Bird's Eye View Based Pretrained World model for Visual Navigation

Kiran Lekkala, Chen Liu and Laurent Itti

University of Southern California, Los Angeles 90007,
klekkala@usc.edu

Abstract. This paper presents a novel system that fuses components in a traditional World Model into a robust system, that is pretrained entirely on a dataset of unlabeled videos and random trajectories from a simulator. The pretrained model, when frozen and deployed, zero-shot transfers to unseen environments for fast Reinforcement Learning of a 1-layer policy or planning. To facilitate transfer, we use a representation that is based on *Bird's Eye View (BEV)* images. Thus, our agent learns representations of the observation by first learning to translate from complex *First-Person View (FPV)* based RGB images to BEV representations, then learning to navigate using those representations. Later, when deployed, the agent uses the perception model that translates FPV-based RGB images to embeddings that were learned by the FPV to BEV translator and that can be used by the downstream policy. The incorporation of state-checking modules using *Anchor images* and Mixture Density LSTM not only interpolates uncertain and missing observations but also enhances the robustness of the model in the real-world. We trained the model using data from a Differential drive robot in the CARLA simulator. Our methodology's effectiveness is shown through the deployment of trained models onto a real-world Differential drive robot, where using the BEV representation leads to better transfer and faster learning. Lastly we release a comprehensive codebase, dataset and models for training and deployment (https://sites.google.com/usc.edu/world-model-sim2real/).

1 Introduction

Reinforcement Learning (RL) has predominantly been conducted in simulator environments, primarily due to the prohibitive costs associated with conducting trial-and-error processes in the real world. With the advances in graphics and computational technologies, there has been a significant development in realistic simulators that capture the system (robot) information. RL is a widely sought-after learning method because of its only need of a sparse reward signal for the task. However, it is compute-intensive and slow, especially when we train models end-to-end in simulators (23). An alternative for RL is Imitation learning or Behaviour cloning, but it necessitates the collection of expert data.

In this paper, we formulate a new setting for *Zero-shot* transfer for Visual Navigation without Maps, involving unlabeled expert videos and random trajectory rollouts obtained from the CARLA simulator, as outlined in Fig. 1. To avoid

any Sim2Real gap within the control pipeline and focus only on the perception transfer, we built a Differential-drive-based robot in the CARLA simulator that closely resembles our real-world robot. Using this setup, we construct a large dataset consisting of *First-person view (FPV)* and *Bird's eye view (BEV)* image sequences from the CARLA (5) simulator. The system is pre-trained entirely on these unlabeled videos and random trajectory datasets obtained from the simulator. The system is then frozen and deployed for an online visual navigation task. This pretraining is inexpensive as it runs in a simulator, but we hypothesize that BEV maps contain crucial information to facilitate learning of future navigation tasks. Here, we hence seek to answer the question of whether such pre-training can benefit and accelerate the learning of downstream visual navigation tasks.

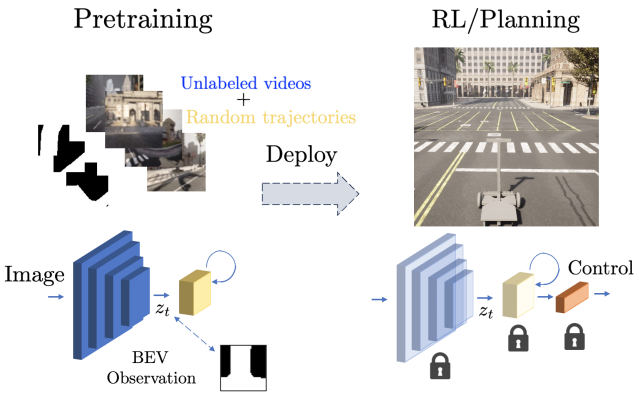


Fig. 1. Overview of our system We first pretrain the visual navigation system on a set of large-scale dataset, consisting of unlabeled expert videos (expert videos without actions used to pretrain the encoder) and random trajectory rollouts (used to train memory module.), collected in the simulator. Once the model is pretrained. The frozen model is deployed for performing Visual Navigation either using Reinforcement Learning or Planning.

2 Related work

Although, many methods (6; 10; 12; 13) use simulators for learning through an extensive amount of experiences that could be used to train a model policy end to end, some recent works (2) have shown promising results, on various tasks (9**?**), using encoders that are pretrained on large unlabelled expert data and then train a significantly smaller network on top of the frozen encoder. Since these encoders are not trained on a specific task, we call it pretraining. Representations estimated using these *pretrained and frozen* encoders would help the model remain lightweight and flexible, which is desirable for mobile platforms.

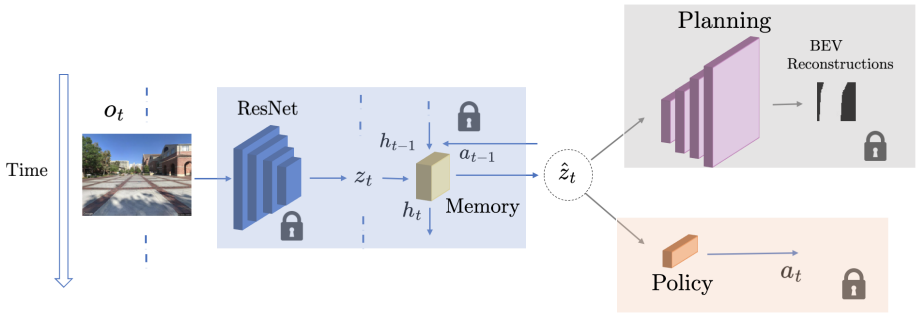


Fig. 2. Working of the System. RGB observation o_t at time step t is passed to the perception model (blue) that compresses it into an embedding z_t. The memory model takes the current latent representation z_t and uses the historical context to refine the state into $\hat{z}_t$. These embeddings could either be used to train a control policy (orange) or to reconstruct the Bird's Eye View (BEV) for planning (grey). Both utilities result in an action command a_t.

In our work, we employ such an approach with a new pre-training objective (to reconstruct BEV maps from FPV inputs), which we show provides very good generalization for downstream robotics tasks. Since learning representations does not involve any dynamics, any navigation dataset consisting of FPV-BEV could be used to pretrain the encoder. By training a Vision encoder using a large aggregated dataset, this could be a comparable alternative to the current ViT's (18) used for Robotics.

Bird's Eye View (BEV) based representation allows for a compact representation of the scene, invariant to any texture changes, scene variations, occlusions or lightning differences in an RGB image. This makes for an optimal representation for *PointGoal Navigation*. Furthermore, it is one of the most efficient and lightweight form of information, since the BEV maps (occupancy maps) that we use are binary. For example, the corresponding BEV image of an 1MB FPV image is around 0.5KB. Some works estimate BEV maps from RGB images, such as (14), (17) and (19). However, these map predictions from FPV images are typically only evaluated for visual tasks, with a lack of evidence that BEV-based representations can be useful for robotic tasks. Furthermore, (1) have shown that reconstruction-based methods like VAE (11) perform close to Random encoders. Incorporating these representations as inputs for training downstream models for robotic tasks to ensure their compatibility indeed is challenging. Our pretraining approach not only allows for learning visual representations that are optimal for robotic tasks, but also allows these representations to reconstruct the corresponding BEV map. Together, they allow the lightweight policy model to efficiently learn the task through these representations.

Recurrent world-models. (7) introduces a novel approach to RL, incorporating a vision model for sensory data representation and a memory model for capturing

temporal dynamics, all of which collectively improve agent performance. Apart from the advantages of pertaining each module, some of the modules in this architecture can be frozen after learning the representation of the environment, paving the way for more efficient and capable RL agents.

We propose a novel training regime and develop a perception model pre-trained on a large simulated dataset to translate FPV-based RGB images into embeddings that align with the representations of the corresponding BEV images. Along with that, we upgrade the existing world models framework using novel model-based *Temporal State Checking (TSC)* and *Anchor State Checking (ASC)* methods that add robustness to the navigation pipeline when transferred to the real world. We release the code for pre-training, RL training and ROS-based deployment of our system on a real-world robot, FPV-BEV dataset and pre-trained models. With the above contributions, we hope move closer towards open-sourcing a robust Visual Navigation system that uses pre-trained models trained on large datasets and simulations for efficient representation learning.

3 Proposed Method

For an autonomous agent to navigate using camera imagery, we use a simple system that consists of a perception model and a control model as shown in 2. The perception model takes input observation o_t and outputs an embedding z_t that is then passed on to the policy, as part of the control model to output an action vector a_t, throttle and steer. We first outline the perception model, with the objective of efficiently learning compact intermediate representations compatible with downstream policy learning, solely from a sequence of observations from the simulator. We then describe our second contribution, which involves the enhancement of the robustness and stability of the predictions during real-world evaluation.

3.1 Perception model

When training the perception model, we focus on 3 main principles. Firstly, z_t, the embedding vector should always be consistent with the BEV reconstruction. Secondly, BEV images must be represented in a continuous latent space that has smooth temporal transitions to similar BEV images. Finally, the perception model must efficiently utilize an unlabelled sequence of images as an expert video portraying optimal behaviour. This would also allow for unsupervised training/fine-tuning of the model using real-world expert videos, which we leave for future work.

The perception model consists of a *ResNet-50* (8) that is tasked with processing the observation o_t obtained from an RGB camera, with the primary objective of comprehending the environmental context in which the robot operates, and compresses o_t into a consistent intermediate representation, z_t, which when decoded through a BEV decoder, outputs a BEV image x_t. Our choice for BEV observations is rooted in their capacity to convey the surrounding roadmaps

with minimal information redundancy. To learn such representations from a set of FPV and corresponding binary BEV images, prior methods (14) train a *Variational Autoencoder (VAE)* (11) to encode an RGB image o_t that is decoded using $z_t \in \mathbb{R}^{B \times d}$, where B is the batch size and d is the embedding dimension. Given that we have batches $\mathbf{x}$ (BEV predictions through the model logits) and $\mathbf{y}$ (ground-truth BEV observations), we could then optimize the following reconstruction loss $\mathcal{L}_R$:

$$\mathcal{L}_R = -[\mathbf{y} \cdot log(\mathbf{x}) + (1 - \mathbf{y}) \cdot log(1 - \mathbf{x})] \tag{1}$$

Using the above loss, the VAE Encoder will learn to embed the FPV observations $\mathbf{o}$ into $\mathbf{z}$ that will be reconstructed by the decoder to their corresponding BEV outputs/reconstructions $\mathbf{x}$, and $\mathbf{y}$ being the corresponding ground-truth BEV outputs. Additionally, KL (*Kullback Leibler*) divergence forces the embeddings, to be within a Gaussian distribution of zero-mean and unit-covariance, that allows for smooth interpolation. The representations learnt by VAE would embed 2 FPV observations that are very similar, for example, 2 straight roads, but a have slight variation in the angle to be closer, than a straight road and an intersection. The following is the loss function used to train a VAE baseline.

$$\mathcal{L}_{ELBO} = \mathcal{L}_R + \beta \cdot KL[\mathcal{N}(\mu, \sigma^2) \,||\, \mathcal{N}(0, 1)] \tag{2}$$

Although, the above ELBO loss would allow the model to learn appropriate representations for understanding the observation, these representations do not capture the temporal understanding of the task. Typically, representations for robotics embed observations in such a way to make it easier for the policy to learn the behaviour of an objective quickly and efficiently. One of the earliest methods for self-supervised learning, Time-Contrastive Networks (22) disambiguates temporal changes by embedding representations closer in time, closer in the embedding space and farther otherwise by optimizing the following loss function, which is used in the Time-contrastive learning (TCN) baseline.

$$\mathcal{L}_{InfoNCE} = \mathbb{E}_{\mathbf{z}^{ps}} \left[-\log \frac{\mathcal{S}_\phi \left(\mathbf{z}^{an}, \mathbf{z}^{ps} \right)}{\mathbb{E}_{\mathbf{z}^{ng}} \mathcal{S}_\phi \left(\mathbf{z}^{an}, \mathbf{z}^{ng} \right)} \right] \tag{3}$$

In the above function, $\mathbf{z^{an}}$, $\mathbf{z^{ps}}$ and $\mathbf{z^{ng}}$ are a batch of embeddings corresponding to anchors, positives and negatives and $\mathcal{S}_\phi$ is the similarity metric of the embeddings from the encoder f_ϕ. For a given single observation sample o_t, the embedding obtained as an anchor z^{an}, we uniformly sample a frame within a temporal distance threshold d_{thresh} to obtain z^{ps} at timestep $t + \delta$ and z^{ng}, anywhere from $t + \delta$ to the end of the episode. However, recently (16) has shown that in-domain embeddings learnt by TCN are discontinuous, leading to sub-optimal policies. To alleviate this problem, we also add the reconstruction loss $\mathcal{L}_R$ that enhances the stability of the training process, and helps learn better representations. To achieve the FPV-BEV translation using our method, we optimize the model parameters using the following contrastive with reconstruction loss $\mathcal{L}_{CR}$ for image encoding.

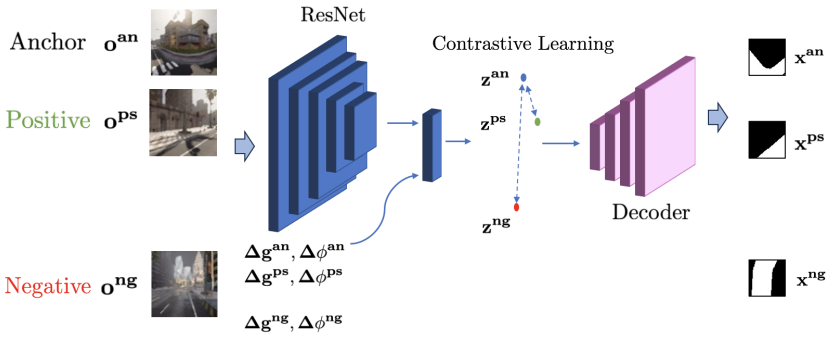


Fig. 3. Training pipeline for the perception model. (a) During the training phase, the ResNet model is trained using a set of temporal sequences, consisting of pairs of input (FPV images $\mathbf{o}$, displacement $\mathbf{\Delta g}$ and orientation to goal $\mathbf{\Delta}\phi$) and output (BEV images $\mathbf{x}$) from the simulator. Our contrastive loss embeds positives $\mathbf{z_{ps}}$ closer to the anchor $\mathbf{z_{an}}$ and negatives $\mathbf{z_{ng}}$ farther away. (b) In the bottom, we pictorially show the input embeddings $\mathbf{z_t}$ from FPV images, actions $\mathbf{a_t}$ and the output $\mathbf{z_{t+1}}$ that is used to train the memory module.

$$\mathcal{L}_{CR} = \mathcal{L}_R + \beta \cdot \mathcal{L}_{InfoNCE} \tag{4}$$

In the above loss function, β balances the reconstruction with the contrastive loss, since the model optimizes the reconstruction loss slower than the contrastive loss. Using the above loss function, the model learns more temporally continuous and smoother embeddings as it constrains the proximity of the embeddings not only using the contrastive learning loss but also based on the BEV reconstructions.

3.2 Temporal model with Robustness modules

To enhance the robustness of the perception model and transfer it to the real world setting, we implemented an additional model in the pipeline. Fig. 4 shows our proposed method of robustness enhancement. This involves the integration of an LSTM, functioning as a *Memory* model. The LSTM was trained on sequences $\{\langle o_j, a_j \rangle\}_{j=0}^{j=T}$ gathered from sequences $\{\mathcal{T}_0, \mathcal{T}_1, .., \mathcal{T}_n\}$ in the simulator. The primary outcome of this Memory model is to effectively infuse historical context $\{\langle z_j, a_j \rangle\}_{j=0}^{j=T}$ into the prediction of $\hat{z}_t$, which forms a candidate of z_t, and enhancing the robustness of the perception module when confronted with the unseen real-world data.

$$\hat{z}_t \sim P(\hat{z}_t \mid a_{t-1}, \hat{z}_{t-1}, h_{t-1}) \tag{5}$$

where $a_{t-1}, \hat{z}_{t-1}, h_{t-1}$ respectively denotes action, state prediction at the previous timestep, and historical hidden state at the time step $t-1$. $\hat{z}_t$ is the

latent representation that is given as an input to the policy. We optimize M with the below loss function:

$$\mathcal{L}_M = -\frac{1}{T}\sum_{t=1}^{T}\log(\sum_{j=1}^{K}\theta \cdot \mathcal{N}(z_t|\mu_j, \sigma_j)) \tag{6}$$

where $\{T, K, \theta_j, \mathcal{N}(z_t|\mu_j, \sigma_j)\}$ is, respectively, the training batch size, number of Gaussian models, Gaussian mixture weights with the constraint $\sum_{j=1}^{K}\theta_j = 1$, and the probability of ground truth at time step t conditioned on predicted mean μ_j and standard variance σ_j for Gaussian model j. Note this this is the same loss objective used in *Mixture Density Network RNN* (MDN-RNN) (7).

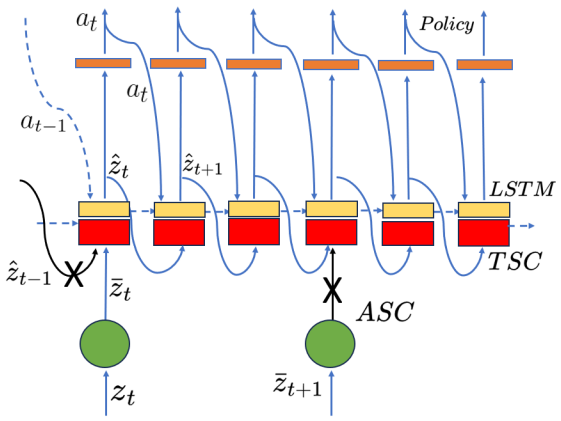
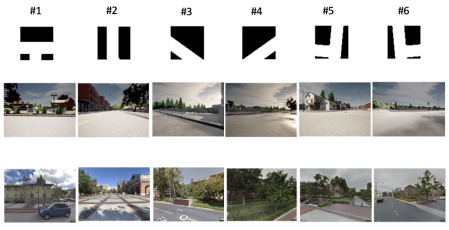


Fig. 4. Robustness enhancement using Memory module. *TSC* (red) only takes input from the representation z_t when it comes with a high confidence score. Otherwise, it takes the previous prediction by the LSTM $\hat{z}_{t-1}$ as interpolation. *ASC* (green) improves the representation of the incoming observation by making it in-domain. The crosses above correspond to rejecting the precepts and using the model's state prediction as the current state.

Nonetheless, it is noteworthy that z_t that is obtained from the ResNet-50 may be slightly distinct from the latent distribution of BEV images when the perception model is applied to real-world observations o_t, potentially impacting the performance of the LSTM and the policy. To mitigate this concern, we collected a dataset $\mathcal{R}$ comprising of the BEV-based latent embeddings $s \in \mathcal{R}$ of 1439 FPV images which we define as the *BEV anchors*. In practice, upon obtaining the output vector z_t from the ResNet-50, we measure its proximity to each $s \in \mathcal{R}$, subsequently identifying the closest match. We replace z_t with the identified anchor embedding $\bar{z}_t$, ensuring that both the LSTM and the policy consistently uses the pre-defined BEV data distribution. We pass $\bar{z}_t$ as an input to the LSTM, along with the previous action a_{t-1} to get the output $\hat{z}_{t+1}$. Again,

we find the closest match $\hat{s}_t \in \mathcal{R}$ for $\hat{z}_t$. We call this module *Anchor State Checking (ASC)*:

$$\bar{z} = \arg\min_{s \in S} \|z - s\| \tag{7}$$



	#1	#2	#3	#4	#5	#6	SR
Simulation	31	33	29	31	34	22	
RN Classifier	58.1	75.8	69.0	**96.8**	29.4	68.2	0.71
Ours	**90.3**	**93.9**	**82.8**	90.3	**82.4**	**81.8**	**0.73**
Real-World	37	103	32	37	39	37	
RN Classifier	47.0	43.8	54.8	91.4	9.3	31.8	0.46
Ours	**89.2**	**69.9**	**59.4**	**91.9**	**35.9**	**56.8**	**0.65**

Fig. 5. Out-of-domain and real-world evaluation We constructed two 6-class validation datasets: one from the simulator (upper-portion in the table) and another from real-world street-view data (lower-portion). The values in the header rows correspond to number of data samples. Each class corresponds to the BEV images shown above. We specify accuracies for each class. Along with that, we also specify the success rate (SR) of the agent, when the encoder is deployed for real-world visual navigation. Our method outperformed the ResNet classifier (baseline) on both the unseen simulation dataset, the real-world validation dataset and real-world navigation as shown above.

We also utilize the LSTM model for rejecting erroneous predictions by the ResNet-50, further enhancing the system's robustness against noise. If the processed prediction $\bar{z}_t$ from the perception model is estimated with confidence score τ_t, obtained from either cosine-similarity or MSE, below a predefined threshold ρ, we deliberately discard $\bar{z}_t$ and opt for $\hat{z}_t$. In such instances, we resort to the output of the LSTM at the previous time-step. This module is known as *Temporal State Checking (TSC)*:

$$\hat{z}_t = \begin{cases} \bar{z}_t, & \tau_t \geq \rho, \\ \hat{z}_{t-1}, & \tau_t < \rho. \end{cases} \tag{8}$$

Apart from adding robustness to the system using TSC, the utilization of the Memory model also serves as the crucial purpose of performing interpolation for the robots state in instances where actual observations o_t are delayed, ensuring the continuity and reliability of the entire system. There is often a notable discrepancy in the update frequencies between control signals and camera frames, since control signals often exhibit a significantly higher update rate (50Hz) compared to the incoming stream of camera frames (15Hz). Values mentioned in

brackets is in regards to our setup. This is also beneficial in the case of the recent large vision-language models like RT-X (4) that could solve many robotic tasks, but with a caveat of operating at a lower frequency, typically around 5Hz.

4 Experimental platform and setup

To leverage the extensive prior knowledge embedded in a pre-trained model, we opt to train a ResNet-50 (8) model after initializing with ImageNet pre-trained weights on a large-scale dataset containing FPV-BEV image pairs captured in the simulator. We collected the train dataset from the CARLA simulator to train both the Perception and the Memory model. Along with that, we also collected the validation and the test datasets from 2 different real-world sources. Following are the details on the collected datasets.

	Class#1				Class#2				Class#3				Class#4				Class#5				Class#6				**MEAN**		
	#	ACC	MSE	CE	#	ACC	MSE	CE	#	ACC	MSE	CE	#	ACC	MSE	CE	#	ACC	MSE	CE	#	ACC	MSE	CE	**ACC**	**MSE**	**CE**
Baseline	860	**47**	0.17	2.12	1378	39	0.29	6.58	1506	**82**	0.2	3.45	1106	**81**	0.18	1.95	10	0	**0.13**	2.89	140	0	0.22	2.16	41.5	0.19	3.19
Ours (ASC)	860	22	**0.16**	**0.71**	1378	**81**	**0.14**	**0.45**	1506	64	**0.12**	**0.72**	1106	73	**0.13**	**0.49**	10	0	0.17	**2.19**	140	**42**	0.22	**1.49**	**47.0**	**0.15**	**1.0**
Baseline	657	33	0.23	1.38	1853	64	0.34	9.07	1214	**85**	0.22	4.84	326	**79**	0.39	10.47	640	19	0.46	11.25	310	21	0.4	9.06	45	2.4	9.35
Ours (ASC)	657	**52**	**0.11**	**0.80**	1853	**75**	**0.13**	**1.37**	1214	42	**0.15**	**2.16**	326	76	**0.14**	**1.62**	640	**58**	**0.35**	**6.29**	310	**28**	**0.31**	**3.93**	**55.1**	**0.19**	**2.69**
Baseline	1644	20	0.21	1.54	1287	49	0.31	6.36	1003	**57**	0.18	2.74	973	**94**	0.18	2.24	0	-	-	-	93	0	0.19	**1.65**	44.0	0.21	2.9
Ours (ASC)	1644	**22**	0.21	**0.74**	1287	**71**	**0.13**	**0.44**	1003	51	**0.09**	**0.60**	973	78	**0.12**	**0.74**	0	-	-	-	93	**15**	**0.15**	1.70	**47.4**	**0.13**	**0.84**
Baseline	679	33	0.21	0.13	2268	69	0.30	0.74	846	**88**	0.22	0.38	1087	**90**	0.19	0.24	63	20	0.30	0.91	57	10	0.34	0.63	51.6	0.26	0.5
Ours (ASC)	679	**61**	**0.10**	**0.07**	2268	**94**	**0.08**	**0.07**	846	58	**0.07**	**0.08**	1087	50	0.19	**0.06**	63	**79**	**0.11**	**0.13**	57	**45**	**0.07**	**0.16**	**64.5**	**0.1**	**0.09**
Ours (ASC)	35	3	0.30	0.50	1079	73	0.40	**0.81**	69	28	0.30	0.90	131	**91**	0.30	**0.42**	0	-	-	-	177	7	**0.40**	5.21	40.4	0.34	1.5
Ours (ASC+TSC)	35	**77**	**0.18**	**0.32**	1079	**78**	**0.30**	1.13	69	**81**	**0.21**	**0.77**	131	88	**0.23**	0.77	0	-	-	-	177	**50**	0.41	**1.16**	**74.8**	**0.26**	**0.82**
Ours (ASC)	50	44	0.17	**3.24**	1511	**70**	0.32	4.38	151	28	0.20	**0.60**	198	37	0.30	1.40	0	-	-	-	0	-	-	-	44.7	0.24	2.4
Ours (ASC+TSC)	50	**90**	**0.14**	3.89	1511	69	**0.29**	**4.21**	151	**60**	**0.19**	0.69	198	**84**	**0.18**	**0.44**	0	-	-	-	0	-	-	-	**75.7**	**0.2**	**2.3**
Ours (ASC)	253	81	**0.10**	0.70	1914	**89**	0.33	4.49	33	12	0.20	1.00	190	42.6	0.20	0.90	0	-	-	-	0	-	-	-	56.1	0.2	1.7
Ours (ASC+TSC)	253	**97**	0.13	**0.50**	1914	84	**0.17**	**2.40**	33	**88**	**0.17**	**0.85**	190	**88**	**0.19**	**0.70**	0	-	-	-	0	-	-	-	**89.2**	**0.16**	**1.1**
Ours (ASC)	0	-	-	-	974	83	0.32	4.42	17	29	0.26	5.22	0	-	-	-	88	90.9	0.24	4.95	78	26	0.37	9.83	57.2	0.29	6.1
Ours (ASC+TSC)	0	-	-	-	974	83	**0.29**	**3.99**	17	**94**	**0.19**	**3.34**	0	-	-	-	88	**92.0**	**0.23**	**4.62**	78	**59**	**0.19**	**3.42**	**82.0**	**0.22**	**3.8**

Fig. 6. Ablation experiments on the Test Dataset. Classes in the above table have the same correspondences as the classes in Fig. 5. Each double-row corresponds to a data sequence. We demonstrate that our approach not only attains high ACC (accuracy), but also provides a more granular BEV representation compared to the naive classifier, as indicated by the MSE (Mean Squared Error) and CE (Cross-Entropy) metrics. **(a). In the upper portion of the table**, we assessed our method independently of the LSTM on an unseen temporal sequence from the simulator, contrasting it with the baseline CNN classifier. **(b) In the lower portion**, we compared the performance of system with and without LSTM on a real-world data sequence. Note that dashes in the table indicate the absence of a class in the respective sequence. We compute the mean values for each row as shown in the last column.

4.1 Experimental platform

For evaluating Zero-shot real-world transfer, we built a hardware apparatus, which is a Non-Holonomic, Differential-drive robot (*Beobotv3*) for the task of

visual navigation. Our system is implemented using the *ROS* (Robotic Operating System) middleware and uses a *Coral EdgeTPU*, which is an ASIC chip designed to run CNN models for edge computing for all the compute. We used this EdgeTPU to run the forward inference of the ResNet-50 through a ROS node.

The CARLA simulator had been primarily tailored to self-driving applications, that use *Ackermann steering*; we further developed an existing differential drive setup using *Schoomatic* (15) and upgraded the CARLA simulator. We find this necessary because our real-world hardware system is based on differential-drive and to enable seamless transfer without any Sim2Real gap in the control pipeline, both the control systems need to have similar dynamics. In response to this limitation, Luttkus (15) designed a model for the integration of a differential-drive robot into the CARLA environment. Building upon their work, we undertook the development of a version of CARLA simulator catering to differential-drive robots for reinforcement learning, subsequently migrating it into the newly introduced CARLA *0.9.13*.

4.2 Data collection

Train dataset from CARLA simulator Within the CARLA simulator, we have access to the global waypoints along various trajectories. To allow more diversity, we randomly sampled a range of different orientations and locations. Leveraging this setup, we facilitated the generation of a large dataset of FPV-BEV images. We augmented the simulator's realism by introducing weather randomization and non-player traffic into the simulated environment.

Validation dataset from Google Street View Using the Google Street View API, we obtained all the panoramic images from various locations on the USC campus. The panoramic images were segmented with a Horizontal Field of View (FoV) of 90 degrees and are manually segregated into 6 different classes as shown in Fig. 5. We then manually assigned a prototypical BEV image to each of the 6 classes. The validation dataset does not have any temporal sequencing and is primarily focused on having a broader and more uniform data distribution across all the classes. Due to these reasons, this dataset becomes an optimal choice for evaluating the perception model.

Test dataset from Beobotv3 To evaluate the quality of representations estimated by the entire system, we record a video sequence using a mobile robot. More precisely, we recorded a set of 5 *ROSBag* sequences at different locations of the USC campus. Later, we labelled all the frames in a *ROSBag* sequence, similar to the above paragraph. However, unlike the validation set, the test dataset has temporal continuity, which helps us judge the entire navigation system.

5 Evaluation and Results

Through our experiments, we aim to answer the following questions in regards to our proposed method.

	CPU	Edge-TPU	GPU
Policy Learning (RL)			
Encoder+Policy	562	50	24
Encoder+ASC+Policy	566	55	37
Encoder+ASC+TSC+Policy	578	67	47
Planning			
Encoder+Decoder	697	82	37
Encoder+ASC+Decoder	702	86	41
Encoder+ASC+TSC+Decoder	719	98	55

Fig. 7. Comparison of runtime. Computation costs (runtime in *milliseconds*) of each module in the navigation system for policy learning and planning are shown above.

1. *How good are the representations obtained from the pretrained model for learning to navigate using online RL?*
2. *How well can we plan using the BEV reconstructions from the pretrained model?*
3. *Does contrastive learning help learning good representations compared to an auxiliary task?*
4. *What are the performance benefits by adding ASC, TSC, and both?*
5. *How efficient and optimal is the navigation system when transferred to the Real-world setup?*

Policy Learning. We performed RL experiments by deploying the frozen pretrained encoder and training a 1-layer policy in the CARLA simulator Fig. 8. The task for the agent is to navigate to a goal destination using an RGB image $(o_t,\ \Delta g_t, \phi_t)$. We accomplished this by training a policy employing the PPO algorithm (21). The design of the reward function is rooted in proportionality to the number of waypoints the robot achieves to the designated goal point. In each timestep, the policy receives the current embedding of the observation z_t concatenated with the directional vector pointing towards the waypoint tasked with producing a pair of (throttle, steer) values. We compared our method with VAE (reconstructing only the BEV image; Eqn. 2), TCN (trained using Eqn. 3), Random (Randomly initialized encoder and frozen), CLIP-RN50 (18). Note that, many of the prior works (1; 3) have shown that randomly initialized and frozen encoders do learn decent features from an observation.

Planning We use the TEB planner (20) to compute the action using an occupancy map (BEV reconstruction) to perform a task. Typically, occupancy map-based planners like TEB, use LiDAR data to compute the map of the environment and estimate a plan to perform the task, but in our case, we reconstruct the occupancy map using embedding obtained from RGB inputs. These maps are straightforward to compute in the case of our method and the VAE baseline, since these methods use a decoder. For the other baselines like the Random, CLIP and TCN encoder, we freeze the encoder and train the decoder to upsample the embeddings to estimate the BEV reconstruction. The results obtained for the planning task are shown in Fig. 8 as dotted lines. The success rates correspond to percentage of rollouts that achieve the goal destination.

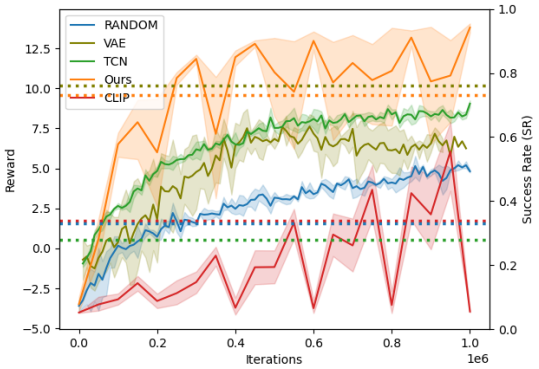


Fig. 8. Policy learning and Planning experiments on navigation task using pretrained representations. Using a pretrained *ResNet* encoder, we compare our method with different baselines. The training curves are obtained when we train a 1-layer policy, using RL, that takes the embeddings from the frozen encoder. The x and y axis corresponds to iterations and the cumulative reward, with the shaed regions showing the 95% confidence intervals. We also perform planning experiments, where the BEV reconstructions are used to navigate to the goal, as shown the by the success rate (SR), through the dotted lines corresponding to each method.

Quantitative Analysis We evaluated the performance of our ResNet-50 model using the real-world validation dataset to evaluate the out-of-distribution capabilities of the models and the results are shown in Table 5. The performance of our perception model on both simulation and real-world dataset are compared to the baseline, which is a 6-way ResNet-50 classifier. Our perception model identifies the closest matching class for the output embedding. The baseline is a ResNet-50 model trained on a 6-class training dataset comprising 140,213 labelled FPV images. This proves that contrastive learning using BEV prediction enables better generalization to out-of-domain data and better transfer from simulator to real (Table 5).

Ablation Experiments for state checking Following a similar approach, we used the Test dataset to evaluate the entire system. Apart from the accuracy also used Cross entropy (CE) and Mean Square error (MSE) to judge the quality of reconstructions by the LSTM model. These results are shown in Table 6. Similar to the above experiments, we also used data from the unseen Town from the CARLA simulator to asses the predictions of our system, as shown in the top half of the Figure. 6. The metrics presented in this table exhibit a slight decrease compared to Table 5. This can be attributed to the increased presence of abnormal observations and higher ambiguity between classes within the time-series data obtained from the robot, as opposed to the manually collected and labelled dataset in the validation dataset.

Evaluation on a Real-world system We perform experiments on a Real-world robot, where the agent is tasked with navigating to a given destination location, using the pretrained *ResNet* encoder and the trained policy in the Carla simulator. Success rates (SR) for planning experiments for our model are shown in Fig. 5. For both policy learning and planning, we specify the computation costs in Table. 7. As mentioned before, the success rates correspond to the percentage of rollouts that achieve the goal destination.

6 Discussion and Future work

In this paper we proposed a robust navigation system that is trained entirely in a simulator and frozen when deployed. We learn compact embeddings of an RGB image for Visual Navigation that are aligned with temporally closer representations and reconstruct corresponding BEV images. By decoupling the perception model from the control model, we get the added advantage of being able to pretrain the encoder using a set of observation sequences irrespective of the robot dynamics. Our system also consists of a memory module that enhances the robustness of the navigation system and is trained on an offline dataset from the simulator. Although our experiments in this paper are limited to data obtained through the simulator, one of the primary advantages of our methods is the ability to use additional simulator/real-world FPV-BEV datasets by aggregating with the current dataset. We leave this for future work.

7 Acknowledgements

This work was supported by the National Science Foundation (award 2318101), C-BRIC (one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA) and the Army Research Office (W911NF2020053). The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

References

[1] Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M., Hjelm, R.D.: Unsupervised state representation learning in atari. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. pp. 8766–8779 (2019), https://proceedings.neurips.cc/paper/2019/hash/6fb52e71b837628ac16539c1ff911667-Abstract.html

[2] Arndt, K., Hazara, M., Ghadirzadeh, A., Kyrki, V.: Meta reinforcement learning for sim-to-real domain adaptation. CoRR abs/1909.12906 (2019), http://arxiv.org/abs/1909.12906

[3] Burda, Y., Edwards, H., Pathak, D., Storkey, A.J., Darrell, T., Efros, A.A.: Large-scale study of curiosity-driven learning. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=rJNwDjAqYX

[4] Collaboration, O.X.: Open x-embodiment: Robotic learning datasets and RT-X models. CoRR abs/2310.08864 (2023), https://doi.org/10.48550/arXiv.2310.08864

[5] Dosovitskiy, A., Ros, G., Codevilla, F., López, A.M., Koltun, V.: CARLA: an open urban driving simulator. In: 1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings. Proceedings of Machine Learning Research, vol. 78, pp. 1–16. PMLR (2017), http://proceedings.mlr.press/v78/dosovitskiy17a.html

[6] Ge, Y., Li, Y., Wu, D., Xu, A., Jones, A.M., Rios, A.S., Fostiropoulos, I., Wen, S., Huang, P., Murdock, Z.W., Sahin, G., Ni, S., Lekkala, K., Sontakke, S.A., Itti, L.: Lightweight learner for shared knowledge lifelong learning. CoRR abs/2305.15591 (2023), https://doi.org/10.48550/arXiv.2305.15591

[7] Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada. (7), pp. 2455–2467, http://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution

[8] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 770–778. IEEE Computer Society (2016), https://doi.org/10.1109/CVPR.2016.90

[9] Josifovski, J., Malmir, M., Klarmann, N., Zagar, B.L., Navarro-Guerrero, N., Knoll, A.C.: Analysis of randomization effects on sim2real transfer in reinforcement learning for robotic manipulation tasks. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Ky-

oto, Japan, October 23-27, 2022. pp. 10193–10200. IEEE (2022), https://doi.org/10.1109/IROS47612.2022.9981951

[10] Kaspar, M., Munoz Osorio, J.D., Bock, J.: Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization. arXiv e-prints arXiv:2002.11635 (Feb 2020)

[11] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), http://arxiv.org/abs/1312.6114

[12] Lekkala, K., Itti, L.: Shaped policy search for evolutionary strategies using waypoints*. In: IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021. pp. 9093–9100. IEEE (2021), https://doi.org/10.1109/ICRA48506.2021.9561607

[13] Lekkala, K.K., Mittal, V.K.: Artificial intelligence for precision movement robot. In: 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN). pp. 378–383. IEEE (2015)

[14] Lu, C., van de Molengraft, M.J.G., Dubbelman, G.: Monocular semantic occupancy grid mapping with convolutional variational encoder–decoder networks. IEEE Robotics and Automation Letters 4(2), 445–452 (apr 2019), https://doi.org/10.1109%2Flra.2019.2891028

[15] Luttkus, L., Krönes, P., Mikelsons, L.: Scoomatic: Simulation and validation of a semi-autonomous individual last-mile vehicle. In: Sechste IFToMM D-A-CH Konferenz 2020: 27./28. Februar 2020, Campus Technik Lienz. vol. 2020 (Feb 21, 2020), https://nbn-resolving.org/urn:nbn:de:hbz:464-20200221-092453-2

[16] Ma, Y.J., Sodhani, S., Jayaraman, D., Bastani, O., Kumar, V., Zhang, A.: VIP: towards universal visual reward and representation via value-implicit pre-training. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Rwanda, May 1-5, 2023. OpenReview.net (2023), https://openreview.net/pdf?id=YJ7o2wetJ2

[17] Pan, B., Sun, J., Leung, H.Y.T., Andonian, A., Zhou, B.: Cross-view semantic segmentation for sensing surroundings. IEEE Robotics and Automation Letters 5(3), 4867–4873 (jul 2020), https://doi.org/10.1109%2Flra.2020.3004325

[18] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR (2021), http://proceedings.mlr.press/v139/radford21a.html

[19] Reiher, L., Lampe, B., Eckstein, L.: A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view. In: 23rd IEEE International Conference on Intelligent Transportation Systems, ITSC 2020,

Rhodes, Greece, September 20-23, 2020. pp. 1–7. IEEE (2020), https://doi.org/10.1109/ITSC45102.2020.9294462

[20] Rösmann, C., Hoffmann, F., Bertram, T.: Integrated online trajectory planning and optimization in distinctive topologies. Robotics Auton. Syst. 88, 142–153 (2017), https://doi.org/10.1016/j.robot.2016.11.007

[21] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR abs/1707.06347 (2017), http://arxiv.org/abs/1707.06347

[22] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S.: Time-contrastive networks: Self-supervised learning from video. In: 2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018. pp. 1134–1141. IEEE (2018), https://doi.org/10.1109/ICRA.2018.8462891

[23] Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., Batra, D.: DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=H1gX8C4YPr