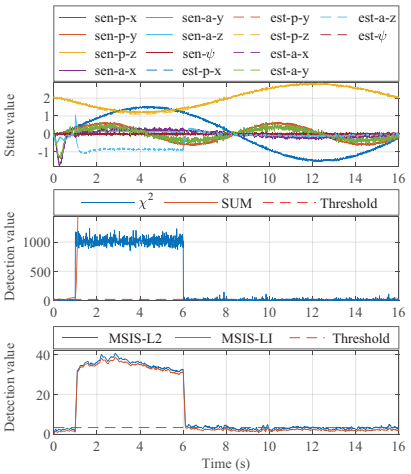
FIGURE 7.5: Detection of cyber attacks on the IMU sensor. In each sub-figure, **top**: the sensor readings (e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **middle**: the chi-square detector and the SUM detector; **bottom**: the MSIS detectors with Euclidean norm (MSIS-L2) and infinity norm (MSIS-LI).

### 7.5.1 Absence of cyber attacks

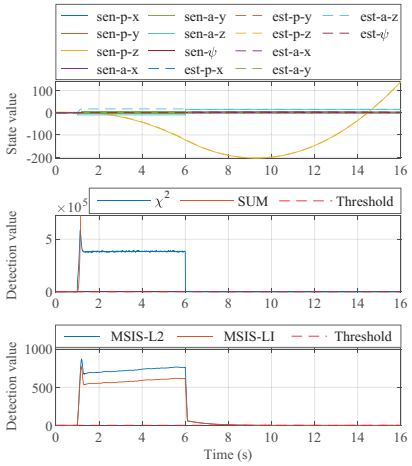
Although there is no cyber attack inflicted on the system, a false alarm is obtained throughout the test for both SUM and chi-square detectors (see Fig. 7.4(a)). The quadrotor dynamics along the complex trajectory produce the large innovation values that trigger the false alarm. This shows that both SUM and chi-square detectors are not applicable for drone applications. Using the MSIS-LI detector (Fig. 7.4(c)), no false alarm is triggered. The MSIS-LI reflects the accurate security status of the quadrotor. Although the false alarm is triggered for MSIS-L2, this is




(a) Random attack



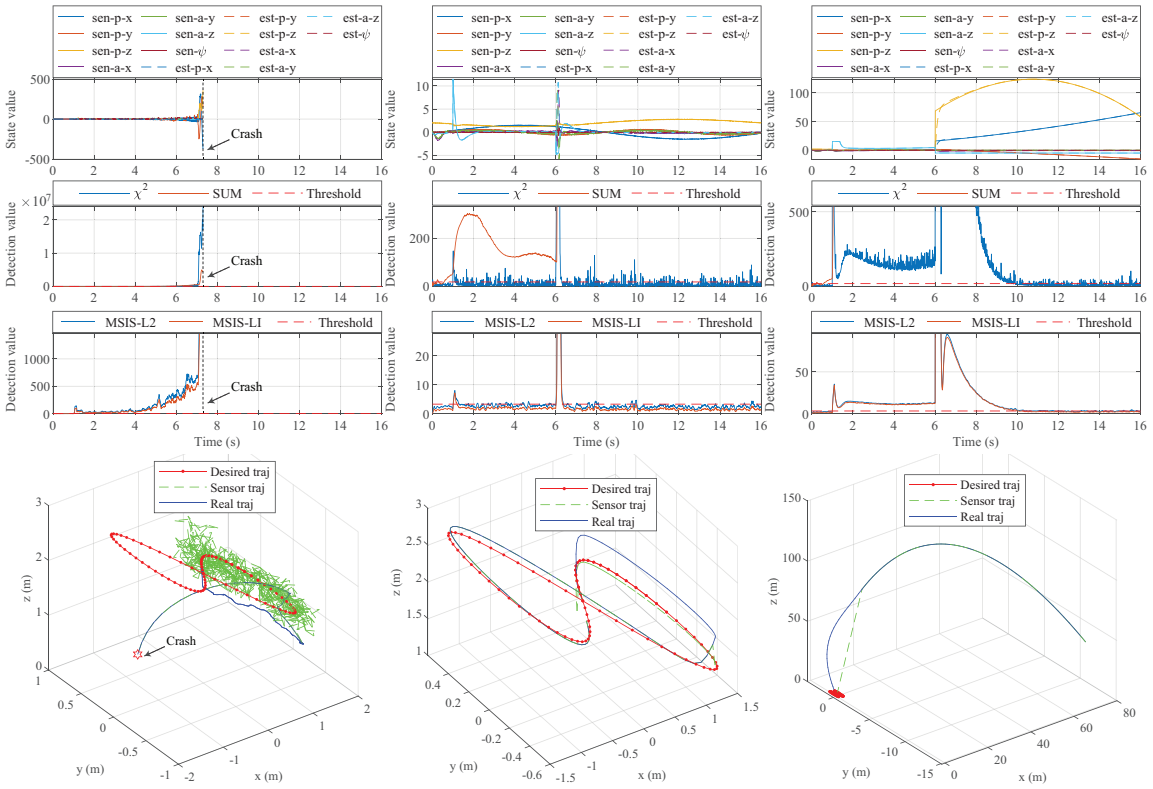

(b) FDI attack




(c) DoS attack

FIGURE 7.6: Detection of cyber attacks on the thrust actuator. In each subfigure (a), (b) and (c), **top**: the sensor readings a(e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **middle**: the chi-square detector and the SUM detector; **bottom**: the MSIS detectors with Euclidean norm (MSIS-L2) and infinity norm (MSIS-LI).

due to the preset threshold strictly specified to illustrate the better performance of using the infinity norm ($l_\infty$) over the Euclidean norm ($l_2$). The MSIS detectors incorporate the calibration phase which is conducted at the beginning of the test for 1 second. Comparing Fig. 7.4(b) (without calibration phase) and Fig. 7.4(c) (with calibration phase), without the calibration phase, the indicator detection values for the MSIS detectors are shown to be much higher at the beginning (first 1 second) of the test that would trigger the false alarm. This necessitates the inclusion of the calibration phase into the MSIS framework to reduce the false alarm rate while the quadrotor undergoes its maneuver along a complex trajectory. In the following




(a) Random attack (b) FDI attack (c) DoS attack

FIGURE 7.7: Detection of cyber attacks on position sensors. In each sub-figure (a), (b) and (c), from top to bottom, **1**: the sensor readings (e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **2**: the chi-square detector and the SUM detector; **3**: The MSIS detectors; **4**: The 3D trajectory of the quadrotor. In (a), the quadrotor becomes unstable and crashes to the ground under the random attack.

TABLE 7.3: Performance comparison with $n = 10$, $k = 800$

| **Detector** | **Computation Time** | **Memory Cost** | **Accuracy** |
|---|---|---|---|
| $\chi^2$ | $0.011ms$ | 8 *Bytes* | 87.5% |
| SUM | $0.022ms$ | 19200 *Bytes* | 34.9% |
| MSIS-L2 | $0.014ms$ | 80 *Bytes* | 81.2% |
| MSIS-LI | $0.014ms$ | 80 *Bytes* | 99.3% |

sections, for the sake of completeness, the detection results of the MSIS detector are compared with the SUM and chi-square detectors to further show the superiority of the MSIS detection method.

TABLE 7.4: Performance of the MSIS-LI detector with different $n$

| n | Computation Time | Lag Time | Accuracy |
|---|---|---|---|
| 1 | $0.0132ms$ | $0ms$ | 93.3% |
| 10 | $0.0137ms$ | $20ms$ | 99.3% |
| 100 | $0.0140ms$ | $180ms$ | 93.4% |

### 7.5.2 Cyber attacks on IMU sensor

#### 7.5.2.1 Random attack

For the chi-square detector, there is a high frequency fluctuation during the attack period (see Fig. 7.5(a)). As such, the alarm is frequently triggered even when there is no attack as shown in Fig. 7.4(a). The alarm of the SUM detector is triggered at $0.5s$ and remains triggered throughout the test. The detector value rises to $2.4 \times 10^4$ at $6s$ and remains above the threshold. This means the SUM detector can only detect the attack at the beginning. Therefore, the security status of the dynamic quadrotor cannot be reflected accurately using the chi-square detector and the SUM detector in real time. The MSIS detectors show a higher level of detection performance (see Table 7.3). However, the MSIS-LI is more reliable than MSIS-L2 for the same preset threshold because the alarm for MSIS-LI remains untriggered after the attack (no false alarm). We can also observe that there is a small time lag in detecting the cyber attack with both MSIS detectors. The alarm is triggered 0.02 seconds after the attack occurs and it stops 0.09 seconds after the end of the attack. The choice of $n$ will affect the performance of the MSIS-LI detector (see Table 7.4) and can be adjusted according to the specific scenarios.

#### 7.5.2.2 FDI attack

In Fig. 7.5(b), only the MSIS detectors can accurately detect the attack throughout the test. The chi-square detector manages to detect the attack but with a high false alarm rate. The SUM detector value rises to a maximum of approximately $8 \times 10^4$ at $6s$ and falls to about $3 \times 10^4$ at the end of the simulation. Therefore, it cannot reflect the security status of the quadrotor accurately in real time. Here, we can observe a similar time lag in the MSIS detectors, where the alarm lags by $0.11s$ for MSIS-LI and by $0.44s$ for MSIS-L2 after the end of the attack. The detection

value remains relatively constant during the period under attack, which is a useful characteristic for the implementation of the isolation technique in section 7.4.3.

#### 7.5.2.3 DoS attack

In the DoS attack, the controller receives no sensor readings. In Fig. 7.5(c), the MSIS-LI has the highest accuracy as it successfully triggers the alarm during the attack period. The MSIS-L2 shows an increased false alarm rate, while the chi-square detector and the SUM detector cannot detect the DoS attack accurately.

### 7.5.3 Cyber attacks on thrust actuator

The detection results for attacks on the thrust actuator in Fig. 7.6 are similar to that on the IMU sensor, such that the MSIS-LI detector accurately detects the attacks for the entire test period whereas the SUM and chi-square detectors are not able to prevent the false alarm after the attacks ended. As compared to the DoS attack on the IMU sensor (Fig. 7.5(c)), the false alarm rate in the MSIS-L2 detector is lower for the DoS attack on the thrust actuator. There is a time lag for both MSIS detectors in stopping the alarm after the end of the DoS attack, i.e., $2.31s$ for the MSIS-LI and $2.36s$ for the MSIS-L2.

### 7.5.4 Cyber attacks on position sensors

Fig. 7.7 shows the detection results for cyber attacks on position sensors and the trajectory of the quadrotor. Attacks on position sensors will cause the trajectory of the quadrotor to deviate from the desired trajectory. For the random attack, the quadrotor becomes unstable and finally crashes to the ground (see Fig. 7.7(a)). In Fig. 7.7(b), The MSIS detector can accurately identify the start and the end of the FDI attack while the SUM detector can only indicate the start of the FDI attack. As for the DoS attack in Fig. 7.7(c), there is a large time lag in the MSIS detector in identifying the end of the attack, which is $4.1s$ for the MSIS-LI and $4.18s$ for the MSIS-L2. Similar to the earlier discussion, the SUM detector ceases to work after the initial detection.

Although the MSIS detector manages to identify the start and end points of the FDI attack, it fails during the attack period in between the start and end points. Therefore, the limitation of the MSIS detector lies in the detection of specially designed FDI attacks where the drone's system parameters are known to the attacker. To overcome the limitation, a signal holder can be designed to keep the alarm triggered until the next indicator signal peak arrives.

### 7.5.5 Isolation of cyber attacks

Results of the isolation capability of the three types of data-driven cyber attacks are shown in Fig. 7.8, where random attack, FDI attack and DoS attack are inflicted on the thrust actuator at time periods $[1s, 2s]$, $[3s, 4s]$ and $[5s, 6s]$, respectively. In addition, we investigated the effect of the sliding window length $n$ on the lag time and the isolation process. The time window length is $n = 100$ and $n = 10$ in Fig. 7.8(c) and Fig. 7.8(d) respectively. The MSIS detector with $n = 100$ fails to isolate the cyber attacks because of the longer lag time, but the MSIS detector with $n = 10$ can isolate these cyber attacks successfully. In Fig. 7.8(d), using the MSIS, we can recognize these three types of data-driven cyber attacks by observing the relative magnitudes of the sample covariance of normalized innovation sequences after the alarm is triggered. During the alarm, $\boldsymbol{S}_{DoS,k} > \boldsymbol{S}_{R,k} > \boldsymbol{S}_{FDI,k}$ where $\boldsymbol{S}_{R,k}$, $\boldsymbol{S}_{FDI,k}$ and $\boldsymbol{S}_{DoS,k}$ denote the sample covariance of normalized innovation sequences for random attack, FDI attack and DoS attack respectively. Note that at the two end points of the alarm, we cannot recognize these cyber attacks immediately.

### 7.5.6 Summary

The chi-square detector and the SUM detector are not applicable for use in complex maneuver conditions as they cannot reflect the security status of the quadrotor accurately. As the SUM detector takes the summation of the whole historical innovation sequence within the window, the innovation sequence cannot be normalized by its covariance at each step. Therefore, the SUM detector fails even when no attack occurs for a dynamic quadrotor. On the other hand, the chi-square detection method has high frequency fluctuations that triggers the false alarm even when no attack occurs for a quadrotor maneuvering in a complex trajectory. Overall,

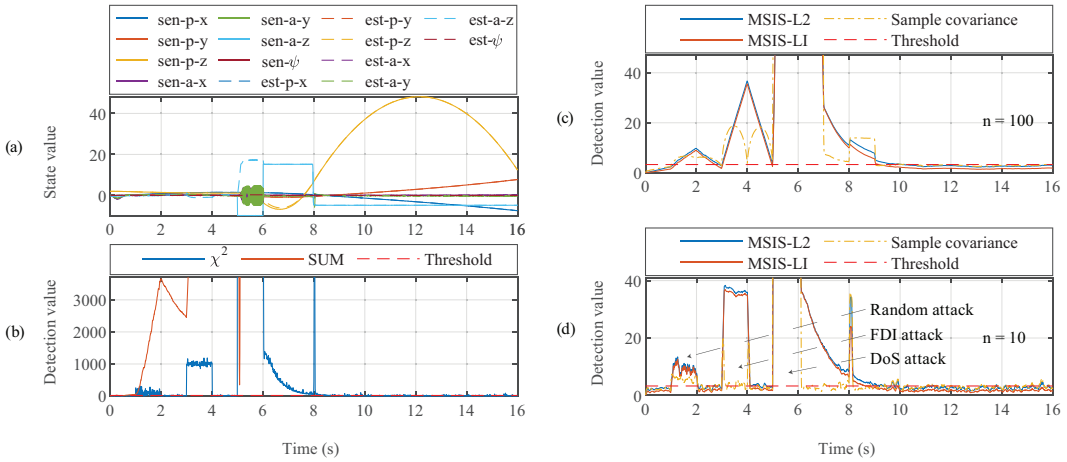



FIGURE 7.8: Isolation of three types of cyber attacks on thrust actuator for a dynamic quadrotor. **(a)** the sensor readings (e.g. sen-p-x is the sensor value of position $x$) and the state estimates (e.g. est-p-x is the estimate of position $x$); **(b)** the chi-square detector and the SUM detector; **(c)** the MSIS detectors with window length $n = 100$; **(d)** the MSIS detectors with window length $n = 10$.

compared to the chi-square detector and the SUM detector, the MSIS detector will be more stable and effective for drone applications.

Even though there is a certain time lag with the MSIS detector, it can be adjusted by choosing a suitable time window length to reduce the lag time. With a same threshold preset, the MSIS-LI is shown to be more reliable than the MSIS-L2 detector. Without the calibration phase, there exist apparent false alarms at the beginning where the quadrotor initiates the periodic maneuver. The MSIS detector with calibration phase can mitigate the fluctuations caused by maneuvering and detect the existing cyber attacks effectively. However, the detector has limitations in the performance as its performance will degrade if the period time of the desired trajectory is reduced or if the angular velocity is increased to a certain extent due to much higher nonlinearities found in higher speed rotation flight.

## 7.6 Conclusion

In this chapter, we developed the Modified Sliding Innovation Sequences (MSIS) detector based on an EKF optimal estimation method to detect and isolate cyber attacks inflicted on the sensors and actuators of a quadrotor drone. By applying the operator norm to the normalized innovation sequence within a sliding window, state estimation-based cyber attack detection on a quadrotor undergoing rapid turns in

a complex trajectory has been made possible with a reduced false alarm rate. Compared to the chi-square detector and the SUM detector, simulation results of a quadrotor undergoing periodic trajectory tracking showed that the MSIS detector can reflect the real-time security status of the quadrotor by detecting the start and end of the cyber attacks accordingly. The MSIS detector can effectively mitigate the high frequency fluctuations that exist in the chi-square detector. By implementing the calibration phase with an iteration procedure, the MSIS detector becomes more accurate and feasible for real-time quadrotor applications. Furthermore, we verified the capability of the MSIS detector for cyber attack isolation by evaluating the sample covariance of the normalized innovation sequence. However, the abnormal data introduced by adversarial attacks to the vision-based learning has not been explored. In the future work, we will focus on improving the accuracy of the detection method for agile quadrotor maneuvers under the specially designed FDI attacks and adversarial attacks into the vision-based learning model where real-time parameters of the quadrotor drone are obtained by the attacker.

# Chapter 8

# Conclusions and future work

This chapter summarizes the findings of this thesis and provides open questions for future research.

## 8.1 Conclusions

This thesis presents significant contributions to vision-based learning for drones, addressing key challenges in collaborative and adversarial environments. It introduces the ACEMSL framework, enhancing drone swarm efficiency and collaboration, and the POMA framework, which solves credit assignment and variable observation issues in visual drone swarms. Both frameworks have shown effectiveness in simulations and real-world applications, proving their potential for multiagent systems in real-world applications. Additionally, this thesis extends to multi-drone detection and tracking in adversarial settings with the creation of the ICG-Drone dataset and the Drone-YOLOSORT system, improving drone operational capabilities. The AMS-DRL approach for multi-pursuit evasion demonstrates the power of deep adversarial networks in drone safety, while the MSIS detector offers a robust solution for cyber attack detection and isolation, highlighting the importance of security in drone operations. These contributions lay the groundwork for future research in drone technology and its applications. The specific conclusions are elaborated in the following.

Chapter 2 has comprehensively explored the rapidly growing field of vision-based learning for drones, particularly emphasizing their evolving role in multi-drone systems and complex environments such as search and rescue missions and adversarial settings. The investigation revealed that drones are increasingly becoming sophisticated, autonomous systems capable of intricate tasks, largely driven by advancements in AI, machine learning, and sensor technology. Crucially, the integration of visual perception with machine learning algorithms, including deep reinforcement learning, opens up new avenues for drones to operate with greater efficiency and intelligence. The review categorized vision-based control methods into indirect, semi-direct, and end-to-end methods, offering a nuanced understanding of how drones perceive and interact with their environment. Applications of vision-based learning drones, spanning from single-agent to multiagent and heterogeneous systems, demonstrate their versatility and potential in various sectors, including agriculture, industrial inspection, and emergency response. However, this review also brings forth challenges such as sample efficiency, credit assignment in multi-drone systems, and ensuring effective deployment of drones in real-world scenarios. Safety and security, especially in the context of adversarial environments, remain paramount concerns that need ongoing attention. These challenges formulate the motivations for this thesis.

Chapter 3 has investigated sample efficiency and collaboration with sparse reward for a visual drone swarm in collaborative environments. We propose a data-efficient RL-based approach called ACEMSL to address the challenges of collaborative target search (CTS) for a visual drone swarm in a cluttered 3D space of sparse reward. Our approach improves upon existing curriculum learning by introducing a novel AEC algorithm and integrating it into task-specific multistage learning. Without prior knowledge of the target, the drone swarm performs CTS using only local visual perception and egocentric observations. Simulation results show that ACEMSL outperforms SOTA direct RL methods and the multistage learning CM3 in policy optimality, training runtime, as well as search efficiency. We further demonstrate the generalization and Sim2Real capabilities of our trained models with real-world flight tests in an unseen physical environment. The experiment results verify the effectiveness of a visual drone swarm performing CTS based on the metrics of SR and TTR. To the best of our knowledge, this work is the first to achieve and demonstrate CTS with a visual drone swarm in the real world.

Chapter 4 has further studied the credit assignment and variable observation for a visual drone swarm in collaborative environments. We present the POMA framework, an advanced solution that leverages MADRL for the complex problem of collaborative multitarget search and navigation. Our proposed method incorporates novel adaptive curriculum learning based on results from Chapter 3 and mixed individual-group credit assignment mechanisms that balance individual and group contributions in sparse reward environments. Simultaneously, the attention mechanism in POMA refines variable local observations, leading to significant enhancements in collaboration and scalability. Experimental results conducted on drone simulations demonstrate that our model attains superior performance over baseline methods. We further deploy the trained model over a visual drone swarm and conduct physical tests on different missions. Real-world flight experiments in various missions demonstrate the effectiveness and generalization capability of our approach. The framework's ability to balance individual and group contributions in sparse reward environments, coupled with its demonstrated effectiveness in drone swarm management for real-world applications, underscores its potential to enhance efficiency, decision-making, and adaptability across different diverse and critical fields when collaborative behaviors are required.

Chapter 5 has realized real-time multi-drone detection and tracking for visual drones in adversarial environments. We propose a whole life cycle multi-drone detection and tracking system designed for drone pursuit-evasion operations. We created a new drone dataset, "ICG-Drone", which covers a range of indoor and outdoor scenarios. We benchmarked the performance of several object detection models, such as YOLOv5, YOLOv8, and Swin Transformer, and then fine-tuned the most promising model, YOLOv5, using a GA search strategy. Our edge-accelerated Drone-YOLOSORT system meets real-time performance criteria with about 24.3 FPS, accelerating inference speed by 6 times through TensorRT during inferencing, and the developed ROS package ensures seamless integration into existing visual drone systems for complex multi-pursuit evasion tasks in adversarial environments.

Chapter 6 has developed a learning-based multi-pursuit evasion approach for multi-drone systems in adversarial environments. With the assumption of perfect relative position information from accurate 3D object detection and tracking, we present a

novel approach, AMS-DRL, for ensuring the safe navigation of drones under physical attacks from multiple pursuers. The AMS-DRL method trains a deep adversarial neural network to learn from the actions of the pursuers and respond effectively to avoid collisions and reach the target destination. Through extensive simulations and physical experiments, our results show that this approach outperforms existing learning-based baselines in terms of navigation success rate. Additionally, the success rate heatmap provides valuable insights into the impact of spatial geometry on drone navigation performance. This research highlights the benefits of DRL in solving complex and challenging problems in drone navigation. In addition, the performance of the learning-based chaser policy provides benchmarks for research on anti-drone techniques with DRL.

Chapter 7 has addressed cyber attack detection and isolation problem for drones in adversarial environments. We developed the Modified Sliding Innovation Sequences (MSIS) detector based on an EKF optimal estimation method to detect and isolate cyber attacks inflicted on the sensors and actuators of a quadrotor drone. By applying the operator norm to the normalized innovation sequence within a sliding window, state estimation-based cyber attack detection on a quadrotor undergoing rapid turns in a complex trajectory has been made possible with a reduced false alarm rate. Compared to the chi-square detector and the SUM detector, simulation results of a quadrotor undergoing periodic trajectory tracking showed that the MSIS detector can reflect the real-time security status of the quadrotor by detecting the start and end of the cyber attacks accordingly. The MSIS detector can effectively mitigate the high frequency fluctuations that exist in the chi-square detector. By implementing the calibration phase with an iteration procedure, the MSIS detector becomes more accurate and feasible for real-time quadrotor applications. Furthermore, we verified the capability of the MSIS detector for cyber attack isolation by evaluating the sample covariance of the normalized innovation sequence.

## 8.2 Future work

While this thesis has made significant contributions in the domain of vision-based learning for drones, numerous challenges remain to be addressed that impede its real-world applicability on a large scale. These challenges span various aspects,

from data collection and simulation accuracy to operational efficiency and safety concerns. This section provides several insights for future research.

**Large scale dataset:** A major impediment in the field is the absence of a comprehensive, public dataset similar to Open X-Embodiment [337] in robotic manipulation. This unified dataset should ideally encompass a wide range of scenarios and tasks to facilitate generalizable learning for vision-based drones. A potential solution is the collaborative development of a diverse, multi-purpose dataset by academic and industry stakeholders, incorporating various environmental, weather, and lighting conditions. Thus, the continued development of a unified vision-drone dateset and standard benchmarks can greatly improve the generalization capability of vision-based learning for drones in various missions.

**Sample efficiency:** Enhancing sample efficiency in the learning process for drones is crucial, particularly in environments where data collection is hazardous or impractical. Even though this thesis provides adaptive embedded curriculum and multistage learning to improve sample efficiency, there are still challenges in fully exploiting the samples generated in simulation and ensuring the realism and diversity of these simulated environments. Developing algorithms that leverage transfer learning, few-shot learning, and continuous learning could provide significant strides in learning efficiently from limited datasets.

**Real world deployment:** Transitioning from controlled simulation environments to real-world deployment (Sim2Real) involves addressing unpredictability in environmental conditions, regulatory compliance, and adaptability to diverse operational contexts. Domain randomization was widely used in this thesis to address the Sim2Real issue in a certain way, but it is limited to predicted scenarios with known domain distributions. Hence, developing robust and adaptive algorithms capable of on-the-fly learning and decision-making, along with rigorous field testing under varied conditions, can be the future direction.

**Large vision-language model in open world** The proposed vision-based learning methods for drones in this thesis require explicit task descriptions and formal constraints, while in an open world, it is hard to provide all necessary formulations at the beginning to find the optimal solution. For instance, in a complex search and rescue mission, the drone swarm can only find the targets first and conduct rescue based on the information collected. In each stage, the task may change, and

there is no prior explicit problem at the start. Human interactions are necessary during this mission. With large vision-language models, the potential of drone autonomy can be greatly increased. Through interactions in the open world or few-shot imitation, vision-based learning can emerge with full autonomy for drone applications in the future.

**Safety and security** Ensuring the safety and security of drone operations is paramount, especially in densely populated or sensitive areas. This includes not only physical safety but also cyber security concerns. This thesis discussed multi-pursuit evasion, and cyber attack detection and isolation for controllers. However, higher protection strategies are expected, including the resilience of drones to adversarial attacks. Besides, adversarial attacks aimed at misleading vision-based systems and DRL algorithms should be further explored. Addressing these concerns requires a multifaceted approach. Improving the robustness of learning models against adversarial attacks and investigating the explainability of learning models for drones are imperative in future work.

# List of Author's Awards, Patents, and Publications[1]

## Awards

- **Gold Medal**, "MassPrint Technology, Leader of Mass Transfer Technology for Micro LED," in *7th China International College Students' 'Internet+' Innovation and Entrepreneurship Competition.*

## Patents

- M. Feroskhan, L. Chen, **J. Xiao**, Y. Zheng, N. A. Alagappan, "A novel design and controller of a coaxial drone," *NTU TD, Ref: 2024-017.*

## Journal Articles

- **J. Xiao**, P. Pisutsin, M. Feroskhan, "Toward Collaborative Multitarget Search and Navigation with Attention-Enhanced Local Observation," *Advanced Intelligent Systems*, 2024[pdf].
- **J. Xiao**, M. Feroskhan, "Learning Multi-Pursuit Evasion for Safe Targeted Navigation of Drones", *IEEE Transactions on Artificial Intelligence*, 2024 [pdf].
- **J. Xiao**, J. H. Chee, M. Feroskhan,"Real-time Multi-Drone Detection and Tracking for Pursuit Operations with Parameter Search and Edge Accelerator," *IEEE Transactions on Intelligent Vehicles*, 2024[pdf]

[1]The superscript † indicates co-first authors

- **J Xiao**, P. Pisutsin, M. Feroskhan, "Collaborative Target Search With a Visual Drone Swarm: An Adaptive Curriculum Embedded Multistage Reinforcement Learning Approach", *IEEE Transactions on Neural Networks and Learning Systems*, 2023[pdf]

- **J. Xiao**, M. Feroskhan, "Cyber attack detection and isolation for a quadrotor uav with modified sliding innovation sequences", IEEE Transactions on Vehicular Technology, 2022[pdf].

- L. Chen†, **J. Xiao**†, Y. Zheng, N A. Alagappan, M. Feroskhan, "Design, Modeling, and Control of a Coaxial Drone", *IEEE Transactions on Robotics*, 2024[pdf] (Co-first author).

- L. Chen†, **J. Xiao**†, C.T.W. Rui, M. Feroskhan, "Air-Ground Collaborative Control for Angle-Specified Heterogeneous Formations", *IEEE Transactions on Intelligent Vehicles*, 2024[pdf] (Co-first author).

- **J. Xiao**, R. Zhang, Y. Zhang, M. Feroskhan, "Vision-based Learning for Drones: A Survey", *IEEE Transactions on Neural Networks and Learning Systems*, 2023 [pdf] (Under Review).

- L. Chen, **J. Xiao**, R.C.H. Lin, M. Feroskhan, "Angle-Constrained Formation Maneuvering of Unmanned Aerial Vehicles", *IEEE Transactions on Control Systems Technology*, 2023[pdf]

## Conference Proceedings

- **J. Xiao**, Y.X.M. Tan, X. Zhou, M. Feroskhan, "Learning Collaborative Multi-Target Search for A Visual Drone Swarm", in *2023 IEEE Conference on Artificial Intelligence*, 2023[pdf].

# Bibliography


[1] James Davey. British retailer tesco to trial grocery home deliveries through drones, 2020. URL `https://acesse.dev/aVqMD`. Accessed: Feb. 21, 2024. xxi, 2

[2] Juliet Van Wagenen. Faa grants commercial uas exemptions for surveying, 2014. URL `https://www.aviationtoday.com/2014/12/10/faa-grants-commercial-uas-exemptions-for-surveying`. Accessed: Feb. 21, 2024. xxi, 2

[3] Mike Santora. Precision agriculture: the hype around drone technology, 2021. URL `https://www.bearingtips.com/precision-agriculture-the-hype-around-drone-technology`. Accessed: Feb. 21, 2024. xxi, 2

[4] Bob Trebilcock. Are distribution centers (dcs) ready for drones? *Modern Materials Handling*, 2021. URL `https://www.mmh.com/article/are_distribution_centers_dcs_ready_for_drones`. Accessed: Feb. 05, 2024. xxi, xxiii, 2, 88

[5] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40), 2020. ISSN 24709476. doi: 10.1126/scirobotics.aaz9712. xxi, 2, 5, 14, 29, 30, 31, 38, 40

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015. xxi, 21, 22

[7] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. xxi, 21, 22, 120

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. xxi, 21, 23

[9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE*

*conference on computer vision and pattern recognition*, pages 779–788, 2016. xxi, 23, 24

[10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. xxi, 23, 24, 120

[11] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022. xxi, 25

[12] Ye Zheng, Zhang Chen, Dailin Lv, Zhixing Li, Zhenzhong Lan, and Shiyu Zhao. Air-to-air visual detection of micro-uavs: An experimental evaluation of deep learning. *IEEE Robotics and Automation Letters*, 6(2):1020–1027, 2021. xxi, 26, 27

[13] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019. xxi, 29, 30, 31, 144, 146, 147

[14] Tao Huang, Shuanfeng Zhao, Longlong Geng, and Qian Xu. Unsupervised monocular depth estimation based on residual neural network of coarse–refined feature extractions for drone. *Electronics*, 8(10):1179, 2019. xxi, 31

[15] I Iswanto, A Ma'arif, O Wahyunggoro, and A Imam. Artificial potential field algorithm implementation for quadrotor path planning. *Int. J. Adv. Comput. Sci. Appl*, 10(8):575–585, 2019. xxi, 30, 32

[16] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021. xxi, xxii, 2, 3, 15, 16, 28, 31, 32, 33, 35, 36, 38, 40, 58, 144, 147

[17] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017. xxi, 15, 31, 33, 34, 56, 72, 77, 78

[18] Qiaoyun Wu, Xiaoxi Gong, Kai Xu, Dinesh Manocha, Jingxuan Dong, and Jun Wang. Towards target-driven visual navigation in indoor scenes via generative imitation learning. *IEEE Robotics and Automation Letters*, 6(1): 175–182, 2020. xxi, 15, 33, 34

[19] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *2013 IEEE international conference on robotics and automation*, pages 1765–1772. IEEE, 2013. xxii, 28, 36, 37, 38

[20] Moulay A Akhloufi, Sebastien Arola, and Alexandre Bonnet. Drones chasing drones: Reinforcement learning and deep search area proposal. *Drones*, 3(3): 58, 2019. xxii, 36, 37, 38

[21] Michele Mancini, Gabriele Costante, Paolo Valigi, and Thomas A Ciarfuglia. Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4296–4303. IEEE, 2016. xxii, 39

[22] Amarjot Singh, Devendra Patil, and SN Omkar. Eye in the sky: Real-time drone surveillance system (dss) for violent individuals identification using scatternet hybrid deep learning network. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1629–1637, 2018. xxii, 40, 41

[23] Linjie Xing, Xiaoyan Fan, Yaxin Dong, Zenghui Xiong, Lin Xing, Yang Yang, Haicheng Bai, and Chengjiang Zhou. Multi-uav cooperative system for search and rescue based on yolov5. *International Journal of Disaster Risk Reduction*, 76:102972, 2022. xxii, 3, 41, 44

[24] Bing Lu and Yuhong He. Species classification using unmanned aerial vehicle (uav)-acquired high spatial resolution imagery in a heterogeneous grassland. *ISPRS Journal of Photogrammetry and Remote Sensing*, 128:73–85, 2017. xxii, 40, 41, 42

[25] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023. xxii, 3, 15, 27, 38, 41, 43, 144

[26] Bosen Lin, Lizhen Wu, and Yifeng Niu. End-to-end vision-based cooperative target geo-localization for multiple micro uavs. *Journal of Intelligent & Robotic Systems*, 106(1):13, 2022. xxii, 3, 44

[27] Mark E Campbell and William W Whitacre. Cooperative tracking using vision measurements on seascan uavs. *IEEE Transactions on Control Systems Technology*, 15(4):613–626, 2007. xxii, 3, 44

[28] Jingjing Gu, Tao Su, Qiuhong Wang, Xiaojiang Du, and Mohsen Guizani. Multiple moving targets surveillance based on a cooperative network for multi-uav. *IEEE Communications Magazine*, 56(4):82–89, 2018. xxii, 3, 44, 46

[29] Yusen Cao, Fugui Qi, Yu Jing, Mingming Zhu, Tao Lei, Zhao Li, Juanjuan Xia, Jianqi Wang, and Guohua Lu. Mission chain driven unmanned aerial vehicle swarms cooperation for the search and rescue of outdoor injured human targets. *Drones*, 6(6):138, 2022. xxii, 3, 44

[30] Guanchong Niu, Lan Wu, Yunfan Gao, and Man-On Pun. Unmanned aerial vehicle (uav)-assisted path planning for unmanned ground vehicles (ugvs) via disciplined convex-concave programming. *IEEE Transactions on Vehicular Technology*, 71(7):6996–7007, 2022. xxii, 47, 48

[31] Guanchong Niu, Qingkai Yang, Yunfan Gao, and Man-On Pun. Vision-based autonomous landing for unmanned aerial and ground vehicles cooperative systems. *IEEE robotics and automation letters*, 7(3):6234–6241, 2021. xxii, 47, 48

[32] Ivan Kalinov, Alexander Petrovsky, Valeriy Ilin, Egor Pristanskiy, Mikhail Kurenkov, Vladimir Ramzhaev, Ildar Idrisov, and Dzmitry Tsetserukou. Warevision: Cnn barcode detection-based uav trajectory optimization for autonomous warehouse stocktaking. *IEEE Robotics and Automation Letters*, 5(4):6647–6653, 2020. xxii, 47, 48, 49

[33] Sara Minaeian, Jian Liu, and Young-Jun Son. Vision-based target detection and localization via a team of cooperative uav and ugvs. *IEEE Transactions on systems, man, and cybernetics: systems*, 46(7):1005–1016, 2015. xxii, 47, 48

[34] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Design Automation Conference*, pages 731–736. IEEE, 2010. 1, 174

[35] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011. 1, 174

[36] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, 2017. 1

[37] P. M. Kornatowski, M. Feroskhan, W. J. Stewart, and D. Floreano. Downside up:rethinking parcel position for aerial delivery. *IEEE Robotics and Automation Letters*, 5(3):4297–4304, 2020. doi: 10.1109/LRA.2020.2993768. 1, 174

[38] Przemyslaw Mariusz Kornatowski, Mir Feroskhan, William J Stewart, and Dario Floreano. A morphing cargo drone for safe flight in proximity of humans. *IEEE Robotics and Automation Letters*, 5(3):4233–4240, 2020. 1, 174

[39] Parham Nooralishahi, Clemente Ibarra-Castanedo, Shakeb Deane, Fernando López, Shashank Pant, Marc Genest, Nicolas P Avdelidis, and Xavier PV Maldague. Drone-based non-destructive inspection of industrial sites: A review and case studies. *Drones*, 5(4):106, 2021. 1

[40] Nikki J Stehr. Drones: The newest technology for precision agriculture. *Natural Sciences Education*, 44(1):89–91, 2015. 1

[41] Daniel Câmara. Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios. In *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*, pages 1–4. IEEE, 2014. 1

[42] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023. 1

[43] Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721, 2021. 1

[44] Boyu Zhou, Hao Xu, and Shaojie Shen. Racer: Rapid collaborative exploration with a decentralized multi-uav system. *IEEE Transactions on Robotics*, 39(3):1816–1835, 2023. doi: 10.1109/TRO.2023.3236945. 2

[45] Xin Zhou, Jiangchao Zhu, Hongyu Zhou, Chao Xu, and Fei Gao. Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 4101–4107. IEEE, 2021. 2, 5, 28, 29, 30, 144, 146, 147

[46] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. In *Conference on Robot Learning*, pages 133–145. PMLR, 2018. 2, 40

[47] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. ISSN 1552-3098. doi: 10.1109/tro.2018.2853729. 2

[48] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10651–10657. IEEE, 2020. 3, 5, 40

[49] Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. Ajna: Generalized deep uncertainty for minimal perception on parsimonious robots. *Science Robotics*, 8(81):eadd5139, 2023. 3

[50] Giuseppe Loianno, Justin Thomas, and Vijay Kumar. Cooperative localization and mapping of mavs using rgb-d sensors. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4021–4028. IEEE, 2015. 3, 44, 45

[51] Pengfei Tong, Xuerong Yang, Yajun Yang, Wei Liu, and Peiyi Wu. Multi-uav collaborative absolute vision positioning and navigation: A survey and discussion. *Drones*, 7(4):261, 2023.

[52] Nathan Piasco, Julien Marzat, and Martial Sanfourche. Collaborative localization and formation flying using distributed stereo-vision. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1202–1207. IEEE, 2016. 3, 44, 45

[53] Daqian Liu, Xiaomin Zhu, Weidong Bao, Bowen Fei, and Jianhong Wu. Smart: Vision-based method of cooperative surveillance and tracking by multiple uavs in the urban environment. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):24941–24956, 2022. 3, 44, 45

[54] Negar Farmani, Liang Sun, and Daniel J Pack. A scalable multitarget tracking system for cooperative unmanned aerial vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, 53(4):1947–1961, 2017. 3, 44, 45

[55] Mohammed Jouhari, Abdulla Khalid Al-Ali, Emna Baccour, Amr Mohamed, Aiman Erbad, Mohsen Guizani, and Mounir Hamdi. Distributed cnn inference on resource-constrained uavs for surveillance systems: Design and optimization. *IEEE Internet of Things Journal*, 9(2):1227–1242, 2021. 3, 44, 46

[56] Won Joon Yun, Soohyun Park, Joongheon Kim, MyungJae Shin, Soyi Jung, David A Mohaisen, and Jae-Hyun Kim. Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-uav control. *IEEE Transactions on Industrial Informatics*, 18(10):7086–7096, 2022. 3, 44

[57] Yazhe Tang, Yuchao Hu, Jinqiang Cui, Fang Liao, Mingjie Lao, Feng Lin, and Rodney SH Teo. Vision-aided multi-uav autonomous flocking in gps-denied environment. *IEEE Transactions on industrial electronics*, 66(1):616–626, 2018. 3, 44, 46

[58] Jürgen Scherer, Saeed Yahyanejad, Samira Hayat, Evsen Yanmaz, Torsten Andre, Asif Khan, Vladimir Vukadinovic, Christian Bettstetter, Hermann Hellwagner, and Bernhard Rinner. An autonomous multi-uav system for search and rescue. In *Proceedings of the first workshop on micro aerial vehicle networks, systems, and applications for civilian use*, pages 33–38, 2015. 3, 44

[59] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, et al. Swarm of micro flying robots in the wild. *Science Robotics*, 7(66):eabm5954, 2022. 3, 5, 89

[60] Lun Quan, Longji Yin, Tingrui Zhang, Mingyang Wang, Ruilin Wang, Sheng Zhong, Xin Zhou, Yanjun Cao, Chao Xu, and Fei Gao. Robust and efficient trajectory planning for formation flight in dense environments. *IEEE Transactions on Robotics*, 39(6):4785–4804, 2023. doi: 10.1109/TRO.2023.3301295. 3, 5

[61] Jean-Paul Yaacoub, Hassan Noura, Ola Salman, and Ali Chehab. Security analysis of drones systems: Attacks, limitations, and recommendations. *Internet of Things*, 11:100218, 2020. 4

[62] Moulay A Akhloufi, Sebastien Arola, and Alexandre Bonnet. Drones chasing drones: Reinforcement learning and deep search area proposal. *Drones*, 3(3): 58, 2019. 4

[63] Bernard Lagan. Bird attacks on drones force google to suspend home deliveries, 2021. URL `https://l1nq.com/MvlQi`. 4

[64] Fabian Schilling, Julien Lecoeur, Fabrizio Schiano, and Dario Floreano. Learning vision-based flight in drone swarms by imitation. *IEEE Robotics and Automation Letters*, 4(4):4523–4530, 2019. 5, 89

[65] Fabian Schilling, Fabrizio Schiano, and Dario Floreano. Vision-based drone flocking in outdoor environments. *IEEE Robotics and Automation Letters*, 6 (2):2954–2961, 2021. 5

[66] Stewart Geyer and Eric Johnson. 3d obstacle avoidance in adversarial environments for unmanned aerial vehicles. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6542, 2006. 5, 36, 38

[67] Linhai Xie, Sen Wang, Andrew Markham, and Niki Trigoni. Towards monocular vision based obstacle avoidance through deep reinforcement learning. *arXiv preprint arXiv:1706.09829*, 2017. 5, 28

[68] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M Shamim Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2020. 5, 176

[69] F. Farivar, M. S. Haghighi, A. Jolfaei, and M. Alazab. Artificial intelligence for detection, estimation, and compensation of malicious attacks in nonlinear cyber-physical systems and industrial iot. *IEEE Transactions on Industrial Informatics*, 16(4):2716–2725, 2020. doi: 10.1109/TII.2019.2956474. 5, 176

[70] Dan Ye and Tian-Yu Zhang. Summation detector for false data-injection attack in cyber-physical systems. *IEEE Transactions on Cybernetics*, 50(6): 2338–2345, 2020. ISSN 2168-2267. doi: 10.1109/tcyb.2019.2915124. 6, 8, 175, 176, 177, 194

[71] Kebina Manandhar, Xiaojun Cao, Fei Hu, and Yao Liu. Detection of faults and attacks including false data injection attack in smart grid using kalman filter. *IEEE transactions on control of network systems*, 1(4):370–379, 2014. 6, 176, 189

[72] 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. 6, 57, 66

[73] Chenxi Xiao, Peng Lu, and Qizhi He. Flying through a narrow gap using end-to-end deep reinforcement learning augmented with curriculum learning and sim2real. *IEEE Transactions on Neural Networks and Learning Systems*, 34(5):2701–2708, 2023. doi: 10.1109/TNNLS.2021.3107742. 15, 36, 58, 144, 147

[74] Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021. 6, 57, 58, 66, 91, 95

[75] Andrew Cohen, Ervin Teng, Vincent-Pierre Berges, Ruo-Ping Dong, Hunter Henry, Marwan Mattar, Alexander Zook, and Sujoy Ganguly. On the use and misuse of abosrbing states in multi-agent reinforcement learning. *RL in Games Workshop AAAI 2022*, 2022. 7, 89, 92, 93, 98, 104

[76] Nan Jiang, Kuiran Wang, Xiaoke Peng, Xuehui Yu, Qiang Wang, Junliang Xing, Guorong Li, Guodong Guo, Qixiang Ye, Jianbin Jiao, Jian Zhao, and Zhenjun Han. Anti-uav: A large-scale benchmark for vision-based uav tracking. *IEEE Transactions on Multimedia*, 25:486–500, 2023. doi: 10.1109/TMM.2021.3128047. 7, 121, 122, 127

[77] Jie Zhao, Jingshu Zhang, Dongdong Li, and Dong Wang. Vision-based anti-uav detection and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):25323–25334, 2022. doi: 10.1109/TITS.2022.3177627. 7, 122, 138

[78] Kenneth O Stanley. Why open-endedness matters. *Artificial life*, 25(3):232–235, 2019. 8, 145

[79] Charles A Holt and Alvin E Roth. The nash equilibrium: A perspective. *Proceedings of the National Academy of Sciences*, 101(12):3999–4002, 2004. 8, 145

[80] Zhengen Zhao, Yimin Huang, Ziyang Zhen, and Yuzhe Li. Data-driven false data-injection attack design and detection in cyber-physical systems. *IEEE Transactions on Cybernetics*, pages 1–9, 2020. ISSN 2168-2267. doi: 10.1109/tcyb.2020.2969320. 8, 175, 176

[81] Yuanqiang Zhou, Kyriakos G. Vamvoudakis, Wassim M. Haddad, and Zhong-Ping Jiang. A secure control learning framework for cyber-physical systems under sensor and actuator attacks. *IEEE Transactions on Cybernetics*, pages 1–13, 2020. ISSN 2168-2267. doi: 10.1109/tcyb.2020.3006871. 8, 175, 176, 177

[82] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011. 12

[83] Nathan Michael, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, et al. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, 2012. 14

[84] Hongcan Guan, Xiliang Sun, Yanjun Su, Tianyu Hu, Haitao Wang, Heping Wang, Chigang Peng, and Qinghua Guo. UAV-lidar aids automatic intelligent powerline inspection. *International Journal of Electrical Power and Energy Systems*, 130:106987, sep 2021. ISSN 01420615. doi: 10.1016/j.ijepes.2021.106987. 14

[85] Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, and Michele Grassi. Uncooperative pose estimation with a lidar-based system. *Acta Astronautica*, 110:287–297, 2015. 14

[86] Zejie Wang, Zhen Zhao, Zhao Jin, Zhengping Che, Jian Tang, Chaomin Shen, and Yaxin Peng. Multi-stage fusion for multi-class 3d lidar detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3120–3128, 2021. 14

[87] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5:1–26, 2016. 15, 28

[88] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2502–2509. IEEE, 2018.

[89] Davide Scaramuzza and Zichao Zhang. *Aerial Robots, Visual-Inertial Odometry of*, pages 1–9. Springer Berlin Heidelberg, Berlin, Heidelberg, 2020. ISBN 978-3-642-41610-1. doi: 10.1007/978-3-642-41610-1_71-1. URL `https://doi.org/10.1007/978-3-642-41610-1_71-1`. 15, 28, 181

[90] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 15, 28

[91] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, aug 2018. ISSN 15523098. doi: 10.1109/TRO.2018.2853729.

[92] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José M M Montiel, and Juan D Tardós. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. 2021. doi: 10.1109/TRO.2021.3075644. 15, 28

[93] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020. 15

[94] Wenliang Gao, Kaixuan Wang, Wenchao Ding, Fei Gao, Tong Qin, and Shaojie Shen. Autonomous aerial robot using dual-fisheye cameras. *Journal of Field Robotics*, 37(4):497–514, 2020. 15

[95] Varun Ravi Kumar, Senthil Yogamani, Hazem Rashed, Ganesh Sitsu, Christian Witt, Isabelle Leang, Stefan Milz, and Patrick Mäder. Omnidet: Surround view cameras based multi-task visual perception network for autonomous driving. *IEEE Robotics and Automation Letters*, 6(2):2830–2837, 2021. 15, 36, 38

[96] A. Dominik Haumann, Kim D. Listmann, and Volker Willert. DisCoverage: A new paradigm for multi-robot exploration. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 929–934, 2010. ISBN 9781424450381. doi: 10.1109/ROBOT.2010.5509993. 15, 55

[97] Aaron Hao Tan, Federico Pizarro Bejarano, Yuhan Zhu, Richard Ren, and Goldie Nejat. Deep reinforcement learning for decentralized multi-robot exploration with macro actions. *IEEE Robotics and Automation Letters*, 8(1): 272–279, 2022. 15

[98] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017. 15

[99] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint*, 2019. 15

[100] Mudassar Liaq and Yungcheol Byun. Autonomous uav navigation using reinforcement learning. *International Journal of Machine Learning and Computing*, 9(6), 2019. 15

[101] Chunxue Wu, Bobo Ju, Yan Wu, Xiao Lin, Naixue Xiong, Guangquan Xu, Hongyan Li, and Xuefeng Liang. Uav autonomous target search based on deep reinforcement learning in complex disaster scene. *IEEE Access*, 7: 117227–117245, 2019. 58

[102] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1205–1212. IEEE, 2021. 15

[103] Jiaping Xiao, Phumrapee Pisutsin, and Mir Feroskhan. Collaborative target search with a visual drone swarm: An adaptive curriculum embedded multistage reinforcement learning approach. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2023. doi: 10.1109/TNNLS.2023.3331370. 16, 38, 89, 100, 118, 144

[104] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 16, 17, 18, 19, 34

[105] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017. 17

[106] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999. 17, 19

[107] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999. 19

[108] Yu-han Chang, Tracey Ho, and Leslie Kaelbling. All learning is local: Multi-agent learning in global reward games. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. 18

[109] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *Proceedings of the AAAI conference on artificial intelligence*, 32(1), 2018. 19, 35, 91, 93

[110] Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2019. 19, 57, 58, 59, 65, 69, 72, 77, 78, 84

[111] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 20, 34, 69, 77, 78, 91

[112] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019. doi: 10.1109/TNNLS.2018.2876865. 20

[113] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023. 20

[114] Hamid Laga, Laurent Valentin Jospin, Farid Boussaid, and Mohammed Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):1738–1764, 2020. 20

[115] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022. doi: 10.1109/TNNLS.2021.3084827. 20

[116] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 21, 120

[117] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 21, 25, 120, 125, 132, 133

[118] Yang Liu, Yao Zhang, Yixin Wang, Feng Hou, Jin Yuan, Jiang Tian, Yang Zhang, Zhongchao Shi, Jianping Fan, and Zhiqiang He. A survey of visual transformers. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2023. doi: 10.1109/TNNLS.2022.3227717. 21

[119] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 23

[120] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision*, pages 280–296. Springer, 2022. 25

[121] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *The Eleventh International Conference on Learning Representations*, 2022. 25

[122] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 25, 92, 94, 101

[123] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021. 26

[124] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *European Conference on Computer Vision*, pages 3–20. Springer, 2022. 26

[125] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 27

[126] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 27

[127] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 27

[128] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 27

[129] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7363–7372, 2019. 27

[130] Daniel Tan Wei Xun, Yoke Lin Lim, and Sutthiphong Srigrarom. Drone detection using yolov3 with transfer learning on nvidia jetson tx2. In *2021 Second International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, pages 1–6, 2021. doi: 10.1109/ICA-SYMP50206.2021.9358449. 27, 121, 122

[131] Kartik Mohta, Michael Watterson, Yash Mulgaonkar, Sikang Liu, Chao Qu, Anurag Makineni, Kelsey Saulnier, Ke Sun, Alex Zhu, Jeffrey Delmerico, et al. Fast, autonomous flight in gps-denied and cluttered environments. *Journal of Field Robotics*, 35(1):101–120, 2018. 29, 38

[132] Fei Gao, William Wu, Jie Pan, Boyu Zhou, and Shaojie Shen. Optimal Time Allocation for Quadrotor Trajectory Generation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4715–4722. Institute of Electrical and Electronics Engineers Inc., dec 2018. ISBN 9781538680940. doi: 10.1109/IROS.2018.8593579. 38

[133] Fei Gao, Luqi Wang, Boyu Zhou, Xin Zhou, Jie Pan, and Shaojie Shen. Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments. *IEEE Transactions on Robotics*, 36(5):1526–1545, 2020. 38

[134] Fei Gao, William Wu, Wenliang Gao, and Shaojie Shen. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *Journal of Field Robotics*, 36(4):710–733, jun 2019. ISSN 15564967. doi: 10.1002/rob.21842.

[135] Fei Gao, Luqi Wang, Boyu Zhou, Luxin Han, Jie Pan, and Shaojie Shen. Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments, may 2019. ISSN 1552-3098.

[136] Boyu Zhou, Jie Pan, Fei Gao, and Shaojie Shen. Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, 2021. 30, 38, 146, 147

[137] Lun Quan, Zhiwei Zhang, Xingguang Zhong, Chao Xu, and Fei Gao. Eva-planner: Environmental adaptive quadrotor planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 398–404. IEEE, 2021. 38, 146

[138] Yuhang Zhang, Qing Yu, Kin Huat Low, and Chen Lv. A self-supervised monocular depth estimation approach based on uav aerial images. In *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, pages 1–8. IEEE, 2022. 29

[139] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008. 29

[140] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. doi: 10.1007/s10514-012-9321-0. URL `https://octomap.github.io`. Software available at `https://octomap.github.io`. 29

[141] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 30, 146

[142] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136. 30, 146

[143] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000. 30

[144] Federico Augugliaro, Angela P Schoellig, and Raffaello D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex

programming approach. In *2012 IEEE/RSJ international conference on Intelligent Robots and Systems*, pages 1917–1922. IEEE, 2012. 30, 146

[145] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505. IEEE, 1985. 31

[146] Xi Dai, Yuxin Mao, Tianpeng Huang, Na Qin, Deqing Huang, and Yanan Li. Automatic obstacle avoidance of quadrotor uav via cnn-based learning. *Neurocomputing*, 402:346–358, 2020. 31, 38, 147

[147] Malik Aqeel Anwar and Arijit Raychowdhury. Navren-rl: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images. In *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 1–6. IEEE, 2018. 32, 38, 147

[148] Yuhang Zhang, Kin Huat Low, and Chen Lyu. Partially-observable monocular autonomous navigation for uav through deep reinforcement learning. In *AIAA AVIATION 2023 Forum*, page 3813, 2023. 31

[149] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 33

[150] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 33

[151] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015. ISSN 14764687. doi: 10.1038/nature14236. 33, 65

[152] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992. 35

[153] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. 35

[154] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. *Advances in Neural Information Processing Systems*, 33:12402–12413, 2020. 35

[155] 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. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020. 35, 91, 92

[156] Jiaping Xiao, Yi Xuan Marcus Tan, Xinliang Zhou, and Mir Feroskhan. Learning collaborative multi-target search for a visual drone swarm. In *2023 IEEE Conference on Artificial Intelligence (CAI)*, pages 5–7, 2023. doi: 10.1109/CAI54212.2023.00012. 36, 144

[157] 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. 36

[158] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018. 36, 69, 72, 73, 161

[159] Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, and Alejandro Hernandez Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016. 36

[160] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016. 36

[161] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[162] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[163] 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 *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017. 36, 70

[164] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. 36

[165] Fabian Schilling, Julien Lecoeur, Fabrizio Schiano, and Dario Floreano. Learning vision-based flight in drone swarms by imitation. *IEEE Robotics and Automation Letters*, 4(4):4523–4530, 2019. doi: 10.1109/LRA.2019.2935377. 38

[166] Yuhan Xie, Minghao Lu, Rui Peng, and Peng Lu. Learning agile flights through narrow gaps with varying angles using onboard sensing. *IEEE Robotics and Automation Letters*, 8(9):5424–5431, 2023. doi: 10.1109/LRA.2023.3295655. 38

[167] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. 39

[168] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 39

[169] Yuxuan Liu, Lujia Wang, and Ming Liu. Yolostereo3d: A step back to 2d for efficient stereo 3d detection. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13018–13024. IEEE, 2021. 39

[170] Wei Li, Hongliang Li, Qingbo Wu, Xiaoyu Chen, and King Ngi Ngan. Simultaneously detecting and counting dense vehicles from drone images. *IEEE Transactions on Industrial Electronics*, 66(12):9651–9662, 2019. 40

[171] Hailing Zhou, Hui Kong, Lei Wei, Douglas Creighton, and Saeid Nahavandi. On detecting road regions in a single uav image. *IEEE transactions on intelligent transportation systems*, 18(7):1713–1722, 2016.

[172] Naser Hossein Motlagh, Miloud Bagaa, and Tarik Taleb. Uav-based iot platform: A crowd surveillance use case. *IEEE Communications Magazine*, 55 (2):128–134, 2017. 40, 41

[173] Michael A Goodrich, Bryan S Morse, Damon Gerhardt, Joseph L Cooper, Morgan Quigley, Julie A Adams, and Curtis Humphrey. Supporting wilderness search and rescue using a camera-equipped mini uav. *Journal of Field Robotics*, 25(1-2):89–110, 2008. 40, 42

[174] Eleftherios Lygouras, Nicholas Santavas, Anastasios Taitzoglou, Konstantinos Tarchanidis, Athanasios Mitropoulos, and Antonios Gasteratos. Unsupervised human detection with an embedded vision system on a fully autonomous uav for search and rescue operations. *Sensors*, 19(16):3542, 2019.

[175] Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *IEEE robotics & automation magazine*, 19(3):46–56, 2012. 40

[176] J Senthilnath, Manasa Kandukuri, Akanksha Dokania, and KN Ramesh. Application of uav imaging platform for vegetation analysis based on spectral-spatial methods. *Computers and Electronics in Agriculture*, 140:8–24, 2017. 40, 42

[177] Mohammad R Khosravi and Sadegh Samadi. Bl-alm: A blind scalable edge-guided reconstruction filter for smart environmental monitoring through green iomt-uav networks. *IEEE Transactions on Green Communications and Networking*, 5(2):727–736, 2021.

[178] Cenk Donmez, Osman Villi, Suha Berberoglu, and Ahmet Cilek. Computer vision-based citrus tree detection in a cultivated environment using uav imagery. *Computers and Electronics in Agriculture*, 187:106273, 2021. 40

[179] Daeho Kim, Meiyin Liu, SangHyun Lee, and Vineet R Kamat. Remote proximity monitoring between mobile construction resources using camera-mounted uavs. *Automation in Construction*, 99:168–182, 2019. 40, 42

[180] Tung Khuc, Tuan Anh Nguyen, Hieu Dao, and F Necati Catbas. Swaying displacement measurement for structural monitoring using computer vision and an unmanned aerial vehicle. *Measurement*, 159:107769, 2020. 43

[181] Billie F Spencer Jr, Vedhus Hoskere, and Yasutaka Narazaki. Advances in computer vision-based civil infrastructure inspection and monitoring. *Engineering*, 5(2):199–222, 2019. 40

[182] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido CHE de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone. *Journal of Field Robotics*, 37(4):667–692, 2020. 40, 43

[183] Matthias Muller, Guohao Li, Vincent Casser, Neil Smith, Dominik L Michels, and Bernard Ghanem. Learning a controller fusion network by online trajectory filtering for vision-based uav racing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 43

[184] Matthias Muller, Vincent Casser, Neil Smith, Dominik L Michels, and Bernard Ghanem. Teaching uavs to race: End-to-end regression of agile controls in simulation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 40

[185] Robert Penicka and Davide Scaramuzza. Minimum-time quadrotor waypoint flight in cluttered environments. *IEEE Robotics and Automation Letters*, 7 (2):5719–5726, 2022. 40

[186] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast:

Optimal methods meet learning for drone racing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696. IEEE, 2019. 40, 144

[187] Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52 (2):1–31, 2019. 47

[188] Daqian Liu, Weidong Bao, Xiaomin Zhu, Bowen Fei, Zhenliang Xiao, and Tong Men. Vision-aware air-ground cooperative target localization for uav and ugv. *Aerospace Science and Technology*, 124:107525, 2022. 47

[189] Jianqiang Li, Genqiang Deng, Chengwen Luo, Qiuzhen Lin, Qiao Yan, and Zhong Ming. A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems. *IEEE Transactions on Vehicular Technology*, 65(12):9585–9596, 2016.

[190] Lele Zhang, Feng Gao, Fang Deng, Lele Xi, and Jie Chen. Distributed estimation of a layered architecture for collaborative air–ground target geolocation in outdoor environments. *IEEE Transactions on Industrial Electronics*, 70 (3):2822–2832, 2022. 47

[191] Zhe-Cheng Xu, Bin-Bin Hu, Bin Liu, XD Wang, and Hai-Tao Zhang. Vision-based autonomous landing of unmanned aerial vehicle on a motional unmanned surface vessel. In *2020 39th Chinese Control Conference (CCC)*, pages 6845–6850. IEEE, 2020. 47, 48

[192] Cheng Hui, Chen Yousheng, Li Xiaokun, and Wong Wing Shing. Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision. In *Proceedings of the 32nd Chinese control conference*, pages 5895–5901. IEEE, 2013. 47

[193] An Adaptable. Building an aerial–ground robotics system for precision farming. *IEEE ROBOTICS & AUTOMATION MAGAZINE*, 2021. 47, 49

[194] Ebtehal Turki Alotaibi, Shahad Saleh Alqefari, and Anis Koubaa. Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7: 55817–55832, 2019. 54

[195] Zhe Chen, Javier Alonso-Mora, Xiaoshan Bai, Daniel D Harabor, and Peter J Stuckey. Integrated task assignment and path planning for capacitated multi-agent pickup and delivery. *IEEE Robotics and Automation Letters*, 6(3): 5816–5823, 2021. 54

[196] Liangming Chen, Kun Cao, Lihua Xie, Xiaolei Li, and Mir Feroskhan. 3-d network localization using angle measurements and reduced communication. *IEEE Transactions on Signal Processing*, 70:2402–2415, 2022. 54

[197] Liangming Chen, Jiaping Xiao, Reuben Chua Hong Lin, and Mir Feroskhan. Angle-constrained formation maneuvering of unmanned aerial vehicles. *IEEE Transactions on Control Systems Technology*, 31(4):1733–1746, 2023. doi: 10.1109/TCST.2023.3240286. 54, 141

[198] J Spaethe, L Chittka, and P Skorupski. Visual search and decision making in bees: time, speed and accuracy. *International Journal of Comparative Psychology*, 2006. 54

[199] Jorge Pena 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. 55

[200] Enrica Soria, Fabrizio Schiano, and Dario Floreano. Predictive control of aerial swarms in cluttered environments. *Nature Machine Intelligence*, 3(6): 545–554, 2021. 55

[201] Takahiro Miki, Marija Popović, Abel Gawel, Gregory Hitz, and Roland Siegwart. Multi-agent time-based decision-making for the search and action problem. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2365–2372, 2018. doi: 10.1109/ICRA.2018.8460996. 56

[202] Ayan Dutta, Anirban Ghosh, and O. Patrick Kreidl. Multi-robot informative path planning with continuous connectivity constraints. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3245–3251, 2019. doi: 10.1109/ICRA.2019.8794090. 55

[203] Xiang Cao, Daqi Zhu, and Simon X Yang. Multi-auv target search based on bioinspired neurodynamics model in 3-d underwater environments. *IEEE transactions on neural networks and learning systems*, 27(11):2364–2374, 2015. 55

[204] Wanqing Zhao, Qinggang Meng, and Paul W. H. Chung. A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario. *IEEE Transactions on Cybernetics*, 46(4):902–915, 2016. doi: 10.1109/TCYB.2015.2418052.

[205] Samira Hayat, Evşen Yanmaz, Timothy X Brown, and Christian Bettstetter. Multi-objective uav path planning for search and rescue. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 5569–5574. IEEE, 2017. 55

[206] Yu-Jun Zheng, Yi-Chen Du, Hai-Feng Ling, Wei-Guo Sheng, and Sheng-Yong Chen. Evolutionary collaborative human-uav search for escaped criminals. *IEEE Transactions on Evolutionary Computation*, 24(2):217–231, 2019. 55, 69

[207] Bardienus P Duisterhof, Shushuai Li, Javier Burgués, Vijay Janapa Reddi, and Guido CHE de Croon. Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9099–9106. IEEE, 2021. 55

[208] Wu Deng, Junjie Xu, Xiao-Zhi Gao, and Huimin Zhao. An enhanced msiqde algorithm with novel multiple strategies for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(3):1578–1587, 2022. doi: 10.1109/TSMC.2020.3030792.

[209] Wu Deng, Junjie Xu, Huimin Zhao, and Yingjie Song. A novel gate resource allocation method using improved pso-based qea. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1737–1745, 2022. doi: 10.1109/TITS.2020.3025796. 55

[210] Tianze Luo, Budhitama Subagdja, Di Wang, and Ah-Hwee Tan. Multi-agent collaborative exploration through graph-based deep reinforcement learning. In *2019 IEEE International Conference on Agents (ICA)*, pages 2–7, 2019. doi: 10.1109/AGENTS.2019.8929168. 55, 84

[211] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):14413–14423, 2020. doi: 10.1109/TVT.2020.3034800. 55, 84

[212] Xiangyu Liu and Ying Tan. Feudal latent space exploration for coordinated multi-agent reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–9, 2022. doi: 10.1109/TNNLS.2022.3146201. 55

[213] Qiaoyun Wu, Jun Wang, Jing Liang, Xiaoxi Gong, and Dinesh Manocha. Image-goal navigation in complex environments via modular learning. *IEEE Robotics and Automation Letters*, 7(3):6902–6909, 2022. doi: 10.1109/LRA.2022.3178810. 56, 84

[214] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. *arXiv preprint arXiv:2103.08624*, 2021. 58, 144, 162

[215] Zhixi Shen, Lian Tan, Shuangshuang Yu, and Yongduan Song. Fault-tolerant adaptive learning control for quadrotor uavs with the time-varying cog and full-state constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(12):5610–5622, 2021. doi: 10.1109/TNNLS.2021.3071094.

[216] Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022. 58, 144

[217] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019. 58, 149

[218] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019. 89, 91

[219] Yuxuan Liu, Hu Liu, Yongliang Tian, and Cong Sun. Reinforcement learning based two-level control framework of uav swarm for cooperative persistent surveillance in an unknown urban area. *Aerospace Science and Technology*, 98:105671, 2020. 59

[220] Weinan Gao, Mohammed Mynuddin, Donald C. Wunsch, and Zhong-Ping Jiang. Reinforcement learning-based cooperative optimal output regulation via distributed adaptive internal model. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5229–5240, 2022. doi: 10.1109/TNNLS.2021.3069728. 58

[221] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 63, 73

[222] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018. 65

[223] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 69

[224] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 1146–1155. PMLR, 2017. 69

[225] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022. 73, 89, 91

[226] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011. 80

[227] Minnan Luo, Xiaojun Chang, Liqiang Nie, Yi Yang, Alexander G. Hauptmann, and Qinghua Zheng. An adaptive semisupervised feature analysis for video semantic recognition. *IEEE Transactions on Cybernetics*, 48(2): 648–660, 2018. doi: 10.1109/TCYB.2017.2647904. 84

[228] Jiaping Xiao and Mir Feroskhan. Cyber attack detection and isolation for a quadrotor uav with modified sliding innovation sequences. *IEEE Transactions on Vehicular Technology*, 71(7):7202–7214, 2022. doi: 10.1109/TVT.2022.3170725. 84

[229] Dalin Zhang, Lina Yao, Kaixuan Chen, Sen Wang, Xiaojun Chang, and Yunhao Liu. Making sense of spatio-temporal preserving representations for eeg-based human intention recognition. *IEEE Transactions on Cybernetics*, 50(7):3033–3044, 2020. doi: 10.1109/TCYB.2019.2905157. 85

[230] Kaixuan Chen, Lina Yao, Dalin Zhang, Xianzhi Wang, Xiaojun Chang, and Feiping Nie. A semisupervised recurrent convolutional attention model for human activity recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 31(5):1747–1756, 2020. doi: 10.1109/TNNLS.2019.2927224. 85

[231] Amedeo Fabris, Emanuele Aucone, and Stefano Mintchev. Crash 2 squash: An autonomous drone for the traversal of narrow passageways. *Advanced Intelligent Systems*, 4(11):2200113, 2022. 88

[232] Malika Meghjani, Sandeep Manjanna, and Gregory Dudek. Multi-target rendezvous search. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2596–2603. IEEE, 2016. 88

[233] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593, 2018. doi: 10.1109/ACCESS.2018.2831228. 88

[234] Gábor Vásárhelyi, Csaba Virágh, Gergő Somorjai, Tamás Nepusz, Agoston E Eiben, and Tamás Vicsek. Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20):eaat3536, 2018. 88

[235] Enrica Soria, Fabrizio Schiano, and Dario Floreano. Predictive control of aerial swarms in cluttered environments. *Nature Machine Intelligence*, 3(6): 545–554, 2021. 89

[236] Jiahu Qin, Qichao Ma, Yang Shi, and Long Wang. Recent advances in consensus of multi-agent systems: A brief survey. *IEEE Transactions on Industrial Electronics*, 64(6):4972–4983, 2016. 89

[237] Zhe Liu, Qiming Liu, Ling Tang, Kefan Jin, Hongye Wang, Ming Liu, and Hesheng Wang. Visuomotor reinforcement learning for multirobot cooperative navigation. *IEEE Transactions on Automation Science and Engineering*, 19(4):3234–3245, 2021. 89

[238] Jinpeng Li, Zhiyuan Su, and Yiyu Qiu. Dynamic motion planning model for multirobot using graph neural network and historical information. *Advanced Intelligent Systems*, page 2300036, 2023. 89, 91, 95

[239] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020. 90, 98, 104

[240] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010. 90

[241] Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35:16509–16521, 2022.

[242] Chuming Li, Jie Liu, Yinmin Zhang, Yuhong Wei, Yazhe Niu, Yaodong Yang, Yu Liu, and Wanli Ouyang. Ace: Cooperative multi-agent q-learning with bidirectional action-dependency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8536–8544, 2023. 90

[243] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998. 90

[244] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017. 91

[245] Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th international conference on autonomous agents and multiagent systems*, pages 225–232. ACM, 2011. 91

[246] Chao Yan, Xiaojia Xiang, Chang Wang, Fei Li, Xiangke Wang, Xin Xu, and Lincheng Shen. Pascal: Population-specific curriculum-based madrl for collision-free flocking with large-scale fixed-wing uav swarms. *Aerospace Science and Technology*, page 108091, 2023. 94

[247] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: a short survey. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4819–4825, 2021. 97

[248] Li Wang, Yupeng Zhang, Yujing Hu, Weixun Wang, Chongjie Zhang, Yang Gao, Jianye Hao, Tangjie Lv, and Changjie Fan. Individual reward assisted multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 23417–23432. PMLR, 2022. 99

[249] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2020. URL `https://arxiv.org/pdf/1809.02627.pdf`. 102

[250] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, 7:48572–48634, 2019. 118

[251] Dong Yin, Xuan Yang, Huangchao Yu, Siyuan Chen, and Chang Wang. An air-to-ground relay communication planning method for uavs swarm applications. *IEEE Transactions on Intelligent Vehicles*, 8(4):2983–2997, 2023. doi: 10.1109/TIV.2023.3237329. 118

[252] Alexander Von Moll, David Casbeer, Eloy Garcia, Dejan Milutinović, and Meir Pachter. The multi-pursuer single-evader game: A geometric approach. *Journal of Intelligent & Robotic Systems*, 96:193–207, 2019. 118

[253] Jiaping Xiao and Mir Feroskhan. Ams-drl: Learning multi-pursuit evasion for safe targeted navigation of drones. *arXiv preprint arXiv:2304.03443*, 2023. 118

[254] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 118

[255] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *European Conference on Computer Vision*, pages 107–122. Springer, 2020. 121

[256] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8844–8854, 2022. 118

[257] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022. 118

[258] Sushil Pratap Bharati, Yuanwei Wu, Yao Sui, Curtis Padgett, and Guanghui Wang. Real-time obstacle detection and tracking for sense-and-avoid mechanism in uavs. *IEEE Transactions on Intelligent Vehicles*, 3(2):185–197, 2018. doi: 10.1109/TIV.2018.2804166. 118

[259] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012. 119, 129

[260] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013. 119

[261] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022. 120

[262] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 120

[263] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 121, 129, 134

[264] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015. 121

[265] Maciej Pawełczyk and Marek Wojtyra. Real world object detection dataset for quadcopter unmanned aerial vehicle detection. *IEEE Access*, 8:174394–174409, 2020. 121, 122, 138

[266] Ye Zheng, Zhang Chen, Dailin Lv, Zhixing Li, Zhenzhong Lan, and Shiyu Zhao. Air-to-air visual detection of micro-uavs: An experimental evaluation of deep learning. *IEEE Robotics and Automation Letters*, 6(2):1020–1027, 2021. doi: 10.1109/LRA.2021.3056059. 121, 122, 127, 138

[267] Yuliang Zhao, Zhongjie Ju, Tianang Sun, Fanghecong Dong, Jian Li, Ruige Yang, Qiang Fu, Chao Lian, and Peng Shan. Tgc-yolov5: An enhanced yolov5 drone detection model based on transformer, gam & ca attention mechanism. *Drones*, 7(7):446, 2023. 121, 122

[268] T. Lin. Labelimg github repository. `https://github.com/tzutalin/labelImg`, 2015. 125

[269] Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. Rtmdet: An empirical study of designing real-time object detectors. *arXiv preprint arXiv:2212.07784*, 2022. 125, 132, 133

[270] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 125

[271] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 131

[272] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017. doi: 10.1109/ICIP.2017.8296962. 131

[273] Yonatan Tariku Tesfaye, Eyasu Zemene, Andrea Prati, Marcello Pelillo, and Mubarak Shah. Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. *arXiv preprint arXiv:1706.06196*, 2017. 131

[274] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 132

[275] Glenn Jocher, Alex Stoken, Jirka Borovec, Liu Changyu, Adam Hogan, Laurentiu Diaconu, Jake Poznanski, Lijun Yu, Prashant Rai, Russ Ferriday, et al. ultralytics/yolov5: v3. 0. *Zenodo*, 2020. 133

[276] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. URL `https://github.com/ultralytics/ultralytics`. 133

[277] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40): eaaz9712, 2020. 144, 146, 147, 162, 167

[278] Minghao Lu, Han Chen, and Peng Lu. Perception and avoidance of multiple small fast moving objects for quadrotors with only low-cost rgbd camera. *IEEE Robotics and Automation Letters*, 7(4):11657–11664, 2022. doi: 10.1109/LRA.2022.3205114. 144

[279] Han Chen and Peng Lu. Real-time identification and avoidance of simultaneous static and dynamic obstacles on point cloud for uavs navigation. *Robotics and Autonomous Systems*, 154:104124, 2022. 144

[280] Xu Fang, Chen Wang, Lihua Xie, and Jie Chen. Cooperative pursuit with multi-pursuer and one faster free-moving evader. *IEEE Transactions on Cybernetics*, 52(3):1405–1414, 2022. doi: 10.1109/TCYB.2019.2958548. 144, 148

[281] Bogdan Vlahov, Eric Squires, Laura Strickland, and Charles Pippin. On developing a uav pursuit-evasion policy using reinforcement learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 859–864. IEEE, 2018. 148

[282] Yuanda Wang, Lu Dong, and Changyin Sun. Cooperative control for multi-player pursuit-evasion games with reinforcement learning. *Neurocomputing*, 412:101–114, 2020. 148

[283] Cristino de Souza, Rhys Newbury, Akansel Cosgun, Pedro Castillo, Boris Vidolov, and Dana Kulić. Decentralized multi-agent pursuit using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4552–4559, 2021. doi: 10.1109/LRA.2021.3068952. 148

[284] Ruilong Zhang, Qun Zong, Xiuyun Zhang, Liqian Dou, and Bailing Tian. Game of drones: Multi-uav pursuit-evasion game with online motion planning by deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):7900–7909, 2023. doi: 10.1109/TNNLS.2022.3146976. 144, 148, 162, 163, 165

[285] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019. 144

[286] Jiang Zhao, Yibo Wang, Zhihao Cai, Ningjun Liu, Kun Wu, and Yingxun Wang. Learning visual representation for autonomous drone navigation via a contrastive world model. *IEEE Transactions on Artificial Intelligence*, pages 1–14, 2023. doi: 10.1109/TAI.2023.3283488. 144

[287] Guohua Wu, Mingfeng Fan, Jianmai Shi, and Yanghe Feng. Reinforcement learning based truck-and-drone coordinated delivery. *IEEE Transactions on Artificial Intelligence*, 4(4):754–763, 2023. doi: 10.1109/TAI.2021.3087666. 144

[288] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011. 144

[289] Yong-bo Chen, Guan-chen Luo, Yue-song Mei, Jian-qiao Yu, and Xiao-long Su. Uav path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 47(6):1407–1420, 2016. 147

[290] Alex M. Andrew. Reinforcement Learning: An Introduction, 1998. ISSN 0368492X. 147

[291] Rufus Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999. 148

[292] Jaime F Fisac and S Shankar Sastry. The pursuit-evasion-defense differential game in dynamic constrained environments. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 4549–4556. IEEE, 2015. 148

[293] Zhiqiang Pu, Huimu Wang, Zhen Liu, Jianqiang Yi, and Shiguang Wu. Attention enhanced reinforcement learning for multi agent cooperation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8235–8249, 2023. doi: 10.1109/TNNLS.2022.3146858. 148

[294] Nikolaos-Marios T. Kokolakis and Kyriakos G. Vamvoudakis. Safety-aware pursuit-evasion games in unknown environments using gaussian processes and finite-time convergent reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2022. doi: 10.1109/TNNLS.2022.3203977. 148

[295] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *International conference on machine learning*, pages 805–813. PMLR, 2015. 149

[296] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016. 149

[297] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. 149

[298] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 149

[299] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2019. 149

[300] Drew Fudenberg and Jean Tirole. *Game theory*. MIT press, 1991. 151, 152

[301] Howie Choset. Robotic motion planning: Potential functions. *Robotics Institute, Carnegie Mellon University*, 2010. 164

[302] R. Poovendran. Cyber–physical systems: Close encounters between two parallel worlds [point of view]. *Proceedings of the IEEE*, 98(8):1363–1366, 2010. doi: 10.1109/JPROC.2010.2050377. 174

[303] Edward Ashford Lee and Sanjit A Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016. 174

[304] Bas Vergouw, Huub Nagel, Geert Bondt, and Bart Custers. Drone technology: Types, payloads, applications, frequency spectrum issues and future developments. In *The future of drone use*, pages 21–45. Springer, 2016. 174

[305] C. Kwon and I. Hwang. Reachability analysis for safety assurance of cyber-physical systems against cyber attacks. *IEEE Transactions on Automatic Control*, 63(7):2272–2279, 2018. doi: 10.1109/TAC.2017.2761762. 174

[306] S. M. Giray. Anatomy of unmanned aerial vehicle hijacking with signal spoofing. In *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*, pages 795–800, 2013. doi: 10.1109/RAST.2013.6581320.

[307] K. Hartmann and K. Giles. Uav exploitation: A new domain for cyber power. In *2016 8th International Conference on Cyber Conflict (CyCon)*, pages 205–221, 2016. doi: 10.1109/CYCON.2016.7529436.

[308] Hong-Jun Ma, Yanli Liu, Tianbo Li, and Guang-Hong Yang. Nonlinear high-gain observer-based diagnosis and compensation for actuator and sensor faults in a quadrotor unmanned aerial vehicle. *IEEE Transactions on Industrial Informatics*, 15(1):550–562, 2018.

[309] Karanjot Singh, Mohit Mehndiratta, and Mir Feroskhan. Quadplus: Design, modeling, and receding horizon-based control of a hyperdynamic quadrotor. *IEEE Transactions on Aerospace and Electronic Systems*, pages 1–1, 2021. doi: 10.1109/TAES.2021.3133314. 174

[310] S. Sun, G. Cioffi, C. de Visser, and D. Scaramuzza. Autonomous quadrotor flight despite rotor failure with onboard vision sensors: Frames vs. events. *IEEE Robotics and Automation Letters*, 6(2):580–587, 2021. doi: 10.1109/LRA.2020.3048875. 174

[311] Noah Shachtman. Computer virus hits us. drone fleet. *CNN. com*, 2011. 174

[312] K. Hartmann and C. Steup. The vulnerability of uavs to cyber attacks - an approach to the risk assessment. In *2013 5th International Conference on Cyber Conflict (CYCON 2013)*, pages 1–23, 2013. 174

[313] Fred Samland, Jana Fruth, Mario Hildebrandt, Tobias Hoppe, and Jana Dittmann. Ar. drone: security threat analysis and exemplary attack to track persons. In *Intelligent Robots and Computer Vision XXIX: Algorithms and Techniques*, volume 8301, page 83010G. International Society for Optics and Photonics, 2012. 174

[314] Vishal Dey, Vikramkumar Pudi, Anupam Chattopadhyay, and Yuval Elovici. Security vulnerabilities of unmanned aerial vehicles and countermeasures: An experimental study. In *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, pages 398–403. IEEE, 2018. 174

[315] M. Hooper, Y. Tian, R. Zhou, B. Cao, A. P. Lauf, L. Watkins, W. H. Robinson, and W. Alexis. Securing commercial wifi-based uavs from common security attacks. In *MILCOM 2016 - 2016 IEEE Military Communications Conference*, pages 1213–1218, 2016. doi: 10.1109/MILCOM.2016.7795496. 174

[316] Johann-Sebastian Pleban, Ricardo Band, and Reiner Creutzburg. Hacking and securing the ar. drone 2.0 quadcopter: investigations for improving the

security of a toy. In *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014*, volume 9030, page 90300L. International Society for Optics and Photonics, 2014.

[317] Y. Kwon, J. Yu, B. Cho, Y. Eun, and K. Park. Empirical analysis of mavlink protocol vulnerability for attacking unmanned aerial vehicles. *IEEE Access*, 6:43203–43212, 2018. doi: 10.1109/ACCESS.2018.2863237.

[318] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan. Survey on uav cellular communications: Practical aspects, standardization advancements, regulation, and security challenges. *IEEE Communications Surveys Tutorials*, 21(4):3417–3442, 2019. doi: 10.1109/COMST.2019.2906228. 174

[319] Mark A Gregory and David Glance. Cyber crime, cyber security and cyber warfare. In *Security and the Networked Society*, pages 51–95. Springer, 2013. 174

[320] Jayna Locke. Can Drones Be Hacked, Tracked, and Used to Carry Passengers?, 2021. URL `https://www.digi.com/blog/post/can-drones-be-hacked-tracked-and-carry-passengers`. 174

[321] Alireza Abbaspour, Kang K Yen, Parisa Forouzannezhad, and Arman Sargolzaei. A neural adaptive approach for active fault-tolerant control design in uav. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50 (9):3401–3411, 2018. 176

[322] Hyunseong Lee, Guoyi Li, Ashwin Rai, and Aditi Chattopadhyay. Anomaly detection of aircraft system using kernel-based learning algorithm. In *AIAA Scitech 2019 Forum*, page 1224, 2019. 176

[323] A. Ayad, H. E. Z. Farag, A. Youssef, and E. F. El-Saadany. Detection of false data injection attacks in smart grids using recurrent neural networks. In *2018 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, 2018. doi: 10.1109/ISGT.2018.8403355. 176

[324] Alireza Abbaspour, Kang K Yen, Shirin Noei, and Arman Sargolzaei. Detection of fault data injection attack on uav using adaptive neural network. *Procedia Computer Science*, 95:193–200, 2016. 176

[325] Liang Xiao, Caixia Xie, Minghui Min, and Weihua Zhuang. User-centric view of unmanned aerial vehicle transmission against smart attacks. *IEEE Transactions on Vehicular Technology*, 67(4):3420–3430, 2017. 176

[326] L. Xiao, X. Lu, D. Xu, Y. Tang, L. Wang, and W. Zhuang. Uav relay in vanets against smart jamming with reinforcement learning. *IEEE Transactions on Vehicular Technology*, 67(5):4087–4097, 2018. doi: 10.1109/TVT.2018.2789466. 176

[327] Eman Mousavinejad, Fuwen Yang, Qing-Long Han, and Ljubo Vlacic. A novel cyber attack detection method in networked control systems. *IEEE Transactions on Cybernetics*, 48(11):3254–3264, 2018. ISSN 2168-2267. doi: 10.1109/tcyb.2018.2843358. 176

[328] D. B. Rawat and C. Bajracharya. Detection of false data injection attacks in smart grid communication systems. *IEEE Signal Processing Letters*, 22(10): 1652–1656, 2015. doi: 10.1109/LSP.2015.2421935. 176

[329] Riccardo MG Ferrari and André MH Teixeira. Detection and isolation of replay attacks through sensor watermarking. *IFAC-PapersOnLine*, 50(1): 7363–7368, 2017. 177

[330] Xinyu Wang, Xiaoyuan Luo, Yuyan Zhang, and Xinping Guan. Detection and isolation of false data injection attacks in smart grids via nonlinear interval observer. *IEEE Internet of Things Journal*, 6(4):6498–6512, 2019.

[331] X. Wang, X. Luo, M. Zhang, Z. Jiang, and X. Guan. Detection and isolation of false data injection attacks in smart grid via unknown input interval observer. *IEEE Internet of Things Journal*, 7(4):3214–3229, 2020. doi: 10.1109/JIOT.2020.2966221. 177

[332] M. W. Mueller, M. Hehn, and R. D'Andrea. A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Transactions on Robotics*, 31(6):1294–1310, 2015. doi: 10.1109/TRO.2015.2479878. 178

[333] Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007. 180

[334] A. P. Schoellig, C. Wiltsche, and R. D'Andrea. Feed-forward parameter identification for precise periodic quadrocopter motions. In *2012 American Control Conference (ACC)*, pages 4313–4318, 2012. doi: 10.1109/ACC.2012.6315248. 184, 195

[335] M. L. Psiaki and T. E. Humphreys. Gnss spoofing and detection. *Proceedings of the IEEE*, 104(6):1258–1270, 2016. doi: 10.1109/JPROC.2016.2526658. 186

[336] Yilin Mo and Bruno Sinopoli. False data injection attacks in control systems. In *Preprints of the 1st workshop on Secure Control Systems*, pages 1–6, 2010. 186

[337] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023. 209