View-Action Representation Learning for Active First-Person Vision

Changjae Oh and Andrea Cavallaro

***Abstract*—In visual navigation, a moving agent equipped with a camera is traditionally controlled by an input action and the estimation of the features from a sensory state (i.e. the camera view) is treated as a pre-processing step to perform high-level vision tasks. In this paper, we present a representation learning approach that, instead, considers both state and action as inputs. We condition the encoded feature from the state transition network on the action that changes the view of the camera, thus describing the scene more effectively. Specifically, we introduce an action representation module that generates decoded higher dimensional representations from an input action to increase the representational power. We then fuse the output from the action representation module with the intermediate response of the state transition network that predicts the future state. To enhance the discrimination capability among predictions from different input actions, we further introduce triplet ranking loss and N-tuplet loss functions, which in turn can be integrated with the regression loss. We demonstrate the proposed representation learning approach in reinforcement and imitation learning-based mapless navigation tasks, where the camera agent learns to navigate only through the view of the camera and the performed action, without external information.**

***Index Terms*—Representation learning, triplet ranking loss, N-tuplet loss, mapless navigation.**

I. Introduction

VISUAL navigation generates through specific actions new input data by changing or selecting views in order to perform, for example, object detection [1], visual categorisation [2], [3], or image enhancement [4], [5], [6]. Visual navigation with a camera-equipped agent has been actively investigated for data collection [5], [6], [7], manipulation [8], and autonomous driving [9], [10].

Visual navigation can be map-based and mapless. *Map-based navigation* splits the task into two sub-tasks, namely the reconstruction of the geometry of the environment through navigable space and obstacles, and the subsequent path planning to enable navigation [11], [12], [13]. The geometry can be recovered through structure from motion or simultaneous localisation and mapping (SLAM) approaches [14], which require accurate mapping of obstacles and parameter adjustments in different environments. With *mapless navigation* the camera agent learns to navigate without reconstructing an explicit map of the environment. Instead of using explicit external information such as pre-defined features, a topological map, or tracking, the agent implicitly learns from the environment how to navigate to a goal. Deep neural networks can be used to learn to navigate using various strategies, such as feed-forward model [15], reinforcement learning (RL) [7], [16], [17], [18], [19], [20], or imitation learning (IL) [21], [22]. In this case, the control heavily depends on the representations of the input data: the camera agent is trained to perform actions to reach, from the current camera view (state), the final goal state. The learned representations play here an important role and should encode information that is useful for efficient navigation [23].

While learning methods to control the camera agent has been thoroughly investigated [24], there is limited understanding on how to design an efficient neural network-based architecture for the representation from state-action pairs. Generic Convolutional Neural Network (CNN) architectures, whose feature representations are simply generated from an input image only, are usually employed but they do not consider the joint effect with an input action [7], [16], [17].

To address this limitation, we investigate a deep neural network that employs an input state-action (view-action) pair to generate effective representations for mapless navigation, and validate the proposed network with RL [7], [25] and IL [26]. We introduce a forward model, that learns a representation which encodes an input image (state) and an action into CNNs, to predict the future state from the current state-action pair. In particular, we present an action representation module for efficient representation learning. This module expands the dimensions of an input action to improve the representational power of the network during training. We use joint regression and N-tuplet loss functions to predict the future state from a selected input action (regression loss) while discriminating predictions from different actions (N-tuplet loss) to encode meaningful features effectively during training. This paper substantially extends our previous work [20] with (i) joint regression and N-tuplet loss functions that generalise the joint regression and triplet ranking loss functions; (ii) variants of fusion approaches that combine the action representation module and the state-transition network; and (iii) qualitative and quantitative comparisons under RL and IL-based mapless navigation tasks.

II. Related Work

A. Visual Navigation with and without External Information

Traditional *map-based visual navigation* methods use explicit information of the environment such as a global map

Changjae Oh and Andrea Cavallaro are with the Centre for Intelligent Sensing, Queen Mary University of London, E1 4NS, London, United Kingdom. c.oh@qmul.ac.uk; a.cavallaro@qmul.ac.uk

This work was supported by the EPSRC Project NCNR (EP/R02572X/1) and by the CHIST-ERA programme through the project CORSMAL, under EPSRC grant EP/S031715/1.

or a known target position [27]. A camera agent can navigate towards a known target position using obstacle avoidance [28]. Navigation can also be performed based on the optimal planned path to reach the goal [29], [30]. Alternatively, navigation can be performed by reconstructing a map during exploration. Topological mapping estimates the global map from image collections obtained from a camera agent. Feature matching is generally performed between image pairs to obtain correspondences that can be used to estimate relative geometric information [31], [32]. Recently, map-based approaches have exploited deep neural networks to capture the environment using a spatial memory and then planning paths given partial information [33].

Mapless navigation employs the representational power of deep neural networks to learn a model that implicitly encodes obstacles and path to achieve the goal. Learning feed-forward CNN is one approach that trains the model with a collected supervised dataset [15]. The model learns to classify the optimal direction at each location to reach the target. Mapless navigation requires learning relationships between actions and the environment using for example RL and IL. An RL-based approach implicitly learns a policy to generate a decision to move towards the goal by trial-and-error. The model explores the environment and learns a policy by receiving an *extrinsic reward* signal when the agent achieves the goal. A brute-force approach to solve this problem is to learn a policy that depends on the current state [25] or one with the goal state [16]. However, the agent hardly receives the extrinsic reward as the final goal is generally far away from the initial state (sparse extrinsic rewards problem). To address this problem, *intrinsic rewards* reshape the original reward function to encourage the agent to explore the environment by seeking unseen areas rather than previously visited ones. A visitation count approach maximises the number of reaching less-frequently visited states [34], [35], [36]. An alternative is to maximise the information gain for the agent and to reduce the uncertainty about the environment [37], [38]. Curiosity-driven exploration measures the error between predicted and real future states: a high error is interpreted as an unseen area that cannot be predicted accurately, thus enabling to be exploited as the intrinsic reward [7], [39], [40], [41].

Another issue is the investigation of the network architecture for handling an action or language-based instruction beyond an input image. Since the image and another input are used to control the camera agent, designing an effective fusion model is an important problem [20], [42], [43]. An early fusion approach for goal directed navigation fuses the goal information with the input state followed by a convolution process to generate the feature for navigation [42]. Alternatively, a gated attention architecture fuses language-based instruction with the input state [43]. Moreover, an action representation module can be used to increase the dimension of the input action represented as a one-hot code to increase the representation power of the network [20].

B. Representation Learning for Camera Control

Representations to control a camera agent can be learned with forward or inverse dynamics models [24].

A *forward model* commonly learns temporal dynamics where the future state is predicted from the current state and action. This approach can be regarded as predicting video sequences. The general idea of the forward model is to encode information that is helpful to predict the future state using the current state and action. The input image is mapped to a high-level feature that encodes the future state, and then an auto-encoder reconstructs the future state on the image space [34], [44], [45], [46]. Examples include an action conditional auto-encoder that predicts the next state of a game [34]; an auto-encoder that generates a learned low-dimensional embedding of images that enables control in a locally linear latent space [44]; a variational auto-encoder that generates the representation from a sensor input image to manipulate an object [45]; and an auto-encoder with a skip connection that predicts the future state for the network to encode information that is useful to perform a target task [46].

An *inverse model* learns to predict the action that should be performed to move from the current to the future state [47]. The inverse model is generally incorporated with a forward model: the inverse model supervises the forward model to encode information that helps determine the action, providing a stabilising effect on the dynamics model [7], [20], [48]. The inverse model classifies the action that lies on a lower dimensional space and can, therefore, help overcome the problem of predicting the future state from the forward model. For example, forward and inverse models can be used to learn how to move objects to target locations by poking [48] or to visually explore simulation environments [7], [20].

III. View-Action Representation Learning

A. Problem Description

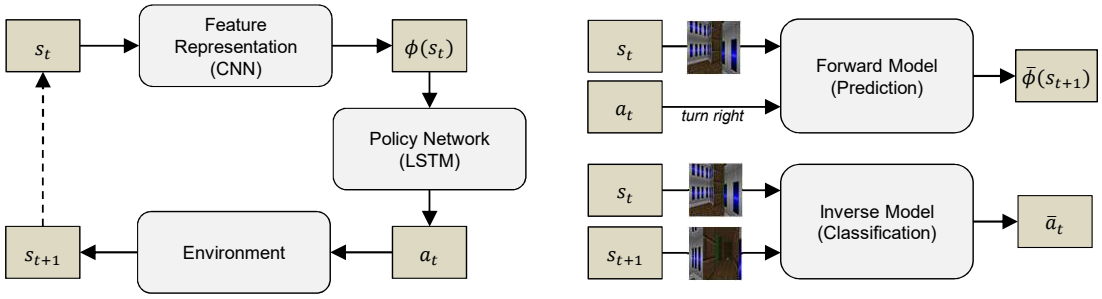
Let a moving camera agent explore the environment over some discrete time steps. At each time step t, the agent receives a state, $s_t \in \mathbb{R}^{H \times W}$, the image of $H \times W$ size from its first-person view, and selects an action, $a_t \in \mathbb{R}^N$, with policy, $\pi : s_t \rightarrow a_t$, from a set of possible N actions. The action, a_t, is commonly represented as a one-hot N-dimensional code.[1] After executing the action, the agent receives the next state, s_{t+1}.

We aim to produce a D–dimensional representation, $\phi(s_t) \in \mathbb{R}^D$, of the state, s_t, for efficient first-person-vision mapless navigation, which can be employed to the policy, π, rather than naïvely using a raw input, s_t. During representation learning by exploring the environment, the agent may receive a reward, r_t, which supervises the camera agent to reach the target. The reward is an optional value to the representation learning for visual navigation, which can be employed as direct supervision to the final target (extrinsic reward) or as a self-supervision for the model to encode meaningful information for achieving the final target (intrinsic reward).

B. Forward Model

Given a state-action pair, (s_t, a_t), at time t, the forward model, $f(\cdot)$, predicts s_{t+1} as a high-level feature representa-

[1]These codes describe categorical data as numerical ones, e.g. (1,0,0), (0,1,0), (0,0,1)



(a) (b)

Fig. 1. (a) Mapless visual navigation of a camera agent that generates action a_t with an encoded feature, $\phi(s_t)$, from the current state, s_t. The state s_t is encoded to a high-level feature, $\phi(s_t)$, that contains meaningful information for navigation. The generated feature, $\phi(s_t)$, is then employed as input to the policy network, which includes a long short-term memory (LSTM) network to determine an action of the agent, a_t, based on the temporal history. The agent then moves based on the determined action, a_t, and acquires the next state, s_{t+1}, from the environment. (b) Concurrently, given s_t, s_{t+1}, and a_t at each step, the forward and inverse models learn to predict the future state, $\bar{\phi}(s_{t+1})$, and classify the performed action, $\bar{a}_t$, respectively.

tion, $\bar{\phi}(s_{t+1}) \in \mathbb{R}^D$, to constrain the state transition and to encode information that relates to the task (see Fig. 1(b)):

$$\bar{\phi}(s_{t+1}) = f(s_t, a_t; \theta_A, \theta_S), \tag{1}$$

where the network parameters, θ_A and θ_S, are learned for the action representation module and the state transition network, respectively.

The proposed action representation module consists of three deconvolutional layers with nonlinear activations (Exponential Linear Unit or ELU) [49]. The input action a_t, represented as a one-hot code, passes through three deconvolutional layers and ELUs to map a single input to multiple outputs. Then the output of the action representation module is fused with an intermediate response of the state transition network and fused to predict the state in the higher dimensional feature space, $\bar{\phi}(s_{t+1})$, which is more expressive for training. Fig. 2 compares the architectures of the conventional and proposed forward models. The proposed model (see Fig. 2(b)) decodes an action with the one-hot code to a feature map with increased dimension in height, width, and depth. The decoded feature is then fused with the intermediate feature from the state-transition network. The estimated feature in (1) considerably differs from previous works (Fig. 2(a)) that represented an input action as a one-hot code that was simply concatenated with the response from the fully-connected (FC) layers of the state transition network [7], [17], [26], [48], [50].

To learn the forward model, we minimise the loss function, L_F:

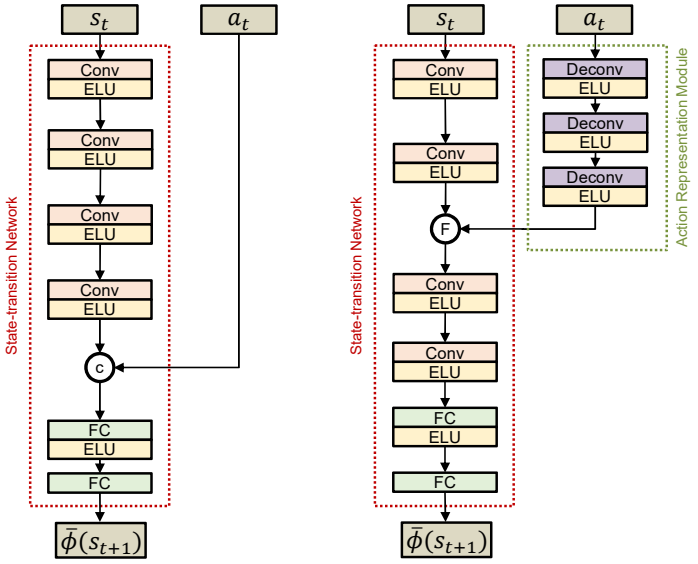
$$L_F = L_{F_R} + \gamma_p L_{F_p}, \tag{2}$$

where L_{F_R} is a regression loss, $p \in \{T, N\}$, and γ_p controls the effect of a triplet ranking loss, L_{F_T} (or an N-tuplet loss, L_{F_N}).

The *regression loss function*, L_{F_R}, is commonly used to minimise the prediction error:

$$L_{F_R}\left(\bar{\phi}(s_{t+1}), \phi(s_{t+1})\right) = \left\|\bar{\phi}(s_{t+1}) - \phi(s_{t+1})\right\|_2^2, \tag{3}$$

where $\phi(s_{t+1})$ is the feature representation from the image (state), s_{t+1}. L_{F_R} ensures that the predicted state from an input state-action pair is close to the future state. As the state



(a) (b)

Fig. 2. Comparison between (a) the conventional forward model and (b) our models with different fusion methods (in F): concatenation (in C) and gated fusion. The proposed models, which consist of a state transition network combined with an action representation module, estimates the feature of the (future) state $\bar{\phi}(s_{t+1})$ from an input state-action pair (s_t, a_t). The one-hot code of an input action passes through three deconvolution (Deconv) layers with Exponential Linear Unit (ELU) to generate decoded responses that are then fused to an intermediate response of the state transition network. The fused responses estimate the feature of the future state after subsequent convolutions (Conv) and fully connected (FC) layers.

prediction without any actions is the current state itself, s_{t+1} can be encoded into a feature representation as:

$$\phi(s_{t+1}) = f(s_{t+1}, a_{t+1} = \emptyset; \theta_A, \theta_S), \tag{4}$$

where $a_{t+1} = \emptyset$ denotes no action (*no-op*).

Although using only L_{F_R} in (3) can provide satisfactory performance, we extend it to consider negative samples, predictions from a current state with other actions, together with the positive sample, $\phi(s_{t+1})$. Based on the intuition that the state-transition network should encode a correct feature and keep predictions from other action different, we enhance

training and the localisation ability with a triplet ranking loss and a N-tuplet loss. These loss functions penalise negative features while maintaining the property of L_{F_R}.

The *triplet ranking loss function* [20], [51], L_{F_T}, pushes $\bar{\phi}\left(s_{t+1}\right)$ to be far from a prediction from s_t with one of different $N-1$ input actions from the current action a_t, which we define as $\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)$, with $i=1,...,N-1$, while maintaining the property of (3), in the form:

$$L_{F_T}\left(\bar{\phi}\left(s_{t+1}\right),\phi\left(s_{t+1}\right),\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)\right)=\max\left\{0,m+\right.$$
$$\left.\left\|\bar{\phi}\left(s_{t+1}\right)-\phi\left(s_{t+1}\right)\right\|_2^2-\left\|\bar{\phi}\left(s_{t+1}\right)-\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)\right\|_2^2\right\}, \tag{5}$$

where m is a margin. Here an action, different from the current action a_t, is randomly selected.

The triplet ranking loss function samples one action among the $N-1$ remaining ones, as shown in Fig. 3, thus making training difficult [20]. In fact, it is difficult to exploit the process of pushing $\bar{\phi}\left(s_{t+1}\right)$ far from predictions from s_t with $N-1$ different actions, $\bar{\phi}\left(\tilde{s}_{t+1}^{1}\right)$, $\bar{\phi}\left(\tilde{s}_{t+1}^{2}\right),...,\bar{\phi}\left(\tilde{s}_{t+1}^{N-1}\right)$, at the same time.

To address this limitation, we adopt an *N-tuplet loss* function [53] in the form:

$$L_{F_N}\left(\bar{\phi}\left(s_{t+1}\right),\phi\left(s_{t+1}\right),\{\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)\}_{i=1}^{N-1}\right)=$$
$$\log\left(1+\sum_{i=1}^{N-1}\exp\left(\bar{\phi}\left(s_{t+1}\right)^T\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)-\bar{\phi}\left(s_{t+1}\right)^T\phi\left(s_{t+1}\right)\right)\right), \tag{6}$$

where $\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)$ denotes the prediction from s_t and one of $N-1$ remaining actions. The N-tuplet loss function enables the network to discriminate between $N-1$ predictions from remaining actions and the prediction from the selected input action, as shown in Fig. 3. As the training proceeds, the distance between $\bar{\phi}\left(s_{t+1}\right)$ and $\phi\left(s_{t+1}\right)$ becomes closer, unlike the distance between $\bar{\phi}\left(s_{t+1}\right)$ and $\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)$ is small in the observation (image) space, because they originate from the same input image s_t.

L_{F_N} in (6) can be considered as a classification problem: maximising the probability for $\bar{\phi}\left(s_{t+1}\right)$ to be categorised into $\phi\left(s_{t+1}\right)$:

$$\log\left(1+\sum_{i=1}^{N-1}\exp\left(\bar{\phi}\left(s_{t+1}\right)^T\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)-\bar{\phi}\left(s_{t+1}\right)^T\phi\left(s_{t+1}\right)\right)\right)$$
$$=-\log\frac{\exp\left(\bar{\phi}\left(s_{t+1}\right)^T\phi\left(s_{t+1}\right)\right)}{\exp\left(\bar{\phi}\left(s_{t+1}\right)^T\phi\left(s_{t+1}\right)\right)+\sum_{i=1}^{N-1}\exp\left(\bar{\phi}\left(s_{t+1}\right)^T\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)\right)}, \tag{7}$$

where the positive future prediction $\bar{\phi}\left(s_{t+1}\right)$ can be considered as a feature vector, and $\phi\left(s_{t+1}\right)$ and $\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)$ as weight vectors. L_{F_N} makes $\bar{\phi}\left(s_{t+1}\right)$ closer to $\phi\left(s_{t+1}\right)$ while other $\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)$ become negative samples.

In the training procedure, the parameters from the action representation module in the forward model are implicitly learned from an input action, while there is no explicit loss function related to the module. In summary, the action representation module derives high dimensional features from a simple one-hot code, producing more effective feature representations to predict the future state from the forward model.

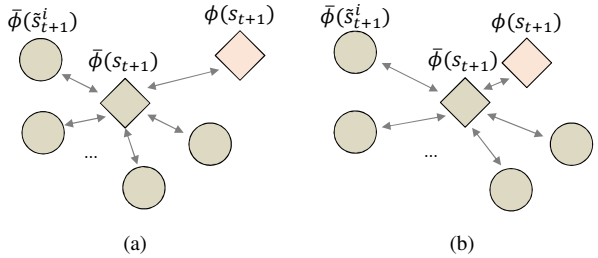


Fig. 3. Effect of the N-tuplet loss function for learning the forward model. From (a) to (b): the N-tuplet loss aims to encode $\bar{\phi}\left(s_{t+1}\right)$ and $\phi\left(s_{t+1}\right)$ close in the prediction feature space, while pulling the predictions from all remaining actions, $\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)$ (gray circles), away from $\bar{\phi}\left(s_{t+1}\right)$. The N-tuplet loss is a generalisation of the triplet ranking loss, where only one out of all remaining actions is sampled as $\bar{\phi}\left(\tilde{s}_{t+1}^{i}\right)$.

C. Fusion

We consider two different methods, namely concatenation and gated fusion, to combine the responses from the action representation module with the state-transition networks.

The decoded responses of the one-hot code of an input action can be *concatenated* to an intermediate response of the state transition network as shown in Fig. 2(b). The concatenated responses estimate the feature of the future state after subsequent convolutions (Conv) and fully connected (FC) layers.

We further present a *gated fusion* approach [54] to combine the action representation with the state-transition network effectively, which thus can be easily adapted to other pre-trained models. As shown in Fig. 2(b), the responses from the action representation module are element-wise multiplied to the responses from the intermediate layer from the state-transition network. This enables the interaction between the decoded action representation and the state representation. Unlike concatenation [20], the multiplication is performed channel-wise, thus fusion does not increase the size of the responses. Table I summarises the detailed architectures.

D. Inverse Model and Policy Network

It is worth noting that the relationship between forward and inverse models is important: the inverse model can provide supervision to learn representations that the forward model regularises by learning to predict s_{t+1} [7], [48]. The inverse model learns to recognise an actual action, a_t, from the input states, s_t and s_{t+1}, which explains the transition of s_t into s_{t+1} [24].

Learning the inverse model can impose constraints on the encoded representation to be able to efficiently predict actions. We employ the inverse model that generates a predicted action, $\bar{a}_t$, to change from $\phi\left(s_t\right)$ to $\phi\left(s_{t+1}\right)$ as follows [7], [20]:

$$\bar{a}_t=g\left(\phi\left(s_t\right),\phi\left(s_{t+1}\right);\theta_I\right), \tag{8}$$

where θ_I denotes the network parameters of the inverse model. As shown in Table II, the inverse model $g\left(\cdot\right)$ concatenates two

TABLE I
CONFIGURATION DETAILS OF FORWARD MODELS WITH CONCATENATION (FORWARD MODEL-C) AND GATED FUSION (FORWARD MODEL-G), WHICH CONSIST OF CONVOLUTION (CONV), DECONVOLUTION (DECONV), AND FULLY CONNECTED (FC) LAYERS WITH EXPONENTIAL LINEAR UNIT (ELU). THE ACTION REPRESENTATION MODULE DECODES THE RESPONSES BY SETTING THE VALUE OF STRIDE TO 2 [52]. THE FILTER SIZE IN THE BRACKETS REPRESENTS OUTPUT DEPTH, INPUT DEPTH, WIDTH, HEIGHT OF THE FILTER. 'CONCAT' AND 'DOT PROD' DENOTE CONCATENATION AND DOT PRODUCT, RESPECTIVELY.

Forward Model-C				**Forward Model-G**			
State Transition Network		Action Representation Module		State Transition Network		Action Representation Module	
Layer	**Filter** (stride)	**Layer**	**Filter** (stride)	**Layer**	**Filter** (stride)	**Layer**	**Filter** (stride)
Conv1/ELU	$[32, 1, 3, 3]$ (1)	Deconv1/ELU	$[4, 4, 3, 3]$ (2)	Conv1/ELU	$[32, 1, 3, 3]$ (1)	Deconv1/ELU	$[8, 4, 3, 3]$ (2)
Conv2/ELU	$[32, 32, 3, 3]$ (1)	Deconv2/ELU	$[8, 4, 3, 3]$ (2)	Conv2/ELU	$[32, 32, 3, 3]$ (1)	Deconv2/ELU	$[16, 8, 3, 3]$ (2)
Concat	-	Deconv3/ELU	$[8, 8, 3, 3]$ (2)	Dot prod	-	Deconv3/ELU	$[32, 16, 3, 3]$ (2)
Conv3/ELU	$[40, 40, 3, 3]$ (1)			Conv3/ELU	$[32, 32, 3, 3]$ (1)		
Conv4/ELU	$[32, 40, 3, 3]$ (1)			Conv4/ELU	$[32, 32, 3, 3]$ (1)		
FC1/ELU	$[256, 288, 1, 1]$ (1)			FC1/ELU	$[256, 288, 1, 1]$ (1)		
FC2	$[D, 256, 1, 1]$ (1)			FC2	$[D, 256, 1, 1]$ (1)		

TABLE II
CONFIGURATION DETAILS OF THE INVERSE MODEL THAT CONSISTS OF FULLY CONNECTED (FC*) LAYERS AND EXPONENTIAL LINEAR UNIT (ELU). 'CONCAT' DENOTES A CONCATENATION OF TWO RESPONSES.

Inverse Model	
Action Classification Network	
Layer	**Filter** (stride)
Concat	-
FC*1/ELU	$D \times 2D \times 1 \times 1$ (1)
FC*2	$N \times D \times 1 \times 1$ (1)

feature vectors and passes them through the subsequent FC*1 ($D \times 2D \times 1 \times 1$)[2], ELU, and FC*2 ($N \times D \times 1 \times 1$) layers. The loss function, $L_I(\bar{a}_t, a_t)$, is a soft-max as the problem in (8) generates a discrete action label, which can be considered as a classification among several possible discrete actions.

Finally, the encoded feature from the forward model, $\phi(s_t)$, is fed into a policy network, $\pi(\cdot)$, that consists of a long short-term memory (LSTM) network, which memorises information for several timesteps. The LSTM network consists of a memory cell with D units to process the temporal dependencies during training [7], [25], [26]. Since the agent can be equipped with memory of previous states, using the LSTM network is important for mapless navigation that needs to remember the previously visited area.

IV. REINFORCEMENT AND IMITATION LEARNING

We integrate the proposed model with two approaches for mapless navigation, namely RL-based navigation and IL-based navigation.

A. Reinforcement Learning for Mapless Navigation

In the RL-based navigation, rewards are given when training the camera agent [7], [20]. The objective of the training is to maximise the extrinsic reward that is received when the agent reaches the target.

We additionally exploit two separate FC layers to estimate the value function and state functions that consist of a cell state and hidden states for memorising the past steps and generating the action a_t. We consider curiosity-driven exploration where the goal is to optimise the model with additional extrinsic and intrinsic rewards [7]. The extrinsic reward can be obtained when the model reaches the target. The intrinsic rewards are obtained by measuring the prediction error, which encourages the agent to explore the unseen area. In the end, the intrinsic rewards relate to the achievement of the final goal.

During training, we optimise the function in the form:

$$\min_{\theta_A, \theta_S, \theta_I, \theta_P} -E_\pi \left[\sum\nolimits_{t=0}^{k} r_t \right] + \beta L_F + (1-\beta) L_I, \qquad (9)$$

where $r_t = r_t^{ext} + r_t^{int}$, and r_t^{ext} and r_t^{int} are extrinsic and intrinsic rewards, respectively; and k, which can vary for each episode but is upper bounded, is the number of time steps when the agent moved. E_π is the expectation of rewards generated by π. The hyper-parameter, β, controls the weight between the forward and inverse models as the balance between the two is important to train the model. We employ L_{F_R} as r_t^{int} in [7].

B. Imitation Learning for Mapless Navigation

IL trains a policy to follow an expert (human) demonstration, that can be seen as prior information about the environment provided by human behaviour. Here, the controls or paths from the expert are provided and the agent then tries to imitate them by behavioural cloning or inverse reinforcement learning. *Behavioural cloning* directly learns a policy through state-action pairs provided by an expert and without the agent interacting with the environment during training [55], [56], [57]. *Inverse reinforcement learning* learns to estimate a reward function based on state-action pairs from an expert [58]. The reward function is also used to infer an imitation policy combined with RL [22], [59], [60], [61]. Imitation learning achieves good performance, but the expert demonstration is labour intensive, prone to bias, and needed for each new task. To mitigate the problem that IL is labour intensive, a recent work proposed imitation of an agent behaviour without any expert supervision [26], which considers the agent to achieve the final goal regardless of its intermediate action.

In the integration with IL, we use the output of the LSTM network directly to estimate the state functions to generate the

[2]Note that FC* for the inverse model uses different parameters from FC.

action a_t. The representation is learned from data collected offline. The learned model is then transferred to the camera agent to perform a goal-finding task by providing a target image as an additional input. We consider zero-shot imitation where the goal is to optimise the model without expert supervision [26].

In training with the collected data, the goal of the policy is to sequentially generate actions conditioned on the current state, starting from time step t_1 to reach the goal observation at time step t_E. In addition to forward and inverse models for training the agent, the consistency loss [26] is introduced as follows:

$$L_C\left(\hat{\phi}\left(s_{t+1}\right), \phi\left(s_{t+1}\right)\right) = \left\|\hat{\phi}\left(s_{t+1}\right) - \phi\left(s_{t+1}\right)\right\|_2^2, \quad (10)$$

where $\hat{\phi}\left(s_{t+1}\right) = f\left(s_t, \hat{a}_t; \theta_A, \theta_S\right)$ is the predicted representation from $\hat{a}_t = \pi\left(s_t, s_{t_{end}}; \theta_P\right)$, that aims to generate sequential actions to reach the goal. The consistency loss is computed at each time step and jointly optimised with forward and inverse losses over the whole trajectory during time steps. Since there is no expert trajectory in this scenario, the consistency loss helps reach the goal more easily [26]. From time step t_1 to t_E, the model is trained with the following loss functions:

$$\min_{\theta_A, \theta_S, \theta_I, \theta_P} \Big[\sum_{t=t_1}^{t_E} \beta L_F\left(\bar{\phi}\left(s_{t+1}\right), \phi\left(s_{t+1}\right), \{\bar{\phi}\left(\tilde{s}_{t+1}^i\right)\}_{i=1}^{N-1}\right) + (1-\beta) L_I\left(\bar{a}_t, a_t\right) + \eta L_C\left(\hat{\phi}\left(s_{t+1}\right), \phi\left(s_{t+1}\right)\right)\Big], \quad (11)$$

where η controls the effect of L_C. At each step, the current state, s_t, the current action, a_t, and the goal state, $s_{t_{end}}$, are used as inputs for the policy to sequentially generate an action, $\hat{a}_t$, with the objective is to move the agent towards the goal.

V. Validation

This section shows how the view-action representation learning is crucial for different learning approaches to efficiently achieve the goal of mapless navigation. We use VizDoom and Gazebo as environments for RL-based and IL-based mapless navigation, respectively. For a fair comparison, we follow the same architecture for the state transition network and inverse model. The state transition network consists of 4 convolution (Conv) layers followed by fully connected (FC) layers and the inverse model consists of two FC layers. A nonlinear activation, exponential linear units (ELUs), is added after each Conv and FC layer except for the last FC output. Our model additionally takes three deconvolution (Deconv) layers with ELUs for the Action Representation Module. Table I shows the details of the network configuration of two forward models including concatenation (Forward model-C) and gated fusion (Forward model-G) process. In the experiment, the number of action is $N = 4$. These actions are *move forward*, *turn right*, *turn left*, and *no-op*. The encoded feature dimension is $D = 256$. All agents are trained using images converted to greyscale and resized to 42×42 pixels [25]. We use the ADAM solver [62] with initial learning rate 10^{-4}.

TABLE III
Success ratios with different fusions and loss functions. Success refers to the agent achieving the goal during trials within 20M steps. The first and second best results are reported in bold and underlined, respectively.

Model		Success ratio (%)		
Fusion	Loss	Dense	Sparse	Ext. Sparse
Concatenation	L_{F_R}	**96.78 ± 17.63**	91.33±28.13	87.10±33.51
	L_{F_T}	8.13 ± 27.33	0.0006 ± 0.00	11.25 ± 3.35
	L_{F_N}	2.13 ± 1.33	0.001 ± 0.00	7.25 ± 1.35
	$L_{F_R} + L_{F_T}$	96.06 ± 19.43	94.45 ± 22.87	87.13 ± 33.48
	$L_{F_R} + L_{F_N}$	94.50 ± 22.23	**95.52 ± 22.78**	**88.22 ± 32.22**
Gated	L_{F_R}	94.25 ± 19.23	89.45 ± 29.09	71.45 ± 45.05
	L_{F_T}	9.05 ± 26.58	5.54 ± 11.12	8.25 ± 7.13
	L_{F_N}	1.93 ± 0.93	0.001 ± 0.00	3.25 ± 0.35
	$L_{F_R} + L_{F_T}$	92.74 ± 25.94	91.47 ± 27.92	73.75 ± 44.00
	$L_{F_R} + L_{F_N}$	**95.75 ± 20.14**	**92.58 ± 26.20**	**87.53 ± 33.03**

A. Reinforcement Learning

Let Ours-C-T, Ours-C-N, Ours-G-T, and Ours-G-N be four combinations of the proposed approach with two fusion methods (Concatenation, C, and Gated fusion, G) and two additive loss functions (triplet ranking loss, T, and N-tuplet loss, N). We compare our models with another self-supervised network, ICM (Intrinsic Curiosity Module) [7], and a network that only considers extrinsic rewards in training, A3C (a vanilla Asynchronous Advantage Actor-Critic) [25]. Our models and ICM are built on A3C. For a fair comparison, the same architecture for the state transition network is employed, except for Ours-C-T and Ours-C-N, which increase the number of parameters in Conv3 layer to concatenate the response from the Action Representation Module. Also, our models additionally increase the number of parameters by adopting the Action Representation Module. To train the networks, sixteen workers are used to perform RL following the asynchronous training protocol in A3C [25].

In the *VizDoom MyWayHome* environment [63], [64] the goal is to reach an armour (see Fig. 4). We consider three settings, namely dense, sparse, and extremely sparse extrinsic rewards [7] as shown in Fig. 5. With the dense setting, the agent can randomly spawn one of 17 locations (some of which are close to the goal). In the sparse and extremely sparse settings, the agent takes at least 270 and 350 steps (actions) to reach the goal state, respectively. Episodes are terminated when the agent reaches the armour or when 2100 time steps are completed. The agent can perform four discrete actions: *move forward*, $a_t = (1,0,0,0)$; *turn right*, $a_t = (0,1,0,0)$; *turn left*, $a_t = (0,0,1,0)$; and *no-op*, $a_t = (0,0,0,1)$. For efficient training, we set the action to be repeated four times [25]. The total number of steps taken by all workers is 20M, and the value of the hyper-parameter for β is set among $\{0.15, 0.18, 0.2\}$ which shows the best result, and $\gamma_T = 1.0$, $m = 3 \times 10^{-5}$ and $\gamma_N = 0.001$.

Note that the objective of the RL-based mapless navigation is learning to reach the goal efficiently by receiving extrinsic rewards. We thus show the effectiveness of the proposed model

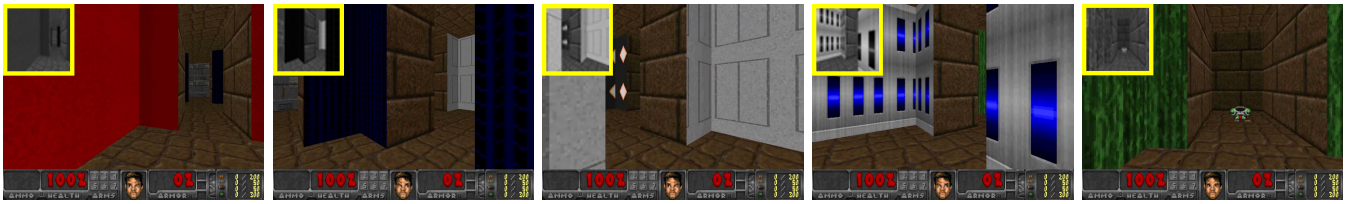


Fig. 4. Navigation in the VizDoom environment. Original images are cropped, resized, and converted to grayscale images (in yellow box) for training and testing. In testing, the agent takes the current state as an input and determines the next action to perform until it reaches the goal or has moved by a maximum, pre-defined number of steps.

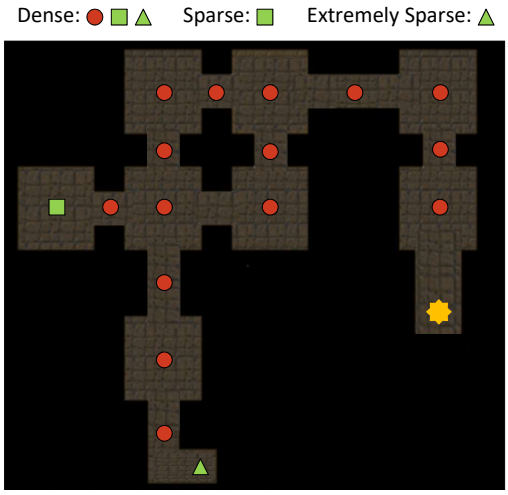


Fig. 5. A VizDoom map with spawning points (circle, square, and triangle) and the goal location (in yellow). In dense setting, the agent can be randomly spawned from 17 different points. In sparse and extremely sparse settings, the agent is spawned about 270 and 350 steps away from the goal, respectively.

by presenting the average success ratio within fixed global steps in Fig. 6 and Table III. Fig. 6 shows that with the dense setting, all models have good performance, whereas with sparser rewards the proposed model has good performance and the performance of the other models degrades. In settings with sparse and extremely sparse extrinsic rewards, A3C fails to perform navigation as it has insufficient feedback to improve itself during training and the policy cannot be trained efficiently. ICM has good performance with sparse rewards, but slow convergence in the environment containing extremely sparse rewards. As the sparsity of the extrinsic rewards increases, our models generally outperform other models, indicating effective learning of the features during exploration.

To investigate the contribution of the components within the proposed network, we further conduct an ablation analysis by training the agent with different fusion methods and loss functions and compute the success rate during 20M total steps. Note that we use the inverse model for all experiments. Table III shows a good performance when L_{F_T} or L_{F_N} are additionally used with L_{F_R}. The concatenation approach generally shows better performance than the gated fusion approach. L_{F_R} combined with L_{F_N} performs better than the one with triplet ranking loss L_{F_T}. When the network is trained only with L_{F_T} or L_{F_N}, the agent is unable to perform the

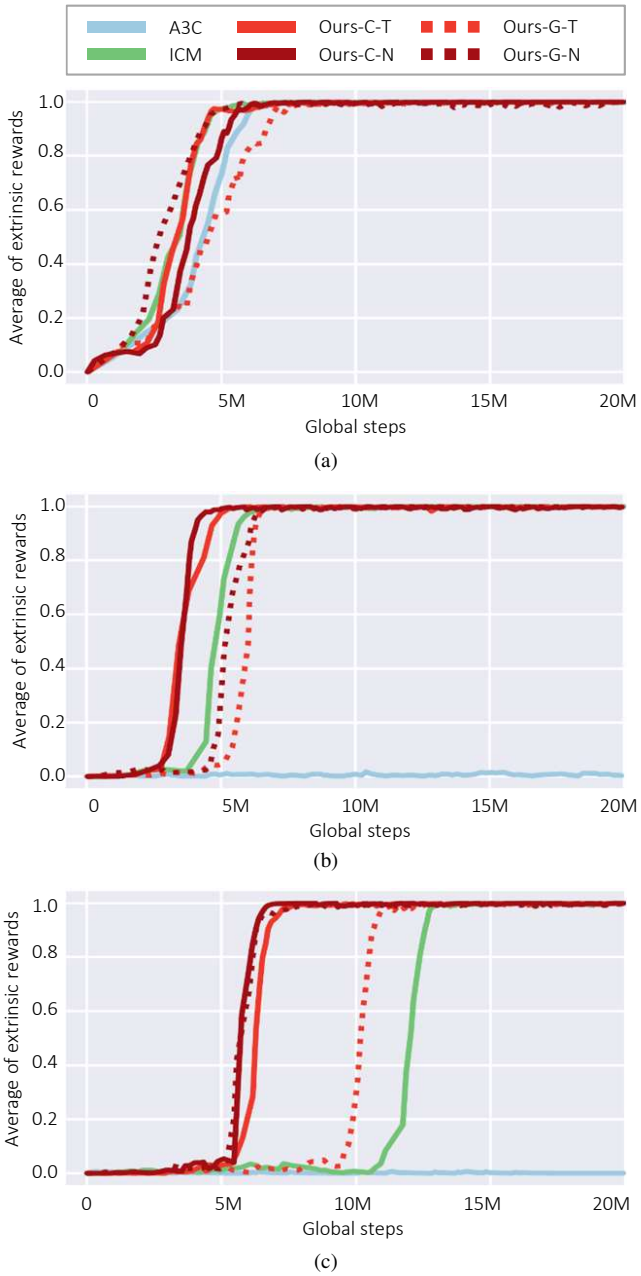
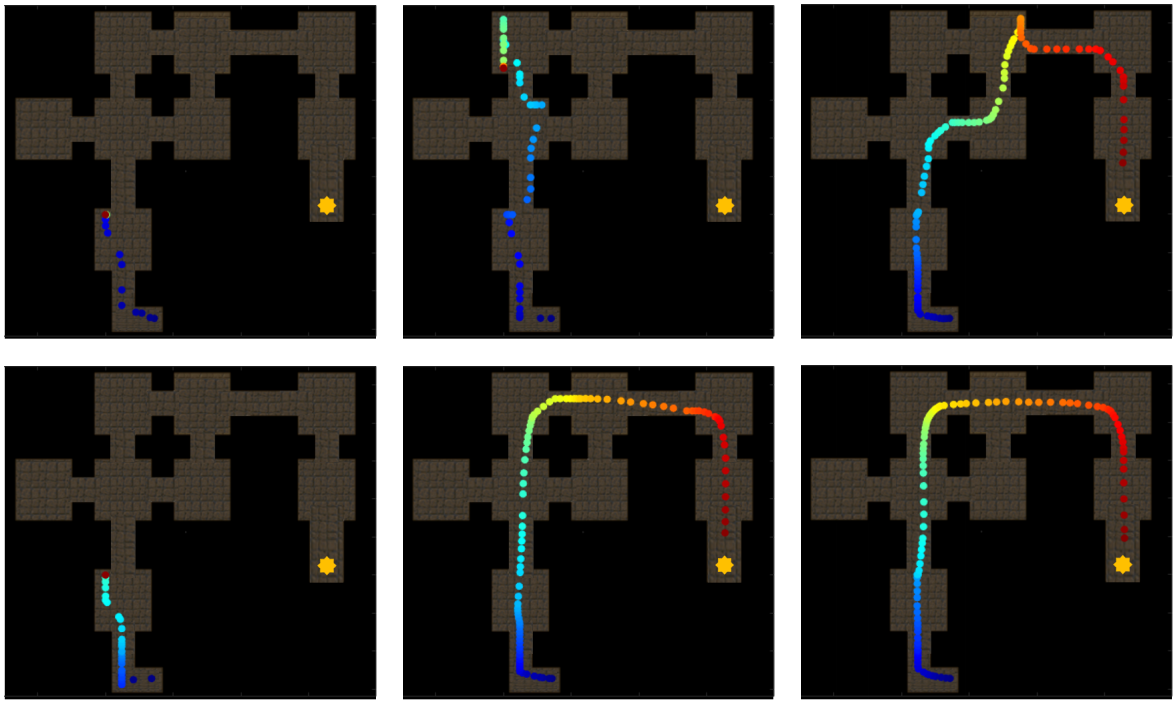
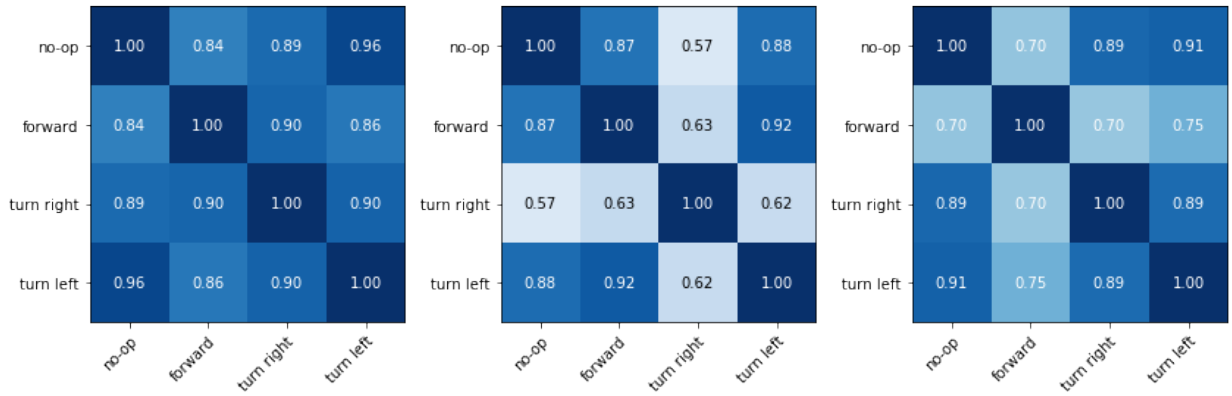


Fig. 6. Extrinsic rewards for an agent with A3C, ICM, and Ours-* in an environment with (a) dense, (b) sparse, and (c) extremely sparse extrinsic rewards. Four combinations of our model are evaluated by considering various options in fusion (concatenation, C, and gated fusion, G) and the additional loss (triplet ranking loss, T, and N-tuplet loss, N).



(a) A3C (b) ICM (c) Ours-C-N

Fig. 7. Sample trajectory of (a) A3C (b) ICM (c) Ours-C-N, trained with (top) 8M and (bottom) 20M steps in the VizDoom environment with extremely sparse rewards. The agent trajectories are presented in yellow lines. The A3C agent cannot reach the goal when it is trained with extremely sparse reward setting. Both ICM and Ours-C-N agents have good performance with 20M global training steps. However, ICM has slower convergence in training than Ours-C-N.



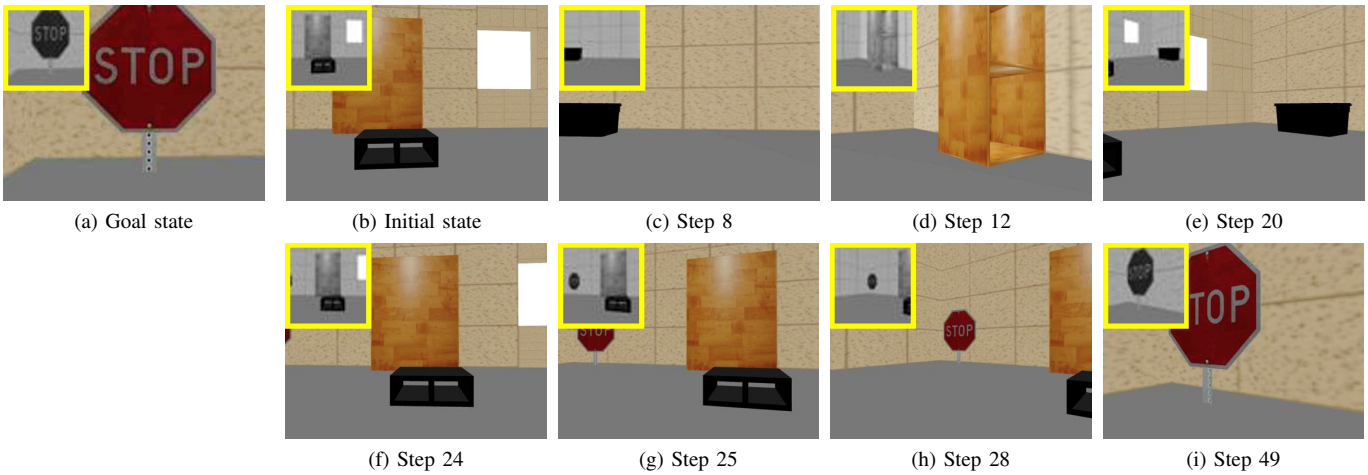
(a) Ours-C-R (b) Ours-C-T (c) Ours-C-N

Fig. 8. Confusion matrices of the predicted features generated from the same state with four different actions: *no-op*, *move forward (forward)*, *turn right*, and *turn left*. Results with (a) Ours-C-R, (b) Ours-C-T, and (c) Ours-C-N.

navigation task, since the L_{F_R} is directly related to the intrinsic reward, r_t^{int}, which should decrease as the exploration is processed [20].

Fig. 7 shows the trajectory generated with extremely sparse extrinsic rewards. Each agent is trained with 8M and 20M global steps respectively in order to demonstrate the effectiveness of learning representation with respect to the timesteps. Due to the extremely sparse extrinsic rewards, the A3C agent has insufficient feedback to improve itself during training. In fact, the agent fails to perform navigation in both 8M and 20M global training steps, as shown in Fig. 7(a). The ICM and Ours-C-N agents show good performance when the global training steps are set to 20M. When the agent is trained with 8M global steps with the setting of extremely sparse rewards, as shown in Fig. 7 (top), the agent of Ours-C-N still achieves the goal while others cannot reach the goal, indicating that our model effectively learns the features during exploration.

Fig. 8 shows the effect of L_{F_T} and L_{F_N} for training the forward model and compares the predicted feature from the same state with four different actions. We use a forward model with randomly initialised parameters, Ours-C-R , Ours-C-T, and Ours-C-N. We generate $\bar{\phi}\left(s_{t+1}\right) = f\left(s_t, a_t; \theta_A, \theta_S\right)$ in (1) by fixing s_t while changing a_t. The similarity between two features, e.g., $\bar{\phi}_1\left(s_{t+1}\right)$ and $\bar{\phi}_2\left(s_{t+1}\right)$, is measured as $\exp\left(-\left\|\bar{\phi}_1\left(s_{t+1}\right) - \bar{\phi}_2\left(s_{t+1}\right)\right\|_2^2\right)$. The result is averaged by



(a) Goal state (b) Initial state (c) Step 8 (d) Step 12 (e) Step 20

(f) Step 24 (g) Step 25 (h) Step 28 (i) Step 49

Fig. 9. Goal-finding task in Gazebo. Original images are resized, and converted to grayscale (see the yellow square) for training and testing. At test time, the camera agent takes (a) the goal and (b) current state as inputs, and performs the goal-finding task by exploring the environment. (c-f) The agent first looks around since there is no overlap between the initial and the goal state. (g-i) Once the goal is observed, the agent moves towards it.

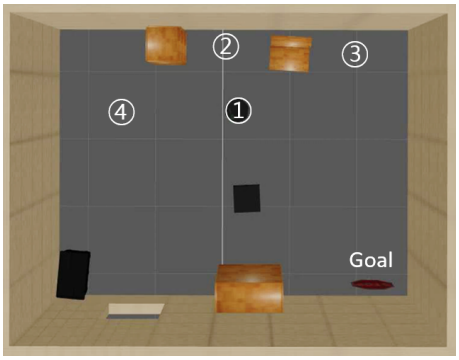


Fig. 10. Goal-finding task in Gazebo with four different spawning points for the agent.

TABLE IV
THE NUMBER OF SUCCESSES AND AVERAGE NUMBER OF STEPS TO REACH THE GOAL (IN BRACKETS) FOR IL-BASED MAPLESS NAVIGATION IN GAZEBO. THE STEPS ARE COUNTED ONLY WITH SUCCESSFUL CASES.

Trial	1	2	3	4
Random	Fail	Fail	Fail	Fail
Inverse [47]	Fail	Fail	Fail	Fail
ZSI [26]	2/5 (39 steps)	3/5 (49 steps)	2/5 (42 steps)	Fail
Ours-C-R	3/5 (66 steps)	3/5 (87 steps)	2/5 (70 steps)	Fail
Ours-C-T	2/5 (28 steps)	1/5 (35 steps)	2/5 (107 steps)	1/5 (67 steps)
Ours-C-N	5/5 (44 steps)	4/5 (75 steps)	4/5 (70 steps)	1/5 (70 steps)

using images collected in the VizDoom environment during navigation. Ours-C-T and Ours-C-N (Fig. 8(b) and Fig. 8(c)) have better discriminative performance between two features generated from different actions than Ours-C-R (Fig. 8(a)). The feature prediction of Ours-C-T is biased to the *turn right* action and has distinctive low similarities with the feature predictions from other actions.

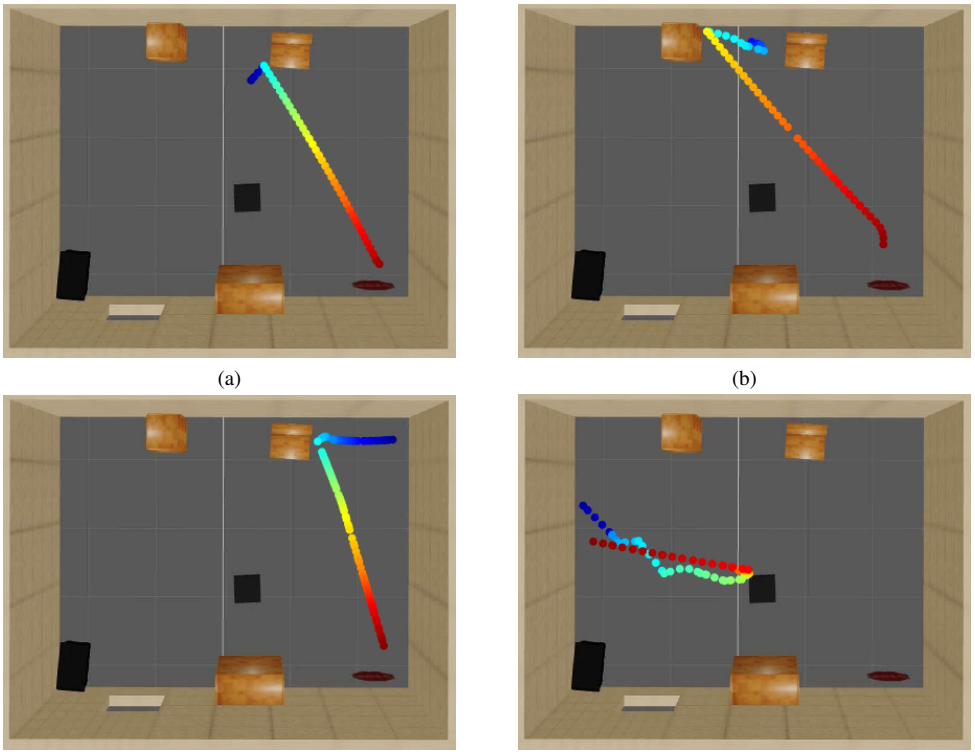
B. Imitation Learning

We compare our model, Ours-C-R (the forward model trained only with L_{F_R}), Ours-C-T and Ours-C-N, with other approaches, random search, Inv (Inverse model) [47], and ZSI (Zero-shot Imitation) [26]. Note that ZSI encodes the input state with the conventional forward model as shown in Fig. 2 and learns to minimise the loss function in (11).

We first collect state-action pairs for every step in a Gazebo TurtleBot2 environment [65] (see Fig. 10). A turtlebot with first person view (RGB image) collects an image and action for each time step. We follow the data collection described in [26]. During data collection, the agent randomly generates one action from four available discrete actions (*move forward*, *turn right*, *turn left*, and *no-op*). If the agent reaches an object or the wall, then it moves backward and then turns right or left by a uniformly sampled angle between 90-270 degrees. The agent autonomously repeated this process and collected 70K state-action pairs data. In the training phase, we use the collected data where the goal image is changed every 20 steps (frames) since no specific goal is set during the training. The agent thus can imitate the exploratory behaviour with various goal images.

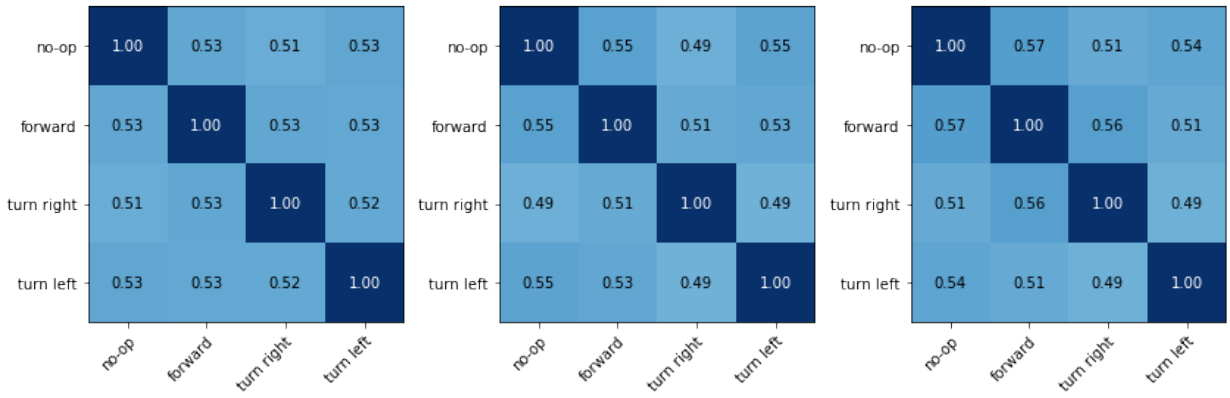
We test whether the learned agent without any goal or expert supervision can find a way to reach the target. We provide a single image of the goal for the agent to find a way to get to the target, as shown in Fig. 9(a). We place the turtlebot where there is no overlap between the initial and target state, 20 to 40 steps away from the target. The agent explores the environment to find the goal and is successful if it stops, within 150 steps, close to the goal. The stopping criterion for the agent is based on a normalised l_2 distance between the feature representations of input and goal images. We use the feature generated by $\phi(\cdot)$ in (4). The hyper-parameter β and η in (11) are empirically both set to 0.1, and γ_N is set to 0.001.

As the objective is learning to navigate without expert supervision, we evaluate the number of successes with a fixed number of trials, i.e. five times, as shown in Table IV. The agent with random navigation and learned by Inverse [47] fails



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

Fig. 11. Sample trajectory of Ours-C-N in different locations in the Gazebo environment. The trajectory is colour-coded, starting from blue and ending with red. The camera agent first looks around until it finds the goal. The agent moves towards to reach the goal, once the goal is observed and stops when it (a-c) achieves the goal or (d) has completed a maximum, pre-defined number of steps.



(a) Ours-C-R (b) Ours-C-T (c) Ours-C-N

Fig. 12. Confusion matrices of the predicted features generated from the same state with four different actions: *no-op*, *move forward (forward)*, *turn right*, and *turn left*. Results with (a) Ours-C-R, (b) Ours-C-T, and (c) Ours-C-N.

to achieve the goal. Since the model learned by [47] does not learn the forward model, information about the surrounding environment is not encoded in the representation effectively. ZSI shows good results in terms of the number of steps as it moves towards the detected goal after looking around. However, it fails to reach the goal when it is not detected in the early stages of navigation. The results show that Ours-C-N outperforms other methods, which suggests that the learned representation encodes effective information for understanding the environment.

Fig. 11 shows the trajectory for the goal-finding task performed by Ours-C-N. In the early stage, the agent looks around to find the goal. Once the goal is detected, as shown in Fig. 11(a)-(c), the agent moves forward to reach the object. However, the agent may fail to reach the goal when the movement is blocked by an obstacle as shown in Fig. 11(d).

Finally, Fig. 12 shows the confusion matrices based on the predicted features. The result is averaged by using images collected in Gazebo during navigation by following the same process as described previously for generating the results in Fig. 8. In Fig. 12, because training uses a large amount of data collected offline, it is possible to notice a good performance

in discriminating other feature predictions. In Fig. 12(b) and (c), the similarity between *no-op* and *forward* is higher than others, whereas the similarity between *turn left* and *turn right* is the lowest.

VI. Conclusion

We proposed a view-action representation learning method that expands the dimensions of one-hot codes of input actions and fuses them with a state-transition network. In the context of mapless visual navigation, we also presented two loss functions, triplet ranking loss and N-tuplet loss, each of which can be additionally combined with the regression loss for effective representation learning. We integrated the proposed networks trained with the joint loss functions in RL and IL-based mapless navigation tasks. The validation shows that the proposed networks have faster training convergence than previous networks in RL-based mapless navigation tasks. As future work we will increase the granularity of the camera control with a larger set of actions and validate the proposed view-action representation learning method with other robot types, such as arms and drones.

ACKNOWLEDGMENT

We would like to appreciate Ahmad Ataka and Simon Butcher for their support to set up the simulation environment.

References

[1] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016.

[2] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015.

[3] D. Jayaraman and K. Grauman, "End-to-end policy learning for active visual categorization," *IEEE Trans. Pattern Anal. Machine Intell. (TPAMI)*, vol. 41, no. 7, pp. 1601–1614, 2018.

[4] Q. Cao, L. Lin, Y. Shi, X. Liang, and G. Li, "Attention-aware face hallucination via deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017.

[5] D. Jayaraman and K. Grauman, "Learning to look around: intelligently exploring unseen environments for unknown tasks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018.

[6] S. K. Ramakrishnan and K. Grauman, "Sidekick policy learning for active visual exploration," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018.

[7] D. Pathak, A. E. Pulkit Agrawal, and T. Darrell, "Curiosity-driven exporation by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017.

[8] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012.

[9] M. E. Angelopoulou and C.-S. Bouganis, "Vision-based egomotion estimation on fpga for unmanned aerial vehicle navigation," *IEEE Trans. Circuits Syst. Video Technol. (TCSVT)*, vol. 24, no. 6, pp. 1070–1083, 2013.

[10] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, R. Hadsell *et al.*, "Learning to navigate in cities without a map," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018.

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[13] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. Montiel, "Towards semantic slam using a monocular camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2011.

[14] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.

[15] S. Brahmbhatt and J. Hays, "Deepnav: Learning to navigate large cities," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017.

[16] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[17] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," in *ICRA 2018 Workshop on Machine Learning in Planning and Control of Robot Motion*, 2018.

[18] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi, "Visual semantic planning using deep successor representations," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017.

[19] S. Siriwardhana, R. Weerasekera, and S. Nanayakkara, "Target driven visual navigation with hybrid asynchronous universal successor representations," *arXiv preprint arXiv:1811.11312*, 2018.

[20] C. Oh and A. Cavallaro, "Learning action representations for self-supervised visual exploration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[21] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Proc. IEEE Intell. Veh. Symposium (IV)*, 2017.

[22] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robot. Autom. Lett. (RA-L)*, vol. 3, no. 4, pp. 4423–4430, 2018.

[23] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," *arXiv preprint arXiv:1808.04355*, 2018.

[24] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview," *Neural Networks*, vol. 108, pp. 379–392, 2018.

[25] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016.

[26] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *Proc. Int. Conf. Learn. Rep. (ICLR)*, 2018.

[27] G. Dudek and M. Jenkin, *Computational principles of mobile robotics*. Cambridge university press, 2010.

[28] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 5, pp. 1179–1187, 1989.

[29] D. Kim and R. Nevatia, "Symbolic navigation with a generic map," *Auton. Robots*, vol. 6, no. 1, pp. 69–88, 1999.

[30] G. Oriolo, M. Vendittelli, and G. Ulivi, "On-line map building and navigation for autonomous mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1995.

[31] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose, "Navigation using an appearance based topological map," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2007.

[32] F. Fraundorfer, C. Engels, and D. Nistér, "Topological mapping, localization and navigation using image collections," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2007.

[33] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017.

[34] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015.

[35] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, "# exploration: A study of count-based exploration for deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017.

[36] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.

[37] S. Mohamed and D. J. Rezende, "Variational information maximisation for intrinsically motivated reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015.

[38] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.

[39] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Trans. Evol. Comput.*, vol. 11, no. 2, pp. 265–286, 2007.
[40] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. Int. Conf. Simulation of Adaptive Behavior: From animals to animats*, 1991.
[41] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015.
[42] A. Walsman, Y. Bisk, S. Gabriel, D. K. Misra, Y. Artzi, Y. Choi, and D. Fox, "Early fusion for goal directed robotic vision," *arXiv preprint arXiv:1811.08824*, 2018.
[43] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2018.
[44] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015.
[45] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2016.
[46] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections," *arXiv preprint arXiv:1710.05268*, 2017.
[47] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.
[48] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.
[49] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.
[50] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017.
[51] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015.
[52] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015.
[53] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.
[54] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Gated-attention readers for text comprehension," *arXiv preprint arXiv:1606.01549*, 2016.
[55] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Systems (RAS)*, vol. 57, no. 5, pp. 469–483, 2009.
[56] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Comput.*, vol. 3, no. 1, pp. 88–97, 1991.
[57] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011.
[58] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2004.
[59] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016.
[60] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016.
[61] J. Ho, J. Gupta, and S. Ermon, "Model-free imitation learning with policy optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016.
[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[63] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
[64] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, 2016.
[65] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2004.

Changjae Oh received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2011, 2013, and 2018, respectively. From 2018 to 2019, he was a postdoctoral research assistant at the school of electrical engineering and computer science from Queen Mary University of London, UK, where he has been a Lecturer since 2019. His research interests include image processing for machine perception and vision-based perception for robot control.

Andrea Cavallaro received the Ph.D. degree in electrical engineering from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 2002. He is Professor of Multimedia Signal Processing and the founding Director of the Centre for Intelligent Sensing at Queen Mary University of London (QMUL, UK), Turing Fellow at the Alan Turing Institute, the UK National Institute for Data Science and Artificial Intelligence, and Fellow of the International Association for Pattern Recognition. He is Editor-in-Chief of Signal Processing: Image Communication; Senior Area Editor for the IEEE Transactions on Image Processing; Chair of the IEEE Image, Video, and Multidimensional Signal Processing Technical Committee; and an IEEE Signal Processing Society Distinguished Lecturer.