CARFF: Conditional Auto-encoded Radiance Field for 3D Scene Forecasting – Supplementary Material –

Jiezhi "Stephen" Yang[1*], Khushi Desai[2*], Charles Packer[3], Harshil Bhatia[4], Nicholas Rhinehart[3], Rowan McAllister[5], and Joseph E. Gonzalez[3]

[1] Harvard University MA 02138, USA
[2] Columbia University NY 10025, USA
[3] UC Berkeley CA 94720, USA
[4] Avataar.ai KA 560103, India
[5] Toyota Research Institute CA 94022, USA

1 Datasets

1.1 CARLA Datasets

A complete figure of the actor and ego configurations across scenes and the progression of timestamps for the Single-Scene Approaching Intersection, Multi-Scene Approaching Intersection, and the Multi-Scene Two Lane Merge is visualized in Fig. 6.

1.2 Hand-manipulation Dataset

We generate an additional hand-manipulation dataset involving a robotic manipulator engaged in probabilistic reaching tasks utilizing VUER fig(5). The experimental setup consists of two target objects placed on a table in a room: a Rubik's cube and a green tennis ball. The robotic hand's configurations during the reaching and grasping phases for both objects are illustrated in Fig. 6.

2 Implementation Details

2.1 Pose-Conditional VAE

Architecture: We implement PC-VAE on top of a standard PyTorch VAE framework. The encoder with convolutional layers is replaced with a single convolutional layer and a Vision Transformer (ViT) Large 16 (1) pre-trained on ImageNet (4). We modify fully connected layers to project ViT output of size 1000 to mean and variances with size of the latent dimension, 8. During training, the data loader returns the pose of the camera angle represented by an integer value. This value is one-hot encoded and concatenated to the re-parameterized encoder outputs, before being passed to the decoder. The decoder input size is increased to add the number of poses to accommodate the additional pose information.

* Core contributors

Fig. 1: PC-VAE encoder inputs, ground truth timestamps, and reconstructions for a CARLA dataset and Hand-manipulation dataset. The encoder input, I_c^t, among the other ground truth images I_c viewed from camera pose c at different timestamps, is reconstructed across a new set of poses c'' respecting timestamp t, generating $I_{c''}^t$. This is a full grid of the reconstructions.

PC-VAE Hyperparameters	
Latent Size	8
LR	0.004
KLD Weight Start	0.000001
KLD Weight End	$0.00001 - 0.00004$*
KLD Increment Start	50 epochs
KLD Increment End	80 epochs

Table 1: PC-VAE experimental setup and hyperparameters. The main hyperparameters in PC-VAE training on the three datasets are latent size, LR, and KLD weight. For KLD scheduling, the KLD increment start refers to the number of epochs at which the KLD weight begins to increase from the initial KLD weight. KLD increment end is the number of epochs at which the KLD weight stops increasing at the maximum KLD weight. The asterisk (*) marks the hyperparameter that is dataset-dependent.

Optimization: We utilize a single RTX 3090 graphics card for all our experiments. The PC-VAE model takes approximately 22 hours to converge using this GPU. During this phase, we tune various hyperparameters including the latent size, learning rate and KL divergence loss weight to establish optimal training tailored to our model (see Tab. 1). In order to optimize for the varied actor configurations and scenarios generated within the CARLA (2) simulator, we slightly adjust hyperparameters differently for each dataset.

The learning rate (LR) and KL divergence (KLD) weight are adjusted to find an appropriate balance between the effective reconstruction of pose conditioning in the latent space, and the regularization of latents. Regularization pushes the latents toward Gaussian distributions and keeps the non-expressive latents in an over-parameterized latent space to be standard normal. This stabilizes the sampling process and ensures stochastic behavior of latent samples in case of occlusion. To achieve this balance, we use a linear KLD weight scheduler, where the weight is initialized at a low value for KLD increment start epoch (see Tab. 1). This allows the model to initially focus on achieving highly accurate conditioned reconstructions. The KLD weight is then steadily increased until KLD increment end epoch is reached, ensuring probabilistic behavior under partial observability.

2.2 Mixture Density Network

With the POMDP, the mixture density network (MDN) takes in the mean and variances of the latent distributions of the current belief state $q_\phi(z_{t-1}|I_c^{t-1})$ and outputs the next belief state's estimated posterior distribution. To better model the uncertainty of the predicted belief state distribution, the output is a mixture of Gaussian $q'_\phi(z_t|I_c^{t-1})$ modeled through a multi-headed MLP.

Architecture: The shared backbone simply contains 2 fully connected layers and rectified linear units (ReLU) activation with hidden layer size of 512. Additional heads with 2 fully connected layers are used to generate μ_i and σ_i^2. The mixture

weight, π_i, is generated from a 3 layer MLP network. We limit the number of Gaussians, $K = 2$.

Optimization: We train our network for $30,000$ epochs using the batch size of 128 and an initial LR of 0.005, and apply LR decay to optimize training. This takes approximately 30 minutes to train utilizing the GPU. During training, the dataloader outputs the means and variances at the current timestamp and indexed view, and the means and variances for the next timestamp, at a randomly sampled neighboring view. This allows the MDN to learn how occluded views advance into all the possible configurations from potentially unoccluded neighboring views, as a mixture of Gaussian.

At each iteration, the negative log-likelihood loss is computed for 1000 samples drawn from the predicted mixture of distributions $q'_\phi(z_t|I_c^{t-1})$ with respect to the ground truth distribution $q_\phi(z_t|I_c^t)$. While the MDN is training, additional Gaussian noise, given by $\epsilon \sim \mathcal{N}(0, \sigma^2)$, is added to the means and variances of the current timestamp $t-1$, where $\sigma \in [0.001, 0.01]$. The Gaussian noise and LR decay help prevent overfitting and reduce model sensitivity to environmental artifacts like moving trees, moving water, etc.

2.3 NeRF

Architecture: We implement our NeRF as a decoder to our belief state to recover 3D observations utilizing an existing PyTorch implementation of Instant-NGP (3). We concatenate the latents to the inputs of two parts of the Instant-NGP architecture: the volume density network, $\sigma(\mathbf{x})$, for the density values, and the color network, $C(\mathbf{r})$, for conditional RGB generation. While the overall architecture is kept constant, the input dimensions of each network are modified to allow additional latent concatenation.

Optimization: Empirically, we observe that it is essential to train the NeRF such that it learns the distribution of scenes within the PC-VAE latent space. Using only pre-defined learned samples to train may run the risk of relying on non-representative samples. On the other hand, direct re-sampling during each training iteration in Instant-NGP may lead to delayed training progress, due to NeRF's sensitive optimization. In our optimization procedure, we use an LR of 0.002 along with an LR decay and start with pre-defined latent samples. Then we slowly introduce the re-sampled latents. We believe that this strategy progressively diminishes the influence of a single sample, while maintaining efficient training. Based on our observations, this strategy contributes towards Instant-NGP's ability to rapidly assimilate fundamental conditioning and environmental reconstruction, while simultaneously pushing the learning process to be less skewed towards a single latent sample.

3 GUI Interface

For ease of interaction with our inference pipeline, our NeRF loads a pre-trained MDN checkpoint, and we build a graphical user interface (GUI) using DearPyGUi

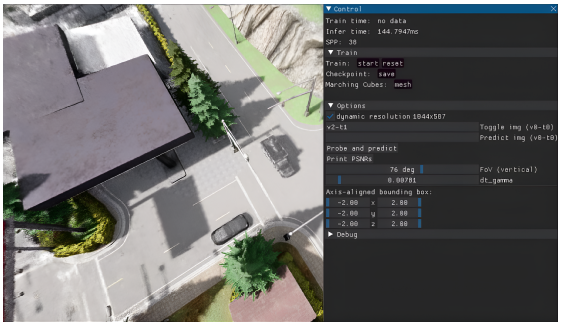


Fig. 2: NeRF graphical user interface. The GUI allows us to toggle and predict with an input image path. The probe and predict function probes the current location of the car and predicts the next. The screenshot is sharpened for visual clarity in the paper.

for visualization purposes. We implement three features in the GUI: (a) predict, (b) probe and predict, and (c) toggle.

Predict: We implement the function to perform prediction directly from a given image path in the GUI. We use the distribution $q_\phi(z_{t-1}|I_c^{t-1})$ from PC-VAE encoder, corresponding to the input image I_c^{t-1}, to predict the latent distribution for the next timestamp belief state $q'_\phi(z_t|I_c^{t-1})$. This process is done on the fly through the MDN. A sample from the predicted distribution is then generated and used to condition the NeRF. This advances the entire scene to the next timestamp.

Probe and predict: The sampled latent from the predicted distribution does not correspond to a singular distribution and hence we can not directly predict the next timestamp. To make our model auto-regressive in nature, we perform density probing. We probe the density of the NeRF at the possible location coordinates of the car to obtain the current timestamp and scene. This is then used to know the actual state sampled from the belief state probability distributions. The new distribution enables auto-regressive predictions using the predict function described above.

Toggle: The NeRF generates a scene corresponding to the provided input image path using learned latents from PC-VAE. This function tests the NeRF decoder's functionality with a given belief state. When the input image is a fully observable view (corresponding to a unknown belief state), the NeRF renders clear actor and ego configurations respecting the input. This allows us to visualize the scene at different timestamps and in different configurations.

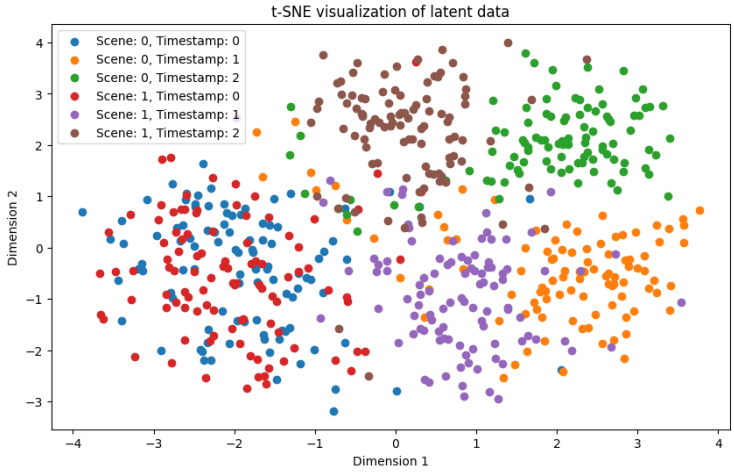


Fig. 3: Latent sample distribution clustering. The distributions of latent samples for the Multi-Scene Two Lane Merge dataset are separable through t-SNE clustering. In the figure, the clusters for *Scene 0, Timestamp 0* and *Scene 1, Timestamp 0* overlap in distribution because they represent the same initial state of the environment under dynamics uncertainty.

4 CARFF Evaluation

4.1 Pose-Conditional VAE

Reconstruction Quality: To analyze the reconstruction performance of the model during training, we periodically plot grids of reconstructed images. These grids consist of (a) randomly selected encoder inputs drawn from the dataset, (b) the corresponding ground truth images for those inputs at each timestamp at the same camera pose, and (c) reconstructed outputs at randomly sampled poses respecting the input scene and timestamp. An example reconstruction grid is provided in Fig. 1. The grid enables visual assessment of whether the model is capable of accurately reconstructing reasonable images using the encoder inputs, conditioned on the poses. This evaluation provides us with visual evidence of improvement in reconstruction quality. We also quantitatively analyze the progressive improvement of reconstruction through the average PSNR calculated over the training data (see Fig. 4).

The PC-VAE outputs in Fig. 1 only provides visual confirmation to assess the quality of the latents learned by PC-VAE. Utilization of the 3D decoder later in our method allows us to produce more high resolution visualizations of the scene that can be used for further downstream tasks.

Latent Space Analysis To assess the quality of the latents generated by PC-VAE, we initially use t-SNE plots to visualize the latent distributions as clusters. Fig. 3 shows that the distributions of the latent samples for the Multi-

Architectures	Train PSNR	SVM Accuracy	NV PSNR
Multi-Scene Approaching Intersection			
PC-VAE	**26.47**	**89.17**	**26.37**
PC-VAE w/o CL	26.20	83.83	26.16
Vanilla PC-VAE	25.97	29.33	25.93
PC-VAE w/o Freezing	24.82	29.83	24.78
PC-VAE w/ MobileNet	19.37	29.50	19.43
Vanilla VAE	26.04	14.67	9.84
Multi-Scene Two Lane Merge			
PC-VAE	**25.50**	**88.33**	**25.84**
PC-VAE w/o CL	24.38	29.67	24.02
Vanilla PC-VAE	24.75	29.67	24.96
PC-VAE w/o Freezing	23.97	28.33	24.04
PC-VAE w/ MobileNet	17.70	75.00	17.65
Vanilla VAE	25.11	28.17	8.49

Table 2: PC-VAE metrics and ablations across Multi-Scene datasets. CARFF's PC-VAE outperforms other encoder architectures across the Multi-Scene datasets in reconstruction and pose-conditioning.

Scene Two Lane Merge dataset are separable. While t-SNE is good at retaining nearest-neighbor information by preserving local structures, it performs weakly in preserving global structures. Therefore, t-SNE may be insufficient in capturing the differences in distributions for all our datasets.

Instead, we pivot to Support Vector Machine to perform a quantitative evaluation of the separability of the latents. We utilize a Radial Basis Function (RBF) kernel with the standard regularization parameter ($C = 1$). We perform 10-fold validation on the latents to calculate the accuracy as a metric for clustering. See Tab. 2 for the results.

Beyond separability, we analyze the recall and accuracy of the learned latents directly from PC-VAE under partial and full observations. This achieves very high accuracy even under a large number of samples while retraining decent recall, enabling downstream MDN training. (See Fig. 5)

For the additional Hand-manipulation Object Reaching dataset, we used a similar setup as detailed in Fig **??** with $n = 1$ to $n = 50$ samples from the MDN's predicted latent distributions from an potentially partially observable image input. Similar to the CARLA dataset results, we achieve an ideal margin of belief state coverage generated under partial observation (recall), and the proportion of correct beliefs sampled under full observation (accuracy).

4.2 Fully Observable Predictions

One of the tasks of the MDN is to forecast the future scene configurations under full observation. We quantitatively evaluate our model's ability to forecast future scenes by comparing bird's-eye views rendered from the NeRF with chosen

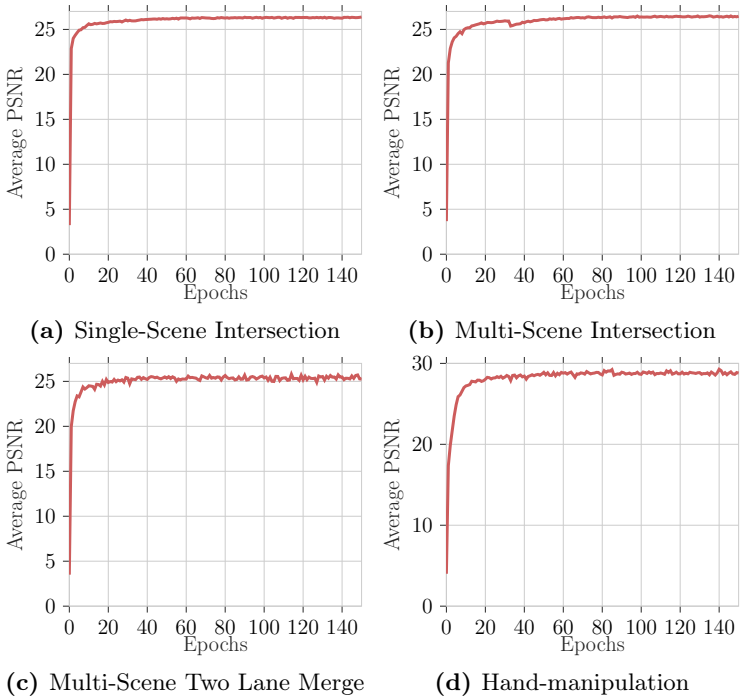
Single-Scene Approaching Intersection										
Result	I_{t_1}	I_{t_2}	I_{t_3}	I_{t_4}	I_{t_5}	I_{t_6}	I_{t_7}	I_{t_8}	I_{t_9}	$I_{t_{10}}$
$\tilde{I}_{t_1}$	**29.01**	5.97	6.08	6.52	6.44	6.03	6.31	6.36	6.26	6.28
$\hat{I}_{t_2}$	5.42	**27.51**	3.07	4.67	4.58	4.17	4.43	4.51	4.39	4.39
$\hat{I}_{t_3}$	6.06	2.81	**28.12**	4.47	4.68	4.19	4.05	4.61	4.47	4.52
$\hat{I}_{t_4}$	7.01	5.37	5.03	**29.40**	4.99	5.08	5.03	5.41	5.28	5.32
$\hat{I}_{t_5}$	6.87	5.2	4.93	5.00	**29.44**	4.53	4.46	5.19	5.05	5.09
$\hat{I}_{t_6}$	6.29	4.55	4.27	4.8	4.24	**29.02**	4.02	4.53	4.38	4.44
$\hat{I}_{t_7}$	6.76	5.05	4.76	5.31	5.14	4.36	**29.50**	4.50	4.86	4.93
$\hat{I}_{t_8}$	6.73	5.02	4.74	5.25	5.10	4.64	4.76	**29.46**	4.41	4.86
$\hat{I}_{t_9}$	6.75	5.00	4.70	5.23	5.07	4.64	4.85	4.52	**29.55**	4.42
$\hat{I}_{t_{10}}$	6.79	5.06	4.75	5.30	5.15	4.69	4.93	5.01	4.34	**29.55**

Multi-Scene Approaching Intersection						
Result	I_{t_1}	I_{t_2}	I_{t_3}	I_{t_4}	I_{t_5}	I_{t_6}
$\tilde{I}_{t_1}$	**28.10**	5.24	5.50	1.67	3.29	3.92
$\hat{I}_{t_2}$	5.23	**28.02**	6.11	4.70	3.21	4.84
$\hat{I}_{t_3}$	5.43	6.03	**27.97**	4.85	4.53	2.93
$\tilde{I}_{t_4}$	1.71	4.73	5.00	**28.26**	2.25	3.08
$\tilde{I}_{t_5}$	3.68	3.24	4.91	2.76	**28.21**	2.99
$\hat{I}_{t_6}$	4.02	4.91	3.27	3.13	2.61	**28.26**

Multi-Scene Two Lane Merge						
Result	I_{t_1}	I_{t_2}	I_{t_3}	I_{t_4}	I_{t_5}	I_{t_6}
$\tilde{I}_{t_1}$	**28.27**	5.31	6.41	**28.23**	4.77	5.42
$\tilde{I}_{t_2}$	5.22	**28.23**	5.17	5.27	2.91	4.01
$\hat{I}_{t_3}$	6.32	5.09	**28.14**	6.33	5.01	4.28
$\tilde{I}_{t_4}$	**28.27**	5.27	6.37	**28.23**	4.72	5.37
$\tilde{I}_{t_5}$	4.64	2.73	5.01	4.71	**28.08**	5.29
$\hat{I}_{t_6}$	5.32	4.02	4.32	5.33	5.34	**28.17**

Hand-manipulation Object Reaching						
Result	I_{t_1}	I_{t_2}	I_{t_3}	I_{t_4}	I_{t_5}	I_{t_6}
$\tilde{I}_{t_1}$	**30.71**	10.74	10.63	**30.71**	9.81	11.13
$\tilde{I}_{t_2}$	10.07	**30.32**	9.25	10.07	10.88	10.17
$\hat{I}_{t_3}$	9.98	9.11	**30.37**	9.98	10.78	10.02
$\tilde{I}_{t_4}$	**30.66**	10.69	10.58	**30.66**	9.77	11.09
$\tilde{I}_{t_5}$	8.19	10.21	10.1	8.19	**29.49**	9.99
$\hat{I}_{t_6}$	10.96	10.6	10.49	10.96	11.08	**30.67**

Table 3: Complete PSNR values for fully observable predictions for all CARLA datasets and the Hand-manipulation dataset. The table contains PSNR values between the ground truth images and either a toggled image (marked as $\tilde{I}_{t_i}$), or a predicted image from the NeRF decoder (marked as $\hat{I}_{t_i}$). Toggled or predicted images that correspond to the correct ground truth are bolded and have a extremely high PSNR value, indicating high fidelity results. The PSNR values for incorrect correspondances are replaced with the difference between the incorrect PSNR and the bolded PSNR associated with a correct correspondance.



(a) Single-Scene Intersection **(b)** Multi-Scene Intersection

(c) Multi-Scene Two Lane Merge **(d)** Hand-manipulation

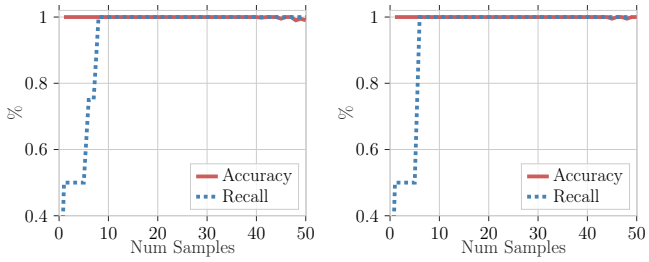
Fig. 4: Average train PSNR plot for all CARLA datasets and the hand-manipulation dataset. The plot shows the increase in average training PSNR of all images for each dataset, over the period of the training process.

ground truth images of the scene for the various timestamps (see Tab. 3). The values are calculated and displayed for all four datasets. In Tab. 3, images are marked as either toggled ($\tilde{I}_{t_i}$) or predicted ($\hat{I}_{t_i}$). Toggled images in the table cannot be predicted deterministically due to it being the first timestamp in the dataset, or the state of the previous timestamps across scenes being the same in case of dynamics uncertainty. Due to the same reason, in the Multi-Scene Two Lane Merge and the Hand-manipulation Object Reaching Datasets, there are additional bolded PSNR values for the pairs $(I_{t_1}, \tilde{I}_{t_4})$ and $(I_{t_4}, \tilde{I}_{t_1})$.

We demonstrate that the toggled or predicted images that correspond to the correct ground truth show a PSNR value around 29, indicating high fidelity 3D reconstruction and clear visual decodings as the output of CARFF.

4.3 Architecture Ablations

The results presented in Tab. 4 substantiate the benefits of our PC-VAE encoder architecture compared to other formulations. Specifically, a non-conditional VAE fails in SVM accuracy as it only reconstructs images and does not capture the underlying 3D structures. Vanilla PC-VAE and PC-VAE without freezing weights



(a) Approaching Intersection **(b)** Two Lane Merge

Fig. 5: Multi-Scene dataset accuracy and recall curves from learned latents. We test our framework across $n = 1$ and $n = 50$ samples from PC-VAE's latent distributions from ego-centric image input. Across the number of samples n, we achieve an ideal margin of belief state coverage generated under partial observation (recall), and the proportion of correct beliefs sampled under full observation (accuracy) for the MDN to learn. As we significantly increase the number of samples, the accuracy starts to decrease due to randomness in latent distribution resampling.

require careful fine-tuning of several hyper-parameters and don't generalize well to drastic camera movements. Our experiments show that our proposed model is capable of sustaining stochastic characteristics via latent representations in the presence of occlusion, while simultaneously ensuring precise reconstructions.

Encoder Architectures	Train PSNR	SVM Acc.	NV PSNR
PC-VAE	**26.30**	**75.20**	**25.24**
PC-VAE w/o CL	26.24	70.60	24.80
Vanilla PC-VAE	26.02	25.70	24.65
PC-VAE w/o Freezing	24.57	5.80	24.60
PC-VAE w/ MobileNet	17.14	19.70	17.16
Vanilla VAE	24.15	10.60	11.43

Table 4: PC-VAE ablations. CARFF's PC-VAE encoder outperforms other architectures in image reconstruction and pose-conditioning. We evaluate on: PC-VAE without Conv. Layer, PC-VAE with a vanilla encoder, PC-VAE without freezing ViT weights, PC-VAE replacing ViT with MobileNet, and non pose-conditional VAE.

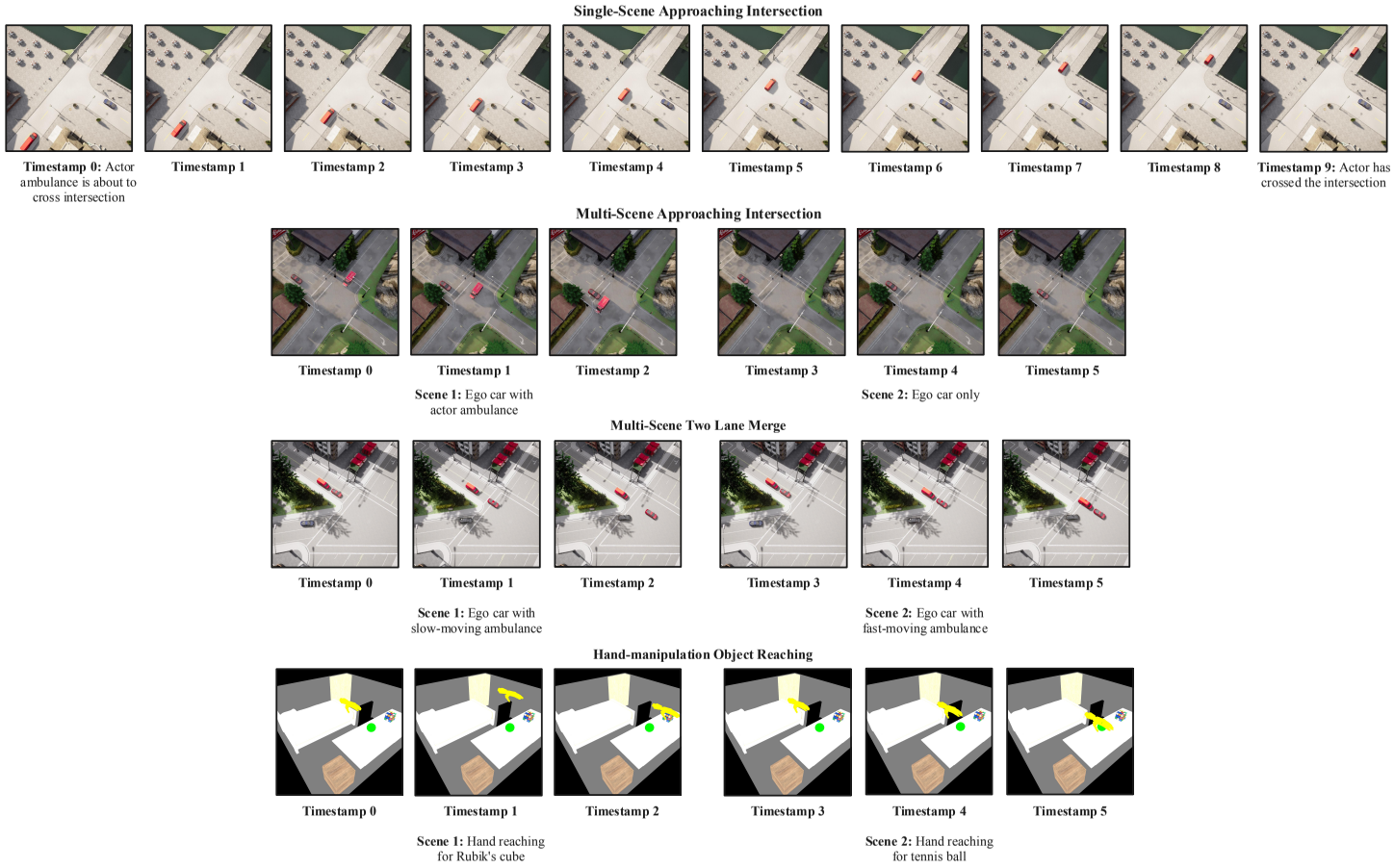


Fig. 6: Single-Scene Approaching Intersection, Multi-Scene Approaching Intersection, Multi-Scene Two Lane Merge, and Hand-manipulation Object Reaching Datasets. The actor and ego car configurations and the hand configurations for the timestamps and scenes of the three CARLA datasets, and the hand-manipulation dataset are visualized at a single camera pose. The colors of the cars for the Multi-Scene Approaching Intersection have been slightly modified for greater contrast and visual clarity in the paper.

Bibliography

[1] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Int. Conf. Learn. Represent. (2021)

[2] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Conf. on Robol Learning. pp. 1–16 (2017)

[3] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 1–15 (2022)

[4] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge (2015)

[5] Yang, G.: VUER: A 3D visualization and data collection environment for robot learning (2024), `https://github.com/vuer-ai/vuer`