TU Delft

Delft University of Technology

Active Classification of Moving Targets With Learned Control Policies

Serra-Gómez, Álvaro; Montijano, Eduardo; Böhmer, Wendelin ; Alonso-Mora, Javier

DOI
10.1109/LRA.2023.3271508

Publication date
2023

Document Version
Final published version

Published in
IEEE Robotics and Automation Letters

Citation (APA)
Serra-Gómez, Á., Montijano, E., Böhmer, W., & Alonso-Mora, J. (2023). Active Classification of Moving Targets With Learned Control Policies. *IEEE Robotics and Automation Letters*, *8*(6), 3717-3724. https://doi.org/10.1109/LRA.2023.3271508

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

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Active Classification of Moving Targets With Learned Control Policies

Álvaro Serra-Gómez, Eduardo Montijano, *Member, IEEE*, Wendelin Böhmer, and Javier Alonso-Mora, *Senior Member, IEEE*

***Abstract*—In this paper, we consider the problem where a drone has to collect semantic information to classify multiple moving targets. In particular, we address the challenge of computing control inputs that move the drone to informative viewpoints, position and orientation, when the information is extracted using a "black-box" classifier, e.g., a deep learning neural network. These algorithms typically lack of analytical relationships between the viewpoints and their associated outputs, preventing their use in information-gathering schemes. To fill this gap, we propose a novel attention-based architecture, trained via Reinforcement Learning (RL), that outputs the next viewpoint for the drone favoring the acquisition of evidence from as many unclassified targets as possible while reasoning about their movement, orientation, and occlusions. Then, we use a low-level MPC controller to move the drone to the desired viewpoint taking into account its actual dynamics. We show that our approach not only outperforms a variety of baselines but also generalizes to scenarios unseen during training. Additionally, we show that the network scales to large numbers of targets and generalizes well to different movement dynamics of the targets.**

***Index Terms*—Machine learning for robot control, reactive and sensor-based planning, surveillance robotic systems.**

I. Introduction

In SURVEILLANCE and tracking applications an autonomous drone may be tasked with collecting relevant information from multiple targets, e.g., recognize people with blue eyes. Recent deep learning approaches show excellent results at detecting and categorizing single and multiple elements with images [1] or LiDAR [2]. However, these methods are generally not enough for active classification with a mobile drone, which also requires planning of the drone's movement and reasoning over the targets' future behavior.

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

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

Moreover, to simplify the training, the policy is abstracted from the low-level dynamics of the drone, which are instead considered inside a low-level MPC controller at test time. Finally, the estimations are updated with an efficient information fusion method, conflation, suited to be used with black-box perception algorithms. A full overview of the method is shown in Fig. 1. Experiments under different conditions show that our approach outperforms a variety of baselines and is robust to scenarios unseen during training.

Manuscript received 15 November 2022; accepted 10 April 2023. Date of publication 28 April 2023; date of current version 9 May 2023. This letter was recommended for publication by Associate Editor J.R. Martinez-de Dios and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported in part by the U.S. Office of Naval Research Global under Grant N62909-19-1-2027 and in part by MCIN/AEI/10.13039/501100011033 and ERDF under Grant PID2021-125514NB-I00. A way of making Europe. *(Corresponding author: Álvaro Serra-Gómez.)*

Álvaro Serra-Gómez and Javier Alonso-Mora are with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: a.serragomez@tudelft.nl; j.alonsomora@tudelft.nl).

Eduardo Montijano is with the Departamento de Informática e Ingeniería de Sistemas and I3A, Universidad de Zaragoza, 50018 Zaragoza, Spain (e-mail: emonti@unizar.es).

Wendelin Böhmer is with the Department of Software Technology, Delft University of Technology, 2600 GA Delft, The Netherlands (e-mail: j.w.bohmer@tudelft.nl).

Code available at https://github.com/tud-amr/AC-LCP.git.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3271508, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3271508

II. Related Works

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

A. Multi-View Active Classification

The problem of active classification is typically solved by pre-defining a set of viewpoints through which trajectories are planned to gather sufficient information to solve an active classification task. One-step greedy planners for active object classification [3] select object-dependent viewpoints based on class uncertainty and observation occlusions. Some non-myopic methods [4] account for the cost of movement and information

2377-3766 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.

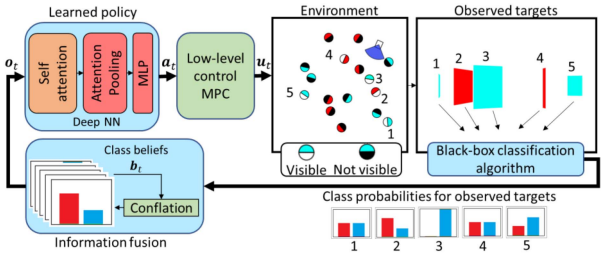


Fig. 1. Overview of our method for active classification. The objective is to classify all targets into different classes, red and blue in the example. The method has three parts. First, observable targets, targets 1 to 5, represented with white back in the figure, are detected by the onboard sensor.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

gain between viewpoints to solve the problem of active classification. Others, e.g., [5], formulate the problem as a partially observable Markov Decision Process (POMDP) and plans a path over viewpoints while accounting for measurement costs, occlusions and possible misclassifications. Similarly, [6] computes plans using a variation of Monte Carlo tree search. However, these methods generally require access to a model to estimate viewpoint usefulness a priori. Learning methods are non-myopic and allow to use black-box models. Recent works leverage the use of DRL for static multi-target pose estimation and active classification by learning a policy that moves the camera towards viewpoints that reduce observation uncertainty [7] or maximize information gain to enhance object classification [8]. Nevertheless, these methods assume that targets are static, which makes them unsuitable for active classification with moving targets. Although some works consider dynamic targets, they are often limited to a pre-defined closed environment [9] or, as its the case for aerial videography methods [10], [11], focus on target information visibility which requires prior knowledge on where this information is visible from.

B. Learning for Motion Planning in Dynamic Environments

Learning approaches in motion planning tasks have the potential to encode and identify patterns in complex motion planning tasks. Recent advances in Deep Representation Learning show promise in learning latent representations of the state space, capturing the underlying structure and symmetries in dynamic environments [12]. Recent works in learning-based motion-planning policies rely on Convolutional Neural Networks (CNN) [13], [14] for visual-based navigation, or encode the state of multiple static/dynamic elements in the environment using Graph Convolutional Networks (GCN) [15], [16] or Long-Short Term Memory layers (LSTMs) [17], [18].

Yet these strategies require a priori knowledge on the priority order of the elements in the encoded sequence, or the structure of the encoded graph [19]. Instead, DeepSets [20] and self-attention based architectures [21], [22] are permutation invariant and enable encoding element sets without making further assumptions on their structure. Most related to our approach, [23] uses a combination of Self-attention [22] and DeepSets [20] to learn a policy that coordinates a multi-robot system to track a set of dynamic targets. However, DeepSets aggregate all targets' information by assigning equal weights to them. In contrast, we leverage the use of self-attention [22] and attention-based set-function approximators [24] to effectively encode the dynamic environment, learning the importance of each target in our future decision.

III. Problem Formulation

Consider a drone with state $\mathbf{x}_t$ at discrete time t, control inputs $\mathbf{u}_t$ and dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, obtained for a sampling time of τ_l seconds. The drone has to collect information from a set $\mathcal{J} = \{1, \ldots, M\}$ of M dynamic targets, where $\mathbf{y}_t^j$ is the state of target j at time step t, represented in the drone's reference frame and $\mathcal{Y}_t = \{\mathbf{y}_t^j\}_{j=1,\ldots,M}$. The targets follow their own dynamics, $\mathbf{y}_{t+1}^j = g^j(\mathbf{y}_t^j)$, which are unknown to the drone. We assume that the drone flies above the targets, neglecting physical collisions with them. Nevertheless, the targets can still collide with each other and, more importantly, they can occlude the visibility of others to the drone, depending on where they are. Besides, we do not deal with how to measure and track the targets' relative information, assuming they are provided by some external perception algorithm, e.g., [25]. For the sake of simplicity, in the following we omit the t subscript, except when needed.

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

To compute the beliefs, every $\tau_h \gg \tau_l$ seconds, the drone is able to make an observation and use a black-box perception algorithm (e.g., a pre-trained CNN classifier) to compute a probability distribution over the class set for each target. We denote that distribution as $\mathcal{P} = h(\mathcal{Y}) = \{\mathbf{p}^j\}_{j=1,\ldots,M}$, where $\mathbf{p}^j = (p^{j1}, \ldots, p^{jC})$ is the probability vector for target j, and p^{jc} is the probability of target j belonging to class c. For unobserved targets, $\mathbf{p}^j$ is a uniform distribution. As it happens with the majority of real CNN classifiers, we assume the drone has no available model to map how the relative positions relate to the observed probability distributions. Beliefs are then computed by fusing these measurements, $\mathbf{b}_t^j = \zeta(\mathbf{p}_{1:t}^j)$, where ζ is the conflation operator [26], a function that models how the beliefs can be computed from the history of classification probabilities given by the black-box sensor (see Section IV-A).

Under these conditions, the problem considered in the paper is to actuate the drone in such a way that it is able to classify all targets as quickly as possible, i.e., make l_t^j true for all j in the minimum possible value of t. To address it, we formulate a sequential decision-making problem that the drone solves at every time step t. The objective of this problem is to find the actions over a time horizon T that minimize the accumulated

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.

entropy of all the targets' beliefs,

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

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

IV. Methodology

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

A. Target Class Observations and Belief Updates

An important aspect to consider is how to aggregate the different observations made by the drone about each target's class to produce the class beliefs. The first issue to consider is that standard Bayesian recursive estimation is not advisable because the measurement likelihood model for the update, $\mathbb{P}(\mathbf{p}_t^j|\mathbf{b}_{t-1})$, is not available from a black-box sensor. To train and learn an accurate pose-dependent model of the likelihood, a dense dataset must be built first. Then, to use it for optimal viewpoint search all targets and their occlusions must be considered. This process is costly and scales badly. Instead, in this paper we propose to use a mathematical method called conflation [26]. Conflation is used to aggregate probability distributions obtained from measurements over the same phenomena under different circumstances. Notably, this technique has the property of minimizing the loss of Shannon information when flattening multiple independent probability distributions into a single one, that is, computing $\mathbf{b}_t^j$ given the measurements $\mathbf{p}_{0:t}^j$. In addition, it is a commutative and associative operator, enabling easy and efficient recursive computation and making it appealing for onboard computation. The conflation is defined by

$$\mathbf{b}_t^j = \zeta(\mathbf{p}_{1:t}^j) \equiv \zeta(\mathbf{b}_{t-1}^j, \mathbf{p}_t^j) = \frac{\mathbf{b}_{t-1}^j \odot \mathbf{p}_t^j}{(\mathbf{b}_{t-1}^j)^\top \mathbf{p}_t^j}, \tag{2}$$

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

Lastly, although (2) considers all the measurements equally, it can easily be extended to a weighted version using the weights as powers of the probability distributions. This could be useful for example in cases where the black-box also outputs a confidence measurement over the class probabilities.

B. Viewpoint Control Policy

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

Finally, the reward function, R, is shaped based on the problem described in (1), and additional factors that the policy should take into account. First of all, there is a dense reward, $R_{\mathcal{H}}$, proportional to how much the entropy of each belief, $\mathbf{b}_t^j$, has decreased each time step, t, due to the new information gathered. Additionally, there is one sparse reward, denoted by R_l, for individual target classifications, when any l_t^j changes from zero to one at time t, and another, $R_{\mathcal{J}}$, for completing the classification of all of the existing targets, when $\sum_j l_t^j = M$. On the other hand, to promote solving the task quickly and efficiently, the reward includes a constant penalty associated to the time required to classify the targets, R_t, and another one proportional to the distance to the recommended viewpoint, R_a, to favor small motions. The formal definition of all the reward terms is

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

$$\text{where} \quad R_{\mathcal{H}}(\mathbf{s}_t, \mathbf{s}_{t+1}) = w_{\mathcal{H}} \sum_{j=1}^{M} \left(\mathcal{H}(\mathbf{b}_t^j) - \mathcal{H}(\mathbf{b}_{t+1}^j)\right),$$

$$R_l(\mathbf{s}_t, \mathbf{s}_{t+1}) = w_l \sum_{j=1}^{M} (l_{t+1}^j - l_t^j),$$

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.

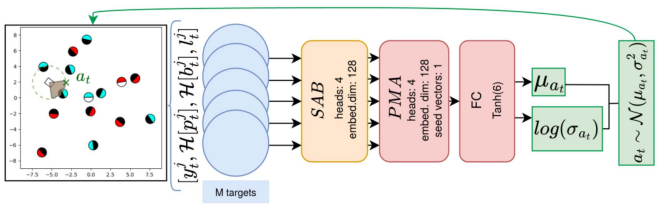


Fig. 2. Policy neural network architecture. The sequence of target information is fed to the self-attention block (SAB) which encodes and identifies how each target pose affects the visibility of the others from the drone's perspective. This information is fed to the pooling multi-head attention layer (PMA) and mapped through a fully connected layer (FC) to obtain the policy viewpoint recommendation.

$$R_{\mathcal{J}}(\mathbf{s}_{t+1}) = \begin{cases} w_{\mathcal{J}} & \text{if } \sum_{j=1}^{M} l_{t+1}^j = M \\ 0 & \text{otherwise} \end{cases},$$
$$R_t = w_t,$$
$$R_a(\mathbf{a}_t) = w_a(\|\Delta \mathbf{q}_t\| + |\Delta \psi_t|), \tag{3}$$

with $w_{\mathcal{H}}$, w_l, $w_{\mathcal{J}}$, w_t and w_a weights that scale each term.

2) Architecture: The ability of any learned policy $\pi_\phi(\mathbf{a}_t|\mathbf{o}_t)$ to generalize beyond the exact situations seen during training, e.g., more targets or changing target behaviors, depends crucially on the chosen neural architecture, shown in Fig. 2. We arrange the information available of each target, $\mathbf{o}_t^j$, into a set $\mathbf{o}_t$. The main challenge arises from the set being large and changing over time. Inspired by Relational Graph Convolutional Networks [27] and self-attention mechanisms [22] used on static knowledge graphs, we implement a self-attention block (SAB) to identify the relationship between the poses of all targets, i.e., at time t. Thus, the first layer is,

$$\tilde{\mathbf{e}}_i^{1,h} = F(\mathbf{o}_t^i; \mathbf{W}_{q,h}^1) + \sum_{j \in \mathcal{J}} \lambda_{i,j}^h F(\mathbf{o}_t^j; \mathbf{W}_{v,h}^1),$$
$$\mathbf{e}_i^1 = LN\left(Res^1\left(LN\left(concat\left(\left\{\tilde{\mathbf{e}}_i^{1,h}\right\}_{h=1\dots H}\right)\right)\right)\right),$$
$$\lambda_{i,j}^h = \text{softmax}\left(\frac{1}{\sqrt{d_h}} F(\mathbf{o}_t^i; \mathbf{W}_{q,h}^1)^\top F(\mathbf{o}_t; \mathbf{W}_{k,h}^1)\right)_j, \tag{4}$$

where $i \in \mathcal{J}$, $Res^l(x) = x + \sigma(F(x; \mathbf{W}^l))$, with σ being a ReLU activation function and F a parametric affine transformation. LN stands for Layer Normalization. $\mathbf{W}^I \in \mathbb{R}^{d_{enc} \times (d_h H+1)}$ and $\mathbf{W}_{w,h}^1 \in \mathbb{R}^{d_h \times (d_{in}+1)}, w \in \{v, q, k\}$, are learnable parameters. d_{in}, d_h, d_{enc} are the dimensionality of the input, each head h, and the first layer. Note that each head h encodes a different relation $\boldsymbol{\lambda}^h$ between targets. The purpose of this first layer is to encode information such as target visibility, perspective from which each target's information can be observed, occlusions, and possible simultaneous observations.

Next, we draw inspiration from Set Tranformers [24], used in static set-structured data, to aggregate the features of all latent target representations. We employ a pooling multi-head attention mechanism (PMA) where a learned seed vector per head $\mathbf{v}_s^h \in \mathbb{R}^{d_h}$ is employed to compute the attention weights for a single query,

$$\tilde{\mathbf{e}}^{2,h} = \mathbf{v}_s^h + \sum_{j \in \mathcal{J}} \lambda_j^h F(\mathbf{e}_j^1; \mathbf{W}_{v,h}^2),$$
$$\mathbf{e}^2 = LN\left(Res^2\left(LN\left(concat\left(\left\{\tilde{\mathbf{e}}^{2,h}\right\}_{h=1\dots H}\right)\right)\right)\right),$$
$$\lambda_j^h = \text{softmax}\left(\left\{\frac{1}{\sqrt{d_h}} \mathbf{v}_s^{h,\top} F(\mathbf{e}_j^1; \mathbf{W}_{k,h}^2)\right\}_{j \in \mathcal{J}}\right)_j. \tag{5}$$

This results in a latent vector $\mathbf{e}^2$ that is mapped, through a fully connected layer, to the parameters $\mu_{\mathbf{a}_t}$ and $\log(\sigma_{\mathbf{a}_t})$ of a diagonal Gaussian distribution $\mathcal{N}(\mu_{\mathbf{a}_t}, \sigma_{\mathbf{a}_t})$ over viewpoints. The learned policy π_ϕ samples recommended viewpoints $\mathbf{a}_t$ from this distribution. We use the Proximal Policy Optimization (PPO) algorithm to train the network [28], [29]. As learning algorithms like PPO also require an estimate of the state-value $V^{\pi_\phi}(\mathbf{s}_t) = \mathbb{E}[\sum_{t'=t}^{\infty} \gamma^{t'-t} R(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{t'} \sim \mathcal{T}, \mathbf{a}_{t'} \sim \pi_\phi]$, where γ is the discount factor, another linear layer predicts $V^{\pi_\phi}(\mathbf{s}_t) \approx \mathbf{v}_v^\top \mathbf{e}^2$. The latter is only used during training to guide the policy. We combine both the surrogate loss and KL-divergence term to stabilize training. We also use an entropy regularization term to encourage exploration [30]. We refer the reader to [28] for more information on the algorithm equations and details.

C. Low-Level Controller

To account for the drone dynamics, the recommended viewpoint position $\mathbf{a}_t$ is tracked with an MPC low-level controller [31]. Let $\mathbf{x}_{at}$ denote the viewpoint state output by the policy in the world frame, obtained from the current drone state and setting to zero the information that is not considered in $\mathbf{a}_t$, e.g., roll and pitch. Every τ_l seconds, the controller solves the following receding horizon constrained optimization problem,

$$\min_{\mathbf{x}_{1:N}, \mathbf{u}_{0:N-1}} \sum_{k=0}^{N-1} J^k(\mathbf{x}_k, \mathbf{u}_k) + J^N(\mathbf{x}_N, \mathbf{x}_{at})$$
$$\text{s.t.} \quad \mathbf{x}_0 = \mathbf{x}_t, \quad \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$$
$$\mathbf{u}_k \in \mathcal{U}, \quad 0 \le k \le N-1 \tag{6}$$

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

$$J^N(\mathbf{x}_N, \mathbf{x}_{at}) = w_g \frac{\|\mathbf{x}_N - \mathbf{x}_{at}\|}{\|\mathbf{x}^0 - \mathbf{x}_{at}\|}, \tag{7}$$

the stage and terminal costs, weighted by w_u and w_g respectively. For more details we refer the reader to [31].

V. Implementation Details

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

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.

TABLE I
HYPERPARAMETERS FOR PPO TRAINING ALGORITHM

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

A. Simulated Perception Environment

1) Observation Model: We consider a synthetic pinhole camera with focal length equal to 400 pixels that acquires images of 640×480 pixels. A target is modeled as visible if we can draw a line between its center and the drone's without any collision. For each target we consider only the half part of the cylinder that is facing forward and project it into the image frame, if it is visible from the camera perspective. This generates an image of a trapezoid with area and skew depending on the relative position and orientation of the target (Fig. 1). We use these two parameters to determine the probability distribution of the observation over the class set, decreasing the probability of the true class exponentially with the skew and increasing it linearly with the area. We also penalize heavily trapezoids that do not fit in the image. Our method does not have any knowledge of neither the observation model nor the classification model, which makes them black-box to it without loss of generality.

2) Training Conditions: Our high-level policy and the learned baselines are trained in closed environments of $50 \times 50\,\mathrm{m}^2$ with simulation steps of $\tau_h = 0.25\,s$. As shown in Fig. 1, targets are modeled as cylinders of $0.6\,m$ radius, 1.8 m of height, with their class information only visible from the front. The targets follow constant velocity dynamics and belong to either class *red* or class *blue*. Targets' speed in every axis is sampled around 1 m/s and clipped at 1.5 m/s. Whenever targets collide among themselves or against walls, they rebound conserving kinetic energy and momentum.

The drone's maximum angular and linear speed in each axis is respectively $\dot{\psi}^{max} = 60°/s$ and 2 m/s. This is to ensure that the drone can reach targets moving away from it. To reduce computation costs, only during training we assume that the drone follows first-order dynamics, and control its velocity to guide the drone to the recommended viewpoint. Each episode, each method is given 100 seconds to classify all targets ($l_t^j = 1$, $\forall j \in \mathcal{J}$, $b_{\max} = 0.95$). Episodes are finished after reaching the timeout, or successfully classifying all targets. Values for the reward weights are $w_{\mathcal{J}} = 100$, $w_l = 5$, $w_{\mathcal{H}} = 1$, $w_t = 0.3$, $w_a = 0.01$.

B. Training Algorithm

The learning algorithm and the training of our policy are implemented using the RLlib framework [29]. Table I shows the hyperparameters tuned for training the PPO algorithm. We tuned these hyperparameters because they regulate the speed and quality of training and are more problem-specific. Other hyperparameters follow the default values provided by the framework. We refer the reader to [28], [32], [33] for a detailed analysis on the effect of these hyperparameters on the training algorithm and how to tune them. State space exploration is enhanced [34] to make the learned policy more robust at test time, by randomizing the drone's initial poses, and the targets' initial poses, velocities and beliefs over their classes. Some of the targets are randomly selected and enforced to remain static.

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

The policy and value function (detailed in section IV-B) were trained for $9.6e7$ steps in environments containing up to twelve targets, sampled uniformly (first phase). Then, they were further trained for $3.2e7$ steps in environments containing a random number between one to six targets (second phase). The training of the algorithm was done using an Intel i9-9900 CPU@3.10 GHz computer.

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

VI. SIMULATED RESULTS

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

A. Baselines

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

- *Hand-crafted*: Hard-coded policy that guides the drone to a position 2 m in front of an unobserved target, pointing the camera at it.
- *Single-target (Ours)*: This is an ablation of our method. A learned single-target policy is given the information of an unobserved target and guides the drone to viewpoints allowing it to classify it as fast as possible. It is trained in scenarios with just one target.
- *LSTM encoder [17], [18]*: The self-attention and attention pooling layers are substituted by a linear layer followed by an LSTM. Every unclassified target is inversely sorted by proximity to the drone. The first layer encodes each target's information, and the sequence is fed into the LSTM.
- *DeepSets decoder [23]*: We replace the attention pooling layer by a mean pooling layer.

The latter two baselines are recently proposed scalable architectures employed in different, albeit similar [23], problems.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.

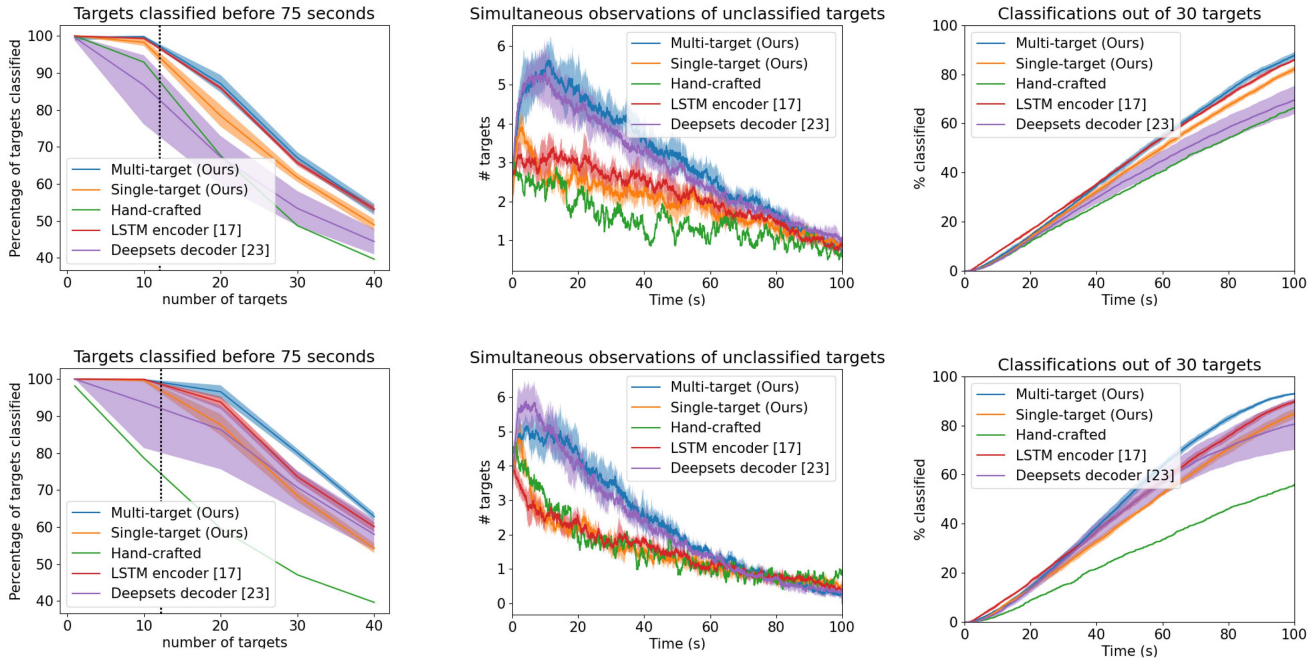


Fig. 3. **Top row)** Experiments with targets following constant velocity dynamics, as in training. **Bottom row)** Experiments with targets following social forces dynamics not seen during training. **Left)** Comparison of the percentage of targets classified before timeout in environments with 1 to 40 targets. The black line denotes the maximum amount of targets seen during training. **Center)** Evolution of simultaneous observations along the episode in environments of 30 targets. **Right)** Classification speed in environments of 30 targets.

B. Test Conditions

At test time, the whole pipeline described in section IV is used, including the drone dynamics and controller described in section IV-C. We evaluate our method's robustness under target behavior both seen and unseen during training. Aside from constant velocity dynamics, we test our policy in environments where targets follow social forces pedestrian dynamics [35]. As in training, each method is given up to 100 seconds to classify all targets. Each policy is trained with 5 different seeds. The results of all seeds are averaged and shown with their standard deviations. For every test, we evaluate and average each method's results over 50 episodes. Initial conditions of all evaluation episodes are randomised but maintained equal across our method and all baselines.

Since the first two methods are designed to classify one target, the extension to multiple targets is done through their sequential classification. Unclassified targets are ordered by their distance to the drone. The drone classifies the first in the list before moving on to the next one. Observations of other targets are still accounted for in the information fusion and belief computation.

C. Scalability Analysis

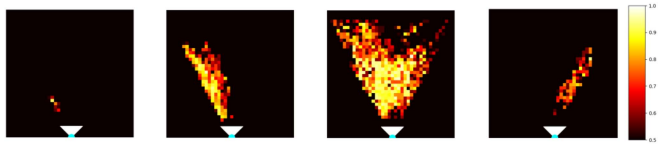
We evaluate each method in environments containing up to 40 targets to test their scalability to larger number of targets than seen during training. In Fig. 3-**Left**, we report each method's percentage of classified targets at the 75 s mark in environments containing different numbers of targets. In general, as expected, there is a drop in the percentage of targets that can be classified when their quantity increases. This is due to the task theoretically requiring more steps and an increase of occlusions generated by other targets. The results show that our method is the one able to generalize best to target dynamics unseen during training and generally outperforms all other baselines in environments containing much larger amounts of targets than seen during training. While our method does not take any assumption on how target information should be weighted, [17] relies heavily on the priority order given to targets, as analysed in [18]. This is the reason why, in our experiments, [17] shows similar scalability results under target dynamics seen during training (Fig. 3, **Top row**) but does not generalize as well as our method to unseen dynamics (Fig. 3, **Bottom row**).

D. Policy Behavior

We provide an empirical analysis on each method's behavior and the effect on its performance. In Figs. 3-**Center** and 3-**Right**, we test our policy in simulated perception environments with 30 targets and report the number of unclassified targets simultaneously observed along the episode. During the first half of the episode, our method consistently is shown to observe and provide classification estimates of more targets than other baselines, which results in faster classification of targets. This shows that our method is able to learn the effect of each target on the observations of others and discover groups of simultaneously observable targets. The latter is difficult to achieve by single-target classification baselines or methods that assume distance-based relations among targets. Similarly, the lower performance albeit similarly high simultaneous observations of the *DeepSets decoder* baseline, especially at the end of the episode when there are less targets to classify, suggest that the attention-based pooling layer allows better aggregation of each target's latent information, effectively establishing a classification priority order. The sharp decline in simultaneous observations of

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.

Fig. 4. Pedestrian models, as seen from the drone's perspective in the realistic environment. Each pedestrian is tracked using AirSim's API. Its class is represented by a red number in the front, classified using YoloV3.



(a) -100^o (b) -140^o (c) $\pm$180^o (d) 140^o

Fig. 5. Probability heat-map of the employed black-box classifier for a target of class 1. For a robot placed at the bottom, facing upwards and a FOV of 90°, every plot shows the output probability of identifying the real class of the target at different relative positions and a fixed relative orientation from the robot.

unclassified targets is due to simultaneously observable targets becoming classified, which results in a sparser distribution of unclassified targets.

VII. Photo-Realistic Results

To ascertain our framework's capabilities under realistic conditions, we test its performance in an environment generated with Unreal Engine using AirSim [36] to simulate drone control and perception. We simulate our targets using open-source 3D human meshes, produced in *MakeHuman* [37], which move to random goals while avoiding collisions in an environment of $50 \times 50\, m^2$. As shown in Fig. 4, to remain close to the use-case shown in the earlier simpler environment, pedestrian classes are denoted by a red number in their front.

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

An implementation of the YoloV3 [38] algorithm is used to detect and classify the digit in each processed image. We train the algorithm using a dataset of rotated, up-scaled and down-scaled MNIST images. Both YoloV3 implementation and the adapted dataset have been obtained from [39]. The output of the algorithm consists in a set of bounding boxes, each one associated to a detected digit and a normalized score stating how sure the algorithm is of it's detection. We rule out those boxes containing a number of pixels smaller than a threshold $B_a = 2000$ and take the box with the highest score. The normalized score of the detected digit is used as the target class probability estimate, distributing the remaining probability mass among the other classes. The output is a uniform distribution when no digits are detected, or the normalized score is below the uniform distribution.

2) Training Conditions: We use the training setup explained in Section V-A, using the realistic observation model. However, to avoid the computational cost of running Unreal/Airsim and YoloV3, for training we use AirSim to extract a dense library of pedestrian observations of all classes in different relative poses from the drone's FOV. For each class, each pose-dependent image is used to compute and save a probability distribution. During training, for each target in the drone's FOV, we use the probability distribution of the library image whose pose is the closest to the target's current relative pose. Being in a controlled environment, during training we monitor the output of the perception system to detect classification errors, and substitute them by a uniform probability. This enables our policy to prioritise good over bad classification viewpoints, adding an additional robustness mechanism to outlier classifications to the control policy. An example of the resulting normalized classification score is given in Fig. 5.

3) Real Perception Environment: This time we test our method's performance in the photo-realistic environment. In Fig. 6(a) and (b), we only analyze the performance of the viewpoint recommendation policy without having AirSim's different drone dynamics and in-built controller affecting the results, which we use for comparison in Fig. 6(c). We show results for our method and how it compares in this new setting to the baselines presented in Section VI-A.

As in section VI, we evaluate each method 50 times in environments containing up to 40 targets for 75 and 100 seconds, and show the resulting mean and standard error. In all our experiments, the mean percentage of misclassifications was under 3% with no significant differences among different methods. Fig. 6 shows that the results presented in Section VI hold in the new photo-realistic environment, even when realistic drone dynamics and control are used. Note that the different performance in comparison to Fig. 3 is due to the different observation model and the fact that we take measurements directly from the viewpoints recommended by the DRL policy in Fig. 6(a) and (b). This is very relevant from the Sim-to-Real perspective, considering that the policy has been trained in a different environment with a perception system approximated from the one used during testing.

VIII. Conclusion

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

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.

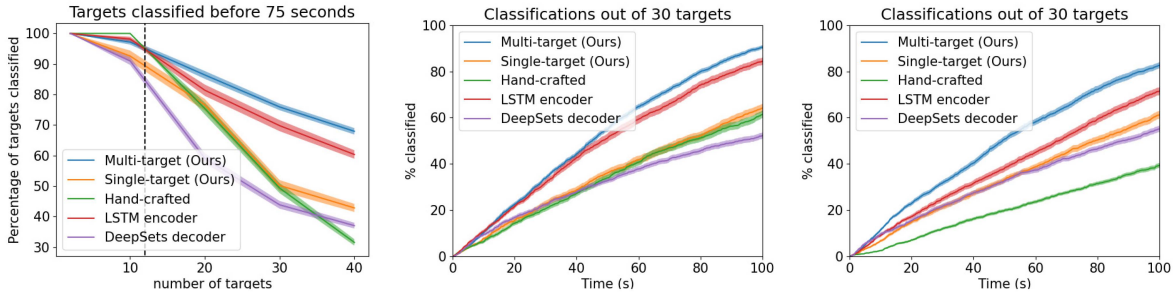


Fig. 6. Mean and standard error over 50 experiments in the photo-realistic simulator. (a) Comparison of the percentage of targets classified before timeout of 75 seconds in environments with 1 to 40 targets, where measurements are directly taken from viewpoints that the DRL policy suggests. (b) Classification speed in the experiment of (a) with 30 targets. (c) Classification speed in environments of 30 targets with realistic drone dynamics and AirSim's controller.

REFERENCES

[1] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[2] I. Alonso, L. Riazuelo, L. Montesano, and A. Murillo, "Domain adaptation in LiDAR semantic segmentation by aligning class distributions," in *Proc. Int. Conf. Inform. Control, Autom. Robot.*, 2021, Art. no. 124899.

[3] T. Patten, M. Zillich, R. C. Fitch, M. Vincze, and S. Sukkarieh, "Viewpoint evaluation for online 3-D active object classification," *IEEE Robot. Automat. Lett.*, vol. 1, no. 1, pp. 73–81, Jan. 2016.

[4] M. Popović, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran, "Online informative path planning for active classification using UAVs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5753–5758.

[5] N. Atanasov, B. Sankaran, J. Le Ny, G. J. Pappas, and K. Daniilidis, "Nonmyopic view planning for active object classification and pose estimation," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1078–1090, Oct. 2014.

[6] T. Patten, W. Martens, and R. Fitch, "Monte Carlo planning for active object classification," *Auton. Robot.*, vol. 42, no. 02, pp. 391–421, 2018.

[7] J. Sock, G. Garcia-Hernando, and T.-K. Kim, "Active 6D multi-object pose estimation in cluttered scenarios with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020, pp. 10564–10571.

[8] Q. Xu et al., "Towards efficient multiview object detection with adaptive action prediction," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 13423–13429.

[9] D. Kent and S. Chernova, "Human-centric active perception for autonomous observation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1785–1791.

[10] A. Alcántara, J. Capitán, R. Cunha, and A. Ollero, "Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles," *Robot. Auton. Syst.*, vol. 140, 2021, Art. no. 103778.

[11] B. F. Jeon, D. Shim, and H. Jin Kim, "Detection-aware trajectory generation for a drone cinematographer," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1450–1457.

[12] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," 2021, *arXiv:2104.13478*.

[13] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "DenseCAvoid: Real-time navigation in dense crowds using anticipatory behaviors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 11345–11352.

[14] Y. Cui, H. Zhang, Y. Wang, and R. Xiong, "Learning world transition model for socially aware robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9262–9268.

[15] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11785–11792.

[16] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2754–2761, Apr. 2020.

[17] B. Brito, M. Everett, J. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation among pedestrians," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4616–4623, Jul. 2021.

[18] M. Everett, Y. F. Chen, and J. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10357–10377, 2021.

[19] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson, "My body is a cage: The role of morphology in graph-based incompatible control," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=N3zUDGN5lO

[20] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep Sets," in *Advances in Neural Information Processing Systems*, vol. 30. Red Hook, NY, USA: Curran Associates, Inc., 2017.

[21] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6015–6022.

[22] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 1–11.

[23] C. D. Hsu, H. Jeong, G. J. Pappas, and P. Chaudhari, "Scalable reinforcement learning policies for multi-agent control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4785–4791.

[24] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3744–3753.

[25] Y. Sung and P. Tokekar, "Algorithm for searching and tracking an unknown and varying number of mobile targets using a limited fov sensor," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 6246–6252.

[26] T. Hill and J. Miller, "How to combine independent data sets for the same quantity," *Chaos: An Interdiscipl. J. Nonlinear Sci.*, vol. 21, no. 3, 2011, Art. no. 033102.

[27] M. Schlichtkrull, T. Kipf, P. Bloem, R. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Extended Semantic Web Conf.*, 2018, pp. 593–607.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[29] E. Liang et al., "RLlib: Abstractions for distributed reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3053–3062.

[30] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1352–1361.

[31] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for MAVs in dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 776–783, Apr. 2019.

[32] L. Engstrom et al., "Implementation matters in deep RL: A case study on PPO and TRPO," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1etN1rtPB

[33] M. Andrychowicz et al., "What matters for on-policy deep actor-critic methods? a large-scale study," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=nIAxjsniDzg

[34] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

[35] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, 1998, Art. no. 4282.

[36] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*. Berlin, Germany: Springer, 2018, pp. 621–635.

[37] "Makehuman community," 2022. [Online]. Available: http://www.makehumancommunity.org

[38] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, *arXiv:1804.02767*.

[39] "TensorFlow 2 YOLOv3 mnist detection training tutorial. Pylessons, 2020," [Online]. Available: https://pylessons.com/YOLOv3-TF2-mnist

Authorized licensed use limited to: TU Delft Library. Downloaded on May 16,2023 at 12:25:36 UTC from IEEE Xplore. Restrictions apply.