sciendo

JAISCR, 2022, Vol. 12, No. 2, pp. 135 – 148
10.2478/jaiscr-2022-0009

HANDLING REALISTIC NOISE IN MULTI-AGENT SYSTEMS WITH SELF-SUPERVISED LEARNING AND CURIOSITY

Márton Szemenyei*, Patrik Reizinger

Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics,
1117, Budapest, Magyar Tudosok krt. 2.

**E-mail: szemenyei.marton@vik.bme.hu*

Submitted: 27th September 2021; Accepted: 18th December 2021

Abstract

[1]Most reinforcement learning benchmarks – especially in multi-agent tasks – do not go beyond observations with simple noise; nonetheless, real scenarios induce more elaborate vision pipeline failures: false sightings, misclassifications or occlusion. In this work, we propose a lightweight, 2D environment for robot soccer and autonomous driving that can emulate the above discrepancies. Besides establishing a benchmark for accessible multi-agent reinforcement learning research, our work addresses the challenges the simulator imposes. For handling realistic noise, we use self-supervised learning to enhance scene reconstruction and extend curiosity-driven learning to model longer horizons. Our extensive experiments show that the proposed methods achieve state-of-the-art performance, compared against actor-critic methods, ICM, and PPO.

Keywords: deep reinforcement learning, multi-agent environment, autonomous driving, robot soccer, self-supervised learning

1 Introduction

Reinforcement learning has undergone unprecedented evolution, matching or even surpassing human abilities in, e.g., computer games [31]. As benchmarks developed from a single-paper/single-task version to standardized suites with stable interfaces [2, 10], reinforcement learning applications skyrocketed. The environments also became more complex: simulators now often use 3D physics engines [1, 13].

The above trend focuses on idealized scenarios; it usually neglects modeling realistic failures, such as partial observations, hidden objects and classification errors. Although we accept that this trend benefits progress, real-world applications require more focus on robustness. When the agents can receive observations with realistic noise, they can adapt to such situations. We provide the means to do this by developing a simulator for robot soccer and urban autonomous driving with a focus on realistic observations and make the back-end lightweight by using a 2D physics engine and simulated object detection – ensuring fast simulations and i.i.d. data collection. This results in an accessible solution for robustness research.

[1]The research reported in this paper and carried out at BME has been supported by the NRDI Fund (TKP2020 NC, Grant No. BME-NC) based on the charter of bolster issued by the NRDI Office under the auspices of the Ministry for Innovation and Technology.

We also propose novel neural architectures for these environments to handle the additional complexity. An efficient attention mechanism captures temporal and inter-object relations, self-supervised learning [16, 24] incentivizes the agent's localization and state reconstruction capabilities against failures of the vision system, and curiosity-driven exploration [18, 3] constitutes our long-term/multi-step prediction strategy, which is our contribution extending short-horizon algorithms [18, 19].

Our contributions can be summarized as follows:

1. We develop highly efficient and accessible multi-agent environments to simulate the failures of processing pipelines.
2. We design a neural architecture to address observations with realistic noise.
3. We propose self-supervised approaches to extract information about other agents and dynamic objects.
4. We extend the paradigm of curiosity-driven intrinsic motivation for longer time horizons.

As part of the evaluation, we benchmark our simulator against actor-critic networks [15], Proximal Policy Optimization (PPO) [23], and the Intrinsic Curiosity Module (ICM) [18]. To reduce the effect of random seed selection, we evaluate all methods with ten different random seeds, eight parallel environments, and compare the rewards via statistical testing.

2 Related work

As we are interested in advancing multi-agent deep reinforcement learning systems, we review the relevant work that contributed to achieving (super)human-level performance in computer vision [21, 20] and reinforcement learning [15, 31, 18]. We also discuss self-supervised learning, providing a context for our proposal.

2.1 Reinforcement Learning

We focus on model-based algorithms as they open the door to handle credit assignment. Namely, an *intrinsic reward* can be formulated to guide the agent when the *external reward* is sparse.

This helps learning and even enables *self-play*, as in the World Models [9] paper. Learning a model also enables faster training (models can exploit GPUs, whereas simulators usually run on the CPU), a safer process (no interaction with the real world), and more intuitive debugging (as scientist can use the model to analyze the agent).

Another paradigm is *curiosity-driven learning* [18, 3], where the intrinsic reward incentivizes the agent to explore its environment and to model it better – the ensemble version is called *disagreement* [19]. A large-scale study [3] shows that curiosity enables learning without extrinsic rewards, but also improves performance if applied together.

We have extended curiosity-driven learning in [22] with the attention mechanism [30]: an adaptive reward, called Rational Curiosity Module (RCM), improved the performance in Atari [2] and multi-agent environments [28].

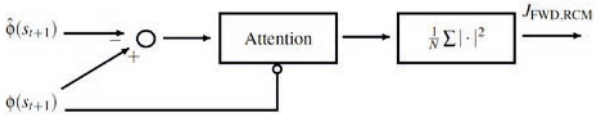

Figure 1. The RCM loss [22]. ϕ stands for features, $\hat{}$ for predictions, and the circle in the Attention block indicates conditioning

The authors argue that not all features are important at every time step: the RCM loss (Figure 1) uses attention to focus on a feature subspace; these are selected by conditioning (denoted by a circle) on the features of the next time step $\phi(s_{t+1})$.

We have shown the equivalence of RCM with adaptive weighted least squares [28], namely:

$$J_{\text{FWD,RCM}} = \frac{1}{N} e_{t+1}^T e_{t+1} \tag{1}$$

$$e_{t+1} = \sqrt{W}\left(\phi(s_{t+1}) - \hat{\phi}(s_{t+1})\right), \tag{2}$$

where W stands for the weights of the Attention network, as in [28].

Multi-agent tasks increase the complexity of the credit assignment problem; to learn the contribu-

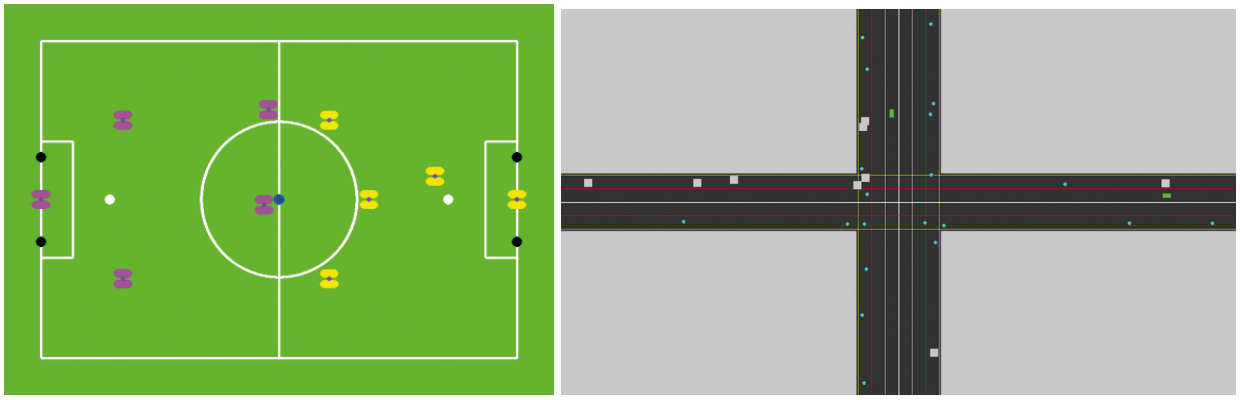

Figure 2. The RoboCup (left) and Driving (right) environments.

tions of each agent, Foerster et. al. [7] proposes a counterfactual approach.

2.2 Self-Supervised Learning

Self-supervised learning enables learning representations without labeling data: by creating intermediate tasks, labels are automatically generated. The paradigm's advantages include improved generalization, faster convergence, and a considerably smaller data set size.

Computer vision has revolutionized self-supervised learning: applications include distortion [6] or rotation [8] correction, image inpainting [4], and colorization [35]. Generative models (e.g., Autoencoders [36], World Models [9], and Bidirectional GANs [5]) also have a self-supervised interpretation.

For video, tracking objects [33], colorizing video [32], and predicting video frame order or the "flow of time" [34] were handled by self-supervised methods.

Reinforcement learning is no exception: curiosity forms sequence-based self-supervised learning, since it predicts the next state and the action leading to that state. Grasp2Vec [11] learns an object-centric representation using metric learning; Time Contrastive Networks [25] learn to embed frames in multi-view videos, enforcing similarity of different perspectives' representations; Imagined Goals [17] explore the environment early on by defining additional goals.

3 Environments

In this section, we describe the two cooperative-competitive environments: *1) RoboCup* for robot soccer and *2) Driving* for autonomous vehicles in an urban scenario, introduced in [28]).

We decided to deviate from the direction of the literature: instead of 3D physics simulation [10, 13] or complicated tasks (such as StarCraft II [31]), our focus was modeling realistic noise and the failures of vision pipelines (e.g. false positive and false negative sightings).

Our contribution is presented in the form of two OpenAI Gym-compatible [2] tasks (Figure 2), where we abstract away from realistic graphics and low-level physics (such as robot joint control), ensuring fast experimentation. We emphasize that our solution is among the few for multi-agent systems. While Gym-compatible simulators exist for robot soccer [14], these provide rendering and focus on low-level actions, while our environment is aimed at learning high-level strategies.

Both environments offer high-level action spaces; For the driving environment, the action space consists of two discrete actions $[gas/break,\ left/right]$, while the robot soccer environment offers four possible discrete actions: $[step,\ turn,\ kick,\ turn\ head]$. In the following, we elaborate the description of the observation and reward systems.

3.1 Observations

To maintain efficiency, the environment provides abstract observations that an object detection algorithm would extract; failure emulation is

still possible via *partial observations*, meaning limited object visibility; constraining factors are agent-object distance and field of view (the latter only for robot soccer).

In partial observation mode, the environment provides a list of seen objects in the given agent's field of view. Each object observation includes the objects location relative to the agent, its orientation, and other object-specific properties that can be determined visually (such as the team of a robot, or whether it's fallen).

The robot soccer environment can optionally emulate semantic segmentation by image-based observations (i.e., generating semantically labeled images for both robot cameras). Although the detected objects are in the agent's coordinate system, but localization should happen in absolute coordinates, the task is non-trivial.

Both observation modes support failure emulation including false negatives, false positives, misclassifications, and inaccurate localization. Objects may also be lost due to occlusion. This functionality supports two modes: *1) random mode:* errors occur with a fixed, adjustable probability, *2) realistic mode:* the probability of random errors depends on observation reliability:

1. The magnitude of position noise and the probability of false negatives increases with the distance from the agent.
2. False negatives are more likely when two or more objects are nearby or at the edge of the field of view.
3. Visually similar objects (such as balls and penalty crosses or vehicles and obstacles) are more likely to be confused than dissimilar ones.
4. False-positive sightings are more likely in certain places (such as false-positive balls at the feet of other nearby robots).

The above list was assembled based on our previous experiences in neural network-based object detection in similar environments [26, 29, 27].

3.2 Reward system

We define three reward categories: *1) primary, 2) rule-based, and 3) auxiliary*.

Primary rewards are scored for succeeding (scoring in *RoboCup*, or reaching the destination in *Driving*).

Rule-based rewards make the agents avoid breaking rules with negative scores: pushing each other or walking off the field are erratic in robot soccer, whereas crashing or leaving the road are penalized in autonomous driving.

Auxiliary rewards incentivize actions leading to long-term success. As such moves are usually not optimal in the short-term, these rewards are smaller than the primary or rule-based ones. When a robot moves towards the ball, or kicks it towards the opponent's goal, it gets an auxiliary reward. This type can also be negative: if a car drives in the opposite lane, it gets penalized.

Observation rewards are a special case of auxiliary rewards for robot soccer: robots are incentivized to observe several *different* objects in a given time interval (leading to moving the robot's head). Detecting the ball is worth the most, followed by robots and field objects.

4 Model

This section introduces the model architecture we designed for both robot soccer and autonomous driving. Due to diverse failure modes (false positives, false negatives, misclassifications), we could not utilize the results from the literature – the few environments providing agent-centric partial observations only account for occlusion [1].

Our additional goals also underlined a new agent design: to handle a variable amount of detections, we designed a novel encoder (subsection 4.1); to capture long-term dependencies, we deployed the attention mechanism (subsection 4.2); to make learning more efficient, we applied self-supervised learning (subsection 4.3). At the end of this section, we provide an overview of the architecture (Figure 6) and the training algorithm (Algorithm 1).

The *RoboCup* environment imposes a further constraint: as the Nao robots' control is relatively slow (a single action takes $500 - 1000$ ms), reinforcement learning algorithms need to adapt to this time delay. In practice, this means that after taking a single action, multiple observations are received –

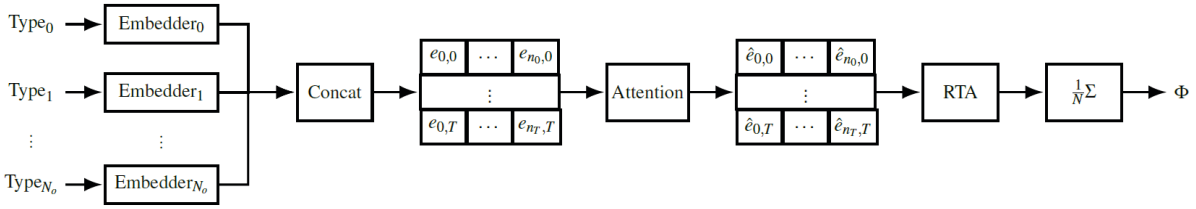

Figure 3. The encoder architecture. Here $\hat{\cdot}$ is denotes that attention was applied to the embedded objects.

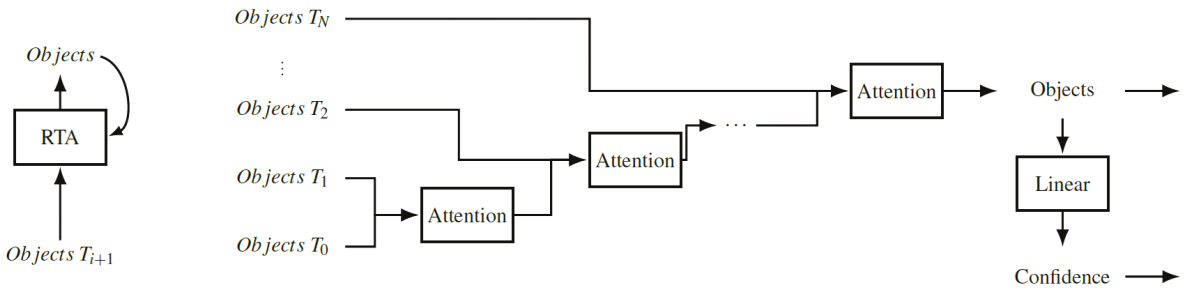

Figure 4. The Recurrent Temporal Attention (RTA) module (left: recurrent form, right: rolled out in time).

as the vision pipeline is considerably faster.

4.1 Encoder

The encoder (Figure 3) needs to learn a compact representation from corrupted observations, while also accounting for both *intratemporal* (within a time step) and *intertemporal* (between time steps) object interactions.

The first step is to embed each object with a small Multi-Layer Perceptron (MLP); each object type has its embedder network with same-sized outputs. Second, a self-attention layer (with weight sharing) consumes the concatenated features to predict intratemporal dependencies – the presence or relative position of objects can establish occlusion or other failures.

Third, to learn intertemporal relationships, we propose Recurrent Temporal Attention (RTA), an Recurrent Neural Network (RNN) cell with multi-head attention with sequences as observations. The key point is that the output is fed back to the RTA (Figure 4).

In practice, we need to pad all the input sequences to match the longest input to ensure that no detections are lost due to false negatives. This is because the length of the output sequence of the attention module is the same as the length of the target input sequence, so false negatives in some inputs can force the sequence to decrease in length.

However, false positives can still remain in the sequence, since there is no mechanism to remove them. We address this by including a linear layer with sigmoid activation – a common practice [21, 20] for computing confidence scores. RTA also uses bias vectors to ensure numerical stability (in the softmax) when no objects are detected.

Finally, we compute a confidence-weighted average of the objects, resulting in a single embedding vector.

4.2 Long-term forward prediction

One of our key contributions is that we extend curiosity-driven reinforcement learning for longer time horizons (up to the rollout length). Other intrinsic rewards focus on short-horizons [18, 19] – the World Models architecture [9] contains an RNN, but - in our assessment - this differs from our goal.

For brevity, we denote $\phi(s_t)$ as ϕ_t in this section. Using step-wise predictions of $p\left(\hat{\phi}_{t+i}|\phi_{t+i-1}, a_{t+i-1}\right)$ would be equivalent to applying the ICM [18] multiple times, assuming Markovian state dynamics.

However, the proposed method reuses predic-

tions of $\hat{\phi}_{t+i}$ for future feature distributions (note the hat symbol denoting predicted values right to the conditioning bar)

$$p\left(\hat{\phi}_{t+i}|\phi_t, \hat{\phi}_{t+1}, \dots \hat{\phi}_{t+i-1}, a_{t+i-1}\right). \tag{3}$$

The advantage of this approach is that the agent is forced to model longer dependencies, but this naive formulation comes at a cost: feeding back predictions accumulates the error, leading to potentially disastrous performance.

Our solution is to adapt the loss formulation (Figure 5): we use the squared loss as in [18], but we make it adaptive with the attention mechanism as in [22]. This way, the network is able to provide an intrinsic reward for longer horizons, while compensating for the accumulating prediction error.

We calculate each $\hat{\phi}_{t+i}$ with MLPs so we do not suffer from slower RNNs, then the squared losses W_i are weighted based on the previous weights – our conjecture is that by identifying the "error characteristics" of the previous step, the predictions will be more reliable. Note that although feature prediction is not sequential, calculating the weighting factors is.

The modified loss can be formulated as follows:

$$J_{\text{LTP}} = \frac{1}{N}\sum_{i=t+1}^{t+N+1} J_{\text{LTP},i} \tag{4}$$

$$J_{\text{LTP},i} = \text{Attention}\left(W_i\right) \odot \left[\phi_i - \hat{\phi}_i\right]^2 \tag{5}$$

$$W_i = \begin{cases} \left[\phi_{i-1} - \hat{\phi}_{i-1}\right]^2, & i > t+1 \\ 1, & i = t+1, \end{cases} \tag{6}$$

where LTP is a shorthand for Long-Term Prediction, and W_i has the same dimensionality as ϕ_t ($\odot$ denotes the Hadamard-product). As the input for the first prediction is the ground-truth state, it is not weighted.

4.3 Localization and reconstruction

The agents observe their environment in their own coordinates, but successful learning requires global localization (e.g. to not to score own goals). Thus, we extend the learning objective with *localization* and *reconstruction* objectives.

Localization deploys self-supervised learning for predicting position and orientation in an absolute sense (for robots with rotating heads this means learning two orientations). Reconstruction aims for learning about other dynamic objects seen during the rollout: the reconstruction of the state vector extracts absolute information and reduces position noise – thus, accounting for false negatives and positives.

These two tasks are formulated as an object detection problem: given inaccurate and faulty observations, the agents predict the position, orientation and other relevant properties of themselves and dynamic objects. We use anchors, which are fixed to the field, to predict multiple objects of the same type – as in YOLO [21, 20]. To minimize the distance between objects and anchors, we selected the Hungarian algorithm.

The localization and reconstruction heads are trained in tandem with other parts of the architecture, using the ground truth data retrieved from the environment. This amounts to a form of self-supervised learning, where the agent can learn useful representations of the input by learning to localize and reconstruct. Notably, these heads can also be pre-trained to speed up the convergence using randomly generated data.

Algorithm 1. Training procedure for the proposed model using RCM, Reconstruction and Long-Term Prediction (LTP)

Result: Model losses
$s_0, obs_0 \leftarrow env.reset()$;
while *Episode running* **do**
  while *Rollout number not reached* **do**
    $\Phi_t \leftarrow encoder(obs_t)$;
    $a_t \leftarrow actor(\Phi_t)$;
    $Q_t \leftarrow critic(\Phi_t)$;
    $\hat{a}_{t-1}, \hat{\Phi}_{t+1} \leftarrow RCM(\Phi_t, \Phi_{t-1}, a_t)$;
    $\hat{\Phi}_{t+2}, \dots, \hat{\Phi}_{t+N} \leftarrow predictor(\Phi_t)$;
    $\hat{s}_t \leftarrow reconstruction(\Phi_t)$;
    $s_{t+1}, obs_{t+1}, r_t \leftarrow env.step(a_t)$;
  end
  $\Phi_{t+1} \leftarrow encoder(obs_{t+1})$;
  $L_p \leftarrow Policy\_loss(r, a, Q)$;
  $L_v \leftarrow Value\_loss(r, Q)$;
  $L_{RCM} \leftarrow RCM\_loss(\Phi, \hat{\Phi}, a, \hat{a})$;
  $L_{pred} \leftarrow Prediction\_loss(\Phi, \hat{\Phi})$;
  $L_{recon} \leftarrow Reconstruction\_loss(s, \hat{s})$;
end

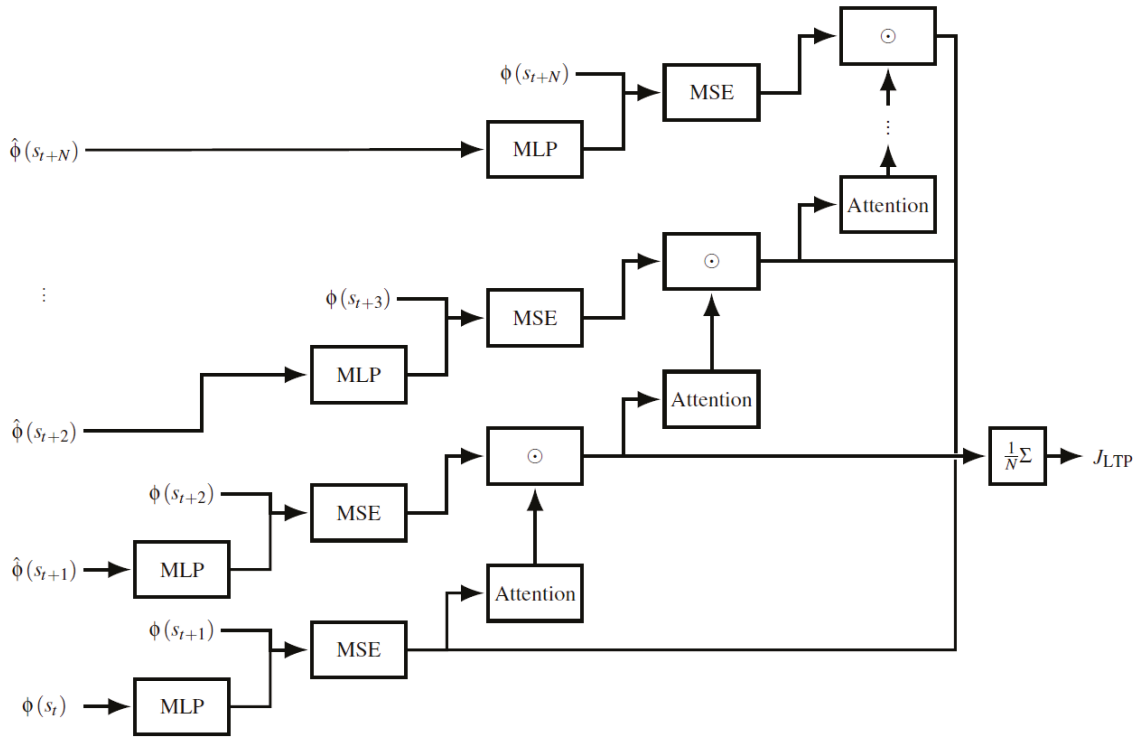

Figure 5. The Long-term forward prediction module. The output of each MLP module is $\hat{\phi}_{i+1}$, given $\hat{\phi}_i$ (or $\phi(s_i)$) as input. The $\odot$ denotes the Hadamard-product.

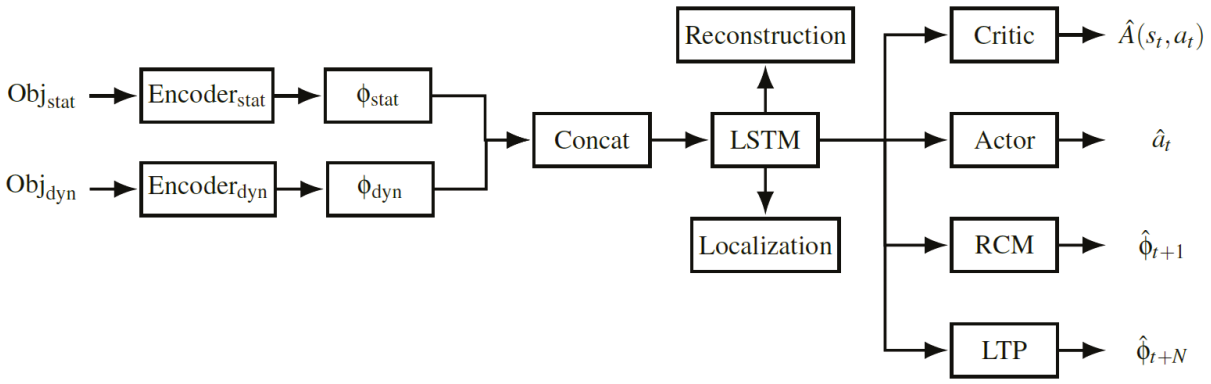

Figure 6. The network architecture. The subscripts stat and dyn denote components (objects, encoder, and features) for static and dynamic objects, respectively.$\phi(s_t)$ stands for features, a_t for actions, while A for the advantage function; whereas $\hat{}$ denotes predicted values.

5 Results

After discussing the training procedure (subsection 5.1), we report the performance of the proposed methods, benchmarked against actor-critic [15], ICM [18], and PPO [23] (subsection 5.4). Besides the merits of LTP (subsection 5.2) and reconstruction (subsection 5.3), we also detail the computational cost of the simulator (subsection 5.5) – supporting the claim of an accessible tool.

5.1 Training

The agent of section 4 was compared against A2C, ICM, PPO, and combinations thereof. We used the partial observation mode with realistic noise (of magnitude 1.0), eight parallel environments, two agents for Driving, and two agents *in each team* for RoboCup. The best average reward (within a ten-episode moving window) served as our performance metric; additionally, the best average observation reward (specific to RoboCup, see subsection 3.2) was calculated.

The baseline agent was an A2C [15] architecture extended with the RCM module (denoted by **RCM**), while 'our' method included the LTP extension as well (**Pred**). We report the performance when combining LTP and reconstruction (**Recon**) – as Driving did not benefit from reconstruction due to the 360° vision of vehicles and the irrelevance of global localization, we benchmarked the **Recon** model only in RoboCup.

For reducing variance, we averaged each configuration over eight seeds and carried out a Bayesian t-test to report the results' significance – we opted for the independent samples version from BEST [12], since having the same seed does not necessarily make two runs comparable. We report the 95% Confidence Interval (CI) on the difference of means, the mean of the CI, and the Probability of Improvement (PoI) – PoI describes the probability of a positive effect size, i.e. the proposal's superiority.

A clear limitation of our work is the lack of sophisticated hyperoptimization: the extensive comparisons and the limited hardware resources constrained us. Nonetheless, we manually tuned the hyperparameters (Table 1) on the same environments, but on different tasks – their purpose was to show convergence while avoiding information "leaks" about the real task.

Table 1. Training hyperparameters

Hyperparameter	**Value**
Learning rate	10^{-4}
Max gradient norm	0.5
Number of updates	60,000
Number of steps	240,000
Rollout size	6
ICM loss coeff.	10^{-2}
Value loss coeff.	0.5
Entropy loss coeff.	0.1
Forward loss coeff.	10^{-2}
Long-term Prediction loss coeff.	10^{-3}
Reconstruction loss coeff.	10^{-2}

5.2 Effects of LTP

We compared LTP in three scenarios: Driving using partial and RoboCup with both full and partial observations. For these tests, the same architecture with LTP turned off was used as baseline.

Table 2. Improvements in the mean reward by using LTP. ΔR is the expected improvement of the mean reward, while the Confidence Intervals (CI) and the Probability of Improvement (PoI) are also given. R_b is the mean reward of the baseline model.

Env	ΔR	95% **CI**	**PoI**	R_b
Robo	0.18	[0.01, 0.35]	97.4%	13.06
Robo (Full)	0.47	[0.14, 0.76]	99.7%	19.28
Driving	3.51	[0.89, 5.89]	99.55%	22.96

We can conclude that LTP significantly outperformed the baseline (Table 2) in all cases: the effect in RoboCup was small, whereas in Driving it was 15%. Although the effect size is small, the effect of LTP on the run time of the training is not notable ($< 1\%$), as the training process is highly CPU-bound by the environment.

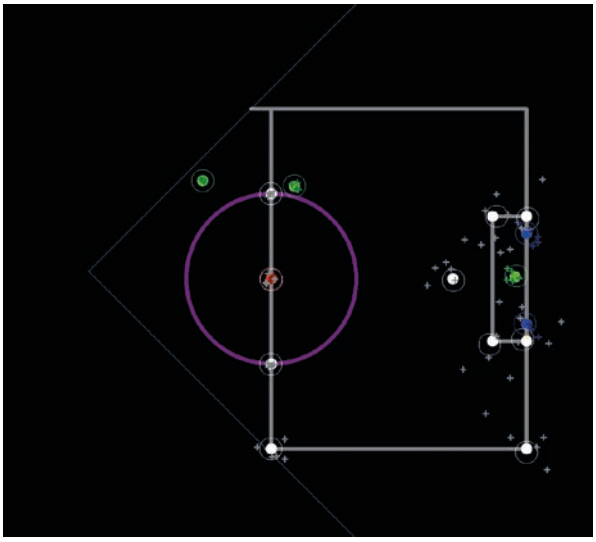

Figure 7. Long-Term Prediction (LTP) in RoboCup: filled circles represent ground truth, crosses represent observations, and the empty circles stand for the LTP predictions

Independent from the effect size, (LTP) had another advantage: it provided stable means for the agent to reconstruct the scene from noisy observations (Figure 5).

5.3 Effects of localization and reconstruction

The **Recon** agent was pre-trained with random, rollout-sized sequences – as pre-training can improve convergence times significantly and the data is easy to generate. We measured the success of pre-training with the Average Precision (AP) score at different distance thresholds (Table 3); to put the accuracy in perspective, the soccer field is 9×6 meters.

Table 3. The Average Precision (AP) achieved for localization and reconstruction.

Distance threshold	25 cm	50 cm	1 m
Localization	83%	100%	100%
Reconstruction	54%	70%	92%

Localization clearly outperformed reconstruction: as reconstruction (dynamic objects' absolute position) depends on localization (agents' orientation and position), a small error in the latter can considerably deteriorate the performance of the former.

We measured the performance of the localization and reconstruction modules (**Recon**) in RoboCup with partial observations (Table 4). As the model includes LTP, we compared against the baseline (**RCM**) and its extended version (including LTP, denoted as **Pred**) to account for the potential improvements introduced by long-term forward prediction.

Table 4. Results of the Bayesian t-test.

Comparison	95% **CI**	**Mean**	**PoI**	R_b
RCM vs **Recon**	$[2.22, 2.6]$	2.41	100%	13.06
Pred vs **Recon**	$[2.01, 2.44]$	2.23	100%	13.25

The conclusion is clear: **Recon** significantly outperformed both **RCM** and **Pred**. Indeed, the difference between both baselines was negligible compared to the 19% improvement provided by **Recon**.

The same comparisons with the observation reward (Table 5) highlighted how important localizing relevant objects is, resulting in a significant gain of 8%.

Table 5. Results of the Bayesian t-test for the observation rewards.

Comparison	95% **CI**	**Mean**	**PoI**	R_b
RCM vs **Recon**	$[0.36, 0.58]$	0.47	100%	5.94
Pred vs **Recon**	$[0.35, 0.58]$	0.46	100%	5.95

The improvements are undoubtedly promising, but there is still a long way to go: although our proposal beat the random policy, human players (the authors) easily outperformed the agent (Table 6). Finally, training curves are presented in Figure 8.

Table 6. Comparison against the random and the human baselines.

Baseline	**Random**	**Human**	**Pred**	**Recon**
RoboCup	5.75	22.57	13.25	15.48
Driving	0.7	42.3	26.47	N/A

5.4 Ablation study with PPO

This section discusses the result of our ablation study, where we used PPO.

Using curiosity-driven exploration had negligible – in some cases negative – effects: a clear con-

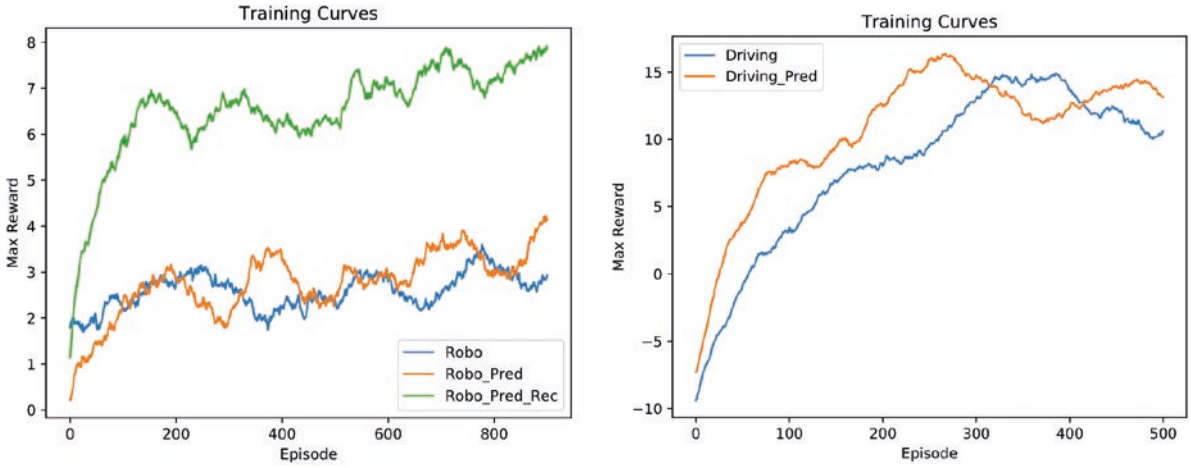

Figure 8. The RoboCup (left) and Driving (right) training curves. The y axis displays the maximal reward an agent achieved in a particular episode normalized, so that starting rewards would be zero.

sequence of incentivizing exploration and reporting the *extrinsic* reward.

For Driving, RCM and the LTP (Table 7) significantly improved the performance; nonetheless, the effect size was relatively minor.

In RoboCup (Table 8), each combination improved the reward, except when only curiosity-driven exploration was used. Notably, the use of **Recon** meant the largest effect size.

5.5 Training times

We report the execution times of our multi-agent simulator to show that we created a lightweight benchmark suite. Running experiments in both environments is possible even with modest resources, as our test configuration shows: a single Titan Xp GPU, an i7-9700K CPU, and 32 GB RAM were sufficient to achieve reasonable run times (Table 9)–a single training used eight parallel environments with four agents.

Table 9. Time to train agents with 4 agents per environment, 8 environments in parallel

Env	# of runs	1 episode	100k updates
Robo	1	29 s	20 h
Robo	4	67 s	47 h
Driving	1	17 s	5 h
Driving	4	35 s	10 h

An experiment with 100,000 updates took less than one day, and running four times as many parallel experiments increased the run time by only $100-145\%$; thus, fulfilling our aim of an efficient simulator.

6 Conclusion

The main results of this paper are twofold: we introduced a multi-agent simulator with novel functionality, and we developed a reinforcement learning method with novel self-supervised and intrinsic motivation solutions.

Our *DynEnv* simulator stands out by providing lightweight means for multi-agent research; a single GPU suffices for running extensive experiments. Moreover, it is capable of generating imperfect observations beyond including noise: the options range from false positives and negatives to misclassifications.

More challenging observations require a more elaborate model: we address imperfect observations by proposing self-supervised extensions for scene reconstruction. Modelling long-term dependencies was also crucial to improve performance; thus, our contribution extends the curiosity paradigm for long-term strategies via the Long-Term Prediction (LTP) algorithm. The proposed methods are instrumental in surpassing state-of-the-art methods – including ICM [18] or PPO [23].

Table 7. Component ablation in the Driving environment for PPO. ■ stands for baseline, while ✓ for additional components in the comparison. Note that RCM is an extension of ICM.

PPO	ICM	RCM	Pred	95% CI	Mean	PoI
■	✓			$[-2.77, -0.29]$	-1.59	71.25%
■		✓		$[-0.38, 2.01]$	0.76	88.44%
■		✓	✓	$[0.48, 2.56]$	1.50	99.69%
■	■	✓		$[1.09, 3.86]$	2.45	99.94%
■	■	✓	✓	$[1.92, 4.36]$	3.10	100%
■		■	✓	$[-0.36, 1.87]$	0.80	91.56%

Table 8. Component ablation in the RoboCup environment for PPO. ■ stands for baseline, while ✓ for additional components in the comparison. Note that RCM is an extension of ICM.

PPO	ICM	RCM	Pred	Recon	95% CI	Mean	PoI
■	✓				$[-2.70, -0.89]$	-1.79	0.04%
■		✓			$[-4.17, -2.27]$	-3.24	0%
■		✓	✓		$[-0.54, 0.96]$	0.19	67.34%
■		✓	✓	✓	$[14.02, 15.73]$	14.90	100%
■	■	✓			$[-2.41, -0.12]$	-1.23	2%
■	■	✓	✓		$[1.12, 2.87]$	1.98	100%
■		■	✓		$[2.53, 4.15]$	3.36	100%
■	■	✓	✓	✓	$[15.74, 17.70]$	16.70	100%
■		■	✓	✓	$[16.58, 18.47]$	17.53	100%

Although our experiments show clear improvement, the comparison with human baselines also emphasizes the need for further investigation if we want to achieve superhuman performance. We hope that our simulator becomes a basis for developing new algorithms for multi-agent scenarios.

To make our work reproducible, the environments are available as a Python package *(DynEnv)*, while the source is at `https://github.com/szemenyeim/DynEnv`.

References

[1] Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, 2020.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. 6 2016.

[3] Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019.

[4] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. May 2015.

[5] Jeff Donahue, Philipp Krahenbahl, and Trevor Darrell. Adversarial feature learning. May 2016.

[6] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. 6 2014.

[7] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 2974–2982. AAAI Press, 2018.

[8] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. March 2018.

[9] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 2455–2467, 2018.

[10] Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A research framework for distributed reinforcement learning. 6 2020.

[11] Eric Jang, Coline Devin, Vincent Vanhoucke, and Sergey Levine. Grasp2vec: Learning object representations from self-supervised grasping. Proceedings of The 2nd Conference on Robot Learning, in PMLR 87:99-112 (2018), November 2018.

[12] John K. Kruschke. Bayesian estimation supersedes the t test. Journal of Experimental Psychology: General, 142(2):573–603, 2013.

[13] Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. Emergent coordination through competition. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019.

[14] Felipe B. Martins, Mateus G. Machado, Hansenclever F. Bassani, Pedro H. M. Braga, and Edna S. Barros. rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. 6 2021.

[15] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, volume 48 of JMLR Workshop and Conference Proceedings, pages 1928–1937. JMLR.org, 2016.

[16] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018,

NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 9209–9220, 2018.

[17] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. July 2018.

[18] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of Proceedings of Machine Learning Research, pages 2778–2787. PMLR, 2017.

[19] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 5062–5071. PMLR, 2019.

[20] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 779–788. IEEE Computer Society, 2016.

[21] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 6517–6525. IEEE Computer Society, 2017.

[22] Patrik Reizinger and Márton Szemenyei. Attention-based curiosity-driven exploration in deep reinforcement learning. In 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020, pages 3542–3546. IEEE, 2020.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[24] P. Sermanet, C. Lynch, J. Hsu, and S. Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. In 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 486–487, 2017.

[25] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. 4 2017.

[26] Marton Szemenyei and Vladimir Estivill-Castro. Real-time scene understanding using deep neural networks for RoboCup SPL. In RoboCup 2018: Robot World Cup XXII, pages 96–108. Springer International Publishing, 2019.

[27] Márton Szemenyei and Vladimir Estivill-Castro. Fully neural object detection solutions for robot soccer. Neural Computing and Applications, 4 2021.

[28] Marton Szemenyei and Patrik Reizinger. Attention-based curiosity in multi-agent reinforcement learning environments. In 2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO). IEEE, 5 2019.

[29] Marton Szemenyei and Vladimir Estivill-Castro. ROBO: Robust, fully neural object detection for robot soccer. In RoboCup 2019: Robot World Cup XXIII, pages 309–322. Springer International Publishing, 2019.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008, 2017.

[31] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782):350–354, 2019.

[32] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. 6 2018.

[33] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. May 2015.

[34] Donglai Wei, Joseph Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, 6 2018.

[35] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. March 2016.

[36] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. November 2016.

Márton Szemenyei received a Ph.D. degree in the Discipline of Informatics at the Budapest University of Technology and Economics in 2021. Currently, he is an assistant professor at the Department of Control Engineering and Information Technology at the same university. His main research topics include computer vision, deep learning and reinforcement learning. He has also published several papers on the application of information technology to environmental problems.

Patrik Reizinger has received the B.Sc. and M.Sc. degrees in electrical and electronics engineering in 2019 and 2021 at the Budapest University of Technology and Economics (BUTE), Hungary, and is currently pursuing the Ph.D. degree at the University of Tübingen. His research interests include artificial intelligence for computer vision and control, reinforcement learning, and causal discovery.