Semantic Tracklets: An Object-Centric Representation for Visual Multi-Agent Reinforcement Learning

Iou-Jen Liu*[1], Zhongzheng Ren*[1], Raymond A. Yeh*[1], Alexander G. Schwing[1]

***Abstract*— Solving complex real-world tasks, *e.g.*, autonomous fleet control, often involves a coordinated team of multiple agents which learn strategies from visual inputs via reinforcement learning. Many existing multi-agent reinforcement learning (MARL) algorithms however don't scale to environments where agents operate on visual inputs. To address this issue, algorithmically, recent works have focused on non-stationarity and exploration. In contrast, we study whether scalability can also be achieved via a disentangled representation. For this, we explicitly construct an object-centric intermediate representation to characterize the states of an environment, which we refer to as 'semantic tracklets.' We evaluate 'semantic tracklets' on the visual multi-agent particle environment (VMPE) and on the challenging visual multi-agent GFootball environment. 'Semantic tracklets' consistently outperform baselines on VMPE, and achieve a $+2.4$ higher score difference than baselines on GFootball. Notably, this method is the first to successfully learn a strategy for five players in the GFootball environment using only visual data. For more, please see our project page: https://ioujenliu.github.io/SemanticTracklets**

I. Introduction

Many real-world tasks, such as autonomous fleet control [1] and swarm robot control [2, 3], are naturally modeled as visual multi-agent systems. In these systems, multiple agents learn to coordinate based on pixel inputs.

Many existing works in multi-agent reinforcement learning (MARL) study learning of multi-agent coordination [4–13]. In common to all these works is the use of a compact observation vector which summarizes the situation for each agent. For instance, on the simulated particle world [14] the Multi-Agent Actor-Critic [4] method uses the locations and velocities of controlled and neighboring particles as input to achieve compelling results, albeit, training times are often significantly longer than those of single-agent reinforcement learning.

Despite this success, few works for MARL focus on visual agents which act purely upon image observations. Indeed, Visual Multi-Agent Reinforcement Learning (VMARL) which operates on pixel inputs adds additional complexity to the already challenging MARL task, increasing training times often significantly. Unsurprisingly, current VMARL methods trained end-to-end with conv-nets hardly learn useful polices in complex environments. For example, in the Google Research Football (GFootball) environment, end-to-end training with conv-nets doesn't result in meaningful policies and remains an open problem [15]. To tackle this challenge,

[1]University of Illinois at Urbana-Champaign, IL, USA.
iliu3@illinois.edu
* Equal contribution.

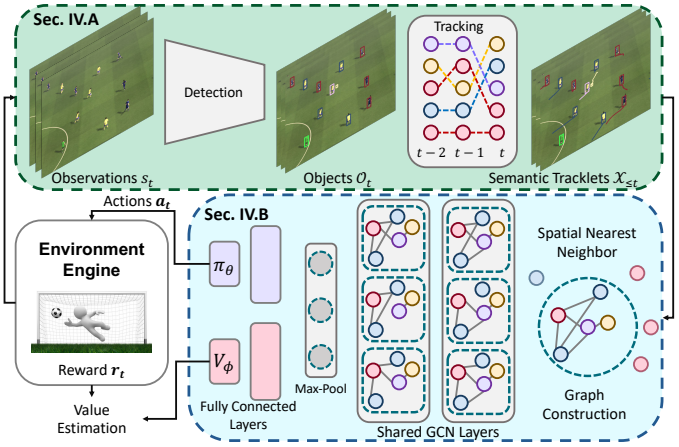

Fig. 1. The proposed visual MARL system illustrated using the GFootball environment as an example. Given image observations, we first perform object detection to identify the objects (*e.g.*, ball, players on the home and the visiting team shown in **yellow**, **red** and **blue**). We perform tracking to maintain the identity of the detected objects across frames before extracting semantic tracklets. Next, we construct a graph representation using the K-nearest neighbors around the controlled agent (active player), shown in **purple**. Lastly, we compute the policy and value function via a graph convolutional net (GCN) and interact with the environment.

existing works primarily use imitation learning to reduce the long training times. For example, Jain *et al.* [16], who study a collaborative navigation task involving up to three agents, first train with imitation learning [17] followed by RL fine-tuning. However, imitation learning requires "expert" demonstrations that may be hard to obtain. This is particularly true in complex MARL environments like GFootball, where strategies are necessary. Note, a ground-truth "expert" strategy may not even exist.

In this paper, we study an alternative approach to make VMARL more scalable. We design an intermediate representation for more efficient training by building helpful inductive biases into the models. For this we propose 'semantic tracklets,' an object-centric representation to characterize the surroundings of each agent. Different from imitation learning, this approach does not require "expert" demonstrations, *i.e.*, *annotations in the form of actions*. Instead, it uses the more accessible alternative of *annotations in the form of object labels*, which can be collected by "anyone" without domain knowledge of the underlying task.

At a high-level and pictorially sketched in Fig. 1, semantic tracklets contain two components: the 'semantics' which answer "what" is the object; and the 'tracklets' which answer "where" is the object. This representation provides a useful inductive bias for VMARL, where the agents correspond to an "object" in the given pixel inputs, and agents' interaction

with the environment involves spatial movements. Furthermore, such an object-centric representation enables the use of graph-nets which naturally model coordination/interactions among the agents while being invariant to the agents' ordering, an important property for homogeneous agents [8, 18].

We evaluate our approach on two environments, the Visual Multi-Agent Particle Environment (VMPE), and the challenging Google Research Football Environment (GFootball) with only visual input [15]. On VMPE, we compare semantic tracklets with a baseline that directly uses pixel inputs, and a baseline that uses semantic segmentation as an intermediate representation [19]. Semantic tracklets consistently outperform baselines on VMPE. On GFootball *11* vs. *11* full game, semantic tracklets achieve a $+2.4$ higher score difference than baselines. We show that 'semantic tracklets' permit to learn, for the first time, collaboration among five players in the GFootball environment using only visual input. The players learn to cooperate and outperform their opponent.

II. Related Work

Intermediate Representations. In reinforcement learning, the representation which characterizes the state of an environment has a direct impact on the accuracy and efficiency of the learned policy. While end-to-end learning-based methods, which learn a policy from the pixel space [20–25], have achieved impressive results, others have demonstrated that designing intermediate representations may be beneficial [19, 26]. For example, Hong *et al*. [19] study the use of semantic segmentation to tackle the 'virtual-to-real' [19, 27, 28] problem in single-agent robot navigation tasks.

More recently, object-centric representations demonstrate promising results in various domains including robot learning [29] and autonomous-driving [30]. For example, Ye *et al*. [30] build an object-centric representation for learning polices to perform robotic manipulation tasks. Generally, these approaches rely on object detection, trained with annotated data, to maintain a structured representation of the objects.

We note that the aforementioned methods consider only single-agent settings. The interactions between multiple controlled agents, which are critical for multi-agent learning, are not discussed. Moreover, some of the existing works [30] rely on supervised learning or imitation learning to train policies. Different from these works, we study an object-centric representation for VMARL. The proposed semantic tracklets permit to capture the interaction of multiple agents via graph-nets and to learn cooperative policies efficiently.

Multi-Agent RL. To efficiently learn policies in multi-agent systems, a variety of multi-agent RL algorithms have been proposed [4–13, 31–34]. For example, to cope with non-stationarity, 'Multi-agent Actor-critic' [4] uses a centralized critic which operates on all agents' observations and actions. 'Monotonic Value Function Factorisation' [9] advocates estimating joint action-values as a non-linear combination of per-agent values. However, all approaches assume a compact feature vector as input. Scaling of MARL approaches to agents operating with visual observations remains open in those works.

Visual Multi-Agent RL. Jain *et al*. [16] study interaction of two visual agents in AI2Thor [35]. For training to scale, imitation learning [17] is used initially, which requires access to an 'expert,' *i.e*., *annotations for actions*. For complex environments and particularly for multi-agent tasks, such an 'expert' may not be available. Hence, we study an alternative: building an object-centric representation to learn the policies efficiently. For this our method only requires *annotations of objects*, which are obtained more easily or are even already available from off-the-shelf detectors.

Visual-rich environments, *e.g*., AI2Thor [35], Habitat [36], and iGibson [37], permit to study embodied AI tasks, such as navigation and question answering. While some of the environments [35, 37] have recently been extended to support multi-agent training, the involved tasks are often light on collaboration. This is largely due to the fact that navigation is considered important, which leaves little room for collaboration. For this reason, here, we consider GFootball [15], where agents are required to play soccer given as observation rendered game frames. The environment supports both multi-agent and single agent settings of visual RL. The model can control any number of players on a team, from one to all players, making it particularly suitable to study collaboration.

For GFootball, Kurach *et al*. [15] consider controlling only one visual agent in the *11* vs. *11* full game. The method of Kurach *et al*. [15] requires 500 parallel actors and more than 100 million environment training steps. In contrast, with 'semantic tracklets,' we successfully learn polices to control up to five visual agents in the *11* vs. *11* full game within 20 million environment steps.

III. Preliminaries

RL studies how agents should interact with an environment to maximize their expected future rewards. We first provide background on single- and multi-agent RL.

Single-Agent RL. An environment is commonly modeled as a Markov Decision Process (MDP). Formally, an MDP is defined by: the environment's state space $\mathcal{S}$, the set of actions $\mathcal{A}$ which an agent can perform, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \Delta_{\mathcal{S}}$ specifying for each state the probability with which it is reached next when performing an action in the current state, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ denoting the prize for executing an action in a given state.

An agent's interaction with the environment is described via a policy π_θ, parameterized by θ. Given the state $s_t \in \mathcal{S}$ at time step t, the policy models the probability of performing an action a, *i.e*., $\pi_\theta(a|s_t) \in [0, 1]$ and $\sum_{a\in\mathcal{A}} \pi_\theta(a|s_t) = 1$.

RL aims to find the policy π_θ that maximizes the expected return $J(\theta) \triangleq \mathbb{E}_{s_t\sim\rho^{\pi_\theta}, a_t\sim\pi_\theta}[R_t]$, where $R_t \triangleq \sum_{k=0}^{K} \gamma^k r_{t+k}$ is the expected discounted return. Here, $K \in \mathbb{Z}^+$ denotes the time horizon, ρ^{π_θ} is the state distribution induced by π_θ, $r_{t+k} = \mathcal{R}(s_{t+k}, a_{t+k})$, and γ is a discount factor.

Proximal Policy Optimization (PPO). To learn the parameters θ of the policy π_θ, PPO [23] is a commonly used

algorithm. PPO employs the 'surrogate' objective function

$$\hat{J}_{\text{PPO}}(\theta) = \mathbb{E}_{(s_t,a_t,r_t)\sim\rho_{\pi_\theta}}[\min(\xi_t(\theta)D_t, [\![\xi_t(\theta)]\!]_{1-\epsilon}^{1+\epsilon}D_t) + \lambda H(\pi_\theta(\cdot|s_t))], \quad (1)$$

where $[\![\cdot]\!]_a^b$ denotes clipping to interval $[a, b]$, and $\xi_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ denotes the probability ratio of the current policy $\pi_\theta(a_t|s_t)$ to a slightly outdated policy $\pi_{\theta_{\text{old}}}(a_t|s_t)$. $D_t = R_t^{(n)} - V_\phi(s_t)$ is the estimated advantage function, where $R_t^{(n)} = \sum_{k=0}^{n-1}\gamma^k r_{t+k} + \gamma^n V_\phi(s_{t+k})$ is the n-step truncated return, and V_ϕ is the value function parameterized by ϕ.

Intuitively, this surrogate objective function encourages to learn a good policy, while preventing policy collapse by ensuring that the new policy does not deviate too much from the old policy. Additionally, the entropy term H is added to encourage exploration. The parameters ϕ of the value function V_ϕ are learned by minimizing the squared loss $J_V(\phi) = \mathbb{E}_{(s_t,r_t)\sim\rho_{\pi_\theta}}[(R_t^{(n)} - V_\phi(s_t))^2]$.

Multi-agent RL. A multi-agent MDP with N agents consists of a state space $\mathcal{S}$, a transition function $\mathcal{T}$, a set of reward functions $\{\mathcal{R}^i\}_{i=1}^N$, and a set of action spaces $\{\mathcal{A}^i\}_{i=1}^N$, where $\mathcal{R}^i$ and $\mathcal{A}^i$ correspond to reward and action space of agent i. The transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \cdots \times \mathcal{A}^N \mapsto \Delta_{\mathcal{S}}$ maps the current state and actions taken by all N agents to a next state.

At each time step t, each agent i takes action a_t^i and receives state s_{t+1} and reward r_t^i from $\mathcal{R}^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \mapsto \mathbb{R}$. We denote all actions and rewards at time t by $\boldsymbol{a_t} = (a_t^1, \ldots, a_t^N)$ and $\boldsymbol{r_t} = (r_t^1, \ldots, r_t^N)$. Note, most MARL works [4, 9, 11] assume each agent receives an individual compact local observation. In contrast, in our setting, each agent only has access to the same rendered image, denoted by s_t. Operating on the same observation is often a more challenging setting than having access to individual local observations. For instance, in many video games, *e.g.*, Starcraft 2, Dota, and Age of Empires, a local first-person visual observation for individual units is not available. In those games, players only have access to the game image. Moreover, in a fulfillment center, low-cost robots can be controlled using classical security camera footage.

Consider N policies $\Pi = \{\pi_{\theta^i}\}_{i=1}^N$ with parameters $\{\theta^i\}_{i=1}^N$, and N value functions $\{V_{\phi^i}\}_{i=1}^N$ with parameters $\{\phi^i\}_{i=1}^N$, which are associated with the N policies. Multi-Agent Actor-Critic [4] extends the objective function of PPO, given in Eq. (1), to read

$$\hat{J}_{\text{MAPPO}}(\theta^i) = \mathbb{E}_{(s_t,\boldsymbol{a_t},\boldsymbol{r_t})\sim\rho_\Pi}[\min(\xi_t^i D_t^i, [\![\xi_t^i(\theta)]\!]_{1-\epsilon}^{1+\epsilon}D_t^i) + \lambda H(\pi_{\theta^i}(\cdot|s_t))]. \quad (2)$$

Here ρ_Π denotes the trajectory distribution induced by Π, and $\xi_t^i = \frac{\pi_{\theta^i}(a_t^i|s_t)}{\pi_{\theta_{\text{old}}^i}(a_t^i|s_t)}$ is the probability ratio. $D_t^i = R_t^{i,(n)} - V_{\phi^i}(s_t)$ is the estimated advantage function, where $R_t^{i,(n)} = \sum_{k=0}^{n-1}\gamma^k r_{t+k}^i + \gamma^n V_{\phi^i}(s_{t+k})$. Value function V_{ϕ^i} is updated by minimizing $J_V(\phi^i) = \mathbb{E}_{(s_t,r_t^i)\sim\rho_{\boldsymbol{\Pi}}}[(R_t^{i,(n)} - V_{\phi^i}(s_t))^2]$. We will next describe how semantic tracklets enable to learn effective VMARL policies.

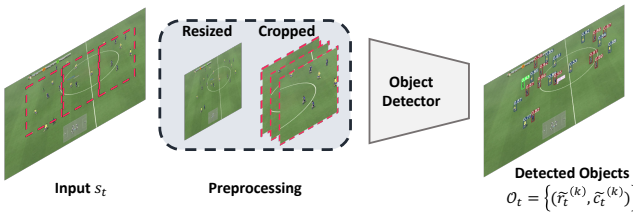

Fig. 2. Illustration of detection pipeline used for GFootball.

IV. VMARL with Semantic Tracklets

Our goal is to develop a representation that permits to effectively learn a multi-agent strategy, *i.e.*, policies Π, given only visual observations. A good set of policies Π maximizes the agents' collaborative abilities to collect rewards.

As previously motivated, we aim to mitigate the issue of scalability for VMARL by designing intermediate representations for the policy and value networks. For this we study 'semantic tracklets,' an object-centric representation that includes inductive biases about "where" and "what," which are useful for VMARL. Semantic tracklets capture both the role and movement of each agent throughout the environment. Given this object-centric representation, we then learn the policy and value functions using graph-nets, as they excel in tasks that involve reasoning and modeling of interactions. In the following we describe our approach using the GFootball environment as a running example.

The overall method is illustrated in Fig. 1 and consists of two main components. We highlight the semantic tracklet generation in green: given observations $s_t \in \mathbb{R}^{H\times W\times 3}$ consisting of pixels, we construct the semantic tracklets. Next, highlighted in blue, semantic tracklets are transformed into complete graphs, where one object corresponds to a node. Subsequently, policy and value networks predict an action and the truncated return for each agent. We describe both components in detail next.

A. Semantic Tracklet Generation

Formally, semantic tracklets for a play up to frame t are represented as an *unordered set* of tuples, *i.e.*,

$$\mathcal{X}_{\leq t} \triangleq \{\mathbf{x}_{\leq t}^{(1)}, \ldots, \mathbf{x}_{\leq t}^{(m)}\}. \quad (3)$$

The set contains m elements corresponding to the m detected objects, *e.g.*, the players and the ball. Each element is an *ordered tuple*

$$\mathbf{x}_{\leq t}^{(o)} \triangleq (\mathbf{x}_t^{(o)}, \mathbf{x}_{t-1}^{(o)}, \ldots), o \in \{1, \ldots, m\}, \quad (4)$$

which is associated with the trajectory of an object, *i.e.*, $\mathbf{x}_t^{(o)} \triangleq [\mathbf{r}_t^{(o)}, \mathbf{c}_t^{(o)}, \mathbf{g}_t^{(o)}]$. Here, $\mathbf{r}_t^{(o)}$ represents the object's role, *e.g.*, ball or person in the GFootball environment, $\mathbf{c}_t^{(o)}$ represents each object's spatial-coordinates, and $\mathbf{g}_t^{(o)}$ subsumes global information, *e.g.*, distance to the edge of the environment.

To generate the semantic tracklets $\mathcal{X}_{\leq t}$ from an observation s_t, we first run an object detector to identify the objects of interest. This process is illustrated in Fig. 2. Given the input image s_t, we resize and crop the image

in preparation for the object detector. This preprocessing permits to detect small objects. Subsequently, the object detector yields information about the relevant objects, *e.g.*, the agents. More details regarding the detectors are provided in the experimental section.

Formally, the detections are first subsumed in an unordered set of objects $\mathcal{O}_t = \{(\tilde{\mathbf{r}}_t^{(k)}, \tilde{\mathbf{c}}_t^{(k)})\}$, containing the role and spatial location for the detected objects. We use the tilde symbol ("˜") to indicate that these detections are not aligned across time, *i.e.*, the k^{th} object for frames at time t and $t-1$ may differ.

To maintain a consistent correspondence of objects across time, we will align the objects in $\mathcal{O}_t$. For this we use an object tracking formulation which identifies the correspondence between objects in $\mathcal{O}_{t-1}$ and $\mathcal{O}_t$.

Specifically, assume we are given an ordered set of objects $\{(\mathbf{r}_{t-1}^{(o)}, \mathbf{c}_{t-1}^{(o)})\}$ for the previous time step $t-1$. Intuitively, tracking of the "unordered" objects $\mathcal{O}_t = \{(\tilde{\mathbf{r}}_t^{(k)}, \tilde{\mathbf{c}}_t^{(k)})\}$ is equivalent to assigning each element in the unordered set to an element from the ordered set of the previous time step. This is formulated as the unbalanced assignment problem

$$\min_{\boldsymbol{M}_t} \sum_{o,k} \boldsymbol{M}_{t,ok} \left(\left\| \mathbf{c}_{t-1}^{(o)} - \tilde{\mathbf{c}}_t^{(k)} \right\|_2^2 + d(\mathbf{r}_{t-1}^{(o)}, \tilde{\mathbf{r}}_t^{(k)}) \right),$$
$$\text{s.t. } \boldsymbol{M}_{t,ok} \in \{0,1\}, \sum_o \boldsymbol{M}_{t,ok} \leq 1, \sum_k \boldsymbol{M}_{t,ok} = 1. \quad (5)$$

Hereby $\boldsymbol{M}_t$ denotes an assignment matrix. The cost for assigning $\tilde{\mathbf{c}}_t^{(k)}$ to $\mathbf{c}_{t-1}^{(o)}$ is the sum of an ℓ_2-loss on the object's coordinates and a distance d between their roles. For example, when $\mathbf{r}_t^{(o)}$ is categorical, d can be the zero-one loss. This assignment cost encourages to match objects with the same role that are spatially close.

We solve the program given in Eq. (5) by reducing the unbalanced assignment problem to a balanced one before using the Hungarian algorithm [38]. This is achieved by adding surrogate matches with zero loss. Given the assignment $\boldsymbol{M}_t$, we update the location estimates to prepare for the next time step, *i.e.*, $\forall o\ (\mathbf{r}_t^{(o)}, \mathbf{c}_t^{(o)}) \leftarrow (\tilde{\mathbf{r}}_t^{(k)}, \tilde{\mathbf{c}}_t^{(k)})$ if object k is assigned to object o, *i.e.*, if $\boldsymbol{M}_{t,ok} = 1$. If no object is assigned, *i.e.*, if it is assigned to a surrogate match, we keep the previous tracked roles and locations, *i.e.*, $(\mathbf{r}_t^{(o)}, \mathbf{c}_t^{(o)}) \leftarrow (\mathbf{r}_{t-1}^{(o)}, \mathbf{c}_{t-1}^{(o)})$. This case happens when there are missing detections.

With the objects aligned across time, we then compute the global information $\mathbf{g}_t^{(o)}$ to complete our proposed semantic tracklets $\mathcal{X}_{\leq t}$. We will next describe how to use graph-nets to learn policies and value functions from this object-centric representation.

B. Policy and Value Networks

Classical deep nets operate on a vector or matrix. Concatenating all elements in the tracklet $\mathcal{X}_{\leq t}$ is an intuitive way to construct a vector. However, vectorizing *implicitly* assumes an ordering of the set's elements. Note, permuting the elements in the input vector of a standard deep net will result in a *different* deep net output. This is undesirable as the environment configuration did not change when permuting the agents. Therefore, we use a graph-net to model the policy $\pi_{\theta^i}(\cdot|\mathcal{X}_{\leq t})$ and the value function $V_{\phi^i}(\mathcal{X}_{\leq t})$. This guarantees permutation invariance w.r.t. the agents' ordering, *i.e.*, the graph-net's output remains identical irrespective of the input permutation. Specifically, we use graph convolutional nets (GCNs) to model both functions.

Graph Construction from Semantic Tracklets. For each controlled agent, we construct a complete graph using the K nearest neighbor objects. Each object o is a node. For each node o, we use an embedding $\Phi_0^{(o)}$, constructed from the semantic tracklets. Each node embedding uses the most recent four frames of the semantic tracklets, *i.e.*, $\Phi_0^{(o)} \triangleq [\mathbf{x}_t^{(o)}, \mathbf{x}_{t-1}^{(o)}, \mathbf{x}_{t-2}^{(o)}, \mathbf{x}_{t-3}^{(o)}]$. We next describe the GCN architecture to learn the policy and value functions from the node embeddings.

Permutation Invariant Architecture. A graph convolution $f_{\text{g}}^{(l)}$ at layer l is defined as follows:

$$f_{\text{g}}^{(l)}(\Phi_{l-1}) \triangleq \sigma\left(\frac{\boldsymbol{A}\Phi_{l-1}W_{\text{other}}^{(l)} + \Phi_{l-1}W_{\text{self}}^{(l)}}{K+1} \right). \quad (6)$$

Here, $K+1$ denotes the number of nodes in the graph, *i.e.*, the K nearest neighbors plus the agent's node. We let $\boldsymbol{A}$ denote the adjacency matrix of the graph, $\Phi_l \in \mathbb{R}^{(K+1)\times K_{\text{in}}^{(l)}}$ denotes the feature matrix at the l^{th} layer, where each row is a $K_{\text{in}}^{(l)}$-dimensional feature $\Phi_l^{(i)}$ of an object. The layer's trainable parameters are $W_{\text{other}}^{(l)}, W_{\text{self}}^{(l)} \in \mathbb{R}^{K_{\text{in}}^{(l)} \times K_{\text{out}}^{(l)}}$, where $K_{\text{in}}^{(l)}$ and $K_{\text{out}}^{(l)}$ denote the input and output dimensions of the l^{th} layer.

Our policy and value network share parameters for their GCN. However, their fully connected layers, *i.e.*, their multi-layer perceptron (MLP), differ. This gives rise to the following formulation:

$$\Phi_L = \text{MaxPool}\left(f_g^L \circ f_g^{L-1} \circ \ldots \circ f_g^1(\Phi_0)\right) \quad (7)$$
$$\pi_{\theta^i}(\cdot|\mathbf{x}_{\leq t}) = \text{Softmax}(f_{\pi,\text{MLP}}(\Phi_L)) \quad (8)$$
$$V_{\phi^i}(\mathbf{x}_{\leq t}) = f_{V,\text{MLP}}(\Phi_L). \quad (9)$$

Note that each row of Φ_0 consists of semantic tracklets $\Phi_0^{(o)}$ of object o. The max pooling, in Eq. (7), is performed over the first (agent) dimension. This ensures a permutation invariant representation Φ_L as max-pooling ignores the permutation. In Eq. (8), $f_{\pi,\text{MLP}}$ refers to a MLP to model the policy function. Similarly, $f_{V,\text{MLP}}$ in Eq. (9) refers to a MLP for the value network. These MLPs consist of fully connected layers with ReLU non-linearity. To learn the policy functions' parameters $\{\theta^i\}_{i=1}^N$, and the value functions' parameters $\{\phi^i\}_{i=1}^N$, the PPO algorithm described in Sec. III is used.

V. Experiments

We conduct experiments on two visual multi-agent environments: the Visual Multiple Particle Environment (VMPE), which mimics the original MPE [4, 8], and the Google Research Football (GFootball) environment. We demonstrate the success of 'semantic tracklets' in those visual multi-agent environments, controlling up to 6 visual agents on the same team at once.

Inter. Rep.	Arch.	*Visual Cooperative Navigation*		*Visual Prey and Predator*		*Visual Cooperative Push*	
		$N = 3$	$N = 6$	$N = 3$	$N = 6$	$N = 3$	$N = 6$
None	CNN	-654.7±21.8	-4780.4±272.1	-27.3±0.4	-107.2±1.2	-349.5±1.1	-1384.1±32.2
Segmentation	CNN	-653.1±4.6	-4518.0±226.1	-26.3±0.4	-96.6±2.4	-352.9±0.9	-1494.3±62.1
Tracklets	MLP	-392±3.1	-4040.0±12.4	-1.3±1.4	46.8±3.4	-345.4±1.4	-1360.9±29.5
Tracklets	GCN	**-381.1±11.2**	**-3642.7±46.8**	**3.6±2.1**	**279.4±24.3**	**-217.6±19.5**	**-1098.9±70.2**

TABLE I

AVERAGE EVALUATION EPISODE REWARDS OF BASELINES AND OUR APPROACH ON VMPE TASKS.

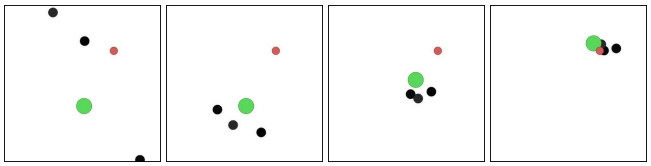

Fig. 3. Qualitative results for *Visual Cooperative Push* ($N = 3$) using semantic tracklets. We observe agents working together to push the large green ball to the red landmark.

Dropout rate (%)	0%	10%	20%	40%
Visual Coop. Navigation ($N = 3$)	-381.1±11	-382.2±9	-391.2±4	-409.1±1
Visual Prey & Pradator ($N = 6$)	279.4±24	273.5±4	253.6±10	252.2±2
Visual Coop. Push ($N = 3$)	-217.6±19	-225.3±10	-233.4±27	-250.3±6

TABLE II

SEMANTIC TRACKLETS' AVERAGE EVALUATION REWARDS *vs.* DIFFERENT DROPOUT RATES.

A. Visual MPE

Environment details. In VMPE, the observation of each agent is an image with agents and landmarks rendered in different colors. This differs from the original MPE [4, 8], where the observation of each agent is a compact vector which summarizes the information of the environment, *i.e.*, the location and velocity of other agents. In the VMPE we consider three tasks with $N = 3$ and $N = 6$ agents:

Visual Cooperative Navigation: N agents work cooperatively to cover N landmarks. The environment reward encourages agents to cover all landmarks.

Visual Prey and Predator: N predators work together to capture $N/3$ faster moving preys. The predators receive positive rewards when colliding with a prey and receive negative rewards when colliding with fellow predators.

Visual Cooperative Push: N agents work cooperatively to push a large heavy ball to a landmark. The agents are rewarded when the big ball approaches the landmark.

Baselines and metrics. We consider a baseline without an intermediate representation, *i.e.*, pixels are directly passed to the policy and value networks. We refer to this via None+CNN. Inspired by Hong *et al.* [19], we also compare with semantic segmentation as an intermediate representation, which we refer to via Segmentation+CNN. To avoid semantic segmentation errors undermining the baseline's performance, *i.e.*, for a stronger baseline, we use ground truth segmentations to train and test the baseline.

For both of the aforementioned representations, the policy and value networks are CNNs. To evaluate our approach, we run 100 evaluation episodes every $1,000$ training episodes. To ensure that the evaluation is rigorous and fair, we follow the evaluation protocol suggested by Colas *et al.* [39] and Henderson *et al.* [40]. We report the final metric, which is the average reward over the last $1,000$ evaluation episodes, *i.e.*, 100 episodes for each of the last ten policies during training.

Implementation details. For semantic tracklets, we use simple thresholding techniques to detect the locations of landmarks and agents. The detected locations and roles are used to construct semantic tracklets. Following [4], we use multi-agent deep deterministic policy gradient (MADDPG) to train our approach and all baselines. Agents are trained for $60,000$ episodes in all tasks (episode length is either 25 or 50 steps).

Visual multi-agent results. In Tab. I, we report quantitative results in final metrics. Across tasks and for different numbers of visual agents we observe semantic tracklets to consistently outperform the baselines that directly use pixels or semantic segmentation as intermediate representation. We also observe that the improvement of using semantic segmentation as intermediate representation over directly using pixels is marginal in this environment. Moreover, semantic tracklets with GCN architecture achieve better rewards than semantic tracklets with MLP. This demonstrates the effectiveness of a GCN-based policy and value network. We visualize our learned polices on *Visual Cooperative Push* ($N = 3$) in Fig. 3, where agents successfully coordinate their behavior to push the large green ball to the red landmark.

Robustness to tracklet quality. We further investigate how tracklet quality impacts RL results. We consider randomly dropping objects' information, *i.e.*, location and role, at different rates using *Visual Cooperative Navigation* ($N = 3$), *Visual Prey and Predator* ($N = 6$), and *Visual Cooperative Push* ($N = 3$). The results are summarized in Tab. II. As shown in Tab. II, when 10% of information is dropped, the rewards drop only slightly across different tasks. When we drop 40% of the information, the reward drops around 10.7%.

B. GFootball

Environment. We consider the following tasks in the GFootball environment with up to $N = 5$ controlled agents:

11 vs. *11:* Two teams, each of 11 players, play a full 90 minutes game and the episode length is 3000. We control $N = 3$ or $N = 5$ agents on the same team aiming to win the football game.

Single goal vs. *lazy:* In this task, the opponents cannot move but they can intercept the ball if it is close by. An episode terminates when the agents score or the maximum episode length of 300 is reached. We control $N = 3$ agents.

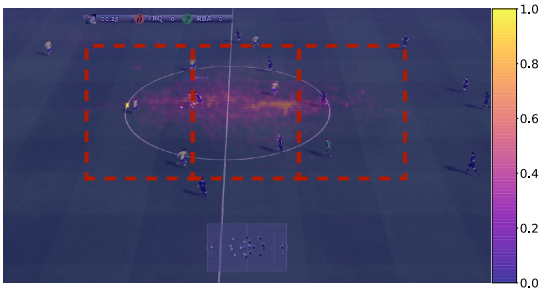

Fig. 4. Probability of ball location in the camera coordinate system. As expected, the camera follows the ball. Hence, the ball appears more likely close to the image center.

Fig. 5. Visualization of the tracking results across different frames with a moving camera. As can be seen, our tracking successfully maintains the identity of the active player, boxed in **purple**, across frames.

Inter. Rep.	Arch.	*11* vs. *11* ($N = 3$)	*11* vs. *11* ($N = 5$)	*single goal* vs. *lazy* ($N = 3$)	*3* vs. *1* with keeper ($N = 3$)	*run, pass and* shoot with keeper ($N = 2$)	pass and shoot with keeper ($N = 2$)
None [15]	CNN	-1.50±0.7	-0.75±0.5	0.10±0.1	0.05±0.07	0.03±0.06	0.08 ±0.05
Tracklets	GCN	**0.54±1.3**	**1.67±1.8**	**0.75±0.3**	**0.98±0.07**	**0.70±0.11**	**0.94±0.02**

TABLE III

AVERAGE SCORE DIFFERENCE OF BASELINES AND OUR APPROACH ON MULTI-AGENT VISUAL GFOOTBALL TASKS.

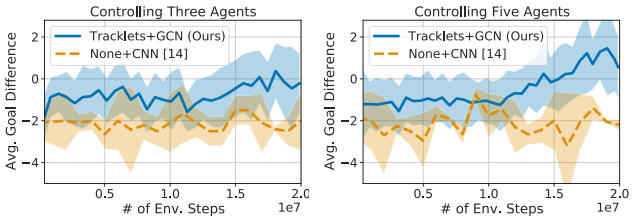

Fig. 6. Training curves. **Left:** Controlling three agents. **Right:** Controlling five agents.

3 vs. *1 with keeper:* Three of our controlled players ($N = 3$) try to score from the edge of the box. One player is at the center and the other two players are on the sides. The center player possesses the ball and faces a defender.

Run, pass and shoot with keeper: Two of our controlled players ($N = 2$) try to score from the edge of the box. One is with the ball and unmarked. The other is at the center, next to a defender, and facing the opponent goal keeper.

Pass and shoot with keeper: Two of our controlled players ($N = 2$) try to score from the edge of the box. One is with the ball and next to a defender. The other one is at the center and facing the opponent goalkeeper.

Baselines and metrics. We compare to Kurach *et al*. [15] who do not use an intermediate representation. Their policy and value networks are modeled using a CNN taking pixel inputs, *i.e*., None+CNN. To evaluate our approach, we run 20 evaluation episodes every 100 training episodes. We report the *absolute metric* [39, 40] of the score difference, which is the best policy's average score difference over 20 evaluation episodes. A positive score difference means the controlled team beats the opponent. Note that in *single goal* vs. *lazy*, the largest possible score difference is one since the episode terminates when the controlled team scores.

Implementation details. Following Kurach *et al*. [15], we train both baseline and our approach with parallel PPO [23] using 24 parallel processes. For *11* vs. *11* and *single goal* vs. *lazy* we train the models for 20M and 5M environment steps.

In order to keep the RL training time low, detection has to be fast. We use YOLOv3(+tiny) [41] as the detection framework. The major challenge is detection of small and fast-moving objects like the ball. To address this challenge, we use the multi-scale scheme illustrated in Fig. 2, which finds the ball and the players simultaneously. Specifically, we detect the players and the ball at two different resolutions as the ball and player sizes differ. To detect the players, we downsample the input image from 720×1280 to 512×512. For detecting the ball, we observe that ball locations are not uniformly distributed as shown in Fig. 4, where we visualize the probability of the ball at each pixel location in the camera view. Leveraging this observation, we perform detection on three cropped regions, shown in Fig. 4, which have a high probability of containing the ball. This permits to avoid a computationally expensive sliding window method. We perform post-processing on the results, which includes standard thresholding, non-maximum suppression for player detections, and using the maximum prediction of the ball, as we know a-priori that there is at most one ball in the game. From the detected objects, we then perform tracking to align the objects across frames as visualized in Fig. 5.

This method detects ball and players at a frame rate of about 50 FPS while having a small memory footprint of around 960 MB. This efficiency permits easy integration of object detection into RL. To train this detector, we collected images with ground-truth bounding boxes from the game engine by running a random policy for 4 episodes (12,000 images). Note, this trained detector can be used across different GFootball tasks with differing agent numbers.

Visual multi-agent results. We report quantitative results in Tab. III, where None+CNN is the baseline [15], which uses image observations and a CNN. Tracklets+GCN is our approach. As shown in Tab. III, our approach has a $+2.42$ higher score difference than the baseline on the *11* vs. *11* task when controlling five visual agents. The training curve

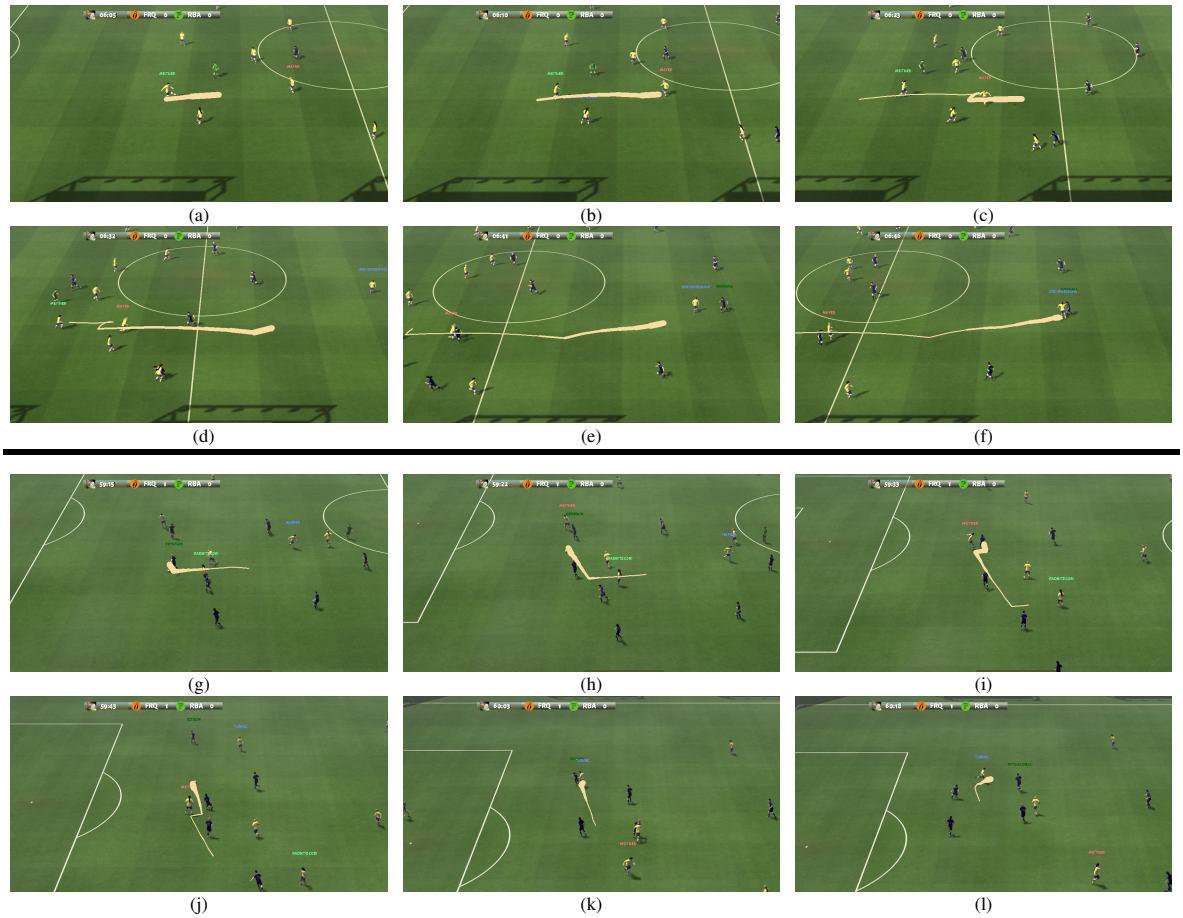

Fig. 7. Multi-agent qualitative results for *11* vs. *11* ($N = 3$). (a-f) Fast break. (g-l) Steal and Pass. We visualize the trajectory of the ball in yellow. The thickness of the line indicates the time direction, *i.e.*, the thicker the more recent in time. Our policy controlled the team in **yellow jerseys**. The controlled players are visualized using a light **red**, **green**, and **blue** name tag on top of each player.

of each method is shown in Fig. 6, where we averaged over three runs with different random seeds. We observe that 'semantic tracklets' are more data efficient. Similarly, for *single goal* vs. *lazy*, *3* vs. *1 with keeper*, *Run, pass and shoot with keeper*, and *Pass and shoot with keeper*, the approach consistently achieves a higher score difference than the baseline.

To better understand the learned policies, in Fig. 7, we visualize the learned coordination when controlling three agents. In Fig. 7 (a-f), our controlled agents pass the ball to players in the front of the court to complete a fast break. In Fig. 7 (g-l), the controlled agents work together to finish a steal and pass. More specifically, in (g,h) the opponent (dark jersey) attempts to pass and in (i-l), the controlled player steals the ball and passes the ball to another controlled player. These results demonstrate that our approach learns intricate control of the agents. Please see the supplementary material for videos.

Visual single-agent results. We also compare our results with those reported by Kurach *et al*. [15] in the single agent setting. As shown in Tab. IV, to achieve a goal difference of -0.4 our approach only needs 7.5M environment steps, whereas None+CNN from Kurach *et al*. [15] needs 100M environment steps. Additionally, we achieve a non-negative goal difference within 20M steps. In contrast, the baseline can hardly learn a meaningful policy within 20M environment steps. This demonstrates the data efficiency of semantic tracklets with GCN: we significantly reduce the number of environment steps to achieve the same reward.

Method	Goal difference						
	-2.0	-1.6	-1.2	-0.8	-0.4	0.0	0.1
None+CNN [15]	–	–	–	–	100	> 100	> 100
Tracklets	**0.2**	**0.4**	**1.8**	**4.0**	**7.5**	**15.7**	**16.0**

TABLE IV

NUMBER OF FRAMES, IN MILLIONS, REQUIRED TO ACHIEVE TARGET GOAL DIFFERENCE FOR *11* VS. *11* SINGLE AGENT SETTING.

VI. CONCLUSION

We study semantic tracklets, an object-centric representation for VMARL. Semantic tracklets permit use of graph-nets and enable efficient learning of policies from visual inputs on VMPE and GFootball. Notably, for the first time, effective policies are learned for five players in the challenging GFootball setting. Compared to prior work, *e.g.*, not using an intermediate representation or using segmentation, semantic tracklets are a compelling way to improve VMARL data efficiency and scalability.

REFERENCES

[1] S. Mariani, G. Cabri, and F. Zambonelli, "Coordination of autonomous vehicles: Taxonomy and survey," *arXiv.*, 2020.

[2] Y. Mohan and S. G. Ponnambalam, "An extensive review of research in swarm robotics," in *Proc. NaBIC*, 2009.

[3] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Front. Robot. AI*, 2020.

[4] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. NeurIPS*, 2017.

[5] E. Bargiacchi, T. Verstraeten, D. Roijers, A. Nowé, and H. Hasselt, "Learning to coordinate with coordination graphs in repeated single-stage multi-agent decision problems," in *Proc. ICML*, 2018.

[6] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," in *Proc. ICML*, 2018.

[7] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "MAVEN: multi-agent variational exploration," in *Proc. NeurIPS*, 2019.

[8] I.-J. Liu, R. A. Yeh, and A. G. Schwing, "PIC: permutation invariant critic for multi-agent deep reinforcement learning," in *Proc. CoRL*, 2019.

[9] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. ICML*, 2018.

[10] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. ICML*, 2018.

[11] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI*, 2018.

[12] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi, "Learning to schedule communication in multi-agent reinforcement learning," in *Proc. ICLR*, 2019.

[13] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in *Proc. ICML*, 2019.

[14] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *arXiv.*, 2017.

[15] K. Kurach, A. Raichuk, P. Stańczyk, M. Zajac, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, and S. Gelly, "Google research football: A novel reinforcement learning environment," *arXiv.*, 2019.

[16] U. Jain, L. Weihs, E. Kolve, M. Rastegari, S. Lazebnik, A. Farhadi, A. Schwing, and A. Kembhavi, "Two body problem: Collaborative visual task completion," in *Proc. CVPR*, 2019.

[17] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. AISTATS*, 2011.

[18] J. Jiang, C. Dun, and Z. Lu, "Graph convolutional reinforcement learning for multi-agent cooperation," in *arXiv.*, 2019.

[19] Z. Hong, Y. Chen, S. Su, T. Shann, Y. Chang, H. Yang, B. H. Ho, C. Tu, Y. Chang, T. Hsiao, H. Hsiao, S. Lai, and C. Lee, "Virtual-to-real: Learning to control in visual semantic segmentation," in *Proc. IJCAI*, 2018.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NeurIPS Deep Learning Workshop*, 2013.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[22] V. Mnih, Adrià, P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, 2016.

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

[24] I.-J. Liu, J. Peng, and A. G. Schwing, "Knowledge flow: Improve upon your teachers," in *Proc. ICLR*, 2019.

[25] Y. Li, I.-J. Liu, Y. Yuan, D. Chen, A. Schwing, and J. Huang, "Accelerating distributed reinforcement learning with in-switch computing," in *Proc. ISCA*, 2019.

[26] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson, "Object-centric forward modeling for model predictive control," in *Proc. ICRA*, 2019.

[27] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Proc. RSS*, 2018.

[28] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proc. CVPR*, 2019.

[29] C. Devin, P. Abbeel, T. Darrell, and S. Levine, "Deep object-centric representations for generalizable robot learning," in *Proc. ICRA*, 2018.

[30] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani, "Object-centric forward modeling for model predictive control," in *CoRL*, 2019.

[31] I.-J. Liu, R. A. Yeh, and A. G. Schwing, "High-throughput synchronous deep rl," in *Proc. NeurIPS*, 2020.

[32] U. Jain, I.-J. Liu, S. Lazebnik, A. Kembhavi, L. Weihs, and A. Schwing, "Gridtopix: Training embodied agents with minimal supervision," in *Proc. ICCV*, 2021.

[33] I.-J. Liu, U. Jain, R. A. Yeh, and A. G. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," in *Proc. ICML*, 2021.

[34] U. Jain, L. Weihs, E. Kolve, A. Farhadi, S. Lazebnik, A. Kembhavi, and A. G. Schwing, "A cordial sync: Going beyond marginal policies for multi-agent embodied tasks," in *Proc. ECCV*, 2020.

[35] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv.*, 2017.

[36] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, , and D. Batra, "Habitat: A platform for embodied AI research." in *Proc. ICCV*, 2019.

[37] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese, "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," in *Proc. ICRA*, 2020.

[38] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, 1955.

[39] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms," in *Proc. ICML*, 2018.

[40] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI*, 2017.

[41] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv.*, 2018.